SUBJECT NAME : MOBILE APPLICATION DEVELOPMENT

SUBJECT CODE : CCS51

SEMESTER : V

Unit-I

UNIT I: INTRODUCTION TO ANDROID PLATFORM

1. **Introduction to Mobile Application Development**
    - **1.1 Android introduction**
    - **1.2 What is Android?**
2. **Various platforms**
3. **Smart phones**
4. **Android platform**
    - **4.1. Common Problem: Abstraction**
    - **4.2. iPHONE Development: Security Model**
5. **Features**
6. **Architecture**
7. **Versions**
8. **ART (Android Runtime)**
9. **ADB (Android Debug Bridge)**
10. **Development environment/IDE:**
11. **Android studio and its working environment**
12. **Emulator setup**
13. **Application framework basics**
14. **XML representation and Android manifest file**
15. **Creating a simple application.**

1. What is mobile application development?

Mobile application development is the process to making software for smart phones anddigital assistants, most commonly for Android and iOS. The software can be preinstalled on the device, downloaded from a mobile app store or accessed through a mobile web browser.

The programming and markup languages used for this kind of Software developmentinclude Java, Swift, C# and HTML5.

1.1 Android introduction:

Android is a complete set of software for mobile devices such as tablet computers, notebooks, smartphones, electronic book readers, set-top boxes etc.

It contains a linux-based Operating System, middleware and key mobile applications.

It can be thought of as a mobile operating system. But it is not limited to mobile only. It is currentlyused in various devices such as mobiles, tablets, televisions etc.

1.2 What is Android

Before learning all topics of android, it is required to know what is android.

Android is a software package and linux based operating system for mobile devices such as tablet computers and smartphones.

It is developed by Google and later the OHA (Open Handset Alliance). Java language is mainly usedto write the android code even though other languages can be used.

The goal of android project is to create a successful real-world product that improves the mobile experience for end users.

2.VARIOUS PLARFORMS:

A Mobile OS **Manages Hardware** and **Software**. Like System OS ,Some OS platforms cover **software stack** while others may only include the lowerlevels. Application Development **platform such,, Symbian ,, Android..** etc., Provide **Programming ,, Native** & **Java Programming.**

Today's Phone as Internet browser, packages, game and Music/video Application platform are built over **lower level kernelOS**

Multiple Operation System: -

Difficult and Time-consuming task and API

Application programming interface (**API**) is a set of subroutinedefinitions, communication protocols, and tools for building software.

- **Symbian OS**
- **RIM Blackberry OS**
- **iPhone OS**
- **Windows Mobile**
- **Linux OS**
- **Palm webOS**
- **Bada OS**
- **MeeGo OS**
- **Android OS**

Symbian OS

- ➢Large share in market worldwide.
- ➢Major handset manufacturers ie., BenQ , Samsung..etc., ➢In 2009, Supported Multiple UserInterface
- ➢UIQ Technologies**(s/w)**
- ➢S60(Formerlly Series 60 user interfafce) – from Nokia,
- ➢MOAP(Mobile Oriented Applications Platform) – from DOCOMO
- ➢In 2009, 3 UIs Merged into single platform
- ➢Open Source
- ➢The upper layer – S60, UIQ & MOAP
- ➢Schedular, Memory Management & Device drivers
- ➢Special focus – N/W, File

RIM Blackberry OS

- ➢Designed for business.
- ➢Provides **Multitasking** & Input devices
- ➢**MIDP 1.0** (Mobile Information Device Profile) and **MIDP 2.0**
- ➢Published – Java on Embedded devices
- ➢**MIDP 2.0** Supported wireless activation
- ➢**WAP 1.2** (wireless Application Protocol) support – Third party application & Multimedia

iPhone OS

- ➢**iPhone uses an OS called iPhone OS**
- ➢**Derived from Mac OS X**
- ➢**Four Layers**
- ➢**Core OS layer**
- ➢**Core service layer**
- ➢**Media layer**
- ➢**Cocoa touch layer**
- ➢**500 megabytes – storage devices**
- ➢**Focused – online marketing**

Windows mobile

- ➢**Based on windows CE 5.2 kernel**
- ➢**Developed by window API**
- ➢**It used – desktop version,, native code, Visual C ++ .net, etc.,**
- ➢**2 improve , touch screen devices, mobile 6 standard**
- ➢**Finally, touch screen and physical keyboard configuration.**

Linux OS

➢**In china, used by Motorola..**

➢**Linux is used as a basis for a number of different platforms developed by several vendors**

- **ie., Andorid etc.,**
- **Mostly incompatible palm source**

Palm webOS

➢**Next generation OS.**

➢**WebOS written in HTML, AJAX & JavaScript ➢Handled with Webkit**

➢**Supported streaming video in RTSP(Real Time Streaming Protocol) ➢Can be accessed from the launcher screen**

Bada OS

➢**Kernel configurable architecture, allow use Linux kernel or RTOS**

➢**Preferred choice for smart phone and smaller memory footprint ➢Offers interactive mappinginside native applications , POI(Point of interest)**

➢**Developed in C++ with the Bada SDK**

➢**Eclipse based IDE**

➢**It can run java ME application**

➢**The first device – "Wave"**

➢**Wave is fully – " touch screen phone**

MeeGo OS

➢**2010, Nokia and Intel – unveiled MeeGo ➢Open – Sourced for all user.**

➢**Developed by Google Inc.**

➢**Open source , Linux – derived platform. ➢To develop app – flexible,**

3.SMART PHONES:

Nokia 9000 Series

3.1 WHAT IS A "SMARTPHONE"

☒**Semi-Smart: Phone that offers features beyond making calls**

- **E-mail**
- **Take pictures**
- **Plays mp3**

☒**Phone that runs a complete Operating System**

- **offers a standardized platform for development**
- **Able to execute arbitrary 3rd party applications**

Today

- **Cell phones in use today ~ 1.2 billion**
- **Smartphones account for 14% ~ 170 Million**

☒Projected 2012

- **Cell phones ~ 1.7 billion**
- **Smartphones 29% ~ 500 MillionWHY?**

☒Different Phones Different Uses

- **Phones for consumer or phone for business**
- **V-Cast vs. Palm**

☒Money

- **Hardware made money**
- **Tried to maintain control over content and services.**
- **Wanted to charge 3rd party developers for the privilege of using their platform.**

☒Digital signing

☒Distribution mechanisms.

4. iPHONE DEVELOPMENT

- **Objective-C**
- **Message based architecture**
- **Similar to Smalltalk**
- **No Java VM or other 3rd party plugins**
- **"An Application may not itself install or launch other executable code by any means, including without limitation through the use of a plug-in architecture, calling other frameworks, other APIs or otherwise. No interpreted code may be downloaded and used in an Application except for code that is interpreted and run by Apple's Published APIs and built-in interpreter(s)." –iPhone SDK EULA**

4.1 COMMON PROBLEM: ABSTRACTION

☒Interface / GUI

- **How does the developer create an interface**

☒Different interaction techniques

☒Graphical capabilities of the phone

4.2 iPHONE DEVELOPMENT: SECURITY MODEL

- **Originally all applications ran as root**
- **Not a whole lot better now**
- **All apps run as "mobile" user**
- **Survived this year's Pwn2Own**
- **Security based on delivery mechanism**
- **All applications must be delivered through theiTunes App Store**
- **Requires apple approval and testing**
- **$99 App Store**
- **$299 Enterprise**
- **Digitally signed by developer**

5. iPHONE DEVELOPMENT: FUTURE

- **iPhone OS 3.0**
- **In app purchases**
- **Accessory APIs**
- **Peer to Peer connectivity**
- **New Game Kit**
- **iPod library access**
- **Embedded maps**
- **Copy & Paste**
- **Video**

5.1 Features of Android

Android is an open source and Linux-based Operating System for mobile devices such as smartphonesand tablet computers. Android was developed by the Open Handset Alliance, led by Google, and other companies.

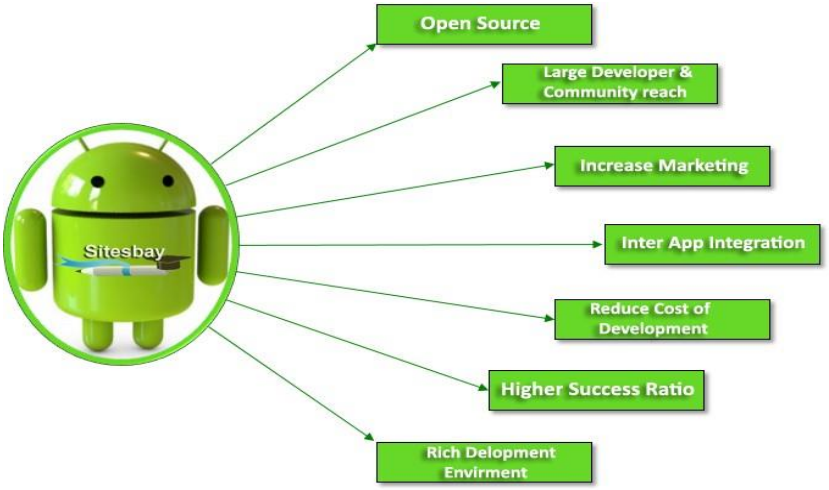


Fig: Features of Android

Android is a powerful operating system competing with Apple 4GS and supports great features. Few of them are listed below;

Simple and User-Friendly Interface: The main features of Android is it is very simple and easy to useand android have user-friendly.

Fast Loading Speed: Android application loading time is very fast.

Supporting the Multiple Languages: It support multiple language. **security**:

Android is design on Linux platform so it provide high security.

Beautiful UI: Android OS basic screen provides a beautiful and intuitive user interface.

Connectivity: GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC andWiMAX.

Storage: SQLite, a lightweight relational database, is used for data storage purposes.

Media support: H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, OggVorbis, WAV, JPEG, PNG, GIF, and BMP.

Messaging: It support SMS and MMS

Web browser:

Based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript enginesupporting HTML5 and CSS3.

Multi-touch:

Android has native support for multi-touch which was initially made available in handsets such as theHTC Hero.

Multi-tasking:

User can jump from one task to another and same time various application can run simultaneously.

Resizable widgets:

Widgets are resizable, so users can expand them to show more content or shrink them to save space.

Multi-Language:

Supports single direction and bi-directional text.

GCM:

Google Cloud Messaging (GCM) is a service that lets developers send short message data to theirusers on Android devices, without needing a proprietary sync solution.

Wi-Fi Direct:

A technology that lets apps discover and pair directly, over a high-bandwidth peer-to-peer connection.

Android Beam:

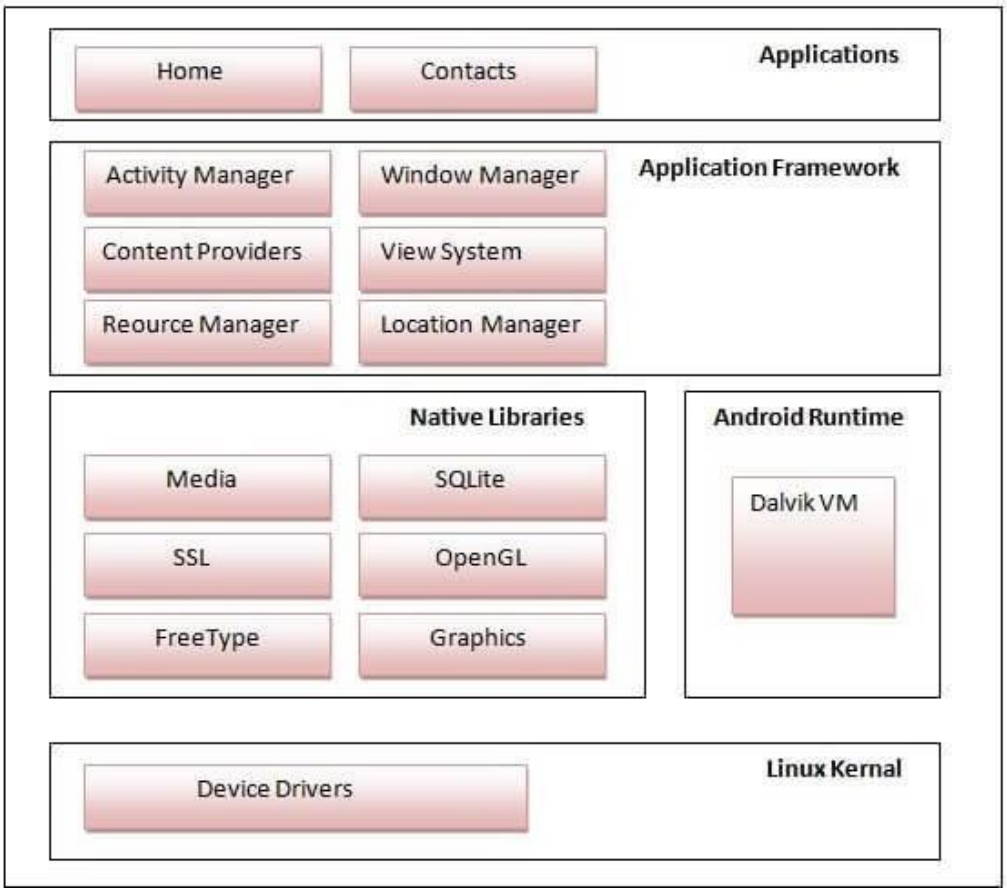
A popular NFC-based technology that lets users instantly share, just by touching two NFC-enabled phones together.

6. Android Architecture

Android architecture or Android software stack is categorized into five parts:

1. linux kernel
2. native libraries (middleware),
3. Android Runtime
4. Application Framework
5. Applications

Let's see the android architecture first.



1) Linux kernel

It is the heart of android architecture that exists at the root of android architecture. Linux kernel is responsible for device drivers, power management, memory management, device management and resource access.

2) Native Libraries

On the top of linux kernel, their are Native libraries such as WebKit, OpenGL, FreeType, SQLite, Media, C runtime library (libc) etc.

The WebKit library is responsible for browser support, SQLite is for database, FreeType for font support, Media for playing and recording audio and video formats.

3) Android Runtime

In android runtime, there are core libraries and DVM (Dalvik Virtual Machine) which is responsible to run android application. DVM is like JVM but it is optimized for mobile devices. It consumes less memory and provides fast performance.

4) Android Framework

On the top of Native libraries and android runtime, there is android framework. Android framework includes Android API's such as UI (User Interface), telephony, resources, locations, Content Providers (data) and package managers. It provides a lot of classes and interfaces for android application development.

---

5) Applications

On the top of android framework, there are applications. All applications such as home, contact, settings, games, browsers are using android framework that uses android runtime and libraries. Android runtime and native libraries are using linux kernal.

7.ANDROID VERSIONS:

Code name	Version number	Linux kernel version[1]	Initial release date
No Codename	1.0	2.1	September 23, 2008
Petit Four	1.1	2.6	February 9, 2009
Cupcake	1.5	2.6.27	April 27, 2009
Donut	1.6	2.6.29	September 15, 2009
Eclair	2.0 – 2.1	2.6.29	October 26, 2009
Froyo	2.2 – 2.2.3	2.6.32	May 20, 2010
Gingerbread	2.3 – 2.3.7	2.6.35	December 6, 2010
Honeycomb	3.0 – 3.2.6	2.6.36	February 22, 2011
Ice Cream Sandwich	4.0 – 4.0.4	3.0.1	October 18, 2011
Jelly Bean	4.1 – 4.3.1	3.0.31 - 3.4.39	July 9, 2012
KitKat	4.4 – 4.4.4	3.10	October 31, 2013
Lollipop	5.0 – 5.1.1	3.16	November 12, 2014
Marshmallow	6.0 – 6.0.1	3.18	October 5, 2015
Nougat	7.0 – 7.1.2	4.4	August 22, 2016
Oreo	8.0 – 8.1	4.10	August 21, 2017
Pie	9.0	4.4.107, 4.9.84, and 4.14.42	August 6, 2018
Android Q	10.0		September 3, 2019

8.Android Runtime

Android Runtime environment is an important part of Android rather than an internal part and it contains components like core libraries and the Dalvik virtual machine. The Android run time is the engine that powers our applications along with the libraries andit forms the basis for the application framework.

Dalvik Virtual Machine (DVM) is a register-based virtual machine like Java VirtualMachine (JVM).

It is specially designed and optimized for android to ensure that a device can run multiple instances efficiently.

It relies on the Linux kernel for threading and low-level memory management.The core libraries in android runtime will enable us to implement an android applications using standard JAVA programming language.

9.ANDROID DEBUG BRIDGE

The Android Debug Bridge (adb) utility permits us to interact with the Android Emulator directly fromthe command line or script.

navigate the filesystem on your Well, now you can with the adb! The adb works as a client/server TCP-based application.

While there are a couple of background processes that run on the development machine and the emulator to enable our functionality, the important thing to understand is that when we run adb, we get access to a running instance of the Android Emulator. Here are a couple of examples of using adb

adb devices

This command will return a list of available Android Emulators; for example, figure 2.8 shows adb locating two running emulator sessions.

Let's connect to the first Android Emulator session and see if our application is installed. We connect with the syntax adb shell. This is how we would connect if we had a single Android Emulator session active, but because there are two emulators running, we need to specify an identifier to connect to the appropriate session:

adb -d 1 shell

Figure 2.9 shows off the Android filesystem and demonstrates looking for a specific installed application, namely our Chapter2 sample application, which we'll be building in the next section.

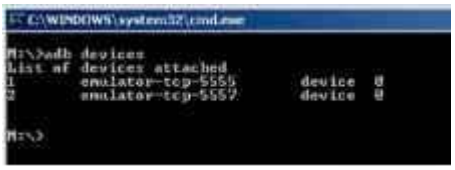
This capability can be very handy when we want to remove a specific file from the emulator's filesystem, kill a process, or generally interact with the operating environment of the Android Emulator.

If you download an application from the internet, for example, you can use the adb command to install an application. For example,

adb shell install someapplication.apk

Installs the application named someapplication to the Android Emulator. The file is copied to the /data/app directory and is accessible from the Android application launcher. Similarly, if you desire to remove an application, you can run adb to remove an application from the Android Emulator. For example, if you desire to remove the Chapter2.apk sample application from a running emulator's filesystem, you can execute the following command from a terminal or Windows command window:

Using the shell command, we can browse Android's filesystem.



The adb tool provides interaction at runtime with the Android Emulator.Using

the shell command, we can browse Android's filesystem.

adb shell rm /data/app/Chapter2.apk

Mastering the command-line tools in the Android SDK is certainly not a requirement of Android application development, but having an understanding of what is available and where to look for capabilities is a good skill to have in your toolbox. If you need assistance with either the aapt or adb command, simply enter the command at the terminal, and a fairly verbose usage/help page is displayed. Additional information on the tools may be found in the Android SDK documentation.

The Android filesystem is a Linux filesystem. While the adb shell command does not provide a very rich shell programming environment as is found on a desktop Linux or Mac OS X system, basic commands such as ls, ps, kill, and rm are available. If you are new to Linux, you may benefit from learning some very basic shell commands.

One other tool you will want to make sure you are familiar with is telnet. Telnet allows you to connect to a remote system with a character-based UI. In this case, the remote system you connect to is the Android Emulator's console. You can accomplish this with the following command:

telnetlocalhost 5554

In this case, localhost represents your local development computer where the Android Emulator has been started because the Android Emulator relies on your computer's loopback IP address of 127.0.0.1. Why port 5554? Recall when we employed adb to find running emulator instances that the output of that command included a name with a number at the end. The first Android Emulator can generally be found at IP port 5555. No matter which port number the Android Emulator is using, the Android Emulator's console may be found at a port number equaling 1 less. For example, if the Android Emulator is running and listed at port 5555, the console is at port 5554.

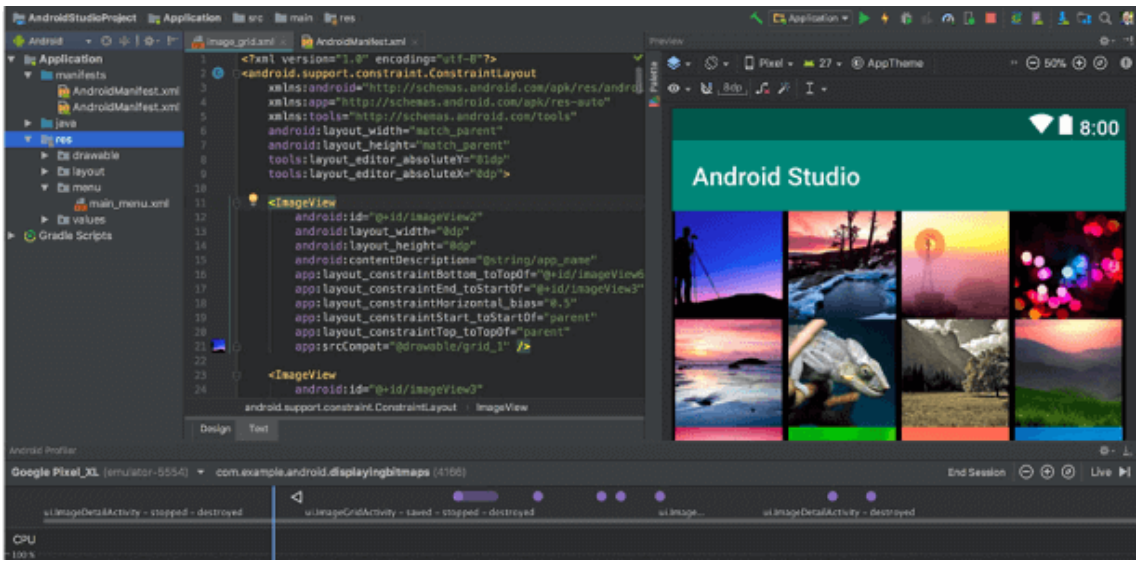
Using a telnet connection to the emulator provides a command-line means for configuring the emulator while it is running and testing telephony features such as calls and text messages.

10.ANDROID STUDIO AND ITS WORKING ENVIRONMENT

Android Studio is the official Integrated Development Environment (IDE) for android application development. Android Studio provides more features that enhance our productivity while building Android apps.

Android Studio was announced on 16th May 2013 at the Google I/O conference as an official IDE for Android app development. It started its early access preview from version 0.1 in May 2013. The first stable built version was released in December 2014, starts from version 1.0.

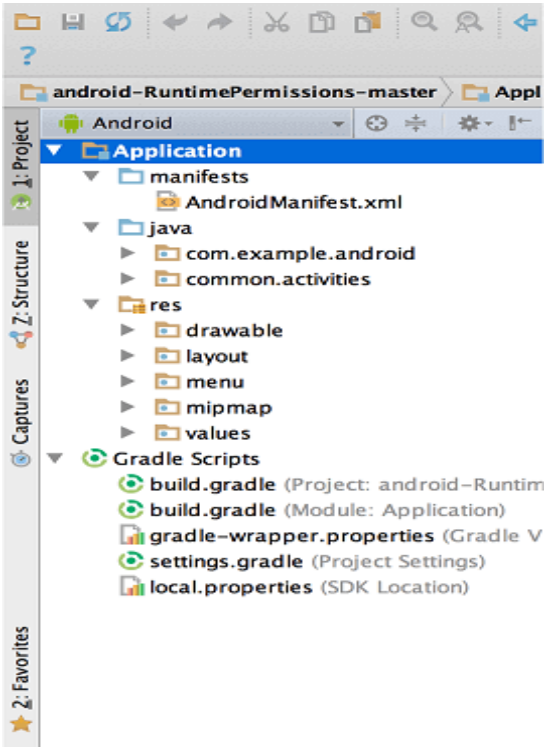
Since 7th May 2019, Kotlin is Google's preferred language for Android application development. Besides this, other programming languages are supported by Android Studio.



11.Android Studio Project Structure

The Android Studio project contains one or more modules with resource files and source code files. These include different types of modules-

- Android app modules
- Library modules
- Google App Engine modules



By default, Android Studio displays our project files in the Android project view, as shown in the above image. This view is formed by modules to provide quick access to our project's key source files.

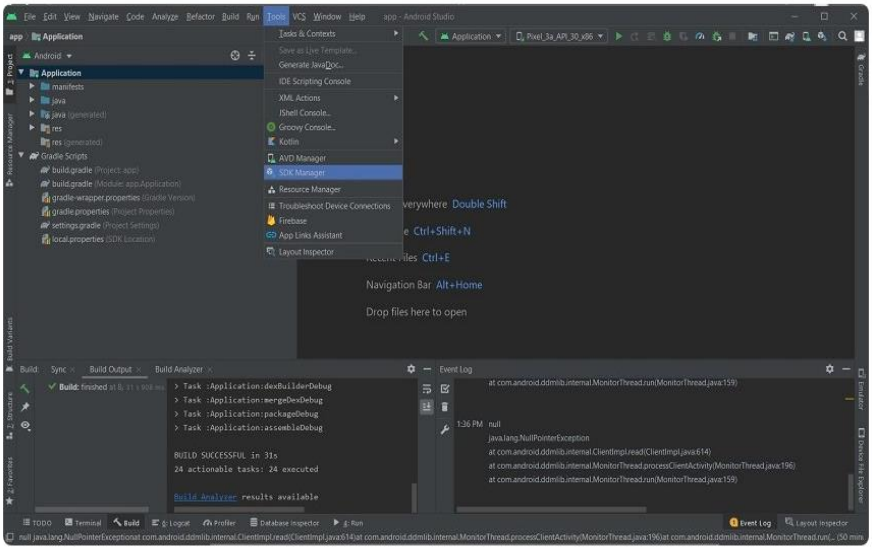
These build files are visible to the top-level under Gradle Scripts. And the app module contains the following folders:

- **manifests:** It contains the AndroidManifest.xml file.
- **java:** It contains the source code of Java files, including the JUnit test code.
- **res:** It contains all non-code resources, UI strings, XML layouts, and bitmap images.

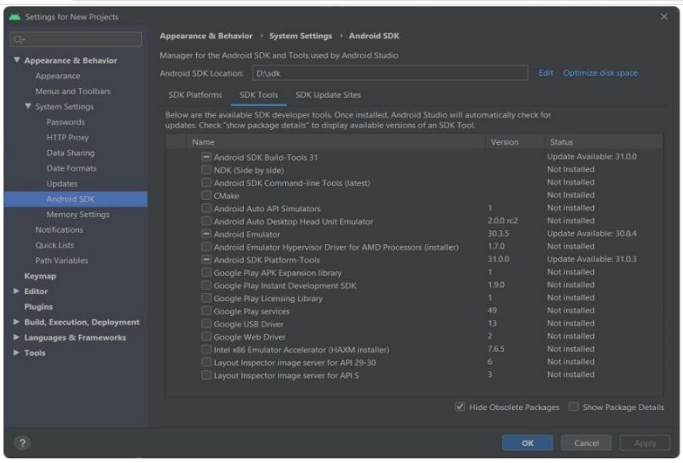
We will see the actual file structure of the project by selecting the **Project** from the **Projectdropdown**.

12.EMULATOR SETUP

To use the Android Emulator, you will need to download it first. You can download it from the SDK manager located in the tools.



Select Tools > SDK Manager. Then, from the settings window, choose Android Emulator. Click Apply, and Android Studio will download the emulator for you.



An Android emulator cannot run on your computer without a virtualization tool for hardware acceleration. Fortunately, Windows 10 comes preloaded with Hyper-V, which is one of the best virtualization tools in the market.

You can discover Hyper-V's status on your computer by running the systeminfo.exe command in your command prompt.

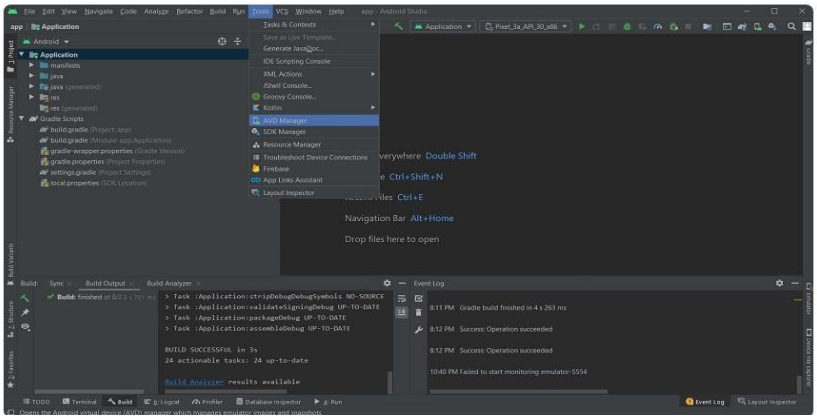
```
                           [02]: fe80::2d1f:b2b1:559b:c018
Hyper-V Requirements:      VM Monitor Mode Extensions: Yes
                           Virtualization Enabled In Firmware: Yes
                           Second Level Address Translation: Yes
                           Data Execution Prevention Available: Yes
```

If Hyper-V is not installed on your computer, then Android Studio allows you to install HAXM through its SDK manager. HAXM is another hardware acceleration tool. HAXM is downloaded and installed through the same window you used to download the Android emulator.

If you install HAXM with Hyper-V already working on your computer, it is bound to cause problems.

Wait for the installations to complete, and then restart your computer and Android Studio.

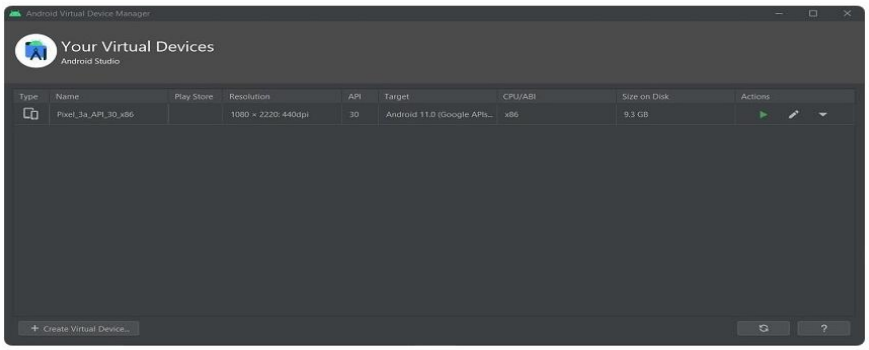
Now, select **Tools > AVD Manager** (for Android Virtual Device), and explore the virtual devices.



Downloading a Virtual Device

There'll be a default virtual device in your AVD manager when you download the Android emulator. However, you can download the device of your choice that has a different screen size or other specifications to test your application for your specific purpose or on a larger scale.

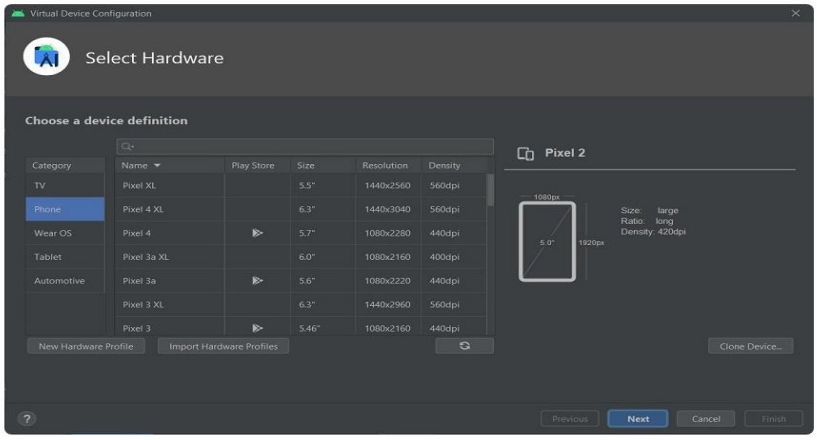
When you open the AVD manager, look at the bottom left corner and you'll see a button for creating a new virtual device.



Each virtual device comprises some hardware and software configurations. The system image of a virtual device represents its software components.

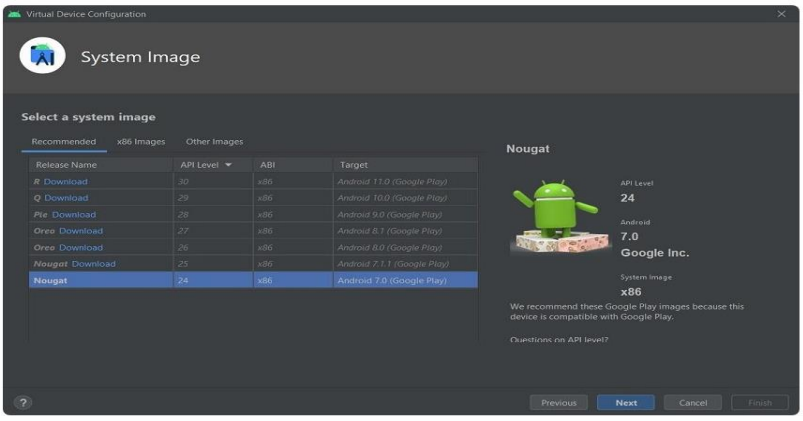
When you set out to create a new virtual device, you'll have to determine its hardware first. This is where you select settings like the screen size, screen resolution, screen pixel density, and RAM. You can define the hardware from scratch or use the default hardware options offered by Android Studio.

In the hardware selection menu, you'll also get to see a Play Store icon beside some of the hardware choices. The system image of these devices will have a Play Store integrated into their interfaces.

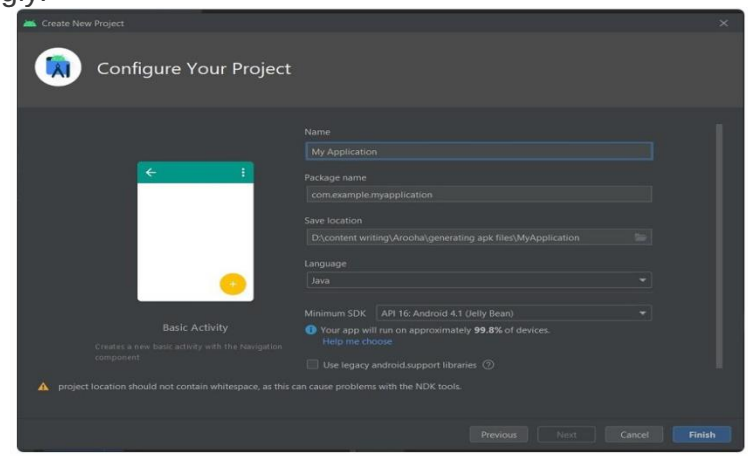


Once you're done selecting the hardware configurations, click next and you'll be taken to the system image menu.

A system image comprises the Android version, its API level, and ABI. These selections have to be in accordance with your project.



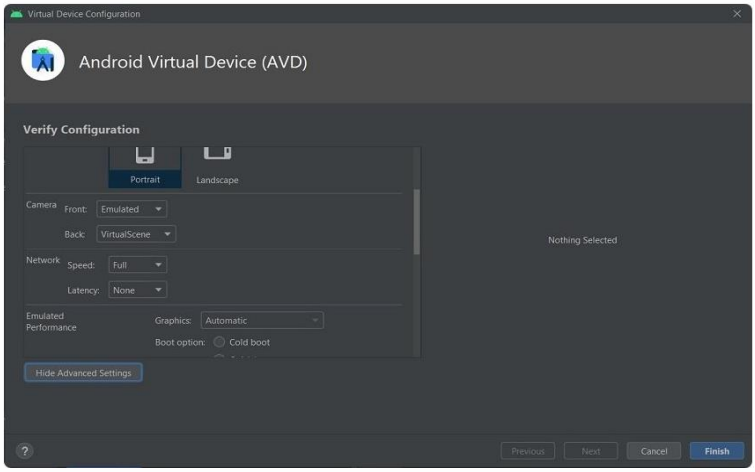
Android Studio asks you about the minimum SDK requirements when creating a new project. These requirements include the API level and the Android version. Recall this information and select the system image accordingly.



Lastly, the ABI tells you about the processor architecture that the system image supports. Assuming you are a beginning Android developer, I would suggest you opt for the ABI that supports x86 architecture.

Select the most relevant system image, and click **Next** to download it if it isn't downloaded already.

Further customizations take place on the last screen for creating a new virtual device. If you look at the bottom, you'll see a button for the advanced settings. Here you can explore the camera settings, network settings, performance, and storage of your virtual device as per your application's needs.

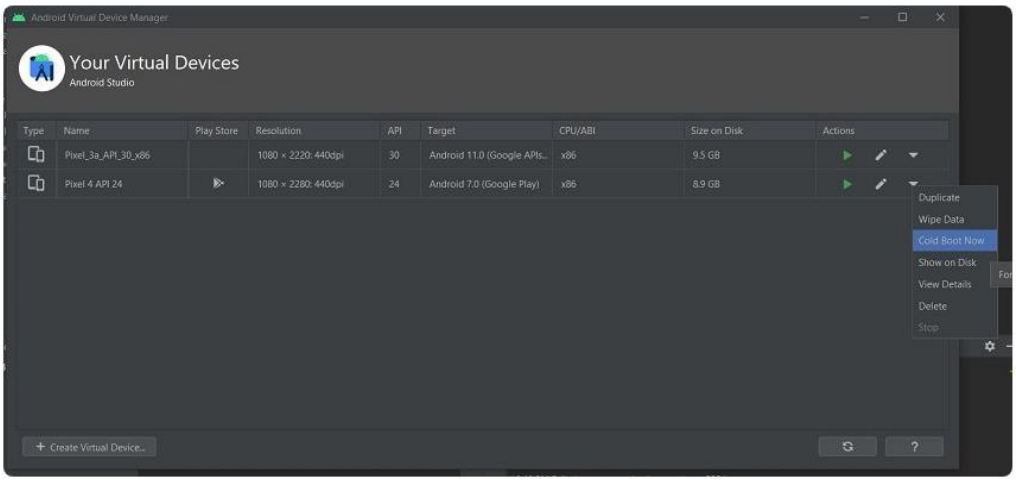


The performance section of the settings gives you three options:

1. **Cold boot**: It will start your device as if you're turning it on for the first time.
2. **Quick boot**: It will remember the last state of your device, and the next time you run the Android emulator it will show you the same screen.
3. **Snapshot**: This refers to the state of the Android emulator. You get to save the state yourself, and it'll kick off from the same page the next time you run the emulator.

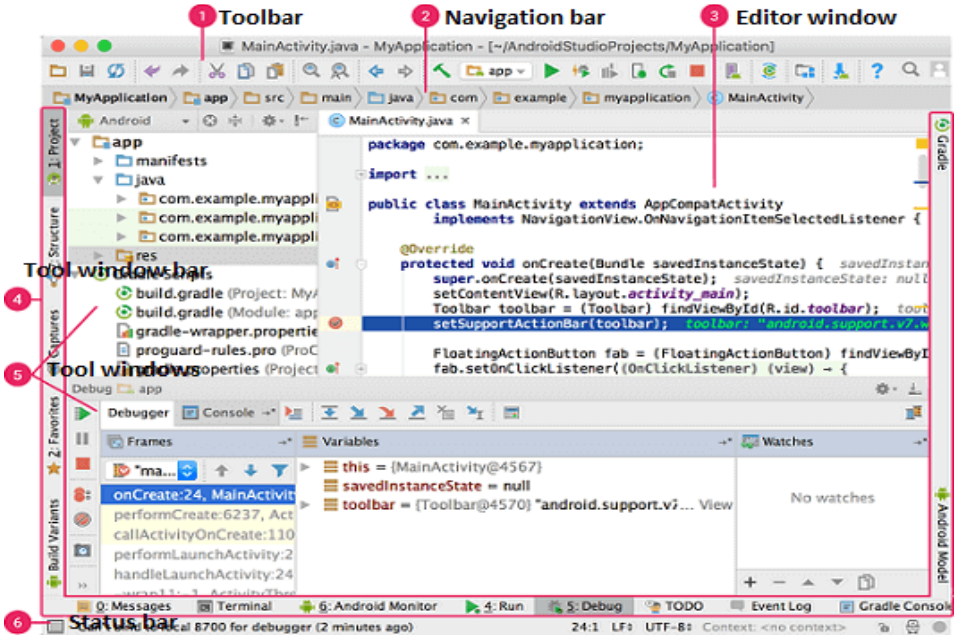
With all the settings in place, you've successfully created a new virtual device, and it should now show up in your AVD manager.

If you want to see how it looks, open the drop-down menu from the **Actions** column and select **Cold Boot Now**. The emulator will show up on your screen.



13.Android Studio User Interface

The Android Studio main window contains the several logical areas which are shown in the below figure:



1. The **toolbar** provides us a wide range of actions, which includes running apps and launching Android tools.
2. The **navigation bar** helps in navigating our project and open files for editing. It gives a compact view of structure visible in the Project window.
3. The **editor window** is a space where we can create and modify our code. On the basis of the current file type, the editor can change. While viewing a layout file, the editor displays the Layout Editor.
4. The **tool window bar** runs around the outside the IDE window and contains buttons that allow as to expand and collapse individual tool windows.

5. The **tool windows** provide us access specific tasks like search, project management, version control, and more. We can able expand and collapse them.
6. The **status bar** displays the status of our project and IDE itself, as well as any messages or warnings.

Gradle build system

Gradle build used as the foundation of the build system in Android Studio. It uses more Android- specific capabilities provided by the Android plugin for Gradle. This build system runs independently from the command line and integrated tool from the Android Studio menu. We can use build features for the following purpose:

- Configure, customize, and extend the build process.
- We can create multiple APKs from our app, with different features using the same project and modules.
- Reuse resource and code across source sets.

14.XML REPRESENTATION AND ANDROIDMANIFEST.XML FILE IN ANDROID

The **AndroidManifest.xml file** *contains information of your package*, including components of the application such as activities, services, broadcast receivers, content providers etc.

It performs some other tasks also:

- It is **responsible to protect the application** to access any protected parts by providing the permissions.
- It also **declares the android api** that the application is going to use.
- It **lists the instrumentation classes**. The instrumentation classes provides profiling and other informations. These informations are removed just before the application is published etc.

This is the required xml file for all the android application and located inside the root directory. A simple AndroidManifest.xml file looks like this:

```
1. <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2.     package="com.javatpoint.hello"
3.     android:versionCode="1"
4.     android:versionName="1.0" >
5.
6.     <uses-sdk
7.         android:minSdkVersion="8"
8.         android:targetSdkVersion="15" />
```

```
9.
10.    <application
11.        android:icon="@drawable/ic_launcher"
12.        android:label="@string/app_name"
13.        android:theme="@style/AppTheme" >
14.        <activity
15.            android:name=".MainActivity"
16.            android:label="@string/title_activity_main" >
17.            <intent-filter>
18.                <action android:name="android.intent.action.MAIN" />
19.19.
20.                <category android:name="android.intent.category.LAUNCHER" />
21.            </intent-filter>
22.        </activity>
23.    </application>
24.24.
25. </manifest>
```

Elements of the AndroidManifest.xml file

The elements used in the above xml file are described below.

<manifest>

manifest is the root element of the AndroidManifest.xml file. It has **package** attribute that describes the package name of the activity class.

<application>

application is the subelement of the manifest. It includes the namespace declaration. This element contains several subelements that declares the application component such as activity etc.

The commonly used attributes are of this element are **icon**, **label**, **theme** etc.

android:icon represents the icon for all the android application components.

android:label works as the default label for all the application components.

android:theme represents a common theme for all the android activities.

<activity>

activity is the subelement of application and represents an activity that must be defined in the AndroidManifest.xml file. It has many attributes such as label, name, theme, launchMode etc.

android:label represents a label i.e. displayed on the screen.

android:name represents a name for the activity class. It is required attribute.

<intent-filter>

intent-filter is the sub-element of activity that describes the type of intent to which activity, service or broadcast receiver can respond to.

<action>

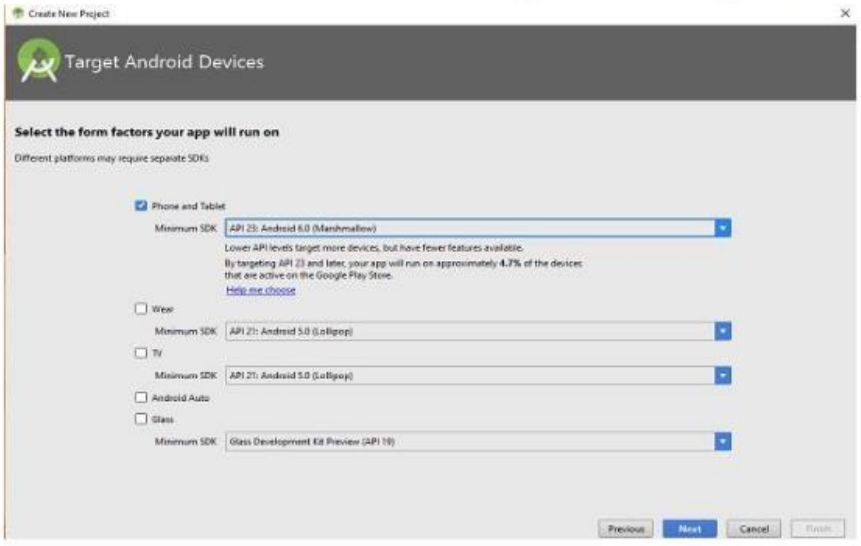
It adds an action for the intent-filter. The intent-filter must have at least one action element.

<category>

It adds a category name to an intent-filter.

15. CREATING A SIMPLE APPLICATION

Create Android Application....

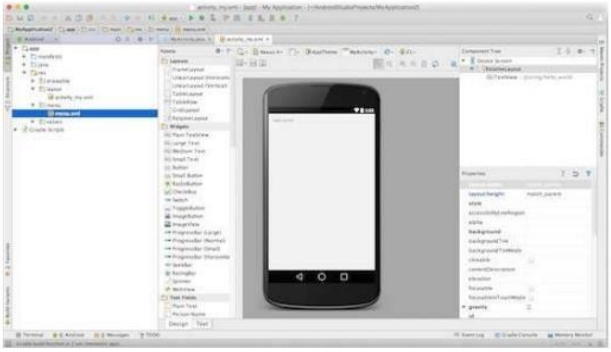


✓ After entered application name, it going to be called select the form factors your application runs on, here need to specify Minimum SDK.

Create Android Application....

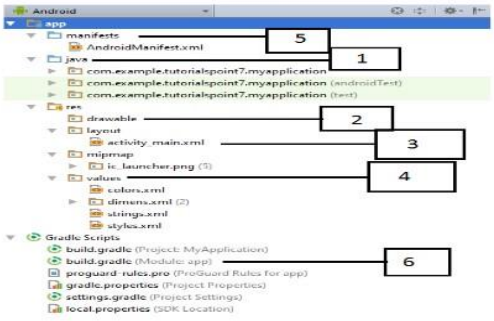
✓ The next level of installation should contain selecting the activity to mobile, it specifies the default layout for Applications.

Create Android Application....



✓ At the final stage it going to be open development tool to write the application code.

Create Android Application....



✓ Before you run your app, you should be aware of a few directories and files in the Android project

Create Android Application....

Sr.No.	Folder, File & Description
1	**Java** This contains the .java source files for your project. By default, it includes an MainActivity.java source file having an activity class that runs when your app is launched using the app icon.
2	**res/drawable-hdpi** This is a directory for drawable objects that are designed for high-density screens.
3	**res/layout** This is a directory for files that define your app's user interface.
4	**res/values** This is a directory for other various XML files that contain a collection of resources, such as strings and colours definitions.
5	**AndroidManifest.xml** This is the manifest file which describes the fundamental characteristics of the app and defines each of its components.
6	**Build.gradle** This is an auto generated file which contains compileSdkVersion, buildToolsVersion, applicationId, minSdkVersion, targetSdkVersion, versionCode and versionName

MARUDHAR KESARI JAIN COLLEGE FOR WOMEN, VANIYAMBADI.

SUBJECT: MOBILE APPLICATION DEVELOPMENT

CODE: CCS51

UNIT II: ANDROID UI DESIGN

Objective: To understand the basic concepts of user interface related to app development.

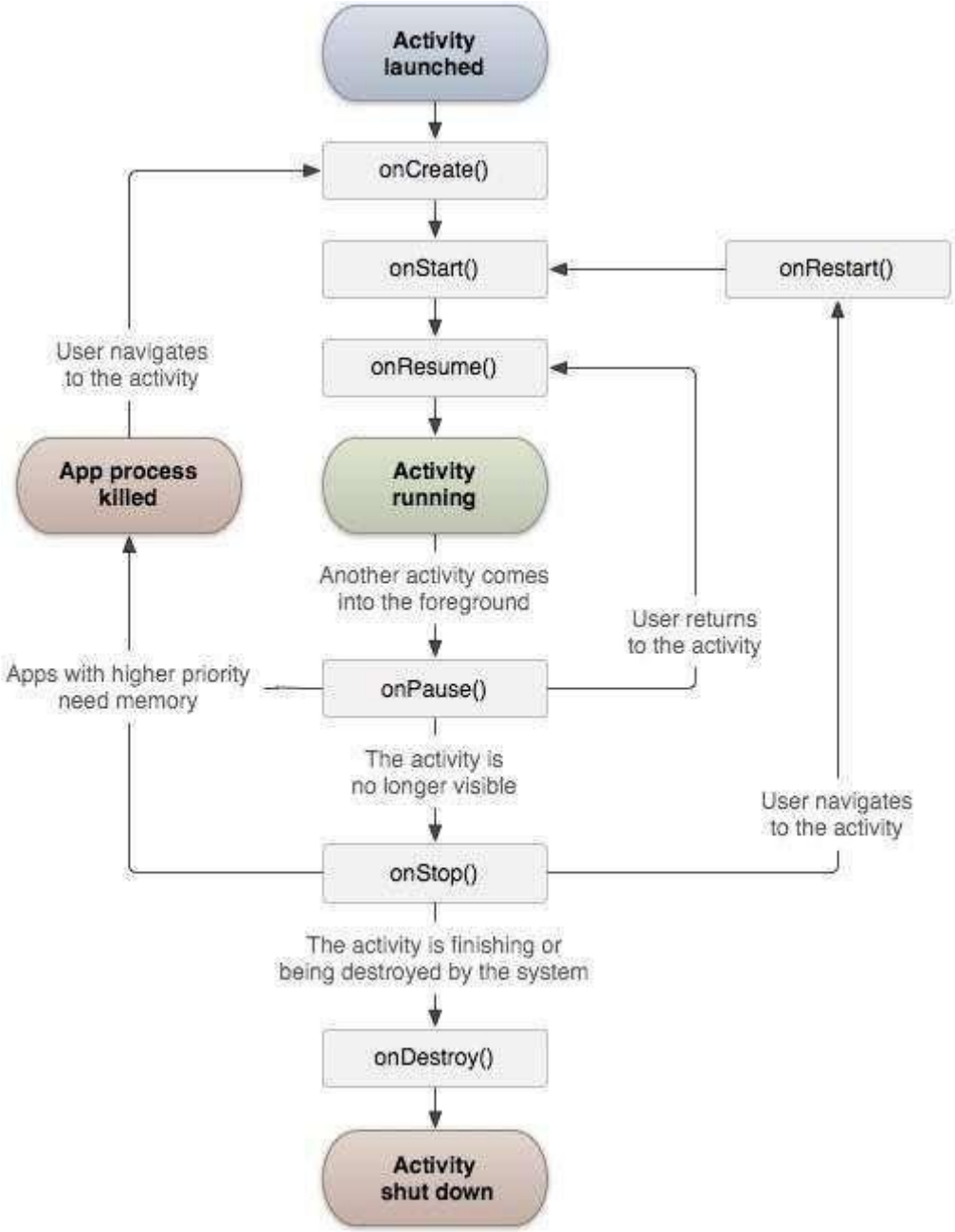
GUI for Android: activities lifecycle–Android v7 support library –Intent: Intent object – Intent filters– Adding categories – Linking activities – User Interface design components–Basic Views – Picker Views – List View –Specialized Fragment– Gallery and Image View – Image Switcher – Grid View, Options Menu – Context Menu– Clock View –Web view–Recycler View.

1.The Life Cycle of an Activity:

An activity represents a single screen with a user interface just like window or frame of Java.Android activity is the subclass of ContextThemeWrapper class.

If you have worked with C, C++ or Java programming language then you must have seen that your program starts from **main()** function. Very similar way, Android system initiates its program with in an **Activity** starting with a call on *onCreate()* callback method. There is a sequence of callback methods that start up an activity and a sequence of callback methods that tear down an activity as shown in the below Activity life cycle diagram: (*image courtesy : android.com*)

The Activity class defines the following call backs i.e. events. You don't need to implement all



the callbacks methods. However, it's important that you understand each one and implement those that ensure your app behaves the way users expect.

Sr.No	Callback & Description
1	**onCreate()** This is the first callback and called when the activity is first created.
2	**onStart()** This callback is called when the activity becomes visible to the user.
3	**onResume()** This is called when the user starts interacting with the application.
4	**onPause()** The paused activity does not receive user input and cannot execute any code and called when the current activity is being paused and the previous activity is being resumed.
5	**onStop()** This callback is called when the activity is no longer visible.
6	**onDestroy()** This callback is called before the activity is destroyed by the system.
7	**onRestart()** This callback is called when the activity restarts after stopping it.

Example

This example will take you through simple steps to show Android application activity life cycle. Follow the following steps to modify the Android application we created in *Hello World Example* chapter

Step	Description
1	You will use Android studio to create an Android application and name it as *HelloWorld* under a package *com.example.helloworld* as explained in the *Hello World Example* chapter.
2	Modify main activity file *MainActivity.java* as explained below. Keep rest of the files unchanged.
3	Run the application to launch Android emulator and verify the result of the changes done in the application.

Following is the content of the modified main activity file **src/com.example.helloworld/MainActivity.java**. This file includes each of the fundamental life cycle methods. The **Log.d()** method has been used to generate log messages −

```
package com.example.helloworld;


import android.os.Bundle;
import android.app.Activity;
import android.util.Log;


publicclassMainActivityextendsActivity{
String msg ="Android : ";


/** Called when the activity is first created. */
@Override
publicvoidonCreate(BundlesavedInstanceState){
super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_main);
Log.d(msg,"The onCreate() event");

}


/** Called when the activity is about to become visible.
*/@Override

protectedvoidonStart(){
super.onStart();

Log.d(msg,"The onStart() event");

}


/** Called when the activity has become visible.
*/@Override

protectedvoidonResume(){
super.onResume();

Log.d(msg,"The onResume() event");

}
```

```
/** Called when another activity is taking focus. */ @Override

protectedvoid onPause(){ super.onPause();

Log.d(msg,"The onPause() event");

}


/** Called when the activity is no longer visible. */ @Override

protectedvoid onStop(){ super.onStop();

Log.d(msg,"The onStop() event");

}


/** Called just before the activity is destroyed. */ @Override

publicvoid onDestroy(){ super.onDestroy();

Log.d(msg,"The onDestroy() event");

}

}
```

2.The Android Support Library

The Android SDK tools include a number libraries collectively called the *Android Support Library*. This package of libraries provides several features that are not built into the standard Android framework, and provides backward compatibility for older devices. Include any of these libraries in your app to incorporate that library's functionality.

2.1 v7 support library

The v7 support library includes both compatibility libraries and additional features.

The v7 support library includes all the v4 support libraries, so you don't have to add those separately. A dependency on the v7 support library is included in every new Android Studio project, and new activities in your project extend from AppCompatActivity.

The v7 support libraries include these specific components:

- v7 appcompat library: Adds support for the app bar UI design pattern and support for Material Design UI implementations.
- v7 cardview library: Provides the CardView class, a view that lets you show information inside cards.
- v7 gridlayout library: Includes the GridLayout class, which allows you to arrange UI elements using a grid of rectangular cells
- v7 mediarouter library: Provides MediaRouter and related media classes that support Google Cast.
- v7 palette library: Implements the Palette class, which lets you extract prominent colors from an image.
- v7 recyclerview library: Provides the RecyclerView class, a view for efficiently displaying large data sets by providing a limited window of data items.
- v7 preference library: Provides APIs to support preference objects in app settings.

2.1.1 Setting up and using the Android Support Library

The Android Support Library package is part of the Android SDK, and available to download in the Android SDK manager. To set up your project to use any of the support libraries, use these steps:

1. Download the support library with the Android SDK manager, or verify that the support libraries are already available.
2. Find the library dependency statement for the support library you're interested in.
3. Add that dependency statement to the dependencies section of your build.gradle (Module: app) file.

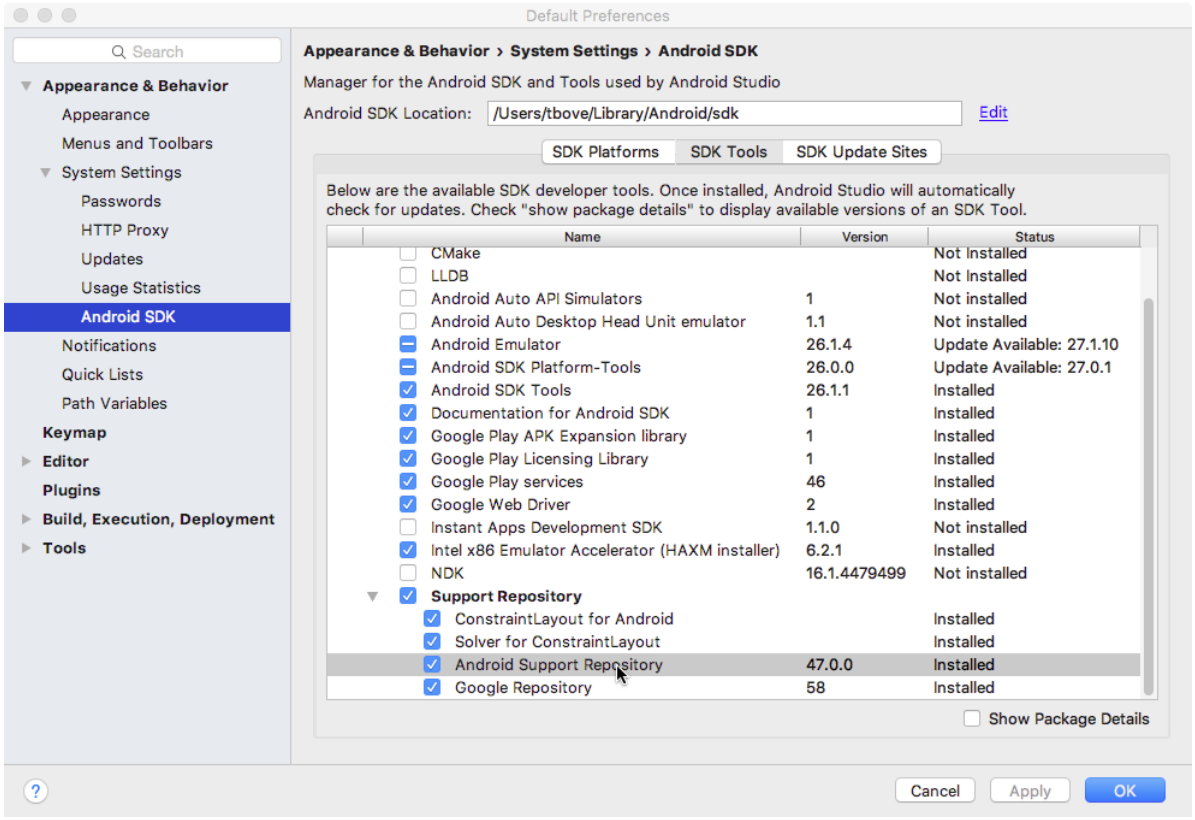
2.1.2 Download the support library

In Android Studio, you'll use the Android Support Repository—the repository in the SDK manager for all support libraries—to get access to the library from within your project.

You may already have the Android support libraries downloaded and installed with Android Studio. To verify that you have the support libraries available, follow these steps:

1. In Android Studio, select **Tools > Android > SDK Manager**, or click the SDK Manager  icon. The SDK Manager preference pane appears.

2. Click the **SDK Tools** tab and expand **Support Repository**, as shown in the figure below.



3. Look for **Android Support Repository** in the list. If **Installed** appears in the Status column, you're all set. Click **Cancel**.

If **Not installed** or **Update Available** appears, click the checkbox next to **Android Support Repository**. A download icon should appear next to the checkbox. Click **OK.**

4. Click **OK** again, and then **Finish** when the support repository has been installed.

3.Android Intent

Android Intent is the *message* that is passed between components such as activities, content providers, broadcast receivers, services etc.

It is generally used with startActivity() method to invoke activity, broadcast receivers etc.

The **dictionary meaning** of intent is *intention or purpose*. So, it can be described as the intention to do action.

The LabeledIntent is the subclass of android.content.Intent class.

Android intents are mainly used to:

- Start the service
- Launch an activity
- Display a web page
- Display a list of contacts
- Broadcast a message
- Dial a phone call etc.

3.1 Types of Android Intents

There are two types of intents in android: implicit and explicit.

1) Implicit Intent

Implicit Intent doesn't specifiy the component. In such case, intent provides information of available components provided by the system that is to be invoked.

For example, you may write the following code to view the webpage.

```
1. Intent intent=new Intent(Intent.ACTION_VIEW);
2. intent.setData(Uri.parse("http://www.javatpoint.com"));
3. startActivity(intent);
```

2) Explicit Intent

Explicit Intent specifies the component. In such case, intent provides the external class to be invoked.

```
1. Intent i = new Intent(getApplicationContext(), ActivityTwo.class);
2. startActivity(i);
```

4.Intent Objects

An Intent object is a bundle of information which is used by the component that receives the intent as well as information used by the Android system.

An Intent object can contain the following components based on what it is communicating or going to perform −

4.1 Action

This is mandatory part of the Intent object and is a string naming the action to be performed — or, in the case of broadcast intents, the action that took place and is being reported. The action largely determines how the rest of the intent object is structured . The Intent class defines a number of action constants corresponding to different intents. Here is a list of Android Intent Standard Actions

The action in an Intent object can be set by the setAction() method and read by getAction().

4.2 Data

Adds a data specification to an intent filter. The specification can be just a data type (the mimeType attribute), just a URI, or both a data type and a URI. A URI is specified by separate attributes for each of its parts −

These attributes that specify the URL format are optional, but also mutually dependent −

- If a scheme is not specified for the intent filter, all the other URI attributes are ignored.
- If a host is not specified for the filter, the port attribute and all the path attributes are ignored.

The setData() method specifies data only as a URI, setType() specifies it only as a MIME type, and setDataAndType() specifies it as both a URI and a MIME type. The URI is read by getData() and the type by getType().

Some examples of action/data pairs are −

Sr.No.	Action/Data Pair & Description
1	**ACTION_VIEW content://contacts/people/1** Display information about the person whose identifier is "1".
2	**ACTION_DIAL content://contacts/people/1** Display the phone dialer with the person filled in.

3	**ACTION_VIEW tel:123** Display the phone dialer with the given number filled in.
4	**ACTION_DIAL tel:123** Display the phone dialer with the given number filled in.
5	**ACTION_EDIT content://contacts/people/1** Edit information about the person whose identifier is "1".
6	**ACTION_VIEW content://contacts/people/** Display a list of people, which the user can browse through.
7	**ACTION_SET_WALLPAPER** Show settings for choosing wallpaper
8	**ACTION_SYNC** It going to be synchronous the data,Constant Value is **android.intent.action.SYNC**
9	**ACTION_SYSTEM_TUTORIAL** It will start the platform-defined tutorial(Default tutorial or start up tutorial)
10	**ACTION_TIMEZONE_CHANGED** It intimates when time zone has changed
11	**ACTION_UNINSTALL_PACKAGE** It is used to run default uninstaller

4.3 Category

The category is an optional part of Intent object and it's a string containing additional information about the kind of component that should handle the intent. The addCategory() method places a category in an Intent object, removeCategory() deletes a category previously added, and getCategories() gets the set of all categories currently in the object. Here is a list of Android Intent Standard Categories.

You can check detail on Intent Filters in below section to understand how do we use categories to choose appropriate activity corresponding to an Intent.

Extras

This will be in key-value pairs for additional information that should be delivered to the component handling the intent. The extras can be set and read using the putExtras() and getExtras() methods respectively. Here is a list of Android Intent Standard Extra Data

Flags

These flags are optional part of Intent object and instruct the Android system how to launch an activity, and how to treat it after it's launched etc.

Sr.No	Flags & Description
1	**FLAG_ACTIVITY_CLEAR_TASK** If set in an Intent passed to Context.startActivity(), this flag will cause any existing task that would be associated with the activity to be cleared before the activity is started. That is, the activity becomes the new root of an otherwise empty task, and any old activities are finished. This can only be used in conjunction with FLAG_ACTIVITY_NEW_TASK.
2	**FLAG_ACTIVITY_CLEAR_TOP** If set, and the activity being launched is already running in the current task, then instead of launching a new instance of that activity, all of the other activities on top of it will be closed and this Intent will be delivered to the (now on top) old activity as a new Intent.
3	**FLAG_ACTIVITY_NEW_TASK** This flag is generally used by activities that want to present a "launcher" style behavior: they give the user a list of separate things that can be done, which otherwise run

	completely independently of the activity launching them.

Component Name

This optional field is an android **ComponentName** object representing either Activity, Service or BroadcastReceiver class. If it is set, the Intent object is delivered to an instance of the designated class otherwise Android uses other information in the Intent object to locate a suitable target.

The component name is set by setComponent(), setClass(), or setClassName() and read by getComponent().

5.Intent Filter

Intent Filter are the components which decide the behavior of

an intent. The Intent is about the navigation of one activity to another, thatcan be

achieve by declaring intent filter. We can declare an Intent Filter for an Activity in

manifest file.

5.1 Intent Filter in Manifest File

There are following three elements in an intent filter:

1. Action

2. Data

3. Category

Note: Every intent filter must contain action element in it. Data and categoryelement is

optional for it.

1. Action:

It represent an activities action, what an activity is going to do. It isdeclared

with the name attribute as given below

An Intent Filter element must contain one or more action element. Action is a string that specifies the action to perform. You can declare your own action as given below. But we usually use action constants defined by Intent class.

Elements in Intent Filter:

<action android:name = "string" />

<intent-filter . . . >
<action android:name="com.example.project.SHOW_CURRENT" />
<action android:name="com.example.project.SHOW_RECENT" />
<action android:name="com.example.project.SHOW_PENDING" />
. . .
</intent-filter>

There are few common actions for starting an activity like ACTION_VIEW

ACTION_VIEW: This is used in an Intent with startActivity(). This helps whenyou redirect to see any website, photos in gallery app or an address to viewin a map app.

There are more actions similar to above like, ACTION_SEND, ACTION_MAIN, ACTION_WEB_SEARCH and many more.

2. <category>

This attribute of Intent filter dictates the behavior or nature of an Intent.

There is a string which contains some additional information about theintent which will be handled by a component.

The syntax of category is as follows:

BROWSABLE – Browsable category, activity allows itself to be opened withweb browser to open the reference link provided in data.

LAUNCHER – Launcher category puts an activity on the top of stack, whenever application will start, the activity containing this category will be opened first.

<category android:name="string" />

<intent-filter . . . >

<category android:name="android.intent.category.DEFAULT" />

<category android:name="android.intent.category.BROWSABLE" />

. . .

</intent-filter>

Linking Activities

Android calling one activity from another activity example

activity_main.xml

<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout_width="match_parent"

android:layout_height="match_parent"

```
tools:context=".MainActivity">
<TextView
android:id="@+id/editText"
android:layout_width="match_parent"
android:layout_height="wrap_content" android:text="First
Activity" android:textAlignment="center"
android:textSize="28sp"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.0"

app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />
<Button
android:id="@+id/btn1" android:text="Go to
News Screen"
android:onClick="newsScreen"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/editText" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Mainactivity.java

```
package com.example.intent2;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class MainActivity extends AppCompatActivity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}
public void newsScreen(View view) {
Intent i = new Intent(getApplicationContext(), MainActivity2.class);
startActivity(i);
}
}
```

activity_main2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"

android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
tools:context=".MainActivity2">

<TextView
android:id="@+id/editText"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Second Activity"
android:textAlignment="center" android:textSize="28sp"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.0"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />

<Button
android:id="@+id/btn2"
android:text="Go to Home Screen" android:onClick="homeScreen"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/editText" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

MainActivity2.java

```
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class MainActivity2 extends AppCompatActivity {

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main2);
}

public void homeScreen(View view) {
Intent i = new Intent(getApplicationContext(), MainActivity.class);
startActivity(i);
}
```

6.User Interface design Components

Android View

The View is a base class for all UI components in android. For example, the EditText class is used to accept the input from users in android apps, which is a sub class of View.

Following are the some of common View subclasses which will be used in android applications.

• TextView

• EditText

• Button

• CheckBox

• RadioButton

• ImageButton

• Progress Bar

• Spinner

Like these we have many View subclasses available in android.

Android ViewGroup

The ViewGroup is a subclass of View and it will act as a base class for layouts and layouts parameters. The ViewGroupwill provide an invisible containers to hold

other Views or ViewGroups and to define the layout properties.

For example, Linear Layout is the ViewGroup that contains a UI controls like button, textview, etc. and other layouts also.

Following are the commonly used ViewGroup subclasses in android applications.

• Linear Layout

• Relative Layout

• Table Layout

• Frame Layout

• Web View

• List View

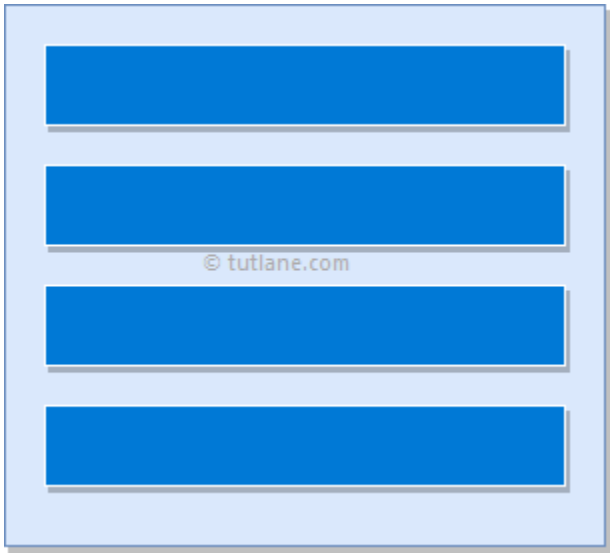
• Grid View

7.Android ListView with Examples

n android, **ListView** is a **ViewGroup** that is used to display the list of scrollable of items in multiple rows and the list items are automatically inserted to the list using an **adapter**.

Generally, the adapter pulls data from sources such as an array or database and converts each item into a result view and that's placed into the list.

Following is the pictorial representation of listview in android applications.



Android ListView Example

Following is the example of creating a **ListView** using **arrayadapter** in android application.

Create a new android application using android studio and give names as **ListView**. In case if you are not aware of creating an app in android studio check this article Android Hello World App.

Now open an **activity_main.xml** file from **\res\layout** path and write the code like as shown below

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical">
   <ListView
      android:id="@+id/userlist"
      android:layout_width="match_parent"
      android:layout_height="wrap_content" >
   </ListView>
</LinearLayout>
```

Once we are done with creation of layout, now we will bind data to our **ListView** using **ArrayAdapter**, for that open main activity file **MainActivity.java** from **\java\com.tutlane.listview** path and write the code like as shown below.

MainActivity.java

```
package com.tutlane.listview;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class MainActivity extends AppCompatActivity {
   private ListView mListView;
   private ArrayAdapter aAdapter;
   private String[] users = { "Suresh Dasari", "Rohini Alavala", "Trishika Dasari", "Praveen
Alavala", "Madav Sai", "Hamsika Yemineni"};
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      mListView = (ListView) findViewById(R.id.userlist);
      aAdapter = new ArrayAdapter(this, android.R.layout.simple_list_item_1, users);
      mListView.setAdapter(aAdapter);
   }
}
```

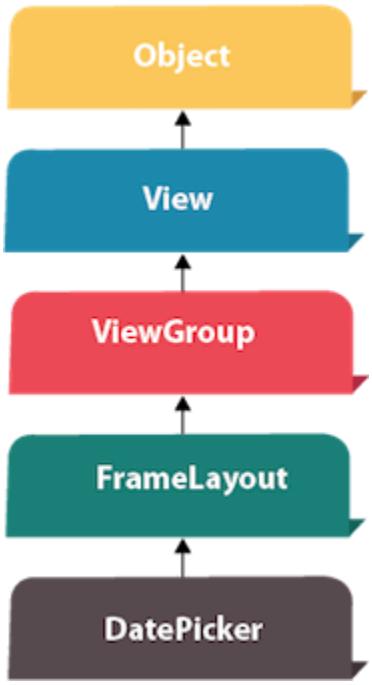
If you observe above code, we are binding static array (**users**) details to **ListView** using **ArrayAdapter** and calling our layout using **setContentView** method in the form of **R.layout.layout_file_name**. Here our xml file name is **activity_main.xml** so we used file name **activity_main**.

Generally, during the launch of our activity, onCreate() callback method will be called by the android framework to get the required layout for an activity.

Output of Android ListView Example

When we run above example using android virtual device (AVD) we will get a result like as shown below.

8.Android DatePicker



Android DatePicker is a widget to select date. It allows you to select date by day, month and year. Like DatePicker, android also provides TimePicker to select time.

The android.widget.DatePicker is the subclass of FrameLayout class.

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     xmlns:app="http://schemas.android.com/apk/res-auto"
4.     xmlns:tools="http://schemas.android.com/tools"
5.     android:layout_width="match_parent"
6.     android:layout_height="match_parent"
7.     tools:context="example.javatpoint.com.datepicker.MainActivity">
8.
9.     <TextView
10.        android:id="@+id/textView1"
```

```
11.     android:layout_width="wrap_content"
12.     android:layout_height="wrap_content"
13.     android:layout_above="@+id/button1"
14.     android:layout_alignParentLeft="true"
15.     android:layout_alignParentStart="true"
16.     android:layout_marginBottom="102dp"
17.     android:layout_marginLeft="30dp"
18.     android:layout_marginStart="30dp"
19.     android:text="" />
20.
21. <Button
22.     android:id="@+id/button1"
23.     android:layout_width="wrap_content"
24.     android:layout_height="wrap_content"
25.     android:layout_alignParentBottom="true"
26.     android:layout_centerHorizontal="true"
27.     android:layout_marginBottom="20dp"
28.     android:text="Change Date" />
29.
30. <DatePicker
31.     android:id="@+id/datePicker"
32.     android:layout_width="wrap_content"
33.     android:layout_height="wrap_content"
34.     android:layout_above="@+id/textView1"
35.     android:layout_centerHorizontal="true"
```

```
36.         android:layout_marginBottom="36dp" />
37.
38. </RelativeLayout>
```

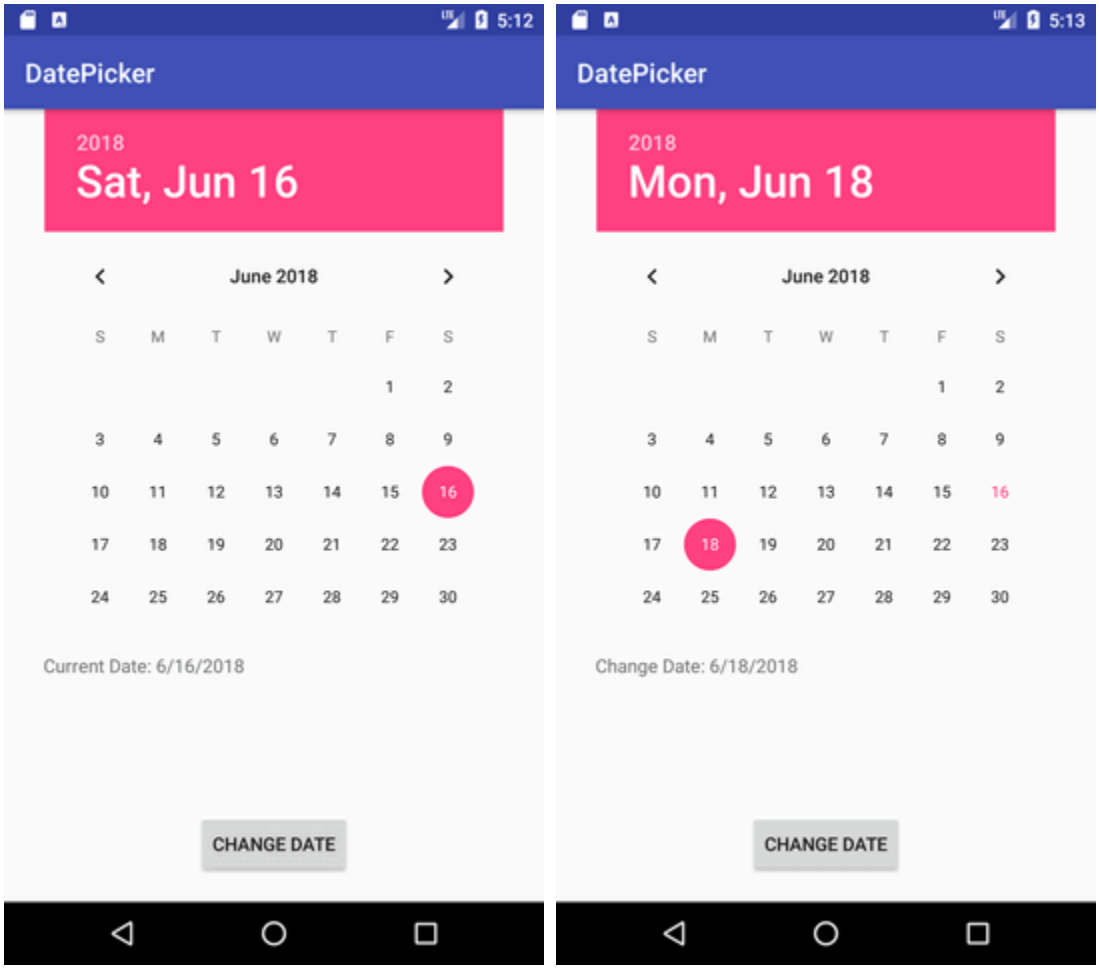
Activity class

File: MainActivity.java

```
1.  package example.javatpoint.com.datepicker;
2.
3.  import android.support.v7.app.AppCompatActivity;
4.  import android.os.Bundle;
5.  import android.view.View;
6.  import android.widget.Button;
7.  import android.widget.DatePicker;
8.  import android.widget.TextView;
9.
10. public class MainActivity extends AppCompatActivity {
11.     DatePicker picker;
12.     Button displayDate;
13.     TextView textview1;
14.     @Override
15.     protected void onCreate(Bundle savedInstanceState) {
16.         super.onCreate(savedInstanceState);
17.         setContentView(R.layout.activity_main);
```

```
18.         textview1=(TextView)findViewById(R.id.textView1);
19.         picker=(DatePicker)findViewById(R.id.datePicker);
20.         displayDate=(Button)findViewById(R.id.button1);
21.
22.         textview1.setText("Current Date: "+getCurrentDate());
23.
24.         displayDate.setOnClickListener(new View.OnClickListener(){
25.             @Override
26.             public void onClick(View view) {
27.
28.                 textview1.setText("Change Date: "+getCurrentDate());
29.             }
30.
31.         });
32.
33.     }
34.     public String getCurrentDate(){
35.         StringBuilder builder=new StringBuilder();;
36.         builder.append((picker.getMonth() + 1)+"/");//month is 0 based
37.         builder.append(picker.getDayOfMonth()+"/");
38.         builder.append(picker.getYear());
39.         return builder.toString();
40.     }
41. }
```

Output:



9.Android Gallery View Example

Pacific Regmi July 19, 2018

A Gallery is an android widget that helps us to display items (images) in horizontal scrolling list and currently selected item at the center. In this android gallery view tutorial, you will learn to display images horizontally and when you click any image, the currently selected image will be displayed in large size using ImageView and also show toast message when you select new item (image).

In xml layout file, you have to add a Gallery widget and an ImageView widget and then give different id to them. Following is the complete example of android gallery view.

Android Gallery Example: How to use Gallery View in Android

XML Layout File

res/layout/gallery_view_example.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="16dp"
    android:orientation="vertical">
    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Android Gallery View Example"
        android:textSize="22sp" />
    <Gallery
        android:id="@+id/gallery"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="#a3a1a1" />
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="fill_parent"
        android:layout_height="200dp"
```

```
        android:layout_marginTop="16dp"
        android:src="@drawable/cat" />
    <TextView
        android:id="@+id/textView5"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:autoLink="web"
        android:gravity="bottom|center"
        android:text="ViralAndroid.com"
        android:textSize="24sp" />
</LinearLayout>
```

Java Activity File

src/GalleryViewExample.java

```
// Android Gallery View Example
package com.viralandroid.androidtutorial;
import android.content.Context;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageView;
```

```
import android.widget.Toast;

public class GalleryVIewExample extends AppCompatActivity {

    Gallery galleryView;

    ImageView imgView;

    //images from drawable

    private int[] imageResource = {

            R.drawable.cat, R.drawable.viral_android_icon, R.drawable.image1,
R.drawable.viral_android_icon, R.drawable.cat, R.drawable.viral_android_icon,
R.drawable.image1

    };

    @Override

    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.gallery_view_example);

        imgView = (ImageView) findViewById(R.id.imageView);

        galleryView = (Gallery) findViewById(R.id.gallery);

        imgView.setImageResource(imageResource[0]);

        galleryView.setAdapter(new myImageAdapter(this));

        //gallery image onclick event

        galleryView.setOnItemClickListener(new AdapterView.OnItemClickListener() {

            public void onItemClick(AdapterView parent, View v, int i, long id) {

                imgView.setImageResource(imageResource[i]);

                int imagePosition = i + 1;

                Toast.makeText(getApplicationContext(), "You have selected image = " +
imagePosition, Toast.LENGTH_LONG).show();

            }
```

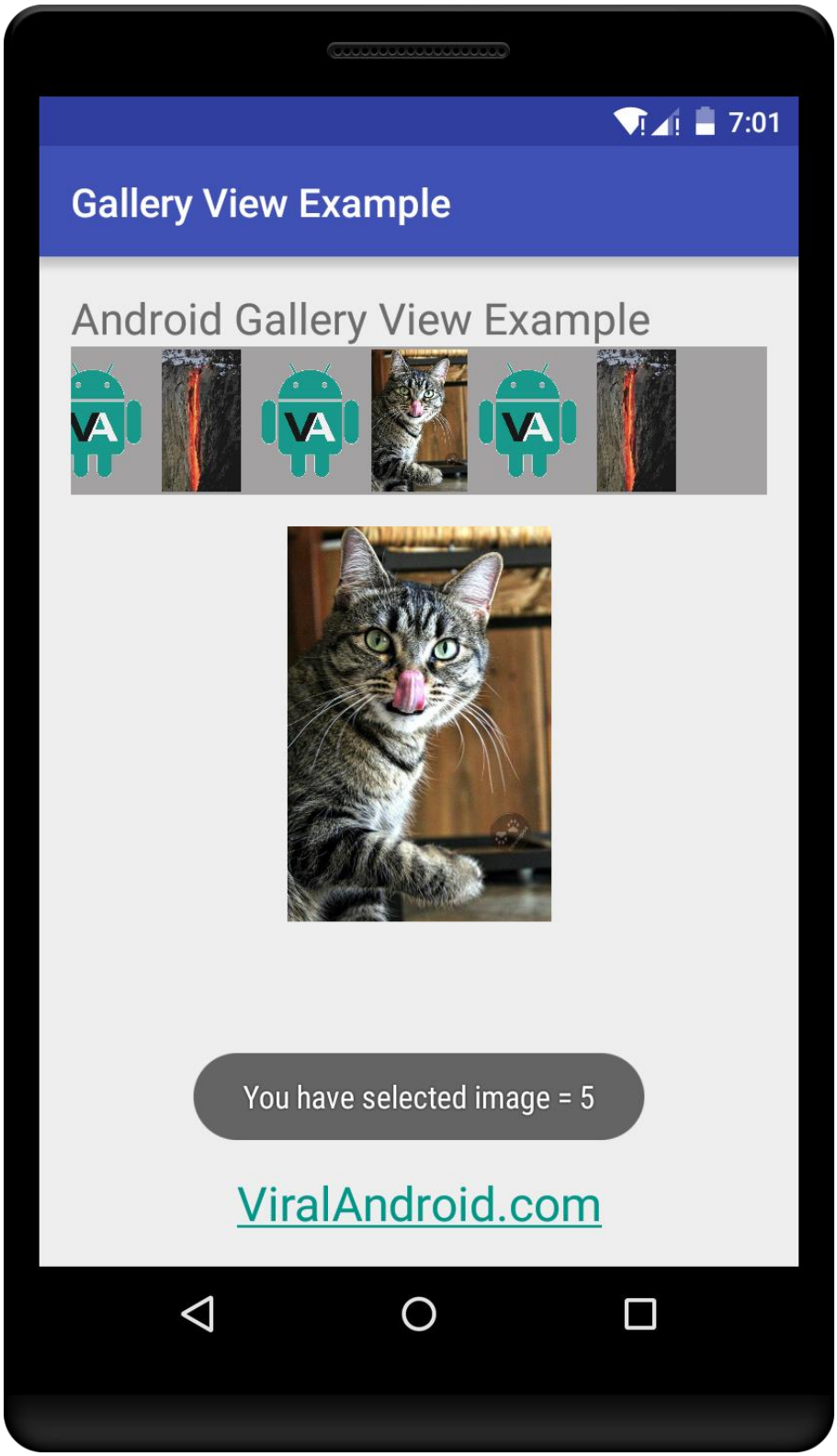
```
    });
}
public class myImageAdapter extends BaseAdapter {
    private Context mcontext;
    public View getView(int position, View convertView, ViewGroup parent) {
        ImageView mgalleryView = new ImageView(mcontext);
        mgalleryView.setImageResource(imageResource[position]);
        mgalleryView.setLayoutParams(new Gallery.LayoutParams(150, 150));
        mgalleryView.setScaleType(ImageView.ScaleType.CENTER_INSIDE);
        //imgView.setImageResource(R.drawable.image_border);
        mgalleryView.setPadding(3, 3, 3, 3);
        return mgalleryView;
    }
    public myImageAdapter(Context context) {
        mcontext = context;
    }
    public int getCount() {
        return imageResource.length;
    }
    public Object getItem(int position) {
        return position;
    }
    public long getItemId(int position) {
        return position;
    }
```

```
    }
}
```

Strings.xml File

res/values/strings.xml

```
<resources>
    <string name="app_name">Gallery View Example</string>
</resources>
```


7:01
Gallery View Example
Android Gallery View Example
You have selected image = 5
ViralAndroid.com

10.Android Image Switcher

Android image switcher provides an animation over images to transition from one image to another. In order to use image switcher, we need to implement **ImageSwitcher** component in .xml file.

The *setFactory()* method of ImageSwitcher provide implementation of *ViewFactory* interface. ViewFactory interface implements its unimplemented method and returns an ImageView.

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.
com/apk/res/android"
3.     xmlns:app="http://schemas.android.com/apk/res-auto"
4.     xmlns:tools="http://schemas.android.com/tools"
5.     android:layout_width="match_parent"
6.     android:layout_height="match_parent"
7.     android:fitsSystemWindows="true"
8.     tools:context="com.example.test.imageswitcher.MainActivity">
9.
10.    <android.support.design.widget.AppBarLayout
11.        android:layout_width="match_parent"
12.        android:layout_height="wrap_content"
13.        android:theme="@style/AppTheme.AppBarOverlay">
14.
15.        <android.support.v7.widget.Toolbar
16.            android:id="@+id/toolbar"
17.            android:layout_width="match_parent"
18.            android:layout_height="?attr/actionBarSize"
19.            android:background="?attr/colorPrimary"
20.            app:popupTheme="@style/AppTheme.PopupOverlay" />
21.
22.    </android.support.design.widget.AppBarLayout>
23.    <include layout="@layout/content_main" />
24.
25. </android.support.design.widget.CoordinatorLayout>
```

content_main.xml

File: content_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="com.example.test.imageswitcher.MainActivity"
    tools:showIn="@layout/activity_main">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Image Switcher Example"

        android:id="@+id/textView"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />

    <ImageSwitcher
        android:id="@+id/imageSwitcher"
        android:layout_width="match_parent"
        android:layout_height="250dp"
        android:layout_marginBottom="28dp"
        android:layout_marginTop="40dp" />

    <Button
```

```
32.     android:layout_width="wrap_content"
33.     android:layout_height="wrap_content"
34.     android:text="Next"
35.     android:id="@+id/button"
36.     android:layout_marginBottom="47dp"
37.     android:layout_alignParentBottom="true"
38.     android:layout_centerHorizontal="true" />
39. </RelativeLayout>
```

Activity class

File: MainActivity.java

```
1.  package com.example.test.imageswitcher;
2.
3.  import android.os.Bundle;
4.  import android.support.v7.app.AppCompatActivity;
5.  import android.support.v7.widget.Toolbar;
6.  import android.view.View;
7.  import android.widget.Button;
8.  import android.widget.ImageSwitcher;
9.  import android.widget.ImageView;
10.import android.widget.ViewSwitcher;
11.
12.import android.app.ActionBar;
13.import android.view.animation.Animation;
14.import android.view.animation.AnimationUtils;
15.
```

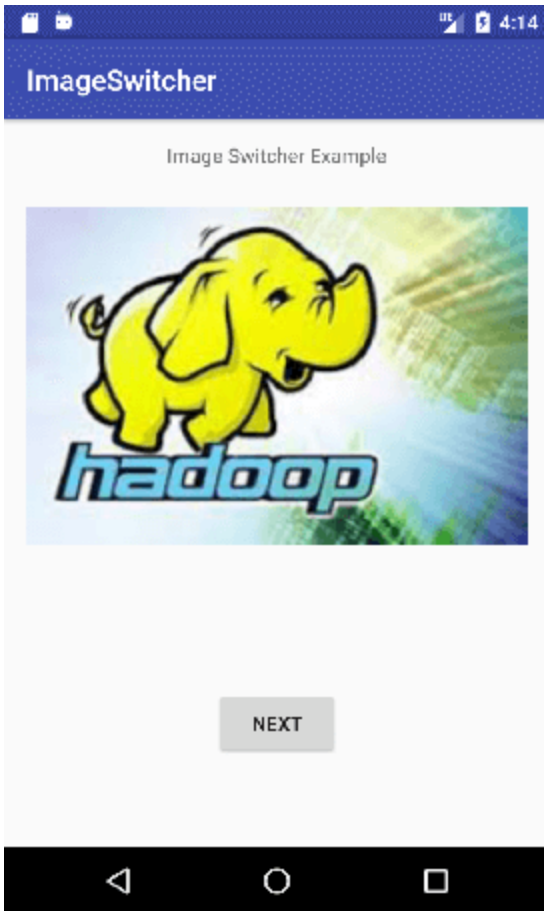
```
16.
17.public class MainActivity extends AppCompatActivity {
18.    ImageSwitcher imageSwitcher;
19.    Button nextButton;
20.
21.    int imageSwitcherImages[] =
22.     {R.drawable.cpp, R.drawable.c_sarp, R.drawable.jsp, R.drawable.m
    ysql, R.drawable.hadoop};
23.
24.    int switcherImageLength = imageSwitcherImages.length;
25.    int counter = -1;

26.    @Override
27.    protected void onCreate(Bundle savedInstanceState) {
28.        super.onCreate(savedInstanceState);
29.        setContentView(R.layout.activity_main);
30.        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
31.        setSupportActionBar(toolbar);
32.
33.        imageSwitcher = (ImageSwitcher) findViewById(R.id.imageSwitch
    er);
34.        nextButton = (Button) findViewById(R.id.button);
35.        imageSwitcher.setFactory(new ViewSwitcher.ViewFactory() {
```

```
36.         @Override
37.         public View makeView() {
38.             ImageView switcherImageView = new ImageView(getApplicat
    ionContext());
39.             switcherImageView.setLayoutParams(new ImageSwitcher.Lay
    outParams(
40.                     ActionBar.LayoutParams.FILL_PARENT, ActionBar.La
    youtParams.FILL_PARENT
41.             ));
42.             switcherImageView.setScaleType(ImageView.ScaleType.FIT_
    CENTER);
43.             switcherImageView.setImageResource(R.drawable.hadoop);
44.             //switcherImageView.setMaxHeight(100);
45.             return switcherImageView;
46.         }
47.       });
48.
49.       Animation aniOut = AnimationUtils.loadAnimation(this, android.R
    .anim.slide_out_right);
50.       Animation aniIn = AnimationUtils.loadAnimation(this, android.R.a
    nim.slide_in_left);
51.
52.      imageSwitcher.setOutAnimation(aniOut);
53.      imageSwitcher.setInAnimation(aniIn);
54.
```

```
55.        nextButton.setOnClickListener(new View.OnClickListener() {
56.            @Override
57.            public void onClick(View v) {
58.                counter++;
59.                if (counter == switcherImageLength){
60.                    counter = 0;
61.                    imageSwitcher.setImageResource(imageSwitcherImages[co
    unter]);
62.                }
63.                else{
64.                    imageSwitcher.setImageResource(imageSwitcherImages[co
    unter]);
65.                }
66.            }
67.        });
68.    }
69.
70.}
```

Output


4:14
ImageSwitcher
Image Switcher Example
hadoop
NEXT

11.GridView in Android

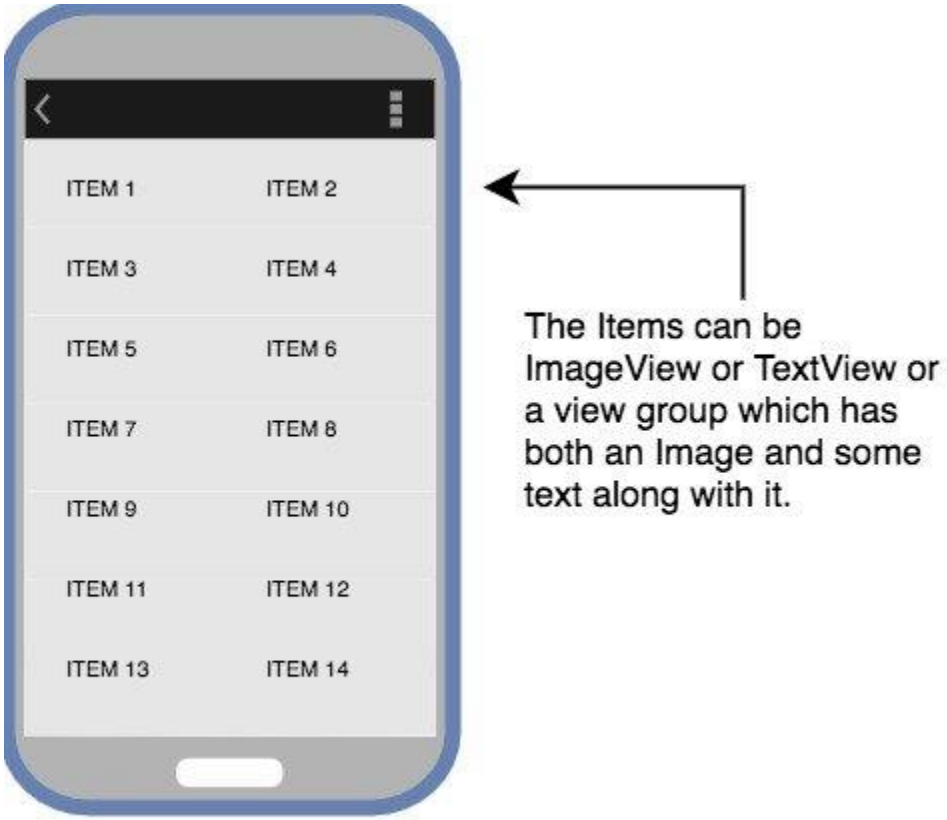
The view objects can be a Text view, an Image view or a view group which has both an image and some text. Best example for this view is phone gallery which shows images in a grid. We can set number of **columns** to specific number and auto-fit the images into the columns.

Vertical and horizontal spacing between every single items of gridView can be set by verticalSpacing and horizontalSpacing.

```
<GridView
    android:id="@+id/gridView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:verticalSpacing="2dp"
```

android:horizontalSpacing="2dp"

android:numColumns="2"/>



GridView in Android

11.1Using Adapter with GridView

In the last tutorial we explained how an Adapter converts data set onto view objects, which are then populated in AdapterView to create the UI components. Now our adapter view is GridView. You can refer the last tutorial for detailed explanations.

We will define a ListView in the main layout XML file activity_main.xml.

<?xml version="1.0" encoding="utf-8"?>

<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"

android:layout_width="match_parent"

android:layout_height="match_parent"

```
    android:background="#FFEB3B"
    tools:context="com.example.android.studytonightandroid.MainActivity">

    <GridView
        android:id="@+id/gridView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:verticalSpacing="10dp"
        android:horizontalSpacing="10dp"
        android:numColumns="2"/>

</android.support.constraint.ConstraintLayout>
```

12.Android Option Menu Example

Android Option Menus are the primary menus of android. They can be used for settings, search, delete item etc.

activity_main.xml

We have only one textview in this file.

File: activity_main.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.a
   ndroid.com/apk/res/android"
3.     xmlns:app="http://schemas.android.com/apk/res-auto"
4.     xmlns:tools="http://schemas.android.com/tools"
5.     android:layout_width="match_parent"
6.     android:layout_height="match_parent"
```

```
    tools:context="example.javatpoint.com.optionmenu.MainActivity">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />

    </android.support.design.widget.AppBarLayout>

    <include layout="@layout/content_main" />

</android.support.design.widget.CoordinatorLayout>
```

File: context_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="example.javatpoint.com.optionmenu.MainActivity"
    tools:showIn="@layout/activity_main">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

menu_main.xml

It contains three items as show below. It is created automatically inside the res/menu directory.

File: menu_main.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="example.javatpoint.com.optionmenu.MainActivity">

    <item  android:id="@+id/item1"
```

```
        android:title="Item 1"/>
    <item  android:id="@+id/item2"
        android:title="Item 2"/>
    <item  android:id="@+id/item3"
        android:title="Item 3"
        app:showAsAction="withText"/>
</menu>
```

Activity class

This class displays the content of menu.xml file and performs event handling on clicking the menu items.

File: MainActivity.java

```
package example.javatpoint.com.optionmenu;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

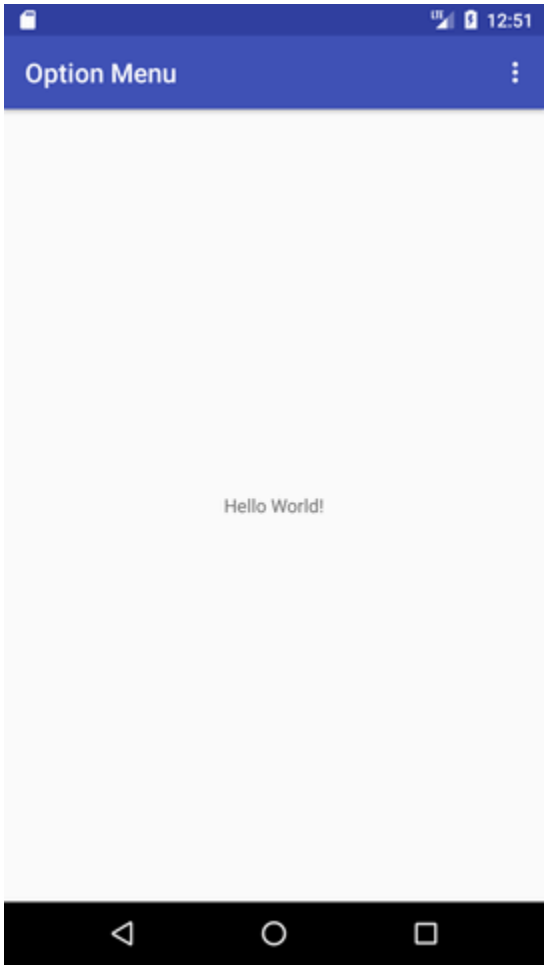
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
15.         setContentView(R.layout.activity_main);
16.         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
17.         setSupportActionBar(toolbar);
18.     }
19.
20.     @Override
21.     public boolean onCreateOptionsMenu(Menu menu) {
22.         // Inflate the menu; this adds items to the action bar if it is present.
23.         getMenuInflater().inflate(R.menu.menu_main, menu);
24.         return true;
25.     }
26.
27.     @Override
28.     public boolean onOptionsItemSelected(MenuItem item) {
29.       int id = item.getItemId();
30.         switch (id){
31.             case R.id.item1:
32.                 Toast.makeText(getApplicationContext(),"Item 1 Selected",Toast.LENGTH_L
    ONG).show();
33.                 return true;
34.             case R.id.item2:
35.                 Toast.makeText(getApplicationContext(),"Item 2 Selected",Toast.LENGTH_L
    ONG).show();
36.                 return true;
37.             case R.id.item3:
```

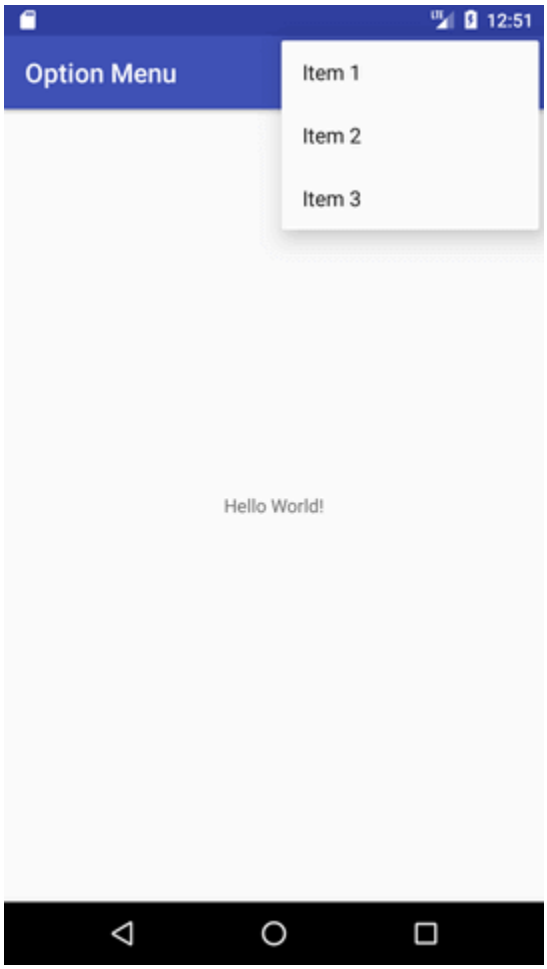
```
38.                 Toast.makeText(getApplicationContext(),"Item 3 Selected",Toast.LENGTH_LONG).show();
39.                 return true;
40.             default:
41.                 return super.onOptionsItemSelected(item);
42.         }
43.     }
44. }
```

Output:

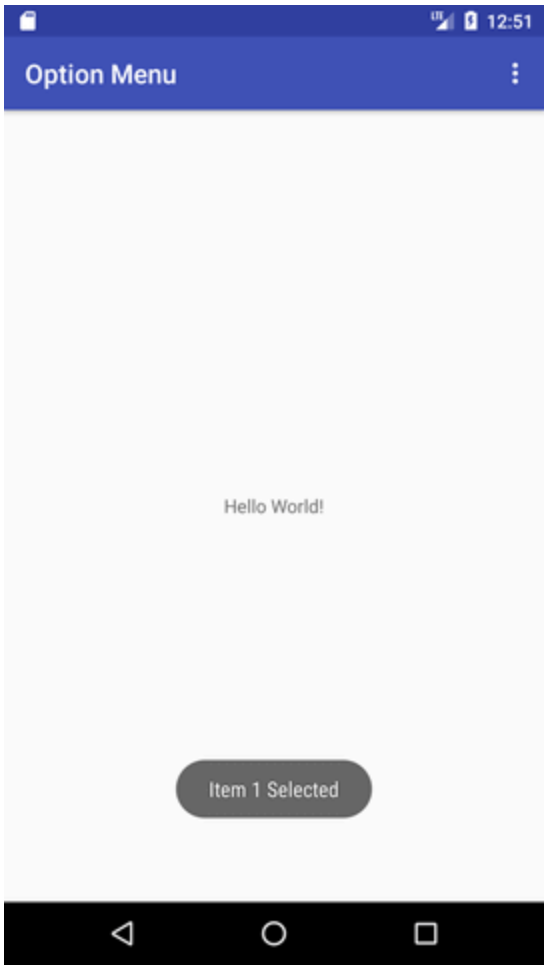
Output without clicking on the menu button.



Output after clicking on the menu button.



utput after clicking on the second menu item .



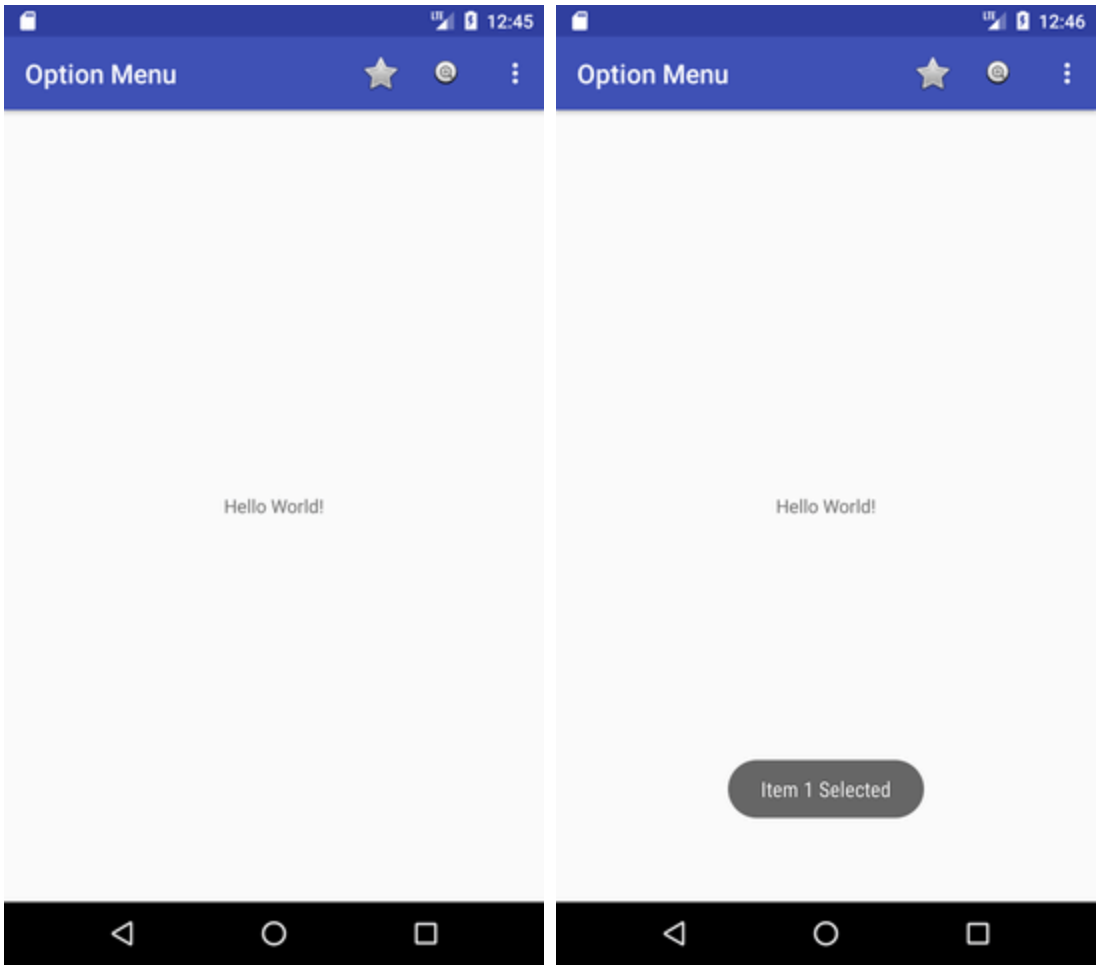
13.Option Menu with Icon

You need to have icon images inside the res/drawable directory. The android:icon element is used to display the icon on the option menu. You can write the string information in the strings.xml file. But we have written it inside the menu_main.xml file.

File: menu_main.xml

```
1. <menu xmlns:android="http://schemas.android.com/apk/res/android"
2.     xmlns:app="http://schemas.android.com/apk/res-auto"
3.     xmlns:tools="http://schemas.android.com/tools"
4.     tools:context="example.javatpoint.com.optionmenu.MainActivity">
5.
6.     <item  android:id="@+id/item1"
```

```
7.          android:title="Item 1"
8.          app:showAsAction="always"
9.          android:icon="@android:drawable/btn_star"/>
10.     <item  android:id="@+id/item2"
11.         android:title="Item 2"
12.         app:showAsAction="ifRoom"
13.         android:icon="@android:drawable/btn_plus"/>
14.     <item  android:id="@+id/item3"
15.         android:title="Item 3"
16.         app:showAsAction="withText"
17.         android:icon="@android:drawable/btn_plus"/>
18. </menu>
```



14 Android Context Menu Example

Android context menu appears when user press long click on the element. It is also known as floating menu.

It affects the selected content while doing action on it.

It doesn't support item shortcuts and icons.

Android Context Menu Example

activity_main.xml

Drag one listview from the pallete, now the xml file will look like this:

File: activity_main.xml

1. **<?xml** version="1.0" encoding="utf-8"**?>**

```
2. <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.androi
   d.com/apk/res/android"
3.     xmlns:app="http://schemas.android.com/apk/res-auto"
4.     xmlns:tools="http://schemas.android.com/tools"
5.     android:layout_width="match_parent"
6.     android:layout_height="match_parent"
7.     tools:context="example.javatpoint.com.contextmenu.MainActivity">
8.
9.     <ListView
10.        android:layout_width="368dp"
11.        android:layout_height="495dp"
12.        android:id="@+id/listView"
13.        android:layout_marginEnd="8dp"
14.        android:layout_marginStart="8dp"
15.        android:layout_marginTop="8dp"
16.        app:layout_constraintEnd_toEndOf="parent"
17.        app:layout_constraintHorizontal_bias="0.0"
18.        app:layout_constraintStart_toStartOf="parent"
19.        app:layout_constraintTop_toTopOf="parent" />
20. </android.support.constraint.ConstraintLayout>
```

main_menu.xml

Create a separate menu_main.xml file in menu directory for menu items.

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <menu xmlns:android="http://schemas.android.com/apk/res/android">
3.     <item android:id="@+id/call"
```

```
        android:title="Call" />
    <item android:id="@+id/sms"
        android:title="SMS" />
</menu>
```

Activity class

Let's write the code to display the context menu on press of the listview.

File: MainActivity.java

```
package example.javatpoint.com.contextmenu;

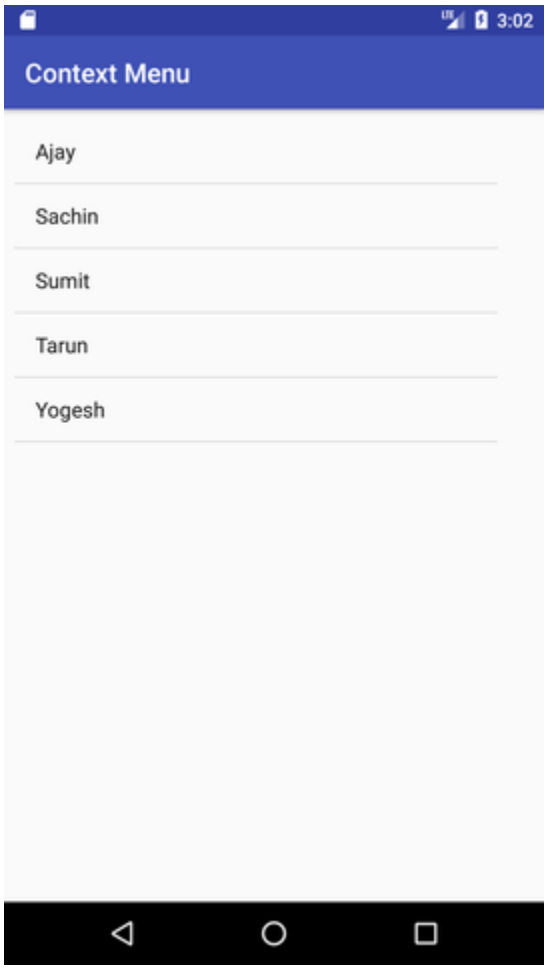
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    ListView listView;
    String contacts[]={"Ajay","Sachin","Sumit","Tarun","Yogesh"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

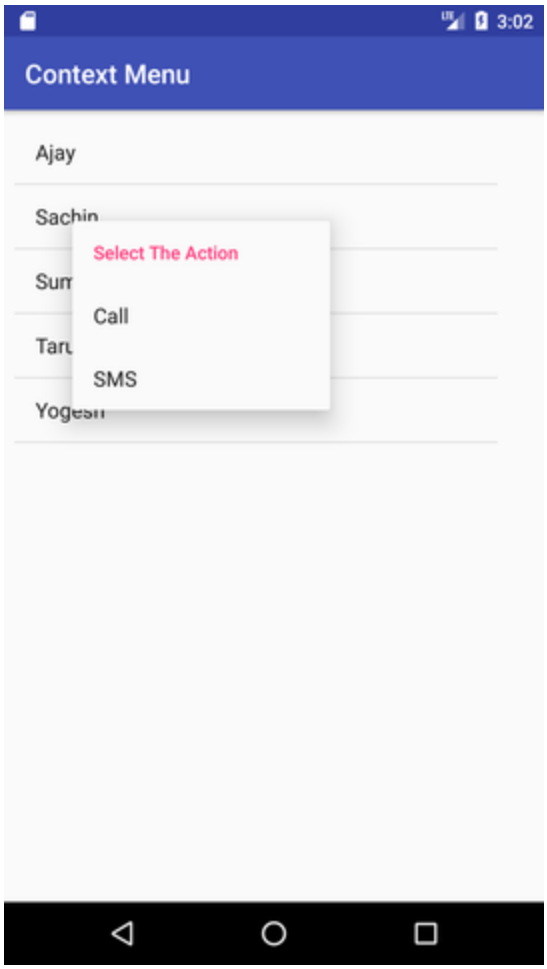
```
19.         setContentView(R.layout.activity_main);
20.         listView=(ListView)findViewById(R.id.listView);
21.         ArrayAdapter<String> adapter=new ArrayAdapter<String>(this,android.R.layout.si
    mple_list_item_1,contacts);
22.         listView.setAdapter(adapter);
23.         // Register the ListView  for Context menu
24.         registerForContextMenu(listView);
25.     }
26.     @Override
27.     public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.Contex
    tMenuInfo menuInfo)
28.     {
29.         super.onCreateContextMenu(menu, v, menuInfo);
30.         MenuInflater inflater = getMenuInflater();
31.         inflater.inflate(R.menu.menu_main, menu);
32.         menu.setHeaderTitle("Select The Action");
33.     }
34.     @Override
35.     public boolean onContextItemSelected(MenuItem item){
36.         if(item.getItemId()==R.id.call){
37.             Toast.makeText(getApplicationContext(),"calling code",Toast.LENGTH_LONG)
    .show();
38.         }
39.         else if(item.getItemId()==R.id.sms){
40.             Toast.makeText(getApplicationContext(),"sending sms code",Toast.LENGTH_L
    ONG).show();
41.         }else{
```

```
42.             return false;
43.         }
44.         return true;
45.     }
46. }
```

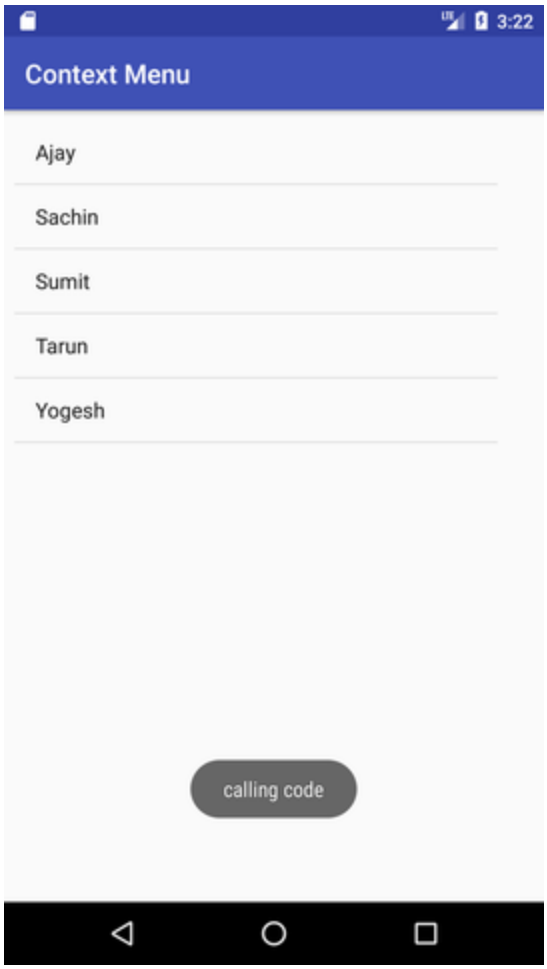
Output:



Output after long press on the listview.



Output after clicking on the context menu.



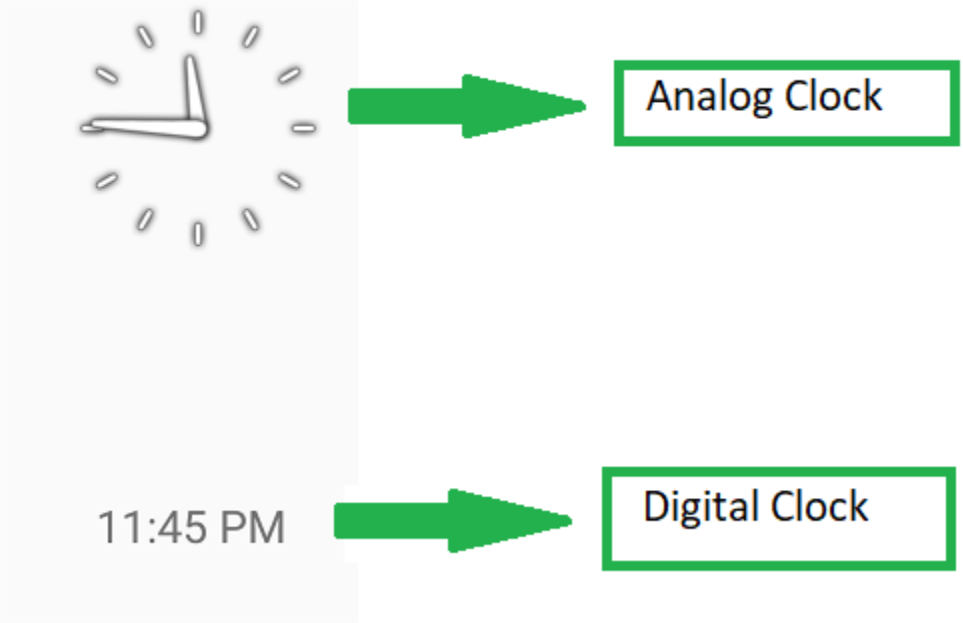
15.clock view

Analog and digital clocks are used for display the time in android application.

1. **Analog clock:** Analog clock is a subclass of **View** class. It represents a circular clock. Around the circle, numbers 1 to 12 appear to represent the hour and two hands are used to show instant of the time- shorter one for the hour and longer is for minutes.

2. **Digital clock:** Digital clock is subclass of **TextView** Class and uses numbers to display the time in “HH:MM” format.

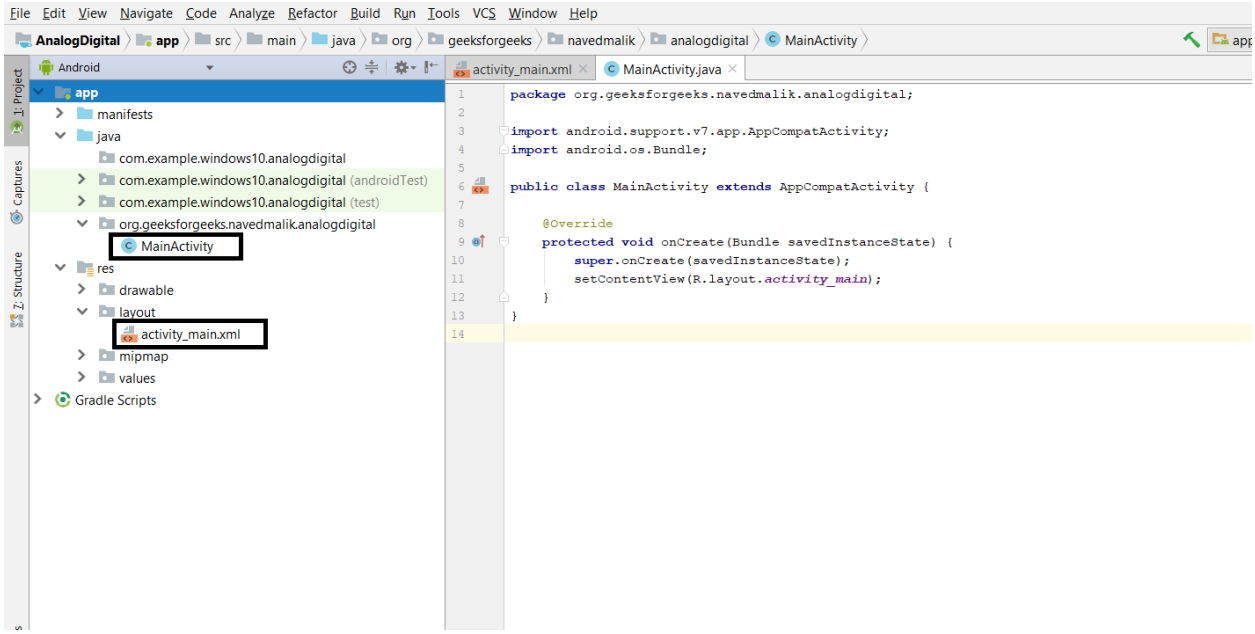
For Example



How to create a Android Analog clock and Digital clock?

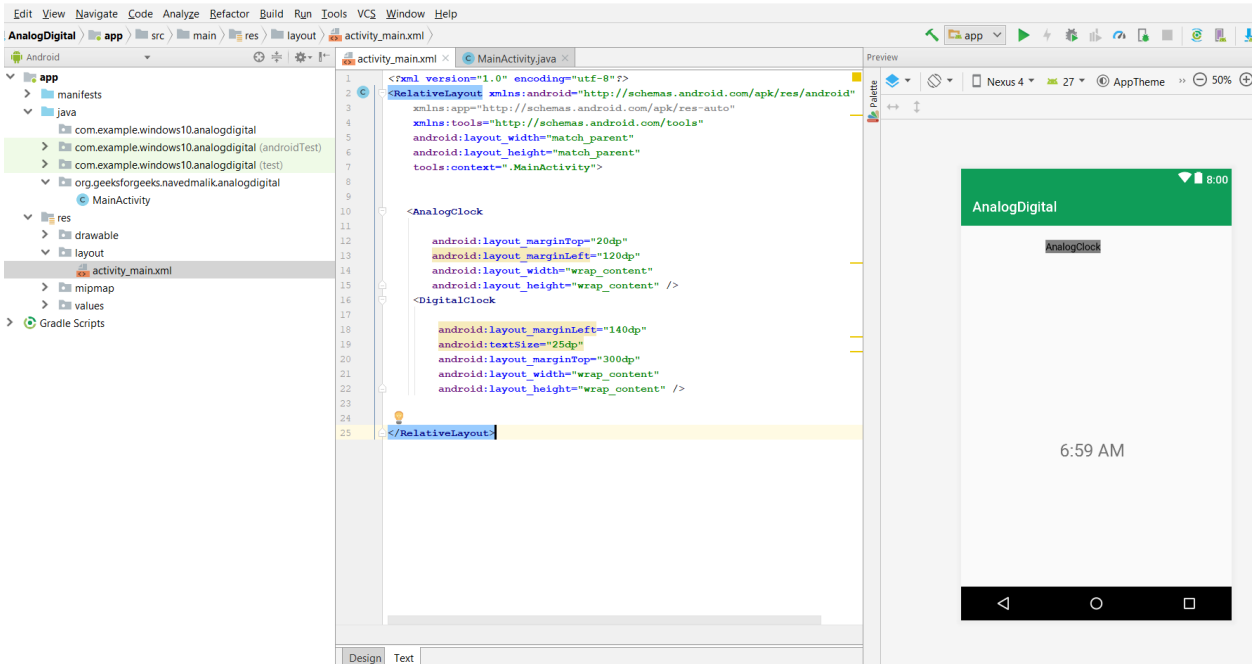
This example will help to develop an Android App that displays an Analog clock and a Digital clock according to the example shown above:
Below are the steps for Creating the Analog and Digital clock Android Application:

- **Step1:** Firstly create a new Android Application. This will create an XML file "activity_main.xml" and a Java File "MainActivity.Java". Please refer the pre-requisites to learn more about this step.



- **Step2:** Open “activity_main.xml” file and add following widgets in a Relative Layout:
    - An **Analog clock**
    - A **Digital clock**

This will make the UI of the Application. There is no need for assignment of IDs as these widgets will display the time by themselves.



- **Step3:** Leave the Java file as it is.
- **Step4:** Now Run the app. Both clocks are displayed on the screen.

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout

    xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"

    xmlns:tools="http://schemas.android.com/tools"

    android:layout_width="match_parent"

    android:layout_height="match_parent"

    tools:context=".MainActivity">



   <AnalogClock

       android:layout_marginTop="20dp"

       android:layout_marginLeft="120dp"

       android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content" />
    <DigitalClock


        android:layout_marginLeft="140dp"
        android:textSize="25dp"
        android:layout_marginTop="300dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</RelativeLayout>


  package org.geeksforgeeks.navedmalik.analogdigital;
  import android.support.v7.app.AppCompatActivity;
  import android.os.Bundle;
  public class MainActivity extends AppCompatActivity {
      @Override
      protected void onCreate(Bundle savedInstanceState)
      {
```

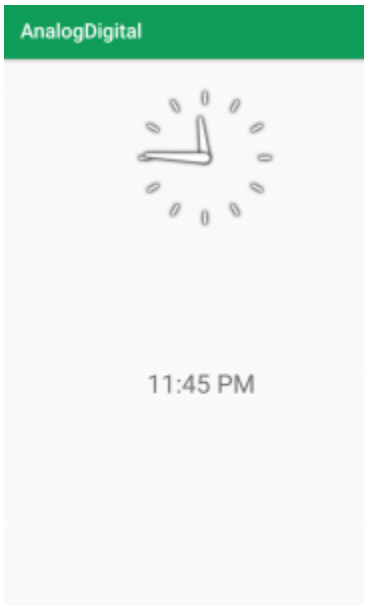
```
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

    }

}
```

Output:



16.Web view

WebView is a view that display web pages inside your application. You can also specify HTML string and can show it inside your application using WebView. WebView makes turns your application to a web application.

In order to add WebView to your application, you have to add **<WebView>** element to your xml layout file. Its syntax is as follows −

```
<WebView  xmlns:android="http://schemas.android.com/apk/res/android"

  android:id="@+id/webview"
```

```
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
/>
```

n order to use it, you have to get a reference of this view in Java file. To get a reference, create an object of the class WebView. Its syntax is −

```
WebView browser = (WebView) findViewById(R.id.webview);
```

In order to load a web url into the WebView, you need to call a method **loadUrl(String url)** of the WebView class, specifying the required url. Its syntax is:

```
browser.loadUrl("http://www.tutorialspoint.com");
```

Apart from just loading url, you can have more control over your WebView by using the methods defined in WebView class. They are listed as follows −

Sr.No	Method & Description
1	**canGoBack()** This method specifies the WebView has a back history item.
2	**canGoForward()** This method specifies the WebView has a forward history item.
3	**clearHistory()** This method will clear the WebView forward and backward history.
4	**destroy()** This method destroy the internal state of WebView.
5	**findAllAsync(String find)** This method find all instances of string and highlight them.

6	**getProgress()** This method gets the progress of the current page.
7	**getTitle()** This method return the title of the current page.
8	**getUrl()** This method return the url of the current page.

If you click on any link inside the webpage of the WebView, that page will not be loaded inside your WebView. In order to do that you need to extend your class from **WebViewClient** and override its method. Its syntax is −

```
private class MyBrowser extends WebViewClient {
   @Override
   public boolean shouldOverrideUrlLoading(WebView view, String url) {
      view.loadUrl(url);
      return true;
   }
}
```

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
   android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
```

```
android:paddingTop="@dimen/activity_vertical_margin"
android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActivity">

<TextView android:text="WebView" android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:id="@+id/textview"
  android:textSize="35dp"
  android:layout_alignParentTop="true"
  android:layout_centerHorizontal="true" />

<TextView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="Tutorials point"
  android:id="@+id/textView"
  android:layout_below="@+id/textview"
  android:layout_centerHorizontal="true"
  android:textColor="#ff7aff24"
  android:textSize="35dp" />

<EditText
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:id="@+id/editText"
  android:hint="Enter Text"
```

```
    android:focusable="true"
    android:textColorHighlight="#ff7eff15"
    android:textColorHint="#ffff25e6"
    android:layout_marginTop="46dp"
    android:layout_below="@+id/imageView"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_alignRight="@+id/imageView"
    android:layout_alignEnd="@+id/imageView" />

<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/imageView"
    android:src="@drawable/abc"
    android:layout_below="@+id/textView"
    android:layout_centerHorizontal="true" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Enter"
    android:id="@+id/button"
    android:layout_alignTop="@+id/editText"
    android:layout_toRightOf="@+id/imageView"
```

```
      android:layout_toEndOf="@+id/imageView" />

   <WebView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/webView"
      android:layout_below="@+id/button"
      android:layout_alignParentLeft="true"
      android:layout_alignParentStart="true"
      android:layout_alignParentRight="true"
      android:layout_alignParentEnd="true"
      android:layout_alignParentBottom="true" />

</RelativeLayout>
```

Following is the content of the **res/values/string.xml**.

```
<resources>
   <string name="app_name">My Application</string>
</resources>
```

Following is the content of **AndroidManifest.xml** file.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.sairamkrishna.myapplication" >
   <uses-permission android:name="android.permission.INTERNET" />
   <application
      android:allowBackup="true"
```

```
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >

    <activity
      android:name=".MainActivity"
      android:label="@string/app_name" >

      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>

    </activity>

  </application>
</manifest>
```

By just changing the url in the url field, your WebView will open your desired website.

17.Recycler View

RecyclerView makes it easy to efficiently display large sets of data. You supply the data and define how each item looks, and the RecyclerView library dynamically creates the elements when they're needed.

As the name implies, RecyclerView *recycles* those individual elements. When an item scrolls off the screen, RecyclerView doesn't destroy its view. Instead, RecyclerView reuses the view for new items that have scrolled onscreen. This reuse vastly improves performance, improving your app's responsiveness and reducing power consumption.

Key classes

Several different classes work together to build your dynamic list.

- **RecyclerView** is the ViewGroup that contains the views corresponding to your data. It's a view itself, so you add RecyclerView into your layout the way you would add any other UI element.
- Each individual element in the list is defined by a *view holder* object. When the view holder is created, it doesn't have any data associated with it. After the view holder is created, the RecyclerView *binds* it to its data. You define the view holder by extending RecyclerView.ViewHolder.
- The RecyclerView requests those views, and binds the views to their data, by calling methods in the *adapter*. You define the adapter by extending RecyclerView.Adapter.
- The *layout manager* arranges the individual elements in your list. You can use one of the layout managers provided by the RecyclerView library, or you can define your own. Layout managers are all based on the library's LayoutManager abstract class.

Steps for implementing your RecyclerView

If you're going to use RecyclerView, there are a few things you need to do. They'll be discussed in detail in the following sections.

- First of all, decide what the list or grid is going to look like. Ordinarily you'll be able to use one of the RecyclerView library's standard layout managers.
- Design how each element in the list is going to look and behave. Based on this design, extend the ViewHolder class. Your version of ViewHolder provides all the functionality for your list items. Your view holder is a wrapper around a View, and that view is managed by RecyclerView.
- Define the Adapter that associates your data with the ViewHolder views.

There are also advanced customization options that let you tailor your RecyclerView to your exact needs.

Plan your layout

The items in your RecyclerView are arranged by a LayoutManager class. The RecyclerView library provides three layout managers, which handle the most common layout situations:

- LinearLayoutManager arranges the items in a one-dimensional list.
- GridLayoutManager arranges all items in a two-dimensional grid:
    - If the grid is arranged vertically, GridLayoutManager tries to make all the elements in each row have the same width and height, but different rows can have different heights.
    - If the grid is arranged horizontally, GridLayoutManager tries to make all the elements in each column have the same width and height, but different columns can have different widths.
- StaggeredGridLayoutManager is similar to GridLayoutManager, but it does not require that items in a row have the same height (for vertical grids) or items in the same column have the same width (for horizontal grids). The result is that the items in a row or column can end up offset from each other.

You'll also need to design the layout of the individual items. You'll need this layout when you design the view holder, as described in the next section.

Implementing your adapter and view holder

Once you've determined your layout, you need to implement your Adapter and ViewHolder. These two classes work together to define how your data is displayed. The ViewHolder is a wrapper around a View that contains the layout for an individual item in the list.
The Adapter creates ViewHolder objects as needed, and also sets the data for those views. The process of associating views to their data is called *binding*.

When you define your adapter, you need to override three key methods:

- onCreateViewHolder(): RecyclerView calls this method whenever it needs to create a new ViewHolder. The method creates and initializes the ViewHolder and its associated View, but does *not* fill in the view's contents—the ViewHolder has not yet been bound to specific data.
- onBindViewHolder(): RecyclerView calls this method to associate a ViewHolder with data. The method fetches the appropriate data and uses the data to fill in the view holder's layout. For example, if the RecyclerView displays a list of names, the method might find the appropriate name in the list and fill in the view holder's TextView widget.

- getItemCount(): RecyclerView calls this method to get the size of the data set. For example, in an address book app, this might be the total number of addresses. RecyclerView uses this to determine when there are no more items that can be displayed.

- public class CustomAdapter extends RecyclerView.Adapter<CustomAdapter.ViewHolder> {

```
    private String[] localDataSet;

    /**
     * Provide a reference to the type of views that you are using
     * (custom ViewHolder).
     */
    public static class ViewHolder extends RecyclerView.ViewHolder {
        private final TextView textView;

        public ViewHolder(View view) {
            super(view);
            // Define click listener for the ViewHolder's View

            textView = (TextView) view.findViewById(R.id.textView);
        }

        public TextView getTextView() {
            return textView;
        }
    }

    /**
     * Initialize the dataset of the Adapter.
     *
     * @param dataSet String[] containing the data to populate views to be used
     * by RecyclerView.
     */
    public CustomAdapter(String[] dataSet) {
        localDataSet = dataSet;
    }

    // Create new views (invoked by the layout manager)
    @Override
    public ViewHolder onCreateViewHolder(ViewGroup viewGroup, int viewType) {
        // Create a new view, which defines the UI of the list item
        View view = LayoutInflater.from(viewGroup.getContext())
                .inflate(R.layout.text_row_item, viewGroup, false);
```

```
        return new ViewHolder(view);
    }

    // Replace the contents of a view (invoked by the layout manager)
    @Override
    public void onBindViewHolder(ViewHolder viewHolder, final int position) {

        // Get element from your dataset at this position and replace the
        // contents of the view with that element
        viewHolder.getTextView().setText(localDataSet[position]);
    }

    // Return the size of your dataset (invoked by the layout manager)
    @Override
    public int getItemCount() {
        return localDataSet.length;
    }
}
```

The layout for the each view item is defined in an XML layout file, as usual. In this case, the app has a text_row_item.xml file like this:

- ```
  <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_width="match_parent"
      android:layout_height="@dimen/list_item_height"
      android:layout_marginLeft="@dimen/margin_medium"
      android:layout_marginRight="@dimen/margin_medium"
      android:gravity="center_vertical">

      <TextView
          android:id="@+id/textView"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="@string/element_text"/>
  </FrameLayout>
  ```
  ```

SUBJECT NAME : MOBILE APPLICATION DEVELOPMENT

SUBJECT CODE : CCS51

CLASS : III BSC CS

SEMESTER : ODD

UNIT III: DATA PERSISTENCE

INDEX

Different Data Persistence schemes: Shared preferences – File Handling – Managing data using SQLite database – Content providers: user content provider – Android in build content providers.

DIFFERENT DATA PERSISTENCE SCHEMES:

1. SHARED PREFERENCES

Interface for accessing and modifying preference data returned by Context.getSharedPreferences(String, int). For any particular set of preferences, there is a single instance of this class that all clients share. Modifications to the preferences must go through an Editor object to ensure the preference values remain in a consistent state and control when they are committed to storage. Objects that are returned from the various get methods must be treated as immutable by the application.

Note: This class provides strong consistency guarantees. It is using expensive operations which might slow down an app. Frequently changing properties or properties where loss can be tolerated should use other mechanisms. For more details read the comments on Editor#commit() and Editor#apply().

FILE HANDLING

File based persistence

Methods of local data persistence

Android allows to persists application data via the file system. For each application the Android system creates a *data/data/[application package]* directory.

Android supports the following ways of storing data in the local file system:

- Files - You can create and update files
- Preferences - Android allows you to save and retrieve persistent key-value pairs of primitive data type.
- SQLite database - instances of SQLite databases are also stored on the local file system.

Files are saved in the *files* folder and application settings are saved as XML files in the *shared_prefs* folder.

If your application creates an SQLite database this database is saved in the main application directory under the *databases* folder.

The following screenshot shows a file system which contains file, cache files and preferences.

Only the application can write into its application directory. It can create additional sub-directories in this application directory. For these sub-directories, the application can grant read or write permissions for other applications.

1.2. Internal vs. external storage

Android has internal storage and external storage. External storage is not private and may not always be availale. If, for example, the Android device is connected with a computer, the computer may mount the external system via USB and that makes this external storage not avaiable for Android applications.

1.3. Application on external storage

As of Android 8 SDK level it is possible to define that the application can or should be placed on external storage. For this set the android:installLocation to preferExternal or auto.

In this case certain application components may be stored on an encrypted external mount point. Database and other private data will still be stored in the internal storage system.

2. Preferences

2.1. Storing key-value pairs

The *SharedPreferences* class allows to persists key-value pairs of primitive data types in the Android file system.

The PreferenceManager class provides methods to get access to these preferences. The following code shows how to access preferences from a certain file

```
# getting preferences from a specified file
SharedPreferences settings = getSharedPreferences("Test", Context.MODE_PRIVATE);
```

Preferences should be created private for the application. They can be accessed via all application components.

A default store for preferences can be accessed via the PreferenceManager.getDefaultSharedPreferences(this) method call. Preference value are accessed via the key and the instance of the SharedPreferences class, as demonstrated in the following listing.

```
SharedPreferences settings = PreferenceManager.getDefaultSharedPreferences(getActivity());
String url = settings.getString("url", "n/a");
```

To create or change preferences you have to call the edit() method on the SharedPreferences object. Once you have changed the value you have to call the apply() method to apply your asynchronously to the file system.

```
Editor edit = preferences.edit();
edit.putString("username", "new_value_for_user");
edit.apply();
```

> The usage of the commit() method is discouraged, as it write the changes synchronously to the file system.

2.2. Preference Listener

You can listen to changes in the preferences via the registerOnSharedPreferenceChangeListener() method on SharedPreferences.

```
SharedPreferences prefs =
```

```
   PreferenceManager.getDefaultSharedPreferences(this);

// Instance field for listener
listener = new SharedPreferences.OnSharedPreferenceChangeListener() {
  public void onSharedPreferenceChanged(SharedPreferences prefs, String key) {
    // Your Implementation
  }
};

prefs.registerOnSharedPreferenceChangeListener(listener);
```

One watch out is that SharedPreferences keeps listeners in a WeakHashMap hence listener may be recycled if your code does not hold a reference to it.

2.3. User interface for preferences

Android allows you to create XML resource files which describes preference key-values. An instance of PreferenceActivity or PreferenceFragment can generate an user interface for these files. The user interfaces takes care of persisting the key-value pairs.

To create a preference resource file of type *XML*.

Android provides the PreferenceFragment class which simplifies the creation of an user interface for maintaining preference values. This fragment can load an XML preference definition file via the method addPreferencesFromResource().

3. Exercise: Prerequisites

The following exercise assumes that you have created an Android project with the top-level package *com.example.android.rssfeed*. This application has at least one entry in the toolbar with the R.id.action_settings id. Once this toolbar entry is selected, an existing fragment is replaced.

4. Exercise: Preference setting for the RSS feed

In this exercise you allow the user to enter his preferred URL for an RSS feed via another fragment. This fragment uses a preference xml file to define the user interface.

4.1. Create preference file

Create an Android XML resource called *mypreferences.xml* in the xml folder. Add entries to this file similar to the following listing.

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >

   <EditTextPreference
      android:key="url"
      android:title="Rss feed URL"
      android:inputType="textUri"/>
   <CheckBoxPreference android:title="Aktiv" android:key="active"/>

</PreferenceScreen>
```

4.2. Create settings fragment

Create the class SettingsFragment. It should extends PreferenceFragment. This fragment loads the preference file and allows the user to change the values.

```
package com.example.android.rssreader;

import android.os.Bundle;
import android.preference.PreferenceFragment;

public class SettingsFragment extends PreferenceFragment  {
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      addPreferencesFromResource(R.xml.mypreferences);
   }
}
```

4.3. Connect your settings fragment

Ensure that you open the preference fragment via the onOptionsItemSelected() method. The relevant code is demonstrated in the following listing.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
   switch (item.getItemId()) {
   // more code...
   case R.id.action_settings:
      // Launch settings activity
      if (getResources().getBoolean(R.bool.twoPaneMode)) {
         getFragmentManager().beginTransaction().replace(R.id.detailFragment, new
```

```
                SettingsFragment()).commit();
        } else {
            getFragmentManager().beginTransaction().addToBackStack(null).replace(R.id
                    .fragment_container, new SettingsFragment()).commit();
        }
        return true;
        // more code...
    }
    // more code...
}
```

4.4. Use the preference value to load the RSS feed

The following code snippet demonstrates how you can access the preference value in a method.

```
// if you use this in a service or activity you can use this
// if you use this in a fragment use getActivity() or getContent() as parameter
SharedPreferences settings = PreferenceManager.getDefaultSharedPreferences(this);
String url = settings.getString("url", "https://www.vogella.com/article.rss");
```

4.5. Validate

Run your application. Select from your action bar the Settings action. You should be able to enter a URL. If you press the back button and press the refresh button, ensure that the value of the url preference is used in your activity.

4.6. Optional: Show the current value in the settings

The following code snippet demonstrates how to show the current value in the preference screen.

```
package com.example.android.rssreader;

import android.content.SharedPreferences;
import android.os.Bundle;
import android.preference.EditTextPreference;
import android.preference.Preference;
import android.preference.PreferenceCategory;
import android.preference.PreferenceFragment;

public class SettingsFragment extends PreferenceFragment implements
SharedPreferences.OnSharedPreferenceChangeListener {
```

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    addPreferencesFromResource(R.xml.mypreferences);
    // show the current value in the settings screen
    for (int i = 0; i < getPreferenceScreen().getPreferenceCount(); i++) {
        initSummary(getPreferenceScreen().getPreference(i));
    }
}

@Override
public void onResume() {
    super.onResume();
    getPreferenceScreen().getSharedPreferences()
            .registerOnSharedPreferenceChangeListener(this);
}

@Override
public void onPause() {
    super.onPause();
    getPreferenceScreen().getSharedPreferences()
            .unregisterOnSharedPreferenceChangeListener(this);
}

@Override
public void onSharedPreferenceChanged(SharedPreferences sharedPreferences,
                                      String key) {
    updatePreferences(findPreference(key));
}

private void initSummary(Preference p) {
    if (p instanceof PreferenceCategory) {
        PreferenceCategory cat = (PreferenceCategory) p;
        for (int i = 0; i < cat.getPreferenceCount(); i++) {
            initSummary(cat.getPreference(i));
        }
    } else {
        updatePreferences(p);
    }
}
```

```
    private void updatePreferences(Preference p) {
        if (p instanceof EditTextPreference) {
            EditTextPreference editTextPref = (EditTextPreference) p;
            p.setSummary(editTextPref.getText());
        }
    }
}
```

5. Android File API

5.1. Using the file API

Access to the file system is performed via the standard java.io classes. Android provides also helper classes for creating and accessing new files and directories. For example the getDir(String, int) method would create or access a directory. The openFileInput(String s) method provides access to an FileInputStream for the file. The openFileOutput(String s, Context.MODE_PRIVATE) method provides access to an FileOutputStream for the file.

All modes except Context.MODE_PRIVATE are deprecated, files should be private to the application.

The following example shows the API usage.

```
public class Util {
    public static void writeConfiguration(Context ctx, String s ) {
        try (FileOutputStream openFileOutput =
             ctx.openFileOutput( "config.txt", Context.MODE_PRIVATE);) {

            openFileOutput.write(s.getBytes());
        } catch (Exception e) {
            // not handled
        }
    }
}

public void readFileFromInternalStorage(String fileName) {
    String eol = System.getProperty("line.separator");
    try (BufferedReader input = new BufferedReader(new InputStreamReader(
        openFileInput(fileName))); ){
      String line;
```

```
      StringBuffer buffer = new StringBuffer();
      while ((line = input.readLine()) != null) {
        buffer.append(line + eol);
      }
    } catch (Exception e) {
       // we do not care
    }
}
```

5.2. External storage

Android supports also access to an external storage system, e.g., the SD card. All files and directories on the external storage system are readable for all applications with the correct permission.

To read from external storage your application need to have the android.permission.READ_EXTERNAL_STORAGE permission.

To write to the external storage system your application needs the android.permission.WRITE_EXTERNAL_STORAGE permission. You get the path to the external storage system via the Environment.getExternalStorageDirectory() method.

Via the following method call you can check the state of the external storage system. If the Android device is connected via USB to a computer, external storage might not be available.

```
Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)
```

The following shows an example for reading from the external storage system.

```
private void readFileFromSDCard() {
    File directory = Environment.getExternalStorageDirectory();
    // assumes that a file article.rss is available on the SD card
    File file = new File(directory + "/article.rss");
    if (!file.exists()) {
       throw new RuntimeException("File not found");
    }
    Log.e("Testing", "Starting to read");
    BufferedReader reader = null;
    try {
       reader = new BufferedReader(new FileReader(file));
       StringBuilder builder = new StringBuilder();
       String line;
       while ((line = reader.readLine()) != null) {
```

```
            builder.append(line);
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (reader != null) {
            try {
                reader.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Android - Shared Preferences

Android provides many ways of storing data of an application. One of this way is called Shared Preferences. Shared Preferences allow you to save and retrieve data in the form of key,value pair.

In order to use shared preferences, you have to call a method getSharedPreferences() that returns a SharedPreference instance pointing to the file that contains the values of preferences.

```
SharedPreferences sharedpreferences = getSharedPreferences(MyPREFERENCES, Context.MODE_PRI
```

The first parameter is the key and the second parameter is the MODE. Apart from private there are other modes available that are listed below −

Sr.No	Mode & description
1	**MODE_APPEND** This will append the new preferences with the already existing preferences
2	**MODE_ENABLE_WRITE_AHEAD_LOGGING** Database open flag. When it is set , it would enable write ahead logging by default
3	**MODE_MULTI_PROCESS** This method will check for modification of preferences even if the sharedpreference instance has already been loaded
4	**MODE_PRIVATE** By setting this mode, the file can only be accessed using calling application
5	**MODE_WORLD_READABLE** This mode allow other application to read the preferences
6	**MODE_WORLD_WRITEABLE** This mode allow other application to write the preferences

You can save something in the sharedpreferences by using SharedPreferences.Editor class. You will call the edit method of SharedPreference instance and will receive it in an editor object. Its syntax is −

```
Editor editor = sharedpreferences.edit();
editor.putString("key", "value");
editor.commit();
```

Apart from the putString method , there are methods available in the editor class that allows manipulation of data inside shared preferences. They are listed as follows −

Sr. NO	Mode & description
1	**apply()** It is an abstract method. It will commit your changes back from editor to the sharedPreference object you are calling
2	**clear()** It will remove all values from the editor
3	**remove(String key)** It will remove the value whose key has been passed as a parameter
4	**putLong(String key, long value)** It will save a long value in a preference editor
5	**putInt(String key, int value)** It will save a integer value in a preference editor
6	**putFloat(String key, float value)** It will save a float value in a preference editor

Example

This example demonstrates the use of the Shared Preferences. It display a screen with some text fields, whose value are saved when the application is closed and brought back when it is opened again.

To experiment with this example, you need to run this on an actual device on after developing the application according to the steps below −

Steps	Description
1	You will use Android studio to create an Android application under a package com.example.sairamkrishna.myapplication.
2	Modify src/MainActivity.java file to add progress code to display the spinning progress dialog.
3	Modify res/layout/activity_main.xml file to add respective XML code.
4	Run the application and choose a running android device and install the application on it and verify the results.

Following is the content of the modified **MainActivity.java.**

```
package com.example.sairamkrishna.myapplication;

import android.content.Context;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
   EditText ed1,ed2,ed3;
   Button b1;

   public static final String MyPREFERENCES = "MyPrefs" ;
   public static final String Name = "nameKey";
   public static final String Phone = "phoneKey";
   public static final String Email = "emailKey";

   SharedPreferences sharedpreferences;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      ed1=(EditText)findViewById(R.id.editText);
      ed2=(EditText)findViewById(R.id.editText2);
      ed3=(EditText)findViewById(R.id.editText3);
```

```
        b1=(Button)findViewById(R.id.button);
        sharedpreferences = getSharedPreferences(MyPREFERENCES, Context.MODE_PRIVATE);

        b1.setOnClickListener(new View.OnClickListener() {
           @Override
           public void onClick(View v) {
              String n  = ed1.getText().toString();
              String ph  = ed2.getText().toString();
              String e  = ed3.getText().toString();

              SharedPreferences.Editor editor = sharedpreferences.edit();

              editor.putString(Name, n);
              editor.putString(Phone, ph);
              editor.putString(Email, e);
              editor.commit();
              Toast.makeText(MainActivity.this,"Thanks",Toast.LENGTH_LONG).show();
           }
        });
     }

  }
```

Following is the content of the modified main activity file**res/layout/activiy_main.xml.**

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
   android:layout_height="match_parent" android:paddingLeft="@dimen/activity_horizont
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActiv

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Shared Preference "
      android:id="@+id/textView"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:textSize="35dp" />
```

```
<TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Tutorials Point"
   android:id="@+id/textView2"
   android:layout_below="@+id/textView"
   android:layout_centerHorizontal="true"
   android:textSize="35dp"
   android:textColor="#ff16ff01" />

<EditText
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:id="@+id/editText"
   android:layout_below="@+id/textView2"
   android:layout_marginTop="67dp"
   android:hint="Name"
   android:layout_alignParentRight="true"
   android:layout_alignParentEnd="true"
   android:layout_alignParentLeft="true"
   android:layout_alignParentStart="true" />

<EditText
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:id="@+id/editText2"
   android:layout_below="@+id/editText"
   android:layout_alignParentLeft="true"
   android:layout_alignParentStart="true"
   android:layout_alignParentRight="true"
   android:layout_alignParentEnd="true"
   android:hint="Pass" />

<EditText
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:id="@+id/editText3"
   android:layout_below="@+id/editText2"
   android:layout_alignParentLeft="true"
   android:layout_alignParentStart="true"
   android:layout_alignParentRight="true"
   android:layout_alignParentEnd="true"
```

```
      android:hint="Email" />

   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Save"
      android:id="@+id/button"
      android:layout_below="@+id/editText3"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="50dp" />

</RelativeLayout>
```

Following is the content of the modified content of file **res/values/strings.xml.**

```
<resources>
   <string name="app_name">My Application</string>
</resources>
```

Following is the content default file **AndroidManifest.xml.**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.sairamkrishna.myapplication" >

   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >

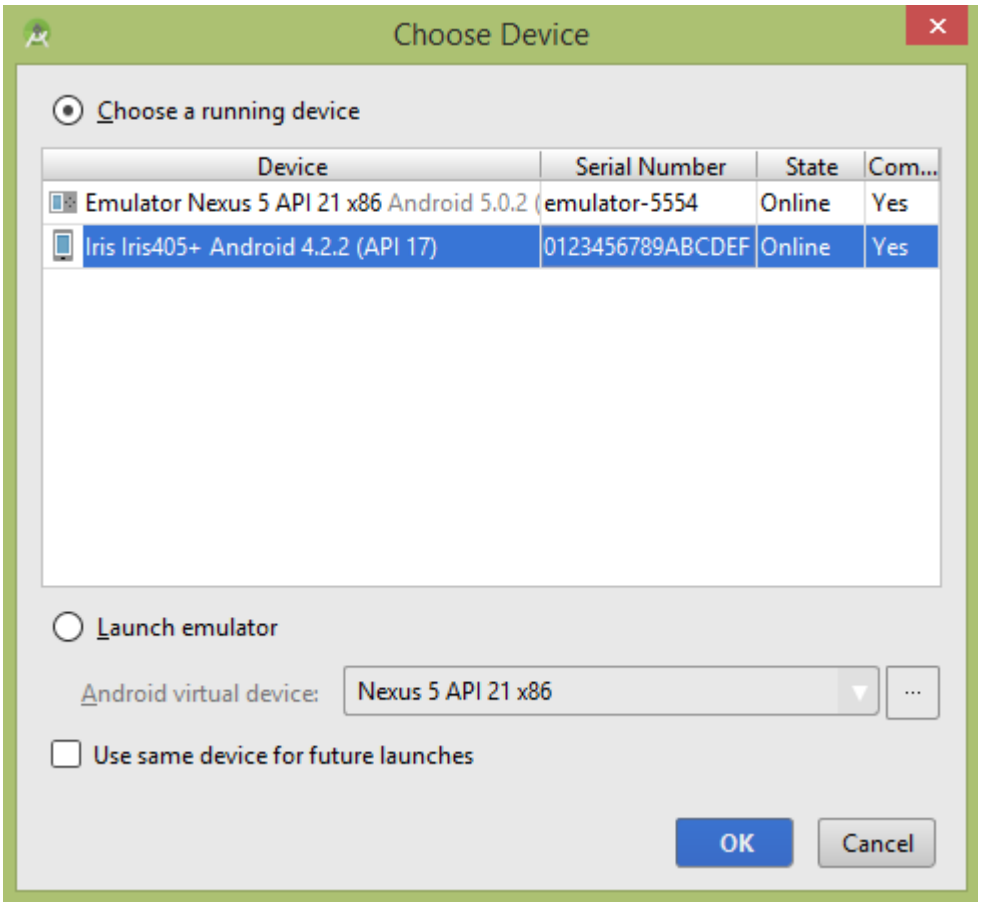
      <activity
         android:name=".MainActivity"
         android:label="@string/app_name" >

         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>

      </activity>
```

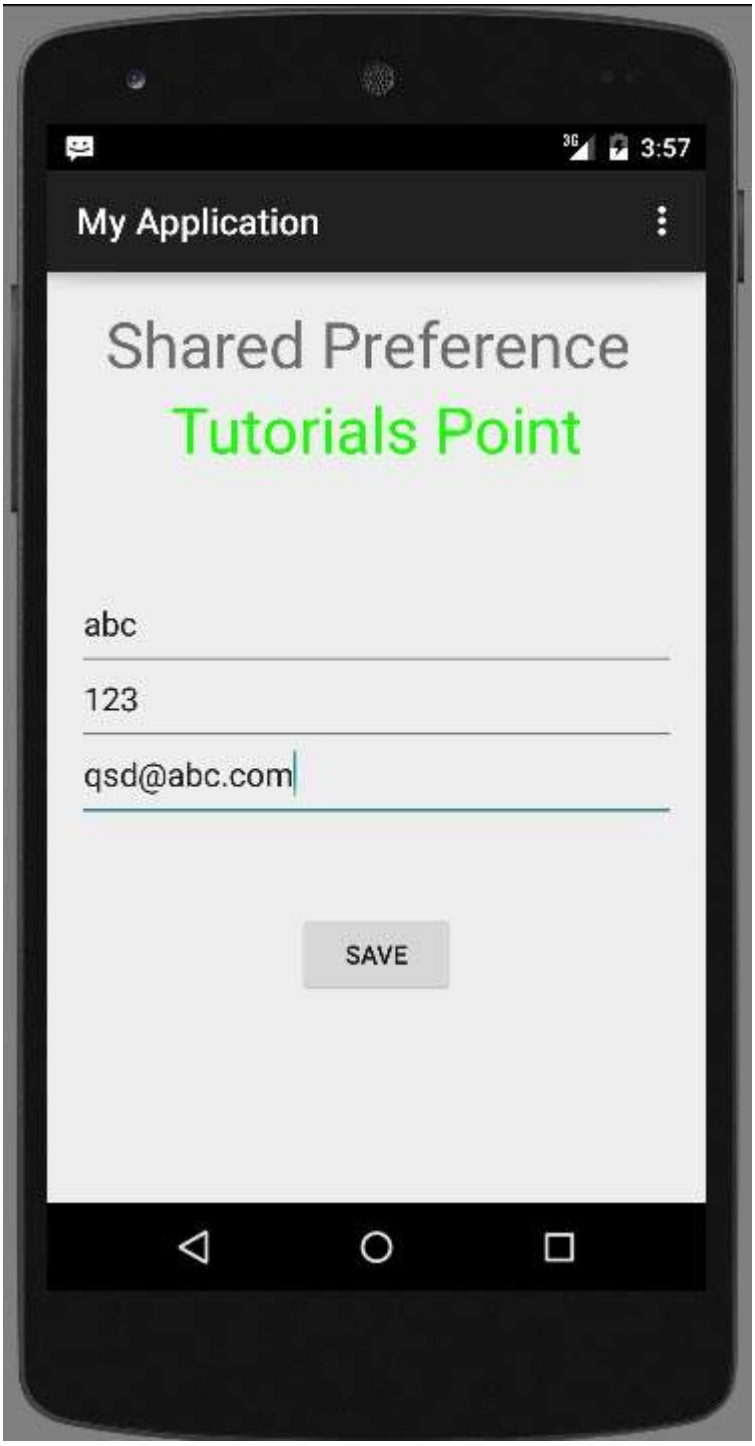
```
    </application>
</manifest>
```

Let's try to run your application. I assume you have connected your actual Android Mobile device with your computer. To run the app from Android studio, open one of your project's activity files and click Run icon from the toolbar. Before starting your application, Android studio will display following window to select an option where you want to run your Android application.

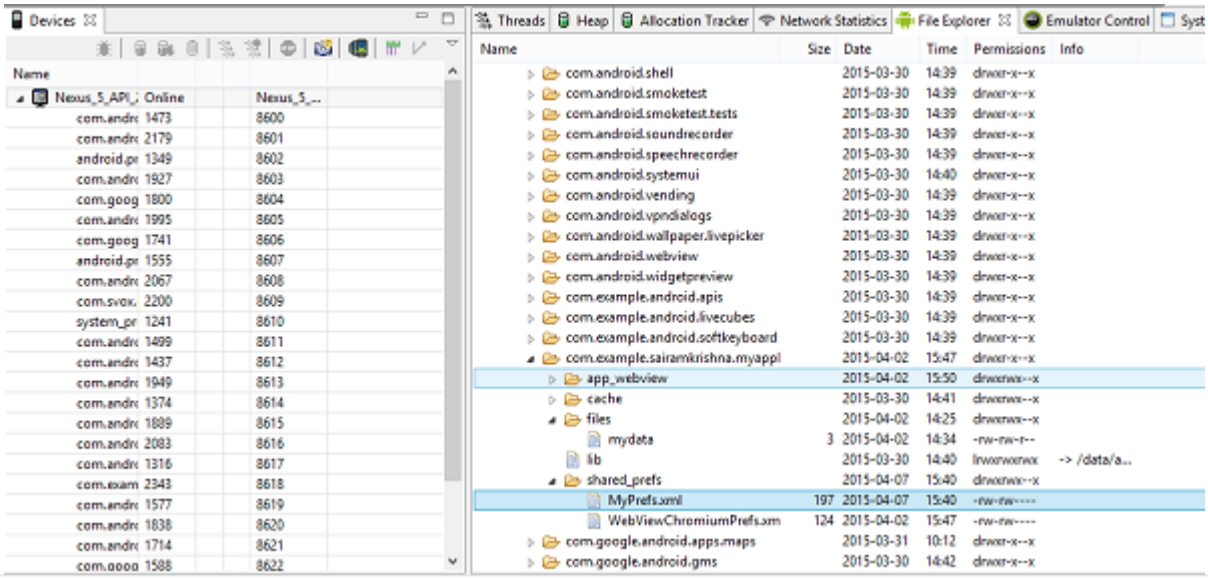


Select your mobile device as an option and then check your mobile device which will display following screen −

Now just put in some text in the field. Like i put some random name and other information and click on save button.



Now when you press save button, the text will be saved in the shared preferences. Now press back button and exit the application. Now open it again and you will see all the text you have written back in your application.


Devices
Name
Nexus_5_API_; Online
Nexus_5_...
com.andr 1473 8600
com.andr 2179 8601
android.pr 1349 8602
com.andr 1927 8603
com.goog 1800 8604
com.andr 1995 8605
com.goog 1741 8606
android.pr 1555 8607
com.andr 2067 8608
com.svox. 2200 8609
system_pr 1241 8610
com.andr 1499 8611
com.andr 1437 8612
com.andr 1949 8613
com.andr 1374 8614
com.andr 1889 8615
com.andr 2083 8616
com.andr 1316 8617
com.exam 2343 8618
com.andr 1577 8619
com.andr 1838 8620
com.andr 1714 8621
com.aaaa 1588 8622
Threads
Heap
Allocation Tracker
Network Statistics
File Explorer
Emulator Control
Name
Size
Date
Time
Permissions
Info
com.android.shell 2015-03-30 14:39 drwxr-x--x
com.android.smoketest 2015-03-30 14:39 drwxr-x--x
com.android.smoketest.tests 2015-03-30 14:39 drwxr-x--x
com.android.soundrecorder 2015-03-30 14:39 drwxr-x--x
com.android.speechrecorder 2015-03-30 14:39 drwxr-x--x
com.android.systemui 2015-03-30 14:40 drwxr-x--x
com.android.vending 2015-03-30 14:39 drwxr-x--x
com.android.vpndialogs 2015-03-30 14:39 drwxr-x--x
com.android.wallpaper.livepicker 2015-03-30 14:39 drwxr-x--x
com.android.webview 2015-03-30 14:39 drwxr-x--x
com.android.widgetpreview 2015-03-30 14:39 drwxr-x--x
com.example.android.apis 2015-03-30 14:39 drwxr-x--x
com.example.android.livecubes 2015-03-30 14:39 drwxr-x--x
com.example.android.softkeyboard 2015-03-30 14:39 drwxr-x--x
com.example.sairamkrishna.myappl 2015-04-02 15:47 drwxr-x--x
app_webview 2015-04-02 15:50 drwxrwx--x
cache 2015-03-30 14:41 drwxrwx--x
files 2015-04-02 14:25 drwxrwx--x
mydata 3 2015-04-02 14:34 -rw-rw-r--
lib 2015-03-30 14:40 lrwxrwxrwx -> /data/a...
shared_prefs 2015-04-07 15:40 drwxrwx--x
MyPrefs.xml 197 2015-04-07 15:40 -rw-rw----
WebViewChromiumPrefs.xm 124 2015-04-02 15:47 -rw-rw----
com.google.android.apps.maps 2015-03-31 10:12 drwxr-x--x
com.google.android.gms 2015-03-30 14:42 drwxr-x--x

Android - SQLite Database

SQLite is a opensource SQL database that stores data to a text file on a device. Android comes in with built in SQLite database implementation.

SQLite supports all the relational database features. In order to access this database, you don't need to establish any kind of connections for it like JDBC,ODBC e.t.c

Database - Package

The main package is android.database.sqlite that contains the classes to manage your own databases

Database - Creation

In order to create a database you just need to call this method openOrCreateDatabase with your database name and mode as a parameter. It returns an instance of SQLite database which you have to receive in your own object.Its syntax is given below

```
SQLiteDatabase mydatabase = openOrCreateDatabase("your database name",MODE_PRIVATE,null);
```

Apart from this , there are other functions available in the database package , that does this job. They are listed below

Sr.No	Method & Description
1	**openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags, DatabaseErrorHandler errorHandler)** This method only opens the existing database with the appropriate flag mode. The common flags mode could be OPEN_READWRITE OPEN_READONLY
2	**openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags)** It is similar to the above method as it also opens the existing database but it does not define any handler to handle the errors of databases
3	**openOrCreateDatabase(String path, SQLiteDatabase.CursorFactory factory)** It not only opens but create the database if it not exists. This method is equivalent to openDatabase method.
4	**openOrCreateDatabase(File file, SQLiteDatabase.CursorFactory factory)** This method is similar to above method but it takes the File object as a path rather then a string. It is equivalent to file.getPath()

Database - Insertion

we can create table or insert data into table using execSQL method defined in SQLiteDatabase class. Its syntax is given below

```
mydatabase.execSQL("CREATE TABLE IF NOT EXISTS TutorialsPoint(Username VARCHAR,Password VA
mydatabase.execSQL("INSERT INTO TutorialsPoint VALUES('admin','admin');");
```

This will insert some values into our table in our database. Another method that also does the same job but take some additional parameter is given below

Sr.No	Method & Description
1	**execSQL(String sql, Object[] bindArgs)** This method not only insert data , but also used to update or modify already existing data in database using bind arguments

Database - Fetching

We can retrieve anything from database using an object of the Cursor class. We will call a method of this class called rawQuery and it will return a resultset with the cursor pointing to the table. We can move the cursor forward and retrieve the data.

```
Cursor resultSet = mydatbase.rawQuery("Select * from TutorialsPoint",null);
resultSet.moveToFirst();
String username = resultSet.getString(0);
String password = resultSet.getString(1);
```

There are other functions available in the Cursor class that allows us to effectively retrieve the data. That includes

Sr.No	Method & Description
1	**getColumnCount()** This method return the total number of columns of the table.
2	**getColumnIndex(String columnName)** This method returns the index number of a column by specifying the name of the column
3	**getColumnName(int columnIndex)** This method returns the name of the column by specifying the index of the column
4	**getColumnNames()** This method returns the array of all the column names of the table.
5	**getCount()** This method returns the total number of rows in the cursor
6	**getPosition()** This method returns the current position of the cursor in the table
7	**isClosed()** This method returns true if the cursor is closed and return false otherwise

Database - Helper class

For managing all the operations related to the database , an helper class has been given and is called SQLiteOpenHelper. It automatically manages the creation and update of the database. Its syntax is given below

```
public class DBHelper extends SQLiteOpenHelper {
   public DBHelper(){
      super(context,DATABASE_NAME,null,1);
   }
   public void onCreate(SQLiteDatabase db) {}
```

```
    public void onUpgrade(SQLiteDatabase database, int oldVersion, int newVersion) {}
}
```

Example

Here is an example demonstrating the use of SQLite Database. It creates a basic contacts applications that allows insertion, deletion and modification of contacts.

To experiment with this example, you need to run this on an actual device on which camera is supported.

Steps	Description
1	You will use Android studio to create an Android application under a package com.example.sairamkrishna.myapplication.
2	Modify src/MainActivity.java file to get references of all the XML components and populate the contacts on listView.
3	Create new src/DBHelper.java that will manage the database work
4	Create a new Activity as DisplayContact.java that will display the contact on the screen
5	Modify the res/layout/activity_main to add respective XML components
6	Modify the res/layout/activity_display_contact.xml to add respective XML components
7	Modify the res/values/string.xml to add necessary string components
8	Modify the res/menu/display_contact.xml to add necessary menu components
9	Create a new menu as res/menu/mainmenu.xml to add the insert contact option
10	Run the application and choose a running android device and install the application on it and verify the results.

Following is the content of the modified **MainActivity.java**.

```
package com.example.sairamkrishna.myapplication;

import android.content.Context;
import android.content.Intent;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;

import android.view.KeyEvent;
```

```
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;

import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;


import java.util.ArrayList;
import java.util.List;

public class MainActivity extends ActionBarActivity {
   public final static String EXTRA_MESSAGE = "MESSAGE";
   private ListView obj;
   DBHelper mydb;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      mydb = new DBHelper(this);
      ArrayList array_list = mydb.getAllCotacts();
      ArrayAdapter arrayAdapter=new ArrayAdapter(this,android.R.layout.simple_list_i

      obj = (ListView)findViewById(R.id.listView1);
      obj.setAdapter(arrayAdapter);
      obj.setOnItemClickListener(new OnItemClickListener(){
         @Override
         public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,long arg3)
            // TODO Auto-generated method stub
            int id_To_Search = arg2 + 1;

            Bundle dataBundle = new Bundle();
            dataBundle.putInt("id", id_To_Search);

            Intent intent = new Intent(getApplicationContext(),DisplayContact.class)

            intent.putExtras(dataBundle);
            startActivity(intent);
         }
      });
```

```
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
       // Inflate the menu; this adds items to the action bar if it is present.
       getMenuInflater().inflate(R.menu.menu_main, menu);
       return true;
    }


    @Override
    public boolean onOptionsItemSelected(MenuItem item){
       super.onOptionsItemSelected(item);

       switch(item.getItemId()) {
          case R.id.item1:Bundle dataBundle = new Bundle();
          dataBundle.putInt("id", 0);

          Intent intent = new Intent(getApplicationContext(),DisplayContact.class);
          intent.putExtras(dataBundle);

          startActivity(intent);
          return true;
          default:
          return super.onOptionsItemSelected(item);
       }
    }

    public boolean onKeyDown(int keycode, KeyEvent event) {
       if (keycode == KeyEvent.KEYCODE_BACK) {
          moveTaskToBack(true);
       }
       return super.onKeyDown(keycode, event);
    }
}
```

Following is the modified content of display contact activity **DisplayContact.java**

```
package com.example.sairamkrishna.myapplication;

import android.os.Bundle;
import android.app.Activity;
import android.app.AlertDialog;
```

```
import android.app.AlertDialog;

import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;


import android.view.Menu;
import android.view.MenuItem;
import android.view.View;

import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class DisplayContact extends Activity {
   int from_Where_I_Am_Coming = 0;
   private DBHelper mydb ;

   TextView name ;
   TextView phone;
   TextView email;
   TextView street;
   TextView place;
   int id_To_Update = 0;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_display_contact);
      name = (TextView) findViewById(R.id.editTextName);
      phone = (TextView) findViewById(R.id.editTextPhone);
      email = (TextView) findViewById(R.id.editTextStreet);
      street = (TextView) findViewById(R.id.editTextEmail);
      place = (TextView) findViewById(R.id.editTextCity);

      mydb = new DBHelper(this);

      Bundle extras = getIntent().getExtras();
      if(extras !=null) {
         int Value = extras.getInt("id");

         if(Value>0){
            //means this is the view part not the add contact part.
```

```
            Cursor rs = mydb.getData(Value);
            id_To_Update = Value;
            rs.moveToFirst();

            String nam = rs.getString(rs.getColumnIndex(DBHelper.CONTACTS_COLUMN_NAMI
            String phon = rs.getString(rs.getColumnIndex(DBHelper.CONTACTS_COLUMN_PH(

            String emai = rs.getString(rs.getColumnIndex(DBHelper.CONTACTS_COLUMN_EM/
            String stree = rs.getString(rs.getColumnIndex(DBHelper.CONTACTS_COLUMN_S
            String plac = rs.getString(rs.getColumnIndex(DBHelper.CONTACTS_COLUMN_CI

            if (!rs.isClosed())  {
               rs.close();
            }
            Button b = (Button)findViewById(R.id.button1);
            b.setVisibility(View.INVISIBLE);

            name.setText((CharSequence)nam);
            name.setFocusable(false);
            name.setClickable(false);

            phone.setText((CharSequence)phon);
            phone.setFocusable(false);
            phone.setClickable(false);

            email.setText((CharSequence)emai);
            email.setFocusable(false);
            email.setClickable(false);

            street.setText((CharSequence)stree);
            street.setFocusable(false);
            street.setClickable(false);

            place.setText((CharSequence)plac);
            place.setFocusable(false);
            place.setClickable(false);
         }
      }
   }

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      // Inflate the menu; this adds items to the action bar if it is present.
      Bundle extras = getIntent().getExtras();
```

```
      if(extras !=null) {
         int Value = extras.getInt("id");
         if(Value>0){
            getMenuInflater().inflate(R.menu.display_contact, menu);
         } else{

            getMenuInflater().inflate(R.menu.menu_main menu);
         }
      }
      return true;
   }

   public boolean onOptionsItemSelected(MenuItem item) {
      super.onOptionsItemSelected(item);
      switch(item.getItemId()) {
         case R.id.Edit_Contact:
         Button b = (Button)findViewById(R.id.button1);
         b.setVisibility(View.VISIBLE);
         name.setEnabled(true);
         name.setFocusableInTouchMode(true);
         name.setClickable(true);

         phone.setEnabled(true);
         phone.setFocusableInTouchMode(true);
         phone.setClickable(true);

         email.setEnabled(true);
         email.setFocusableInTouchMode(true);
         email.setClickable(true);

         street.setEnabled(true);
         street.setFocusableInTouchMode(true);
         street.setClickable(true);

         place.setEnabled(true);
         place.setFocusableInTouchMode(true);
         place.setClickable(true);

         return true;
         case R.id.Delete_Contact:

         AlertDialog.Builder builder = new AlertDialog.Builder(this);
         builder.setMessage(R.string.deleteContact)
```

```
            .setPositiveButton(R.string.yes, new DialogInterface.OnClickListener() {
               public void onClick(DialogInterface dialog, int id) {
                  mydb.deleteContact(id_To_Update);
                  Toast.makeText(getApplicationContext(), "Deleted Successfully",
                     Toast.LENGTH_SHORT).show();
                  Intent intent = new Intent(getApplicationContext(),MainActivity.cl

                  startActivity(intent);
               }
            })
            .setNegativeButton(R.string.no, new DialogInterface.OnClickListener() {
               public void onClick(DialogInterface dialog, int id) {
                  // User cancelled the dialog
               }
            });

            AlertDialog d = builder.create();
            d.setTitle("Are you sure");
            d.show();

            return true;
            default:
            return super.onOptionsItemSelected(item);

         }
      }

      public void run(View view) {
         Bundle extras = getIntent().getExtras();
         if(extras !=null) {
            int Value = extras.getInt("id");
            if(Value>0){
               if(mydb.updateContact(id_To_Update,name.getText().toString(),
                  phone.getText().toString(), email.getText().toString(),
                                       street.getText().toString(), place.getText().toSt
                  Toast.makeText(getApplicationContext(), "Updated", Toast.LENGTH_SHORT
                  Intent intent = new Intent(getApplicationContext(),MainActivity.class
                  startActivity(intent);
               } else{
                  Toast.makeText(getApplicationContext(), "not Updated", Toast.LENGTH_SH
               }
            } else{
               if(mydb.insertContact(name.getText().toString(), phone.getText().toStri
                                       email.getText().toString(), street.getText().t
```

```
                                    place.getText().toString())){
                Toast.makeText(getApplicationContext(), "done",
                                                    Toast.LENGTH_SHORT).show();
            } else{
               Toast.makeText(getApplicationContext(), "not done",
                                                Toast.LENGTH_SHORT).show();

            }
            Intent intent = new Intent(getApplicationContext(),MainActivity.class);
            startActivity(intent);
         }
      }
   }
}
```

Following is the content of Database class **DBHelper.java**

```
package com.example.sairamkrishna.myapplication;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Hashtable;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.DatabaseUtils;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteDatabase;

public class DBHelper extends SQLiteOpenHelper {

   public static final String DATABASE_NAME = "MyDBName.db";
   public static final String CONTACTS_TABLE_NAME = "contacts";
   public static final String CONTACTS_COLUMN_ID = "id";
   public static final String CONTACTS_COLUMN_NAME = "name";
   public static final String CONTACTS_COLUMN_EMAIL = "email";
   public static final String CONTACTS_COLUMN_STREET = "street";
   public static final String CONTACTS_COLUMN_CITY = "place";
   public static final String CONTACTS_COLUMN_PHONE = "phone";
   private HashMap hp;

   public DBHelper(Context context) {
      super(context, DATABASE_NAME , null, 1);
```

```
}

@Override
public void onCreate(SQLiteDatabase db) {
   // TODO Auto-generated method stub
   db.execSQL(
      "create table contacts " +
      "(id integer primary key, name text,phone text,email text, street text,place
   );
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
   // TODO Auto-generated method stub
   db.execSQL("DROP TABLE IF EXISTS contacts");
   onCreate(db);
}

public boolean insertContact (String name, String phone, String email, String stre
   SQLiteDatabase db = this.getWritableDatabase();
   ContentValues contentValues = new ContentValues();
   contentValues.put("name", name);
   contentValues.put("phone", phone);
   contentValues.put("email", email);
   contentValues.put("street", street);
   contentValues.put("place", place);
   db.insert("contacts", null, contentValues);

   return true;
}

public Cursor getData(int id) {
   SQLiteDatabase db = this.getReadableDatabase();
   Cursor res =  db.rawQuery( "select * from contacts where id="+id+"", null );
   return res;
}

public int numberOfRows(){
   SQLiteDatabase db = this.getReadableDatabase();
   int numRows = (int) DatabaseUtils.queryNumEntries(db, CONTACTS_TABLE_NAME);
   return numRows;
}

public boolean updateContact (Integer id, String name, String phone, String email
```

```
      SQLiteDatabase db = this.getWritableDatabase();
      ContentValues contentValues = new ContentValues();
      contentValues.put("name", name);
      contentValues.put("phone", phone);
      contentValues.put("email", email);
      contentValues.put("street", street);
      contentValues.put("place", place);
      db.update("contacts", contentValues, "id = ? ", new String[] { Integer.toStrin
      return true;
   }

   public Integer deleteContact (Integer id) {
      SQLiteDatabase db = this.getWritableDatabase();
      return db.delete("contacts",
      "id = ? ",
      new String[] { Integer.toString(id) });
   }

   public ArrayList<String> getAllCotacts() {
      ArrayList<String> array_list = new ArrayList<String>();

      //hp = new HashMap();
      SQLiteDatabase db = this.getReadableDatabase();
      Cursor res =  db.rawQuery( "select * from contacts", null );
      res.moveToFirst();

      while(res.isAfterLast() == false){

         array_list.add(res.getString(res.getColumnIndex(CONTACTS_COLUMN_NAME)));
         res.moveToNext();
      }
      return array_list;
   }
}
```

Following is the content of the **res/layout/activity_main.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent
   android:layout_height="match_parent"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
```

```
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActiv

<TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"

   android:id="@+id/textView"
   android:layout_alignParentTop="true"
   android:layout_centerHorizontal="true"
   android:textSize="30dp"
   android:text="Data Base" />

<TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Tutorials Point"
   android:id="@+id/textView2"
   android:layout_below="@+id/textView"
   android:layout_centerHorizontal="true"
   android:textSize="35dp"
   android:textColor="#ff16ff01" />

<ImageView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:id="@+id/imageView"
   android:layout_below="@+id/textView2"
   android:layout_centerHorizontal="true"
   android:src="@drawable/logo"/>

<ScrollView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:id="@+id/scrollView"
   android:layout_below="@+id/imageView"
   android:layout_alignParentLeft="true"
   android:layout_alignParentStart="true"
   android:layout_alignParentBottom="true"
   android:layout_alignParentRight="true"
   android:layout_alignParentEnd="true">

   <ListView
```

```
            android:id="@+id/listView1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_centerVertical="true" >
         </ListView>


      </ScrollView>

   </RelativeLayout>
```

Following is the content of the **res/layout/activity_display_contact.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/scrollView1"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   tools:context=".DisplayContact" >

   <RelativeLayout
      android:layout_width="match_parent"
      android:layout_height="370dp"
      android:paddingBottom="@dimen/activity_vertical_margin"
      android:paddingLeft="@dimen/activity_horizontal_margin"
      android:paddingRight="@dimen/activity_horizontal_margin"
      android:paddingTop="@dimen/activity_vertical_margin">

      <EditText
         android:id="@+id/editTextName"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_alignParentLeft="true"
         android:layout_marginTop="5dp"
         android:layout_marginLeft="82dp"
         android:ems="10"
         android:inputType="text" >
      </EditText>

      <EditText
```

```
      android:id="@+id/editTextEmail"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignLeft="@+id/editTextStreet"
      android:layout_below="@+id/editTextStreet"
      android:layout_marginTop="22dp"
      android:ems="10"
      android:inputType="textEmailAddress" />

   <TextView
      android:id="@+id/textView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignBottom="@+id/editTextName"
      android:layout_alignParentLeft="true"
      android:text="@string/name"
      android:textAppearance="?android:attr/textAppearanceMedium" />

   <Button
      android:id="@+id/button1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignLeft="@+id/editTextCity"
      android:layout_alignParentBottom="true"
      android:layout_marginBottom="28dp"
      android:onClick="run"
      android:text="@string/save" />

   <TextView
      android:id="@+id/textView2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignBottom="@+id/editTextEmail"
      android:layout_alignLeft="@+id/textView1"
      android:text="@string/email"
      android:textAppearance="?android:attr/textAppearanceMedium" />

   <TextView
      android:id="@+id/textView5"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignBottom="@+id/editTextPhone"
      android:layout_alignLeft="@+id/textView1"
```

```
      android:text="@string/phone"
      android:textAppearance="?android:attr/textAppearanceMedium" />

   <TextView
      android:id="@+id/textView4"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_above="@+id/editTextEmail"
      android:layout_alignLeft="@+id/textView5"
      android:text="@string/street"
      android:textAppearance="?android:attr/textAppearanceMedium" />

   <EditText
      android:id="@+id/editTextCity"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignRight="@+id/editTextName"
      android:layout_below="@+id/editTextEmail"
      android:layout_marginTop="30dp"
      android:ems="10"
      android:inputType="text" />

   <TextView
      android:id="@+id/textView3"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignBaseline="@+id/editTextCity"
      android:layout_alignBottom="@+id/editTextCity"
      android:layout_alignParentLeft="true"
      android:layout_toLeftOf="@+id/editTextEmail"
      android:text="@string/country"
      android:textAppearance="?android:attr/textAppearanceMedium" />

   <EditText
      android:id="@+id/editTextStreet"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignLeft="@+id/editTextName"
      android:layout_below="@+id/editTextPhone"
      android:ems="10"
      android:inputType="text" >

      <requestFocus />
```

```
      </EditText>

      <EditText
         android:id="@+id/editTextPhone"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_alignLeft="@+id/editTextStreet"
         android:layout_below="@+id/editTextName"
         android:ems="10"
         android:inputType="phone|text" />

   </RelativeLayout>
</ScrollView>
```

Following is the content of the **res/value/string.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="app_name">Address Book</string>
   <string name="action_settings">Settings</string>
   <string name="hello_world">Hello world!</string>
   <string name="Add_New">Add New</string>
   <string name="edit">Edit Contact</string>
   <string name="delete">Delete Contact</string>
   <string name="title_activity_display_contact">DisplayContact</string>
   <string name="name">Name</string>
   <string name="phone">Phone</string>
   <string name="email">Email</string>
   <string name="street">Street</string>
   <string name="country">City/State/Zip</string>
   <string name="save">Save Contact</string>
   <string name="deleteContact">Are you sure, you want to delete it.</string>
   <string name="yes">Yes</string>
   <string name="no">No</string>
</resources>
```

Following is the content of the **res/menu/main_menu.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

   <item android:id="@+id/item1"
```

```
      android:icon="@drawable/add"
      android:title="@string/Add_New" >
   </item>

</menu>
```

Following is the content of the **res/menu/display_contact.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
   <item
      android:id="@+id/Edit_Contact"
      android:orderInCategory="100"
      android:title="@string/edit"/>

   <item
      android:id="@+id/Delete_Contact"
      android:orderInCategory="100"
      android:title="@string/delete"/>

</menu>
```

This is the defualt **AndroidManifest.xml** of this project

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.sairamkrishna.myapplication" >

   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >

      <activity
         android:name=".MainActivity"
         android:label="@string/app_name" >

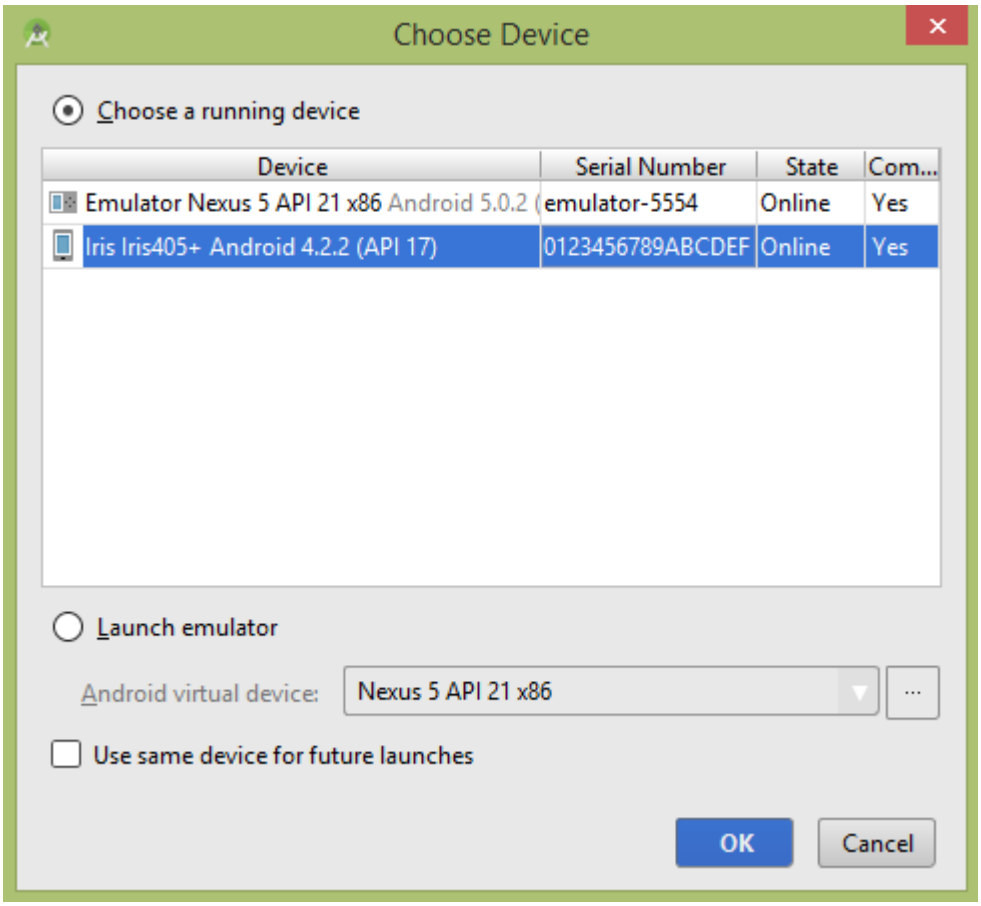
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
```

```
        </activity>

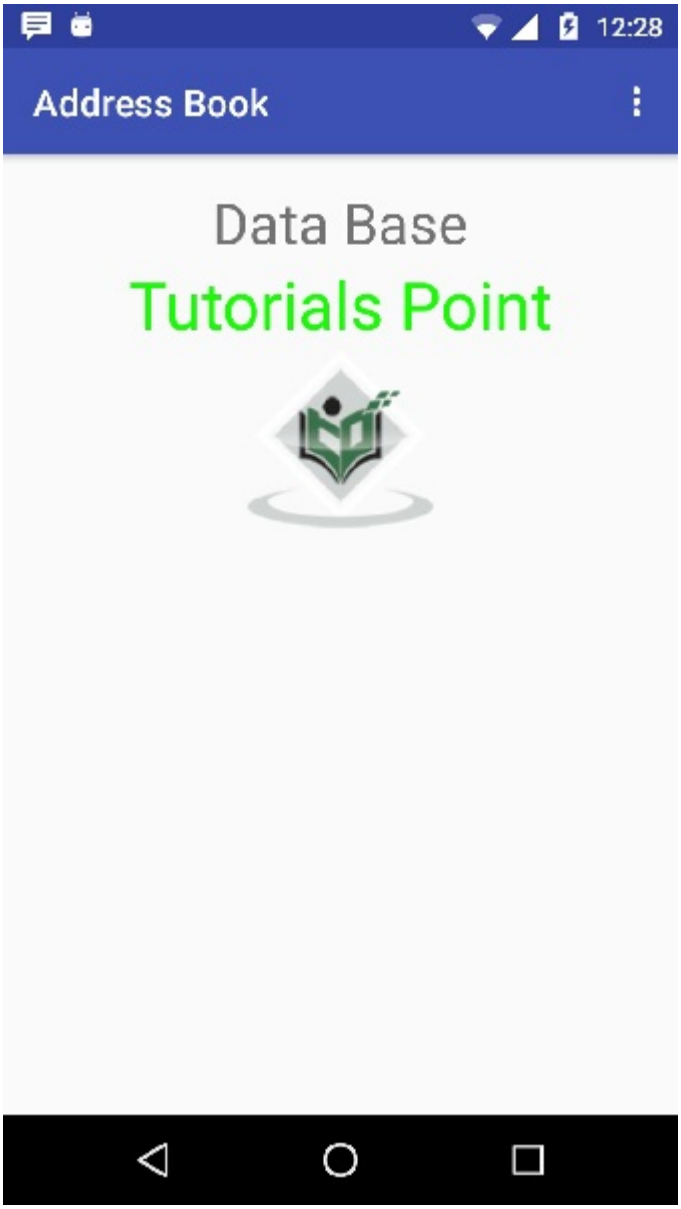
        <activity android:name=".DisplayContact"/>

    </application>
</manifest>
```

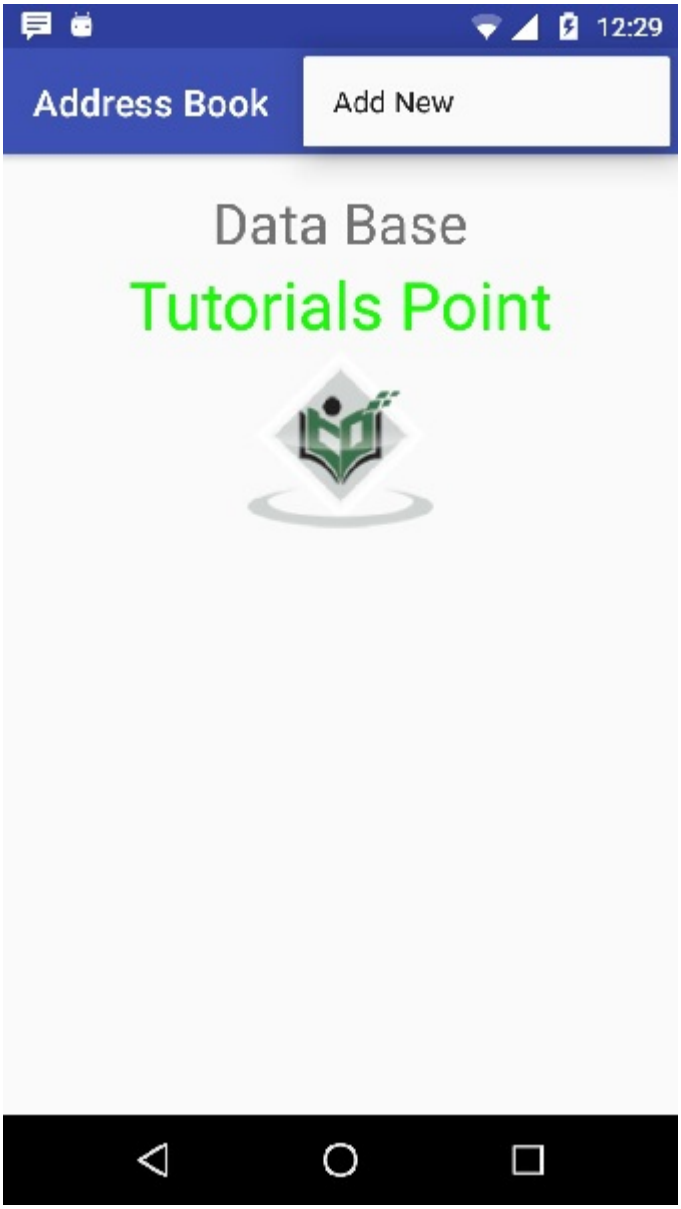
Let's try to run your application. I assume you have connected your actual Android Mobile device with your computer. To run the app from Android studio , open one of your project's activity files and click Run icon from the tool bar. Before starting your application,Android studio will display following window to select an option where you want to run your Android application.



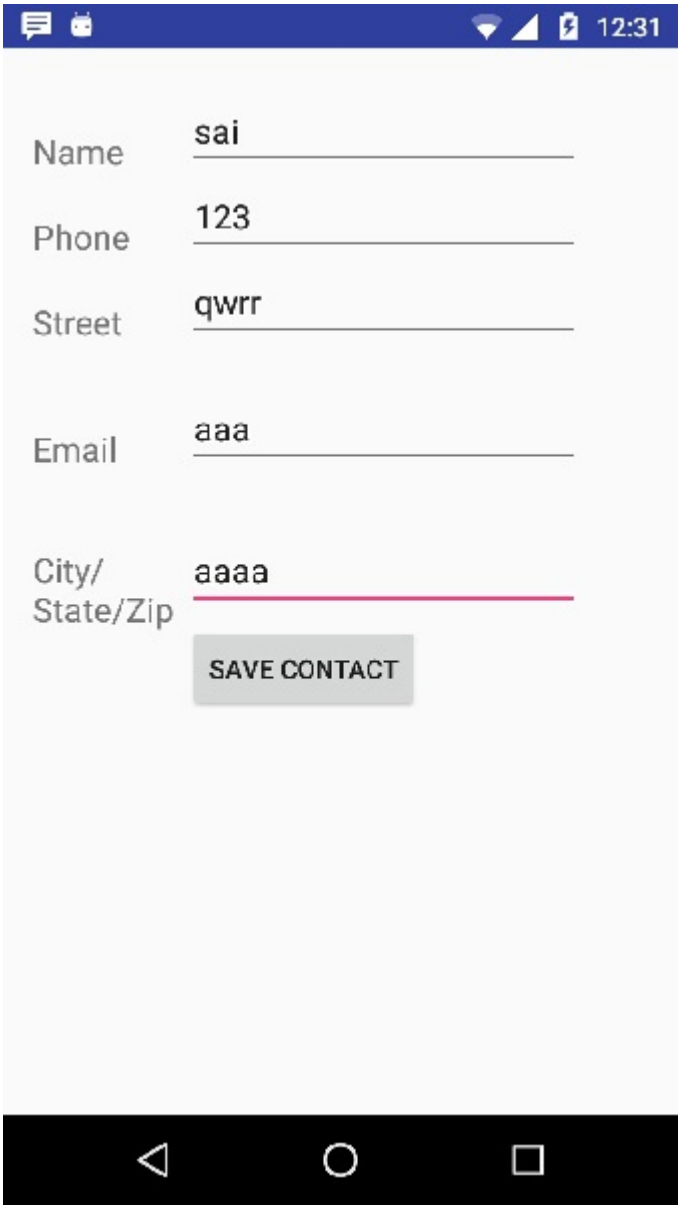
Select your mobile device as an option and then check your mobile device which will display following screen −



Now open your optional menu, it will show as below image: **Optional menu appears different places on different versions**



Click on the add button of the menu screen to add a new contact. It will display the following screen –

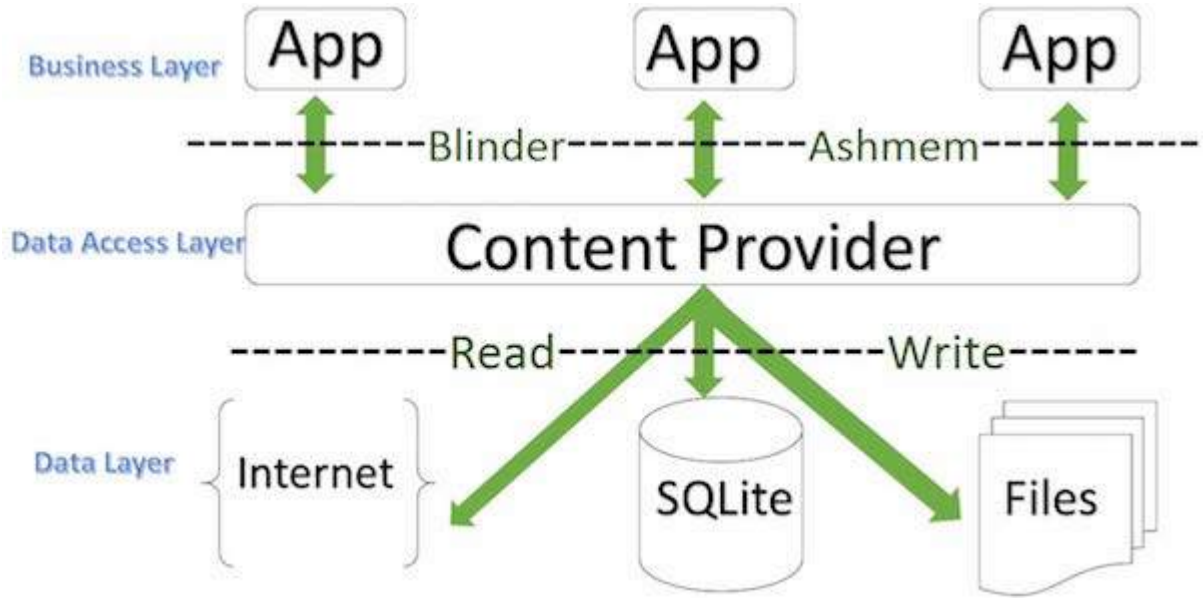


It will display the following fields. Please enter the required information and click on save contact. It will bring you back to main screen.

Now our contact sai has been added.In order to see that where is your database is created. Open your android studio, connect your mobile. Go **tools/android/android device monitor**. Now browse the file explorer tab. Now browse this folder **/data/data/<your.package.name>/databases<database-name>.**

Android - Content Providers

> A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the ContentResolver class. A content provider can use different ways to store its data and the data can be stored in a database, in files, or even over a network.



ContentProvider

> **sometimes it is required to share data across applications. This is where content providers become very useful.**

Content providers let you centralize content in one place and have many different applications access it as needed. A content provider behaves very much like a database where you can query it, edit its content, as well as add or delete content using insert(), update(), delete(), and query() methods. In most cases this data is stored in an **SQlite** database.

A content provider is implemented as a subclass of **ContentProvider** class and must implement a standard set of APIs that enable other applications to perform transactions.

```
public class My Application extends  ContentProvider {
}
```

Content URIs

To query a content provider, you specify the query string in the form of a URI which has following format −

```
<prefix>://<authority>/<data_type>/<id>
```

Here is the detail of various parts of the URI −

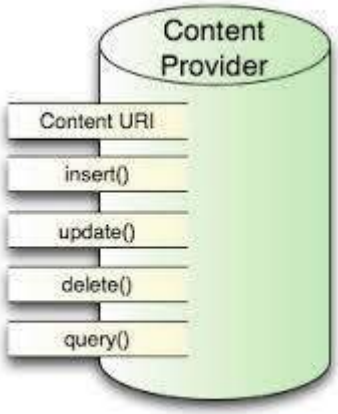
Sr.No	Part & Description
1	**prefix** This is always set to content://
2	**authority** This specifies the name of the content provider, for example *contacts*, *browser* etc. For third-party content providers, this could be the fully qualified name, such as *com.tutorialspoint.statusprovider*
3	**data_type** This indicates the type of data that this particular provider provides. For example, if you are getting all the contacts from the *Contacts* content provider, then the data path would be *people* and URI would look like this*content://contacts/people*
4	**id** This specifies the specific record requested. For example, if you are looking for contact number 5 in the Contacts content provider then URI would look like this *content://contacts/people/5*.

Create Content Provider

This involves number of simple steps to create your own content provider.

- First of all you need to create a Content Provider class that extends the *ContentProviderbaseclass.*
- Second, you need to define your content provider URI address which will be used to access the content.
- Next you will need to create your own database to keep the content. Usually, Android uses SQLite database and framework needs to override *onCreate()* method which will use SQLite Open Helper method to create or open the provider's database. When your application is launched, the *onCreate()* handler of each of its Content Providers is called on the main application thread.
- Next you will have to implement Content Provider queries to perform different database specific operations.
- Finally register your Content Provider in your activity file using <provider> tag.

Here is the list of methods which you need to override in Content Provider class to have your Content Provider working −



ContentProvider

- **onCreate()** This method is called when the provider is started.
- **query()** This method receives a request from a client. The result is returned as a Cursor object.
- **insert()**This method inserts a new record into the content provider.
- **delete()** This method deletes an existing record from the content provider.
- **update()** This method updates an existing record from the content provider.
- **getType()** This method returns the MIME type of the data at the given URI.

Example

This example will explain you how to create your own *ContentProvider*. So let's follow the following steps to similar to what we followed while creating *Hello World Example*−

Step	Description
1	You will use Android StudioIDE to create an Android application and name it as *My Application* under a package *com.example.MyApplication*, with blank Activity.
2	Modify main activity file *MainActivity.java* to add two new methods *onClickAddName()* and *onClickRetrieveStudents()*.
3	Create a new java file called *StudentsProvider.java* under the package *com.example.MyApplication* to define your actual provider and associated methods.
4	Register your content provider in your *AndroidManifest.xml* file using <provider.../> tag
5	Modify the default content of *res/layout/activity_main.xml* file to include a small GUI to add students records.
6	No need to change string.xml.Android studio take care of string.xml file.
7	Run the application to launch Android emulator and verify the result of the changes done in the application.

Following is the content of the modified main activity file **src/com.example.MyApplication/MainActivity.java**. This file can include each of the fundamental life cycle methods. We have added two new methods *onClickAddName()* and *onClickRetrieveStudents()* to handle user interaction with the application.

```
package com.example.MyApplication;

import android.net.Uri;
import android.os.Bundle;
import android.app.Activity;

import android.content.ContentValues;
import android.content.CursorLoader;

import android.database.Cursor;

import android.view.Menu;
import android.view.View;

import android.widget.EditText;
import android.widget.Toast;
```

```
public class MainActivity extends Activity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
   public void onClickAddName(View view) {
      // Add a new student record
      ContentValues values = new ContentValues();
      values.put(StudentsProvider.NAME,
         ((EditText)findViewById(R.id.editText2)).getText().toString());

      values.put(StudentsProvider.GRADE,
         ((EditText)findViewById(R.id.editText3)).getText().toString());


      Uri uri = getContentResolver().insert(
         StudentsProvider.CONTENT_URI, values);

      Toast.makeText(getBaseContext(),
         uri.toString(), Toast.LENGTH_LONG).show();
   }
   public void onClickRetrieveStudents(View view) {
      // Retrieve student records
      String URL = "content://com.example.MyApplication.StudentsProvider";

      Uri students = Uri.parse(URL);
      Cursor c = managedQuery(students, null, null, null, "name");

      if (c.moveToFirst()) {
         do{
            Toast.makeText(this,
               c.getString(c.getColumnIndex(StudentsProvider._ID)) +
                  ", " +  c.getString(c.getColumnIndex( StudentsProvider.NAME)) +
                     ", " + c.getString(c.getColumnIndex( StudentsProvider.GRADE)),
            Toast.LENGTH_SHORT).show();
         } while (c.moveToNext());
      }
   }
}
```

Create new file StudentsProvider.java under *com.example.MyApplication* package and following is the content of **src/com.example.MyApplication/StudentsProvider.java −**

```
package com.example.MyApplication;

import java.util.HashMap;

import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.Context;
import android.content.UriMatcher;

import android.database.Cursor;
import android.database.SQLException;

import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteQueryBuilder;

import android.net.Uri;
import android.text.TextUtils;

public class StudentsProvider extends ContentProvider {
   static final String PROVIDER_NAME = "com.example.MyApplication.StudentsProvider";
   static final String URL = "content://" + PROVIDER_NAME + "/students";
   static final Uri CONTENT_URI = Uri.parse(URL);

   static final String _ID = "_id";
   static final String NAME = "name";
   static final String GRADE = "grade";

   private static HashMap<String, String> STUDENTS_PROJECTION_MAP;

   static final int STUDENTS = 1;
   static final int STUDENT_ID = 2;

   static final UriMatcher uriMatcher;
   static{
      uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
      uriMatcher.addURI(PROVIDER_NAME, "students", STUDENTS);

      uriMatcher.addURI(PROVIDER_NAME, "students/#", STUDENT_ID);
   }
```

```
}

/**
   * Database specific constant declarations
*/

private SQLiteDatabase db;
static final String DATABASE_NAME = "College";
static final String STUDENTS_TABLE_NAME = "students";
static final int DATABASE_VERSION = 1;
static final String CREATE_DB_TABLE =
   " CREATE TABLE " + STUDENTS_TABLE_NAME +
      " (_id INTEGER PRIMARY KEY AUTOINCREMENT, " +
      " name TEXT NOT NULL, " +
      " grade TEXT NOT NULL);";

/**
   * Helper class that actually creates and manages
   * the provider's underlying data repository.
*/

private static class DatabaseHelper extends SQLiteOpenHelper {
   DatabaseHelper(Context context){
      super(context, DATABASE_NAME, null, DATABASE_VERSION);
   }

   @Override
   public void onCreate(SQLiteDatabase db) {
      db.execSQL(CREATE_DB_TABLE);
   }

   @Override
   public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
      db.execSQL("DROP TABLE IF EXISTS " +  STUDENTS_TABLE_NAME);
      onCreate(db);
   }
}

@Override
public boolean onCreate() {
   Context context = getContext();
   DatabaseHelper dbHelper = new DatabaseHelper(context);


   /**
```

```
      /
         * Create a write able database which will trigger its
         * creation if it doesn't already exist.
      */

      db = dbHelper.getWritableDatabase();
      return (db == null)? false:true;
   }

   @Override
   public Uri insert(Uri uri, ContentValues values) {
      /**
         * Add a new student record
      */
      long rowID = db.insert(   STUDENTS_TABLE_NAME, "", values);

      /**
         * If record is added successfully
      */
      if (rowID > 0) {
         Uri _uri = ContentUris.withAppendedId(CONTENT_URI, rowID);
         getContext().getContentResolver().notifyChange(_uri, null);
         return _uri;
      }

      throw new SQLException("Failed to add a record into " + uri);
   }

   @Override
   public Cursor query(Uri uri, String[] projection,
      String selection,String[] selectionArgs, String sortOrder) {
      SQLiteQueryBuilder qb = new SQLiteQueryBuilder();
      qb.setTables(STUDENTS_TABLE_NAME);

      switch (uriMatcher.match(uri)) {
         case STUDENTS:
            qb.setProjectionMap(STUDENTS_PROJECTION_MAP);
         break;

         case STUDENT_ID:
            qb.appendWhere( _ID + "=" + uri.getPathSegments().get(1));
         break;


         default:
```

```
            default:
         }

         if (sortOrder == null || sortOrder == ""){
            /**
               * By default sort on student names
            */
            sortOrder = NAME;
         }

         Cursor c = qb.query(db,   projection,     selection,
            selectionArgs,null, null, sortOrder);
         /**
            * register to watch a content URI for changes
         */
         c.setNotificationUri(getContext().getContentResolver(), uri);
         return c;
      }

      @Override
      public int delete(Uri uri, String selection, String[] selectionArgs) {
         int count = 0;
         switch (uriMatcher.match(uri)){
            case STUDENTS:
               count = db.delete(STUDENTS_TABLE_NAME, selection, selectionArgs);
            break;

            case STUDENT_ID:
               String id = uri.getPathSegments().get(1);
               count = db.delete( STUDENTS_TABLE_NAME, _ID +  " = " + id +
                  (!TextUtils.isEmpty(selection) ? "
                  AND (" + selection + ')' : ""), selectionArgs);
               break;
            default:
               throw new IllegalArgumentException("Unknown URI " + uri);
         }

         getContext().getContentResolver().notifyChange(uri, null);
         return count;
      }

      @Override

      public int update(Uri uri, ContentValues values,
         String selection, String[] selectionArgs) {
```

```
        String selection, String[] selectionArgs) {
        int count = 0;
        switch (uriMatcher.match(uri)) {
            case STUDENTS:
                count = db.update(STUDENTS_TABLE_NAME, values, selection, selectionArgs);
            break;

            case STUDENT_ID:
                count = db.update(STUDENTS_TABLE_NAME, values,
                    _ID + " = " + uri.getPathSegments().get(1) +
                    (!TextUtils.isEmpty(selection) ? "
                    AND (" +selection + ')' : ""), selectionArgs);
                break;
            default:
                throw new IllegalArgumentException("Unknown URI " + uri );
        }

        getContext().getContentResolver().notifyChange(uri, null);
        return count;
    }

    @Override
    public String getType(Uri uri) {
        switch (uriMatcher.match(uri)){
            /**
             * Get all student records
             */
            case STUDENTS:
                return "vnd.android.cursor.dir/vnd.example.students";
            /**
             * Get a particular student
             */
            case STUDENT_ID:
                return "vnd.android.cursor.item/vnd.example.students";
            default:
                throw new IllegalArgumentException("Unsupported URI: " + uri);
        }
    }
}
```

Following will the modified content of *AndroidManifest.xml* file. Here we have added <provider.../> tag to include our content provider:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.MyApplication">

   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:supportsRtl="true"
      android:theme="@style/AppTheme">
         <activity android:name=".MainActivity">
            <intent-filter>
               <action android:name="android.intent.action.MAIN" />
               <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
         </activity>

      <provider android:name="StudentsProvider"
         android:authorities="com.example.MyApplication.StudentsProvider"/>
   </application>
</manifest>
```

Following will be the content of **res/layout/activity_main.xml** file−

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context="com.example.MyApplication.MainActivity">

   <TextView
      android:id="@+id/textView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Content provider"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
```

```
      android:textSize="30dp" />

   <TextView
      android:id="@+id/textView2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Tutorials point "
      android:textColor="#ff87ff09"
      android:textSize="30dp"
      android:layout_below="@+id/textView1"
      android:layout_centerHorizontal="true" />

   <ImageButton
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/imageButton"
      android:src="@drawable/abc"
      android:layout_below="@+id/textView2"
      android:layout_centerHorizontal="true" />

   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/button2"
      android:text="Add Name"
      android:layout_below="@+id/editText3"
      android:layout_alignRight="@+id/textView2"
      android:layout_alignEnd="@+id/textView2"
      android:layout_alignLeft="@+id/textView2"
      android:layout_alignStart="@+id/textView2"
      android:onClick="onClickAddName"/>

   <EditText
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/editText"
      android:layout_below="@+id/imageButton"
      android:layout_alignRight="@+id/imageButton"
      android:layout_alignEnd="@+id/imageButton" />

   <EditText
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
```

```
      android:id="@+id/editText2"
      android:layout_alignTop="@+id/editText"
      android:layout_alignLeft="@+id/textView1"
      android:layout_alignStart="@+id/textView1"
      android:layout_alignRight="@+id/textView1"
      android:layout_alignEnd="@+id/textView1"
      android:hint="Name"
      android:textColorHint="@android:color/holo_blue_light" />

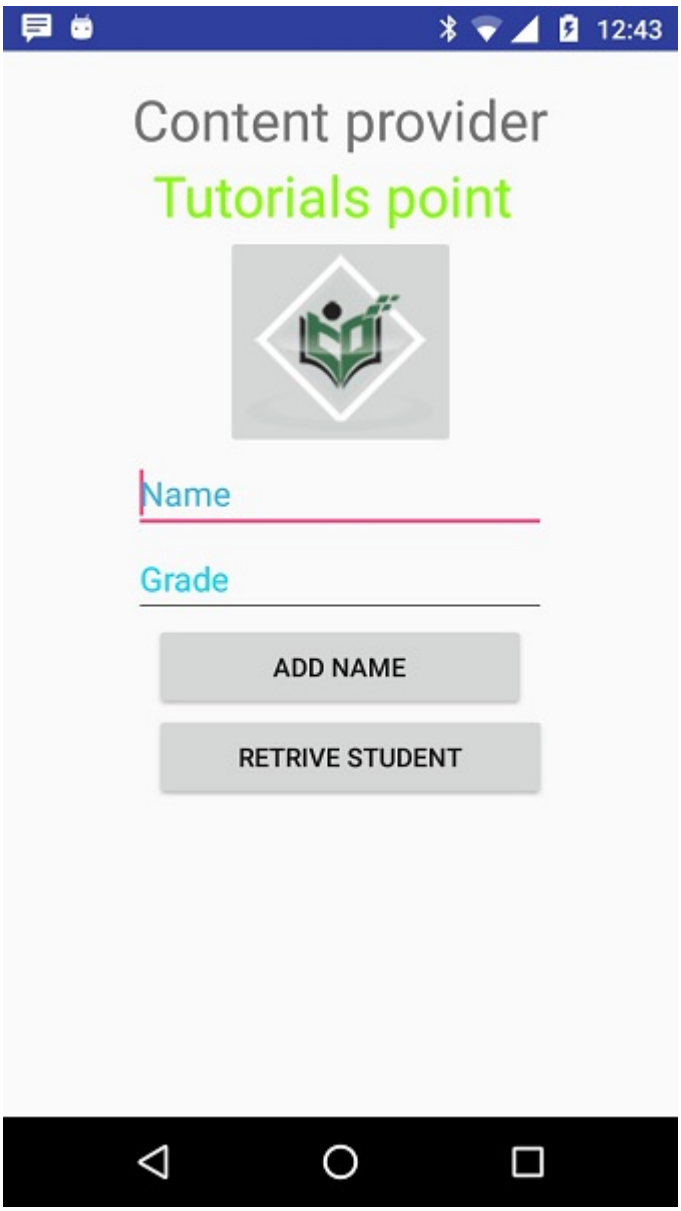
   <EditText
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/editText3"
      android:layout_below="@+id/editText"
      android:layout_alignLeft="@+id/editText2"
      android:layout_alignStart="@+id/editText2"
      android:layout_alignRight="@+id/editText2"
      android:layout_alignEnd="@+id/editText2"
      android:hint="Grade"
      android:textColorHint="@android:color/holo_blue_bright" />

   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Retrive student"
      android:id="@+id/button"
      android:layout_below="@+id/button2"
      android:layout_alignRight="@+id/editText3"
      android:layout_alignEnd="@+id/editText3"
      android:layout_alignLeft="@+id/button2"
      android:layout_alignStart="@+id/button2"
      android:onClick="onClickRetrieveStudents"/>
</RelativeLayout>
```

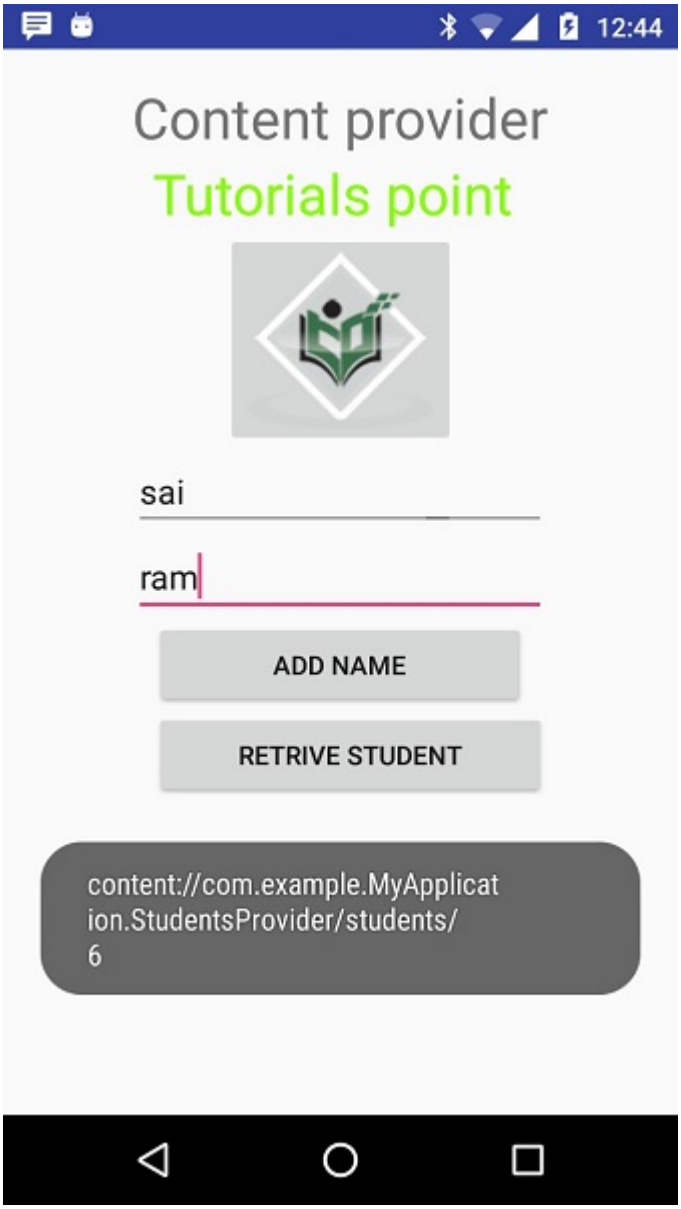
Make sure you have following content of **res/values/strings.xml** file:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">My Application</string>
</resources>;
```

Let's try to run our modified **My Application** application we just created. I assume you had created your **AVD** while doing environment set-up. To run the app from Android Studio IDE, open one of your project's activity files and click Run icon from the tool bar. Android Studio installs the app on your AVD and starts it and if everything is fine with your set-up and application, it will display following Emulator window, be patience because it may take sometime based on your computer speed −



Now let's enter student **Name** and **Grade** and finally click on **Add Name** button, this will add student record in the database and will flash a message at the bottom showing ContentProvider URI along with record number added in the database. This operation makes use of our **insert()** method. Let's repeat this process to add few more students in the database of our content provider.



Once you are done with adding records in the database, now its time to ask ContentProvider to give us those records back, so let's click **Retrieve Students** button which will fetch and display all the records one by one which is as per our the implementation of our **query()** method.

You can write activities against update and delete operations by providing callback functions in **MainActivity.java** file and then modify user interface to have buttons for update and deleted operations in the same way as we have done for add and read operations.

This way you can use existing Content Provider like Address Book or you can use Content Provider concept in developing nice database oriented applications where you can perform all sort of database operations like read, write, update and delete as explained above in the example.

MOBILE APPLICATION DEVELOPMENT

SUBJECT NAME : MOBILE APPLICATION DEVELOPMENT
SUBJRECT CODE : CCS51
CLASS : III BSC CS
SEMESTER : ODD

UNIT IV: ANDROID SERVICES & NETWORK ENVIRONMENT

INDEX:

Services: Introduction to services – Local service – Remote service –

Binding the service –Communication between service and activity –Intent Service – Multi–Threading: Handlers – AsyncTask– Android network programming: HttpUrlConnection– Connecting to REST– based –SOAP based Web services –Broad cast receivers : Local Broadcast Manager–Dynamic broadcast receiver – System Broadcast –Telephony Manager : Sending SMS and making calls.

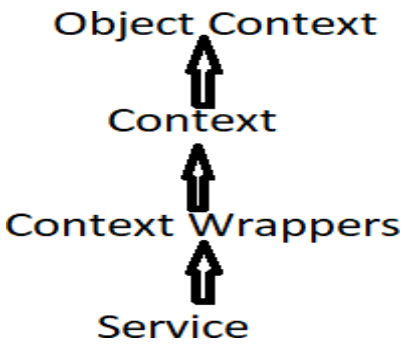
Introduction to services

What is Service?

Service is a component which keeps an **app running in the background** to perform long-running operations based on our requirements.

For Service, **we don't have any user interface** and **it will run the apps in the background** like playing the music in the background or handle network operations when the user in a different app, interacting content providers.

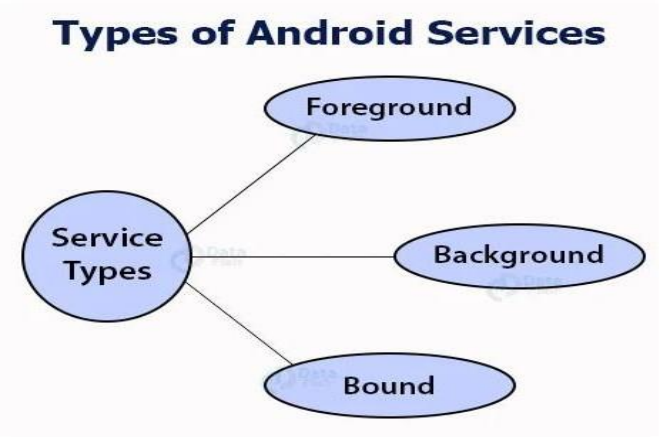
The android.app.Service is subclass of Context Wrapper class.



Characteristics of services

- Started with an Intent.
- Can stay running when user switches applications.
- Lifecycle—which you must manage.
- Other apps can use the service—manage permissions.

Types of Android Services



1. **Foreground Services:**

Foreground services are those services that are visible to the users.

The users can interact with them at easily and track what's

happening.

These services continue to run even when users are using otherapplications.

Foreground services **have own lifecycle** that independent fromactivity or fragment that they were created.

Example

-**Music Player App**(notification of the current song)
-**Fitness App**(show the distance that user has traveled)

2. Background Services:

These services run in the background, such that the user can't see or access them.

These are the tasks that don't need the user to know them.

Background services **have own lifecycle** that independent fromactivity or fragment that they were created.

Example

-Syncing and Storing data

3. Bound Services

This type of android service allows the components of the application like activity or Fragments to bound themselves with it.

Bound services perform their task as long as any applicationcomponent is bound to it.

Many components can bind to one service at a time, but once theyall unbind, the service will destroy.

In order to bind an application component with a service

bindservice () method is used.

Bound services have **no own lifecycle**.That's why, they use the **lifecycle of the activity or fragment they were bounded**.

Lifecycle of Android Services

The life cycle of service will follow two different paths

MOBILE APPLICATION DEVELOPMENT

1. Started Service

2. Bound Service

1. Started Service

A service is **Started** when an application component, such as an activity calls **startService()** method.

Once it started, it will run indefinitely in background even if the component that started is destroyed.

This service can be stopped only in one of the two cases:

- By using the **stopService()** method.
- By stopping itself using the **stopSelf()** method.

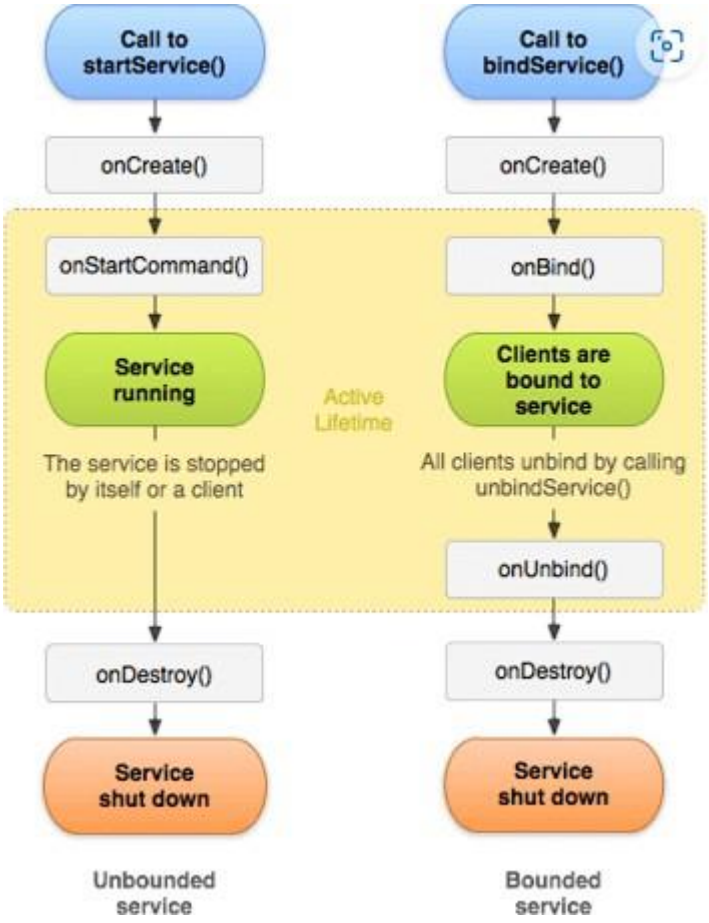
2. Bound Service

A service is **bound** when an application component binds to it bycalling bindService*()*.

A bound service offers a client-server interface that allows components to interact with the service, send requests, get results, andeven do so across processes with interprocess communication (IPC).

This service runs in the background as long as another application isbound to it.

Or it can be unbound according to our requirement by usingthe **unbindService()** method.



Methods of Android Services

The service base class defines certain callback methods to performoperations on applications.

The following are a few important methods of Android services :

- onStartCommand()
- onBind()
- onCreate()
- onUnbind()
- onDestroy()
- onRebind()

S.No.	Callback & Description
1	**onStartCommand()** The system calls this method when another component, such as an activity, requests that the service be started, by calling *startService()*. If you implement this method, it is your responsibility to stop the service when its work is done, by calling *stopSelf()* or *stopService()* methods.
2	**onBind()** The system calls this method when another component wants to bind with the service by calling *bindService()*. If you implement this method, you must provide an interface that clients use to communicate with the service, by returning an *IBinder* object. You must always implement this method, but if you don't want to allow binding, then you should return *null*.
3	**onUnbind()** The system calls this method when all clients have disconnected from a particular interface published by the service.
4	**onRebind()** The system calls this method when new clients have connected to the service, after it had previously been notified that all had disconnected in its *onUnbind(Intent)*.
5	**onCreate()** The system calls this method when the service is first created using *onStartCommand()* or *onBind()*.

	This call is required to perform one-time set-up.
6	**onDestroy()** The system calls this method when the service is no longer used and isbeing destroyed. Your service should implement this to clean up any resources such asthreads, registered listeners, receivers, etc.

Communication between Activity and Service in Android

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
android:layout_height="match_parent" tools:context="MainActivity">
<Button android:id="@+id/buttonStart" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_alignParentTop="true"
android:layout_centerHorizontal="true" android:layout_marginTop="74dp" android:text="Start
Service" />
<Button android:id="@+id/buttonStop" android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true" android:layout_centerVertical="true" android:text="Stop
Service" />
</RelativeLayout>
```

Service class:(*File: MyService.java*)

```
import android.app.Service; import android.content.Intent;
import android.media.MediaPlayer;import android.os.IBinder;
import android.support.annotation.Nullable;import android.widget.Toast;
public class MyService extends Service {MediaPlayer myPlayer;
@Nullable@Override
public IBinder onBind(Intent intent) {return null;
}
@Override
public void onCreate() { Toast.makeText(this, "Service Created",Toast.LENGTH_LONG).show();
myPlayer = MediaPlayer.create(this, R.raw.song);myPlayer.setLooping(false);
```

```
}
@Override
public void onStart(Intent intent, int startId) { Toast.makeText(this, "Service Started", Toast.LENGTH_LONG).show(); myPlayer.start();
}
@Override
public void onDestroy() {
Toast.makeText(this, "Service Stopped",Toast.LENGTH_LONG).show(); myPlayer.stop();
}
}
```

MainActivity.java

```
import android.content.Intent; import android.os.Bundle; import android.view.View; import android.widget.Button;
import android.support.v7.app.AppCompatActivity;
public class MainActivity extends AppCompatActivity implementsView.OnClickListener {
Button buttonStart, buttonStop;@Override
public void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity_main); buttonStart = findViewById(R.id.buttonStart); buttonStop = findViewById(R.id.buttonStop); buttonStart.setOnClickListener(this); buttonStop.setOnClickListener(this);
}
public void onClick(View src) {switch (src.getId()) {
case R.id.buttonStart:
  startService(new Intent(this, MyService.class));break;
case R.id.buttonStop:
  stopService(new Intent(this, MyService.class));break;
}
}}
```

AndroidManifest.xml:

In the AndroidManifest.xml file, we will declare the service.

```
<service
android:name=".MyService" android:enabled="true" android:exported="true"></service>
```

Output:

Intent services

The **Service** is the base class for the **IntentService**.

It uses “work queue process” pattern where
the **IntentService** handles the on-demand requests (expressed as Intents)of clients.

The Service will be started whenever a client sends a request, and it will be stopped after each Purpose has been addressed.

Clients can send the request to start a Service byusing **Context.startService(Intent)**.

Here, a worker thread is created and all requests are handled usingthe worker thread but at a time, only one request will be processed.

To use IntentService you have to extend the **IntentService** and implement the **onHandleIntent(android.content.Intent)**.

Features of Intent Services

1. **Thread management**: It automatically processes all incoming requestsin a separate thread taking the burden of thread management away fromthe developer.

2. **Request Queue**: It queues up all the incoming requests and they areprocessed one by one

3. Stop point: Once the request Queue is empty it automatically stops itself, so the developer need not worry about handling the service lifecycle.

Difference Between Service and IntentService

Service	**IntentService**
You can use Service for the tasks that don't require any UI and also it is not a very long running task.	If you want some background task to be performed for a very long period oftime, then you should use the IntentService.
To start a Service you need to call the **onStartService()** method	To start the IntentService by calling **Context.startService(Intent)**
Service always runs on the **Mainthread**	The IntentService runs on a separate **Worker thread**
There is a chance of blocking themain thread	Tasks will be performed on a queue basis i.e,First come first serve basis.
Easy to interact with UI of theapplication	Difficult to interact with UI of theapplication
To stop service we have to use **stopService()** or **stopSelf()**	No need to stop the service, it will stop automatically.

Multi–Threading

Working on multiple tasks at the same time is **Multitasking**.

Multiple threads running at the same time in a machine iscalled **Multi-Threading**.

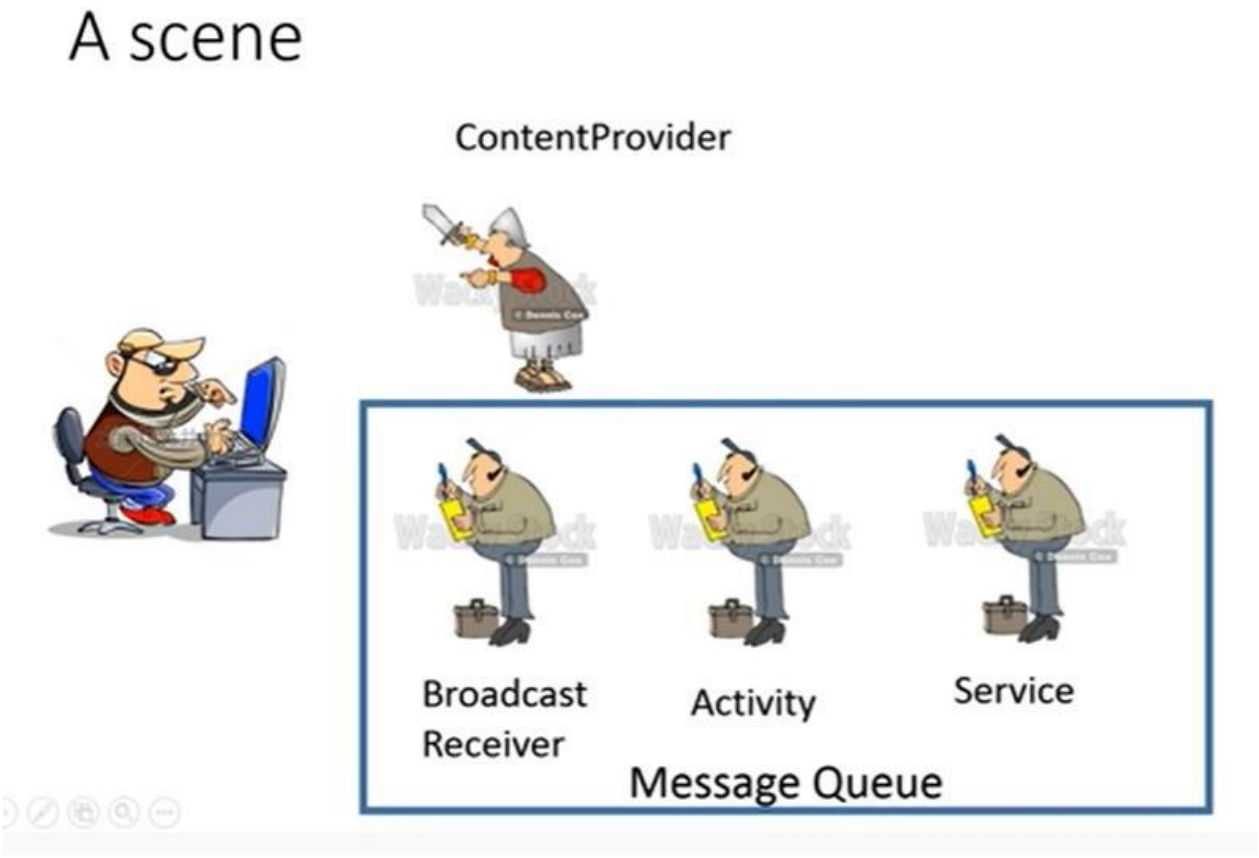
A thread is a unit of a process. Multiple such threads combine toform a process.

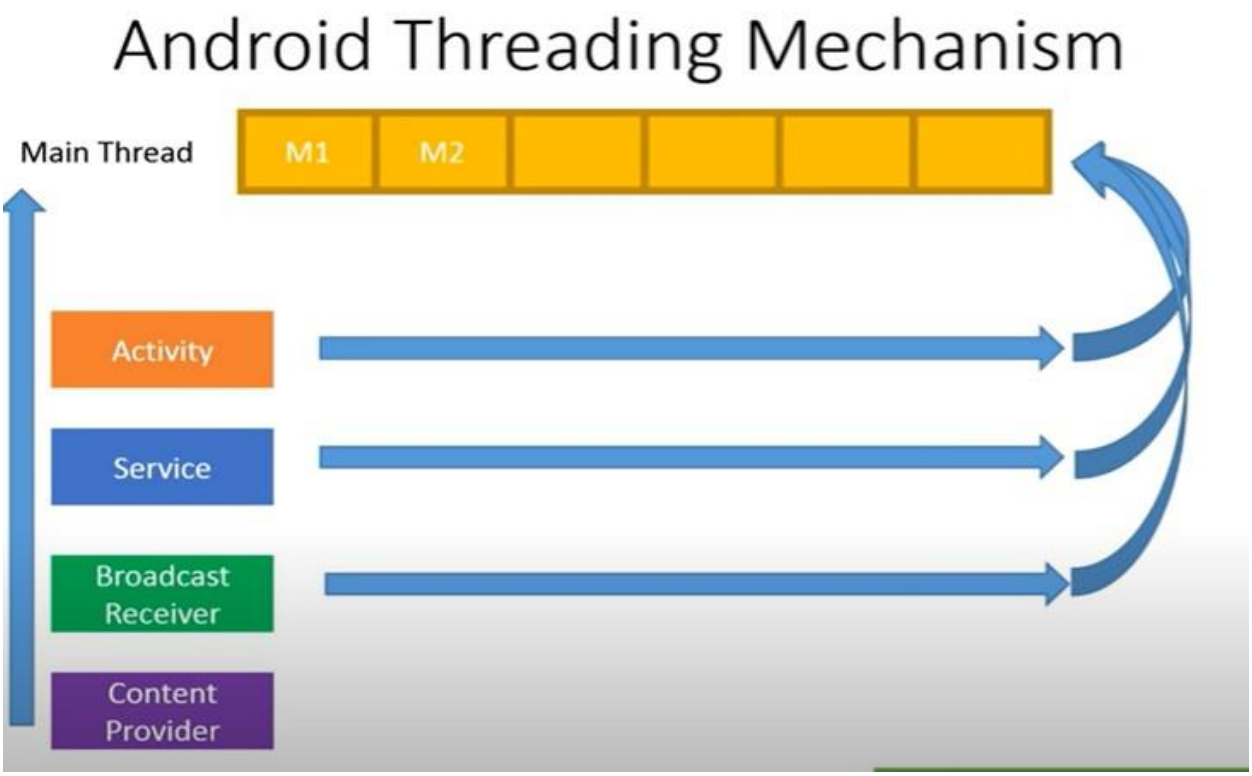
This means when a process is broken, the equivalent number ofthreads are available.

Threading in Android

In Android, you can categorize all threading components into two basiccategories:

1. **Threads that *are* attached to an activity/fragment:** These threads are tied to the lifecycle of the activity/fragment and are terminated assoon as the activity/fragment is destroyed.
2. **Threads that *are not* attached to any activity/fragment:** Thesethreads can continue to run beyond the lifetime of the activity/fragment (if any) from which they were spawned.





Handler in android

A Handler is a threading class defined in thethrough which `android.os` package we can send and

process and objects associated with a thread's `Message` `Runnable` .

Uses of Handler

In android Handler is mainly used to update the main thread from background thread or other than main thread. There are two methods are inhandler.

- **Post()** − it going to post message from background thread to main thread using looper.
- **sendmessage()** − if you want to organize what you have sent toui (message from background thread) or ui functions. you shoulduse sendMessage().

Handler is a class fundamental to how we do threading in android, atthe infrastructure level. It works hand in hand with the Looper.

`Looper` care of dispatching work on its message-loop thread. On the other hand `Handler` will serve two roles:

1. First it will provide an interface to submit messages to itsLooper queue.
2. Secondly it will implement the callback for processing thosemessages when they are dispatched by the Looper

Example for Handler

Write Ex no 6

AsyncTask

Android AsyncTask is an abstract class provided by Android which gives us the liberty to perform heavy tasks in the background and keep theUI thread light thus making the application more responsive.

Android application runs on a single thread when launched.

Due to this single thread model tasks that take longer time to fetchthe response can make the application non-responsive.

To avoid this we use android AsyncTask to perform the heavy tasks in background on a dedicated thread and passing the results back to the UI thread.

Hence use of AsyncTask in android application keeps the UI threadresponsive at all times.

The basic methods used in an android AsyncTask class are definedbelow :

- **doInBackground()** : This method contains the code which needs to be executed in background. In this method we can send results

  multiple times to the UI thread by publishProgress() method. To notify that the background processing has been completed we justneed to use the return statements
- **onPreExecute()** : This method contains the code which is executedbefore the background processing starts
- **onPostExecute()** : This method is called after doInBackground method completes processing. Result from doInBackground is passedto this method
- **onProgressUpdate()** : This method receives progress updates from doInBackground method, which is published via publishProgress method, and this method can use this progress update to update the UI thread

Generic Types in Async Task

- **TypeOfVarArgParams** − It contains information about what type of params used for execution.
- **ProgressValue** − It contains information about progress units. While doing background operation we can update information on ui using onProgressUpdate().
- **ResultValue** −It contains information about result type.

AsyncTask Example

The following code must be present in the MainActivity class in order to launch an AsyncTask.

MyTask myTask = new MyTask();myTask.execute();

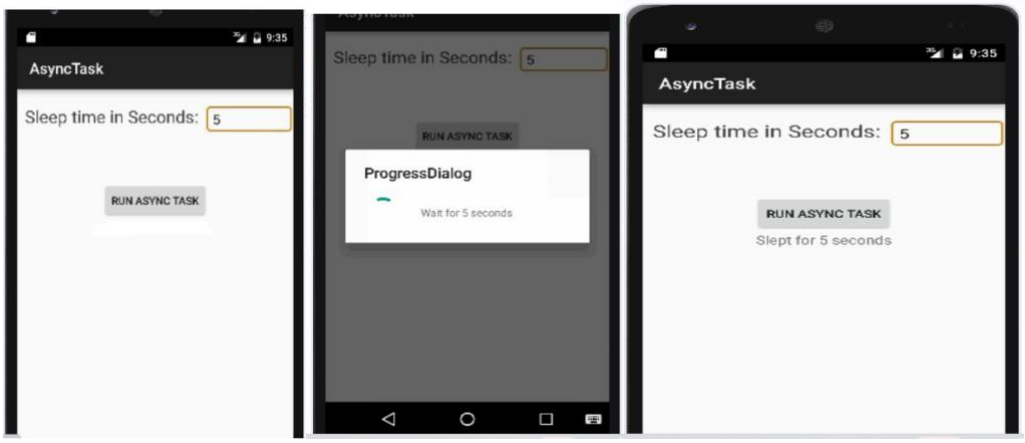
The execute method is used to start the background thread in theexcerpt above, where we've used a sample class name that extends AsyncTask.

public class MainActivity extends AppCompatActivity {@Override protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity_main); ExampleAsync task = new ExampleAsync; task.execute(5);

```
}
private static class ExampleAsync extends AsyncTask<Integer, Integer,String> {

@Override
protected void onPreExecute() {super.onPreExecute();
// ...

}
@Override
protected String doInBackground(Integer... integers) {
// ...
return "Finished!"; }@Override
protected void onProgressUpdate(Integer... values) {super.onProgressUpdate(values);
// ...
}
@Override
protected void onPostExecute(String string) {super.onPostExecute(string);
// ...
}
}
}
```



Broad cast receivers

Broad cast

Broadcasts are messages that the Android system and Android appssend when events occur that might affect the functionality of other apps.

For example, the Android system sends an event when the systemboots up, when power is connected or disconnected, and when headphones are connected or disconnected.

Broadcasts are messaging components used for communicatingacross apps when events of interest occur.

There are two types of broadcasts:

- *System broadcasts* are delivered by the system.
- *Custom broadcasts* are delivered by your app.

System broadcasts

A *system broadcast* is a message that the Android system sendswhen a system event occurs. System broadcasts are wrapped in `Intent` objects.

The intent object's action field contains event details such

as `android.intent.action.HEADSET_PLUG`, which is sent when a wired headset is connected or disconnected.

The intent can also contain more data about the event in its extra field, for example a `boolean` extra indicating whether a headset is connected or disconnected.Examples:

- When the device boots, the system broadcasts a system `Intent` with the action `ACTION_BOOT_COMPLETED`.
- When the device is disconnected from external power, the systemsends a system `Intent` with the action

  field `ACTION_POWER_DISCONNECTED`.

Broadcast receivers

A *Broadcast receiver* (*receiver*) is an Android component which allows us to listens to system-wide broadcast events or intents.

When any of these events occur it brings the application into actionby either creating a status bar notification or performing a task.

For eg : a Broadcast receiver triggers ***battery Low notification thatyou see on your mobile screen***..

Broadcast receivers helps our app communicate with Android OS andother apps.

Two important steps to make Broadcast Receiver works for thesystem broadcasted items:

1. Creating the Broadcast Receiver.
2. Registering Broadcast Receiver.

1. Creating the Broadcast Receiver

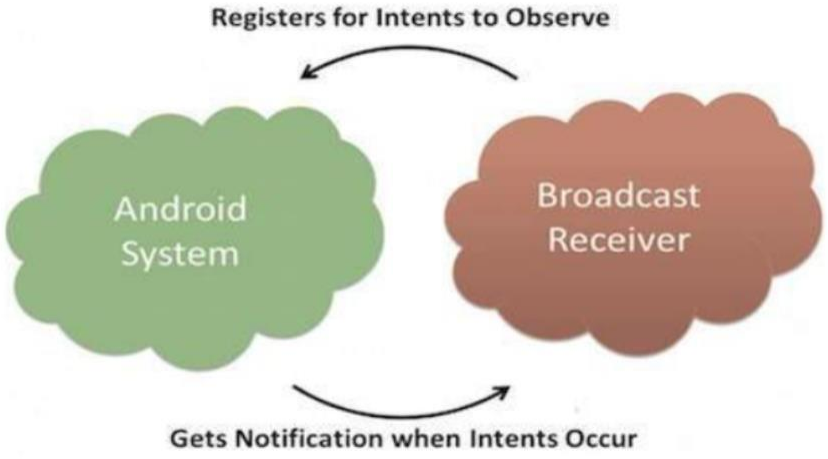
To create a new Broadcast Receiver, we need to create a new classthat extends the Broadcast Receiver class. Then you can override the onReceive function of the class.

Syntax

```
Public class AirplaneModeReceiver extends BroadcastReceiver()
 {
   Override
Public void onReceive(context: Context, intent: Intent)
{
      // body of the function
   }
}
```

2.Registering Broadcast Receiver.

Register the broadcast receiver, either statically or dynamically.



1. Statically (manifest-declared) - This receiver can be registered via theAndroidManifest.xml file.
2. Dynamically (context-registered) - This registers a receiver dynamically via the Context.registerReceiver() method.

System wide generated intents and its Description

Following are some of the important system wide generated intents.

1. **android.intent.action.BATTERY_LOW** : Indicates low batterycondition on the device.
2. **android.intent.action.BOOT_COMPLETED** : This is broadcast once,after the system has finished booting
3. **android.intent.action.CALL** : To perform a call to someone specifiedby the data
4. **android.intent.action.DATE_CHANGED** : The date has changed
5. **android.intent.action.REBOOT** : Have the device reboot
6. **android.net.conn.CONNECTIVITY_CHANGE** : The mobile network or wifi connection is changed(or reset)

1. Static Broadcast Receivers

These types of Receivers are declared in the manifest file and workseven if the app is closed.

Manifest-declared receivers(Statically)

The registration is done in the manifest file, using <**register**> tags.

Firstly, specify the receiver in your manifest file.

```
<receiver android:name=".MyBroadcastReceiver"

android:exported="true">
   <intent-filter>
      <action
android:name="android.net.conn.CONNECTIVITY_CHANGE""/>

   </intent-filter>
</receiver>
```

Secondly, create class as a subclass of BroadcastReceiver class and overriding the onReceive() method where each message is received as aIntent object parameter.

```
public class MyBroadcastReceiver extends BroadcastReceiver
 {
@Override

  public void onReceive(Context context, Intent intent)
{
    Toast.makeText(context, "Intent Detected.", Toast.LENGTH_LONG).show();
  }
}
```

2. Dynamic Broadcast Receivers

These types of receivers work only if the app is active or minimized.

Context-registered receivers(Dynamically)

The registration is done using **Context.registerReceiver()** method.

Firstly, register broadcast receiver programmatically.

```
IntentFilter intentFilter = new IntentFilter();
intentFilter.addAction(getPackageName() +
"android.net.conn.CONNECTIVITY_CHANGE");

MyBroadcastReceiver myReceiver = new MyBroadcastReceiver();
registerReceiver(myReceiver, filter);
```

Secondly, To unregister a broadcast receiver in onStop() or onPause() of theactivity.

Android TelephonyManager

The **android.telephony.TelephonyManager** class provides information about the telephony services such as subscriber id, sim serialnumber, phone network type etc.

activity_main.xml

<RelativeLayout xmlns:androclass="http://schemas.android.com/apk/res/android" xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent" android:layout_height="match_parent" android:paddingBottom="@dimen/activity_vertical_margin" android:paddingLeft="@dimen/activity_horizontal_margin" android:paddingRight="@dimen/activity_horizontal_margin" android:paddingTop="@dimen/activity_vertical_margin" tools:context=".MainActivity" >

<TextView android:id="@+id/textView1" android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_alignParentLeft="true" android:layout_alignParentTop="true" android:layout_marginLeft="38dp" android:layout_marginTop="30dp" android:text="Phone Details:" />

</RelativeLayout>

MainActivity.java

import android.os.Bundle; **import** android.app.Activity; **import** android.content.Context;
import android.telephony.TelephonyManager;
import android.view.Menu;
import android.widget.TextView;

public class MainActivity **extends** Activity {TextView textView1;
@Override
protected void onCreate(Bundle savedInstanceState) { **super**.onCreate(savedInstanceState); setContentView(R.layout.activity_main);

textView1=(TextView)findViewById(R.id.textView1);

//Get the instance of TelephonyManager
TelephonyManager tm=(TelephonyManager)getSystemService(Context.TELEPHONY_SERVICE);

//Calling the methods of TelephonyManager the returns the information
String IMEINumber=tm.getDeviceId();String subscriberID=tm.getDeviceId();
String SIMSerialNumber=tm.getSimSerialNumber(); String networkCountryISO=tm.getNetworkCountryIso();String SIMCountryISO=tm.getSimCountryIso();
String softwareVersion=tm.getDeviceSoftwareVersion(); String voiceMailNumber=tm.getVoiceMailNumber();

```
String strphoneType="";

int phoneType=tm.getPhoneType();


switch (phoneType)
{
case (TelephonyManager.PHONE_TYPE_CDMA):strphoneType="CDMA";
break;
case (TelephonyManager.PHONE_TYPE_GSM):strphoneType="GSM";
break;
case (TelephonyManager.PHONE_TYPE_NONE):strphoneType="NONE";
break;
}
boolean isRoaming=tm.isNetworkRoaming();

String info="Phone Details:\n"; info+="\n IMEI Number:"+IMEINumber; info+="\n SubscriberID:"+subscriberID;
info+="\n Sim Serial Number:"+SIMSerialNumber; info+="\n Network Country ISO:"+networkCountryISO; info+="\n SIM Country ISO:"+SIMCountryISO; info+="\n Software Version:"+softwareVersion; info+="\n Voice Mail Number:"+voiceMailNumber; info+="\n Phone Network Type:"+strphoneType; info+="\n In Roaming? :"+isRoaming;


textView1.setText(info);//displaying the information in the textView
}


}
```

AndroidManifest.xml

```
<usespermission android:name="android.permission.READ_PHONE_STATE"/>
```

Output:

Sending SMS and making callsSending SMS

Android devices can send and receive messages to or from any other phone that supports Short Message Service (SMS).

Android offers the Messenger app that can send and receive SMSmessages.

//Getting intent and Pending Intent instance

Intent intent=**new** Intent(getApplicationContext(),MainActivity.**class**); PendingIntent pi=PendingIntent.getActivity(getApplicationContext(),0, intent,0);

//Get the SmsManager instance and call the sendTextMessage method tosend message

SmsManager sms=SmsManager.getDefault(); sms.sendTextMessage("8802177690", **null**, "hello javatpoint", pi,**null**);

Example of sending sms in android

activity_main.xml

<**RelativeLayout** xmlns:androclass="http://schemas.android.com/apk/res/android" xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent" android:layout_height="match_parent" tools:context=".MainActivity" >

<**EditText** android:id="@+id/editText1" android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_alignParentRight="true" android:layout_alignParentTop="true" android:layout_marginRight="20dp" android:ems="10" />

<**EditText** android:id="@+id/editText2" android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_alignLeft="@+id/editText1" android:layout_below="@+id/editText1" android:layout_marginTop="26dp" android:ems="10" android:inputType="textMultiLine" />

<**TextView** android:id="@+id/textView1" android:layout_width="wrap_content" android:layout_height="wrap_content"
android:layout_alignBaseline="@+id/editText1"
android:layout_alignBottom="@+id/editText1" android:layout_toLeftOf="@+id/editText1" android:text="Mobile No:" />

<**TextView** android:id="@+id/textView2" android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_alignBaseline="@+id/editText2" android:layout_alignBottom="@+id/editText2" android:layout_alignLeft="@+id/textView1" android:text="Message:" />

<**Button**
android:id="@+id/button1" android:layout_width="wrap_content"

android:layout_height="wrap_content" android:layout_alignLeft="@+id/editText2" android:layout_below="@+id/editText2" android:layout_marginLeft="34dp" android:layout_marginTop="48dp" android:text="Send SMS" />

</RelativeLayout>

Write the permission code in Android-Manifest.xml file

You need to write SEND_SMS permission as given below:

<uses-permission android:name="android.permission.SEND_SMS"/>

MainActivity.java

```
package com.example.sendsms;

import android.os.Bundle;
import android.app.Activity; import android.app.PendingIntent;import android.content.Intent;
import android.telephony.SmsManager;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button; import android.widget.EditText;import android.widget.Toast;


public class MainActivity extends Activity {EditText mobileno,message;
    Button sendsms;
    @Override
protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    mobileno=(EditText)findViewById(R.id.editText1);
    message=(EditText)findViewById(R.id.editText2); sendsms=(Button)findViewById(R.id.button1);

//Performing action on button click sendsms.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View arg0) {
    String no=mobileno.getText().toString(); String msg=message.getText().toString();

    //Getting intent and PendingIntent instance
    Intent intent=new Intent(getApplicationContext(),MainActivity.class);

     PendingIntent pi=PendingIntent.getActivity(getApplicationContext(), 0,intent,0);

      //Get the SmsManager instance and call the sendTextMessage methodto send message
    SmsManager sms=SmsManager.getDefault();sms.sendTextMessage(no, null, msg, pi,null);
```

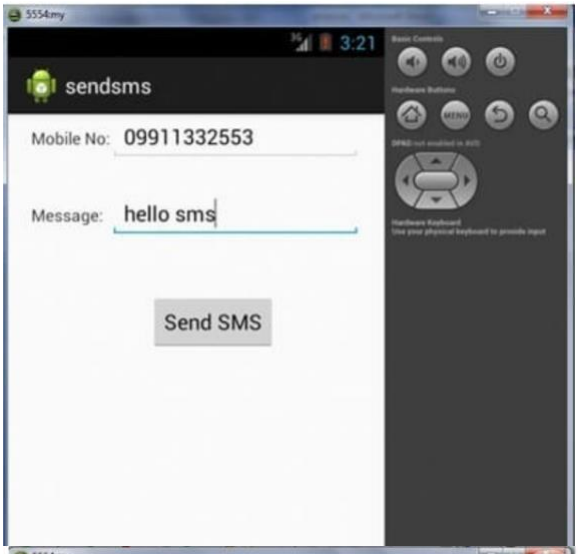
```
Toast.makeText(getApplicationContext(), "Message Sent successfully!",Toast.LENGTH_LONG).show();
        }
        });
        }

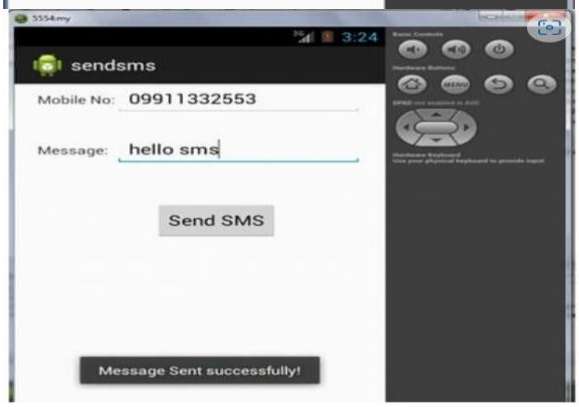
        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.

        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
        }

        }
```

OUTPUT:





Making calls

In android, we can easily make a phone call from our androidapplications by invoking built-in phone calls app
using Intents action (**ACTION_CALL**).

To make a phone call using Intent object in android application, we need to write the code like as shown below.

Intent callIntent = new Intent(Intent.ACTION_CALL); callIntent.setData(Uri.parse("tel:"+8802177690));//change the numberstartActivity(callIntent);

Example of phone call in androidactivity_main.xml

<RelativeLayout xmlns:androclass="http://schemas.android.com/apk/res/android" xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent" android:layout_height="match_parent" tools:context=".MainActivity" >

<Button
android:id="@+id/button1" android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_alignParentTop="true" android:layout_centerHorizontal="true"android:layout_marginTop="118dp" android:text="Call" />

<EditText
android:id="@+id/editText1"
android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_alignParentTop="true" android:layout_centerHorizontal="true" android:layout_marginTop="25dp" android:ems="10" />

</RelativeLayout>

Write the permission code in Android-Manifest.xml file

You need to write CALL_PHONE permission as given below:

<uses-permission android:name="android.permission.CALL_PHONE" />

MainActivity.java

package com.example.phonecall;

import android.net.Uri; **import** android.os.Bundle; **import** android.app.Activity; **import** android.content.Intent;**import** android.view.Menu; **import** android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

```
public class MainActivity extends Activity {EditText edittext1;
    Button button1;@Override
    protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    //Getting the edittext and button instance edittext1=(EditText)findViewById(R.id.editText1);
    button1=(Button)findViewById(R.id.button1);

    //Performing action on button click button1.setOnClickListener(new OnClickListener(){

    @Override
    public void onClick(View arg0) {
    String number=edittext1.getText().toString();
    Intent callIntent = new Intent(Intent.ACTION_CALL);callIntent.setData(Uri.parse("tel:"+number));
    startActivity(callIntent);
    }


    });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.

    getMenuInflater().inflate(R.menu.activity_main, menu);
    return true;
    }

    }
```

OUTPUT:


5554:my
2:02
phonecall
09953038949
Call


5554:my
2:02
09953038949
0:09

MOBILE APPLICATION DEVELOPMENT

SUBJECT NAME : MOBILE APPLICATION DEVELOPMENT SUBJRECT CODE : CCS51
CLASS : III BSC CS
SEMESTER : ODD

UNIT-V

ADVANCED APPLICATION
Loction based services - android google map - property animation overview Android developers - view animation android developers - animate drawable Graphics android developers - multimedia software - android camera tutorial - Environment sensors android developers.

Content

1. Location based services

You must have used apps like **Google Maps, Waze, MapQuest**, etc. These are the applications that help you track the location of the device. They also provide services like finding nearby restaurants, hospitals, petrol pumps, etc. Even the cab drivers now use maps to locate their route.

As technology is emerging, it becomes quite essential for an android developer to learn about location-based services. Through this article, you will understand the various sections of location-based services(LBS). After understanding the concepts, you will also see an implementation of the same.

1.1 What are Android Location-Based Services?

Location-Based Services(LBS) are present in Android to provide you with features like current location detection, display of nearby places, geofencing, etc. It fetches the location using your device's GPS, Wifi, or Cellular Networks.
To build an app with location-based services, you need to access the Google Play Services Module. After that, you need to use a framework called Location Framework, which has many methods, classes, and interfaces to make your task easier.

1.2 Components of Location-Based Services in Android

The classes and the interfaces present in the Location Framework acts as the essential components for LBS. Those components are as follows:

- **LocationManager Class –** It is used to get Location Service access from the system.
- **LocationListener Interface –** It receives updates from the Location Manager class.
- **LocationProvider –** It is the class that provides us with the location for our devices.
- **Location Class –** Its objects carry information about the location. The information includes latitude, longitude, accuracy, altitude, and speed.

1.3 Location Object

The location objects carry the location of your device. The place is in the form of latitude and longitude. On the location object, you can apply the below methods. The below methods help you to get location and other information regarding the location.

1. float distanceTo(Location destination)

It gives the approximate distance between our current location and the destination location.

2. float getAccuracy()

It gives us the accuracy of our location in metres.

3. double getAltitude()

It gives us the altitude of our place above sea level.

4. double getLatitude()

It gives the latitude coordinate of our place in degrees.

5. double getLongitude()

It gives the latitude coordinate of our place in degrees.
6. float getSpeed()
It gives the speed of our location change.
7. void setAccuracy(float accuracy)
Using setAccuracy(), you can set your custom accuracy in metres.
8. void setAltitude(double altitude)
Using setAltitude(), you can set the altitude of your place from sea level in metres.
9. void setBearing(float bearing)
Using the setBearing() method, you can set location bearing in degrees.
10. void setLatitude(double Latitude)
You can even set your location to some other latitude using the setLatitude() method.
11. void setLongitude(double longitude)
You can even set your location to some other longitude using the setLongitude() method.
12. void setSpeed(float speed)
You can even set speed using the setSpeed() method.
13. void reset()
It is used to reset your set location.
14. boolean hasAccuracy()
It says whether or not the location is accurate.
15. boolean hasAltitude()
It says if the place has an altitude or not
16. boolean hasSpeed()
It is true if the place has a speed attached.
17. boolean hasBearing()
It returns true if the place has a bearing or not.

1.4 Location Quality of Service

Quality of Service(QoS) is enabled through the location request object. Location request object requests the proper and accurate location. You can find below the methods that help in the process.

- **setPriority(int priority) –** It allows us to mark the request with priorities. The higher priority request is executed first.
- **setInterval(long millisecond) –** It allows us to set the intervals on which we seek the location updates.
- **setExpirationDuration(long millisecond) –** It allows us to set the duration of the request after which it shall stop.
- **setExpirationTime(long millisecond) –** It allows us to decide the expiration time of our request.

- **setNumUpdates(int number) –** It allows us to set the number of updates we require for a particular place.

2.Android Google Map

Android provides facility to integrate Google map in our application. Google map displays your current location, navigate location direction, search location etc. We can also customize Google map according to our requirement.

2.1 Types of Google Maps

There are four different types of Google maps, as well as an optional to no map at all. Each of them gives different view on map. These maps are as follow:

1. **Normal:** This type of map displays typical road map, natural features like river and some features build by humans.
2. **Hybrid:** This type of map displays satellite photograph data with typical road maps. It also displays road and feature labels.
3. **Satellite:** Satellite type displays satellite photograph data, but doesn't display road and feature labels.
4. **Terrain:** This type displays photographic data. This includes colors, contour lines and labels and perspective shading.
5. **None:** This type displays an empty grid with no tiles loaded.

2.2 Syntax of different types of map

1. googleMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
2. googleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
3. googleMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
4. googleMap.setMapType(GoogleMap.MAP_TYPE_TERRAIN);

2.3 Methods of Google map

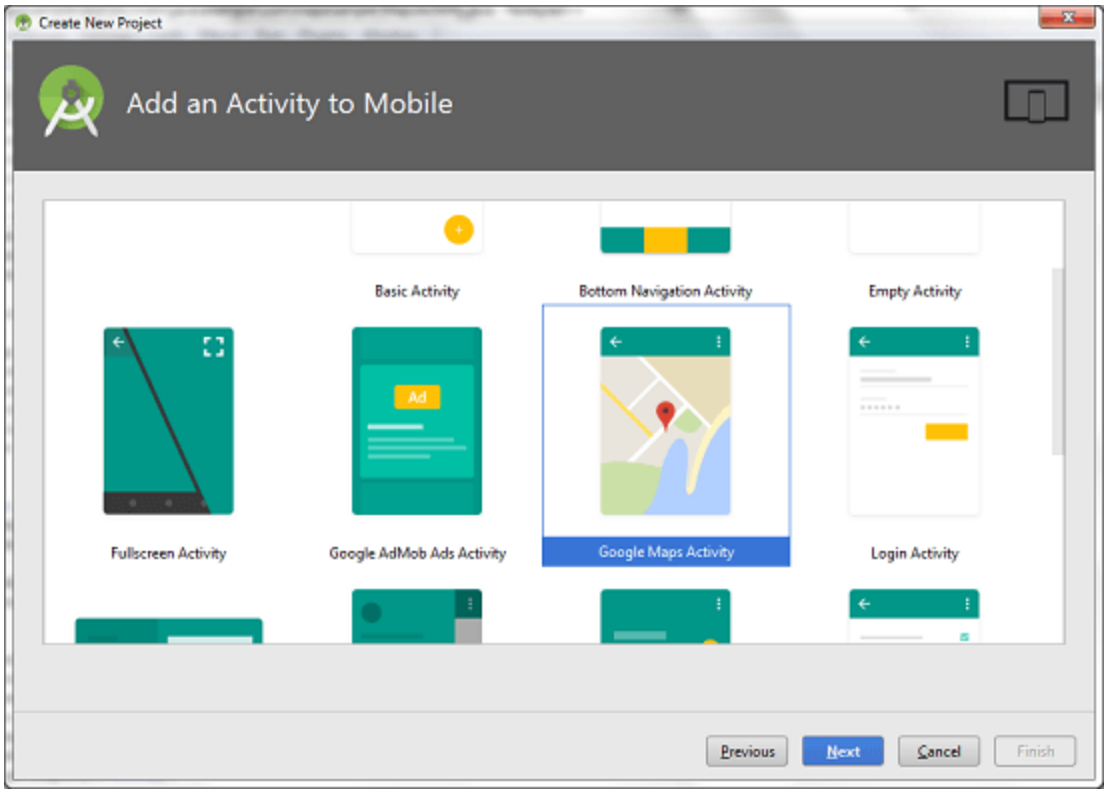
Google map API provides several methods that help to customize Google map. These methods are as following:

Methods	Description
addCircle(CircleOptions options)	This method add circle to map.
addPolygon(PolygonOptions options)	This method add polygon to map.
addTileOverlay(TileOverlayOptions options)	This method add tile overlay to the map.
animateCamera(CameraUpdate update)	This method moves the map according to the update with an animation.

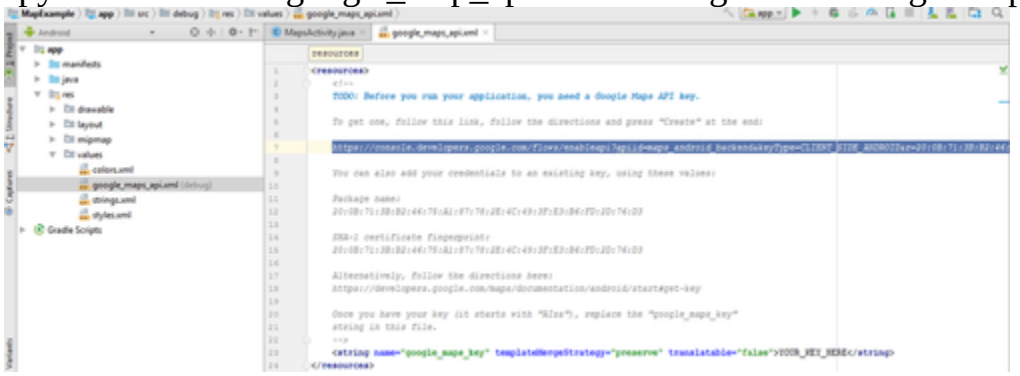
clear()	This method removes everything from the map.
getMyLocation()	This method returns the currently displayed user location.
moveCamera(CameraUpdate update)	This method reposition the camera according to the instructions defined in the update.
setTrafficEnabled(boolean enabled)	This method set the traffic layer on or off.
snapshot(GoogleMap.SnapshotReadyCallback callback)	This method takes a snapshot of the map.
stopAnimation()	This method stops the camera animation if there is any progress.

2.4 Example of Google Map

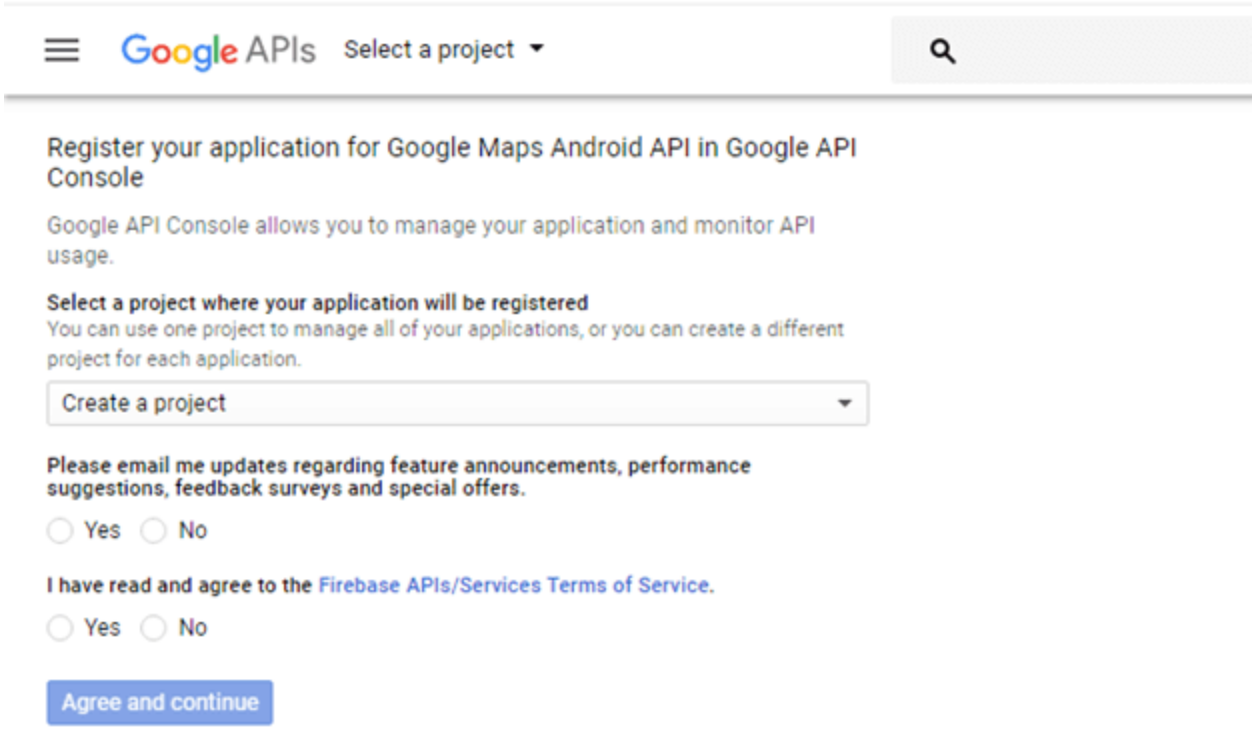
Let's create an example of Google map integrating within our app. For doing this we select Google Maps Activity.



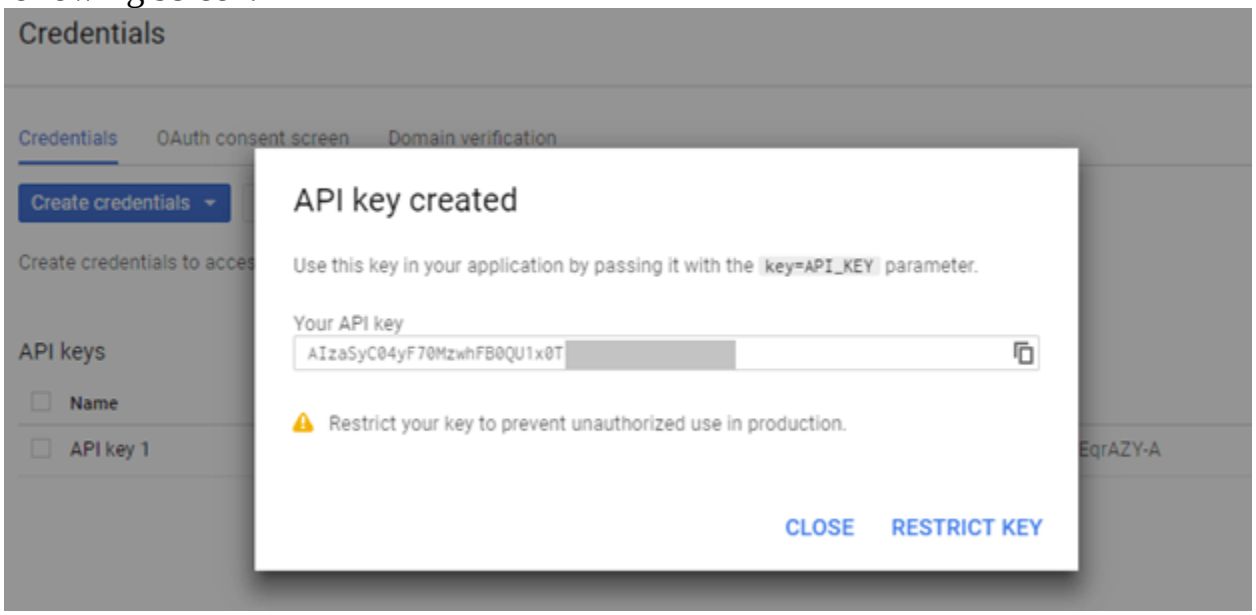
opy the URL from google_map_api.xml file to generate Google map key.



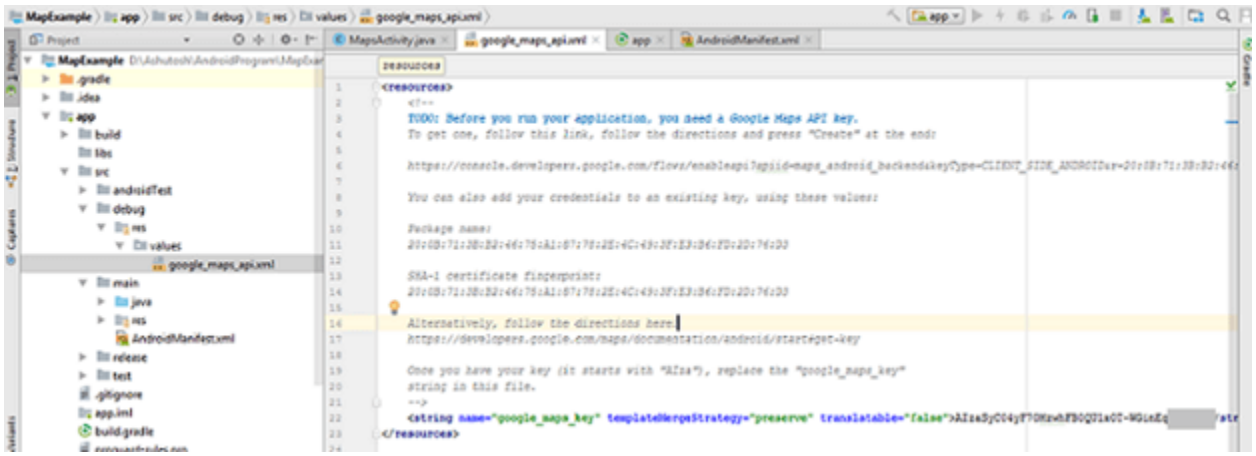
Paste the copied URL at the browser. It will open the following page.



After clicking on Create API key, it will generate our API key displaying the following screen.



Copy this generated API key in our *google_map_api.xml* file



activity_maps.xml

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="example.com.mapexample.MapsActivity" />
```

MapsActivity.java

To get the GoogleMap object in our MapsActivity.java class we need to implement the OnMapReadyCallback interface and override the onMapReady() callback method.

```
package example.com.mapexample;

import android.support.v4.app.FragmentActivity;
import android.os.Bundle;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback{

    private GoogleMap mMap;

```

```
16.    @Override
17.    protected void onCreate(Bundle savedInstanceState) {
18.        super.onCreate(savedInstanceState);
19.        setContentView(R.layout.activity_maps);
20.        // Obtain the SupportMapFragment and get notified when the map is rea
       dy to be used.
21.        SupportMapFragment mapFragment = (SupportMapFragment) getSupp
       ortFragmentManager()
22.                .findFragmentById(R.id.map);
23.        mapFragment.getMapAsync(this);
24.
25.    }
26.
27.    @Override
28.    public void onMapReady(GoogleMap googleMap) {
29.        mMap = googleMap;
30.
31.        // Add a marker in Sydney and move the camera
32.        LatLng sydney = new LatLng(-34, 151);
33.        mMap.addMarker(new MarkerOptions().position(sydney).title("Marker
        in Sydney"));
34.        mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
35.
36.    }
37.}
```

Required Permission

Add the following user-permission in AndroidManifest.xml file.

```
1. <uses-
   permission android:name="android.permission.ACCESS_FINE_LOCATIO
   N" />
2. <uses-
   permission android:name="android.permission.ACCESS_COARSE_LOCA
   TION" />
3. <uses-permission android:name="android.permission.INTERNET" />
```

AndroidManifest.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3.     package="example.com.mapexample">
4.     <!--
```

```
5.          The ACCESS_COARSE/FINE_LOCATION permissions are not requi
    red to use
6.          Google Maps Android API v2, but you must specify either coarse or fi
    ne
7.          location permissions for the 'MyLocation' functionality.
8.      -->
9.      <uses-
    permission android:name="android.permission.ACCESS_FINE_LOCATIO
    N" />
10.     <uses-
    permission android:name="android.permission.ACCESS_COARSE_LOCA
    TION" />
11.     <uses-permission android:name="android.permission.INTERNET" />
12.
13.     <application
14.         android:allowBackup="true"
15.         android:icon="@mipmap/ic_launcher"
16.         android:label="@string/app_name"
17.         android:roundIcon="@mipmap/ic_launcher_round"
18.         android:supportsRtl="true"
19.         android:theme="@style/AppTheme">
20.
21.         <meta-data
22.             android:name="com.google.android.geo.API_KEY"
23.             android:value="@string/google_maps_key" />
24.
25.         <activity
26.             android:name=".MapsActivity"
27.             android:label="@string/title_activity_maps">
28.             <intent-filter>
29.                 <action android:name="android.intent.action.MAIN" />
30.
31.                 <category android:name="android.intent.category.LAUNCHER"
    />
32.             </intent-filter>
33.         </activity>
34.     </application>
35.
36. </manifest>
```

METHODS TO GET CURRENT LOCATION

Sr.No.	Callback Methods & Description
1	**abstract void onConnected(Bundle connectionHint)** This callback method is called when location service is connected to the location client successfully. You will use **connect()** method to connect to the location client.
2	**abstract void onDisconnected()** This callback method is called when the client is disconnected. You will use **disconnect()** method to disconnect from the location client.
3	**abstract void onConnectionFailed(ConnectionResult result)** This callback method is called when there was an error connecting the client to the service.

You should create the location client in **onCreate()** method of your activity class, then connect it in **onStart()**, so that Location Services maintains the current location while your activity is fully visible. You should disconnect the client in **onStop()** method, so that when your app is not visible, Location Services is not maintaining the current location. This helps in saving battery power up-to a large extent.

GET UPDATED LOCATION

sr.No.	Callback Method & Description
1	**abstract void onLocationChanged(Location location)** This callback method is used for receiving notifications from the LocationClient when the location has changed.

EXAMPLE CODING

Keeping this application simple we have just added a SupportMapFragment tag inside content_main.xml as shown below:

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android=https://schemas.android.com.apk/res/android

   xmlns:app=https://schemas.android.com.apk/res/-auto

xmlns:tools=https://schemas.android.com/tools

android:layout_width="match_parent"
```

android:layout_height="mtch_parent"

app:layout_behavoiur="@string/appbar_scrolling_view_behaviour"

tools:context="com.journaldev.interatingmaps.MainActivity"

tools:showIn="@layout/activity_main">

<fragment

Android:id="@+id/map"android:name="com.google.android.gms.maps.SupportMapFragment"

Android:layout_width="match_parent"

Android:layout_gravity="center"

Android:layout_height="match+parent"

/>

</RelativeLayout>

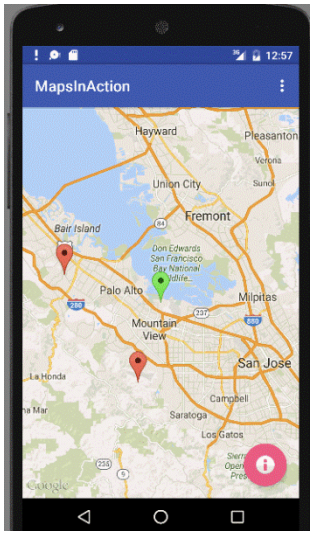
Add the following permission to the AndroidManifest.xml.file.

<uses-permission android:name="amdroid.permission.INTERNET"/>

Build and run this application on the emulator.

Just go to Settings->Apps->Google Play Services nd use that version number in the build.gradle in place of: compile 'com.google.android.gms:play=services:8.3.0'

The following output will be shown:



3.ANIMATION

Android provides a large number of classes and interface for the animation development. Most of the classes and interfaces are given in **android.animation** package.

Android Animation enables you to change the object property and behavior at run time. There are various ways to do animation in android.

The *AnimationDrawable* class provides methods to start and end the animation. Even, you can use time based animation.

Let's have a look at the simple example of android animation.

activity_main.xml

You need to have a view only.

File: activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <View
        />

</RelativeLayout>
```

File: logo.xml

Have a image view only.

```
<?xml version="1.0" encoding="utf-8"?>
<ImageView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/anm"
     >

</ImageView>
```

MainActivity class

File: MainActivity.java

```
package com.javatpoint.animation;

import android.os.Bundle;
import android.app.Activity;
import android.graphics.drawable.AnimationDrawable;
import android.widget.ImageView;

public class MainActivity extends Activity {

   ImageView anm;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.logo);
        anm = (ImageView)findViewById(R.id.anm);

        anm.setBackgroundResource(R.drawable.animation);
      // the frame-by-frame animation defined as a xml file within the drawable folder

        /*
         * NOTE: It's not possible to start the animation during the onCreate.
         */
     }
    public void onWindowFocusChanged (boolean hasFocus) {
        super.onWindowFocusChanged(hasFocus);
        AnimationDrawable frameAnimation =
            (AnimationDrawable) anm.getBackground();
        if(hasFocus) {
            frameAnimation.start();
        } else {
            frameAnimation.stop();
```

```
            }
        }

}
```

You need to create animation.xml file inside res/drawable-hdpi directory.

You need to have many images. Here, we are using 14 images and all the 14 images are located inside res/drawable-mdpi directory.

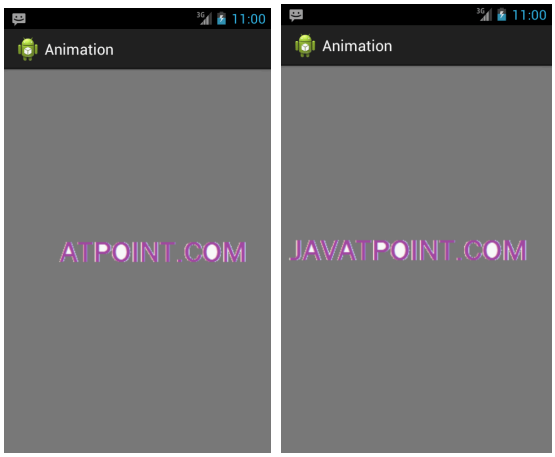
File: animation.xml

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="false">

    <item android:drawable="@drawable/frame0" android:duration="120" />
    <item android:drawable="@drawable/frame1" android:duration="120" />
    <item android:drawable="@drawable/frame2" android:duration="120" />
    <item android:drawable="@drawable/frame3" android:duration="120" />
    <item android:drawable="@drawable/frame4" android:duration="120" />
    <item android:drawable="@drawable/frame5" android:duration="120" />
    <item android:drawable="@drawable/frame6" android:duration="120" />
    <item android:drawable="@drawable/frame7" android:duration="120" />
    <item android:drawable="@drawable/frame8" android:duration="120" />
    <item android:drawable="@drawable/frame9" android:duration="120" />
    <item android:drawable="@drawable/frame10" android:duration="120" />
    <item android:drawable="@drawable/frame11" android:duration="120" />
    <item android:drawable="@drawable/frame12" android:duration="120" />
    <item android:drawable="@drawable/frame13" android:duration="120" />
    <item android:drawable="@drawable/frame14" android:duration="120" />
    <item android:drawable="@drawable/frame14" android:duration="120" />
    <item android:drawable="@drawable/frame13" android:duration="120" />
    <item android:drawable="@drawable/frame12" android:duration="120" />
    <item android:drawable="@drawable/frame11" android:duration="120" />
    <item android:drawable="@drawable/frame10" android:duration="120" />
    <item android:drawable="@drawable/frame9" android:duration="120" />
    <item android:drawable="@drawable/frame8" android:duration="120" />
    <item android:drawable="@drawable/frame7" android:duration="120" />
    <item android:drawable="@drawable/frame6" android:duration="120" />
    <item android:drawable="@drawable/frame5" android:duration="120" />
    <item android:drawable="@drawable/frame4" android:duration="120" />
    <item android:drawable="@drawable/frame3" android:duration="120" />
    <item android:drawable="@drawable/frame2" android:duration="120" />
    <item android:drawable="@drawable/frame1" android:duration="120" />
    <item android:drawable="@drawable/frame0" android:duration="120" />
```

35.
36. **</animation-list>**

Output:



3.1View Animation

You can use the view animation system to perform tweened animation on Views. Tween animation calculates the animation with information such as the start point, end point, size, rotation, and other common aspects of an animation.

A tween animation can perform a series of simple transformations (position, size, rotation, and transparency) on the contents of a View object. So, if you have a TextView object, you can move, rotate, grow, or shrink the text. If it has a background image, the background image will be transformed along with the text. The animation package provides all the classes used in a tween animation.

A sequence of animation instructions defines the tween animation, defined by either XML or Android code. As with defining a layout, an XML file is recommended because it's more readable, reusable, and swappable than hard-coding the animation. In the example below, we use XML. (To learn more about defining an animation in your application code, instead of XML, refer to the AnimationSet class and other Animation subclasses.)

The animation instructions define the transformations that you want to occur, when they will occur, and how long they should take to apply. Transformations can be sequential or simultaneous - for example, you can have the contents of a TextView move from left to right, and then rotate 180 degrees, or you can have the text move and rotate simultaneously. Each transformation takes a set of parameters specific for that transformation (starting size and ending size for size change, starting angle and ending angle for rotation, and so on), and also a set of common parameters (for instance, start time and duration). To make several transformations happen simultaneously, give them the same start time; to make them sequential, calculate the start time plus the duration of the preceding transformation.

The animation XML file belongs in the res/anim/ directory of your Android project. The file must have a single root element: this will be either a single <alpha>, <scale>, <translate>, <rotate>, interpolator element, or <set> element that holds groups of these elements (which may include another <set>). By default, all animation instructions are applied simultaneously. To make them occur sequentially, you must specify the startOffset attribute, as shown in the example below.

The following XML from one of the ApiDemos is used to stretch, then simultaneously spin and rotate a View object.

```
<set android:shareInterpolator="false">
  <scale
    android:interpolator="@android:anim/accelerate_decelerate_interpolator"
    android:fromXScale="1.0"
    android:toXScale="1.4"
    android:fromYScale="1.0"
    android:toYScale="0.6"
    android:pivotX="50%"
    android:pivotY="50%"
    android:fillAfter="false"
    android:duration="700" />
  <set android:interpolator="@android:anim/decelerate_interpolator">
    <scale
      android:fromXScale="1.4"
      android:toXScale="0.0"
      android:fromYScale="0.6"
      android:toYScale="0.0"
      android:pivotX="50%"
      android:pivotY="50%"
      android:startOffset="700"
      android:duration="400"
      android:fillBefore="false" />
    <rotate
      android:fromDegrees="0"
      android:toDegrees="-45"
      android:toYScale="0.0"
      android:pivotX="50%"
      android:pivotY="50%"
      android:startOffset="700"
      android:duration="400" />
  </set>
</set>
```

Screen coordinates (not used in this example) are (0,0) at the upper left hand corner, and increase as you go down and to the right.

Some values, such as pivotX, can be specified relative to the object itself or relative to the parent. Be sure to use the proper format for what you want ("50" for 50% relative to the parent, or "50%" for 50% relative to itself).

You can determine how a transformation is applied over time by assigning an Interpolator. Android includes several Interpolator subclasses that specify various speed curves: for instance, AccelerateInterpolator tells a transformation to start slow and speed up. Each one has an attribute value that can be applied in the XML.

With this XML saved as hyperspace_jump.xml in the res/anim/ directory of the project, the following code will reference it and apply it to an ImageView object from the layout.

3.2Android Property Animation — The ValueAnimator

ValueAnimator provides a timing engine for running animation which calculates the animated values and set them on the target objects. By ValueAnimator you can animate a view **width**, **height**, update its **x** and **y** coordinates or even can change its **background**.

It has an interface **ValueAnimator.AnimatorUpdateListener** which is used to receive callbacks on every animation frame.

```
ValueAnimator widthAnimator = ValueAnimator.ofInt(10, 100);
widthAnimator.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
   @Override
   public void onAnimationUpdate(ValueAnimator animation) {
      int animatedValue = (int) animation.getAnimatedValue();
      someView.getLayoutParams().width = animatedValue;
      someView.requestLayout();
   }
});
```

In above example, I'm updating the width of a view from **10** to **100**.
In **onAnimationUpdate()** method the animated value is used to update the width of that view. On calling **requestLayout()** the framework will redraw the view according to updated width value (this method is needed to be called on every change of value otherwise the view will not be redrawn according to updated value and effect would not be seen).

Besides this simple example, let's have a look at a little complex one. In this example, I'm going to change **x** and **y coordinates** and **size** of the button on click of a **floating action button**. And result is shown below —

4.Graphics android developer

The **android.graphics.Canvas** can be used to draw graphics in android. It provides methods to draw oval, rectangle, picture, text, line etc.
The **android.graphics.Paint** class is used with canvas to draw objects. It holds the information of color and style.
In this example, we are going to display 2D graphics in android
e: activity_main.xml

```
1. <RelativeLayout xmlns:androclass="http://schemas.android.com/apk/res/an
   droid"
2.     xmlns:tools="http://schemas.android.com/tools"
3.     android:layout_width="match_parent"
4.     android:layout_height="match_parent"
5.     android:paddingBottom="@dimen/activity_vertical_margin"
6.     android:paddingLeft="@dimen/activity_horizontal_margin"
7.     android:paddingRight="@dimen/activity_horizontal_margin"
8.     android:paddingTop="@dimen/activity_vertical_margin"
9.     tools:context=".MainActivity" >
10.
11.    <TextView
12.        android:layout_width="wrap_content"
13.        android:layout_height="wrap_content"
14.        android:text="@string/hello_world" />
15.
16. </RelativeLayout>
```

Activity class

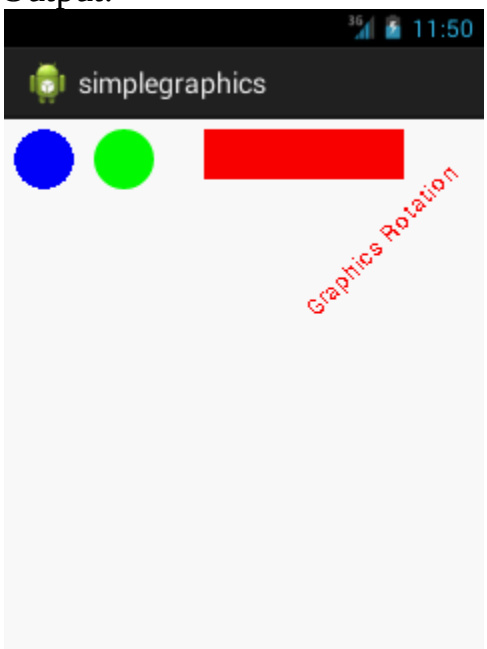
File: MainActivity.java

```
1.  package com.example.simplegraphics;
2.
3.  import android.os.Bundle;
4.  import android.app.Activity;
5.  import android.view.Menu;
6.  import android.content.Context;
7.  import android.graphics.Canvas;
8.  import android.graphics.Color;
9.  import android.graphics.Paint;
10. import android.view.View;
11.
12. public class MainActivity extends Activity {
13.
14.    DemoView demoview;
15.    /** Called when the activity is first created. */
16.    @Override
17.    public void onCreate(Bundle savedInstanceState) {
18.        super.onCreate(savedInstanceState);
19.        demoview = new DemoView(this);
20.        setContentView(demoview);
21.    }
```

```
22.
23.   private class DemoView extends View{
24.       public DemoView(Context context){
25.           super(context);
26.       }
27.
28.       @Override protected void onDraw(Canvas canvas) {
29.           super.onDraw(canvas);
30.
31.           // custom drawing code here
32.           Paint paint = new Paint();
33.           paint.setStyle(Paint.Style.FILL);
34.
35.           // make the entire canvas white
36.           paint.setColor(Color.WHITE);
37.           canvas.drawPaint(paint);
38.
39.           // draw blue circle with anti aliasing turned off
40.           paint.setAntiAlias(false);
41.           paint.setColor(Color.BLUE);
42.           canvas.drawCircle(20, 20, 15, paint);
43.
44.           // draw green circle with anti aliasing turned on
45.           paint.setAntiAlias(true);
46.           paint.setColor(Color.GREEN);
47.           canvas.drawCircle(60, 20, 15, paint);
48.
49.           // draw red rectangle with anti aliasing turned off
50.           paint.setAntiAlias(false);
51.           paint.setColor(Color.RED);
52.           canvas.drawRect(100, 5, 200, 30, paint);
53.
54.           // draw the rotated text
55.           canvas.rotate(-45);
56.
57.           paint.setStyle(Paint.Style.FILL);
58.           canvas.drawText("Graphics Rotation", 40, 180, paint);
59.
60.           //undo the rotate
61.           canvas.restore();
```

```
    }
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}
}
```

Output:



5.MULTIMEDIA SOFTWARE

Android supports multimedia by using the open source multimedia system called *OpenCORE* from PacketVideo Corporation. OpenCORE provides the foundation for Android's media services, which Android wraps in an easy-to-use API. In addition to the OpenCORE framework, the Android platform is migrating to a Google-written multimedia framework named Stagefright. Both frameworks are provided in version 2.2 (Froyo) and in subsequent versions of the SDK. It's anticipated that most, if not all, of the multimedia functionality will be handled by the Stagefright code base.

In this chapter, we'll look at OpenCORE's multimedia architecture and features, and then use it via Android's MediaPlayer API to play audio files, take a picture, play videos, and finally record video and audio from the emulator. To begin, let's look at OpenCORE's multimedia architecture.

Get Android in Action, Second Edition

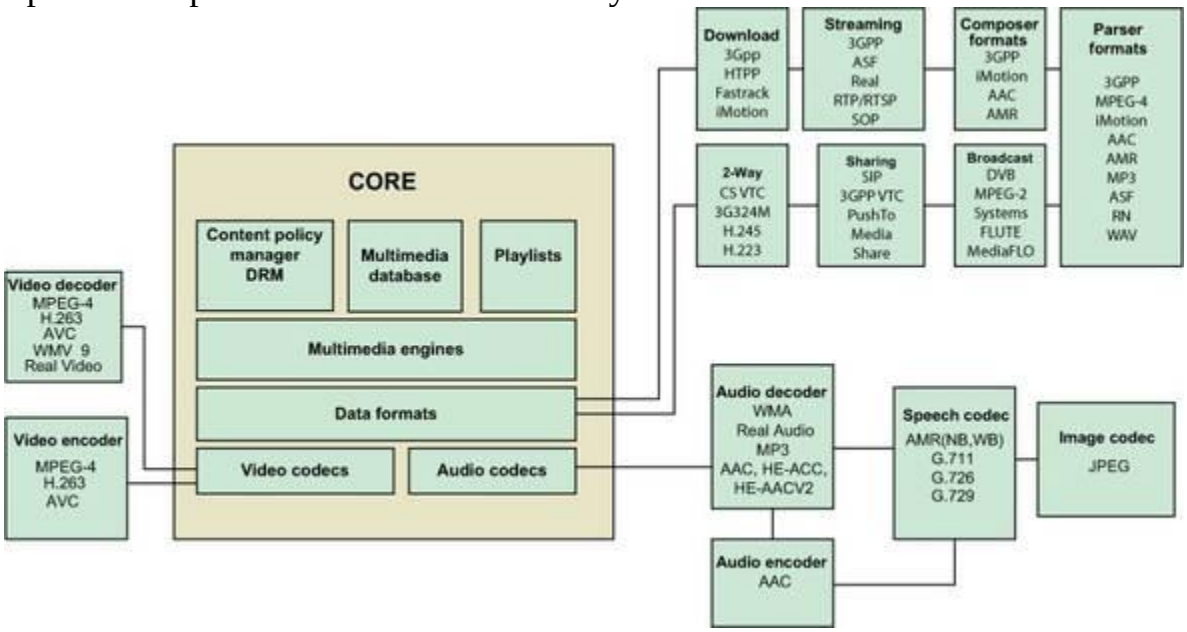
add to cart

5.1 Introduction to multimedia and OpenCORE

Because the foundation of Android's multimedia platform is PacketVideo's Open-CORE, we're going to review OpenCORE's architecture and services. OpenCORE is a Java open source, multimedia platform that supports:

- Interfaces for third-party and hardware media codecs, input and output devices, and content policies
- Media playback, streaming, downloading, and progressive playback, including 3rd Generation Partnership Program (3GPP), Moving Picture Experts Group 4 (MPEG-4), Advanced Audio Coding (AAC), and Moving Picture Experts Group Audio Layer 3(MP3) containers
- Video and image encoders and decoders, including MPEG-4, International Telecommunication Union H.263 video standard (H.263), Advanced Video Coding (AVC H.264), and the Joint Photographic Experts Group (JPEG)
- Speech codecs, including Adaptive Multi-Rate audio codecs AMR-NB and AMR-WB
- Audio codecs, including MP3, AAC, and AAC+
- Media recording, including 3GPP, MPEG-4, and JPEG
- Video telephony based on the 3GPP video conferencing standard 324-M
- PV test framework to ensure robustness and stability; profiling tools for memory and CPU usage

OpenCORE provides all this functionality in a well-laid-out set of services.



5.2 Playing audio

Probably the most basic need for multimedia on a cell phone is the ability to play audio files, whether new ringtones, MP3s, or quick audio notes. Android's Media Player is easy to use. At a high level, all you need to do to play back an MP3 file is follow these steps:

1. Ery rdo MP3 nj vgr re/wrsa directory nj z project (vxnr rzrb xqd nsz cfvc ckb s OYJ rk access files kn xrp roknewt et jec rog itnentre).

2. Treeta s vwn instance kl rdx MediaPlayer pnz ceereefnr xbtg MP3 ud agcliln MediaPlayer.create().

3. Call the MediaPlayer methods prepare() and start().

For'z eotw rhougth nc amepexl xr odtasnmeret ykw mlsipe parj rcse aj. Vartj, cartee s wxn project dlcale MediaPlayer Vmpexal, rpwj nz Activity ecdlal MediaPlayer-Activity. Gwv, cateer z nwk relfod rdnue dcrle/aels tsw. Mx'ff teosr vgt MP3 a nj crqj fodlre. Ext jbar leapemx, wo'ff xzq z ngeirton etl vrb hmxz Hfez 3, ihhcw kbb znc tvreriee klmt MediaPlayer.create. Udnwolao grv Hxzf 3 etmhe navd nzb hns oerth MP3 a xdb yacnf, nsh hdr kmur jn xrp tzw directory. Oxrk, aretce z mplsei Button vtl rgv mcisu ylrpea, cc nhwos jn rog nowgfoill itingsl.

Listing 10.1. main.xml for MediaPlayer Example

<?xml version="1.0" encoding="utf-8"?> <**LinearLayout** xmlns:android="http://schemas.android.com/apk/res/android" android:orientation="vertical" android:layout_width="fill_parent" android:layout_height="fill_parent" > <**TextView** android:layout_width="fill_parent" android:layout_height="wrap_content" android:text="Simple Media Player" /> <**Button** android:id="@+id/playsong" android:layout_width="fill_parent" android:layout_height="wrap_content" android:text="Halo 3 Theme Song" /> </**LinearLayout**>

copy

Mv nykv xr fljf vrb eyt MediaPlayerActivity class, zz snhwo nj qkr iolngfowl islgtni.

Listing 10.2. MediaPlayerActivity.java

```
public class MediaPlayerActivity extends Activity {
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.main);
        Button mybutton = (Button) findViewById(R.id.playsong);
        mybutton.setOnClickListener(new Button.OnClickListener() {
               public void onClick(View v) {
                   MediaPlayer mp =
 MediaPlayer.create(MediaPlayerActivity.this,
 R.raw.halotheme);
                  mp.start();
                  mp.setOnCompletionListener(new OnCompletionListener(){
                         public void onCompletion(MediaPlayer arg0) {

                         }
                    }
                    );
               }
        }
        );
    }
}
```

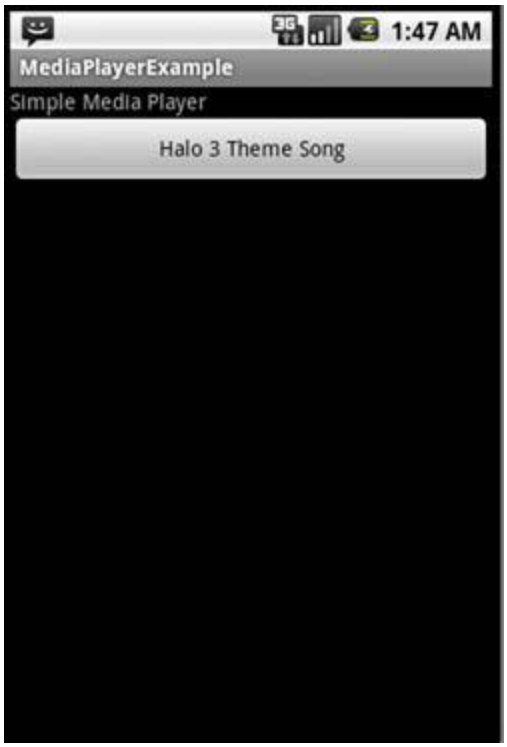
1 Set view and button to play MP3

2 Get context and play MP3

As you can see, playing back an MP3 is easy. In listing 10.2, all we did was use the view that we created in listing 10.1 and map the button, playsong, to mybutton, which we then bound to the setOnClickListener() ❶. Inside the listener, we created the MediaPlayer instance ❷ using the create(Context context, int resourceid) method, which takes our context and a resource ID for our MP3. Finally, we set the setOnCompletionListener, which will perform some task on completion. For the moment, we do nothing, but you might want to change a button's state or provide a notification to a user that the song is over, or ask if the user would like to play another song. If you want to do any of these things, you'd use this method.

Ukw jl ueg lmioecp opr application sun tbn jr, xdb ludhso vkz egnhtismo vfjx figure 10.2. Bfvja ryo uttbon, cnh qgv housld kcut pvr Hefc 3 nkad yedalp dsoz jn uor emulator xjc vqtg kprsesea. Rxg anc czfk orcntlo ukr molvue xl rkg ylacbkpa rgwj rkg evmulo switches xn xrq uvaj xl xur Android emulator phone visualization.

Simple media player example



Kwk zryr wx'vk edloko sr ewp rx ysfb sn audio file, rfo'a xxa wxq qvh anz fhqs s video file.

5.3. Playing video

Playing a video is slightly more complicated than playing audio with the MediaPlayer API, in part because you have to provide a view surface for your video to play on. Android has a VideoView widget that handles that task for you; you can use it in any layout manager. Android also provides a number of display options, including scaling and tinting. So let's get started with playing video by creating a new project called Simple Video Player. Next, create a layout, as shown in the following listing.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
    <VideoView android:id="@+id/video"
    android:layout_width="320px"
    android:layout_height="240px"
    />/
</LinearLayout>
```

1 Add VideoView widget

Cff wx'xk nxyx nj jzrq tislngi cj cqh rop VideoView giedwt ❶.
Bbv VideoView sdporeiv z DJ dgewti jpwr arvg, yfcb, deacvna, irdwen, unc ohetr osbtnut, kiagnm rj ueneysnsrca er hhs thpe vwn. Urko, ow ounk re riewt s class xr pgzf rxb video, hchiw jc snwoh nj org wonfglloi intlsig.

```
public class SimpleVideo extends Activity {

    private VideoView myVideo;
    private MediaController mc;
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        getWindow().setFormat(PixelFormat.TRANSLUCENT);
        setContentView(R.layout.main);
        this.myVideo = (VideoView) findViewById(R.id.video);
        this.myVideo.setVideoPath("sdcard/test.mp4");
        this.mc = new MediaController(this);
        this.mc.setMediaPlayer(this.myVideo);
        this.myVideo.setMediaController(this.mc);
        this.myVideo.requestFocus();
    }
}
```

❶ **Create translucent window**

In this listing, we first created a translucent window, which is necessary for our SurfaceView ❶. Next, we reference the VideoView as a container for playing the video, and use its setVideoPath() to have it look at an SD card for our test MP4. Finally, we set up the MediaController and use the setMediaController() to perform a callback to the VideoView to notify it when our video is finished playing.

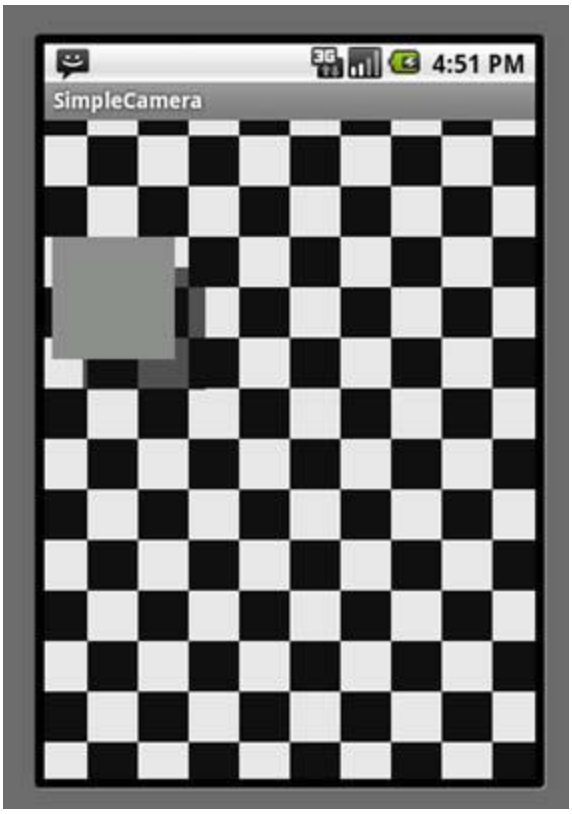
6.ANDROID CAMERA

n important feature of modern cell phones is their ability to take pictures or video using a built-in camera. Some phones even support using the camera's microphone to capture audio. Android, of course, supports all three features and provides a variety of ways to interact with the camera. In this section, we're going to look at how to interact with the camera and take photographs. In the next section, you'll use the camera to take video and save it to an SD card.

Thk'ff do creating z wno project dlealc SmpielYaraem rk ertoadmtnse weq vr tconcen vr z phone 'a aemacr xr uceptar maiegs. Pkt zjru project, wk'ff go ungsi dvr `Camera` class (http://code.google.com/android/reference/android/hardware/Camera.html) rx jor dvr emulator 'c (et phone 'c) rmacae re s `View`. Wcrv el urv xpkz rrus wx'xk ereatcd tlk rjau project seald jywr sihnwog rxb input lmkt rog aamerc, brh rkd nmzj tkvw lte ngkati z upciret ja vonq ph c ngilse method acedll `take-Picture(Camera.ShutterCallback shutter, Camera.PictureCallback raw, Camera.PictureCallback jpeg)`, hcwhi szd erteh kcalcsbla srur wlola pxp rv tcolnor wvq c peucitr jz anetk.

Reefor wo vrb nqs hrfuter nrjv pvr `Camera` class pzn vwd vr cho rou ceaamr, vrf'c arceet ord project. Mk'ff vp creating wrx classes; ebacesu xru cmnj class zj neqf, wk'ff arkeb jr jnrv rwk eossticn. Ptx ory strfi iotcsne, kfok cr ukr olngfoliw intsgil, Braema-Fleapxm.zico.

Test pattern coming from the emulator camera and displayed in the SimpleCamera application



CameraExample.java

```
public class SimpleCamera extends Activity implements
SurfaceHolder.Callback
 {
    private Camera camera;
    private boolean isPreviewRunning = false;
    private SimpleDateFormat timeStampFormat = new
        SimpleDateFormat("yyyyMMddHHmmssSS");
    private SurfaceView surfaceView;
    private SurfaceHolder surfaceHolder;
    private Uri targetResource = Media.EXTERNAL_CONTENT_URI;
    public void onCreate(Bundle icicle)
    {
    super.onCreate(icicle);
    Log.e(getClass().getSimpleName(), "onCreate");
    getWindow().setFormat(PixelFormat.TRANSLUCENT);
    setContentView(R.layout.main);
    surfaceView = (SurfaceView)findViewById(R.id.surface);
    surfaceHolder = surfaceView.getHolder();
    surfaceHolder.addCallback(this);
    surfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
}
@Override
public boolean onCreateOptionsMenu(android.view.Menu menu) {
    MenuItem item =
menu.add(0, 0, 0, "View Photos?");
    item.setOnMenuItemClickListener(new
```

1 Create menu to Android's Photo Gallery

```
            MenuItem.OnMenuItemClickListener() {
            public boolean onMenuItemClick(MenuItem item) {
                Intent intent = new Intent(Intent.ACTION_VIEW,
                    SimpleCamera.this.targetResource);
                startActivity(intent);
                return true;
            }
        });
        return true;
    }
    @Override
    protected void onRestoreInstanceState(Bundle savedInstanceState)
{
        super.onRestoreInstanceState(savedInstanceState);
    }
    Camera.PictureCallback mPictureCallbackRaw = new
        Camera.PictureCallback() {
        public void onPictureTaken(byte[] data, Camera c) {
            SimpleCamera.this.camera.startPreview();
        }
    };
    Camera.ShutterCallback mShutterCallback = new Camera.ShutterCallback()
{
        Public void onShutter() {}
        }
    };
```

2 Create PictureCallback

3 Create ShutterCallback

Xqcj igiltns cj dtaarorhigwtsfr. Erzjt, vw zor riaevalsb ltk managing c `surfaceView` chn rvny arv qd xrb `View`. Kvrk, ow reteac z psmiel vmpn znq nmxh ponoti brrz fwjf fotla toxv txy feacsru yvwn roy user lkccis rob Menu btnuto en oqr phone elhiw oru application jc rniungn ❶. Uunej ze fwfj nukx Android 'z erpucit bworrse snq xfr krp user view kry ptoohs ne rkb rcamea. Uvor, wv cereta bkr ftisr `PictureCallback`, ihwhc aj lcedal woyn z icueprt jc itrfs antke ❷. Rdjz sfitr balckcla psceuatr rob `Picture-Callback` 'c fnxu method, `onPictureTaken(byte[] data, Camera camera)`, vr tcuu xdr stw megai data eclrdyit lvmt ogr aacemr. Oxor, kw tareec z `ShutterCallback`, wchhi snz pv dpao wrdj jrz method, `onShutter()`, re fcqh c sduno; tpox ow pnx'r affz prk method ❸. Mv'ff tenniuco rdjw rgo XaarmeLlmeaxp.eczi jn oru krnk glntsii.

CameraExample.java continued

```
@Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {

        ImageCaptureCallback camDemo = null;
        if(keyCode == KeyEvent.KEYCODE_DPAD_CENTER) {
            try {
                String filename = this.timeStampFormat.format(new Date());
                ContentValues values = new ContentValues();
                values.put(MediaColumns.TITLE, filename);
                values.put(ImageColumns.DESCRIPTION,
                    "Image from Android Emulator");
                Uri uri =
                 getContentResolver().insert(
                Media.EXTERNAL_CONTENT_URI, values);
                camDemo = new ImageCaptureCallback(
                    getContentResolver().openOutputStream(uri));
            } catch(Exception ex ){
            }
        }
        if (keyCode == KeyEvent.KEYCODE_BACK) {
            return super.onKeyDown(keyCode, event);
        }
        if (keyCode == KeyEvent.KEYCODE_DPAD_CENTER) {
            this.camera.takePicture(this.mShutterCallback,
                this.mPictureCallbackRaw, this.camDemo);
            return true;
        }
        return false;
    }
    @Override
    protected void onResume()
{
        Log.e(getClass().getSimpleName(), "onResume");
        super.onResume();
    }
    @Override
```

1 Create method to detect key events

If center key pressed, write file to sdcard 2

If center key depressed, take picture 3

```
    protected void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
    }
    @Override
    protected void onStop()
    {
       super.onStop();
    }
    public void surfaceChanged(SurfaceHolder holder,
int format, int w, int h)
 {
        if (this.isPreviewRunning) {
            this.camera.stopPreview();
        }
        Camera.Parameters p = this.camera.getParameters();
        p.setPreviewSize(w, h);
        this.camera.setParameters(p);
        try{
            this.camera.setPreviewDisplay(holder);
        }catch(IOexcetion e{
            e.printStackTrace();
        }
        this.camera.startPreview();
        this.isPreviewRunning = true;
    }
    public void surfaceCreated(SurfaceHolder holder)
 {
       this.camera = Camera.open();
    }
    public void surfaceDestroyed(SurfaceHolder holder) {
        this.camera.stopPreview();
        this.isPreviewRunning = false;
        this.camera.release();
    }
}
```

Rqja gstilni jz mtek odiceacmtpl nbsr listing 10.5, gluahoht s lgera tnaomu kl rkb vqvz jc boaut managing qro saurcef txl rvq aacmre ytv view. Avd ftsri jnxf ja rvp tatsr lv nc ptenaeinitmmlo xl qkr method `onKeyDown` ❶, hichw scekch rx oka ethwerh xqr rtcene key nx vdr chuq zaw srepsde. Jl rj wzz, wx rka qh brv eonctira vl z file, pnz qh uigsn ykr `ImageCaptureCallback`, hwhic kw'ff fndeei jn listing 10.7, vw tcaeer zn `Outputstream` kr rtwie ptx aeimg data re ❷, icludnign rne fneh xrp agmei ubr dor file mnzx hnc ohrte zrxm data. Qrxv, wk fafz xpr method `takePicture()` cnq zsuc er rj pvr erhet kabacllcs `mShutterCallback`, `mPictureCallbackRaw`, ncg `camDemo`. `mPictureCallbackRaw` jz vbt tsw igmae sng `camDemo` etwirs odr eigam er z file ne rob SD card ❸, cc dxu sns xao jn rvq nfoowlgli initgls.

ImageCaptureCallback.java

```
public class ImageCaptureCallback implements PictureCallback  {     private OutputStream filoutputStream;
public ImageCaptureCallback(OutputStream filoutputStream) {         this.filoutputStream = filoutputStream;     }
public void onPictureTaken(byte[] data, Camera camera) {         try {             this.filoutputStream.write(data);
this.filoutputStream.flush();             this.filoutputStream.close();         } catch(Exception ex) {
ex.printStackTrace();         }     } }
```

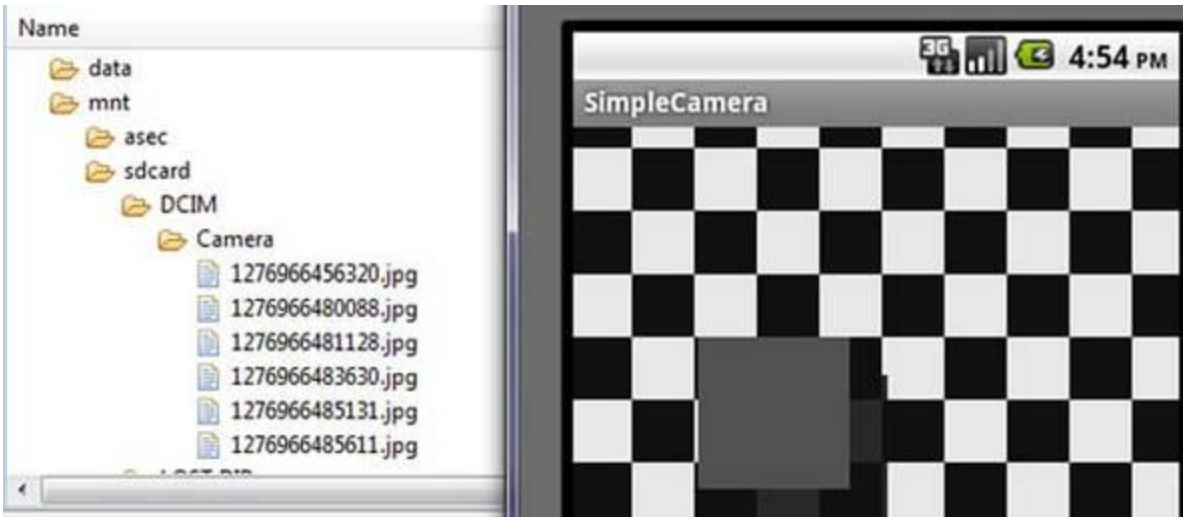
copy

Hvxt, vyr class mmtieenlsp rpv `PictureCallback` enrifecat nzy riepvsod rwk methods. Xux otcncorutsr etsrcea z stream kr irwte data rk, nch qkr cdnseo method, `onPictureTaken`, takes nairby data nzb retwis rk ruk SD card as s JPEG.

Jl hvu iludb jprc project sun statr org emulator uinngrn gsniu prv SD card ieamg xbh creatde rleirea jn zrjp atchrpe, bhx dusloh xoz sehnmogti fkvj figure 10.4 gwxn ybk rstta org Semlip Camera application lmet xrp Android nxbm. Jl ypv fovk rc figure 10.4, qgk'ff ioetnc zn pxp lcabk-hsn-wieth hkecced bkoncaurdg rwuj c ogbncinu pzpt vye. Cxq Android emulator aetsrenge ruzj raor etartnp vr umsletia zn aimge vvlq scuaebe gxr emulator nja'r nglpuli c fjko qxlv mlkt gxr raeamc.

Dxw lj qde klicc opr trneec ttonbu vn xry shqb nj rbx emulator, qrv application ffjw rcox c prctiue. Ck vvz roy retiupc, click rdk Menu otuntb; z onym ppresaa en ryo reacma view wnwido jrpw s leisng niopto, View Vurescit. Jl ugv select View Zersutci, qdx'ot enkta rk rxu Android retiupc roelrpex, qnz yxg shdolu xoc Android 'a eaimg colldeasrhpe penrngesetir vgr number le aeacmr cptarues. Rgk zsn zvfs ock gxr JPEG files rcrp vwtk itwnter rv rpk SD card bd opening rbv DDMS jn Eclipse npc ntiaigavgn rx mrn > cddsra > OTJW > Temara. Rkp zsn kav cn lemapxe nj figure 10.5.

The Android emulator shows placeholder images for each photo taken.



Yz kqg sns vzv, working with vdr acearm nj Android nja'r ilrcprauylat maecipcotdl. Ye xxc vwy z tkfc ecraam ffwj eabhev, egu'ff gzve vr vrar en s zvft hsndtea unlit rkb emulator rsipvode c isepml wcb rx ctnceno kr c mrcaae nv gtpv ocrumtpe. Cjcp evtw-udraon uodnlhs'r hxra bge metl eeoilgnpdv tpeg aacrme applications. Y hwlaet lv Android applications elyarad smkae hseticsidatop oaq lx gxr maearc, aiggnrn tvlm agmes rk ns application rdzr cqcv s pricute lv ddte kcsl rx colknu utxh phone.

Qwx rzry xpb'xv kkzn wqk kry `Camera` class koswr nj Android, fvr'c xxef rs qwk rx atpreuc tx eodrrc audio ktlm z mreaac'a mocri phone. Jn rxy rnvk eisncot, kw'ff oelrpxe dxr `MediaRecorder` class ngz ddx'ff wriet dngerrocis er ns SD card.

6.1 Capturing audio

Now we'll look at using the onboard microphone to record audio. In this section, we're going to use the Android MediaRecorder example from Google Android Developers list, which you can find at http://code.google.com/p/unlocking-android/. The code shown in this section has been updated slightly.

Jn rnleega, rredcingo audio te video lolfsow roy cmxc process jn Android:

1. Create an instance of `android.media.MediaRecorder`

2. Tertae sn instance lv `andriod.content.ContentValues`, sun hyz prroepiest sbzp sa `TITLE`, `TIMESTAMP`, hns xpr zff-mj port crn `MIME_TYPE`

3. Brteae z file path tlx rpx data rx vu xr nsugi `android.content.ContentResolver`

4. Bk kcr s tkb view syapdil kn c view esaucfr, coy `MediaRecorder.setPreviewDisplay()`

5. Set the source for audio, using `MediaRecorder.setAudioSource()`

6. Set output file format, using `MediaRecorder.setOutputFormat()`

7. Set your encoding for audio, using `MediaRecorder.setAudioEncoder()`

8. Use `prepare()` and `start()` to prepare and start your recordings

9. Kzk `stop()` sun `release()` vr lugeyafclr reqz sun clnea yd tbvp idrrengco process

Rouhgh drercigon maide ncj'r leayelispc epxlcom, egy ssn ocx rpsr jr'z mtvx mxoeclp zrnb playing jr. Cv dtensaurnd kqw er vcy orb `MediaRecorder` class, wv'ff kkfe cr sn application. Yrtaee s nwx application dellac SyenhTigrnocdeKmko. Krov, vpq knoq kr prvj bvr AndroidManifest.xml file cnu uqs ory niowlfgol jfnx:

```
<uses-permission android:name="android.permission.RECORD_AUDIO" />
```

copy

Bapj ekbs wjff alowl drk application re rrdceo ory audio files gcn hcfq vmry. Qrvo, taecer rog class onshw nj rvq ligowfnlo nilsigt.

SoundRecordingdemo.java

```
public class SoundRecordingDemo extends Activity  {
    MediaRecorder mRecorder;
    File mSampleFile = null;
    static final String SAMPLE_PREFIX = "recording";
    static final String SAMPLE_EXTENSION = ".mp3";
    private static final String TAG="SoundRecordingDemo";
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        this.mRecorder = new MediaRecorder();
        Button startRecording = (Button)findViewById(R.id.startrecording);
        Button stopRecording = (Button)findViewById(R.id.stoprecording);
        startRecording.setOnClickListener(new View.OnClickListener(){
            public void onClick(View v) {
                startRecording();
            }
        });
        stopRecording.setOnClickListener(new View.OnClickListener(){
            public void onClick(View v) {
                stopRecording();
                addToDB();
            }
        });
    }
    protected void addToDB() {
        ContentValues values = new ContentValues(3);
        long current = System.currentTimeMillis();
        values.put(MediaColumns.TITLE, "test_audio");
        values.put(MediaColumns.DATE_ADDED, (int) (current / 1000));
        values.put(MediaColumns.MIME_TYPE, "audio/mp3");
        values.put(MediaColumns.DATA, mSampleFile.getAbsolutePath());
        ContentResolver contentResolver = getContentResolver();
        Uri base = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
        Uri newUri = contentResolver.insert(base, values);
```

Set metadata for audio ❶

```
        sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE,
newUri));
    }
    protected void startRecording() {
        this.mRecorder = new MediaRecorder();
        this.mRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
        this.mRecorder.setOutputFormat
(MediaRecorder.OutputFormat.THREE_GPP);
        this.mRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
        this.mRecorder.setOutputFile(this.mSampleFile.getAbsolutePath());
        try{this.mRecorder.prepare();
        } catch (IllegalStateException e1) {
                    e1.printStackTrace();
           } catch (IOException e1 {
                    e1.printStackTrace();
           }
        this.mRecorder.start();
        if (this.mSampleFile == null) {
            File sampleDir = Environment.getExternalStorageDirectory();
            try {
                 this.mSampleFile = File.createTempFile(
                     SoundRecordingDemo.SAMPLE_PREFIX,
                     SoundRecordingDemo.SAMPLE_EXTENSION, sampleDir);
            } catch (IOException e) {
                Log.e(TAG,"sdcard access error");
                return;
            }
        }
    }
    protected void stopRecording() {
        this.mRecorder.stop();
        this.mRecorder.release();
    }
}
```

2 Notify music player new audio file created

3 Start recording file

4 Stop recording and release MediaRecorder

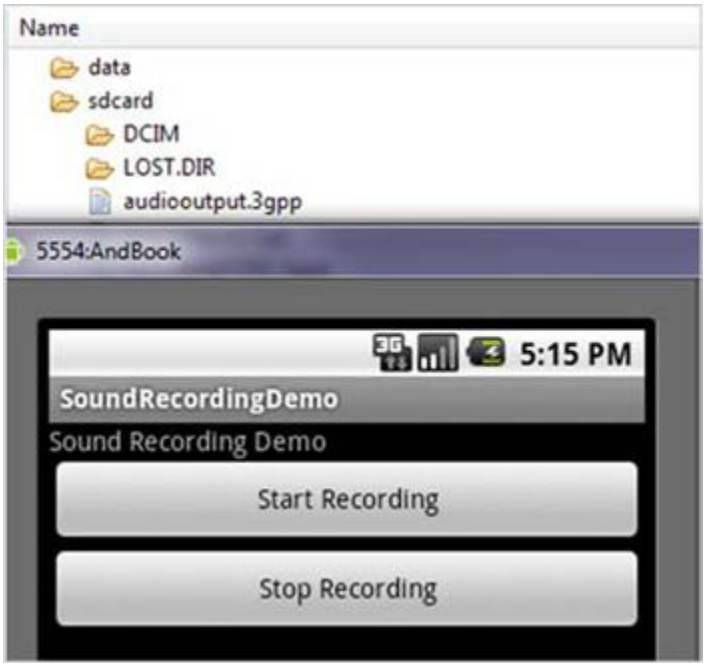
Jn rjgc tnilgis, xqr irtfs ztru le rog sukv jc creating bor sbonttu nuc bntuto rnsesitle rv tasrt zqn zkrg obr recdirngo. Bou itrsf rhct lx ukr itinlsg kpb oqxn re gzg nniotteta vr jz rkd `addToDB()` method. Jn qrcj method, wv rcv fsf roy rxcm data tle xrb audio file wk fqnc rk save, nlignudic drv letit, hcrv, cbn rhgk kl file ❶. Qoro, ow szff kqr `Intent` `ACTION_MEDIA_SCANNER_SCAN_FILE` vr oyntif applications hzap cz Android ’z Music Player rrsu s vnw audio file scu yvnk edecart ❷. Aalingl jurz `Intent` aloswl qc kr vha xbr Music Player kr okxf etl knw files nj c lyaplits nsu ggcf pro files.

Kkw sqrr vw’ke finish yx pvr `addToDB` method, ow cterae kur `startRecording` method, icwhh teraecs z wxn `MediaRecorder` ❸. Ta nj kru steps nj yor nbiingegn lk jyrc tiescno, wv rck ns audio

roesuc, wihch aj rdk mirco phone, zrv ns touput froamt sc `THREE_GPP`, crk uvr audio erneodc grqx re `AMR_NB`, cqn gvrn rxa oqr touptu file path kr eitwr xur file. Grvv, vw axy vrd methods `prepare()` bsn `start()` er leaneb audio necoirgdr.

Vnially, ow ecatre rgo `stopRecording()` method kr xcru gvr `MediaRecorder` xlmt givasn audio 4 qu usgni oyr methods `stop()` nyz `release()`. Jl bpv iudlb gajr application znp tpn prx emulator wurj krd SD card agemi vlmt xry upoerisv tecnosi, pey dhsolu oq sfkh rk cluahn rgv application mlet Eclipse ngz serps rob Srsrt Ycerdiong buttno. Cotrl s vlw endcsos, ssper kqr Sxur Ynroecgid ttbonu nsh vnvg rqv DDMS; vyg shdoul uv fgoc rx nagaveit rx urk dadcsr rlfeod bnc vkc btxh erinogscdr, cs wonhs nj figure 10.6.

Figure 10.6. An example of audio files being saved to the SD card image in the emulator



Jl miusc cj playing ne hteg cmeuropt'a audio meysts, ykr Android emulator ffwj zbvj jr dg znu eorcrd jr cldyreit ltme xrd audio freubf (rj'a ern nerroicgd mtlx z crmoi phone). Rgv acn nrdv iyales rrkz rewtehh jr odedrerc douns qd opening vyr Android Music Player sgn select jnh Zlisalyts > Aletnyec Rbhpv. Jr dulsho chqf phtx drecdore file, nbc xbg lohsud vq fgsx re zxtq ihtynnag ruzr czw playing vn bdet ucrotpem zr yxr ojrm.

Ra kl vnioers 1.5, Android cyh port gv ryo rdroeincg le video, uaohhtgl dnmc redeopeslv foudn jr dcftfulii nsy mcxk odrvsne imteemlpdne rhtie nwe custom ot ouiolstsn re zhh port video endicgorr. Mrgj rdo earsele le 2.0, 2.1, cny 2.2, video syz meeocb tlc esaeir re ktwv djwr, pkrg xtl playing zz fkfw ca onrgiedrc. Aqx'ff oxz pwv qmpa raiese jn qrx noro ticnoes uobat gsuin xqr `MediaRecorder` class re ewirt z pmelsi application lvt rroenigcd video.

Sign in for more free preview time

sign in now

6.2 Recording video

Video recording on Android is no more difficult than recording audio, with the exception that you have a few different fields. But there's one important difference—unlike with recording audio data, Android requires you to first preview a video feed before you can record it by passing it a surface object much like we did with the camera application earlier in this chapter. It's worth repeating this point because when Android started supporting video recording, many developers found themselves unable to record video: You must always provide a surface object. This might be awkward for some applications, but it's currently required in Android up to 2.2. Also, like recording audio, you have to provide several permissions to Android so you can record video. The new one is `RECORD_VIDEO`, which lets you use the camera to record video. The other permissions are `CAMERA`, `RECORD_AUDIO`, and `WRITE_EXTERNAL_STORAGE`, as shown in the following listing. So go ahead and set up a new project called VideoCam and use the permissions in this AndroidManifest.xml.

AndroidManifest.xml

<?xml version="1.0" encoding="utf-8"?> <**manifest** xmlns:android="http://schemas.android.com/apk/res/android" package="com.msi.manning.chapter10.VideoCam" android:versionCode="1" android:versionName="1.0"> <**application** android:icon="@drawable/icon" android:label="@string/app_name"> <**activity** android:name=".VideoCam" android:label="@string/app_name"> <**intent-filter**> <**action** android:name="android.intent.action.MAIN" /> <**category** android:name= "android.intent.category.LAUNCHER" /> </**intent-filter**> </**activity**> </**application**> <**uses-permission** android:name="android.permission.CAMERA"> </**uses-permission**> <**uses-permission** android:name= "android.permission.RECORD_AUDIO"></**uses-permission**> <**uses-permission** android:name= "android.permission.RECORD_VIDEO"></**uses-permission**> <**uses-permission**

```
android:name= "android.permission.WRITE_EXTERNAL_STORAGE" />        <uses-feature
android:name="android.hardware.camera" /> </manifest>
```

copy

Owx brzr dvy'oo defined gvr manifest, uvd onop rx ceeart c epsmli layout crqr say z kht view zkts znq ezkm osttbnu rx trsta, uarv, eausp, nzb dgzf ptxp video rniegcrod. Coq layout jz hwosn nj xrp golifwoln inlgist:

maim.xml

```
<?xml version="1.0" encoding="utf-8"?> <!-- This file is /res/layout/main.xml --> <RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"        android:orientation="vertical"
android:layout_width="fill_parent"        android:layout_height="fill_parent">     <RelativeLayout
android:layout_width="fill_parent"        android:layout_height="wrap_content"
android:id="@+id/relativeVideoLayoutView"            android:layout_centerInParent="true">            <VideoView
android:id="@+id/videoView"      android:layout_width="176px"              android:layout_height="144px"
android:layout_centerInParent="true"/>     </RelativeLayout>     <LinearLayout
android:layout_width="wrap_content"        android:layout_height="wrap_content"
android:orientation="horizontal"        android:layout_centerHorizontal="true"
android:layout_below="@+id/relativeVideoLayoutView">        <ImageButton
android:id="@+id/playRecordingBtn"            android:layout_width="wrap_content"
android:layout_height="wrap_content"            android:background="@drawable/play"            />     <ImageButton
android:id="@+id/bgnBtn"        android:layout_width="wrap_content"        android:layout_height="wrap_content"
android:background="@drawable/record"        android:enabled="false"        />     </LinearLayout>
</RelativeLayout>
```

copy

1. Create an instance of `android.media.MediaRecorder`.
2. Set up a `VideoView`.
3. Yk xrz z vth view layisdp nk c view arsefcu, cpx `MediaRecorder.setPreviewDisplay()`.
4. Set the source for audio, using `MediaRecorder.setAudioSource()`.
5. Set the source for video, using `MediaRecorder.setVideoSource()`.
6. Set your encoding for audio, using `MediaRecorder.setAudioEncoder()`.
7. Set your encoding for video, using `MediaRecorder.setVideoEncoder()`.

8. Set output file format using `MediaRecorder.setOutputFormat()`.

9. Srk rop video cxaj nugsi `setVideoSize()`. (Xr krq jrvm djcr ykov wac riewntt, ether awc z puh jn `setVideoSize` srrd dmilite jr rx 320 bd 240.)

10. Set the video frame rate, using `setVideoFrameRate`.

11. Use `prepare()` and `start()` to prepare and start your recordings.

12. Gxz `stop()` sqn `release()` kr erfugllcay rcdv qns lncea dy thxb rdeoincrg process.

Ta vup zns oao, uisgn video cj otop sairiml rv nsgiu audio. Sx rof'z xy dahea cpn finish btx apxlmee hq gusin rvy bxvs jn xru lnoflgwoi slgiint.

VideoCam.java

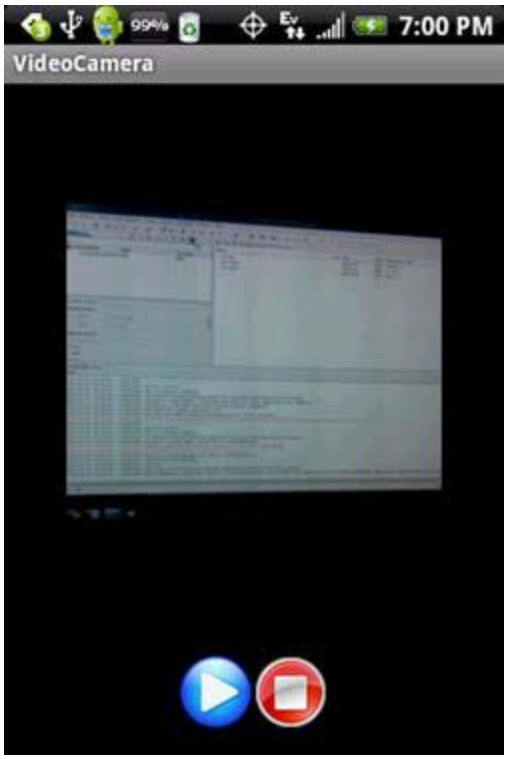
VideoCam.java **public class VideoCam extends Activity implements SurfaceHolder**.**Callback** { **private** MediaRecorder recorder = **null**; **private static final** String OUTPUT_FILE = "/sdcard/uatestvideo.mp4"; **private static final** String TAG = "RecordVideo"; **private** VideoView videoView = **null**; **private** ImageButton startBtn = **null**; **private** ImageButton playRecordingBtn = **null**; **private** Boolean playing = **false**; **private** Boolean recording = **false**; **public void onCreate**(Bundle savedInstanceState) { **super**.onCreate(savedInstanceState); setContentView(R.layout.main); startBtn = (ImageButton) findViewById(R.id.bgnBtn); playRecordingBtn = (ImageButton) findViewById(R.id.playRecordingBtn); videoView = (VideoView)**this**.findViewById(R.id.videoView); **final** SurfaceHolder holder = videoView.getHolder(); holder.addCallback(**this**); holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS); startBtn.setOnClickListener(**new** OnClickListener() { **public void onClick**(View view) { **if**(!VideoCam.**this**.recording & !VideoCam.**this**.playing) { **try** { beginRecording(holder); playing=**false**; recording=**true**; startBtn.setBackgroundResource(R.drawable.stop); } **catch** (Exception e) { Log.e(TAG, e.toString()); e.printStackTrace(); } } **else if**(VideoCam.**this**.recording) { **try** { stopRecording(); playing = **false**; recording= **false**; startBtn.setBackgroundResource(R.drawable.play); }**catch** (Exception e) { Log.e(TAG, e.toString()); e.printStackTrace(); } } } }); playRecordingBtn.setOnClickListener(**new** OnClickListener() { **public void onClick**(View view) { **if**(!VideoCam.**this**.playing & !VideoCam.**this**.recording) { **try** { playRecording(); VideoCam.**this**.playing=**true**; VideoCam.**this**.recording=**false**; playRecordingBtn.setBackgroundResource (R.drawable.stop); } **catch** (Exception e) { Log.e(TAG, e.toString()); e.printStackTrace(); } } **else if**(VideoCam.**this**.playing) { **try** { stopPlayingRecording(); VideoCam.**this**.playing = **false**; VideoCam.**this**.recording= **false**; playRecordingBtn.setBackgroundResource (R.drawable.play); }**catch** (Exception e) { Log.e(TAG, e.toString()); e.printStackTrace(); } } } }); } **public void surfaceCreated**(SurfaceHolder holder) { startBtn.setEnabled(**true**); } **public void surfaceDestroyed**(SurfaceHolder holder) { } **public void surfaceChanged**(SurfaceHolder holder, **int** format, **int** width, **int** height) { Log.v(TAG, "Width x Height = " + width + "x" + height); } **private void playRecording**() { MediaController mc = **new** MediaController(**this**);

```
videoView.setMediaController(mc); videoView.setVideoPath(OUTPUT_FILE); videoView.start(); }
private void stopPlayingRecording() { videoView.stopPlayback(); } private void stopRecording()
throws Exception { if (recorder != null) { recorder.stop(); } } protected void onDestroy() {
super.onDestroy(); if (recorder != null) { recorder.release(); } } private void
beginRecording(SurfaceHolder holder) throws Exception { if(recorder!=null) { recorder.stop();
recorder.release(); } File outFile = new File(OUTPUT_FILE); if(outFile.exists()) {
outFile.delete(); } try { recorder = new MediaRecorder();
recorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);
recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
recorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4); recorder.setVideoSize(320, 240);
recorder.setVideoFrameRate(15); recorder.setVideoEncoder(MediaRecorder.VideoEncoder.MPEG_4_SP);
recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB); recorder.setMaxDuration(20000);
recorder.setPreviewDisplay(holder.getSurface()); recorder.setOutputFile(OUTPUT_FILE);
recorder.prepare(); recorder.start(); } catch(Exception e) { Log.e(TAG, e.toString());
e.printStackTrace(); } } }
```

Cesaeuc sdmq xl qjzr avxq cj msirail re orteh xavb nj prja tphecra, wo kwn'r bseirced vingteehry ruzr'z enhpgaipn. Jl kdh kfvx qcluyik sr rou sovg jn rjpc ilgisnt, gux'ff ernx psrr rj'z eiatlvryle meislp. Ygx itsrf gtnih wx vu jn xrg xvah, dbeisse tsetgin mako fields, aj roz hb ytk cesruaf re zud port rvu amarce qto view, mhdz vjof wo hhj jn rkp milspe mracea application leriear jn jrzu hepartc. Yyo vkrn rdts le rvu xyae rsrp'c jm port rnz cj rpk `beginRecording` method. Lcjtr, ryja method cceshk rx zovm oaty rrcb tygenivhre jc ayrde rv drecro c video file hg amkgin xtpz rruz rvg cearam aj kvlt, ncu cryr rj zsn derrco vrq tpuuto file. Aqnx, jr selyclo sowlfol prk pgiedcrne process oa rk rva dy rqv emraca ktl eordgrinc beefor ilclgan `prepare()` nzy npvr `start()`.

Gauttrlnfenoy, sc wo tndoe jwrg rqx earacm project, hetre'a en czvp dsw rk rcro vygt application jn xrq emulator. Lvt jurz pexmlea, wo'xv shdupe grv application xr z fvaf phone rk oarr drx aracem, cgn cxuy gxr DDMS er onxr xrb file pzrr was dreecrdo nsu er fhcu jr epas. Tge nzc avv ns lxaeemp lx yrv ttopuu, rduacpte rjyw vrp DDMS, kmtl ns HXX Hotv nj figure 10.7.

Photograph of VideoCam application running on an HTC Hero 2.



Without a device to test on, you'll have difficulties debugging your video applications. If you decide to develop a video application, we strongly suggest that you not only obtain an Android device to test on, but that you test every physical device that you hope your application will run on. Although developing Android applications that record data from sensors can be difficult to work with on the emulator, they're relatively straightforward to code, but you need to use a physical Android device to test.

7.ENVIRONMENT SENSORS ANDROID DEVELOPERS.

Most of the android devices have built-in sensors that measure motion, orientation, and various environmental condition. The android platform supports three broad categories of sensors.

- Motion Sensors
- Environmental sensors
- Position sensors

Some of the sensors are hardware based and some are software based sensors. Whatever the sensor is, android allows us to get the raw data from these sensors and use it in our application. For this android provides us with some classes.

Android provides SensorManager and Sensor classes to use the sensors in our application. In order to use sensors, first thing you need to do is to instantiate the object of SensorManager class. It can be achieved as follows.

```
SensorManager sMgr;

sMgr = (SensorManager)this.getSystemService(SENSOR_SERVICE);
```

The next thing you need to do is to instantiate the object of Sensor class by calling the getDefaultSensor() method of the SensorManager class. Its syntax is given below −

```
Sensor light;

light = sMgr.getDefaultSensor(Sensor.TYPE_LIGHT);
```

Once that sensor is declared , you need to register its listener and override two methods which are onAccuracyChanged and onSensorChanged. Its syntax is as follows −

```
sMgr.registerListener(this, light,SensorManager.SENSOR_DELAY_NORMAL);

public void onAccuracyChanged(Sensor sensor, int accuracy) {

}


public void onSensorChanged(SensorEvent event) {

}
```

Getting list of sensors supported

You can get a list of sensors supported by your device by calling the getSensorList method, which will return a list of sensors containing their name and version number and much more information. You can then iterate the list to get the information. Its syntax is given below −

```
sMgr = (SensorManager)this.getSystemService(SENSOR_SERVICE);

List<Sensor> list = sMgr.getSensorList(Sensor.TYPE_ALL);

for(Sensor sensor: list){

}
```

Apart from the these methods, there are other methods provided by the SensorManager class for managing sensors framework. These methods are listed below −

Sr.No	Method & description
1	**getDefaultSensor(int type)** This method get the default sensor for a given type.
2	**getInclination(float[] I)** This method computes the geomagnetic inclination angle in radians from the inclination matrix.
3	**registerListener(SensorListener listener, int sensors, int rate)** This method registers a listener for the sensor
4	**unregisterListener(SensorEventListener listener, Sensor sensor)** This method unregisters a listener for the sensors with which it is registered.
5	**getOrientation(float[] R, float[] values)** This method computes the device's orientation based on the rotation matrix.
6	**getAltitude(float p0, float p)** This method computes the Altitude in meters from the atmospheric pressure and the pressure at sea level.

Example

Here is an example demonstrating the use of SensorManager class. It creates a basic application that allows you to view the list of sensors on your device.

To experiment with this example , you can run this on an actual device or in an emulator.

Steps	Description

1	You will use Android studio to create an Android application under a package com.example.sairamkrishna.myapplication.
2	Modify src/MainActivity.java file to add necessary code.
3	Modify the res/layout/activity_main to add respective XML components.
4	Run the application and choose a running android device and install the application on it and verify the results.

Following is the content of the modified **MainActivity.java**.

```
package com.example.sairamkrishna.myapplication;


import android.app.Activity;

import android.hardware.SensorManager;

import android.os.Bundle;


import android.util.Log;


import android.view.Menu;

import android.view.MenuItem;

import android.view.View;


import android.widget.TextView;


import java.util.List;

import android.hardware.Sensor;
```

```
import android.hardware.SensorManager;


public class MainActivity extends Activity {
   TextView tv1=null;
   private SensorManager mSensorManager;
   @Override


   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);


      tv1 = (TextView) findViewById(R.id.textView2);
      tv1.setVisibility(View.GONE);


      mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
      List<Sensor> mList= mSensorManager.getSensorList(Senso
for (int i = 1; i < mList.size(); i++) {
         tv1.setVisibility(View.VISIBLE);
         tv1.append("\n" + mList.get(i).getName() + "\n" + mList.get(i).getVendor() + "\n" +
mList.get(i).getVersion());
      }
   }


   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
```

```
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
  }

  @Override
  public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.

    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
      return true;
    }
    return super.onOptionsItemSelected(item);
  }
}
```

Following is the modified content of the xml **activity_main.xml**.

In the below code **abc** indicates about the logo of tutorialspoint.com

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
```

```
   android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   android:paddingBottom="@dimen/activity_vertical_margin"
   tools:context=".MainActivity"
   android:transitionGroup="true">

   <TextView android:text="Sensor " android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/textview"
      android:textSize="35dp"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true" />

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Tutorials point"
      android:id="@+id/textView"
      android:layout_below="@+id/textview"
      android:layout_centerHorizontal="true"
      android:textColor="#ff7aff24"
      android:textSize="35dp" />
```

```
   <ImageView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/imageView"
      android:src="@drawable/abc"
      android:layout_below="@+id/textView"
      android:layout_centerHorizontal="true"
      android:theme="@style/Base.TextAppearance.AppCompat" />

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="New Text"
      android:id="@+id/textView2"
      android:layout_below="@+id/imageView"
      android:layout_alignParentBottom="true"
      android:layout_alignParentRight="true"
      android:layout_alignParentEnd="true"
      android:layout_alignParentLeft="true"
      android:layout_alignParentStart="true" />

</RelativeLayout>
```

Following is the content of the **res/values/string.xml**.

```
<resources>
   <string name="app_name">My Application</string>
```

```
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
</resources>
```

Following is the content of **AndroidManifest.xml** file.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.sairamkrishna.myapplication" >
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >

            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>

        </activity>

    </application>
```

</manifest>

Let's try to run our application we just modified. I assume you had created your **AVD** while doing environment setup. To run the app from Android studio, open one of your project's activity files and click Run icon from the toolbar. Android studio installs the app on your AVD and starts it and if everything is fine with your setup and application, it will display following Emulator window −