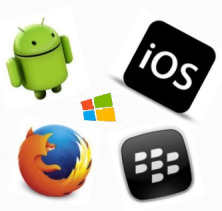
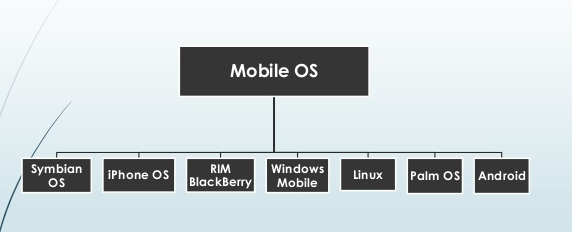
**DATE:** **INTRODUCTION** **TO ANDROID STUDIO**

1. ***MOBILE DEVICE OPERATING SYSTEM***

A mobile operating system, also called a mobile OS, is an operating system that is specifically designed to run on mobile devices such as mobile phones, smartphones, PDAs, tablet computers and other handheld devices.

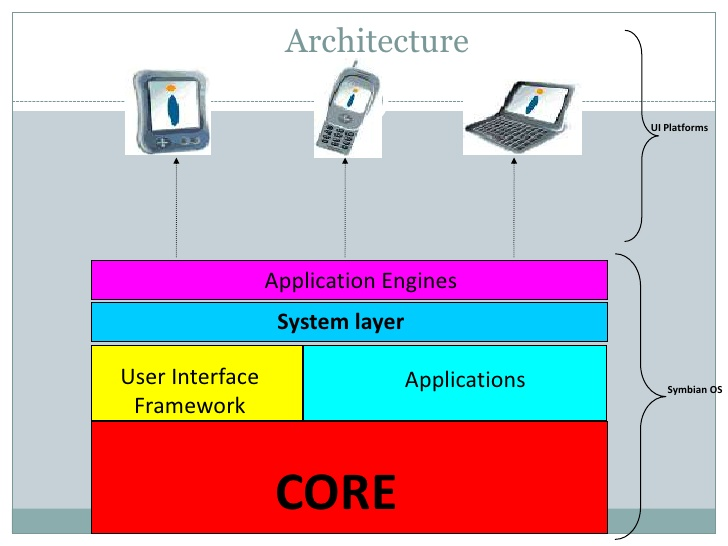
Smart phones are being used not only to make phone calls, but also to carry out a host of other operations such as to make *video calls*, *send multimedia messages, take pictures, paly media files, browse world wide web(WWW), run web applications, etc.*

1. ***A SURVEY OF COMMERCIAL MOBILE OPERATING SYSTEM***

****

**2.1 Symbian OS**

* Symbian OS is a real time, multitasking, pre-emptive, 32-bit operating system that runs on ARM-based processor designs. The design of the Symbian OS is microkernel-based.
* Symbian OS is a multi-tasking operating system which addresses the mass-market and designed for simplifying the work load.
* It is an open platform allows the installation of third party software to enhance the platform.
* **Symbian OS** is officially the [property of **Nokia**](http://mobilephones.pk/nokia). It means that any other company will have to take permission from Nokia before using this operating system.
* Still Symbian is widely used in **low-end phones** but the demand rate has been continuously decreasing. By upgrading Symbian mobile OS, Nokia has made it capable to run smartphones efficiently.
* **Architecture of Symbian OS:**



1. **Core:** Symbian OS core is common to all devices i.e. Kernel, file server, memory management & device drivers.
2. **System Layer:** It provides communication and computing services such as TCP/IP, SMS & database management.
3. **Application Engines:** It’s above the system layer, enabling the software developers to create user interface to data.
4. **User Interface Software:** It can be made or licensed by manufactures.
5. **Applications:** are slotted above the user interface & pre-loaded in the Symbian OS mobile phone.

**Flavours:**

1. **Serial 60:**

* The serial-60 platform is the leading smartphone platform in the world.
* The large sized colour screen, easy-to-use interface and an extensive suite of applications make it well-suited to support advanced features such as rich content downloading and MMS (Multimedia Messaging Service).
* It is mainly used on Nokia’s smartphones and Samsung handsets.

1. **UIQ Interface:**

* UIQ (User Interface Quartz) is a software package developed by UIQ technology for Symbian OS.
* This is a Graphical User Interface layer that provides capabilities for third-party application developers to develop applications and effortlessly create user interfaces.

**Features**

* It supports no of communication and networking protocols including TCP, UDP, PPP, DNS, FTP, WAP, etc.
* For personal area networking it supports Bluetooth, Infrared and USB connectivity.
* It supports pre-emptive multitasking scheduling and memory protection.
* CPU is switched into a low-power mode when the application is not responding to an event.
* It is optimized for low-power and memory requirements. Applications and the OS itself, follow an object-oriented design paradigm.
* Symbian is microkernel-based OS.
* Carbide is an Integrated Development Environment (IDE) toolkit is available for C++ application development on Symbian OS. It works as an Eclipse plug-in and contains editor, compiler, emulator, libraries and header files required for Symbian OS development.

**Advantages:**

1. It is an open platform based on C++, which is easy to configure.
2. Fully multitasking.
3. Small with lots of in-built features.
4. Symbian also been shown to perform well on a mobile phone and is designed for multimedia, almost all compatible applications can be installed.
5. Symbian is the designation of Nokia is no doubt the quality.
6. Symbian is an OS that can be modified all that is therein.
7. Symbian can be formed or changed appearance to taste.
8. Symbian is able to compete with competitors.

**Disadvantages:**

1. System configuration is similar to a computer, Symbian susceptible to the virus.
2. It is not available for PCs

**2.2 Android**

**Definition:**

Android is an open source and Linux-based **Operating System** for mobile devices such as smartphones and tablet computers. Android was developed by the *Open Handset Alliance*, led by Google, and other companies. Java language is mainly used to write the android code even though other languages can be used.

**Goal of Android:**

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

* September 20th, 2008 was the date when **Google** released the first **Android OS** by the name of ‘**Astro**’. After sometime next upgraded versions **‘Bender’** and **‘Cupcake’** were also released.
* The other releases are **Donut, Éclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich and Jelly Bean.** **Marshmallow** (Android 6.0) is so far the latest **Android** version from **Google**. Since the platform is not closed like **iOS**, there are too many [great **Android apps**](http://www.shoutmeloud.com/top-free-android-apps.html) built by developers.
* Just after stepping into the **smartphone** and **tablets market** **Android** gained immense popularity due to its beautiful appearance and efficient working. Many new features were introduced which played a significant role in **Android’s** success.

**Features**

* Android provide the ability to seamlessly use either a phone-based keyboard or a touch screen.
* Android providing a built-in full web browser capable rendering full web pages not just small mobile versions.
* **Android SDK** works in **Eclipse** environment

## Categories of Android applications:

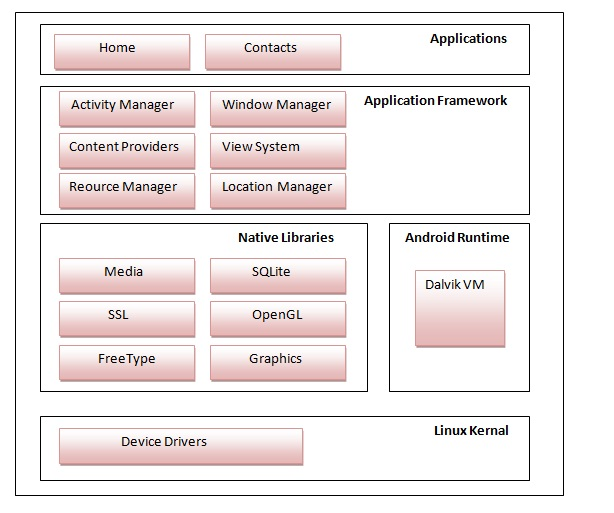
There are many android applications in the market. The top categories are:

* Entertainment
* Tools
* Communication
* Productivity
* Personalization
* Music and Audio
* Social
* Media and Video
* Travel and Local etc.

**Architecture of Android:**

**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



## 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.

* **Android kernel** has been developed based on a version of **Linux kernel**.
* Based on the **Linux kernel code**, Android implements its own *device drivers, memory management, and process management and networking functionalities.*
* **Android** is *multitasking* and *allows application to run concurrently.*
* For example, it is possible to hear music and read or write an email at the same time. This layers is the one that is commonly used by cell phone users.

## 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.

* The available libraries are written using multiple languages such as C and C++. These are called through a java interface.

## 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.

* **Dalvik virtual machine: Dalvik** *translates**java application program into**machine code* of the mobile device and executes it by invoking the operating system.
* A developed application is compiled to ARM native code and installed using Android **native software development kit (SDK)**
* Dalvik is not traditional JVM, but custom VM designed to run multiple instances efficiently on a single device.
* Every **Android** application runs its own process with its own instance of the **Dalvik virtual machine**.

## 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.

* Used to implement standard structure for different applications.
* It essentially provides set of services that an application programmer can make use of
* The services include managers and content providers
* Content providers enable applications to access data from other applications

## 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.

* **Android OS** comes with a set of basic application such as web browser, email client, SMS program, maps, and calendar and contacts repository management programs.
* All these application are written using java programming language **J2ME.**
* Android application do not have control over their own priorities
* Can manage the resources to ensure device responsiveness and kill an application when needed.

**Advantages of Android OS:**

1. It is an Open Source
2. Supports 2D and 3D graphics
3. Supports multiple languages
4. Java support – enables developers to enhance more features.
5. Faster web browser
6. It supports different media formats such as MP4, 3GP, MPEG4, MIDI. There is no need to convert from one format to another.
7. Additional hardware support
8. Video calling
9. Larger developer and community reach
10. Increased marketing
11. Inter App integration
12. Reduced cost of development
13. Rich Development Environment

**Disadvantages of Android OS:**

1. **Slow response - Slow** response of the android in opening apps when compared with different platforms.
2. **Heat –** compared to other operating systems android makes use of processed very efficient. This makes processor to get heat.
3. **Advertisement -** When we use an android app we encounter several adds in between application use, because anyone can make add by inserting some logic in the app program and can interfere in into the phones information.
4. **Continuous Internet connection**

**2.3 Apple iOS**

**Definition:**

**iOS** (originally **iPhone OS**) is a mobile operating system created and developed by Apple Inc. and distributed exclusively for Apple hardware. It is the operating system that presently powers many of the company's mobile devices, including the iPhone, iPad, and iPod touch. It is the second most popular mobile operating system in the world by sales, after Android.

* **iOS** was introduced in **29th June 2007** when the first **iPhone** was developed. Since then **iOS** has been under gone many upgrades and currently the latest one is the **iOS 9.**
* **Apple** has still not allowed any other manufacturer to lay hands on its operating system. Unlike **Android,** **Apple** has more concentrated on the performance along with appearance.
* This is the reason that the basic appearance of **iOS** is almost the same as it was in **2007.** Overall it is *very user-friendly* and is one of the mobile best operating systems in the world.

**Features:**

* **iOS** is a closed and proprietary operating system fully owned and controlled by **Apple**
* It provide several innovative feature that grabbed market attention.
* For example user interactions with **OS** include gestures such as *swipe,* *tap,* *pinch* and *reverse pinch.*
* **iOS** include internal accelerometers used by some applications for shaking the device as the undo command, rotating the device in three dimensions to switch the display mode from portrait to landscape, etc.

**Advantages of iOS:**

1. Performance is awesome
2. Generates less heat when compared to Android
3. Best gaming experience
4. Vast number of applications
5. Suits for business and gaming
6. Excellent UI and fluid responsive
7. Excellent security
8. Multitasking
9. Excellent for media entertainment
10. Multi-language support
11. Apple Pay Support
12. Fits in the hands (Not talking about iPhone 6S Plus)
13. Fingerprint scanning gives tough security
14. Excellent camera quality even though megapixels are less
15. Developers can design apps because less number of models

**Disadvantages of iOS:**

* Not flexible only supports iOS devices
* Not open source
* The main disadvantages of using iOS are costly Apps and no widget support
* Devices are very costly
* Applications are very large when compared to other mobile platforms
* Supports only single SIM
* Very addictive, once if you used iDevices you can’t switch back to another
* Battery performance is very poor on 3G

**2.4 Windows OS**

**Definition:**

A **mobile operating system** for **smartphones** and **mobile** devices from Microsoft based on the **Windows** CE kernel and designed to look and operate similar to desktop versions of Microsoft **Windows**. **Windows Mobile** competes in the **mobile OS** market with Apple's iOS, Google's Android, BlackBerry **OS**, Symbian and others.

* **Windows OS** has also been used in mobile phones, but normal mobile phone users find it a bit difficult to operate it but at the same time it was very popular among people who were used to it.
* This designed by **Microsoft** and **Nokia.**
* The latest **Windows** release by **Microsoft** is known as **Windows 7** which has gained immense popularity among all kind of users. With its colorful and user-friendly interface, it has given **Windows OS** a new life and is currently in demand all over the world.
* Another reason behind its success is that this latest **OS** is used in very powerful devices made by **Nokia.**

**Features**

* The **Graphics / window / event manager (GWE)** component handles all input and output.
* Provides **virtual memory management.**
* Supports **security** through the provision of **cryptographic library.**
* Application development is similar to that in the **Win32 environment.**
* At present, it does not provide *true multitasking,* the Microsoft support true multitasking in the future versions of **Windows OS.**

**2.5 Palm OS (Garnet OS)**

* **Palm OS** was developed by ***Palm Inc in 1996*** especially for **PDAs** (Personal Digital Assistance). **Palm OS** was designed to work on touchscreen **GUI.**
* Some Years later it was upgraded and was able to support **smartphones.**
* Unfortunately, it could not make a mark on the market and currently is not being used in any of the latest top devices.

**Features**

* It is essentially *simple single-tasking operating system.*  As a result, only one application can run at a time. For example, if you are on voice communication, you cannot use the calculator, or read an SMS.
* It has an elementary memory management system. To keep the **OS** *small and fast,*  Palm OS does not isolate the memory areas of applications from each other. Consequently, any misbehaving application can crash the system.
* **Palm** supplies ***Palm emulator***, which emulate the Palm hardware on a PC. This allows Palm programs to be developed and debugged on a PC before being run on the Palm hardware.
* It supports a *handwriting recognition-based system* for user input.
* It supports the facility called ***HotSync*** technology for data synchronization with desktop computers.
* It supports *sound playback* and *recording capabilities.*
* It incorporates a very simple and rudimentary security model in which a device can be locked by password.
* The different interfaces supported include Serial Port/USB, infrared, Bluetooth and Wi-Fi connections.
* It uses a proprietary format to store calendar, address, task and note entries and yet are accessible by third-party applications.

**2.6 Blackberry OS**

* **Blackberry OS** is the property of[**RIM**](http://www.rim.com/) (Research In Motion) and was first released in **1999**.
* **RIM** has developed this operating system for its **Blackberry line** of smartphones.
* The interface style, as well as the Smartphone design, is also different having a trackball for moving on the menu and a qwerty keyboard.

**Features**

It support instant mailing while maintaining a high level of security through on-device hardware based message encryption.

1. **DEVELOPING ANDROID APPLICATIONS USING ECLIPSE IDE AND ANDROID STUDIO:**

# 3.1: Step-by-step guide to Android Application development with Eclipse:

### Set up the Android development environment

Setting up Android development environment takes some time at first. It helps to make sure you don’t do anything wrong to save yourself from the agony of doing the whole process again.

You're required to have Windows XP or later, or Mac OS X 10.5.8 or a later version to start Android application development process. Then, there are four tools that you will need and they are available on the Internet for free:

1. Java JDK5 or JDK6
2. Android SDK
3. Eclipse IDE for Java Developers (optional)
4. Android Development Tools (ADT) Eclipse Plugin (optional)

### Step 1: Setup Java Development Kit (JDK)

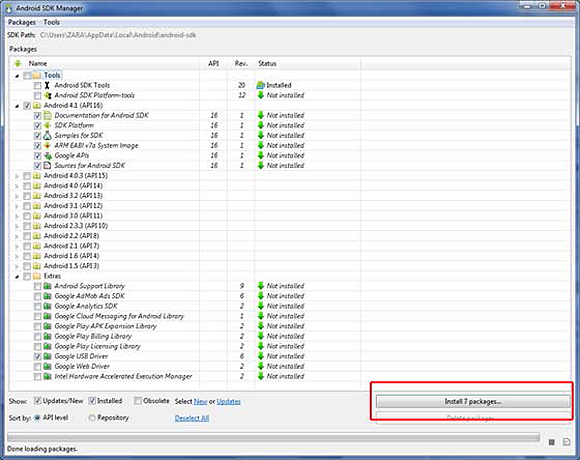
You can download the JDK and install it, which is pretty easy. After that, you just have to set PATH and JAVA\_HOME variables to the folder where you have **java** and **javac.**

**Note for Windows Users:** If you installed the JDK in C:\jdk1.6.0\_15 then you will have to add the following two lines in your C:\autoexec.bat file.

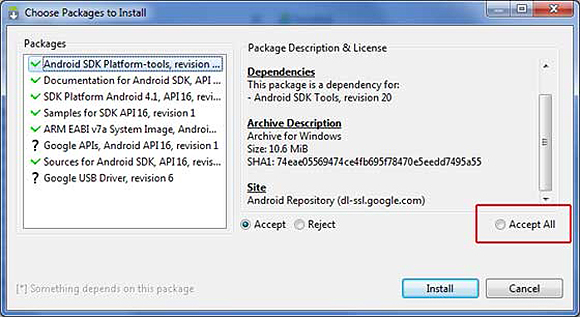
set PATH=C:\jdk1.6.0\_15\bin;%PATH%  
set JAVA\_HOME=C:\jdk1.6.0\_15

### Step 2: Configure Android SDK

After you have successfully installed the Android SDK, it is time to configure it. After installing the Android SDK, you will get a window like this:



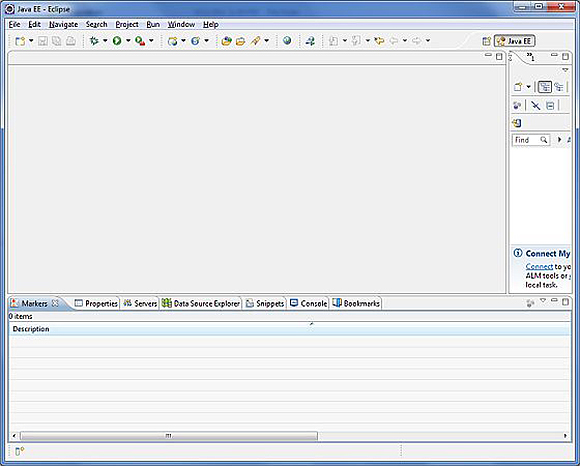
Just de-select the **Documentation for Android SDK** and **Samples for SDK** packages if you want to reduce the installation size and time. Click on **Install 7****packages** to continue with the installation. You will get a dialogue box like this:



It will take some time to install, so in the meanwhile you could do some other task to kill the time. How long will it take? Well, it depends on the speed of your Internet connection. Once it is done, you can close the SDK manager.

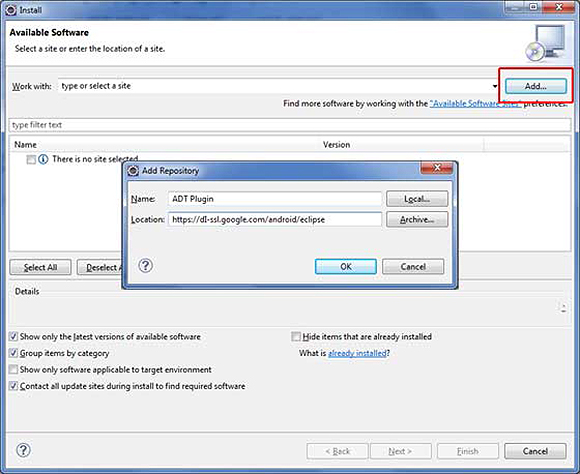
### Step 3: Setup Eclipse IDE

Install the latest version of Eclipse. After successful installation, it should display a window like this:

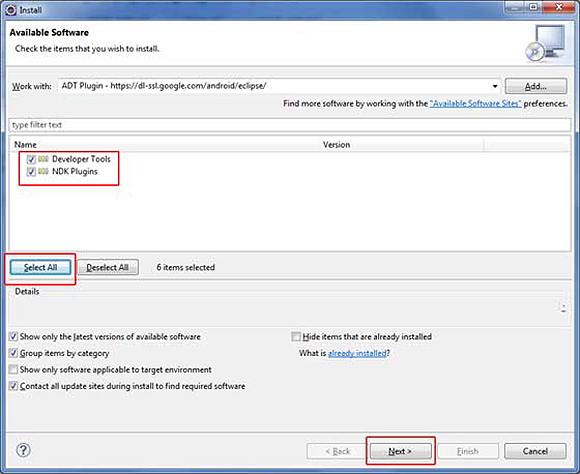


### Step 4: Setup Android Development Tools (ADT) Plugin

Here you will learn to install the Android Development Tool plugin for Eclipse. To do this, you have to click on **Help > Software Updates > Install New Software**. This will display the following dialogue box.

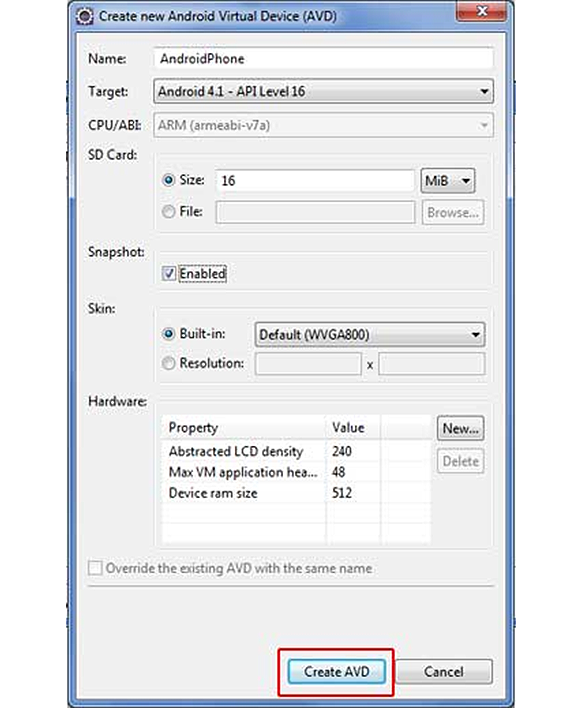


Just click on the Add button as shown in the picture and add **https://dl-ssl.google.com/android/eclipse/** as the location. When you press OK, Eclipse will start to search for the required plug-in and finally it will list the found plug-ins.



### Step 5: Create Android Virtual Device

The last step is to create Android Virtual Device, which you will use to test your Android applications. To do this, open [Eclipse](https://eclipse.org/) and Launch Android AVD Manager from options **Window > AVD Manager** and click on **New** which will create a successful Android Virtual Device. Use the screenshot below to enter the correct values.



You have successfully created Android Application Development environment. You are now ready to create a simple Rock Paper Scissors Android App.

Before we write the code, you need to know how to take input from the user. The most efficient way of taking input from the user is to use the Scanner class, which is found in the java.io package as it is just a two-step process.

Scanner scanner =  new Scanner(System.in);

String input   =  scanner.next();

/\* or String input   =  (new Scanner(System.in)).next(); \*/

I recommend using the [Scanner class](http://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html) because it works equally well on command line as well as with Eclipse that we will use to make the Rock Paper Scissors app.

### Calling the Java code in Eclipse

We will save you the trouble of writing the java code for a simple Rock Paper Scissors app and use can use the code below, but you are free to use your own code if you prefer.

// Android Rock Paper Scissors App

// Rock\_Paper\_Scissors

// \*\*\*\*\*\*\*\*\*\*\*\*\*

import java.util.Scanner;

import java.util.Random;

public class Rock

{

public static void main(String[] args)

{

String personPlay; //User's play -- "R", "P", or "S"

String computerPlay = ""; //Computer's play -- "R", "P", or "S"

int computerInt; //Randomly generated number used to determine

//computer's play

String response;

Scanner scan = new Scanner(System.in);

Random generator = new Random();

System.out.println("Hey, let's play Rock, Paper, Scissors!\n" +

"Please enter a move.\n" + "Rock = R, Paper" +

"= P, and Scissors = S.");

System.out.println();

//Generate computer's play (0,1,2)

computerInt = generator.nextInt(3)+1;

//Translate computer's randomly generated play to

//string using if //statements

if (computerInt == 1)

computerPlay = "R";

else if (computerInt == 2)

computerPlay = "P";

else if (computerInt == 3)

computerPlay = "S";

//Get player's play from input-- note that this is

// stored as a string

System.out.println("Enter your play: ");

personPlay = scan.next();

//Make player's play uppercase for ease of comparison

personPlay = personPlay.toUpperCase();

//Print computer's play

System.out.println("Computer play is: " + computerPlay);

//See who won. Use nested ifs

if (personPlay.equals(computerPlay))

System.out.println("It's a tie!");

else if (personPlay.equals("R"))

if (computerPlay.equals("S"))

System.out.println("Rock crushes scissors. You win!!");

else if (computerPlay.equals("P"))

System.out.println("Paper eats rock. You lose!!");

else if (personPlay.equals("P"))

if (computerPlay.equals("S"))

System.out.println("Scissor cuts paper. You lose!!");

else if (computerPlay.equals("R"))

System.out.println("Paper eats rock. You win!!");

else if (personPlay.equals("S"))

if (computerPlay.equals("P"))

System.out.println("Scissor cuts paper. You win!!");

else if (computerPlay.equals("R"))

System.out.println("Rock breaks scissors. You lose!!");

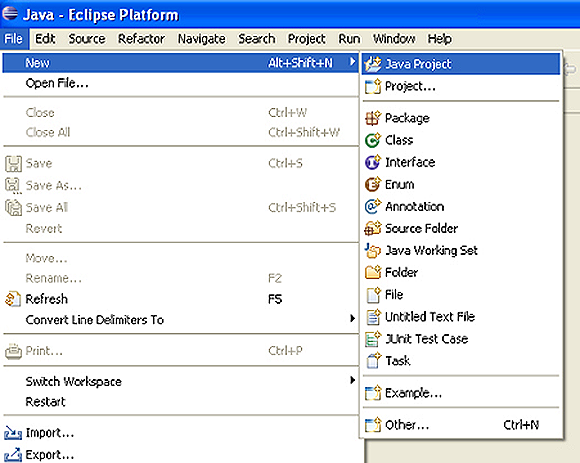
else

System.out.println("Invalid user input.");

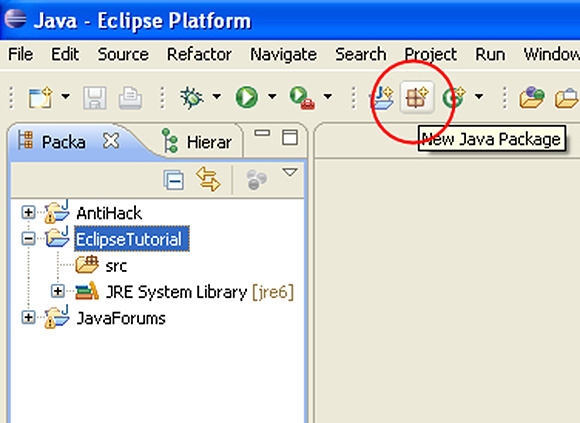
}

}

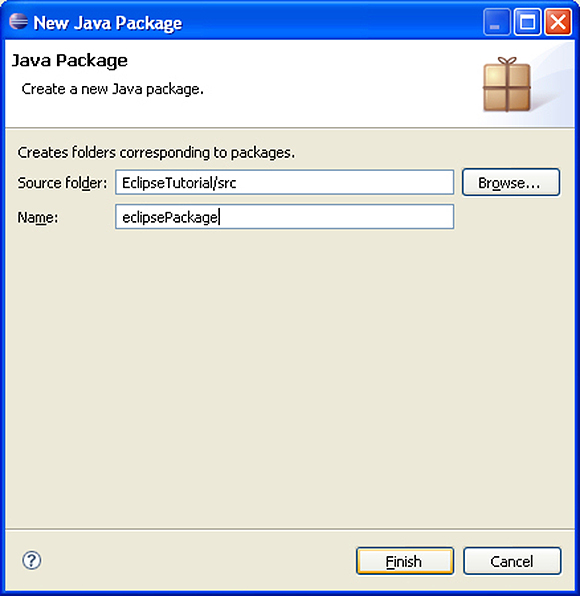
Now calling the Java code in Eclipse is a tricky process and it helps to pay attention while you're doing this. Launch Eclipse and click **File** > **New** > **Java Project**



When the **Create a Java Project** box appears, it's time to give your project a name. Click on Finish to save it and it should appear in the Package Explorer window. Then we are supposed to add a package which will contain all our package files. Click on **New Java Package** icon to do this, as shown in the screenshot below.



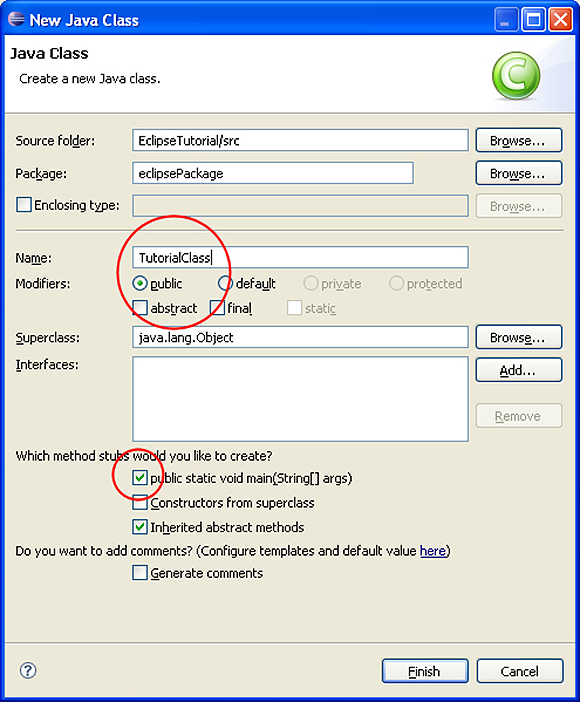
Name your project and then click **Finish**.



Now we need to add a Java Class, which is as easy as adding a Java Package.



After giving it a name, make sure that the following options are checked:



After you create a new class, it will show up in the Work Space where you can write or copy the code.

Congratulations! You have just finished writing your first Java Application in Eclipse. It wasn’t that difficult was it?

Now you need to build the application and to do this, Right Click on your Android Project and select **Android Tools->Export Signed Application Package**. After selecting the export button, select **Create new keystore** and it will take you to the location where you want to save it, so give it a name and save it. Fill in all the required fields that are self-explanatory and save it. You have successfully exported the apk file to your computer and you can test the app it on your android device.

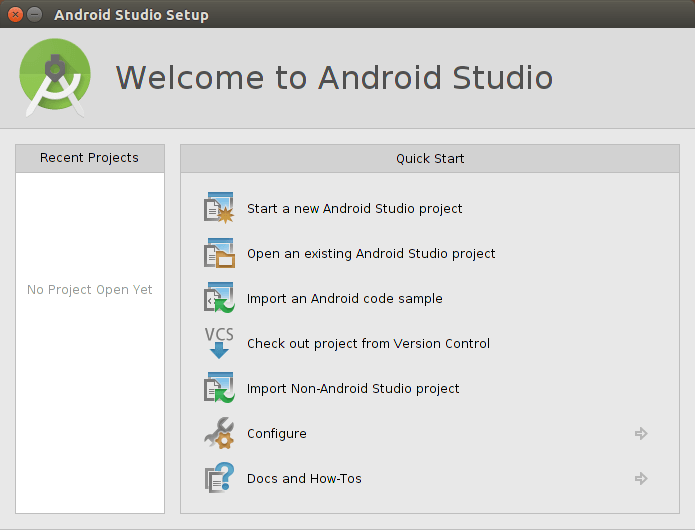
# 3.2: Step-by-step guide to Android Application Development using Android Studio:

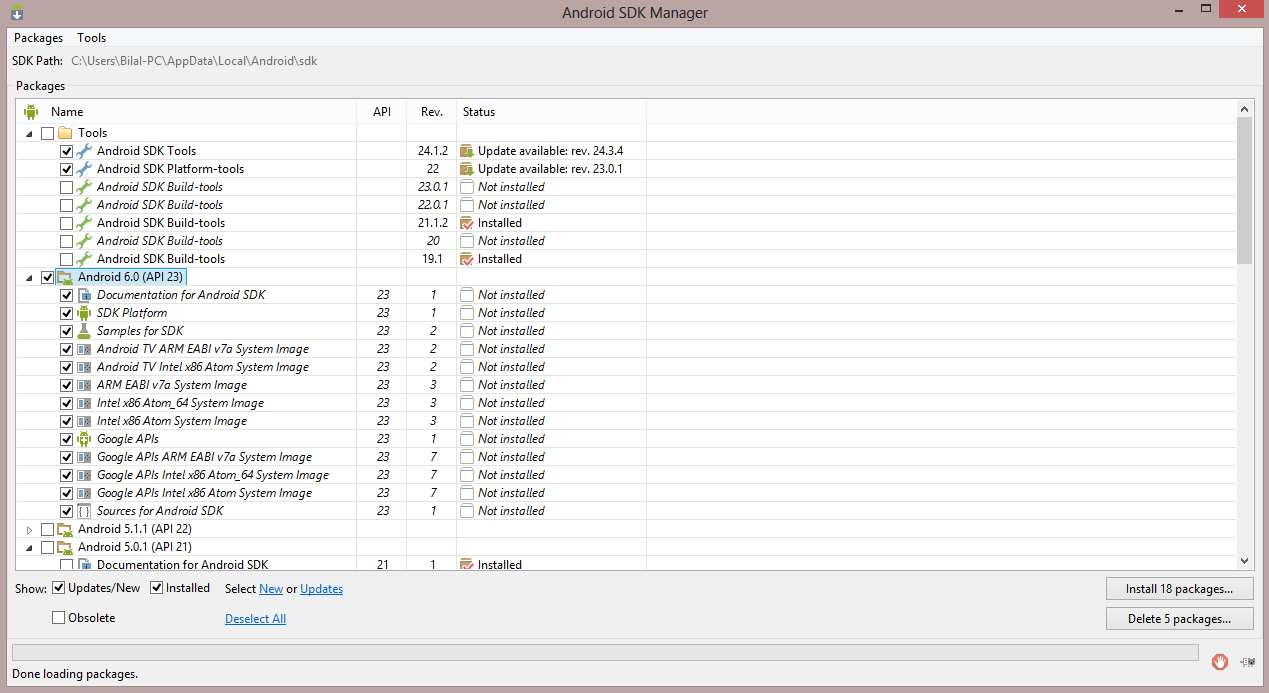
## Steps to follow for developing your first android app:

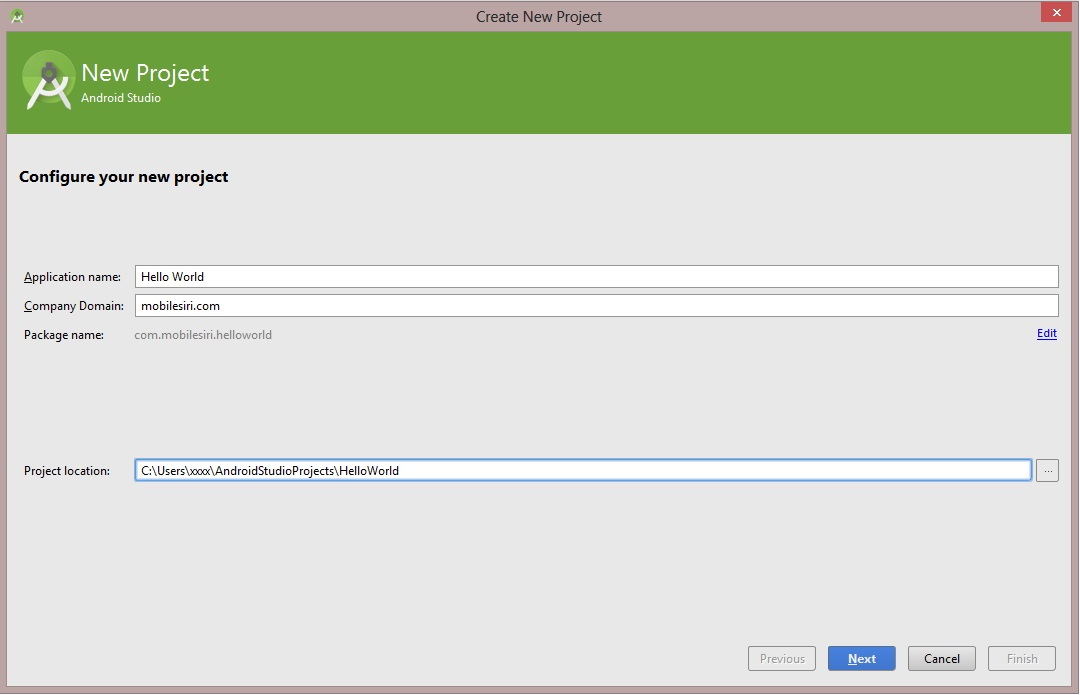
**Create android development environment**   
1. Download JDK (Java Development Kit 7) required for Android Studio  
<http://www.oracle.com/technetwork/java/javase/downloads/index.html>  
(Search out for valid link if not working)

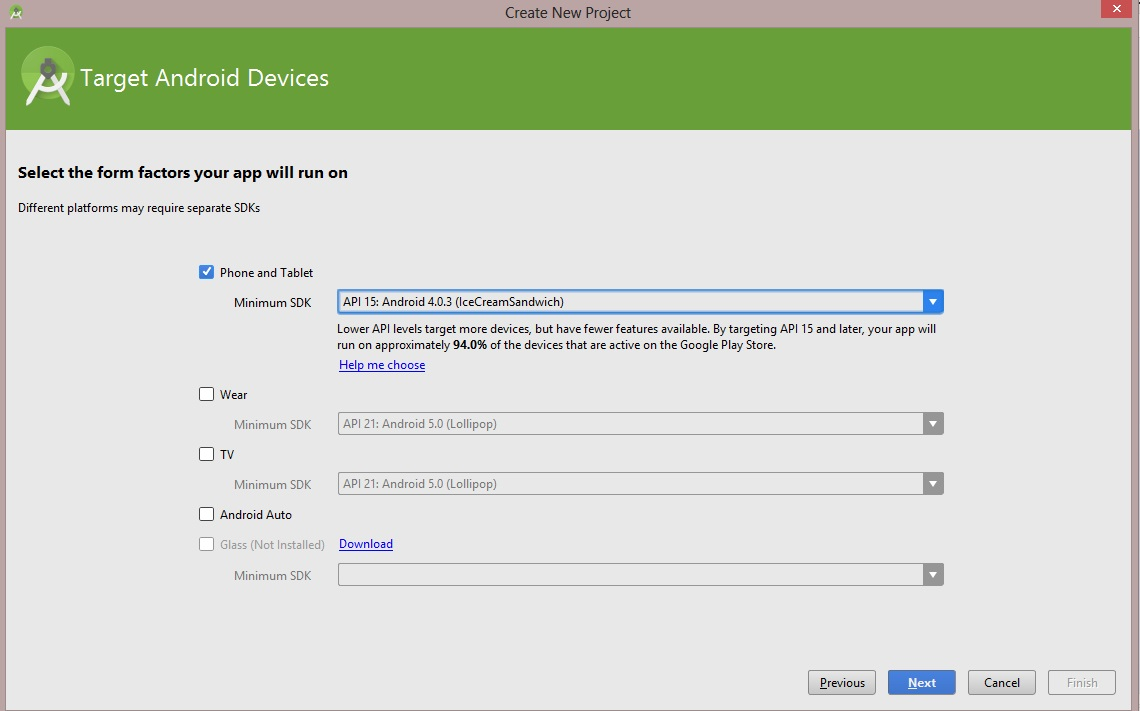
2. Download &install (require as development IDE ) latest and upgraded Android Studio  
<https://developer.android.com/training/basics/firstapp/index.html>  
(Web page gets changed often search out if not working)  
Before follow step three first install and open Android Studio, Android SDK (Software development Kit) could be download and install using SDK Manager. Click **File >> Settings then under Appearance & Behavior select Android SDK under System Settings** node.  
<https://developer.android.com/tools/help/sdk-manager.html>

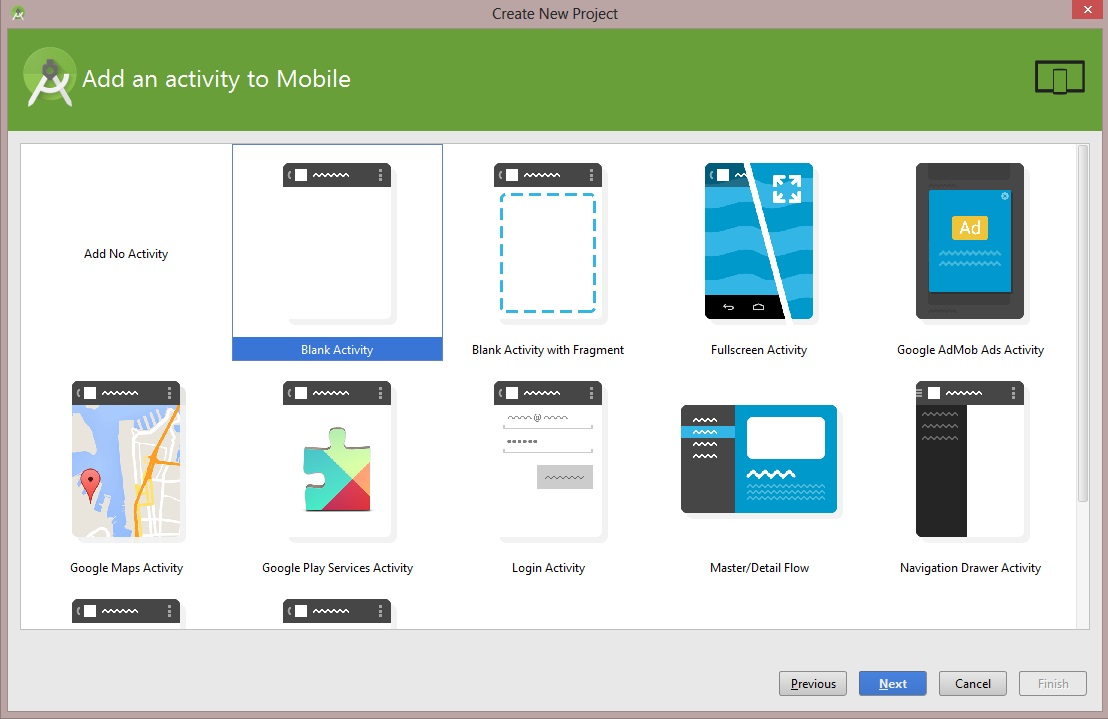
3. Now, open Android Studio you will see welcome screen

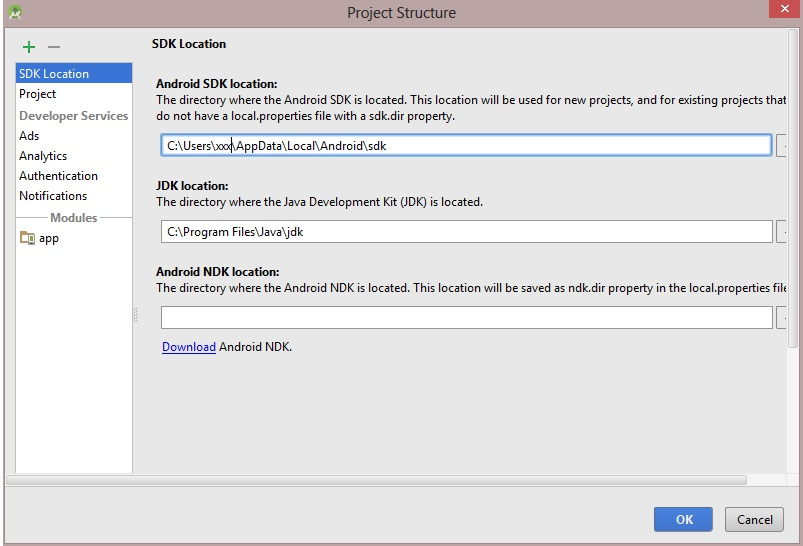
[](http://cdn.mobilesiri.com/wp-content/uploads/2015/09/start-androidstudio-mobilesiri.png)

If you don’t follow set two click **Configure> SDK Manager**  
[](http://cdn.mobilesiri.com/wp-content/uploads/2015/09/configAndroidSdk-mobilesiri.png)

4. Click on Start a new Android Studio in Welcome Screen. Now provide your **application/project name, company domain**, the package name is unique for all application on Google Play.  
[](http://cdn.mobilesiri.com/wp-content/uploads/2015/09/helloworld-mobilesiri-android.jpg)

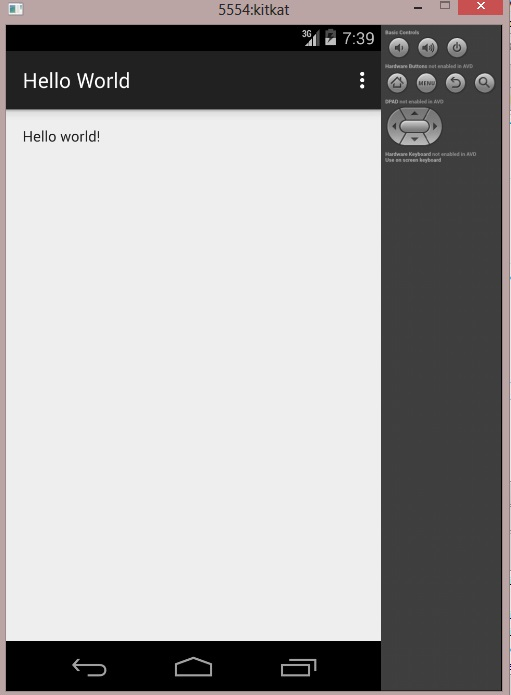
5. Now click NEXT select **Phone and Tablet** in **Select the form factors your app will run**. Select at least **API 15** for **Minimum SDK**, you can choose any.  
[](http://cdn.mobilesiri.com/wp-content/uploads/2015/09/6.jpg)

6. Click next, select **Blank Activity** under **Add an activity to Mobile**  
[](http://cdn.mobilesiri.com/wp-content/uploads/2015/09/addActivity-mobilesiri-androidstudio.jpg)

7. Click next, and click finish. You may also change default but not require.  
8. Now, we have done with our all basic default file for android application development project.  
9. If you follow step two then go to **File -> Project Structure -> SDK Location**. Also, add JDK path if not already added  
[](http://cdn.mobilesiri.com/wp-content/uploads/2015/09/10.jpg)

We are ready with our first Android Application which simply show Hello World on screen. Now, to run this application we need an Android device or alternately an emulator which runs on our computer. Open AVD manager from icon or **Tools > Android > AVD Manager**  
[Description: avdManager](http://cdn.mobilesiri.com/wp-content/uploads/2015/09/avdManager.png)

Start an emulator if already present  
[Description: emulatorStart](http://cdn.mobilesiri.com/wp-content/uploads/2015/09/emu.jpg)  
Or create a new one.

Now, you simply run your project on your virtual device.  
[](http://cdn.mobilesiri.com/wp-content/uploads/2015/09/emulator.jpg)

**Result:**

Thus the android studio has been installed successfully.

**EX.NO:1 APPLICATION USING GUI COMPONENTS, FONT AND COLORS**

**DATE:**

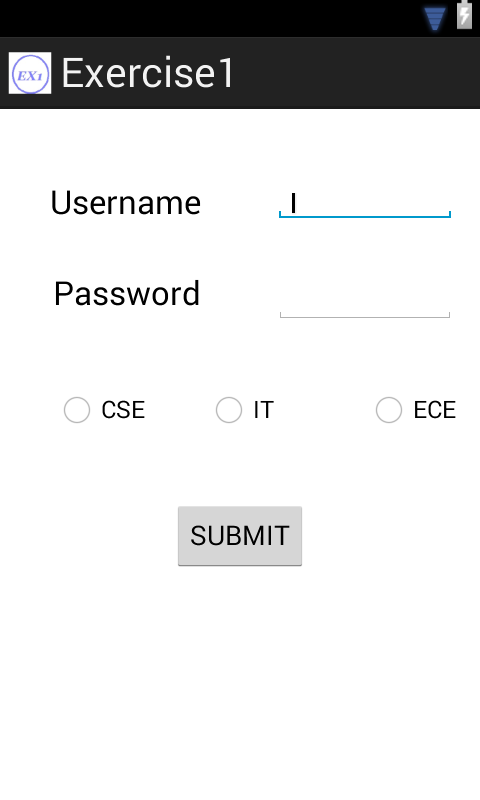
**AIM:**

To develop a mobile application to demonstrate the use of GUI components, Font, Colours.

**STEPS:**

1. Create a new android application by choosing the Application name, minimum required SDK to be API19: Android4.4 (Kitkat) and create blank activity.

2. Design the activity with the GUI components as shown in the graphical layout below. Pick the GUI components from the Form Widget and place them in the graphical layout as required.



3. Create the string resources for the corresponding GUI components in the string.xml file and save it.

4. Customize the look and feel by using the properties, from the properties window or open the activity\_main.xml file and write property rules (to set the properties like background of layout, text size, text color, typeface) for the respective components and save it.

5. To use custom fonts:

5.1: Download the requited font from the internet.

5.2: Copy the downloaded font into the “assets” folder of the current android project.

5.3: Write the necessary java code in “MainActivity.java” file as follows:

Typeface obj=Typeface.createFromAsset(getAssets(), “font.ttf”);

Component c\_obj=(Component)findViewById(R.id.c\_id);

C\_obj.setTypeface(obj);

6. Create an emulator:

6.1: Choose “window” from the menu bar.

6.2: Select “Android Virtual Device Manager”.

6.3: In the dialog box, type AVD Name, choose the required device from the list, give the memory, heap and SD card size. Then click “OK”.

6.4: Start and launch the emulator.

7. Right Click the Application name in the package explorer window and click Run as Android Application. After successful compilation, the projectname.apk file will be launched and executed on the emulator.

**SOURCE CODE:**

**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"*

**android:background=*"#d0d69f"***

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=*"17dp"*

android:layout\_marginTop=*"33dp"*

android:text=*"@string/uname"*

**android:textColor=*"#ff0000"***

**android:textSize=*"30sp"***

android:textAppearance=*"?android:attr/textAppearanceLarge"* />

<TextView

android:id=*"@+id/textView2"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignRight=*"@+id/textView1"*

android:layout\_below=*"@+id/textView1"*

android:layout\_marginTop=*"34dp"*

android:text=*"@string/pword"*

**android:textColor=*"#ffff00"***

**android:textSize=*"30sp"***

android:textAppearance=*"?android:attr/textAppearanceLarge"* />

<Button

android:id=*"@+id/button1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_below=*"@+id/radioButton3"*

android:layout\_centerHorizontal=*"true"*

android:layout\_marginTop=*"44dp"*

android:text=*"@string/submit"*

**android:textColor=*"#0000ff"*** />

<RadioButton

android:id=*"@+id/radioButton1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignLeft=*"@+id/textView2"*

android:layout\_below=*"@+id/editText2"*

android:layout\_marginTop=*"48dp"*

android:text=*"@string/cse"*

**android:textColor=*"#a00000"***

**android:typeface=*"serif"***/>

<RadioButton

android:id=*"@+id/radioButton3"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignBaseline=*"@+id/radioButton1"*

android:layout\_alignBottom=*"@+id/radioButton1"*

android:layout\_alignRight=*"@+id/editText2"*

android:text=*"@string/ece"*

**android:textColor=*"#fa8072"***

**android:typeface=*"monospace"*** />

<RadioButton

android:id=*"@+id/radioButton2"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_above=*"@+id/button1"*

android:layout\_centerHorizontal=*"true"*

android:text=*"@string/it"*

**android:textColor=*"#ff0000"***

**android:typeface=*"sans"***/>

<EditText

android:id=*"@+id/editText1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignBottom=*"@+id/textView1"*

android:layout\_alignTop=*"@+id/textView1"*

android:layout\_toRightOf=*"@+id/radioButton2"*

android:ems=*"10"*

android:inputType=*"textPersonName"* >

<requestFocus />

</EditText>

<EditText

android:id=*"@+id/editText2"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignBaseline=*"@+id/textView2"*

android:layout\_alignBottom=*"@+id/textView2"*

android:layout\_alignLeft=*"@+id/editText1"*

android:layout\_alignParentRight=*"true"*

android:ems=*"10"*

android:inputType=*"textPassword"* />

</RelativeLayout>

**strings.xml**

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

<resources>

<string name=*"app\_name"*>Exercise1</string>

<string name=*"action\_settings"*>Settings</string>

<string name=*"hello\_world"*>Hello world!</string>

<string name=*"uname"*>Username</string>

<string name=*"pword"*>Password</string>

<string name=*"cse"*>CSE</string>

<string name=*"it"*>IT</string>

<string name=*"ece"*>ECE</string>

<string name=*"submit"*>SUBMIT</string>

</resources>

**MainActivity.java**

import android.os.Bundle;

import android.app.Activity;

import android.graphics.Typeface;

import android.view.Menu;

import android.widget.TextView;

public class MainActivity extends Activity {

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

**Typeface font=Typeface.*createFromAsset*(getAssets(),"insomnia.ttf");**

**TextView bt=(TextView)findViewById(R.id.*button1*);**

**bt.setTypeface(font);** };

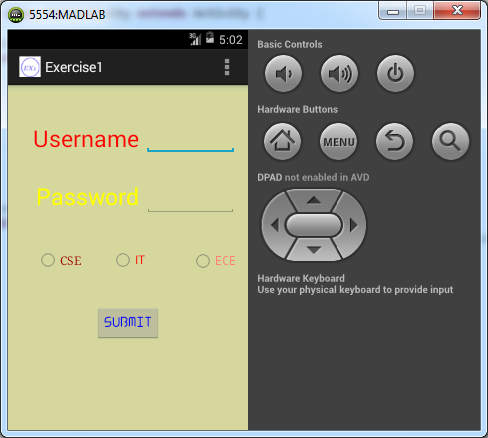
public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.*main*, menu);

return true;} }

**OUTPUT:**



**RESULT:**

Thus the mobile application to demonstrate the use of GUI components, font and colours has been developed, launched and executed successfully.

**EX.NO:2 APPLICATION USING LAYOUT MANAGERS AND**

**DATE: EVENT LISTENERS**

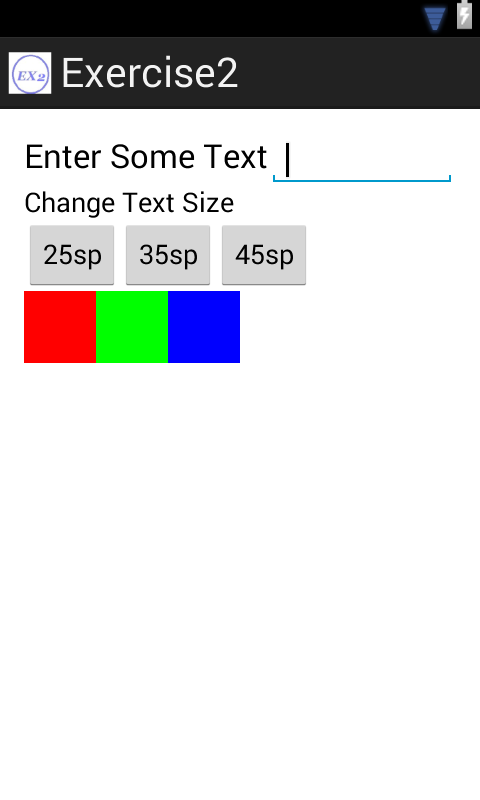
**AIM:**

To develop a mobile application to demonstrate the use of Layout Managers and Event Listeners.

**STEPS:**

1. Create a new android application by choosing the Application name, minimum required SDK to be API19: Android4.4 (Kitkat) and create blank activity.

2. Design the activity with the GUI components as shown in the graphical layout below. Pick the GUI components from the Form Widget and place them in the graphical layout as required. Use the layouts for designing the activity.



3. Open activity.xml file and type the necessary properties and save the file.

4. In MainActivity.java file, add the code required for listening and handling events.

Event Handler Registration:

Button1.setOnClickListener(View.OnClickListener() {

public void onClick(View V)

{

// code todo

}  
});

5. Inside the event handlers, code the statements to change the text size and text color when the corresponding buttons are clicked.

* Methods used to change the TextColor:

Text1.setTextColor(Color.parseColor(“color code”));

* Methods used to change the TextSize:

Text1.setTextSize(float));

6. Save the MainActivity.java file. Right Click the Application name in the package explorer window and click Run as Android Application. After successful compliation.the projectname.apk file will be launched and executed on to the emulator.

**SOURCE CODE:**

**activity\_main.xml**

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

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

android:id=*"@+id/LinearLayout1"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:orientation=*"vertical"*

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"* >

<LinearLayout

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"* >

<TextView

android:id=*"@+id/textView1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"Enter Some Text"*

android:textAppearance=*"?android:attr/textAppearanceLarge"* />

<EditText

android:id=*"@+id/editText1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_weight=*"1"*

android:ems=*"10"* >

<requestFocus />

</EditText>

</LinearLayout>

<TableLayout

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"* >

<TableRow

android:id=*"@+id/tableRow1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"* >

<TextView

android:id=*"@+id/textView2"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"Change Text Size"*

android:layout\_span=*"3"*

android:textAppearance=*"?android:attr/textAppearanceMedium"* />

</TableRow>

<TableRow

android:id=*"@+id/tableRow2"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"* >

<Button

android:id=*"@+id/button1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"25sp"* />

<Button

android:id=*"@+id/button2"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"35sp"* />

<Button

android:id=*"@+id/button3"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"45sp"* />

</TableRow>

</TableLayout>

<GridLayout

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"* >

<Button

android:id=*"@+id/button4"*

style=*"?android:attr/buttonStyleSmall"*

android:layout\_gravity=*"left"*

android:background=*"#ff0000"* />

<Button

android:id=*"@+id/button5"*

style=*"?android:attr/buttonStyleSmall"*

android:layout\_gravity=*"left"*

android:background=*"#00ff00"* />

<Button

android:id=*"@+id/button6"*

style=*"?android:attr/buttonStyleSmall"*

android:layout\_gravity=*"left"*

android:background=*"#0000ff"* />

</GridLayout>

</LinearLayout>

**MainActivity.java:**

package com.example.exercise2;

import android.os.Bundle;

import android.app.Activity;

import android.view.Menu;

import android.graphics.Color;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

public class MainActivity extends Activity {

EditText text1;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

text1=(EditText)findViewById(R.id.*editText1*);

Button size1=(Button)findViewById(R.id.*button1*);

Button col1=(Button)findViewById(R.id.*button4*);

Button size2=(Button)findViewById(R.id.*button2*);

Button col2=(Button)findViewById(R.id.*button5*);

Button size3=(Button)findViewById(R.id.*button3*);

Button col3=(Button)findViewById(R.id.*button6*);

size1.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

text1.setTextSize(25); }

});

col1.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

text1.setTextColor(Color.*parseColor*("#ff0000"));

}

});

size2.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

text1.setTextSize(35);

}

});

col2.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

text1.setTextColor(Color.*parseColor*("#00ff00"));}

});

size3.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

text1.setTextSize(45);}

});

col3.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

text1.setTextColor(Color.*parseColor*("#0000ff"));

}

}); }

@Override

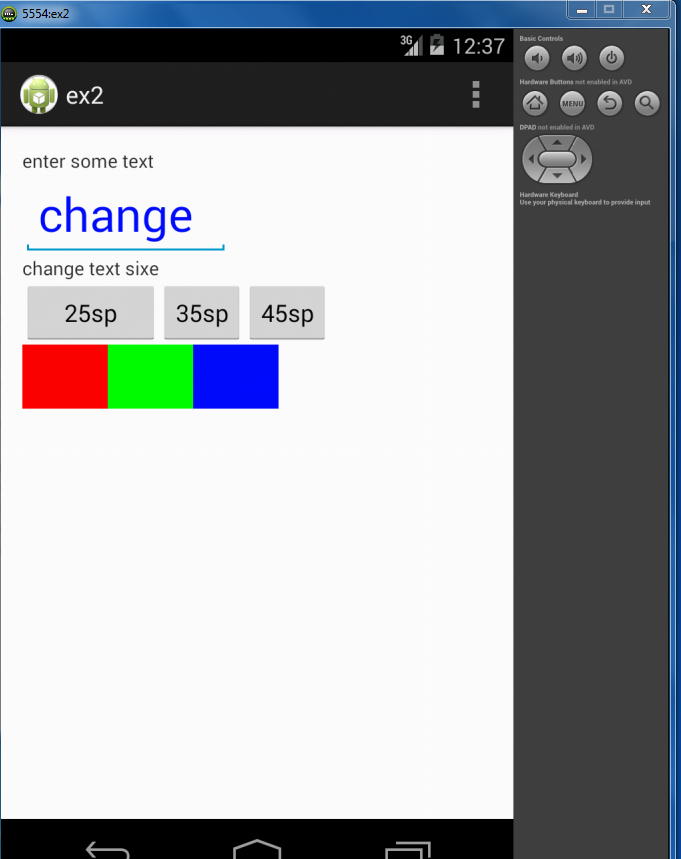
public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.*main*, menu);

return true; }

**OUTPUT:**



**RESULT:**

Thus the mobile application to demonstrate the use of Layout Managers and Event Listeners has been developed, launched and executed successfully.

**EX.NO:3 APPLICATION USING GRAPHICAL PRIMITIVES**

**DATE:**

**AIM:**

To develop a mobile application to draw the basic graphical primitives on the screen.

**STEPS:**

1. Create a new android application by choosing the Application name, minimum required SDK to be API19: Android4.4 (Kitkat) and create blank activity.

2. Let the graphical layout be the same as shown below



3. No change is required the in the activity\_main .xml file.

4. Create an emulator:

4.1: Choose “window” from the menu bar.

4.2: Select “Android Virtual Device Manager”.

4.3: In the dialog box, type AVD Name, choose the required device from the list, give the memory, heap and SD card size. Then click “OK”. Here we use Nexus 4 as the device type. Let heap=16, RAM=512 and SD card size=200

4.4: Start and launch the emulator.

5. In the **protected** **void** onCreate(Bundle savedInstanceState), set the layout as

setContentView(**new** myview(**this**));

6. Code the Mainactivity.java to draw the basic primitives like line,circle,rectangle,square

7. Right Click the Application name in the package explorer window and click Run as Android Application. After successful compilation, the projectname.apk file will be launched and executed on the emulator.

**SOURCE CODE:**

**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"* >

</RelativeLayout>

**MainActivity.java:**

package com.example.exercise4;

import android.app.Activity;

import android.content.Context;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Paint;

import android.os.Bundle;

import android.view.Menu;

import android.view.View;

public class MainActivity extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(new myview(this)); }

private class myview extends View

{

public myview(Context context)

{

super(context);

}

@Override

protected void onDraw(Canvas canvas)

{

super.onDraw(canvas);

Paint paint=new Paint();

paint.setTextSize(40);

paint.setColor(Color.GREEN);

canvas.drawText("Circle", 55, 30, paint);

paint.setColor(Color.RED);

canvas.drawCircle(100, 150,100, paint);

paint.setColor(Color.GREEN);

canvas.drawText("Rectangle", 255, 30, paint);

paint.setColor(Color.YELLOW);

canvas.drawRect(250, 50,400,350, paint);

paint.setColor(Color.GREEN);

canvas.drawText("SQUARE", 55, 430, paint);

paint.setColor(Color.BLUE);

canvas.drawRect(50, 450,150,550, paint);

paint.setColor(Color.GREEN);

canvas.drawText("LINE", 255, 430, paint);

paint.setColor(Color.CYAN);

canvas.drawLine(250, 500, 350, 500, paint);

}}

@Override

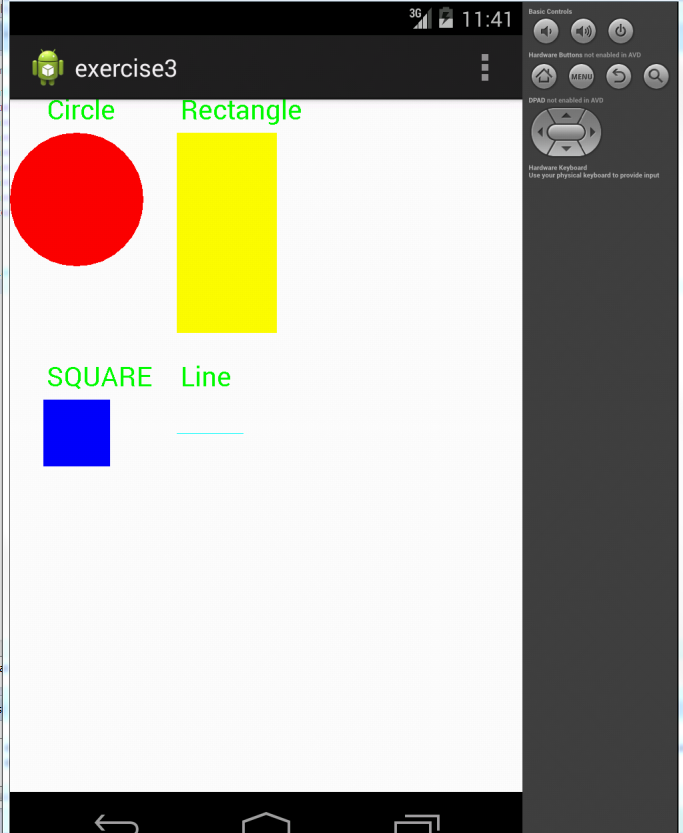
public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.main, menu);

return true; }}

**OUTPUT:**



**RESULT:**

Thus the mobile application to draw the basic graphical primitives on the screen

has been developed, launched and executed successfully.

**EX.NO:4 APPLICATION USING DATABASE**

**DATE:**

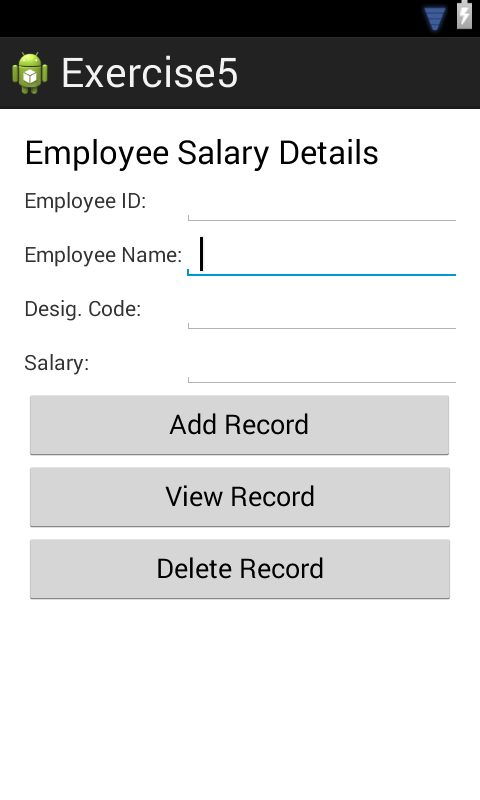
**AIM:**

To develop a mobile application to demonstrate the use of database.

**STEPS:**

1. Create a new android application by choosing the Application name, minimum required SDK to be API19: Android4.4 (Kitkat) and create blank activity.

2. Design the activity with the GUI components as shown in the graphical layout below. Pick the GUI components from the Form Widget and place them in the graphical layout as required.



3. Create the object for database and create the database

SQLiteDatabase db;

db=openOrCreateDatabase("EmployeeDB", Context.*MODE\_PRIVATE*, **null**);

db.execSQL("CREATE TABLE IF NOT EXISTS employee(empid VARCHAR,name VARCHAR,desig VARCHAR,salary VARCHAR);");

4.Code the MainActivity.java for employee details.

5. Create an emulator:

5.1: Choose “window” from the menu bar.

5.2: Select “Android Virtual Device Manager”.

5.3: In the dialog box, type AVD Name, choose the required device from the list, give the memory, heap and SD card size. Then click “OK”.

5.4: Start and launch the emulator.

6. Right Click the Application name in the package explorer window and click Run as Android Application. After successful compilation, the projectname.apk file will be launched and executed on the emulator.

**SOURCE CODE:**

**activity\_main.xml**

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

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

android:id=*"@+id/LinearLayout1"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:orientation=*"vertical"*

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"* >

<TableLayout

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"* >

<TableRow

android:id=*"@+id/tableRow1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"* >

<TextView

android:id=*"@+id/textView1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"Employee Salary Details"*

android:textAppearance=*"?android:attr/textAppearanceLarge"*

android:layout\_span=*"2"* />

</TableRow>

<TableRow

android:id=*"@+id/tableRow2"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"* >

<TextView

android:id=*"@+id/textView2"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"Employee ID:"* />

<EditText

android:id=*"@+id/editText1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:ems=*"10"* />

</TableRow>

<TableRow

android:id=*"@+id/tableRow3"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"* >

<TextView

android:id=*"@+id/textView3"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"Employee Name:"* />

<EditText

android:id=*"@+id/editText2"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:ems=*"10"* >

<requestFocus />

</EditText>

</TableRow>

<TableRow

android:id=*"@+id/tableRow4"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"* >

<TextView

android:id=*"@+id/textView4"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"Desig. Code:"* />

<EditText

android:id=*"@+id/editText3"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:ems=*"10"* />

</TableRow>

<TableRow

android:id=*"@+id/TableRow5"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"* >

<TextView

android:id=*"@+id/TextView5"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"Salary:"* />

<EditText

android:id=*"@+id/EditText4"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:ems=*"10"* />

</TableRow>

</TableLayout>

<Button

android:id=*"@+id/butAdd"*

android:layout\_width=*"287dp"*

android:layout\_height=*"wrap\_content"*

android:text=*"Add Record"* />

<Button

android:id=*"@+id/butView"*

android:layout\_width=*"288dp"*

android:layout\_height=*"wrap\_content"*

android:text=*"View Record"* />

<Button

android:id=*"@+id/butDel"*

android:layout\_width=*"288dp"*

android:layout\_height=*"wrap\_content"*

android:text=*"Delete Record"* />

</LinearLayout>

**MainActivity.java:**

package com.example.exercise5;

import android.os.Bundle;

import android.app.Activity;

import android.view.Menu;

import android.view.View.OnClickListener;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.\*;

import android.widget.\*;

import android.view.\*;

import android.app.AlertDialog.\*;

public class MainActivity extends Activity implements OnClickListener{

EditText id,ename,did,sal;

Button badd,bdel,bview;

SQLiteDatabase db;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

id=(EditText)findViewById(R.id.*editText1*);

ename=(EditText)findViewById(R.id.*editText2*);

did=(EditText)findViewById(R.id.*editText3*);

sal=(EditText)findViewById(R.id.*EditText4*);

badd=(Button)findViewById(R.id.*butAdd*);

bdel=(Button)findViewById(R.id.*butDel*);

bview=(Button)findViewById(R.id.*butView*);

badd.setOnClickListener(this);

bdel.setOnClickListener(this);

bview.setOnClickListener(this); db=openOrCreateDatabase("EmployeeDB", Context.*MODE\_PRIVATE*, null);

db.execSQL("CREATE TABLE IF NOT EXISTS employee(empid VARCHAR,name VARCHAR,desig VARCHAR,salary VARCHAR);");

}

public void onClick(View view)

{ if(view==badd){

if(id.getText().toString()=="" || ename.getText().toString()=="" || did.getText().toString()=="" || sal.getText().toString()=="" )

{ showMessage("Error", "Please enter all values");

return; }

db.execSQL("INSERT INTO employee VALUES('"+id.getText()+"','"+ename.getText()+"','"+did.getText()+"','"+sal.getText()+"');");

showMessage("Success", "Record added");

clearText();}

if(view==bdel){

if(id.getText().toString()=="")

{

showMessage("Error", "Please enter Employee id");

return; }

Cursor c=db.rawQuery("SELECT \* FROM employee WHERE empid='"+id.getText()+"'", null);

if(c.moveToFirst())

{

db.execSQL("DELETE FROM employee WHERE empid='"+id.getText()+"'");

showMessage("Success", "Record Deleted");}

else

{

showMessage("Error", "Invalid Employee id");

}

clearText();}

if(view==bview){

if(id.getText().toString()==""){

showMessage("Error", "Please enter Employee id");

return;}

Cursor c=db.rawQuery("SELECT \* FROM employee WHERE empid='"+id.getText()+"'", null);

if(c.moveToFirst()){

ename.setText(c.getString(1));

did.setText(c.getString(2));

sal.setText(c.getString(3));}

else

{

showMessage("Error", "Invalid Employee id");

clearText();

}}}

public void showMessage(String title,String message){

Builder builder=new Builder(this);

builder.setCancelable(true);

builder.setTitle(title);

builder.setMessage(message);

builder.show();}

public void clearText()

{

id.setText("");

ename.setText("");

did.setText("");

sal.setText("");

id.requestFocus();

}

@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);

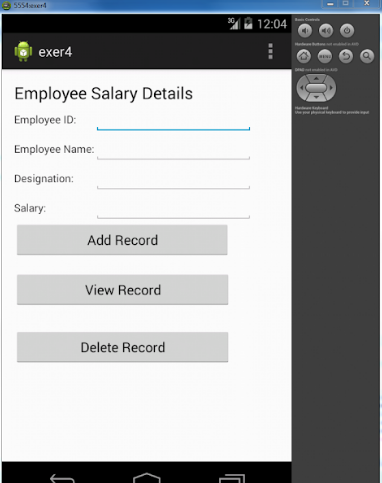
return true;

}

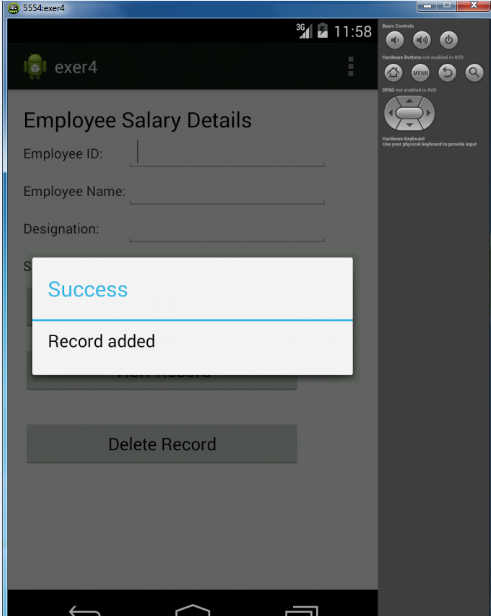
}

**OUTPUT:**

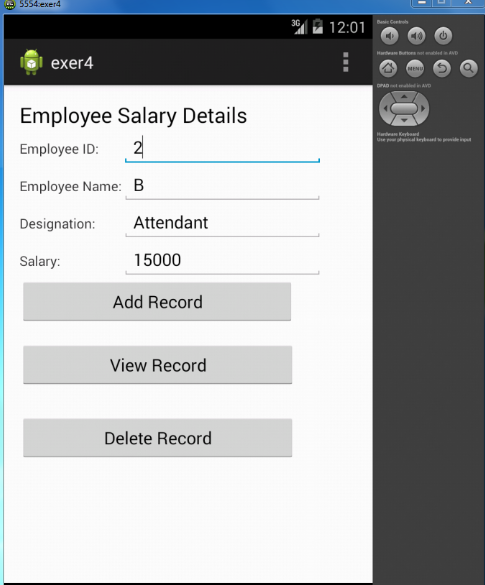
**Employee Details:**



**Adding a new record**



**Viewing a record**



**Deleting a record**



**RESULT:**

Thus the mobile application to demonstrate the use of database has been developed, launched and executed successfully.

**EX.NO:5 DEVELOP AN APPLICATION THAT MAKES USE OF**

**NOTIFICATION MANAGER**

**DATE:**

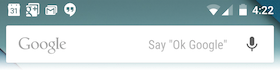
**AIM:**

To develop a mobile application to makes use of Notification Manager

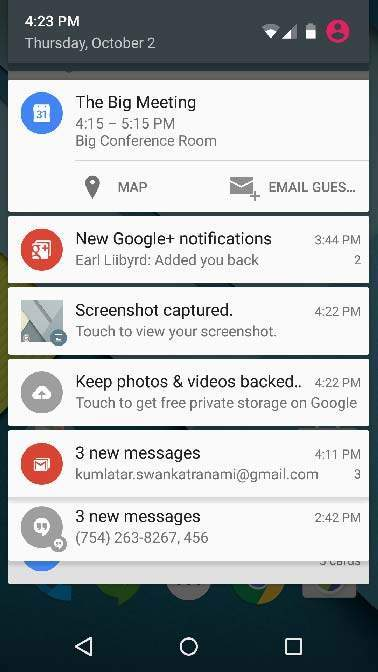
**STEPS:**

A **notification** is a message you can display to the user outside of your application's normal UI. When you tell the system to issue a notification, it first appears as an icon in the notification area. To see the details of the notification, the user opens the notification drawer. Both the notification area and the notification drawer are system-controlled areas that the user can view at any time.

Android **Toast** class provides a handy way to show users alerts but problem is that these alerts are not persistent which means alert flashes on the screen for a few seconds and then disappears.



To see the details of the notification, you will have to select the icon which will display notification drawer having detail about the notification. While working with emulator with virtual device, you will have to click and drag down the status bar to expand it which will give you detail as follows. This will be just **64 dp** tall and called normal view.



Above expanded form can have a **Big View** which will have additional detail about the notification. You can add upto six additional lines in the notification. The following screen shot shows such notification.

## CREATE AND SEND NOTIFICATIONS

You have simple way to create a notification. Follow the following steps in your application to create a notification −

### Step 1 - Create Notification Builder

As a first step is to create a notification builder using *NotificationCompat.Builder.build()*. You will use Notification Builder to set various Notification properties like its small and large icons, title, priority etc.

NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(this)

### Step 2 - Setting Notification Properties

Once you have **Builder** object, you can set its Notification properties using Builder object as per your requirement. But this is mandatory to set at least following −

* A small icon, set by **setSmallIcon()**
* A title, set by **setContentTitle()**
* Detail text, set by **setContentText()**

mBuilder.setSmallIcon(R.drawable.notification\_icon);

mBuilder.setContentTitle("Notification Alert, Click Me!");

mBuilder.setContentText("Hi, This is Android Notification Detail!");

You have plenty of optional properties which you can set for your notification. To learn more about them, see the reference documentation for NotificationCompat.Builder.

### Step 3 - Attach Actions

This is an optional part and required if you want to attach an action with the notification. An action allows users to go directly from the notification to an **Activity** in your application, where they can look at one or more events or do further work.

The action is defined by a **PendingIntent** containing an **Intent** that starts an Activity in your application. To associate the PendingIntent with a gesture, call the appropriate method of *NotificationCompat.Builder*. For example, if you want to start Activity when the user clicks the notification text in the notification drawer, you add the PendingIntent by calling **setContentIntent()**.

A PendingIntent object helps you to perform an action on your applications behalf, often at a later time, without caring of whether or not your application is running.

We take help of stack builder object which will contain an artificial back stack for the started Activity. This ensures that navigating backward from the Activity leads out of your application to the Home screen.

Main activity file **src/com.example.notificationdemo/MainActivity.java**. This file can include each of the fundamental lifecycle methods.

package com.example.notificationdemo;

import android.app.Activity;

import android.app.NotificationManager;

import android.app.PendingIntent;

import android.content.Context;

import android.content.Intent;

import android.support.v4.app.NotificationCompat;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

public class MainActivity extends Activity {

Button b1;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

b1 = (Button)findViewById(R.id.button);

b1.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

addNotification();

}

});

}

private void addNotification() {

NotificationCompat.Builder builder =

new NotificationCompat.Builder(this)

.setSmallIcon(R.drawable.abc)

.setContentTitle("Notifications Example")

.setContentText("This is a test notification");

Intent notificationIntent = new Intent(this, MainActivity.class);

PendingIntent contentIntent = PendingIntent.getActivity(this, 0, notificationIntent,

PendingIntent.FLAG\_UPDATE\_CURRENT);

builder.setContentIntent(contentIntent);

// Add as notification

NotificationManager manager = (NotificationManager) getSystemService(Context.NOTIFICATION\_SERVICE);

manager.notify(0, builder.build());

}

}

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

<?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="400dp"

android:text="Hi, Your Detailed notification view goes here...." />

</LinearLayout>

Main activity file **src/com.example.notificationdemo/NotificationView.java**.

package com.example.notificationdemo;

import android.os.Bundle;

import android.app.Activity;

public class NotificationView extends Activity{

@Override

public void onCreate(Bundle savedInstanceState){

super.onCreate(savedInstanceState);

setContentView(R.layout.notification);

}

}

Following will be the content of **res/layout/activity\_main.xml** file −

<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">

<TextView

android:id="@+id/textView1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Notification Example"

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"

android:layout\_marginTop="48dp" />

<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"

android:layout\_marginTop="42dp" />

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Notification"

android:id="@+id/button"

android:layout\_marginTop="62dp"

android:layout\_below="@+id/imageButton"

android:layout\_centerHorizontal="true" />

</RelativeLayout>

Following will be the content of **res/values/strings.xml** to define two new constants −

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

<resources>

<string name="action\_settings">Settings</string>

<string name="app\_name">tutorialspoint </string>

</resources>

Following is the default content of **AndroidManifest.xml** −

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

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

package="com.example.notificationdemo" >

<application

android:allowBackup="true"

android:icon="@drawable/ic\_launcher"

android:label="@string/app\_name"

android:theme="@style/AppTheme" >

<activity

android:name="com.example.notificationdemo.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=".NotificationView"

android:label="Details of notification"

android:parentActivityName=".MainActivity">

<meta-data

android:name="android.support.PARENT\_ACTIVITY"

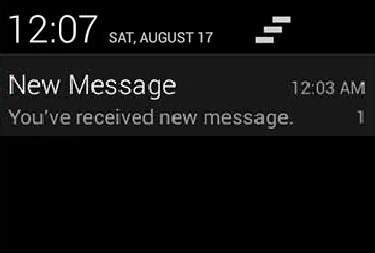
android:value=".MainActivity"/>

</activity>

</application>

</manifest>

OUTPUT

**RESULT:**

Thus the mobileapplication for sending sms has been demonstrated successfully

**EX.NO:6 APPLICATIONS USING MULTITHREADING**

**DATE:**

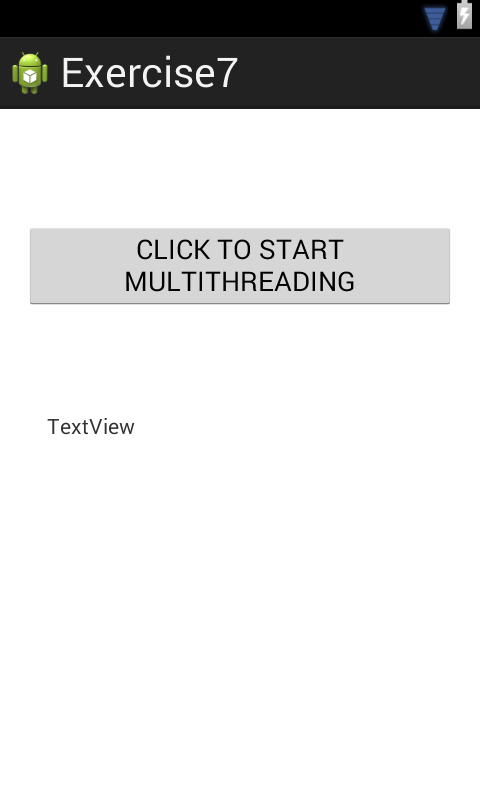
**AIM:**

To develop a mobile application to implement multithreading.

**STEPS:**

1. Create a new android application by choosing the Application name, minimum required SDK to be API19: Android4.4 (Kitkat) and create blank activity.

2. Let the graphical layout be the same as shown below



3. Save the activity\_main .xml file.

4. Create an emulator:

4.1: Choose “window” from the menu bar.

4.2: Select “Android Virtual Device Manager”.

4.3: In the dialog box, type AVD Name, choose the required device from the list, give the memory, heap and SD card size. Then click “OK”. Here we use 3.3 WQVGA (240x400:ldpi) as the device type. Let heap=16, RAM=512 and SD card size=200

4.4: Start and launch the emulator.

5. Declare the objects for thread creation as shown below:

**private** **static** **final** **int thread1,thread2,thread3** and **textview**

6. Code the Mainactivity.java to implement the multithreading concept for simultaneous execution(here we code for three threads).Define the thread as shown below: Thread thread1 = **new** Thread(**new** Runnable() {

**public** **void** run(){));

7. Right Click the Application name in the package explorer window and click Run as Android Application. After successful compilation, the projectname.apk file will be launched and executed on the emulator.

**SOURCE CODE:**

**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" >

<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="59dp"

android:text="CLICK TO START MULTITHREADING"

android:onClick="fetchData" />

<TextView

android:id="@+id/textView1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignLeft="@+id/button1"

android:layout\_below="@+id/button1"

android:layout\_marginLeft="15dp"

android:layout\_marginTop="69dp"

android:text="TextView" />

</RelativeLayout>

**MainActivity.java**

package com.example.exercise7;

import android.os.Bundle;

import android.annotation.SuppressLint;

import android.app.Activity;

import android.view.Menu;

import android.os.Handler;

import android.view.\*;

import android.widget.\*;

@SuppressLint("HandlerLeak")

public class MainActivity extends Activity {

private TextView tvOutput;

private static final int *t1* = 1;

private static final int *t2* = 2;

private static final int *t3* = 3;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

tvOutput=(TextView)findViewById(R.id.*textView1*);}

public void fetchData(View v) {

tvOutput.setText("Main thread");

thread1.start();

thread2.start();

thread3.start();}

Thread thread1 = new Thread(new Runnable() {

@Override

public void run() {

for (int i = 0; i < 3; i++) {

try {

Thread.*sleep*(2000);

} catch (InterruptedException e) {

e.printStackTrace();

}

handler.sendEmptyMessage(*t1*);}}

});

Thread thread2 = new Thread(new Runnable() {

@Override

public void run() {

for (int i = 0; i < 3; i++) {

try {

Thread.*sleep*(2000);

} catch (InterruptedException e) {

e.printStackTrace();}

handler.sendEmptyMessage(*t2*);}}});

Thread thread3 = new Thread(new Runnable() {

@Override

public void run() {

for (int i = 0; i < 3; i++) {

try {

Thread.*sleep*(2000);

} catch (InterruptedException e) {

e.printStackTrace();}

handler.sendEmptyMessage(*t3*);

}

}

});

Handler handler = new Handler() {

public void handleMessage(android.os.Message msg) {

if(msg.what == *t1*) {

tvOutput.append("\nThread1 is executing...");

}

if(msg.what == *t2*) {

tvOutput.append("\nThread2 is executing...");

}

if(msg.what == *t3*)

{

tvOutput.append("\nThread3 is executing..."); }}

};

@Override

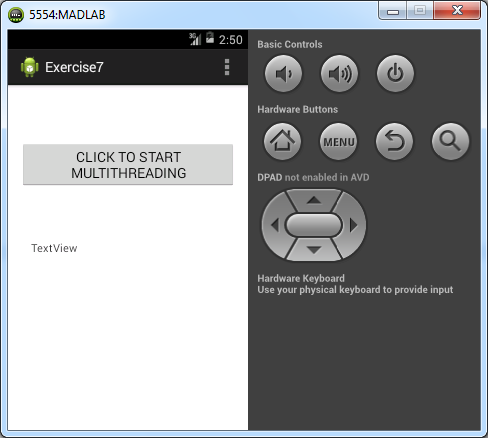
public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

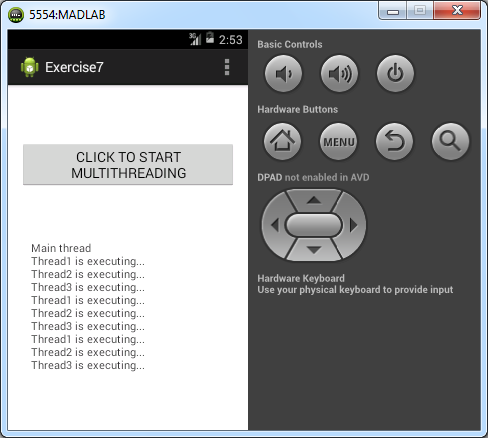
getMenuInflater().inflate(R.menu.*main*, menu);

return true; } }

**OUTPUT:**



**After clicking the button**



**RESULT:**

Thus the mobile application to implement multithreading has been developed, launched and executed successfully.

**EX.NO:7 APPLICATION TO DISPLAY CURRENT LOCATION USING GPS**

**DATE:**

**AIM:**

To develop an application to display the current location using GPS location information.

**PROCEDURE:**

1. Create a new android application by choosing the Application name, minimum required SDK to be API19: Android4.4 (Kitkat) and create blank activity.

2. Design the activity with the necessary GUI components.

**3. Steps to get location in Android:**

1. Provide permissions in manifest file for receiving location updates.
2. Create LocationManager instance as reference to the location service.
3. Request location from LocationManager.
4. Receive location update from LocationListener on change of location.

3.1: To access current location information through location providers, set permissions with android manifest file.

<manifest ... >

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />

<uses-permission android:name="android.permission. ACCESS\_COARSE\_LOCATION" />

<uses-permission android:name="android.permission.INTERNET" />

</manifest>

* ACCESS\_COARSE\_LOCATION is used when we use network location provider for our Android app.
* ACCESS\_FINE\_LOCATION is providing permission for both providers.
* INTERNET permission is must for the use of network provider.

## 3.2: Create LocationManager instance as reference to the location service:

locationManager =

(LocationManager)getSystemService(Context.LOCATION\_SERVICE);

## 3.3: Request current location from LocationManager

locationManager.requestLocationUpdates(LocationManager.GPS\_PROVIDER, 0, 0, this);

## 3.4: Receive location update from LocationListener on change of location

4. Send latitude and longitude to android emulator

* Open DDMS perspective in Eclipse (Window -> Open Perspective)
* Select your emulator device
* Select the tab named emulator control
* In ‘Location Controls’ panel, ‘Manual’ tab, give the Longitude and Latitude as input and ‘Send’.

1. Start and launch the emulator.
2. Right Click the Application name in the package explorer window and click Run as Android Application. After successful compilation, the projectname.apk file will be launched and executed on the emulator.

**SOURCE CODE:**

**activity\_main.xml**

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

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

android:id=*"@+id/LinearLayout1"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:orientation=*"vertical"*

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"* >

<Button

android:id=*"@+id/button1"*

android:layout\_width=*"278dp"*

android:layout\_height=*"wrap\_content"*

android:layout\_marginTop=*"18dp"*

android:text=*"My Location"*

/>

</LinearLayout>

**AndroidManifest.xml**

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

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

package=*"com.example.experiment8"*

android:versionCode=*"1"*

android:versionName=*"1.0"* >

<uses-sdk

android:minSdkVersion=*"19"*

android:targetSdkVersion=*"19"* />

<uses-permission android:name=*"android.permission.ACCESS\_FINE\_LOCATION"* />

<uses-permission android:name=*"android.permission.ACCESS\_MOCK\_LOCATION"* />

<uses-permission android:name=*"android.permission.ACCESS\_COARSE\_LOCATION"* />

<application

android:allowBackup=*"true"*

android:icon=*"@drawable/ic\_launcher"*

android:label=*"@string/app\_name"*

android:theme=*"@style/AppTheme"* >

<activity

android:name=*"com.example.experiment8.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>

**MainActivity.java**

**package** com.example.experiment8;

**import** android.os.Bundle;

**import** android.app.Activity;

**import** android.content.Context;

**import** android.view.Menu;

**import** android.view.View;

**import** android.widget.\*;

**import** android.location.\*;

**public** **class** MainActivity **extends** Activity {

Button but;

**private** **static** **final** **long** *MINIMUM\_DISTANCE*=1;

**private** **static** **final** **long** *MINIMUM\_TIME*=1000;

**protected** LocationManager LocManager;

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

but=(Button)findViewById(R.id.*button1*);

but.setOnClickListener(**new** View.OnClickListener() {

@Override

**public** **void** onClick(View v) {

// **TODO** Auto-generated method stub

showCurrentLocation();

}

});

LocManager=(LocationManager)getSystemService(Context.*LOCATION\_SERVICE*);

LocManager.requestLocationUpdates(LocationManager.*GPS\_PROVIDER*,0,0, **new** MyLocationListener());

}

**protected** **void** showCurrentLocation()

{

Location loc =

LocManager.getLastKnownLocation(LocationManager.*GPS\_PROVIDER*);

**if**(loc!=**null**)

{

String message=String.*format*("Location \n Longitute: %1$s \n Latitude:%2$s", loc.getLongitude(), loc.getLatitude());

Toast.*makeText*(MainActivity.**this**, message, Toast.*LENGTH\_LONG*).show();

}

}

**private** **class** MyLocationListener **implements** LocationListener

{

**public** **void** onLocationChanged(Location loc)

{

String message=String.*format*("New Location \n Longitute: %1$s \n Latitude:%2$s", loc.getLongitude(), loc.getLatitude());

Toast.*makeText*(MainActivity.**this**, message, Toast.*LENGTH\_LONG*).show();

}

**public** **void** onStatusChanged(String s,**int** i,Bundle b)

{

Toast.*makeText*(MainActivity.**this**, "Provider status changed",Toast.*LENGTH\_LONG*).show();

}

**public** **void** onProviderDisabled(String s)

{

Toast.*makeText*(MainActivity.**this**, "GPS turned off",Toast.*LENGTH\_LONG*).show();

}

**public** **void** onProviderEnabled(String s)

{

Toast.*makeText*(MainActivity.**this**, "GPS turned ON",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.*main*, menu);

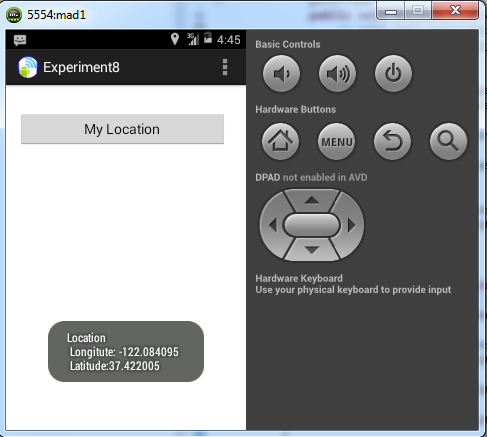
**return** **true**;

}

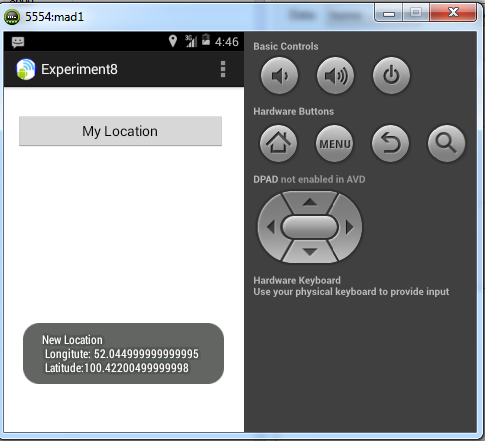
}

**OUTPUT:**

**Showing Last Known Location:**



**Showing New Location:**



**RESULT:**

Thus the mobile application to display the location information using GPS has been developed, launched and executed successfully.

**EX.NO:8 APPLICATION TO WRITE DATA TO THE SD CARD**

**DATE:**

**AIM:**

To develop a mobile application to write into and read data from SD Card.

**PROCEDURE:**

1. Create a new android application by choosing the Application name, minimum required SDK to be API19: Android4.4 (Kitkat) and create blank activity.

2. Design the activity with the necessary GUI components.

3. Set permission to write data in SD card. Go to AndroidManifest.xml file. And

write the following code. The code should come before <application> tab.

<uses-permission

android:name="android.permission.WRITE\_EXTERNAL\_STORAGE">

</uses-permission>

4. In MainActivity.java file,

4.1: Create or open a file in SD card.

File myFile = myFile = **new** File("/sdcard/mysdfile.txt");

4.2: Write the necessary code to write data entered by the user in EditText into

the specified file in the SD card.

4.3: Write the necessary code to read data from the specified file in the SD

card.

5. Start and launch the emulator.

6. Right Click the Application name in the package explorer window and click

Run as Android Application. After successful compilation, the

projectname.apk file will be launched and executed on the emulator.

**SOURCE CODE:**

**activity\_main.xml**

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

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

android:id=*"@+id/LinearLayout1"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:orientation=*"vertical"*

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"* >

<EditText

android:id=*"@+id/editText1"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"100sp"*

android:ems=*"10"*

android:background=*"#cccc99"*>

<requestFocus />

</EditText>

<Button

android:id=*"@+id/writeCard"*

android:layout\_width=*"281dp"*

android:layout\_height=*"wrap\_content"*

android:background=*"#ffff66"*

android:text=*"Write to SD card"* />

<Button

android:id=*"@+id/readCard"*

android:layout\_width=*"284dp"*

android:layout\_height=*"wrap\_content"*

android:background=*"#009900"*

android:text=*"Read SD Card"* />

<Button

android:id=*"@+id/clear"*

android:layout\_width=*"287dp"*

android:layout\_height=*"wrap\_content"*

android:background=*"#ff6600"*

android:text=*"Clear Screen"* />

<Button

android:id=*"@+id/exit"*

android:layout\_width=*"288dp"*

android:layout\_height=*"wrap\_content"*

android:background=*"#FF3300"*

android:text=*"Exit"* />

</LinearLayout>

**AndroidManifest.xml**

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

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

package=*"com.example.experiment9"*

android:versionCode=*"1"*

android:versionName=*"1.0"* >

<uses-sdk

android:minSdkVersion=*"19"*

android:targetSdkVersion=*"19"* />

<uses-permission android:name=*"android.permission.WRITE\_EXTERNAL\_STORAGE"* />

<application

android:allowBackup=*"true"*

android:icon=*"@drawable/ic\_launcher"*

android:label=*"@string/app\_name"*

android:theme=*"@style/AppTheme"* >

<activity

android:name=*"com.example.experiment9.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>

**MainActivity.java**

package com.example.experiment9;

import java.io.\*;

import android.os.Bundle;

import android.app.Activity;

import android.view.Menu;

import android.view.View;

import android.widget.\*;

public class MainActivity extends Activity {

EditText txtData;

Button btnWriteSDFile;

Button btnReadSDFile;

Button btnClearScreen;

Button btnClose;

File myFile;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

txtData=(EditText)findViewById(R.id.editText1);

myFile = new File("/sdcard/mysdfile.txt");

btnWriteSDFile = (Button) findViewById(R.id.writeCard);

btnWriteSDFile.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

// write on SD card file data in the text box

try {

//myFile.createNewFile();

FileOutputStream fOut = new FileOutputStream(myFile);

OutputStreamWriter myOutWriter = new OutputStreamWriter(fOut);

myOutWriter.append(txtData.getText());

myOutWriter.close();

fOut.close();

Toast.makeText(getBaseContext(),

"Done writing SD 'mysdfile.txt'",

Toast.LENGTH\_SHORT).show();

} catch (Exception e) {

Toast.makeText(getBaseContext(), e.getMessage(),

Toast.LENGTH\_SHORT).show();

}

}// onClick

}); // btnWriteSDFile

btnReadSDFile = (Button) findViewById(R.id.readCard);

btnReadSDFile.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

// write on SD card file data in the text box

try {

//File myFile = new File("/sdcard/mysdfile.txt");

FileInputStream fIn = new FileInputStream(myFile);

BufferedReader myReader = new BufferedReader(

new InputStreamReader(fIn));

String aDataRow = "";

String aBuffer = "";

while ((aDataRow = myReader.readLine()) != null)

{

aBuffer += aDataRow + "\n";

}

txtData.setText(aBuffer);

myReader.close();

Toast.makeText(getBaseContext(),

"Done reading SD 'mysdfile.txt'",

Toast.LENGTH\_SHORT).show();

} catch (Exception e) {

Toast.makeText(getBaseContext(), e.getMessage(),

Toast.LENGTH\_SHORT).show();

}

}// onClick

}); // btnReadSDFile

btnClearScreen = (Button) findViewById(R.id.clear);

btnClearScreen.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

// clear text box

txtData.setText("");

}

}); // btnClearScreen

btnClose = (Button) findViewById(R.id.exit);

btnClose.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

// clear text box

finish();

}

}); // btnClose

}

@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:**

**Writing data into SD card Reading data from SD card**



****

**RESULT:**

Thus the mobile application to write and read data into and from SD card has been developed, launched and executed successfully.

**EX.NO:9 APPLICATION TO CREATE AN ALERT UPON RECEIVING A MESSAGE**

**DATE:**

**AIM:**

To develop a mobile application to create an alert upon receiving a message

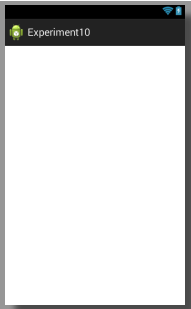
**PROCEDURE:**

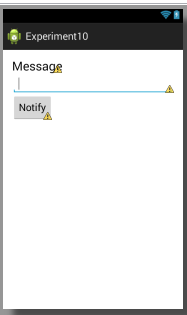
1. Create a new android application by choosing the Application name,

minimum required SDK to be API19: Android4.4 (Kitkat) and create blank

activity.

2. Design the activity with the necessary GUI components.

****

****

**Activity\_main.xml Activity\_result.xml**

3. Right click on the project and create a new Blank Activity to display an alert

message

4. In MainActivity.java file,

4.1: Create an instance for Intent class to forward the message to

ResultActivity

Intent intent=**new** Intent(**this**,ResultActivity.**class**);

4.2: Create an instance for Notification class to build a notification message

Box

Notification noti=**new** Notification.Builder(**this**)

.setContentTitle("New Message")

.setContentText(et.getText().toString())

.setSmallIcon(R.drawable.*ic\_launcher1*)

.setContentIntent(pending).build();

4.3: Create an instance for NotificationManager to obtain the Notification

service.

NotificationManager manager=

(NotificationManager)getSystemService(*NOTIFICATION\_SERVICE*);

5. Start and launch the emulator.

6. Right Click the Application name in the package explorer window and click

Run as Android Application. After successful compilation, the

projectname.apk file will be launched and executed on the emulator.

**SOURCE CODE:**

**activity\_main.xml**

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

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

android:id=*"@+id/LinearLayout1"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:orientation=*"vertical"*

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:text=*"Message"*

android:textAppearance=*"?android:attr/textAppearanceLarge"* />

<EditText

android:id=*"@+id/editText1"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:ems=*"10"* >

<requestFocus />

</EditText>

<Button

android:id=*"@+id/button1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"Notify"*

android:onClick=*"notify"*/>

</LinearLayout>

**MainActivity.java**

**package** com.example.experiment10;

**import** android.os.Bundle;

**import** android.app.\*;

**import** android.app.PendingIntent;

**import** android.view.\*;

**import** android.widget.\*;

**import** android.content.\*;

**public** **class** MainActivity **extends** Activity {

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

}

@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**;

}

**public** **void** notify(View view)

{

EditText et=(EditText)findViewById(R.id.*editText1*);

Intent intent=**new** Intent(**this**,ResultActivity.**class**);

PendingIntent pending=PendingIntent.*getActivity*(**this**, 0, intent, 0);

Notification noti=**new** Notification.Builder(**this**)

.setContentTitle("New Message")

.setContentText(et.getText().toString())

.setSmallIcon(R.drawable.*ic\_launcher1*)

.setContentIntent(pending).build();

NotificationManager manager =(NotificationManager)getSystemService(*NOTIFICATION\_SERVICE*);

noti.flags **|=** Notification.*FLAG\_AUTO\_CANCEL*;

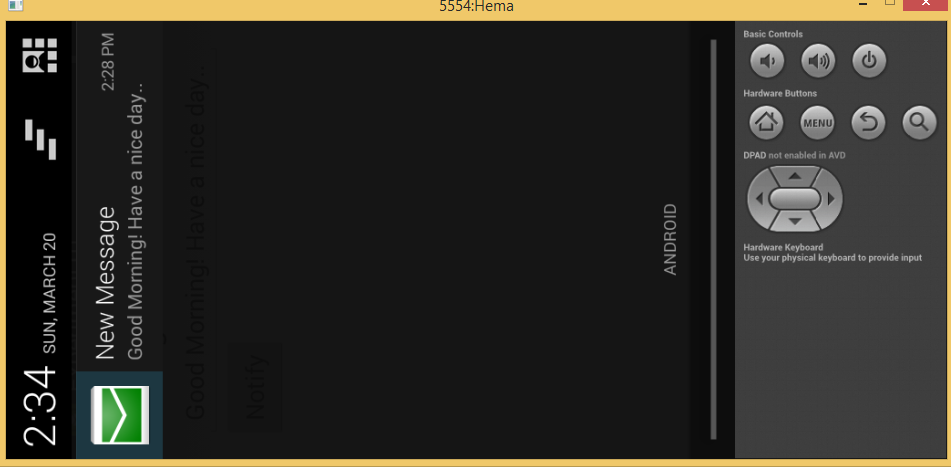
manager.notify(0,noti);

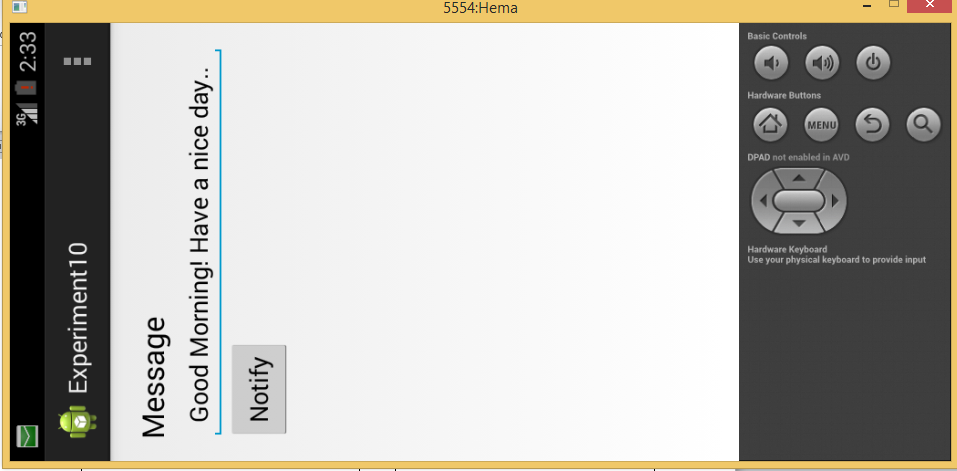
}

}

**OUTPUT:**

**Sending Message…. Notification Alert for the received message…**





**RESULT:**

Thus the mobile application to create an alert upon receiving a message has been developed, launched and executed successfully.

**EX.NO:10 APPLICATIONS TO PARSE SIMPLE RSS FEED**

**DATE:**

**AIM:**

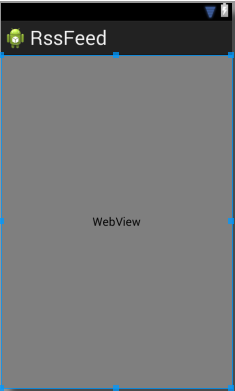
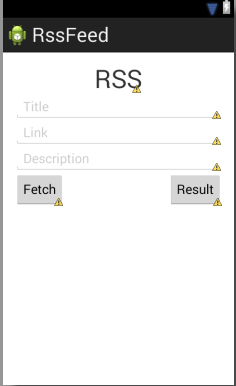
To develop an application that parses RSS feeds from XML RSS File in internet.

**PROCEDURE:**

1. Create a new android application by choosing the Application name, minimum

required SDK to be API19: Android4.4 (Kitkat) and create blank activity.

2. Design two activities with the necessary GUI components.



3. To access the internet, set permissions with android manifest file.

<uses-permission android:name="android.permission.INTERNET" />

4. Create XmlPullParserFactory object and then call its newPullParser() method to

create XMLPullParser. Its syntax is −

private XmlPullParserFactory xmlFactoryObject =

XmlPullParserFactory.newInstance();

private XmlPullParser myparser = xmlFactoryObject.newPullParser();

5. Specify the file for XmlPullParser that contains XML. It could be a file or could

be a Stream.

6. Parse the XML.

7. Start and launch the emulator.

1. Right Click the Application name in the package explorer window and click

Run as Android Application. After successful compilation, the projectname.apk

file will be launched and executed on the emulator.

**SOURCE CODE:**

**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"* >

<TextView

android:id=*"@+id/textview"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentTop=*"true"*

android:layout\_centerHorizontal=*"true"*

android:text=*"RSS"*

android:textSize=*"35dp"* />

<EditText

android:id=*"@+id/editText"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentEnd=*"true"*

android:layout\_alignParentLeft=*"true"*

android:layout\_alignParentRight=*"true"*

android:layout\_alignParentStart=*"true"*

android:layout\_below=*"@+id/textview"*

android:hint=*"Title"*

android:textColorHint=*"#cccccc"* />

<EditText

android:id=*"@+id/editText2"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignEnd=*"@+id/editText"*

android:layout\_alignLeft=*"@+id/editText"*

android:layout\_alignRight=*"@+id/editText"*

android:layout\_alignStart=*"@+id/editText"*

android:layout\_below=*"@+id/editText"*

android:hint=*"Link"*

android:textColorHint=*"#cccccc"* />

<EditText

android:id=*"@+id/editText3"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignEnd=*"@+id/editText2"*

android:layout\_alignLeft=*"@+id/editText2"*

android:layout\_alignRight=*"@+id/editText2"*

android:layout\_alignStart=*"@+id/editText2"*

android:layout\_below=*"@+id/editText2"*

android:hint=*"Description"*

android:textColorHint=*"#cccccc"* />

<Button

android:id=*"@+id/button"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentLeft=*"true"*

android:layout\_alignParentStart=*"true"*

android:layout\_below=*"@+id/editText3"*

android:layout\_centerHorizontal=*"true"*

android:text=*"Fetch"* />

<Button

android:id=*"@+id/button2"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignEnd=*"@+id/editText3"*

android:layout\_alignRight=*"@+id/editText3"*

android:layout\_alignTop=*"@+id/button"*

android:text=*"Result"* />

</RelativeLayout>

**activity\_second.xml**

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

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

android:orientation=*"vertical"* android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*>

<WebView

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:id=*"@+id/webView"*

android:layout\_gravity=*"center\_horizontal"* />

</LinearLayout>

**AndroidManifest.xml**

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

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

package=*"com.example.rssfeed"*

android:versionCode=*"1"*

android:versionName=*"1.0"* >

<uses-sdk

android:minSdkVersion=*"8"*

android:targetSdkVersion=*"17"* />

<application

android:allowBackup=*"true"*

android:icon=*"@drawable/ic\_launcher"*

android:label=*"@string/app\_name"*

android:theme=*"@style/AppTheme"* >

<activity

android:name=*"com.example.rssfeed.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=*"com.example.rssfeed.Second"*

android:label=*"@string/title\_activity\_second"* >

</activity>

</application>

<uses-permission android:name=*"android.permission.INTERNET"*/>

</manifest>

**MainActivity.java**

package com.example.rssfeed;

import android.app.Activity;

import android.content.Intent;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.view.MotionEvent;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

import java.util.Set;

public class MainActivity extends Activity

{

EditText title,link,description;

Button b1,b2;

private String finalUrl="http://tutorialspoint.com/android/sampleXML.xml";

private HandleXML obj;

@Override

protected void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

title = (EditText) findViewById(R.id.editText);

link = (EditText) findViewById(R.id.editText2);

description = (EditText) findViewById(R.id.editText3);

b1=(Button)findViewById(R.id.button);

b2=(Button)findViewById(R.id.button2);

b1.setOnClickListener(new View.OnClickListener()

{

@Override

public void onClick(View v)

{

obj = new HandleXML(finalUrl);

obj.fetchXML();

while(obj.parsingComplete);

title.setText(obj.getTitle());

link.setText(obj.getLink()); description.setText(obj.getDescription());

}

});

b2.setOnClickListener(new View.OnClickListener()

{

@Override

public void onClick(View v)

{

Intent in=new Intent(MainActivity.this,Second.class);

startActivity(in);

}

});

}

@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)

{

int id = item.getItemId();

if (id == R.id.action\_settings)

{

return true;

}

return super.onOptionsItemSelected(item);

}

}

**Second.java**

**package** com.example.rssfeed;

**import** android.app.Activity;

**import** android.os.Bundle;

**import** android.webkit.WebView;

**public** **class** Second **extends** Activity

{

@Override

**protected** **void** onCreate(Bundle savedInstanceState)

{

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_second*);

WebView w1=(WebView)findViewById(R.id.*webView*);

w1.loadUrl("http://tutorialspoint.com/android/sampleXML.xml");

}

}

**HandleXML.java**

package com.example.rssfeed;

import java.io.InputStream;

import java.net.HttpURLConnection;

import java.net.URL;

import org.xmlpull.v1.XmlPullParser;

import org.xmlpull.v1.XmlPullParserFactory;

import android.util.Log;

public class HandleXML

{

private String title = "title";

private String link = "link";

private String description = "description";

private String urlString = null;

private XmlPullParserFactory xmlFactoryObject;

public volatile boolean parsingComplete = true;

public HandleXML(String url)

{

this.urlString = url;

}

public String getTitle()

{

return title;

}

public String getLink()

{

return link;

}

public String getDescription()

{

return description;

}

public void parseXMLAndStoreIt(XmlPullParser myParser)

{

int event;

String text=null;

try

{

event = myParser.getEventType();

while (event != XmlPullParser.END\_DOCUMENT)

{

String name=myParser.getName();

switch (event)

{

case XmlPullParser.START\_TAG:

break;

case XmlPullParser.TEXT:

text = myParser.getText();

break;

case XmlPullParser.END\_TAG:

if(name.equals("title"))

{

title = text;

}

else if(name.equals("link"))

{

link = text;

}

else if(name.equals("description"))

{

description = text;

}

else

{

}

break;

}

event = myParser.next();

}

parsingComplete = false;

}

catch(Exception e)

{

e.printStackTrace();

}

}

public void fetchXML()

{

Thread thread = new Thread( new Runnable()

{

@Override

public void run()

{

try

{

URL url = new URL(urlString);

HttpURLConnection conn = (HttpURLConnection) url.openConnection();

conn.setReadTimeout(10000 /\* milliseconds \*/);

conn.setConnectTimeout(15000 /\* milliseconds \*/);

conn.setRequestMethod("GET");

conn.setDoInput(true); // Starts the query

conn.connect();

InputStream stream = conn.getInputStream();

xmlFactoryObject = XmlPullParserFactory.newInstance();

XmlPullParser myparser = xmlFactoryObject.newPullParser();

myparser.setFeature(XmlPullParser.FEATURE\_PROCESS\_NAMESPACES, false);

myparser.setInput(stream, null); parseXMLAndStoreIt(myparser);

stream.close();

}

catch(Exception e)

{

}

}

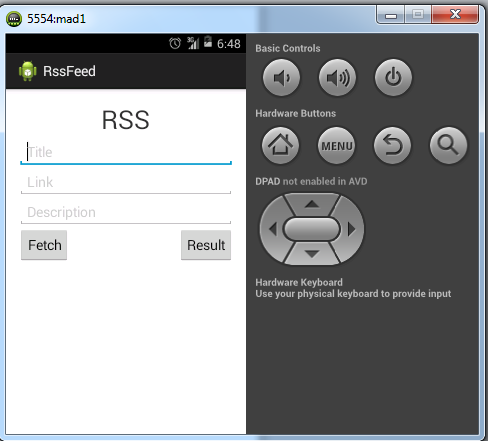
} );

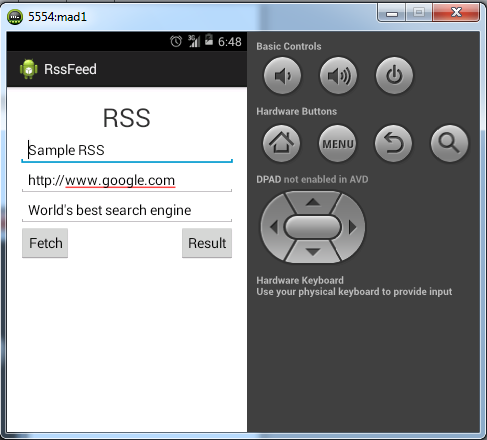
thread.start();

}

}

**OUTPUT:**







**RESULT:**

Thus the mobile application to parse RSS Feed has been developed, launched and executed successfully.

**EX.NO:11 DEVELOP A MOBILE APPLICATION TO SEND**

**DATE: AN EMAIL**

**AIM:**

To send an email from your application, you don’t have to implement an email client from the beginning, but you can use an existing one like the default Email app provided from Android, Gmail, Outlook, K-9 Mail etc. For this purpose, we need to write an Activity that launches an email client, using an implicit Intent with the right action and data. In this example, we are going to send an email from our app by using an Intent object that launches existing email clients.

## STEP :

## 1. Intent Object - Action to send Email

You will use ACTION\_SEND action to launch an email client installed on your Android device. Following is simple syntax to create an intent with ACTION\_SEND action.

Intent emailIntent = new Intent(Intent.ACTION\_SEND);

## 2.Intent Object - Data/Type to send Email

To send an email you need to specify mailto: as URI using setData() method and data type will be to text/plain using setType() method as follows −

emailIntent.setData(Uri.parse("mailto:"));

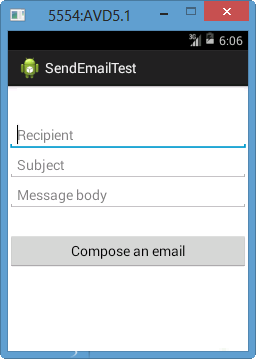
emailIntent.setType("text/plain");

## 3.Intent Object - Extra to send Email

Android has built-in support to add TO, SUBJECT, CC, TEXT etc. fields which can be attached to the intent before sending the intent to a target email client. You can use following extra fields in your email −

|  |  |
| --- | --- |
| Sr.No. | Extra Data & Description |
| 1 | EXTRA\_BCC  A String[] holding e-mail addresses that should be blind carbon copied. |
| 2 | EXTRA\_CC  A String[] holding e-mail addresses that should be carbon copied. |
| 3 | EXTRA\_EMAIL  A String[] holding e-mail addresses that should be delivered to. |
| 4 | EXTRA\_HTML\_TEXT  A constant String that is associated with the Intent, used with ACTION\_SEND to supply an alternative to EXTRA\_TEXT as HTML formatted text. |
| 5 | EXTRA\_SUBJECT  A constant string holding the desired subject line of a message. |
| 6 | EXTRA\_TEXT  A constant CharSequence that is associated with the Intent, used with ACTION\_SEND to supply the literal data to be sent. |
| 7 | EXTRA\_TITLE  A CharSequence dialog title to provide to the user when used with a ACTION\_CHOOSER. |

SAMPLE DESIGN



Main activity file **src/com.example.Tutorialspoint/MainActivity.java**.

package com.example.tutorialspoint;

import android.net.Uri;

import android.os.Bundle;

import android.app.Activity;

import android.content.Intent;

import android.util.Log;

import android.view.Menu;

import android.view.View;

import android.widget.Button;

import android.widget.Toast;

public class MainActivity extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

Button startBtn = (Button) findViewById(R.id.sendEmail);

startBtn.setOnClickListener(new View.OnClickListener() {

public void onClick(View view) {

sendEmail();

}

});

}

protected void sendEmail() {

Log.i("Send email", "");

String[] TO = {""};

String[] CC = {""};

Intent emailIntent = new Intent(Intent.ACTION\_SEND);

emailIntent.setData(Uri.parse("mailto:"));

emailIntent.setType("text/plain");

emailIntent.putExtra(Intent.EXTRA\_EMAIL, TO);

emailIntent.putExtra(Intent.EXTRA\_CC, CC);

emailIntent.putExtra(Intent.EXTRA\_SUBJECT, "Your subject");

emailIntent.putExtra(Intent.EXTRA\_TEXT, "Email message goes here");

try {

startActivity(Intent.createChooser(emailIntent, "Send mail..."));

finish();

Log.i("Finished sending email...", "");

} catch (android.content.ActivityNotFoundException ex) {

Toast.makeText(MainActivity.this, "There is no email client installed.", Toast.LENGTH\_SHORT).show();

}

}

}

Following will be the content of **res/layout/activity\_main.xml** file −

Here abc indicates about tutorialspoint logo

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

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:orientation="vertical" >

<TextView

android:id="@+id/textView1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Sending Mail Example"

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\_above="@+id/imageButton"

android:layout\_alignRight="@+id/imageButton"

android:layout\_alignEnd="@+id/imageButton" />

<ImageButton

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:id="@+id/imageButton"

android:src="@drawable/abc"

android:layout\_centerVertical="true"

android:layout\_centerHorizontal="true" />

<Button

android:id="@+id/sendEmail"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="@string/compose\_email"/>

</LinearLayout>

Following will be the content of **res/values/strings.xml** to define two new constants −

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

<resources>

<string name="app\_name">Tutorialspoint</string>

<string name="compose\_email">Compose Email</string>

</resources>

Following is the default content of **AndroidManifest.xml** −

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

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

package="com.example.Tutorialspoint" >

<application

android:allowBackup="true"

android:icon="@drawable/ic\_launcher"

android:label="@string/app\_name"

android:theme="@style/AppTheme" >

<activity

android:name="com.example.tutorialspoint.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>

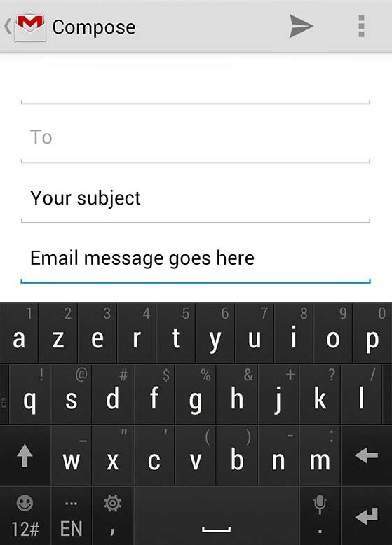
</application>

</manifest>

**OUTPUT :**



Now use **Compose Email** button to list down all the installed email clients. From the list, you can choose one of email clients to send your email. I'm going to use Gmail client to send my email which will have all the provided defaults fields available as shown below. Here **From:** will be default email ID you have registered for your Android device.



**RESULT:**

Thus the mobile application that creates an alert upon receiving a message has been executed successfully.

**EX.NO: 12 APPLICATION TO PLAY AUDIO FILES**

**DATE:**

**AIM:**

To develop a mobile application to play audio files.

**PROCEDURE:**

1. Create a new android application by choosing the Application name, minimum required SDK to be API19: Android4.4 (Kitkat) and create blank activity.

2. Design the activity with the Buttons and Label components.

3. Add some audio to the resource folder depending on the format of the audio that supports your emulator.

4. In MainActivity.java file,include the media player, audio player and media controller in widgets.

5. Set the setDataSource objects and include the media file by using getExternalStorageDirectory() method

6.fectch the path of the located file by usimg the getpath() method.

7.Start the API and launch the Emulator.

8. Right Click the Application name in the package explorer window and click

Run as Android Application. After successful compilation, the

projectname.apk file will be launched and executed on the emulator.

**SOURCE CODE:**

**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\_alignParentTop="true"

        android:layout\_marginTop="30dp"

        android:text="Audio Controller" />

    <Button

        android:id="@+id/button1"

        style="?android:attr/buttonStyleSmall"

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        android:layout\_alignLeft="@+id/textView1"

        android:layout\_below="@+id/textView1"

        android:layout\_marginTop="48dp"

        android:text="start" />

    <Button

        android:id="@+id/button2"

        style="?android:attr/buttonStyleSmall"

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        android:layout\_alignTop="@+id/button1"

        android:layout\_toRightOf="@+id/button1"

        android:text="pause" />

    <Button

        android:id="@+id/button3"

        style="?android:attr/buttonStyleSmall"

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        android:layout\_alignTop="@+id/button2"

        android:layout\_toRightOf="@+id/button2"

        android:text="stop" />

</RelativeLayout>

**MainActivity.java**

package com.example.audiomediaplayer1;

import android.media.MediaPlayer;

import android.net.Uri;

import android.os.Bundle;

import android.app.Activity;

import android.view.Menu;

import android.widget.MediaController;

import android.widget.VideoView;

  public class MainActivity extends Activity {

     @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

            MediaPlayer mp=new MediaPlayer();

        try{

            mp.setDataSource("/sdcard/Music/maine.mp3");//Write your location here

            mp.prepare();

            mp.start();

        }catch(Exception e){e.printStackTrace();}

             }

     @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;

    }

  }

**MainActivity2.java**

package com.example.audioplay;

import android.media.MediaPlayer;

import android.os.Bundle;

import android.os.Environment;

import android.app.Activity;

import android.view.Menu;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

public class MainActivity extends Activity {

    Button start,pause,stop;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

        start=(Button)findViewById(R.id.button1);

        pause=(Button)findViewById(R.id.button2);

        stop=(Button)findViewById(R.id.button3);

        //creating media player

        final MediaPlayer mp=new MediaPlayer();

        try{

                //you can change the path, here path is external directory(e.g. sdcard) /Music/maine.mp3

        mp.setDataSource(Environment.getExternalStorageDirectory().getPath()+"/Music/maine.mp3");

        mp.prepare();

        }catch(Exception e){e.printStackTrace();}

        start.setOnClickListener(new OnClickListener() {

            @Override

            public void onClick(View v) {

                mp.start();

            }

        });

        pause.setOnClickListener(new OnClickListener() {

            @Override

            public void onClick(View v) {

                mp.pause();

            }

        });

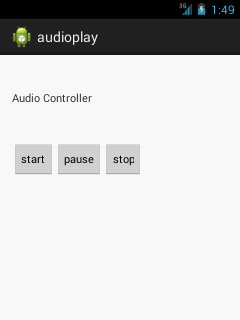
        stop.setOnClickListener(new OnClickListener() {

            @Override

            public void onClick(View v) {

                mp.stop();

**OUTPUT:**



**RESULT:**

Thus the mobile application to enable audio files that has been developed, launched and executed successfully.

**EX NO :13 STOPWATCH USING ANDROID APPLICATION**

**DATE :**

**AIM :**

To develop mobile application to create an android application for stopwatch .

**SOURCE CODE :**

**Activity main . xml**

**xml code**

<?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"

android:background="#C0E7EC"

android:padding="16dp"

tools:context="com.ProjectGurukul.stopwatch.MainActivity">

<TextView

android:id="@+id/time\_view"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:layout\_marginTop="100dp"

android:background="@drawable/border"

android:textAppearance="@android:style/TextAppearance.Large"

android:textSize="50dp" />

<Button

android:id="@+id/start\_button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:layout\_marginTop="50dp"

android:onClick="onClickStart"

android:text="@string/start" />

<Button

android:id="@+id/stop\_button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:layout\_marginTop="50dp"

android:onClick="onClickStop"

android:text="@string/stop" />

<Button

android:id="@+id/reset\_button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="50dp"

android:layout\_gravity="center\_horizontal"

android:onClick="onClickReset"

android:text="@string/reset" />

</LinearLayout>

**Main Activity .java**

package com.ProjectGurukul.stopwatch;

import android.os.Bundle;

import android.app.Activity;

import android.os.Handler;

import android.view.View;

import java.util.Locale;

import android.widget.TextView;

public class MainActivity extends Activity {

private int sec = 0;

private boolean is\_running;

private boolean was\_running;

@Override

protected void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

if (savedInstanceState != null) {

sec = savedInstanceState.getInt("seconds");

is\_running = savedInstanceState.getBoolean("running");

was\_running = savedInstanceState .getBoolean("wasRunning");

}

running\_Timer();

}

@Override

public void onSaveInstanceState(

Bundle savedInstanceState)

{

savedInstanceState.putInt("seconds", sec);

savedInstanceState.putBoolean("running", is\_running);

savedInstanceState.putBoolean("wasRunning", was\_running);

}

@Override

protected void onPause()

{

super.onPause();

was\_running = is\_running;

is\_running = false;

}

@Override

protected void onResume()

{

super.onResume();

if (was\_running) {

is\_running = true;

}

}

public void onClickStart(View view)

{

is\_running = true;

}

public void onClickStop(View view)

{

is\_running = false;

}

public void onClickReset(View view)

{

is\_running = false;

sec = 0;

}

private void running\_Timer()

{

final TextView t\_View = (TextView)findViewById(R.id.time\_view);

final Handler handle = new Handler();

handle.post(new Runnable() {

@Override

public void run()

{

int hrs = sec / 3600;

int mins = (sec % 3600) / 60;

int secs = sec % 60;

String time\_t = String .format(Locale.getDefault(), " %d:%02d:%02d ", hrs,mins, secs);

t\_View.setText(time\_t);

if (is\_running) {

sec++;

}

handle.postDelayed(this, 1000);

}

});

}

}

<resources>

<string name="app\_name">ProjectGurukul Stopwatch</string>

<string name="start"> Start </string>

<string name="reset"> Reset </string>

<string name="stop"> Stop </string>

</resources>

5. Now, we’ll create a folder in the res drawable file and name it as Border.xml. In this file, we will make a border for the Textview for the time, so it looks better. After creating the file write the following code in it:

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:shape="rectangle">

<stroke

android:width="2dp"

android:color="#932424"

/>

</shape>

**OUTPUT**



**RESULT**

Thus the mobile application application to implement stopwatch has been developed , launched and executed successfully .