CSE 2200: Software Development III

Our goal is to work in android development. Android is a complete set of software for mobile devices such as tablet computers, notebooks, smartphones, electronic book readers, set-top boxes etc. It contains a Linux-based Operating System, middleware and key mobile applications. It can be thought of as a mobile operating system. But it is not limited to mobile only. It is currently used in various devices such as mobiles, tablets, televisions etc.

> Operating system: An operating system is system software that manages computer hardware and software resources and provides common services for computer programs

Features of Android

- The important features of android are given below:
- It is open-source.
- Anyone can customize the Android Platform.
- There are a lot of mobile applications that can be chosen by the consumer.
- It provides many interesting features like weather details, opening screen, live RSS (Really Simple Syndication) feeds etc.
- It provides support for messaging services (SMS and MMS), web browser, storage (SQLite), connectivity (GSM, CDMA, Blue Tooth, Wi-Fi etc.), media, handset layout etc.

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.

Android Versions, Codename and API

There are many android versions, codenames and API Level.

Version	Code name	API Level
1.5	Cupcake	3
1.6	Donut	4
2.1	Eclair	7
2.2	Froyo	8
2.3	Gingerbread	9 and 10
3.1 and 3.3	Honeycomb	12 and 13
4.0	Ice Cream Sandwitch	15

4.1, 4.2 and 4.3	Jelly Bean	16, 17 and 18
4.4	KitKat	19
5.0	Lollipop	21
6.0	Marshmallow	23
7.0	Nougat	24-25
8.0	Oreo	26-27

An application programming interface (API) is a set of routines, protocols, and tools for building software applications.
The Android API refers to the collection of various software modules which make up the complete Android SDK. In simpler words the Android API or Android SDK or just plain simple Android basically refers to the same thing. Since the software you write yourself interacts with the Android software to do various things, so the Android part is like an API.

The Android SDK (software development kit) is a set of development tools used to develop applications for Android platform. The Android SDK includes the following:

- Required libraries
- Debugger
- An emulator
- Relevant documentation for the Android application program interfaces (APIs)
- Sample source code
- Tutorials for the Android OS

The development platforms that are compatible with SDK include operating systems like Windows (XP or later), Linux (any recent Linux distribution) and Mac OS X (10.4.9 or later). The components of Android SDK can be downloaded separately. Third party add-ons are also available for download.

Although the SDK can be used to write Android programs in the command prompt, the most common method is by using an integrated development environment (IDE). The recommended IDE is Eclipse with the Android Development Tools (ADT) plug-in. However, other IDEs, such as NetBeans or IntelliJ, will also work. Most of these IDEs provide a graphical interface enabling developers to perform development tasks faster. Since Android applications are written in Java code, a user should have the Java Development Kit (JDK) installed.

Android Studio is the official integrated development environment for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems.

APK is short for Android Package File. An APK is the file format which stores applications developed for the Android operating system. These files can be downloaded from the Google Play store, or manually installed using an Internet browser. APK files can be opened by an Android smartphone or tablet, a Chrome book with the Google Play store enabled, or a computer running an Android emulator.

Android apps are usually developed in Android Studio, using Java, the official coding language supported by Android. However, other languages are supported by the Google Play store, including Kotlin, C++, C#, and BASIC.

Emulator

The Android SDK includes a virtual mobile device emulator that runs on your computer. The emulator lets you prototype, develop and test Android applications without using a physical device.

Figure: An emulator from Android Studio

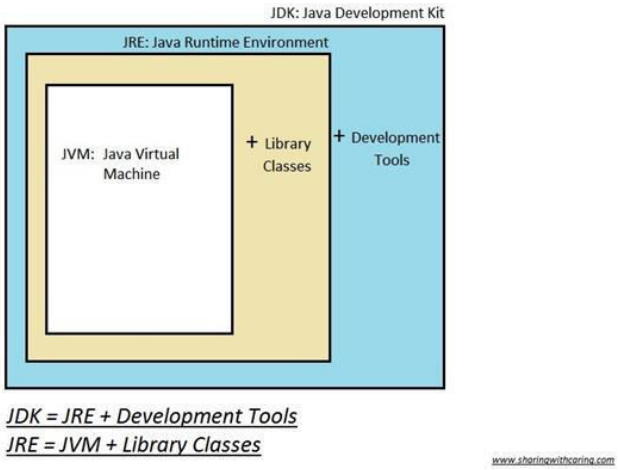
To emulate a real device, it is necessary to first create an AVD with the same device configurations as real device, then launch this AVD from AVD manager.

An Android Virtual Device (AVD) is a configuration that defines the characteristics of an Android phone, tablet, Wear OS, Android TV, or Automotive OS device that you want to simulate in the Android Emulator. The AVD Manager is an interface you can launch from Android Studio that helps you create and manage AVDs. To open the AVD Manager, do one of the following:

- Select **Tools > AVD Manager**.
- Click **AVD Manager** in the toolbar.

Java

Java is a general-purpose programming language that is class-based, object-oriented, and designed to have as few implementation dependencies as possible.



JDK: Java Development Kit (in short JDK) is Kit which provides the environment to Develop and execute (run) the Java program. For eg. You (as Java Developer) are developing an accounting application on your machine, so what do you going to need into your machine to develop and run this desktop app? You are going to need **J-D-K** for that purpose for this you just need to go to official website of sun or oracle to download the latest version of JDK into your machine.

Hence, JDK is a kit (or package) which includes two things

1. Development Tools(to provide an environment to develop your java programs)
2. JRE (to execute your java program). JDK is only used by Java Developers.

JRE: Java Runtime Environment (to say JRE) is an installation package which provides environment to only run (not develop) the java program (or application) onto your machine. For eg (continuing with the same example) after developing your accounting application, you want to run this application into your client's machine. Now in this case your client only need to run your application into his/her machine so your client should install JRE in-order to run your application into his machine.Hence, JRE is only used by them who only wants to run the Java Programs i.e. end users of your system.

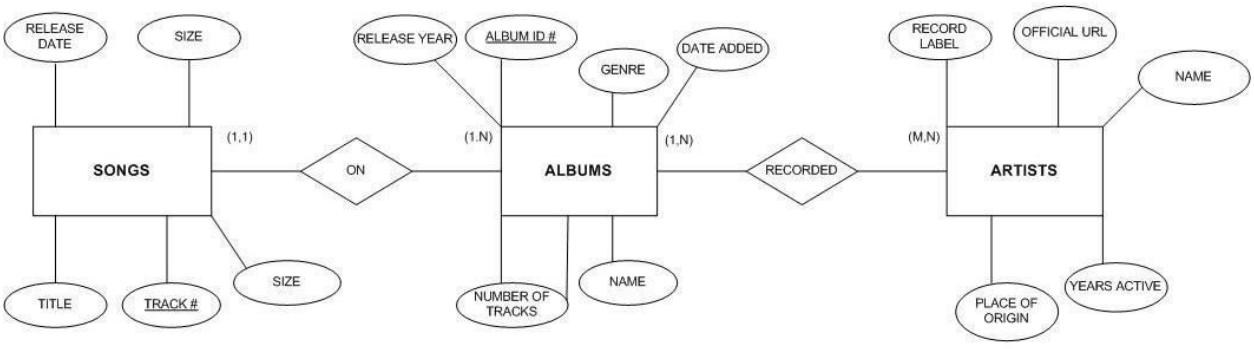
JVM: Java Virtual machine (JVM) is a very important part of both JDK and JRE because it is contained or inbuilt in both. Whatever java program you run using JRE or JDK goes into JVM and JVM is responsible to execute the java program line by line hence it is also known as interpreter(we will discuss about interpreter later) . Hence you don't need to install JVM separately into your machine because it is inbuilt into your JDK or JRE installation package. We'll explore more about JVM soon.

XML

XML stands for *Extensible Markup Language***.** It is used for 'drawing' the interfaces of an application. *XML is used for layout designing*. JAVA is used for writing the backend (developer's end) codes while frontend (user's end) codes are written on XML. A program code has no value without a good layout and design.

Database

A database is a data structure that stores organized information. Most databases contain multiple tables, which may each include several different fields. For example, a company database may include tables for products, employees, and financial records. Each of these tables would have different fields that are relevant to the information stored in the table. SQLite is a popular choice as embedded database software for local/client storage in application software.



Different Layouts

A layout defines the structure for a user interface in an application. In Android Studio there can be different layouts. Some of them are as below:

Linear Layout	Linear Layout is a view group that aligns all children in a single direction, vertically or horizontally.
Relative Layout	Relative Layout is a view group that displays child views in relative positions.
Table Layout	Table Layout is a view that groups views into rows and columns.
Absolute Layout	Absolute Layout enables you to specify the exact location of its children.
Frame Layout	The Frame Layout is a placeholder on screen that you can use to display a single view.

Relative Layout

Android Relative Layout enables to specify how child views are positioned relative to each other. The position of each view can be specified as relative to sibling elements or relative to the parent.

Sample 01

activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >

    <TextView
        android:id="@+id/ob1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Hello CSE"
        android:paddingBottom="20dp"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button"
        android:layout_below="@id/ob1"
        android:id="@+id/button1"
        android:paddingBottom="70dp"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button"
        android:layout_below="@id/button1"
        android:id="@+id/button2" />
```

```
    <TextView
        android:id="@+id/ob2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Hello CSE"
        android:layout_toRightOf="@id/button1"
        android:layout_below="@id/ob1"
        android:paddingBottom="20dp"
        android:background="#DE4"/>

</RelativeLayout>
```

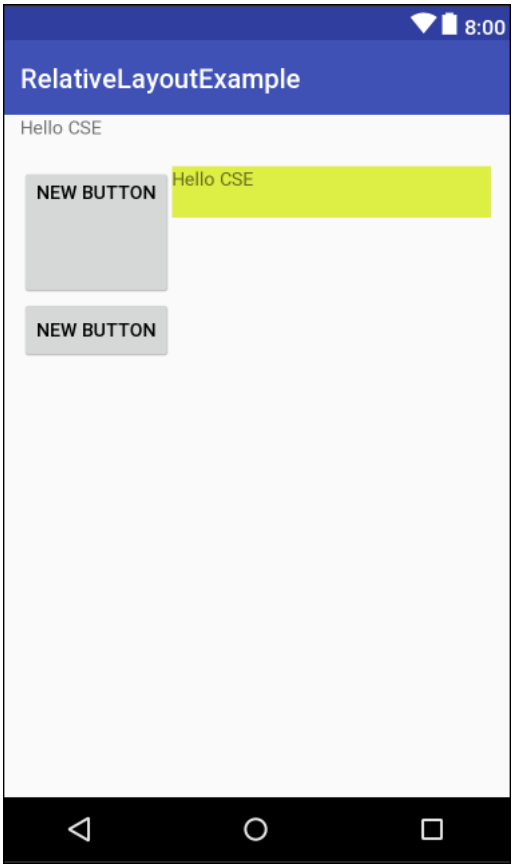


Figure: Layout of sample 01

Figure: Layout of sample 02

Sample 02

activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:background="#9f2">

    <TextView
```

```
        android:id="@+id/ob1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Hello CSE"
        android:paddingBottom="20dp"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button"
        android:layout_below="@id/ob1"
        android:id="@+id/button1"
        android:paddingBottom="70dp"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button"
        android:layout_below="@id/button1"
        android:id="@+id/button2" />

    <TextView
        android:id="@+id/ob2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Hello CSE"
        android:fontFamily="sans-serif-condensed"
        android:textSize="32dp"
        android:textColor="#FE3"
        android:layout_toRightOf="@id/button1"
        android:layout_below="@id/ob1"
        android:paddingBottom="20dp"
        android:background="#D04"/>

</RelativeLayout>
```

For both sample 01 and sample 02:

MainActivity.java

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

Linear Layout

Android Linear Layout is a view group that aligns all children in either *vertically* or *horizontally*.

Sample 03

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal" >

    <Button android:id="@+id/btn01"
        android:layout_width="192dp"
        android:layout_height="wrap_content"
        android:text="1st button"/>

    <Button android:id="@+id/btn02"
        android:layout_width="192dp"
        android:layout_height="wrap_content"
        android:text="2nd button"/>

</LinearLayout>
```

Sample 04

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <Button android:id="@+id/btn01"
        android:layout_width="192dp"
        android:layout_height="wrap_content"
        android:text="1st button"/>

    <Button android:id="@+id/btn02"
        android:layout_width="192dp"
        android:layout_height="wrap_content"
        android:text="2nd button"/>

</LinearLayout>
```

Codes of MainActivity.java for sample 03 and sample 04 are same as sample 01/sample 02

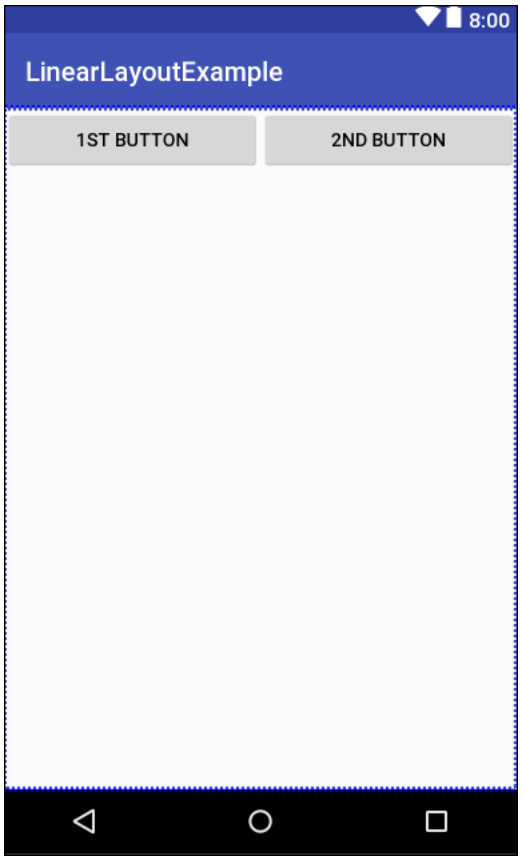


Figure: Layout of sample 03

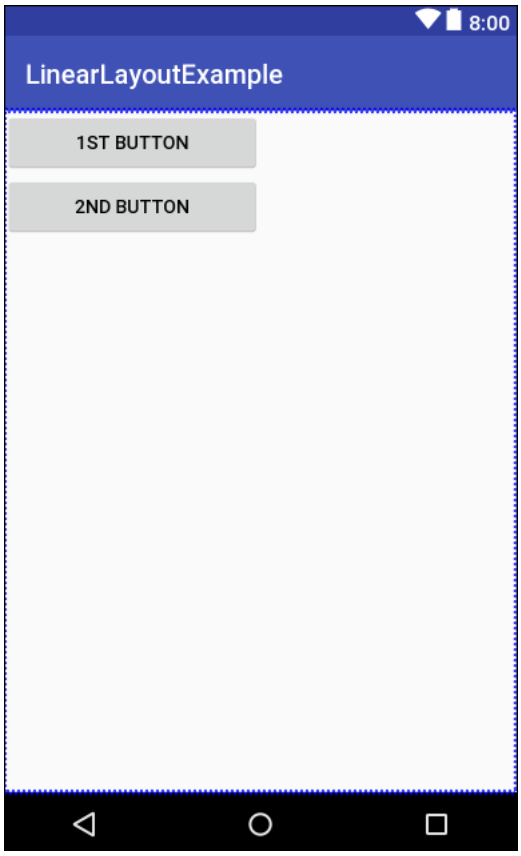


Figure: Layout of sample 04

Table Layout:

Android Table Layout arranges groups of views into rows and columns. <TableRow> element is used to build a row in the table. Each row has zero or more cells; each cell can hold one View object.

Sample 05

MainActivity.java

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

activity_main.xml

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <TableRow
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:paddingBottom="20dp">

        <TextView
            android:text="Time"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_column="1"
            android:textSize="30dp"/>

        <TextClock
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/textClock"
            android:textSize="30dp"
            android:layout_column="2" />

    </TableRow>

    <TableRow
        android:paddingBottom="20dp">

        <TextView
            android:text="First Name"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_column="1"
            android:textSize="30dp"/>

        <EditText
            android:width="200px"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="30dp"/>
    </TableRow>

    <TableRow
        android:paddingBottom="20dp">

        <TextView
            android:text="Last Name"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="30dp"
            android:layout_column="1" />

        <EditText
            android:width="100px"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="30dp"/>
    </TableRow>

    <TableRow
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:paddingBottom="20dp"
        >

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Submit"
            android:id="@+id/button"
            android:layout_column="2" />
    </TableRow>

</TableLayout>
```

Figure: Layout of sample 05

Frame Layout:

Sample 06

MainActivity.java is same as sample 05

activity_main.xml

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <ImageView
        android:id="@+id/backgroundImage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:scaleType="centerCrop"
        android:src="@drawable/android_pic" />

    <TextView
        android:id="@+id/descTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_marginTop="70dp"
        android:background="@android:color/holo_blue_light"
        android:padding="10dp"
        android:text="TextView placed at the top of the Imageview"
        android:textColor="@android:color/white"
        android:textSize="22sp" />

</FrameLayout>
```

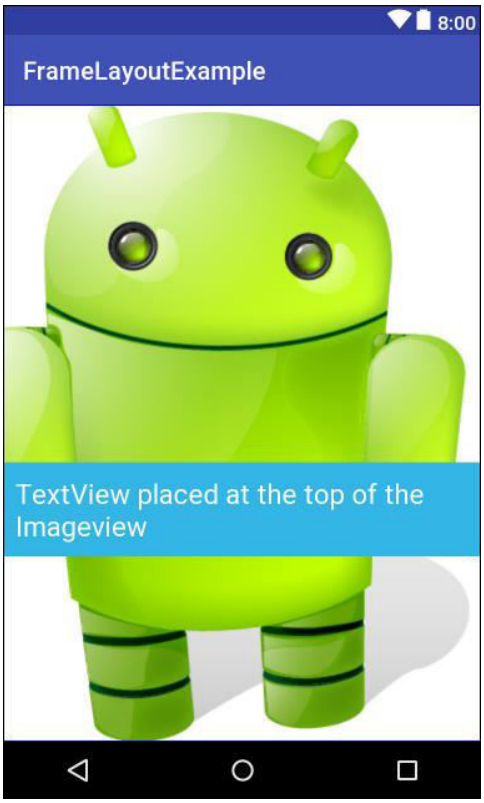


Figure: Layout of sample 06

Frame Layout is used to stack child views on top of each other, with the most recent child on top of the stack.

Absolute Layout

Sample 07

MainActivity.java is same as sample 05

activity_main.xml

```
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:text="OK"
        android:layout_x="50px"
        android:layout_y="361px" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:text="Cancel"
        android:layout_x="344px"
        android:layout_y="361px" />

</AbsoluteLayout>
```

An Absolute Layout lets specify exact locations (x/y coordinates) of its children. Absolute layouts are less flexible and harder to maintain than other types of layouts without absolute positioning.

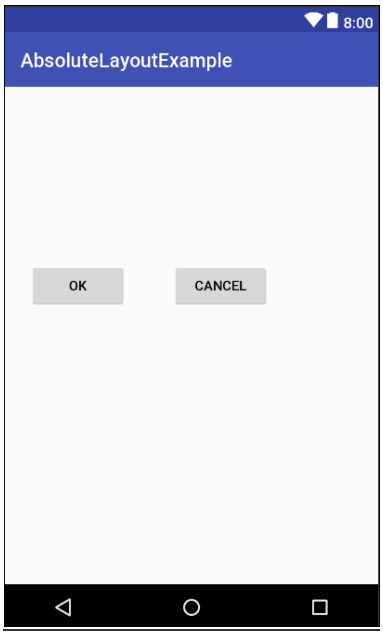
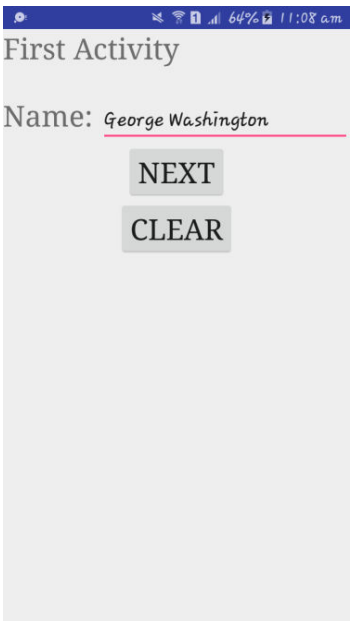


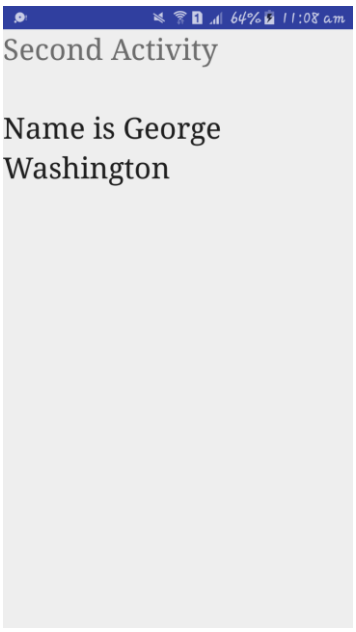
Figure: Layout of sample 07

Goal

1. Enter a name ‘xyz’ into a field
2. There will be two buttons
    a. If “Clear” button is clicked, the filed content will be cleared
    b. If “Next” button is clicked, in another activity “Your name is ‘xyz’” will be displayed

*If **NEXT** button is clicked*

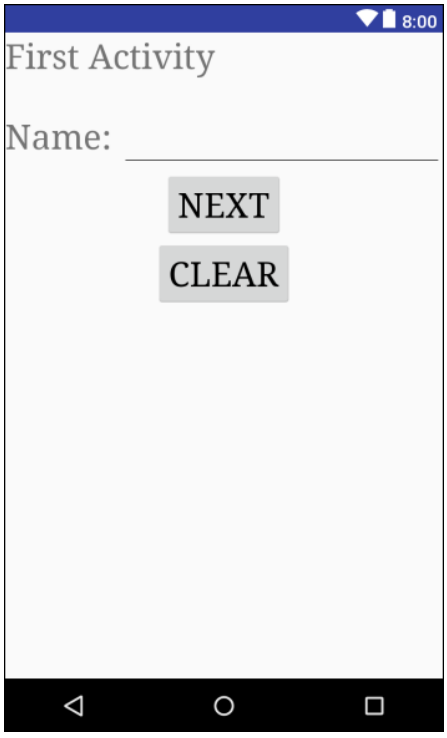




*If **CLEAR** button is clicked*

activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">

  <TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="First Activity"
    android:fontFamily="serif"
    android:textSize="30dp"/>

  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:orientation="horizontal"
    android:layout_marginTop="30dp">

    <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Name: "
      android:fontFamily="serif"
      android:textSize="30dp"/>

    <EditText
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:id="@+id/text01"/>

  </LinearLayout>



  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:text="Next"
    android:fontFamily="serif"
    android:textSize="30dp"
    android:id="@+id/btn01"/>

  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:text="Clear"
    android:fontFamily="serif"
    android:textSize="30dp"
    android:id="@+id/btn02"/>

</LinearLayout>
```

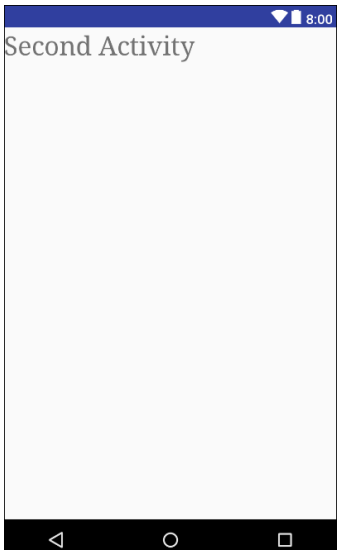


MainActivity.java

```
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {


   private Button btn01 , btn02;
   private EditText text01;


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

      btn01 = (Button) findViewById(R.id.btn01);
      btn01.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {

            EditText editText = (EditText)findViewById(R.id.text01);
            String text = editText.getText().toString();

            Intent myIntent = new Intent(MainActivity.this,SecondActivity.class);
            myIntent.putExtra("mytext",text);
            startActivity(myIntent);
         }
      });

      btn02= (Button) findViewById(R.id.btn02);
      btn02.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {

            text01= findViewById(R.id.text01);
            text01.setText("");

         }
      });
   }


}
```

activity_second.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical">

   <TextView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Second Activity"
      android:fontFamily="serif"
      android:textSize="30dp"/>

   <TextView
      android:id="@+id/textView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="42dp"
      android:fontFamily="serif"
      android:textSize="30dp"
      android:textAppearance="?android:attr/textAppearanceLarge" />

</LinearLayout>
```



SecondActivity.java

```
import android.app.Activity;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.EditText;
import android.widget.TextView;

public class SecondActivity extends Activity {

   TextView mTextview;

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

      String str02= getIntent().getStringExtra("mytext");
      String str03= "Name is "+ str02;

      mTextview = (TextView)findViewById(R.id.textView1);
      mTextview.setText(str03);
   }
}
```

Toast

Toast can be used to display information for the short period of time. A toast contains message to be displayed quickly and disappears after sometime.

An example of toast is as below:

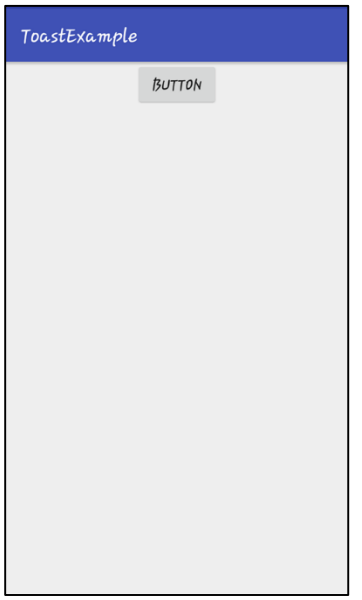
activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:id="@+id/btn01"
            android:text="Button"/>

</LinearLayout>
```

MainActivity.java

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

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

        Button btn01 = (Button) findViewById(R.id.btn01);
        btn01.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                Toast.makeText(getApplicationContext(),"Arigato Sayonara", Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

What If Toast.LENGTH_LONG ?



If the button is clicked

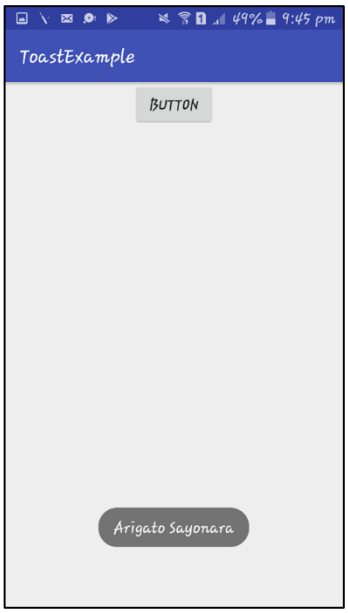


Image View

Image View class is used to display an image file in application. Image file is easy to use but hard to master in Android, because of the various screen sizes in Android devices. *Image View* comes with different configuration options to support different scale types. Scale type options are used for scaling the bounds of an image to the bounds of the *image view*.

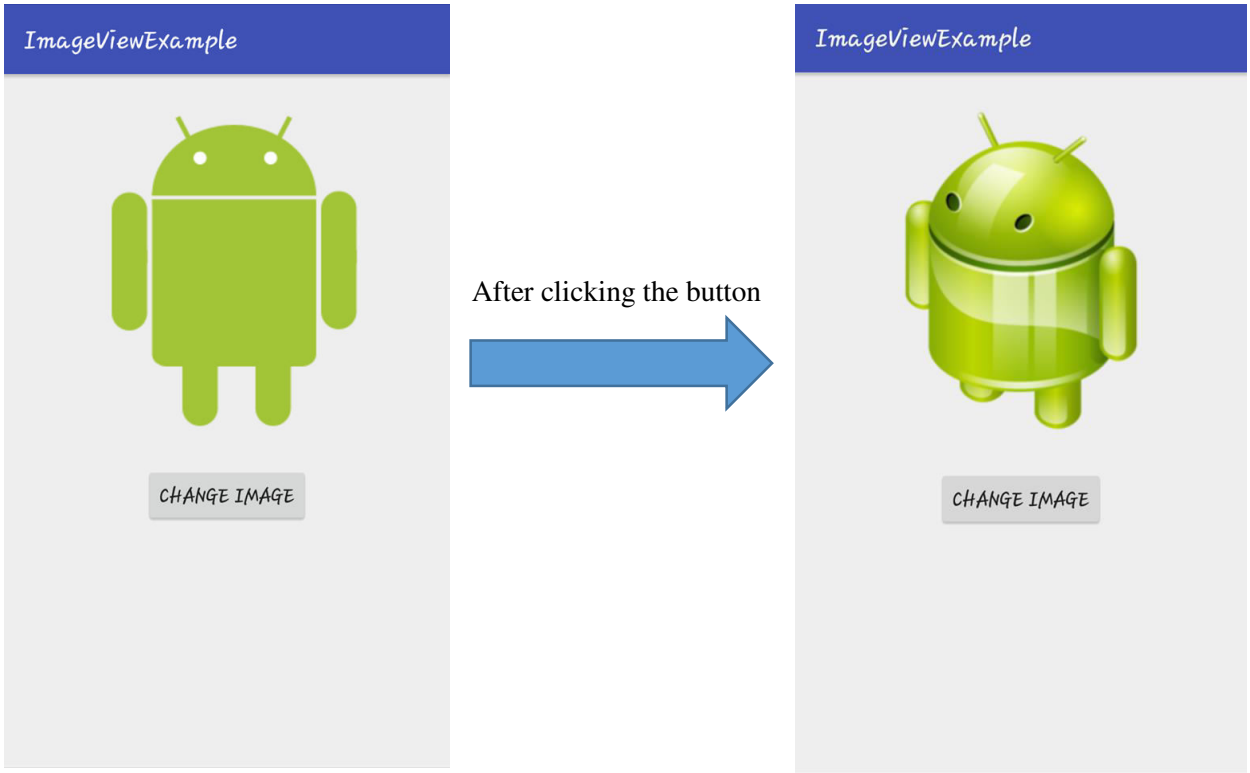
Different scale types:

CENTER	Center the image but doesn't scale the image
CENTER_CROP	Scale the image uniformly
CENTER_INSIDE	Center the image inside the container, rather than making the edge match exactly
FIT_CENTER	Scale the image from center
FIT_END	Scale the image from the end of the container
FIT_START	Scale the image from start of the container
FIT_XY	Fill the image from x and y coordinates of the container
MATRIX	Scale using the image matrix when drawing

An example of image view and button

Scenario: After button click, image of an activity shall change.

i. One activity
ii. Button click leads to change of image



activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:src="@drawable/android"
        android:padding="30dp"/>

    <Button
        android:id="@+id/btnChangeImage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="Change Image" />

</LinearLayout>
```

MainActivity.java

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Button;
import android.widget.ImageView;
import android.view.View;
import android.view.View.OnClickListener;

public class MainActivity extends AppCompatActivity {

    Button button;
    ImageView image;

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

        image = (ImageView) findViewById(R.id.imageView1);

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

        button.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View arg0) {
                image.setImageResource(R.drawable.android3d);
            }
        });
    }
}
```

android.jpg and *android3d.jpg* images have to be stored in the *projectname*\app\src\main\res\drawable folder

Clocks

In Android, Analog Clock is a two handed clock one for hour indicator and the other for minute indicator and Digital Clock & Text Clock both looks like your normal digital watch on hand which displays the hours minutes and seconds in digital format.

Text Clock is a widget same as Digital Clock but as in API level 17 digital clock is deprecated. To use Text Clock in app, minimum API level required is 17.

These components cannot be used to change the time.

Example

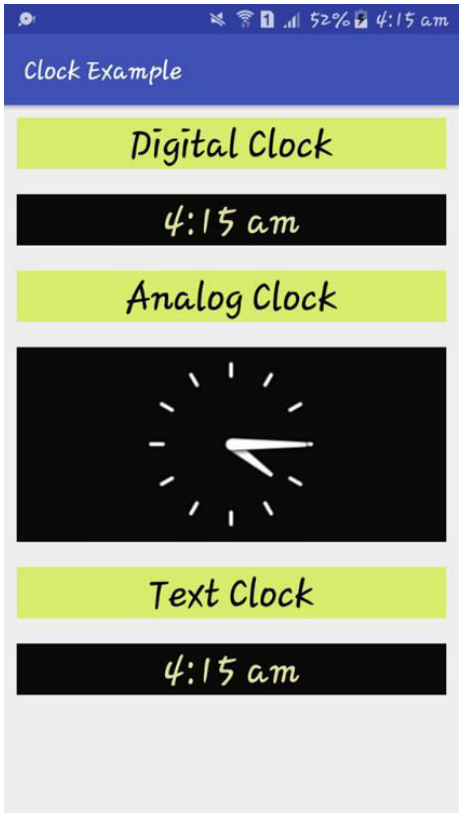
MainActivity.java

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">


    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Digital Clock"
        android:textSize="30dp"
        android:textColor="#090707"
        android:gravity="center_horizontal"
        android:background="#d9ec6d"
        android:layout_margin="10dp"/>

    <DigitalClock
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="30dp"
        android:textColor="#e2e1a4"
        android:gravity="center_horizontal"
        android:background="#0a0b09"
        android:layout_margin="10dp"/>
```

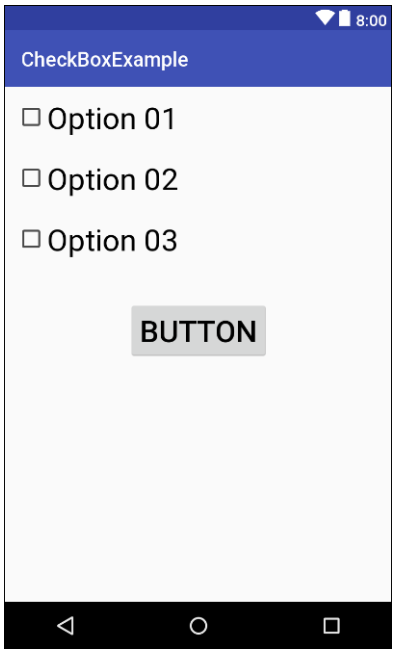
```
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Analog Clock"
        android:textSize="30dp"
        android:textColor="#090707"
        android:gravity="center_horizontal"
        android:background="#d9ec6d"
        android:layout_margin="10dp"/>

    <AnalogClock
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"
        android:background="#0a0b09"
        android:layout_margin="10dp"/>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Text Clock"
        android:textSize="30dp"
        android:textColor="#090707"
        android:gravity="center_horizontal"
        android:background="#d9ec6d"
        android:layout_margin="10dp"/>

    <TextClock
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="30dp"
        android:textColor="#e2e1a4"
        android:gravity="center_horizontal"
        android:background="#0a0b09"
        android:layout_margin="10dp"/>

</LinearLayout>
```



Check Box

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <CheckBox
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Option 01"
        android:id="@+id/chk01"
        android:textSize="30dp"
        android:layout_margin="10dp"/>

    <CheckBox
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Option 02"
        android:id="@+id/chk02"
        android:textSize="30dp"
        android:layout_margin="10dp"/>

    <CheckBox
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Option 03"
        android:id="@+id/chk03"
        android:textSize="30dp"
        android:layout_margin="10dp"/>

    <Button
        android:layout_gravity="center_horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/btn01"
        android:text="Button"
        android:textSize="30dp"
        android:layout_marginTop="30dp"/>

</LinearLayout>
```



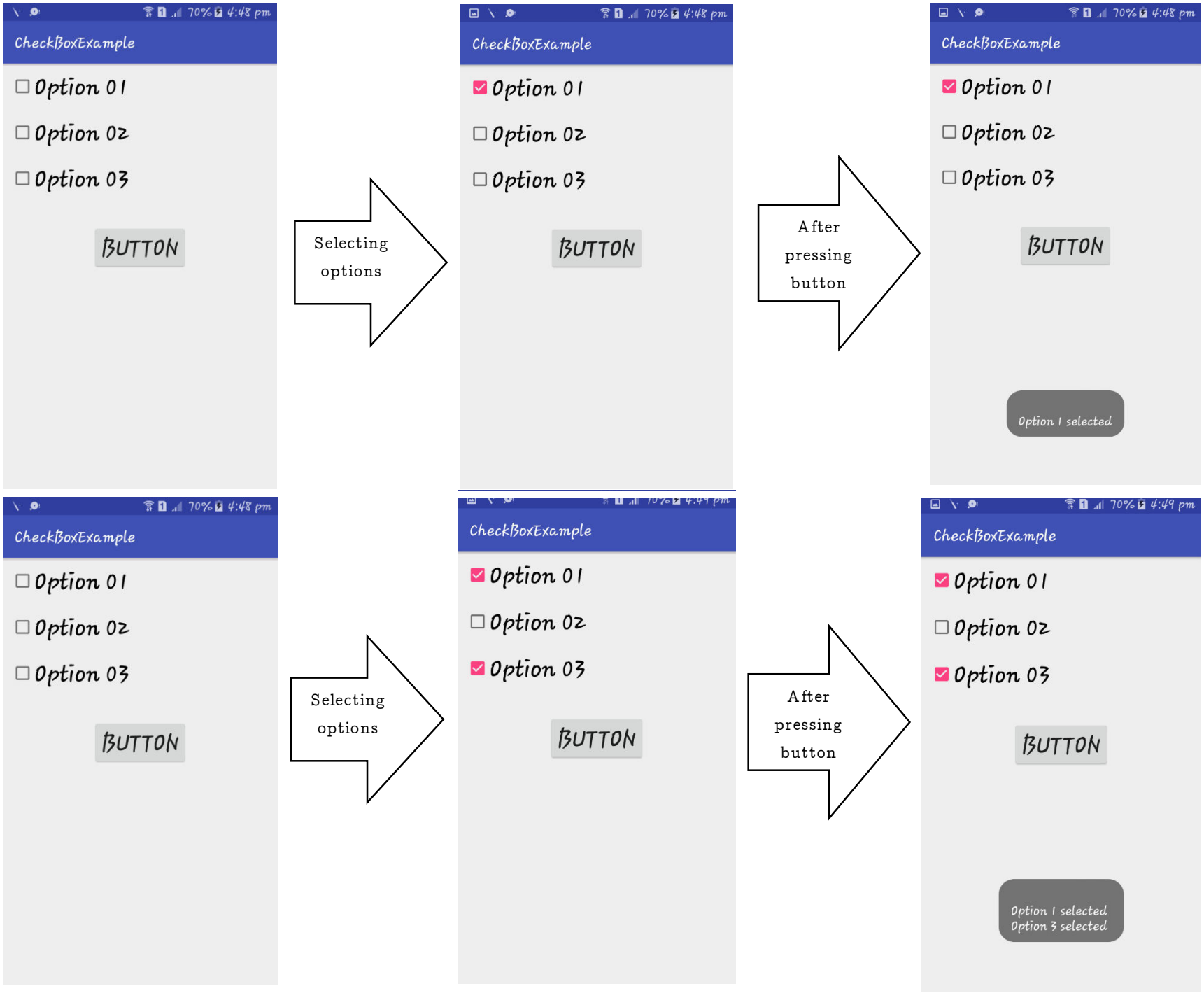
Check Box is a type of two state button either checked or unchecked.

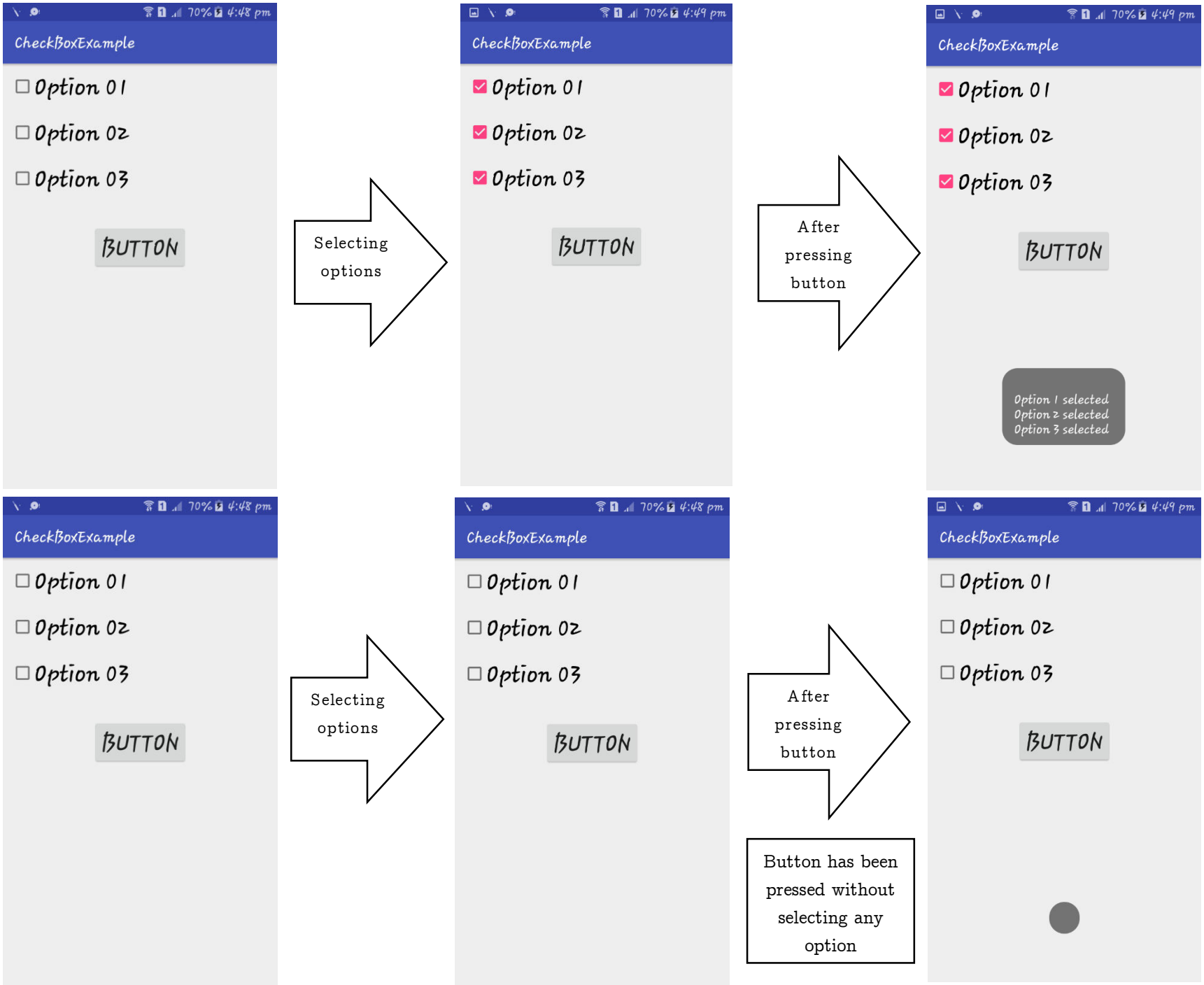
There can be a lot of usage of checkboxes. For example, it can be used to know the hobby of the user, activate/deactivate the specific action etc.

Android CheckBox class is the subclass of CompoundButton class.

MainActivity.java

```
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.Toast;

public class MainActivity extends Activity {

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

        Button btn = (Button) findViewById(R.id.btn01);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                CheckBox chk1 = (CheckBox)findViewById(R.id.chk01);
                CheckBox chk2 = (CheckBox)findViewById(R.id.chk02);
                CheckBox chk3 = (CheckBox)findViewById(R.id.chk03);

                String str="", str1="", str2="", str3="";

                if(chk1.isChecked())
                    str1="\nOption 1 selected";

                if(chk2.isChecked())
                    str2="\nOption 2 selected";

                if(chk3.isChecked())
                    str3="\nOption 3 selected";

                str= str1+str2+str3;

                Toast.makeText(getApplicationContext(), str.toString(), Toast.LENGTH_LONG).show();
            }
        });
    }
}
```

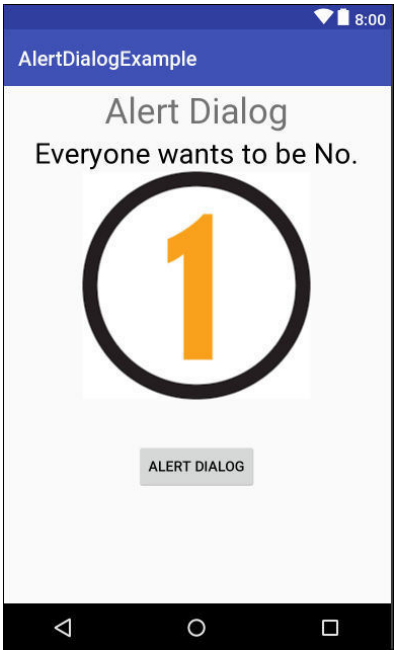

CheckBoxExample
Option 01
Option 02
Option 03
BUTTON
Selecting options
After pressing button
Option 1 selected
Option 3 selected


CheckBoxExample
Option 01
Option 02
Option 03
BUTTON
Selecting options
After pressing button
Option 1 selected
Option 2 selected
Option 3 selected
Button has been pressed without selecting any option

Alert Dialog Box

activity_main.xml

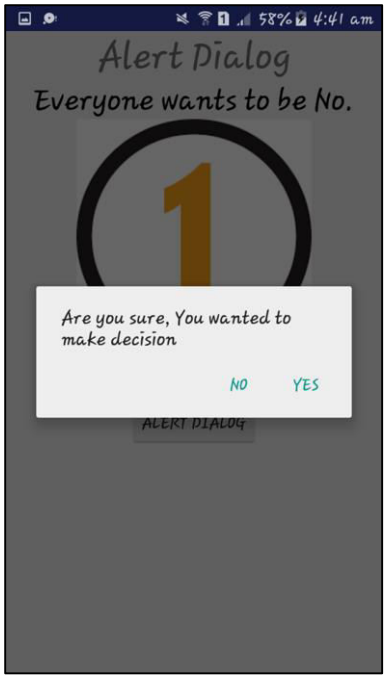
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

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

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Everyone wants to be No."
        android:id="@+id/textView2"
        android:textColor="#000"
        android:textSize="28dp"
        android:layout_below="@+id/textView"
        android:layout_centerHorizontal="true" />

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageView"
        android:layout_centerHorizontal="true"
        android:src="@drawable/image"
        android:layout_below="@+id/textView2" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Alert dialog"
        android:id="@+id/button"
        android:layout_below="@+id/imageView"
        android:layout_marginTop="42dp"
        android:onClick="open"
        android:layout_centerHorizontal="true" />

</RelativeLayout>
```



MainActivity.java

```
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void open(View view)
    {
        AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);
        alertDialogBuilder.setMessage("Are you sure, You wanted to make decision");

                alertDialogBuilder.setPositiveButton("yes", new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface arg0, int arg1) {
                                Toast.makeText(MainActivity.this,"You clicked yes button",Toast.LENGTH_LONG).show();
                            }
                        });

                alertDialogBuilder.setNegativeButton("No",new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                        finish();
            }
        });

        AlertDialog alertDialog = alertDialogBuilder.create();
        alertDialog.show();
    }
}
```

If ALERT DIALOG button is clicked



A alert dialog appears that wants to reconfirm decision with YES and NO options

If YES is pressed

A Toast appears displaying a message

If NO is pressed

[The app closes]

A Dialog is small window that prompts the user to a decision or enter additional information.

Sometimes in an application, if it is necessary to ask the user about taking a decision between yes or no in response of any particular action taken by the user, by remaining in the same activity and without changing the screen, Alert Dialog can be used.