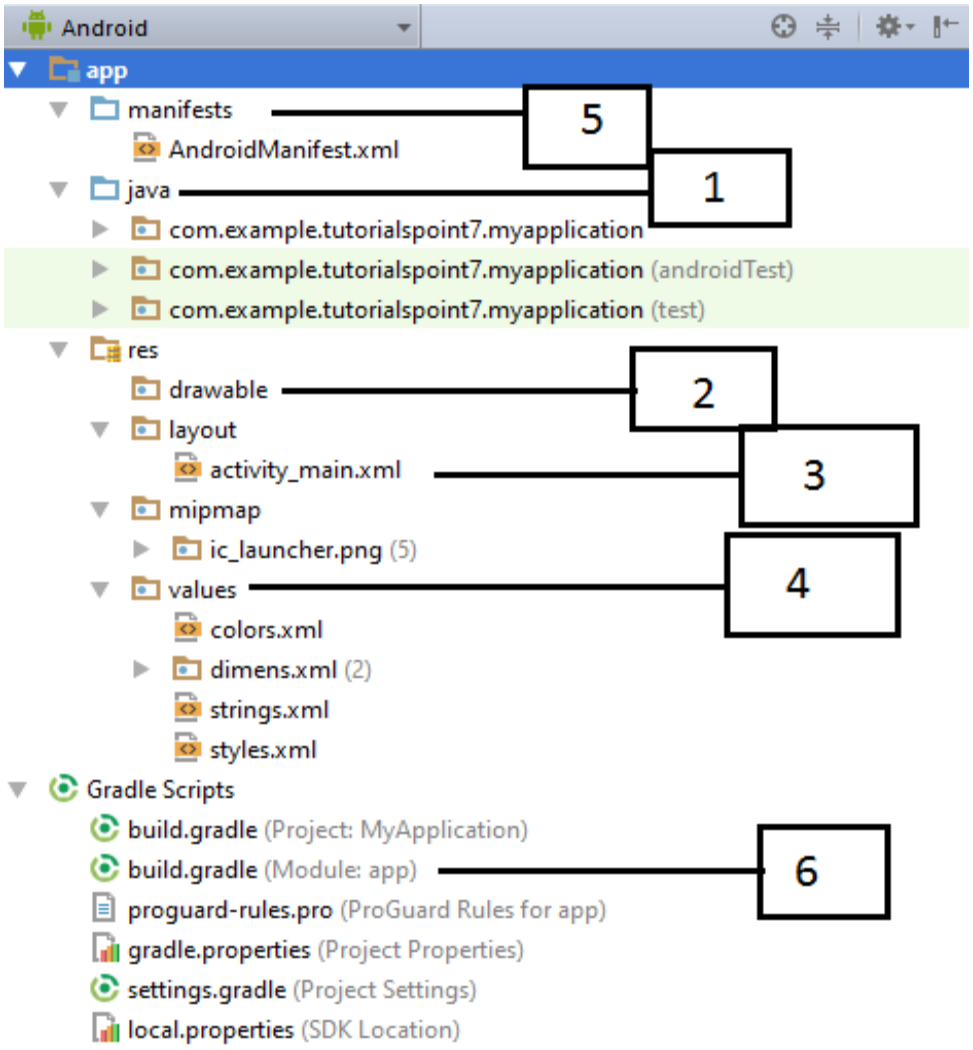
Course-B.Sc.(APS)Computer Science
Semester - VI
Paper- Android Programming

Hello world Example

You should be aware of a few directories and files in the Android project −



1. **Java file:-** This contains the **.java** source files for your project. By default, it includes an *MainActivity.java* source file having an activity class that runs when your app is launched using the app icon.

MainActivity.java

```
package com.example.helloworld;

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

public class MainActivity extends AppCompatActivity {


butoon b1 = (button)findByViewid(R.id.b2);
@Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);


   }

Public void showToast(View v){

        Toast toast = Toast.makeText(getApplicationContext(),"Welcome user", Toast.LENGTH_SHORT);
        toast.setGravity(Gravity.CENTER,20,100); //toast.setMargin(50,50);
        toast.show();
        }

}




}
```

2. **res/drawable:-** this is a directory for drawable object that are designed for high-density screens.

3. **The Layout File:-** For your "Hello World!" application, this file will have following content related to default layout –

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" >

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerHorizontal="true"
      android:layout_centerVertical="true"
      android:padding="@dimen/padding_medium"
      android:text="@string/hello_world"
      tools:context=".MainActivity" />
<button
Android:onClick = "showTost"/>

</RelativeLayout>
```

4. **The Strings File:-** **strings.xml** file is located in the *res/values* folder and it contains all the text that your application uses. For example, the names of buttons, labels, default text, and similar types of strings go into this file. This file is responsible for their textual content. For example, a default strings file will look like as following file −

```
<resources>
   <string name="app_name">HelloWorld</string>
   <string name="hello_world">Hello world!</string>
   <string name="menu_settings">Settings</string>
   <string name="title_activity_main">MainActivity</string>
</resources>
```

5. **The Manifest File:-** Whatever component you develop as a part of your application, you must declare all its components in a **manifest.xml** which resides at the root of the application project directory. This file works as an interface between Android OS and your application, so if you do not declare your component in this file, then it will not be considered by the OS. For example, a default manifest file will look like as following file –

Manifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.tutorialspoint7.myapplication">

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

      <activity android:name=".MainActivity">
         <intent-filter>
```

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

Following is the list of tags which you will use in your manifest file to specify different Android application components −

- <activity>elements for activities
- <service> elements for services
- <receiver> elements for broadcast receivers
- <provider> elements for content providers

Accessing Resources in Code

```
helloworld/
   app/
      manifest/
         AndroidManifest.xml
   java/
      MyActivity.java
      res/
         drawable/
            icon.png
            background.png
         drawable-hdpi/
            myimage.png
            background.png
         layout/
            activity_main.xml
         values/
            strings.xml
```

.java file

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

}
```

Example

Consider a layout *res/layout/activity_main.xml* with the following definition –

android:id, android:gravity, android:orientation

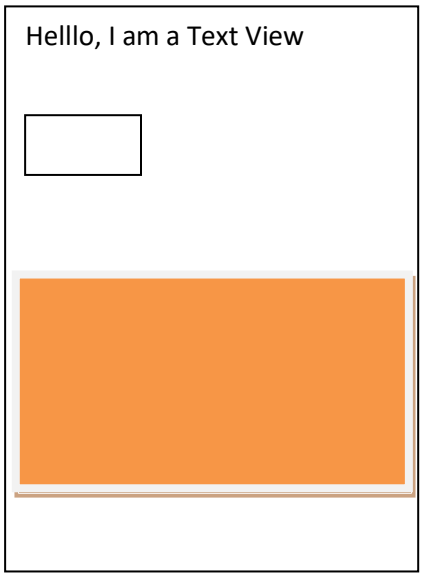
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android=http://schemas.android.com/apk/res/android
android:id = l1
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical " >

  <TextView   android:id="@+id/msg"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/”hello” />

<Button  android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello, I am a Button" />

 <ImageView   android:id="@+id/myIv"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:src="@drawable/myimage" />

</LinearLayout>
```



Example

To access ***res/drawable/myimage.png*** and set an ImageView you will use following code −

```
ImageView Iv = (ImageView) findViewById(R.id.myIv);
LinearLayout Ll = (LinearLayout) findViewById(R.id.L1);

L1.setImageResource(R.drawable.myimage);
```

Example

Consider next example where *res/values/strings.xml* has following definition −

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="app_name">HelloWorld</string>
  <string  name="hello">Hello, World!</string>
</resources>
```

Now you can set the text on a TextView object with ID msg using a resource ID as follows −

TextView Tv = (TextView) findViewById(R.id.Tv);
Tv.**setText(**R.string.hello);

Now you can use these resources in the following layout file to set the text color and text string as follows −

<?xml version="1.0" encoding="utf-8"?>
<TextView
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:textColor="**@color/opaque_red**"
  android:text="**@string/hello**" />

Consider the following resource XML *res/values/strings.xml* file that includes a color resource and a string resource −

<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="opaque_red">#f00</color>
  <string name="hello">Hello!</string>
</resources>

Activity Life Cycle:-

MainActivity.java:- This file includes each of the fundamental life cycle methods. The **Log.d()** method has been used to generate log messages –

```
package com.example.helloworld;

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

public class MainActivity extends Activity {
   String msg = "MainActivity";

   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      Log.d(msg, "The onCreate() event");
   }

   /** Called when the activity is about to become visible. */
   @Override
   protected void onStart() {
      super.onStart();
      Log.d(msg, "The onStart() event");
   }

   /** Called when the activity has become visible. */
```

```
    @Override
    protected void onResume() {
        super.onResume();
        Log.d(msg, "The onResume() event");
    }

    /** Called when another activity is taking focus. */
    @Override
    protected void onPause() {
        super.onPause();
        Log.d(msg, "The onPause() event");
    }

    /** Called when the activity is no longer visible. */
    @Override
    protected void onStop() {
        super.onStop();
        Log.d(msg, "The onStop() event");
    }

    /** Called just before the activity is destroyed. */
    @Override
    public void onDestroy() {
        super.onDestroy();
        Log.d(msg, "The onDestroy() event");
    }
}
```

Output:-

```
23 10:32:07.682 4480-4480/com.example.helloworld D/Android :: The onCreate() event
08-23 10:32:07.683 4480-4480/com.example.helloworld D/Android :: The onStart() event
08-23 10:32:07.685 4480-4480/com.example.helloworld D/Android :: The onResume() event
```

When you lock your screen

```
08-23 10:32:53.230 4480-4480/com.example.helloworld D/Android :: The onPause() event
08-23 10:32:53.294 4480-4480/com.example.helloworld D/Android :: The onStop() event
```

again try to unlock your screen on the Android emulator

```
08-23 10:34:41.390 4480-4480/com.example.helloworld D/Android :: The onStart() event
08-23 10:34:41.392 4480-4480/com.example.helloworld D/Android :: The onResume() event
```

again try to click Back button 

```
08-23 10:37:24.806 4480-4480/com.example.helloworld D/Android :: The onPause() event
08-23 10:37:25.668 4480-4480/com.example.helloworld D/Android :: The onStop() event
08-23 10:37:25.669 4480-4480/com.example.helloworld D/Android :: The onDestroy() event
```

2. A service is a component that runs in the background to perform long-running operations without needing to interact with the user and it works even if application is destroyed.

```
public class MainActivity extends Activity {
    String msg = "MainActivity";

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d(msg, "The onCreate() event");
    }
```

```
   public void startService(View view) {

        Intent i = new Intent(getBaseContext(), ActivityB.class);
        startActivity(i);
   }

   // Method to stop the service
   public void stopService(View view) {
      stopService(new Intent(getBaseContext(), ActivityB.class));
   }
}
```

ActivityB.jave

```
public class ActivityB extends Activity {
   String msg = "ActivityB";

   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      Log.d(msg, "The onCreate() event");
   }

public void onDestroy() {
      super.onDestroy();
      Toast.makeText(this, "Service Destroyed", Toast.LENGTH_LONG).show();
   }
}
```

AndroidManifest.xml

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

      <activity android:name=".ActivityB”>
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />

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

      <service android:name=".MainActivity" />
   </application>
```

Componant of android application

1. Activity –represents a single screen in an app.
   startActivity(Intent i)
2. Service – component that performs operations in the background without user interface.
   startService(Intent i)
3. Broadcast Reciver
   sendBroadcast(Intent i)

There two types of intents:

1. Explicit intent-specify component name
   Intent inte = new Intent(MainActivity.this, ActivityB.class);
   startActivity(inte);
2. Implicit intent – do not specify component name
   Intent inte = new Intent();
   Inte.setAction(inte.ACTION_VIEW);
   Inte.setType("text/plain");
   Inte.setData(Uri.parse("http://www.gooogle.com");
   Inte.putExtra(inte.EXTRA_TEXT, text);
   if(inte.resolveActivity(getPackageManager()!=null)
   {
   startActivity(inte);
   }

Intent filter

In android, **Intent Filter** is an expression in the app's **manifest** file (**ActivityMainfest.xml**) and it is used to specify the type of intents that the component would like to receive. In case if we create **Intent Filter** for an activity, there is a possibility for other apps to start our activity by sending a certain type of intent otherwise the activity can be started only by an explicit intent.

Generally, the Intent Filters (<**intent-filter**>) whatever we define in the manifest file can be nested in the corresponding app components and we can specify the type of intents to accept using these three elements.

<action>

ACTION_SEND

ACTION_VIEW

It defines the name of an intended action to be accepted and it must be a literal string value of an action, not the class constant.

<category>

It defines the name of an intent category to be accepted and it must be the literal string value of an action, not the class constant.

<data>

setData()
setType()
It defines the type of data to be accepted and by using one or more attributes we can specify various aspects of the data URI (scheme, host, port, path) and MIME type.

<Extras>
putExtras(EXTRA_EMIL, value)
putExtras(EXTRA_SUBJECT, value)
Bundle object with all the extra data, then insert the Bundle in the Intent with putExtras()

Intent Filter in Manifest File

Following is the code snippet of defining an activity with Intent Filter (**<intent-filter>**) in the Android Manifest file (**AndroidManifest.xml**) like as shown below.

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

    </intent-filter>
</activity>

 <activity android:name="ActivityB">
    <intent-filter>
      <action android:name="android.intent.action.SEND" />
       <category android:name="android.intent.category.DEFAULT" />
       <data android:mimeType="text/plain"/>
    </intent-filter>
</activity>
```

LinearLayout is one of the UI Group, by using LinearLayout we can arrange the UI Group, by using LinearLayout we can arrange the UI components either in vertical manner or in horizontal manner on after another.

Syntax

```
<LinearLayout
        android:Layout_width = “wrap_content”
        android:Layout_height=”wrap_content”
        android:orientation = “vertical”>
</LinearLayout>
```

RelativeLayout is a view group that displays child views in relative positions. The position of each view can be specified as relative to sibling elements (such as to the left-of or below another view) or in positions relative to the parent *RelativeLayout* area (such as aligned to the bottom, left or center).

Syntax

```
<RelativeLayout
        android:alignParentTop = “true”
        android:alignParentLeft = “true”
        android:layout_marginTop = “54dp”
        android:layout_marginLeft = “40dp”>
</RelativeLayout>
```

Top/Left	Top/Right
Bottom/Left	Bottom/Right

Button move margin will change relatively

Application 3rd

Create an application with first activity with an editText and send button. On click of send button, make use of explicit intent to send text to second activity and display there in text view.

activity_main.xml

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


    <EditText android:id="@+id/EditText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:hint="Enter Your Name..."
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/send"
        android:onClick="send"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Send Button"
        android:layout_marginTop="450dp"
        android:layout_marginLeft="200dp"
        />

</RelativeLayout>
```

MainActivity.java

```java
package com.example.app3;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
 import android.widget.EditText;

public class MainActivity extends AppCompatActivity {

   EditText e1;

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

         e1=(EditText)findViewById(R.id.EditText);

   }
   public void send(View view) {
      String name=e1.getText().toString();
      Intent i=new Intent(MainActivity.this,SecondActivity.class);
      i.putExtra("name",name);
      startActivity(i);

   }
}
```

activity_second.xml

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

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="169dp"
        android:layout_marginLeft="169dp"
        android:layout_marginTop="150dp"
        android:layout_marginEnd="183dp"
        android:layout_marginRight="183dp"
        android:layout_marginBottom="653dp"
        android:text="NextActivity"
        android:textSize="30sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.521"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0" />

    <TextView android:id="@+id/t2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="167dp"
        android:layout_marginLeft="167dp"
        android:layout_marginTop="220dp"
        android:layout_marginEnd="184dp"
        android:layout_marginRight="184dp"
        android:layout_marginBottom="573dp"
        android:text="Welcome"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

SecondActivity.java

```
package com.example.app3;

import androidx.appcompat.app.AppCompatActivity;
 import android.os.Bundle;
import android.widget.TextView;

public class SecondActivity extends AppCompatActivity {

   TextView tV;

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

     tV=(TextView)findViewById(R.id.t2);

      Bundle bundle=getIntent().getExtras();

      String name1=bundle.getString("name");

           tV.setText("Welcome "+name1);

   }
}
```

Application 4th

Create an application with first activity with an editText and send button. On click of send button, make use of implicit intent that uses a SEND ACTION and let user select app from app chooser and navigate to that application.

activity_main.xml

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

   <EditText android:id="@+id/EditText"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerInParent="true"
      android:hint="Enter Your Name..."
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintRight_toRightOf="parent"
      app:layout_constraintTop_toTopOf="parent" />
  <Button
      android:id="@+id/send"
      android:onClick="sendEmail"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Send Button"
      android:layout_marginTop="450dp"
      android:layout_marginLeft="200dp"
      />

  <Button
      android:id="@+id/send1"
      android:onClick="openBrowser"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="open browser"
      android:layout_marginTop="450dp"
      android:layout_marginLeft="200dp"
      />


</RelativeLayout>
```

MainActivity.java

```
package com.example.app4;

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

public class MainActivity extends AppCompatActivity {

    EditText input;

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

     input=(EditText)findViewById(R.id.EditText);
  }

  public void sendEmail(View v) {

     Intent in=new Intent(Intent.ACTION_SEND);
     in.setType("text/plain");
     in.putExtra(in.EXTRA_SUBJECT,"Android Classes");

     String msg=input.getText().toString();
     in.putExtra(in.EXTRA_TEXT,msg);

  if (in.resolveActivity(getPackageManager()!=null
  {
        startActivity(in);
  }
  }//sendEmail

  Public void openBrowser (View v){

  Intent in = new Intent();

  In.setAction(Intent.ACTION_VIEW);
  In.setData(Uri.parse(http://www.yahoo.com));

  if (in.resolveActivity(getPackageManager()!=null
  {
        startActivity(in);
  }

  }

}
 // Intent in = Intent.createChooser(in, "open website using");
```

From:-
Ritu Meena
Shivaji Collage