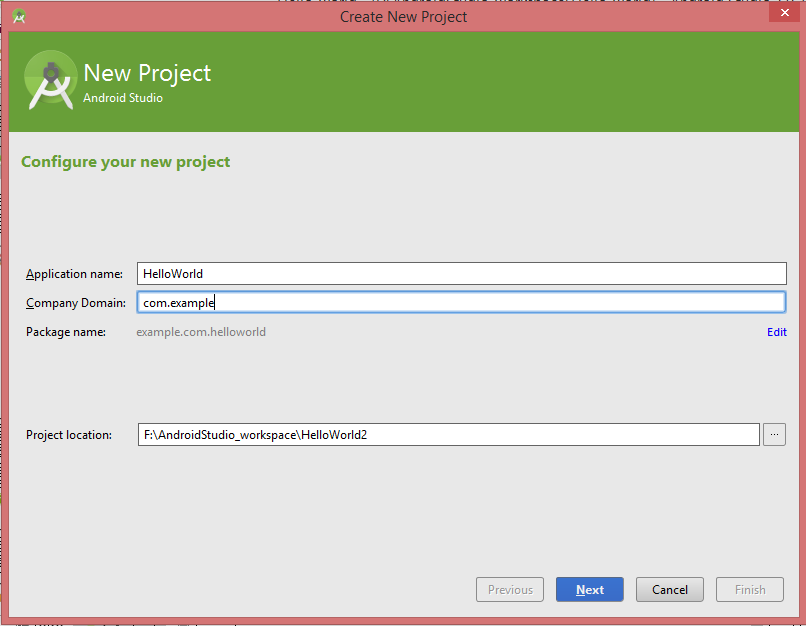
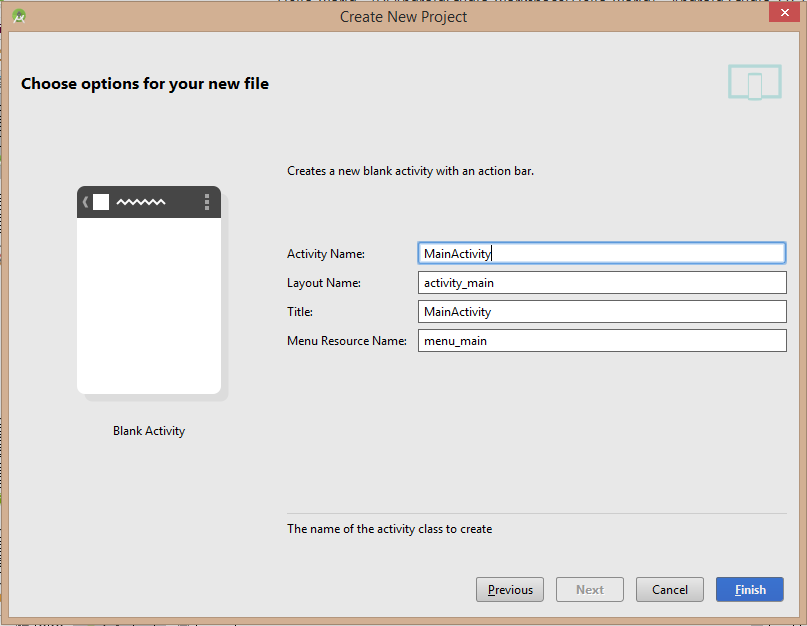
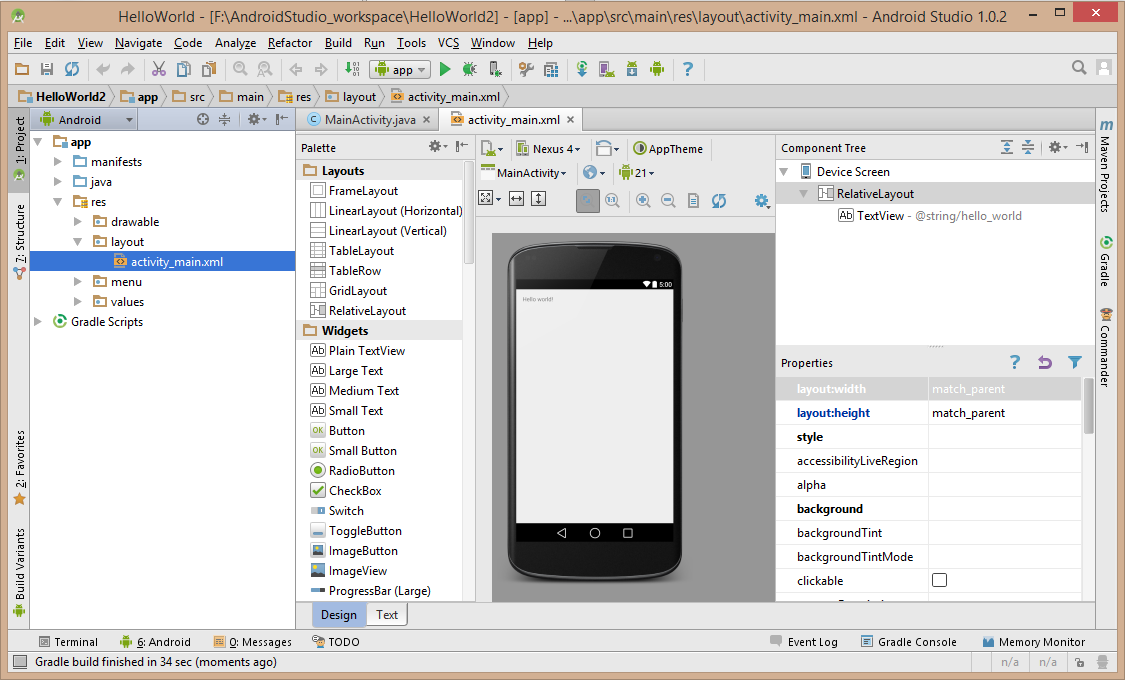
Lab 2: Android Activities (Android Studio)

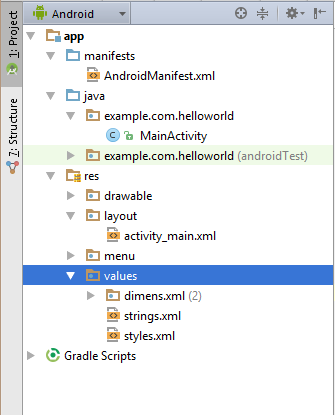
**Step 1:** Create a simple Android Application using Android Studio. Follow the option **File -> NewProject**. Now name your application as HelloWorld using the wizard window as follows:





Now we have a **HelloWorld** application:





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

1. manifests.

**AndroidManifest.xml**: It is the manifest file which describes the fundamental characteristics of the app and defines each of its components.

2. java.  
It contains the .java source files for your project. By default, it includes a MainActivity.java source file having an activity class that runs when your app is launched using the app icon.

3. res/drawable.  
This is a directory for drawable objects that are designed for high-density screens.

4.res/layout.  
This is a directory for files that define your app's user interface.

5. res/values.  
This is a directory for other various XML files that contain a collection of resources, such as strings and colors definitions.

**Step 2:**

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

package example.com.helloworld;

import android.app.Activity;

import android.os.Bundle;

import android.util.Log;

import com.example.helloworld.R;

public class MainActivity extends Activity {

String msg = "Android : ";

/\*\* 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");

}

}

An application can have one or more activities without any restrictions. Every activity you define for your application must be declared in your AndroidManifest.xml file and the main activity for your app must be declared in the manifest with an <intent-filter> that includes the MAIN action and LAUNCHER category as follows:

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

package="com.example.helloworld"

android:versionCode="1"

android:versionName="1.0" >

<uses-sdk

android:minSdkVersion="8"

android:targetSdkVersion="15" />

<application

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

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

android:theme="@style/AppTheme" >

<activity

android:name="example.com.helloworld.MainActivity"

android:label="@string/hello\_world" >

<intent-filter>

<action android:name="android.intent.action.MAIN" />

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

</intent-filter>

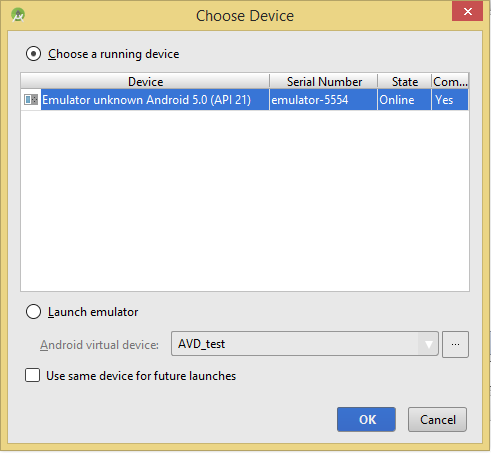
</activity>

</application>

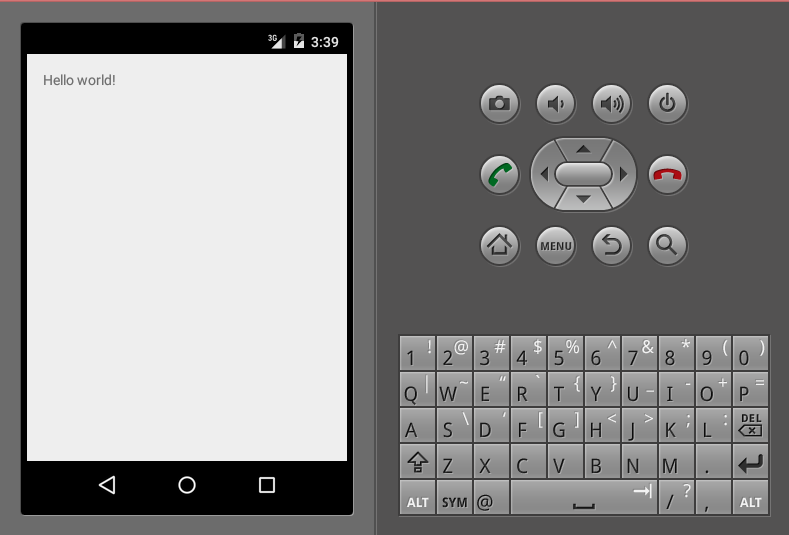
</manifest>

**Step 3：**

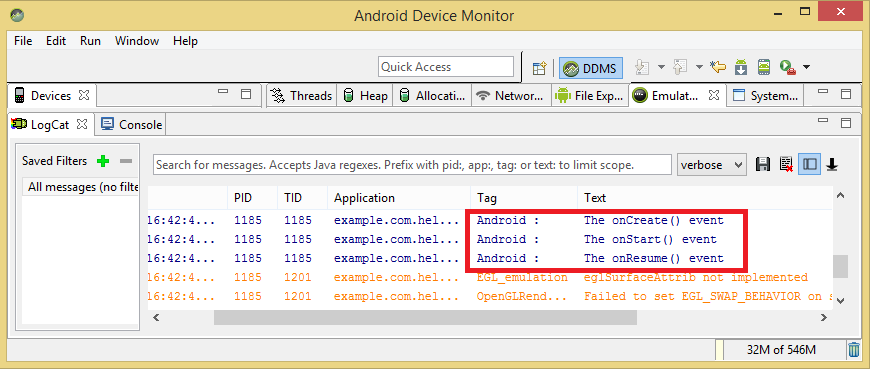
Let's try to run our modified **Hello World application** we just modified. I assume you had created your AVD while doing environment setup. At first, start your existed **AVD**, and then click **Run** Eclipse Run Icon icon from the toolbar. Choose this AVD, and click “OK”:



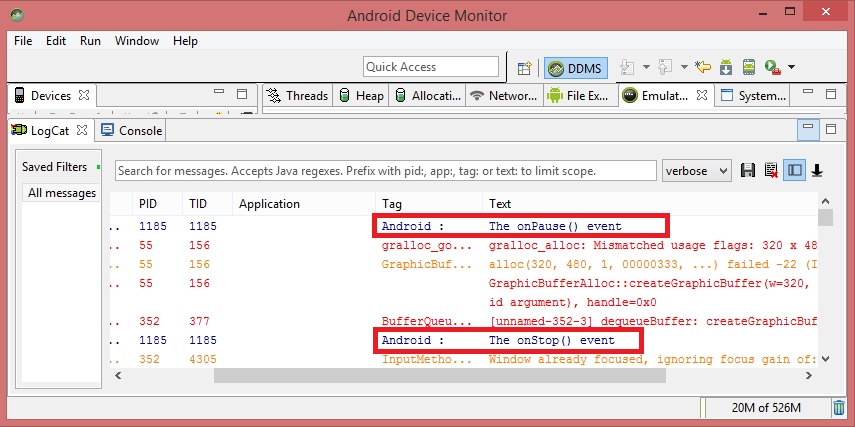
Then the AVD will display like as following:



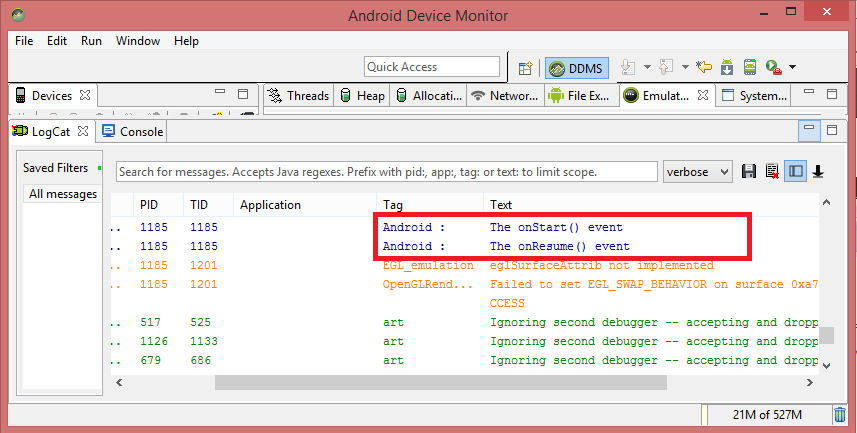
In the Android Device Manager (click  from the toolbar) which has **LogCat** window print out some log which we wrote in MainActivity.java:



When we try to click Red button Android Red Buttonon the Android emulator and it will generate following events messages in LogCat window:



Let us again try to click Menu button Android Menu Buttonon the Android emulator and it will generate following events messages in LogCat window:



Next, let us again try to click Back button Android Back Buttonon the Android emulator and it will generate following events messages in LogCat window and this completes the Acitivity Life Cycle for an Android Application.

(Because the log is too much, so we can search the information on the top of this window)

