ANDROID

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.

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

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

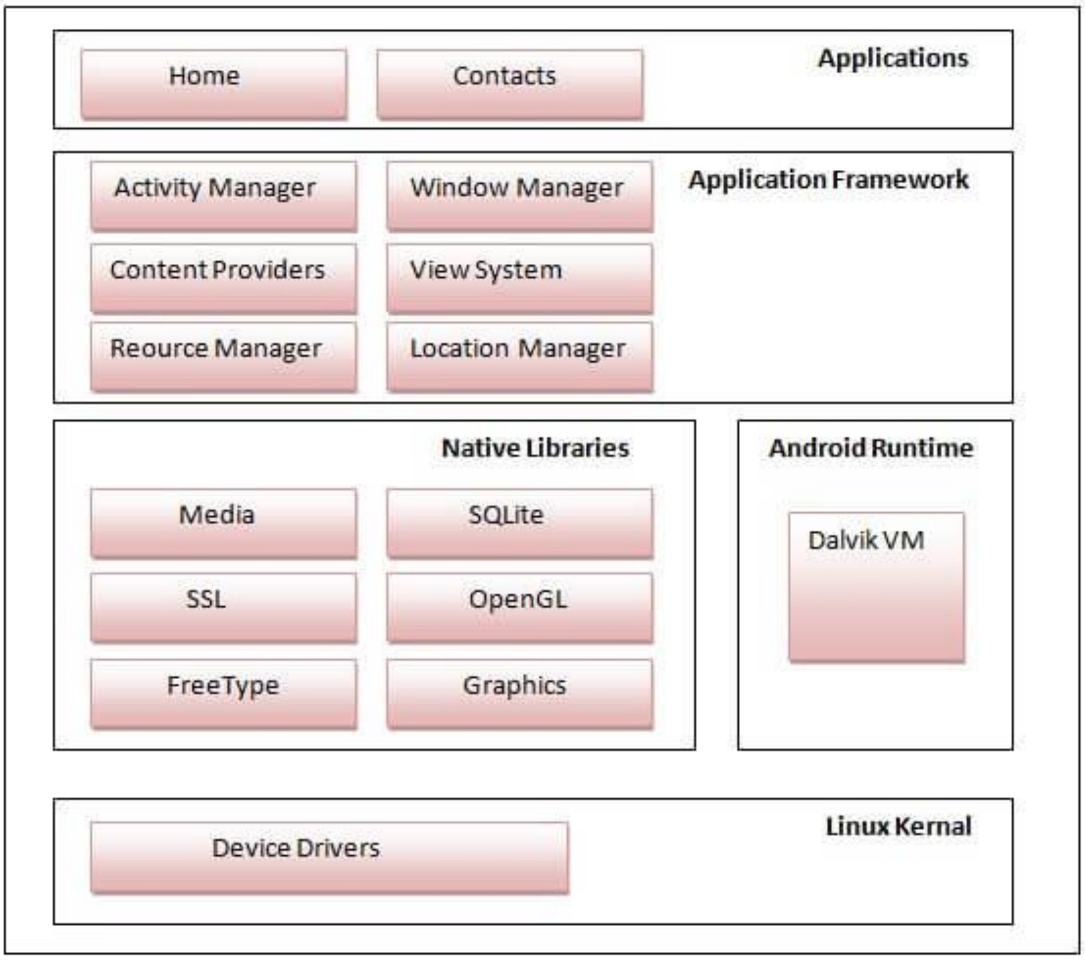
The goal of android project is to create a successful real-world product that improves the mobile experience for end users.

Android Architecture

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

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

Let's see the android architecture first.



1) Linux kernel

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

---

2) Native Libraries

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

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

---

3) Android Runtime

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

---

4) Android Framework

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

---

5) Applications

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

Android Core Building Blocks

Intent
Activity
Service
Broadcast Receiver
Content Provider

An android **component** is simply a piece of code that has a well defined life cycle e.g. Activity, Receiver, Service etc.

The **core building blocks** or **fundamental components** of android are activities, views, intents, services, content providers, fragments and AndroidManifest.xml.

Activity

An activity is a class that represents a single screen. It is like a Frame in AWT.

View

A view is the UI element such as button, label, text field etc. Anything that you see is a view.

Intent

Intent is used to invoke components. It is mainly used to:

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

How to make android apps

In this page, you will know how to create the simple hello android application. We are creating the simple example of android using the Eclipse IDE. For creating the simple example:

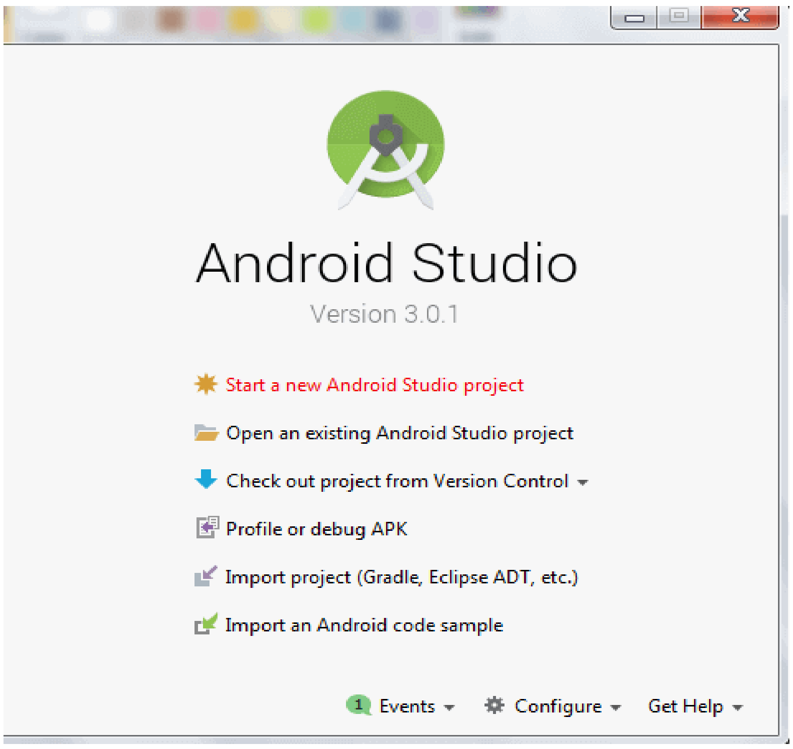
1. Create the new android project
2. Write the message (optional)
3. Run the android application

You need to follow the 3 steps mentioned above for creating the Hello android application.

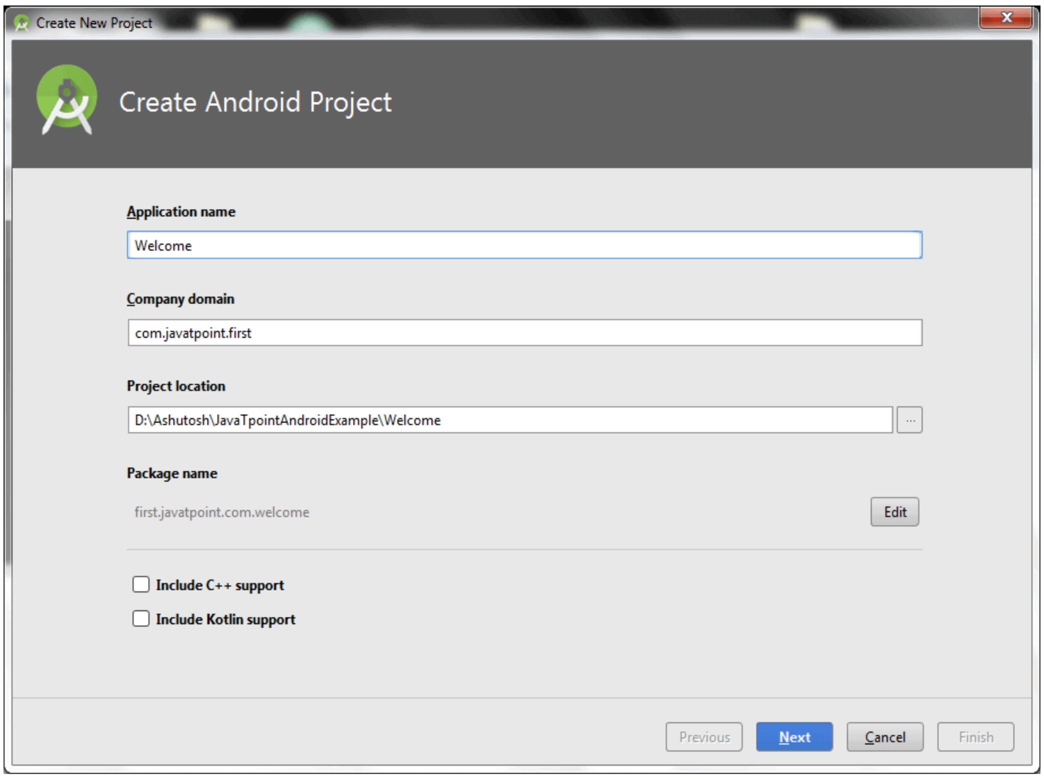
1) Create the New Android project

For creating the new android studio project:

1) Select *Start a new Android Studio project*


Android Studio
Version 3.0.1
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
Events
Configure
Get Help

2) Provide the following information: Application name, Company domain, Project location and Package name of application and click next.



Application name
Welcome

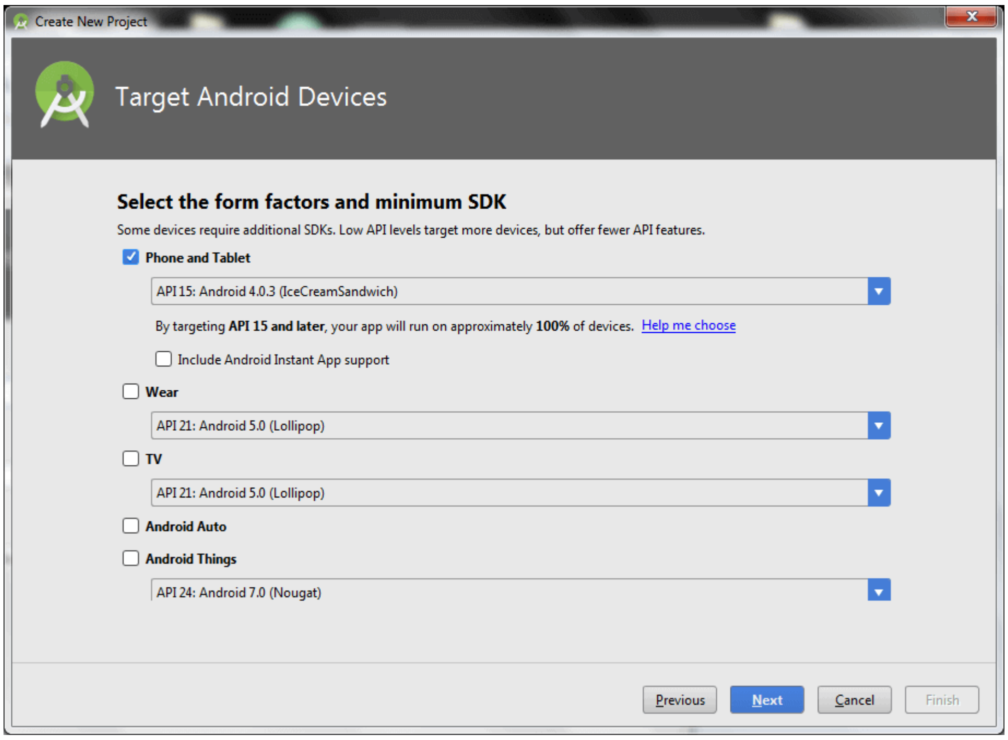
Company domain
com.javatpoint.first

Project location
D:\Ashutosh\JavaTpointAndroidExample\Welcome

Package name
first.javatpoint.com.welcome Edit

- [ ] Include C++ support
- [ ] Include Kotlin support

3) Select the API level of application and click next.



Select the form factors and minimum SDK

Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

[✓] Phone and Tablet

API 15: Android 4.0.3 (IceCreamSandwich)

By targeting **API 15 and later**, your app will run on approximately **100%** of devices. Help me choose

[ ] Include Android Instant App support

[ ] Wear

API 21: Android 5.0 (Lollipop)

[ ] TV

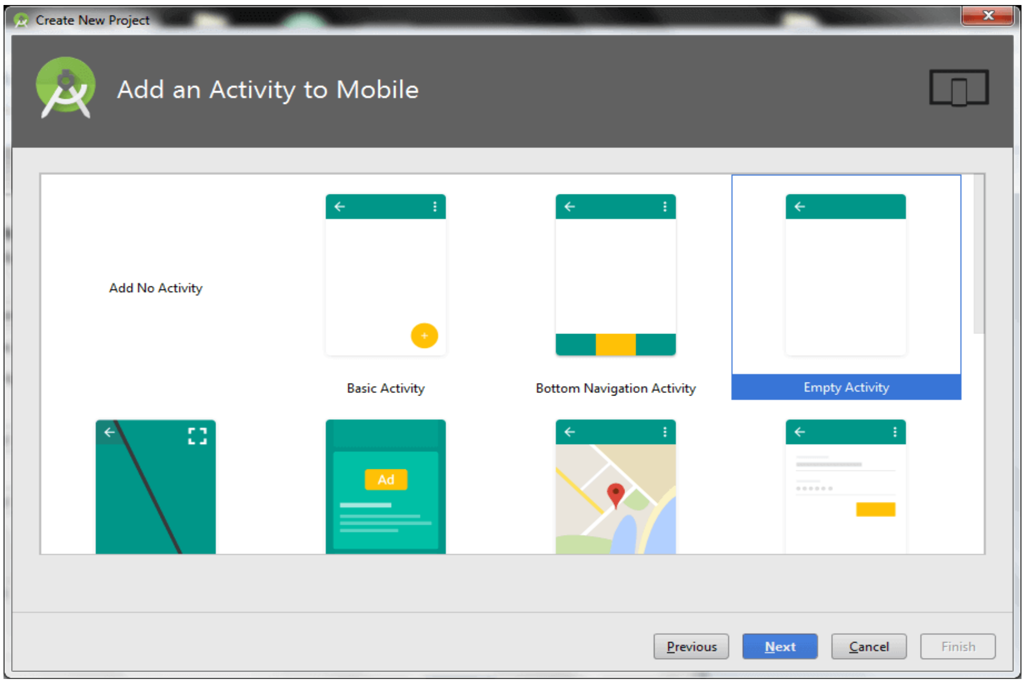
API 21: Android 5.0 (Lollipop)

[ ] Android Auto

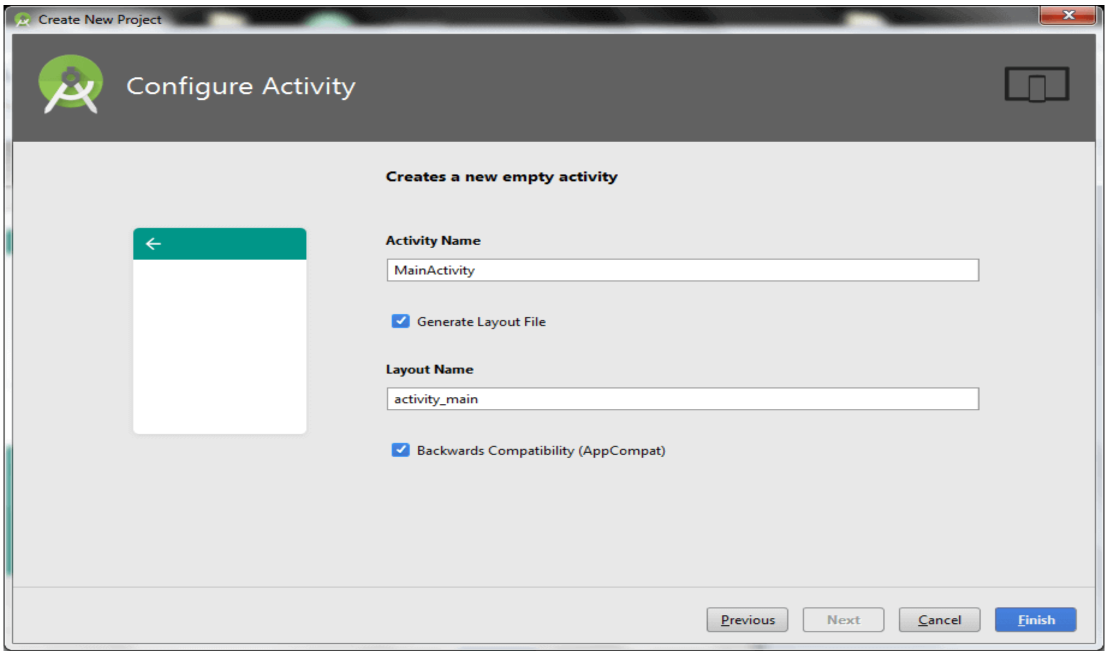
[ ] Android Things

API 24: Android 7.0 (Nougat)

4) Select the Activity type (Empty Activity).



5) Provide the Activity Name and click finish.



Creates a new empty activity

Activity Name: MainActivity

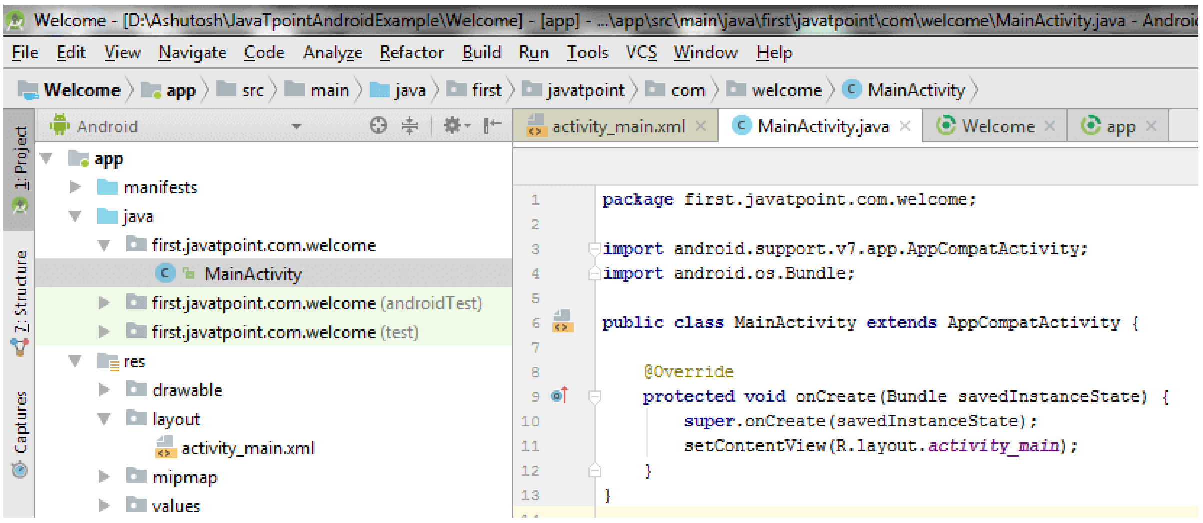
[✓] Generate Layout File

Layout Name: activity_main

[✓] Backwards Compatibility (AppCompat)

After finishing the Activity configuration, Android Studio auto generates the activity class and other required configuration files.

Now an android project has been created. You can explore the android project and see the simple program, it looks like this:


Welcome - [D:\Ashutosh\JavaTpointAndroidExample\Welcome] - [app] - ...\app\src\main\java\first\javatpoint\com\welcome\MainActivity.java - Androi
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Welcome app src main java first javatpoint com welcome MainActivity
Android
app
manifests
java
first.javatpoint.com.welcome
MainActivity
first.javatpoint.com.welcome (androidTest)
first.javatpoint.com.welcome (test)
res
drawable
layout
activity_main.xml
mipmap
values
activity_main.xml
MainActivity.java
Welcome
app
1: Project
Z: Structure
Captures

```
package first.javatpoint.com.welcome;

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);
    }
}
```

2) Write the message

File: activity_main.xml

Android studio auto generates code for activity_main.xml file. You may edit this file according to your requirement.

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

<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/ap
k/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="first.javatpoint.com.welcome.MainActivity">

<TextView

  android:layout_width="wrap_content"

  android:layout_height="wrap_content"

  android:text="Hello Android!"

  app:layout_constraintBottom_toBottomOf="parent"

  app:layout_constraintLeft_toLeftOf="parent"

  app:layout_constraintRight_toRightOf="parent"

  app:layout_constraintTop_toTopOf="parent" />


</android.support.constraint.ConstraintLayout>

}
```

File: MainActivity.java

```
package first.javatpoint.com.welcome;
```

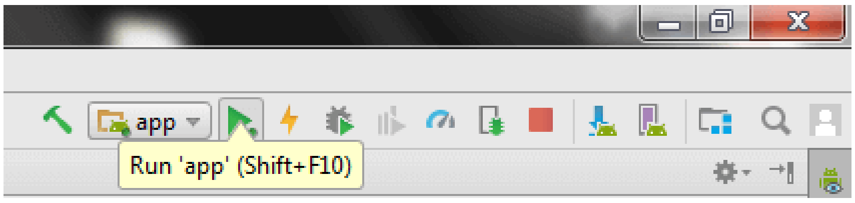
```
import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

  public class MainActivity extends AppCompatActivity {

  @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

    }

}
```

To understand the first android application, visit the next page (internal details of hello android example).

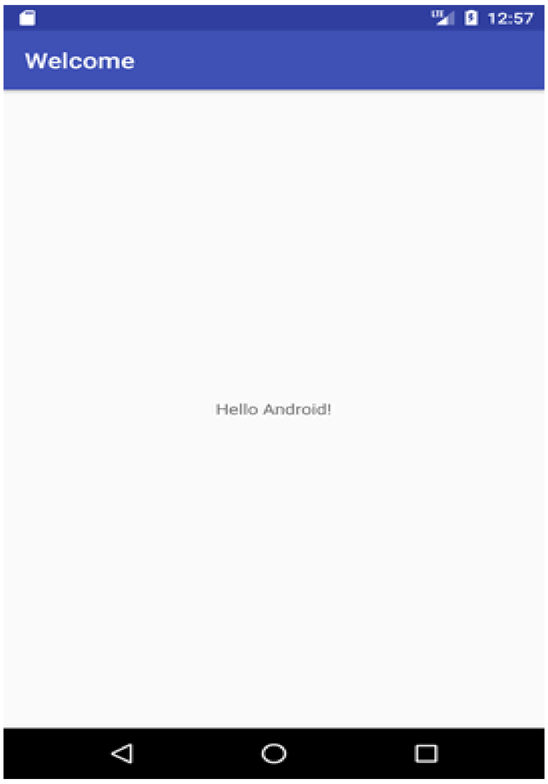
---

3) Run the android application

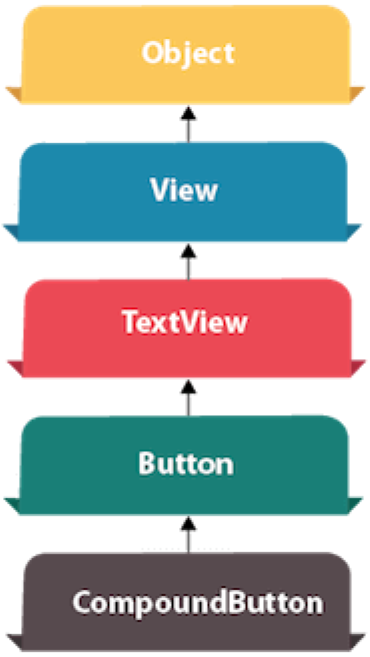
To run the android application, click the run icon on the toolbar or simply press Shift + F10.



The android emulator might take 2 or 3 minutes to boot. So please have patience. After booting the emulator, the android studio installs the application and launches the activity. You will see something like this:



Android Button Example



Android Button represents a push-button. The android.widget.Button is subclass of TextView class and CompoundButton is the subclass of Button class.

There are different types of buttons in android such as RadioButton, ToggleButton, CompoundButton etc.

Android Button Example with Listener

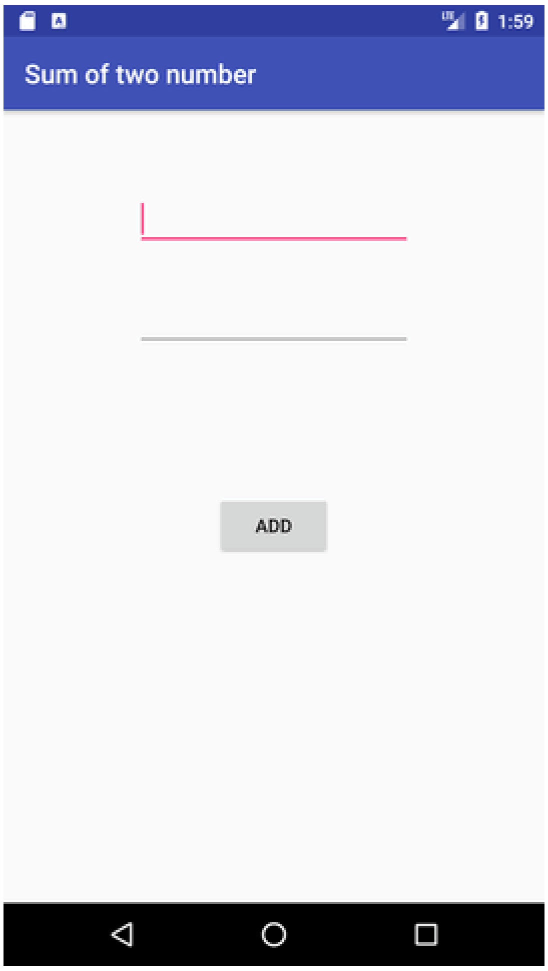
Here, we are going to create two textfields and one button for sum of two numbers. If user clicks button, sum of two input values is displayed on the Toast.

We can perform action on button using different types such as calling listener on button or adding onClick property of button in activity's xml file.

```
button.setOnClickListener(new View.OnClickListener() {
            @Override
             public void onClick(View view) {
               //code
             }
});
<Button
        android:onClick="methodName"
/>
```

Drag the component or write the code for UI in activity_main.xml

First of all, drag 2 textfields from the Text Fields palette and one button from the Form Widgets palette as shown in the following figure.



The generated code for the ui components will be like this:

File: 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="example.javatpoint.com.sumoftwonumber.MainActivity">

    <EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="61dp"
        android:ems="10"
        android:inputType="number"
        tools:layout_editor_absoluteX="84dp"
        tools:layout_editor_absoluteY="53dp" />

    <EditText
        android:id="@+id/editText2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/editText1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="32dp"
        android:ems="10"
        android:inputType="number"
        tools:layout_editor_absoluteX="84dp"
        tools:layout_editor_absoluteY="127dp" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:layout_below="@+id/editText2"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="109dp"
        android:text="ADD"
        tools:layout_editor_absoluteX="148dp"
        tools:layout_editor_absoluteY="266dp" />
</RelativeLayout>
```

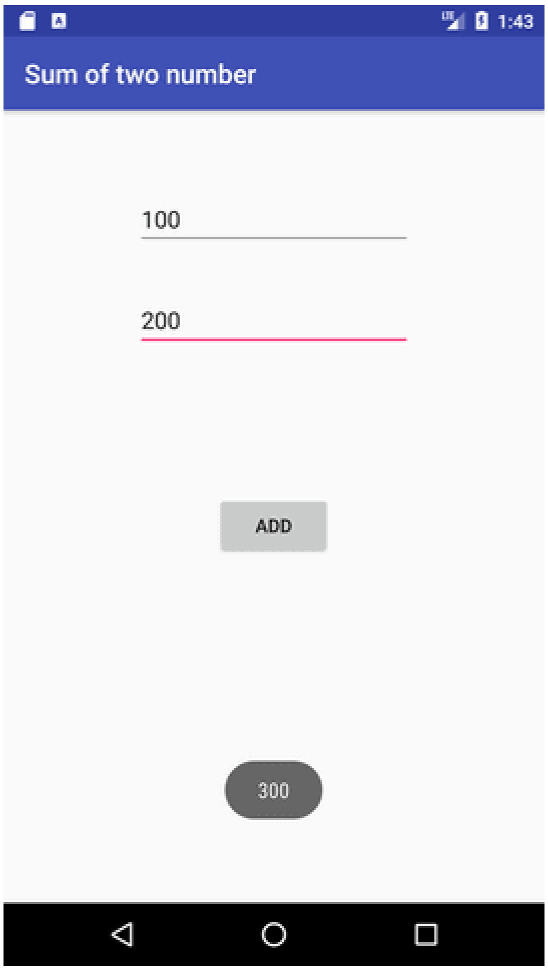
Activity class

Now write the code to display the sum of two numbers.

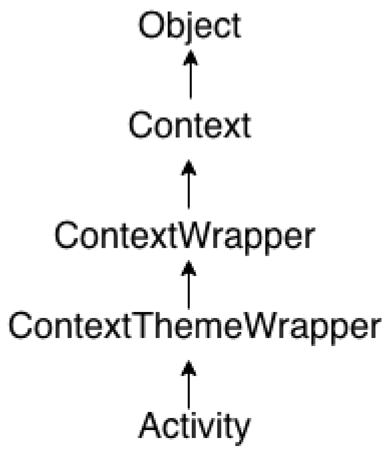
File: MainActivity.java

```
package example.javatpoint.com.sumoftwonumber;

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

public class MainActivity extends AppCompatActivity {
    private EditText edittext1, edittext2;
    private Button buttonSum;

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

        addListenerOnButton();
    }

    public void addListenerOnButton() {
        edittext1 = (EditText) findViewById(R.id.editText1);
        edittext2 = (EditText) findViewById(R.id.editText2);
        buttonSum = (Button) findViewById(R.id.button);

        buttonSum.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String value1=edittext1.getText().toString();
                String value2=edittext2.getText().toString();
                int a=Integer.parseInt(value1);
                int b=Integer.parseInt(value2);
                int sum=a+b;
                Toast.makeText(getApplicationContext(),String.valueOf(sum), Toast.LENGTH
_LONG).show();
            }
        });
    }
}
```

Output:



Android Activity Lifecycle



Android Activity Lifecycle is controlled by 7 methods of android.app.Activity class. The android Activity is the subclass of ContextThemeWrapper class.

An activity is the single screen in android. It is like window or frame of Java.

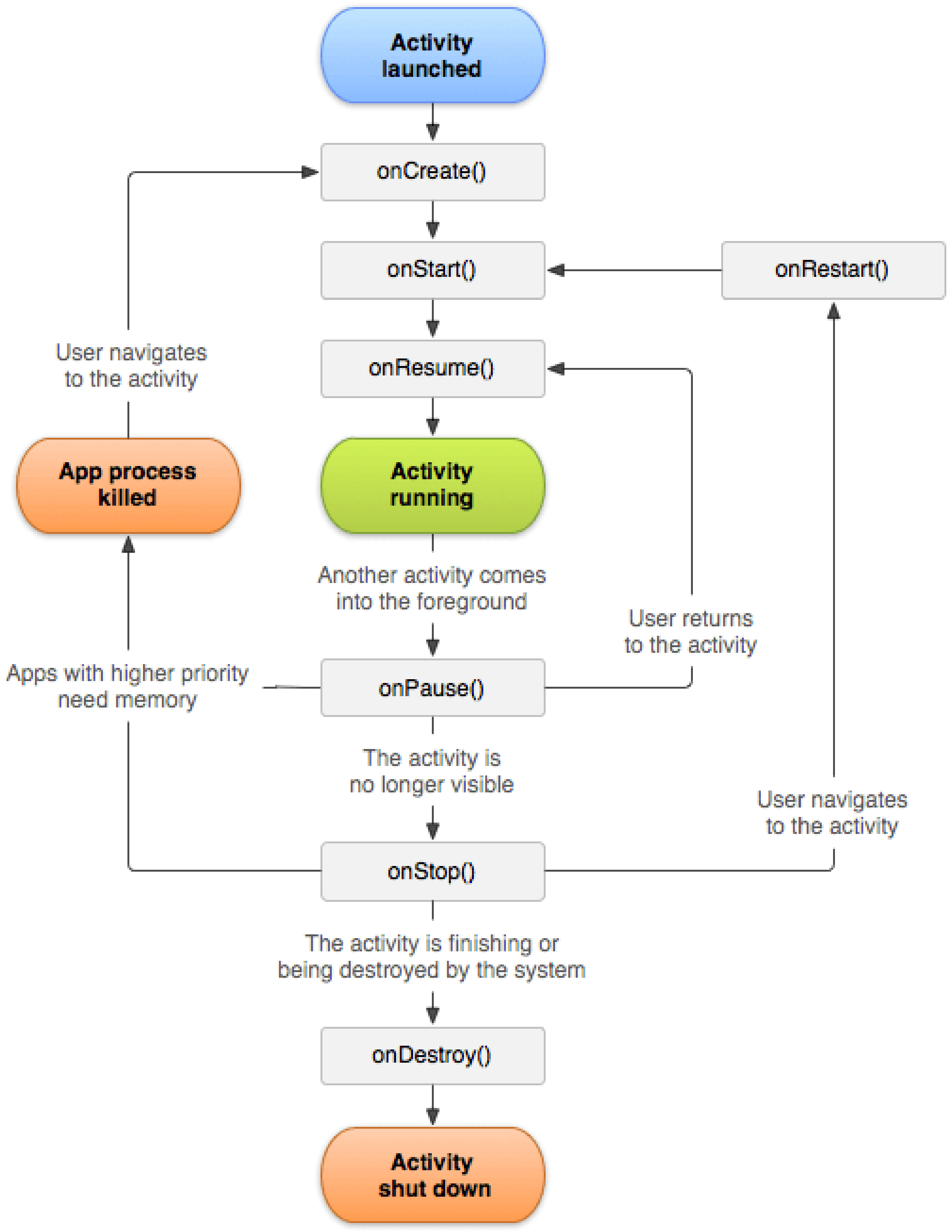
By the help of activity, you can place all your UI components or widgets in a single screen.

The 7 lifecycle method of Activity describes how activity will behave at different states.

Android Activity Lifecycle methods

Let's see the 7 lifecycle methods of android activity.

Method	Description
onCreate	called when activity is first created.
onStart	called when activity is becoming visible to the user.
onResume	called when activity will start interacting with the user.
onPause	called when activity is not visible to the user.
onStop	called when activity is no longer visible to the user.
onRestart	called after your activity is stopped, prior to start.
onDestroy	called before the activity is destroyed.


Activity launched
onCreate()
onStart()
onRestart()
onResume()
User navigates to the activity
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

File: activity_main.xml

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

<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/re
s/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="example.javatpoint.com.activitylifecycle.MainActivity">


        <TextView

            android:layout_width="wrap_content"

            android:layout_height="wrap_content"

            android:text="Hello World!"

            app:layout_constraintBottom_toBottomOf="parent"

            app:layout_constraintLeft_toLeftOf="parent"

            app:layout_constraintRight_toRightOf="parent"

            app:layout_constraintTop_toTopOf="parent" />


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

Android Activity Lifecycle Example

It provides the details about the invocation of life cycle methods of activity. In this example, we are displaying the content on the logcat.

File: MainActivity.java

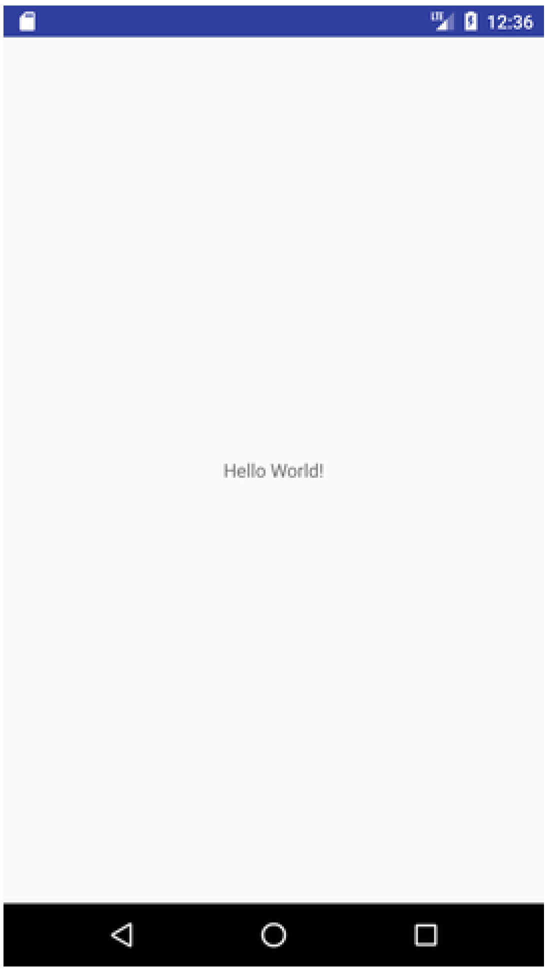
```
    package example.javatpoint.com.activitylifecycle;


    import android.app.Activity;

    import android.os.Bundle;

    import android.util.Log;


    public class MainActivity extends Activity {


        @Override
```

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Log.d("lifecycle","onCreate invoked");
}
@Override
protected void onStart() {
    super.onStart();
    Log.d("lifecycle","onStart invoked");
}
@Override
protected void onResume() {
    super.onResume();
    Log.d("lifecycle","onResume invoked");
}
@Override
protected void onPause() {
    super.onPause();
    Log.d("lifecycle","onPause invoked");
}
@Override
protected void onStop() {
    super.onStop();
    Log.d("lifecycle","onStop invoked");
}
@Override
```

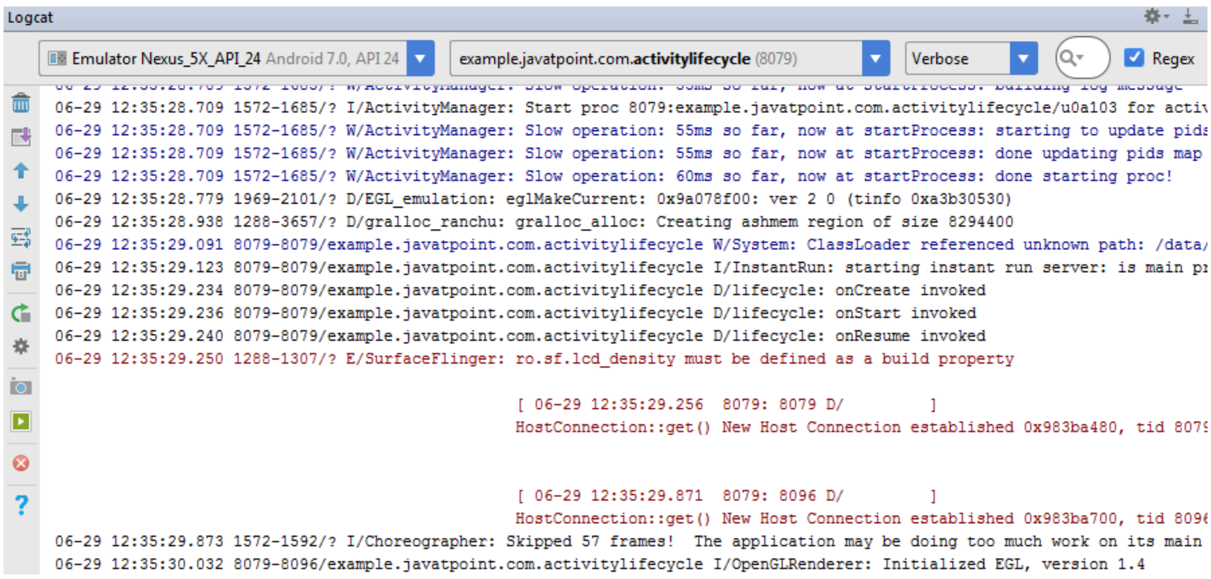
```
    protected void onRestart() {
        super.onRestart();
        Log.d("lifecycle","onRestart invoked");
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.d("lifecycle","onDestroy invoked");
    }
}
```

Output:

You will not see any output on the emulator or device. You need to open logcat.

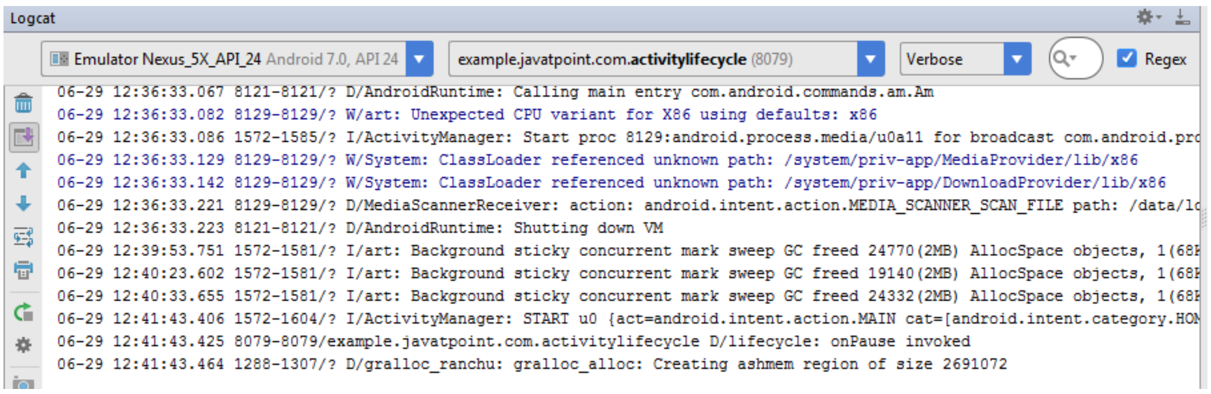


Now see on the logcat: onCreate, onStart and onResume methods are invoked.


Logcat
Emulator Nexus_5X_API_24 Android 7.0, API 24
example.javatpoint.com.activitylifecycle (8079)
Verbose
Regex

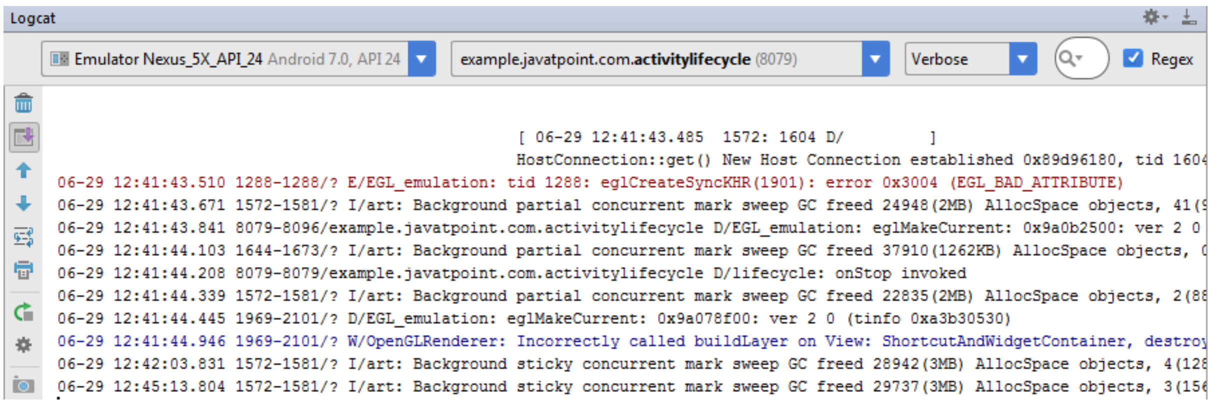
```
06-29 12:35:28.709 1572-1685/? W/ActivityManager: Slow operation: 53ms so far, now at startProcess: building log message
06-29 12:35:28.709 1572-1685/? I/ActivityManager: Start proc 8079:example.javatpoint.com.activitylifecycle/u0a103 for activ
06-29 12:35:28.709 1572-1685/? W/ActivityManager: Slow operation: 55ms so far, now at startProcess: starting to update pids
06-29 12:35:28.709 1572-1685/? W/ActivityManager: Slow operation: 55ms so far, now at startProcess: done updating pids map
06-29 12:35:28.709 1572-1685/? W/ActivityManager: Slow operation: 60ms so far, now at startProcess: done starting proc!
06-29 12:35:28.779 1969-2101/? D/EGL_emulation: eglMakeCurrent: 0x9a078f00: ver 2 0 (tinfo 0xa3b30530)
06-29 12:35:28.938 1288-3657/? D/gralloc_ranchu: gralloc_alloc: Creating ashmem region of size 8294400
06-29 12:35:29.091 8079-8079/example.javatpoint.com.activitylifecycle W/System: ClassLoader referenced unknown path: /data/
06-29 12:35:29.123 8079-8079/example.javatpoint.com.activitylifecycle I/InstantRun: starting instant run server: is main pr
06-29 12:35:29.234 8079-8079/example.javatpoint.com.activitylifecycle D/lifecycle: onCreate invoked
06-29 12:35:29.236 8079-8079/example.javatpoint.com.activitylifecycle D/lifecycle: onStart invoked
06-29 12:35:29.240 8079-8079/example.javatpoint.com.activitylifecycle D/lifecycle: onResume invoked
06-29 12:35:29.250 1288-1307/? E/SurfaceFlinger: ro.sf.lcd_density must be defined as a build property

                                                [ 06-29 12:35:29.256  8079: 8079 D/         ]
                                                HostConnection::get() New Host Connection established 0x983ba480, tid 8079


                                                [ 06-29 12:35:29.871  8079: 8096 D/         ]
                                                HostConnection::get() New Host Connection established 0x983ba700, tid 8096
06-29 12:35:29.873 1572-1592/? I/Choreographer: Skipped 57 frames!  The application may be doing too much work on its main
06-29 12:35:30.032 8079-8096/example.javatpoint.com.activitylifecycle I/OpenGLRenderer: Initialized EGL, version 1.4
```

Now click on the HOME Button. You will see onPause method is invoked.


Logcat
Emulator Nexus_5X_API_24 Android 7.0, API 24
example.javatpoint.com.activitylifecycle (8079)
Verbose
Regex

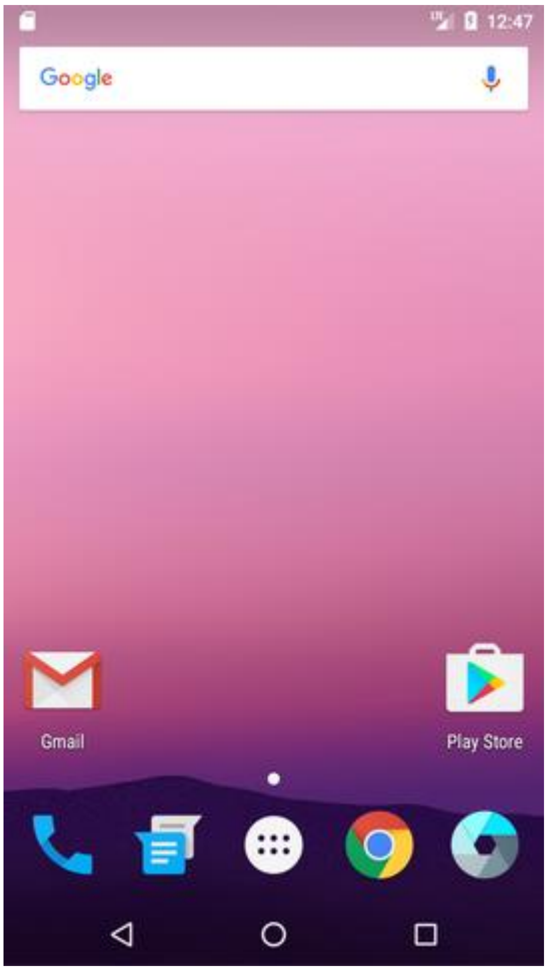
```
06-29 12:36:33.067 8121-8121/? D/AndroidRuntime: Calling main entry com.android.commands.am.Am
06-29 12:36:33.082 8129-8129/? W/art: Unexpected CPU variant for X86 using defaults: x86
06-29 12:36:33.086 1572-1585/? I/ActivityManager: Start proc 8129:android.process.media/u0a11 for broadcast com.android.prc
06-29 12:36:33.129 8129-8129/? W/System: ClassLoader referenced unknown path: /system/priv-app/MediaProvider/lib/x86
06-29 12:36:33.142 8129-8129/? W/System: ClassLoader referenced unknown path: /system/priv-app/DownloadProvider/lib/x86
06-29 12:36:33.221 8129-8129/? D/MediaScannerReceiver: action: android.intent.action.MEDIA_SCANNER_SCAN_FILE path: /data/lc
06-29 12:36:33.223 8121-8121/? D/AndroidRuntime: Shutting down VM
06-29 12:39:53.751 1572-1581/? I/art: Background sticky concurrent mark sweep GC freed 24770(2MB) AllocSpace objects, 1(68K
06-29 12:40:23.602 1572-1581/? I/art: Background sticky concurrent mark sweep GC freed 19140(2MB) AllocSpace objects, 1(68K
06-29 12:40:33.655 1572-1581/? I/art: Background sticky concurrent mark sweep GC freed 24332(2MB) AllocSpace objects, 1(68K
06-29 12:41:43.406 1572-1604/? I/ActivityManager: START u0 {act=android.intent.action.MAIN cat=[android.intent.category.HOM
06-29 12:41:43.425 8079-8079/example.javatpoint.com.activitylifecycle D/lifecycle: onPause invoked
06-29 12:41:43.464 1288-1307/? D/gralloc_ranchu: gralloc_alloc: Creating ashmem region of size 2691072
```

After a while, you will see onStop method is invoked.

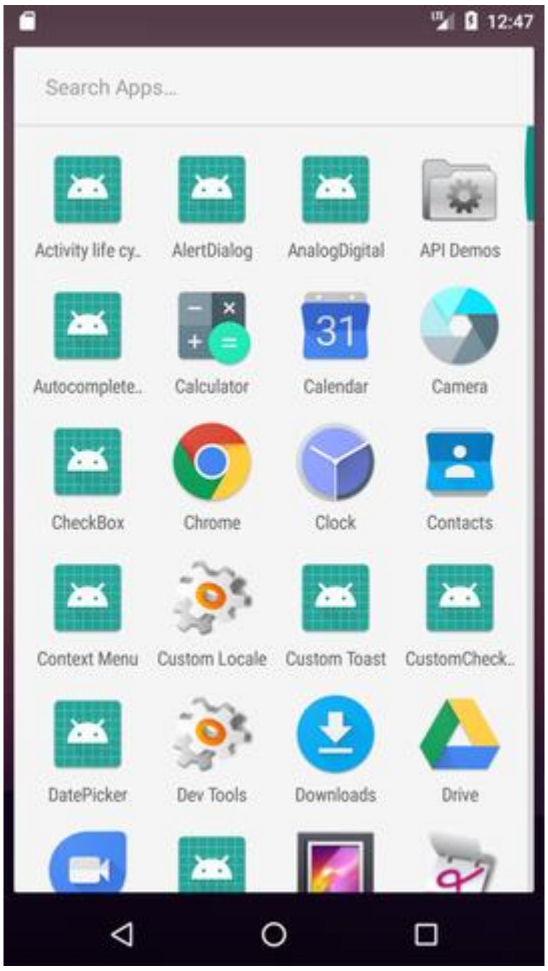

Logcat
Emulator Nexus_5X_API_24 Android 7.0, API 24
example.javatpoint.com.activitylifecycle (8079)
Verbose
Regex

```

                                                [ 06-29 12:41:43.485  1572: 1604 D/         ]
                                                HostConnection::get() New Host Connection established 0x89d96180, tid 1604
06-29 12:41:43.510 1288-1288/? E/EGL_emulation: tid 1288: eglCreateSyncKHR(1901): error 0x3004 (EGL_BAD_ATTRIBUTE)
06-29 12:41:43.671 1572-1581/? I/art: Background partial concurrent mark sweep GC freed 24948(2MB) AllocSpace objects, 41(9
06-29 12:41:43.841 8079-8096/example.javatpoint.com.activitylifecycle D/EGL_emulation: eglMakeCurrent: 0x9a0b2500: ver 2 0
06-29 12:41:44.103 1644-1673/? I/art: Background partial concurrent mark sweep GC freed 37910(1262KB) AllocSpace objects, 0
06-29 12:41:44.208 8079-8079/example.javatpoint.com.activitylifecycle D/lifecycle: onStop invoked
06-29 12:41:44.339 1572-1581/? I/art: Background partial concurrent mark sweep GC freed 22835(2MB) AllocSpace objects, 2(88
06-29 12:41:44.445 1969-2101/? D/EGL_emulation: eglMakeCurrent: 0x9a078f00: ver 2 0 (tinfo 0xa3b30530)
06-29 12:41:44.946 1969-2101/? W/OpenGLRenderer: Incorrectly called buildLayer on View: ShortcutAndWidgetContainer, destroy
06-29 12:42:03.831 1572-1581/? I/art: Background sticky concurrent mark sweep GC freed 28942(3MB) AllocSpace objects, 4(128
06-29 12:45:13.804 1572-1581/? I/art: Background sticky concurrent mark sweep GC freed 29737(3MB) AllocSpace objects, 3(156
```

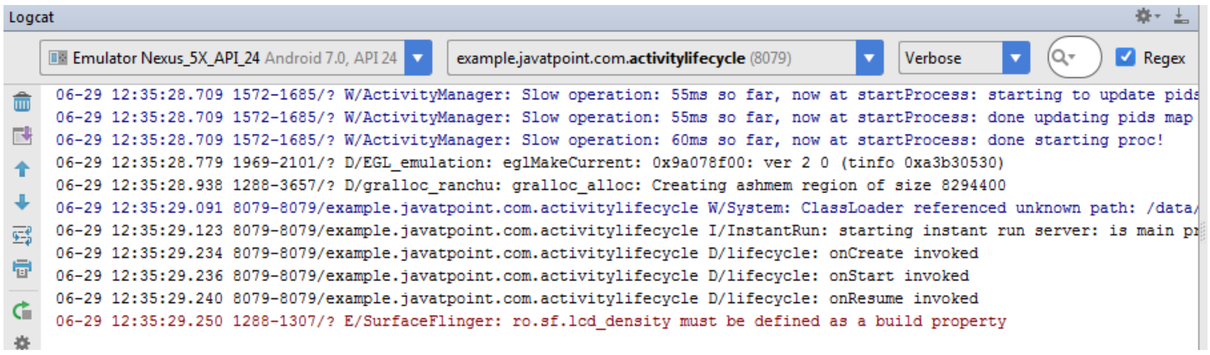
Now see on the emulator. It is on the home. Now click on the center button to launch the app again.



Now click on the lifecycleactivity icon.

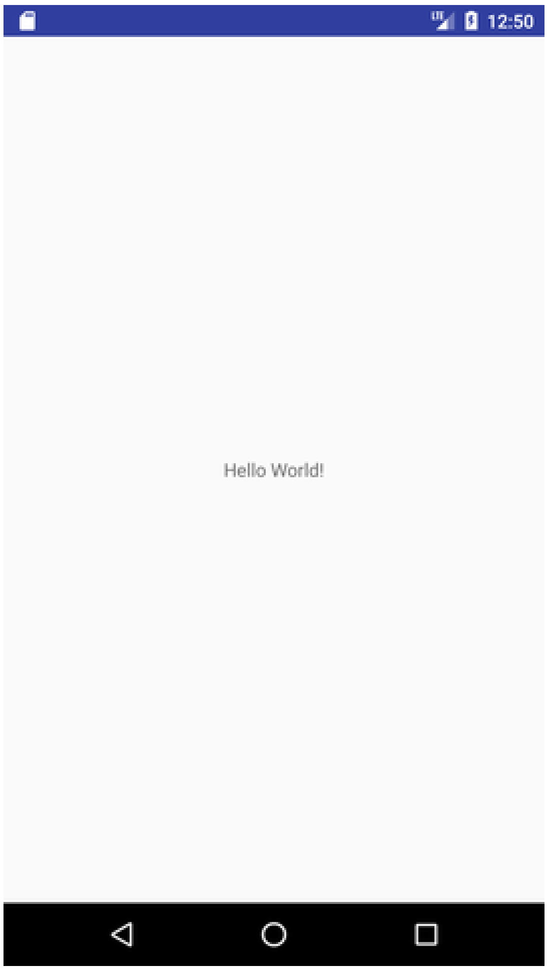


Now see on the logcat: onRestart, onStart and onResume methods are invoked.

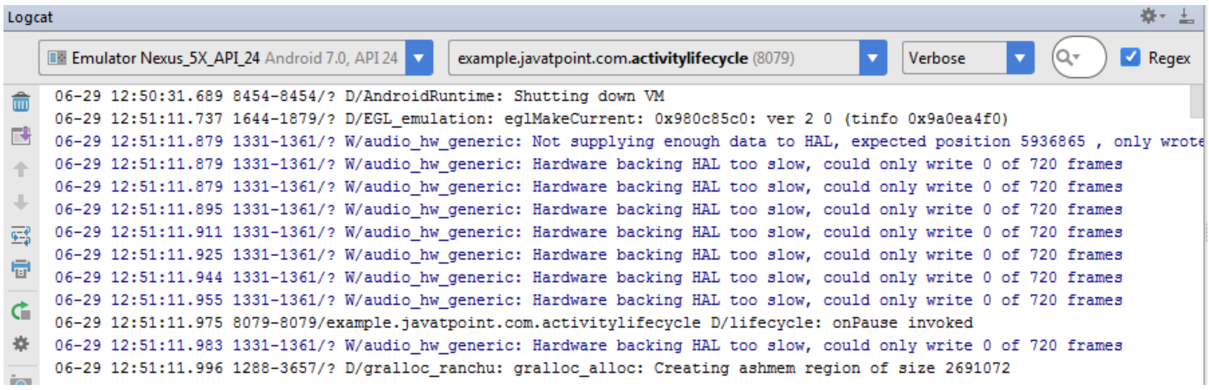

Logcat
Emulator Nexus_5X_API_24 Android 7.0, API 24
example.javatpoint.com.activitylifecycle (8079)
Verbose
Regex

```
06-29 12:35:28.709 1572-1685/? W/ActivityManager: Slow operation: 55ms so far, now at startProcess: starting to update pids
06-29 12:35:28.709 1572-1685/? W/ActivityManager: Slow operation: 55ms so far, now at startProcess: done updating pids map
06-29 12:35:28.709 1572-1685/? W/ActivityManager: Slow operation: 60ms so far, now at startProcess: done starting proc!
06-29 12:35:28.779 1969-2101/? D/EGL_emulation: eglMakeCurrent: 0x9a078f00: ver 2 0 (tinfo 0xa3b30530)
06-29 12:35:28.938 1288-3657/? D/gralloc_ranchu: gralloc_alloc: Creating ashmem region of size 8294400
06-29 12:35:29.091 8079-8079/example.javatpoint.com.activitylifecycle W/System: ClassLoader referenced unknown path: /data/
06-29 12:35:29.123 8079-8079/example.javatpoint.com.activitylifecycle I/InstantRun: starting instant run server: is main pr
06-29 12:35:29.234 8079-8079/example.javatpoint.com.activitylifecycle D/lifecycle: onCreate invoked
06-29 12:35:29.236 8079-8079/example.javatpoint.com.activitylifecycle D/lifecycle: onStart invoked
06-29 12:35:29.240 8079-8079/example.javatpoint.com.activitylifecycle D/lifecycle: onResume invoked
06-29 12:35:29.250 1288-1307/? E/SurfaceFlinger: ro.sf.lcd_density must be defined as a build property
```

If you see the emulator, application is started again.



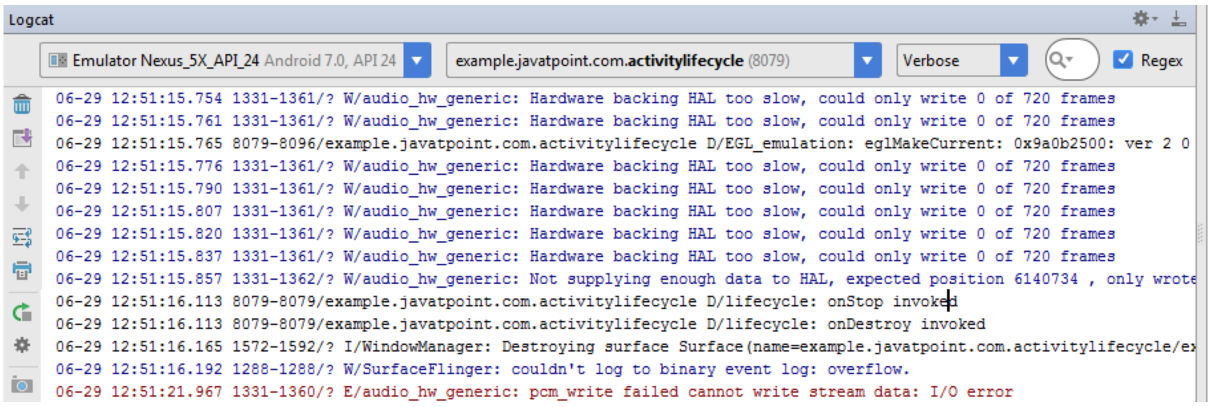
Now click on the back button. Now you will see onPause methods is invoked.



Emulator Nexus_5X_API_24 Android 7.0, API 24 | example.javatpoint.com.activitylifecycle (8079) | Verbose

```
06-29 12:50:31.689 8454-8454/? D/AndroidRuntime: Shutting down VM
06-29 12:51:11.737 1644-1879/? D/EGL_emulation: eglMakeCurrent: 0x980c85c0: ver 2 0 (tinfo 0x9a0ea4f0)
06-29 12:51:11.879 1331-1361/? W/audio_hw_generic: Not supplying enough data to HAL, expected position 5936865 , only wrote
06-29 12:51:11.879 1331-1361/? W/audio_hw_generic: Hardware backing HAL too slow, could only write 0 of 720 frames
06-29 12:51:11.879 1331-1361/? W/audio_hw_generic: Hardware backing HAL too slow, could only write 0 of 720 frames
06-29 12:51:11.895 1331-1361/? W/audio_hw_generic: Hardware backing HAL too slow, could only write 0 of 720 frames
06-29 12:51:11.911 1331-1361/? W/audio_hw_generic: Hardware backing HAL too slow, could only write 0 of 720 frames
06-29 12:51:11.925 1331-1361/? W/audio_hw_generic: Hardware backing HAL too slow, could only write 0 of 720 frames
06-29 12:51:11.944 1331-1361/? W/audio_hw_generic: Hardware backing HAL too slow, could only write 0 of 720 frames
06-29 12:51:11.955 1331-1361/? W/audio_hw_generic: Hardware backing HAL too slow, could only write 0 of 720 frames
06-29 12:51:11.975 8079-8079/example.javatpoint.com.activitylifecycle D/lifecycle: onPause invoked
06-29 12:51:11.983 1331-1361/? W/audio_hw_generic: Hardware backing HAL too slow, could only write 0 of 720 frames
06-29 12:51:11.996 1288-3657/? D/gralloc_ranchu: gralloc_alloc: Creating ashmem region of size 2691072
```

After a while, you will see onStop and onDestroy methods are invoked.



Logcat
Emulator Nexus_5X_API_24 Android 7.0, API 24
example.javatpoint.com.activitylifecycle (8079)
Verbose
Regex

```
06-29 12:51:15.754 1331-1361/? W/audio_hw_generic: Hardware backing HAL too slow, could only write 0 of 720 frames
06-29 12:51:15.761 1331-1361/? W/audio_hw_generic: Hardware backing HAL too slow, could only write 0 of 720 frames
06-29 12:51:15.765 8079-8096/example.javatpoint.com.activitylifecycle D/EGL_emulation: eglMakeCurrent: 0x9a0b2500: ver 2 0
06-29 12:51:15.776 1331-1361/? W/audio_hw_generic: Hardware backing HAL too slow, could only write 0 of 720 frames
06-29 12:51:15.790 1331-1361/? W/audio_hw_generic: Hardware backing HAL too slow, could only write 0 of 720 frames
06-29 12:51:15.807 1331-1361/? W/audio_hw_generic: Hardware backing HAL too slow, could only write 0 of 720 frames
06-29 12:51:15.820 1331-1361/? W/audio_hw_generic: Hardware backing HAL too slow, could only write 0 of 720 frames
06-29 12:51:15.837 1331-1361/? W/audio_hw_generic: Hardware backing HAL too slow, could only write 0 of 720 frames
06-29 12:51:15.857 1331-1362/? W/audio_hw_generic: Not supplying enough data to HAL, expected position 6140734 , only wrote
06-29 12:51:16.113 8079-8079/example.javatpoint.com.activitylifecycle D/lifecycle: onStop invoked
06-29 12:51:16.113 8079-8079/example.javatpoint.com.activitylifecycle D/lifecycle: onDestroy invoked
06-29 12:51:16.165 1572-1592/? I/WindowManager: Destroying surface Surface(name=example.javatpoint.com.activitylifecycle/ex
06-29 12:51:16.192 1288-1288/? W/SurfaceFlinger: couldn't log to binary event log: overflow.
06-29 12:51:21.967 1331-1360/? E/audio_hw_generic: pcm_write failed cannot write stream data: I/O error
```