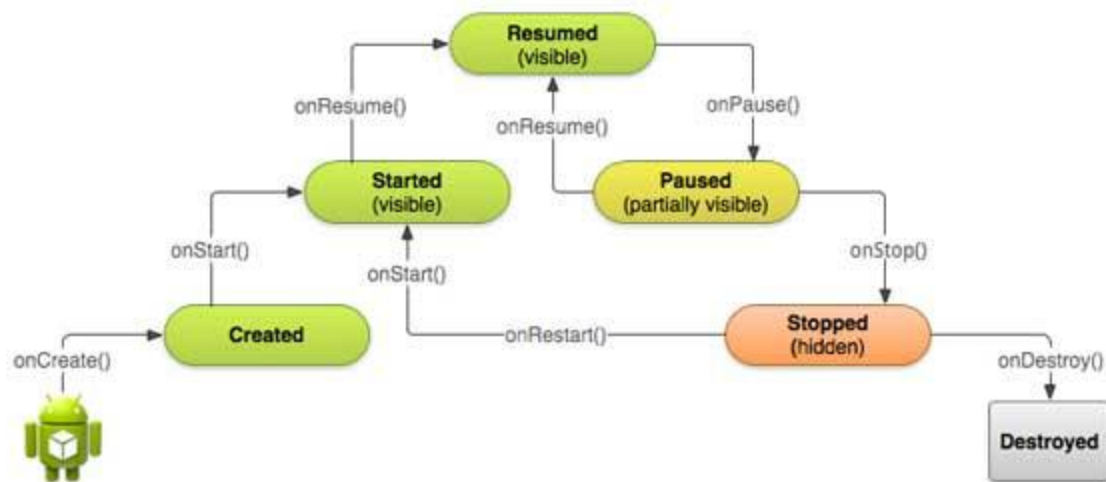


Activities

An activity represents a single screen with a user interface. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.

If you have worked with C, C++ or Java programming language then you must have seen that your program starts from **main()** function. Very similar way, Android system initiates its program with in an **Activity** starting with a call on **onCreate()** callback method. There is a sequence of callback methods that start up an activity and a sequence of callback methods that tear down an activity as shown in the below Activity lifecycle diagram: (*image courtesy : android.com*)



The Activity class defines the following callbacks i.e. events. You don't need to implement all the callbacks methods. However, it's important that you understand each one and implement those that ensure your app behaves the way users expect.

Callback	Description
onCreate()	This is the first callback and called when the activity is first created.
onStart()	This callback is called when the activity becomes visible to the user.
onResume()	This is called when the user starts interacting with the application.

onPause()	The paused activity does not receive user input and cannot execute any code and called when the current activity is being paused and the previous activity is being resumed.
onStop()	This callback is called when the activity is no longer visible.
onDestroy()	This callback is called before the activity is destroyed by the system.
onRestart()	This callback is called when the activity restarts after stopping it.

Example

This example will take you through simple steps to show Android application activity life cycle. Follow the following steps to modify the Android application we created in *Hello World Example* chapter:

Step	Description
1	You will use Eclipse IDE to create an Android application and name it as <i>HelloWorld</i> under a package <i>com.example.helloworld</i> as explained in the <i>Hello World Example</i> chapter.
2	Modify main activity file <i>MainActivity.java</i> as explained below. Keep rest of the files unchanged.
3	Run the application to launch Android emulator and verify the result of the changes done in the application.

Following is the content of the modified main activity file **src/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 com.example.helloworld;

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

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 activity class loads all the UI component using the XML file available in *res/layout* folder of the project. Following statement loads UI components from *res/layout/activity_main.xml* file:

```

setContentView(R.layout.activity_main);

```


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=".MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>

```

If either the MAIN action or LAUNCHER category are not declared for one of your activities, then your app icon will not appear in the Home screen's list of apps.

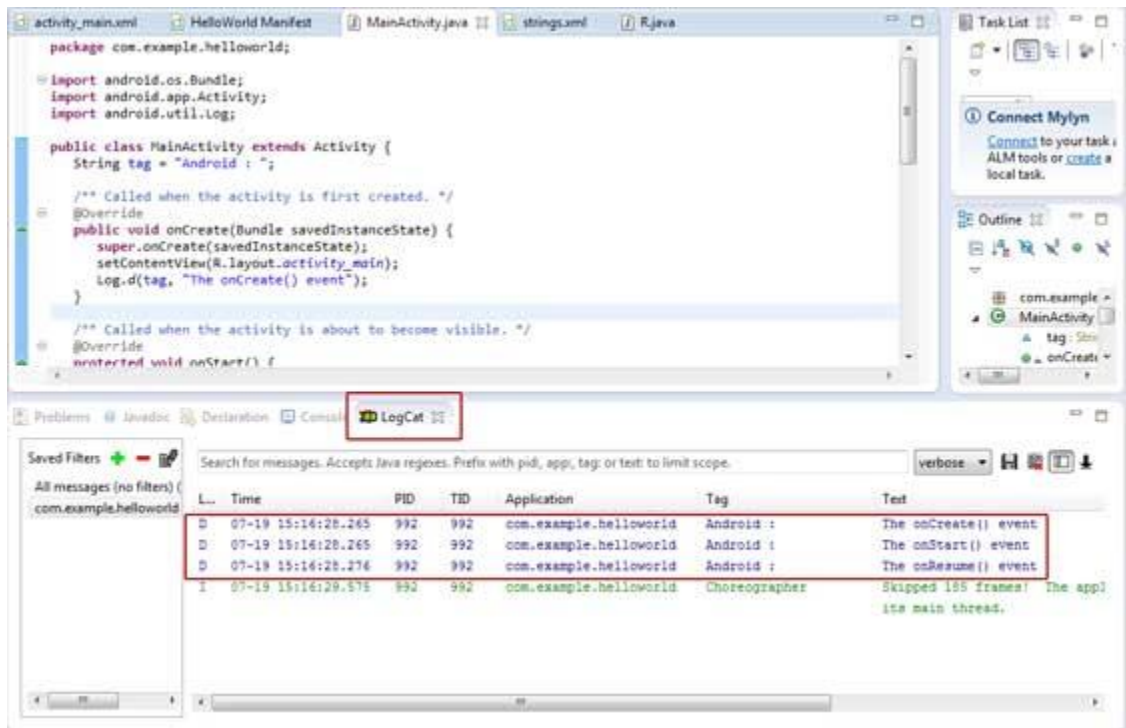
Let's try to run our modified **Hello World!** application we just modified. I assume you had created your **AVD** while doing environment setup. To run the app from Eclipse, open one of your project's activity files and click Run  icon from the toolbar. Eclipse installs the app on your AVD and starts it and if everything is fine with your setup and application, it will display Emulator window and you should see following log messages in **LogCat** window in Eclipse IDE:


```

07-19 15:00:43.405: D/Android : (866): The onCreate() event


```

```
07-19 15:00:43.405: D/Android : (866): The onStart() event
07-19 15:00:43.415: D/Android : (866): The onResume() event
```




Let us try to click Red button  on the Android emulator and it will generate following events messages in **LogCat** window in Eclipse IDE:

```
07-19 15:01:10.995: D/Android : (866): The onPause() event
07-19 15:01:12.705: D/Android : (866): The onStop() event
```

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

```
07-19 15:01:13.995: D/Android : (866): The onStart() event
07-19 15:01:14.705: D/Android : (866): The onResume() event
```

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

```
07-19 15:33:15.687: D/Android : (992): The onPause() event
07-19 15:33:15.525: D/Android : (992): The onStop() event
07-19 15:33:15.525: D/Android : (992): The onDestroy() event
```