# Lesson 4 – Application Lifecycle

Create a new project in Eclipse, with the following settings:

Project name:  **Lesson4_LifeCycle**
Create new project in workspace should be selected.
Build Target: **Android 2.3.3**
Application Name: **Lesson4_LifeCycle**
Package name: **lastname.firstname. Lesson4_LifeCycle** (substitute lastname.firstname with your last name and first name – no spaces, apostrophes or dashes.)
Create Activity should be checked, and **Lesson4LifeCycle** should be the text for it.
Min SDK Version: **10**

Click **Finish**.

Open the main.xml file, found in the **res/layout** folder.

Mody the <TextView> element's XML to look like this:

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

This will allow us to output to the screen from our code, as we did in previous lessons.  Save and close the main.xml file.

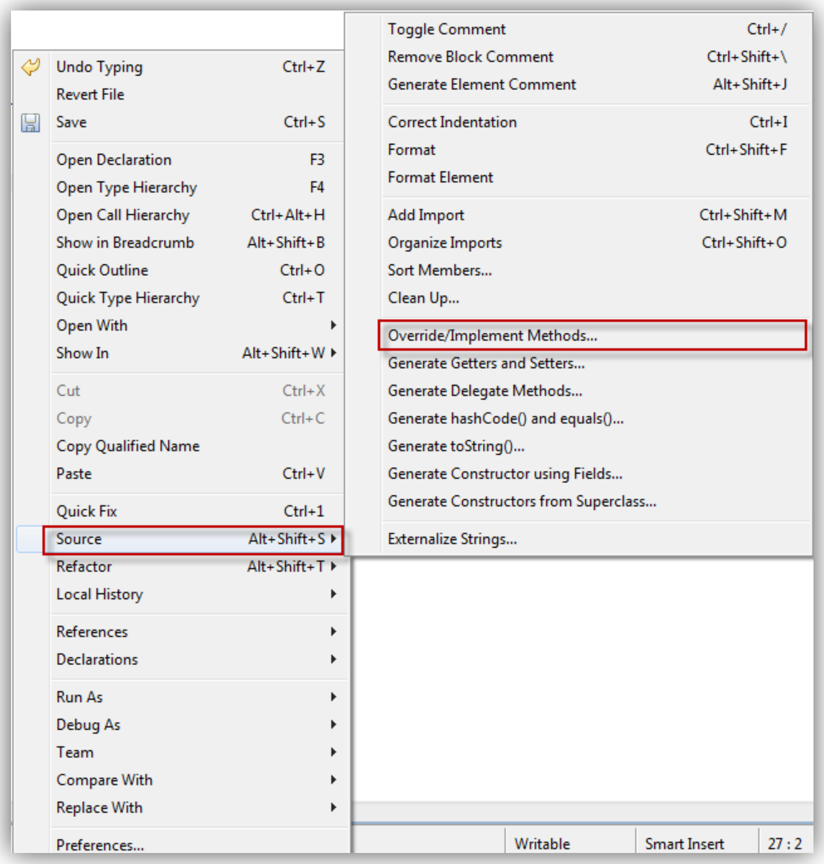
Open Lesson4LifeCycle.java file, in your SRC/package/ folder.  We'll begin with string comparison.

Add the code highlighted in yellow to the file.

```
public class Lesson4LifeCycle extends Activity {
        public static final String DATE_FORMAT = "MM-dd-yyyy HH:mm:ss";
        String sStatus = "";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }



    public static String Now() {
        Calendar cal = Calendar.getInstance();
        SimpleDateFormat sdf = new SimpleDateFormat(DATE_FORMAT);
        return sdf.format(cal.getTime());
    }

    void setStatus(String sEvent) {
        if (sEvent.equalsIgnoreCase("onCreate")) {
                sStatus += "========================\n";
        }

        sStatus += "In " + sEvent + "() at: " + Now() + "\n";
        final TextView tOut = (TextView) findViewById(R.id.txtOut);
        tOut.setText(sStatus);
    }

}
```

The above code creates a new public routine called Now, which will output the current date and time based on the format that we defined using the global "DATE_FORMAT" variable, declared below the class declaration. The setStatus routine will be used to output information to the screen. This way we can use the same routine for all of the events within our application, instead of re-writing the same thing over and over again.

Next, right-click in a blank part of the screen, and select **Source**, then choose **Override / Implement Methods…**



Select the following methods from the Activity node in the dialog box.

- onDestroy
- onPause
- onRestart
- onResume
- onStart
- onStop

Modify each of the routines that were created to look like the code on the next page. You're adding the items that are highlighted in yellow. Note the onCreate and onStop events. In onCreate, you're adding code to retrieve a value from SharedPreferences. SharedPreferences works much like the Windows registry in the sense that you're storing data needed by your application, and you are storing a key / value pair. In this case, the key is called "screen_msg", and contains all of the text that we are displaying on the screen. By default, it will load an empty string, if the value doesn't exist within shared preferences. This default is set in the oSettings.getString("screen_msg", "") line. The "" is the default value to load. In onStop, SharedPreferences are used again, but this time, we save or "putString" into the "screen_msg" variable. The string, is sStatus – our global variable which controls all of the content that we see in our application's screen.

```
@Override
public void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	setContentView(R.layout.main);
	SharedPreferences oSettings = getSharedPreferences(
			"screen_msg", 0);
	sStatus = oSettings.getString("screen_msg", "");
	setStatus("onCreate");
}


@Override
protected void onDestroy() {
	super.onDestroy();
	setStatus("onDestroy");
}

@Override
protected void onPause() {
	super.onPause();
	setStatus("onPause");
}

@Override
protected void onRestart() {
	super.onRestart();
	setStatus("onRestart");
}

@Override
protected void onResume() {
	super.onResume();
	setStatus("onResume");
}

@Override
protected void onStart() {
	super.onStart();
	setStatus("onStart");
}

@Override
protected void onStop() {
	super.onStop();
	setStatus("onStop");
	// Save Settings here!
	SharedPreferences oSettings =
	     getSharedPreferences("screen_msg", 0);
	SharedPreferences.Editor oEdit = oSettings.edit();
	oEdit.putString("screen_msg", sStatus);
	oEdit.commit();
}
```

To test the application, run it several times.  Use the emulator’s application settings control to force stop the application, to see which events are fired when you simply navigate away from your application vs. stop the application completely.  Review the video along with this lesson.