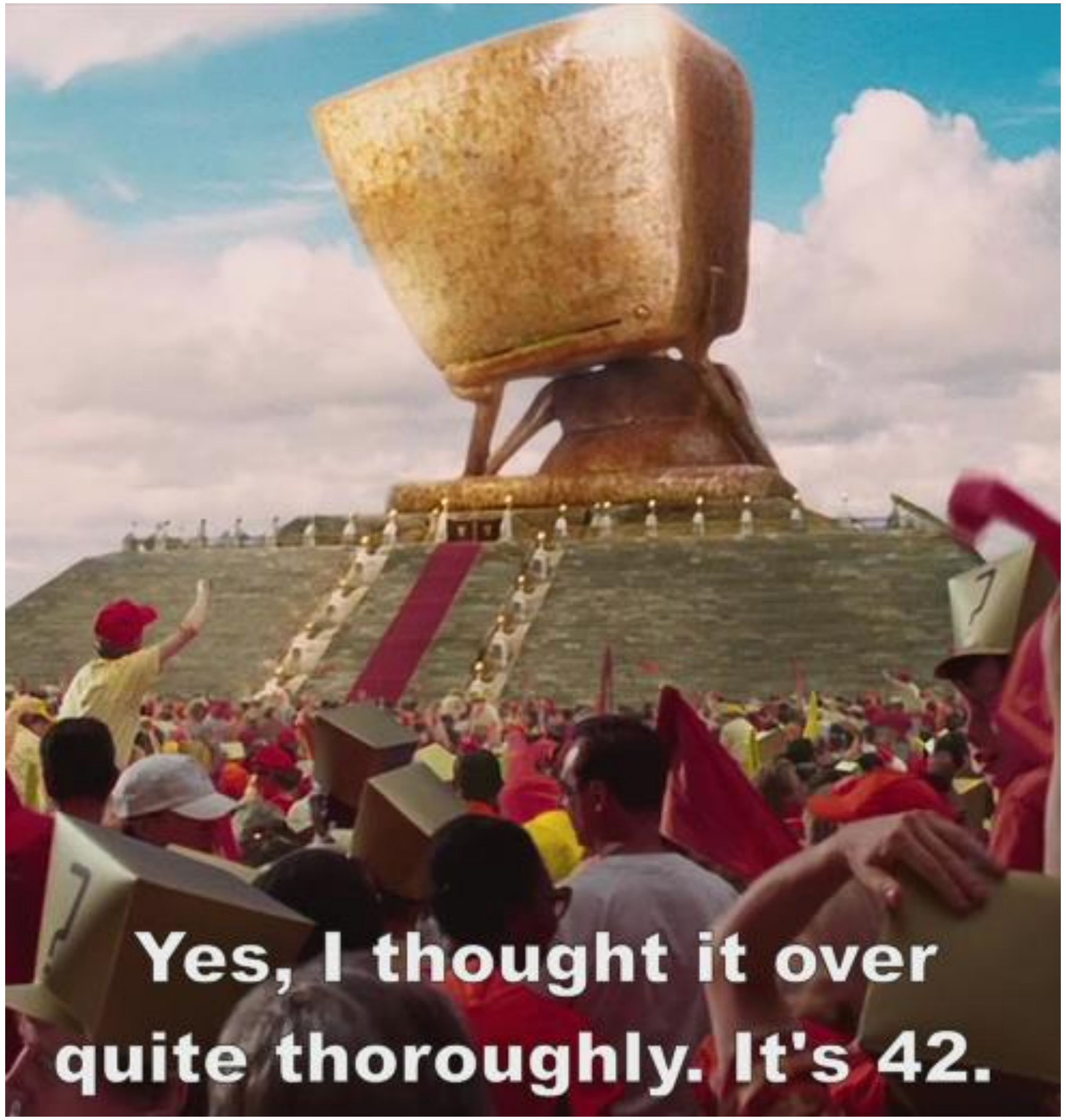
Android Course Day 1

Lucio Cossio

Luis Mazoni

Introduction

- Who are we?
- Why are we here?



Course Agenda

Day 1

- **Development environment and tools**
- **Android project structure**
    - Source, tests , resources, manifest
- **Activity**
    - Lifecycle
    - Layout interaction
    - ActionBar
- **Practice**

Day 2

- **Review Day 1**
- **Layout**
    - View Group
        - Relative, Linear
    - Views
        - TextView, Edit Text, Button
        - String resources
    - View listeners
- **Practice**

Day 3

- **Review Day 2**
- **Intents**
    - Open new activity
    - Sending data
    - Actions
- **Android Manifest**
    - Overview
    - Add activities
- **Practice**

Day 4

- **Review Day 3**
- **Fragments**
    - Lifecycle
    - Inflate Layouts
    - Fragment Manager
    - Arguments
- **Practice**

Day 5

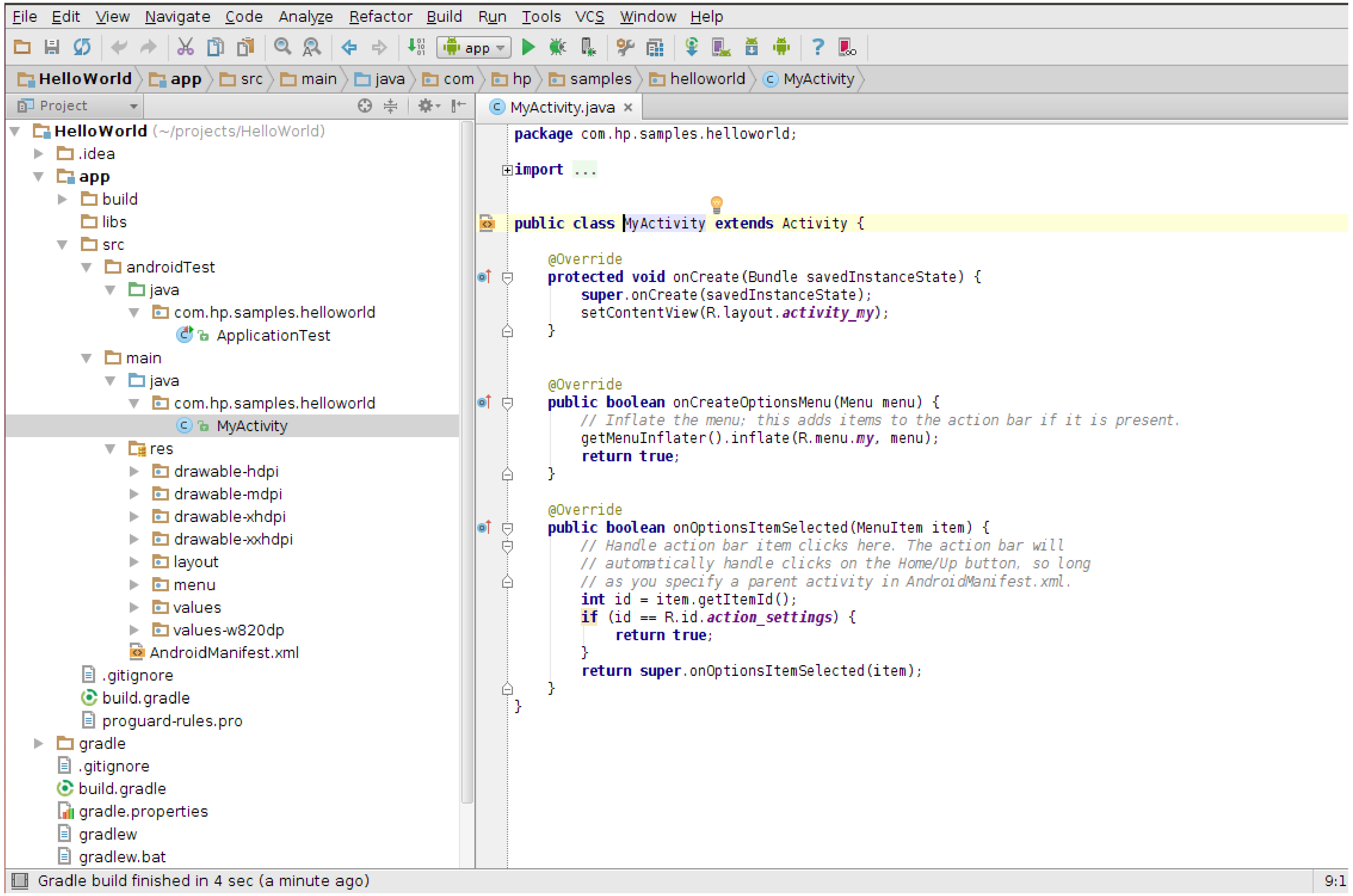
- **Review Day 4**
- **ListView**
    - Adapter
    - View Holder
- **Async Tasks**
    - UI Thread
- **Practice**

Development Environment and Tools

- Java JDK 7
- Android SDK 4.4
- Android Studio (0.8x +)
- Genymotion (uses VirtualBox)

Android project structure

Android Studio


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
app
HelloWorld › app › src › main › java › com › hp › samples › helloworld › MyActivity
Project
MyActivity.java

```
HelloWorld (~/projects/HelloWorld)
  .idea
  app
    build
    libs
    src
      androidTest
        java
          com.hp.samples.helloworld
            ApplicationTest
      main
        java
          com.hp.samples.helloworld
            MyActivity
        res
          drawable-hdpi
          drawable-mdpi
          drawable-xhdpi
          drawable-xxhdpi
          layout
          menu
          values
          values-w820dp
        AndroidManifest.xml
    .gitignore
    build.gradle
    proguard-rules.pro
  gradle
  .gitignore
  build.gradle
  gradle.properties
  gradlew
  gradlew.bat
```

```java
package com.hp.samples.helloworld;

import ...

public class MyActivity extends Activity {

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


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.my, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

Gradle build finished in 4 sec (a minute ago)   9:1

Android project structure

Gradle

- It uses **gradle** by Default
  - build.gradle –> controls building and project dependencies
  - gradle.properties -> default gradle configurations
- What is **gradle**?

  Gradle can automate the building, testing, publishing and deployment. Gradle combines the power and flexibility of ant with the dependency management and conventions of Maven into a more effective way to build.

Android project structure

Android Folders

- Folder structure:
  - Your app module
    - src
      - androidTest
        - **java <- tests here**
      - main
        - **java <- app code here**
        - **res <- android resources (layouts, strings, dimensions, drawables)**
        - **AndroidManifest.xml <- info to android system about your app (java package name, activities, services, permissions, api required,...)**

Activity

An Activity is an application component that provides a screen with which users can interact in order to do something. Each activity is given a window in which to draw its user interface.

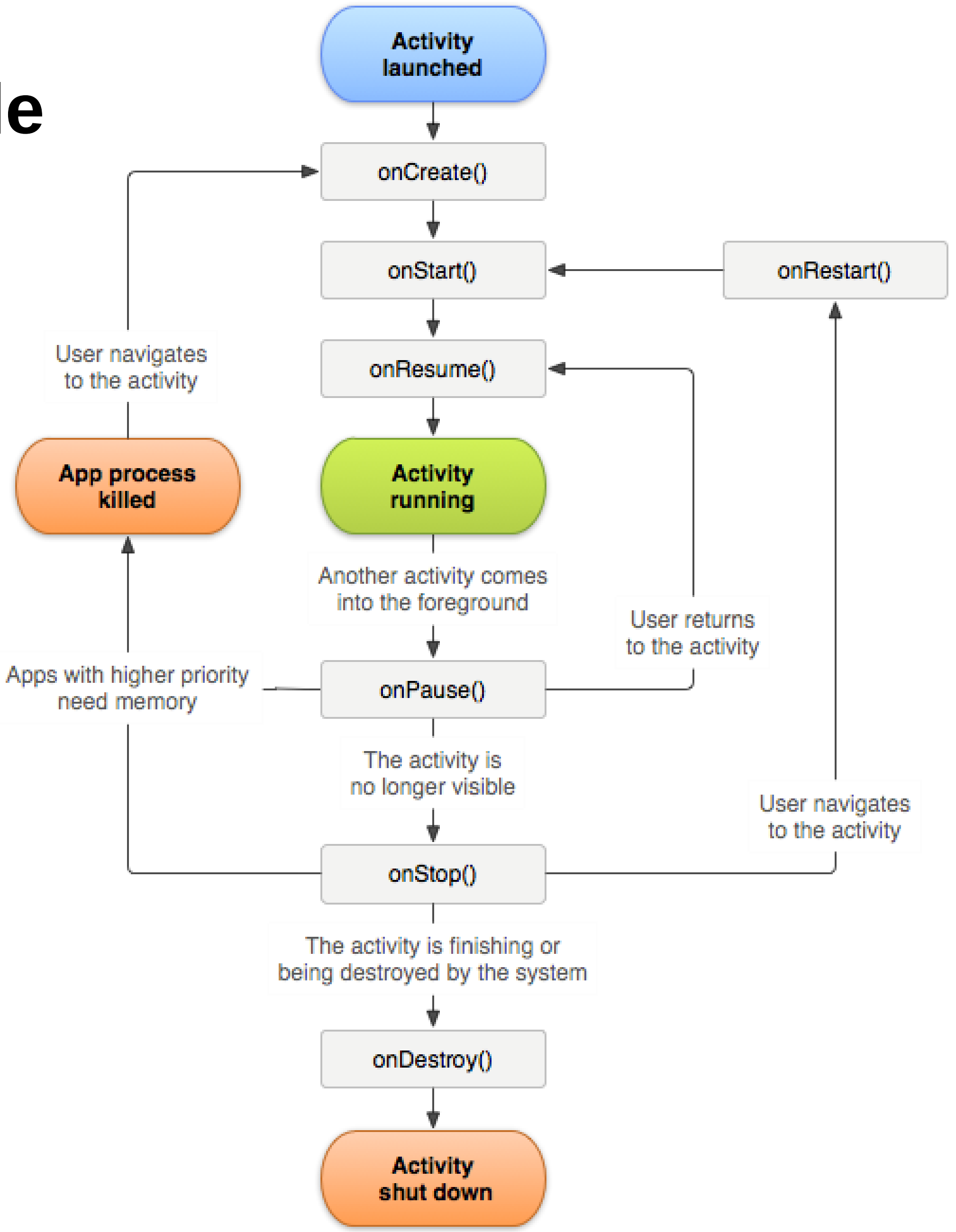
Typically, one activity in an application is specified as the "main" activity, which is presented to the user when launching the application for the first time. Each activity can then start another activity in order to perform different actions. Each time a new activity starts, the previous activity is stopped, but the system preserves the activity in a stack.

Activity - onCreate

onCreate(Bundle) is where you initialize your activity. Most importantly, here you will usually call setContentView(int) with a layout resource defining your UI, and using findViewById(int) to retrieve the widgets in that UI that you need to interact with programmatically.

setContentView() can be used with both a layout resource or a instance of a ViewGroup object

Activity Lifecycle


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
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

Activity Lifecycle

An activity has essentially four states:

- If an activity in the foreground of the screen (at the top of the stack), it is *active* or ***running***.
- If an activity has lost focus but is still visible (that is, a new non-full-sized or transparent activity has focus on top of your activity), it is ***paused***. A paused activity is completely alive (it maintains all state and member information and remains attached to the window manager), but can be killed by the system in extreme low memory situations.

Activity Lifecycle

- If an activity is completely obscured by another activity, it is ***stopped***. It still retains all state and member information, however, it is no longer visible to the user so its window is hidden and it will often be killed by the system when memory is needed elsewhere.
- If an activity is paused or stopped**, the system can drop the activity from memory** by either asking it to finish, or simply killing its process. When it is displayed again to the user, it must be completely restarted and restored to its previous state.

Activity Lifecycle

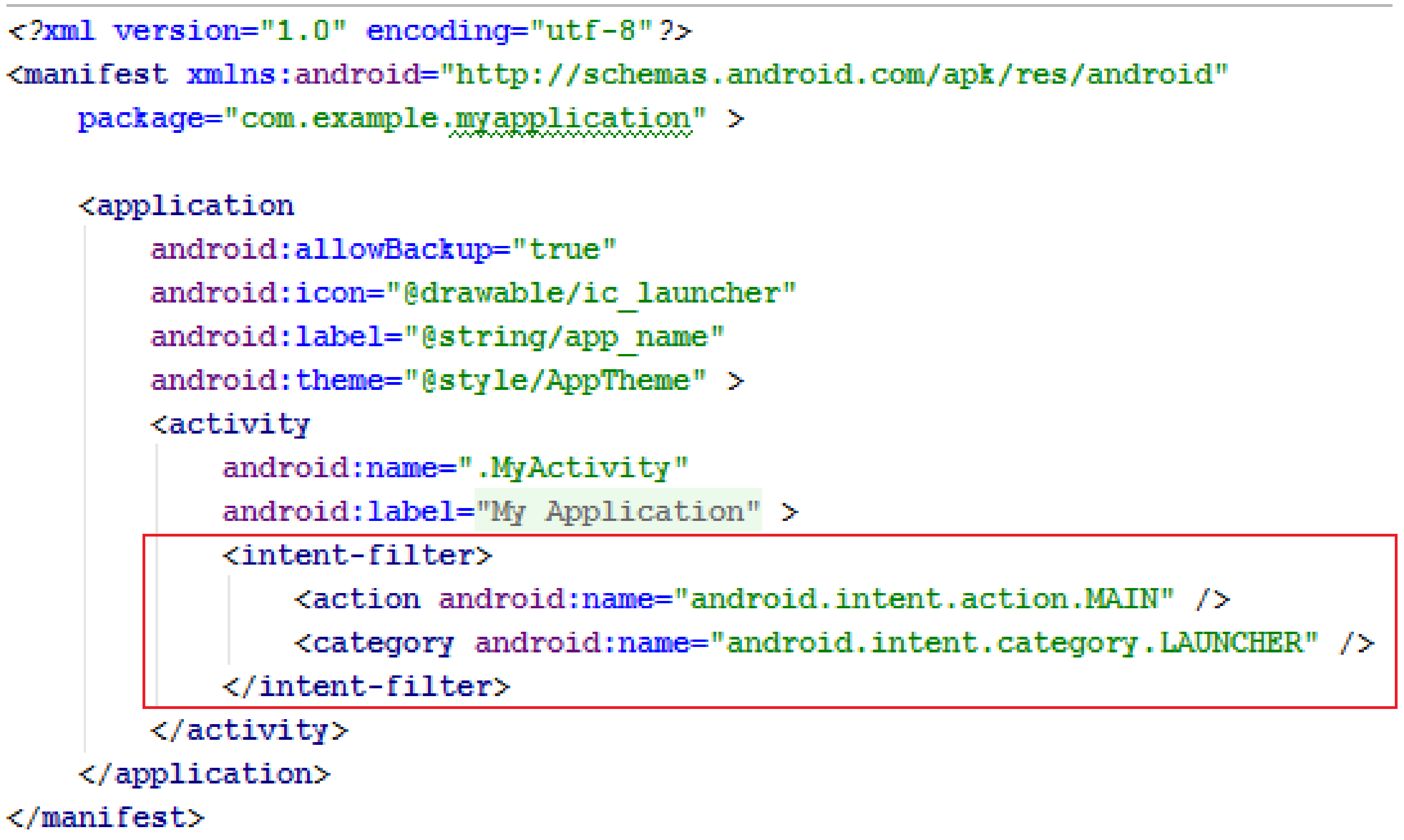
- You should use the onPause() method to write any persistent data (such as user edits) to storage. In addition, the method onSaveInstanceState(Bundle) is called before placing the activity in such a background state, allowing you to save away any dynamic instance state in your activity into the given Bundle, to be later received in onCreate(Bundle) if the activity needs to be re-created.

Configuration Changed

- Unless you specify otherwise, a configuration change (such as a change in screen orientation, language, input devices, etc) will cause your current activity to be *destroyed*, going through the normal activity lifecycle process of onPause(), onStop(), and onDestroy() as appropriate.
- If the activity had been in the foreground or visible to the user, once onDestroy() is called in that instance then a new instance of the activity will be created, with whatever savedInstanceState the previous instance had generated from onSaveInstanceState(Bundle).

Practice:
Create and run new project on genymotion

Main Activity

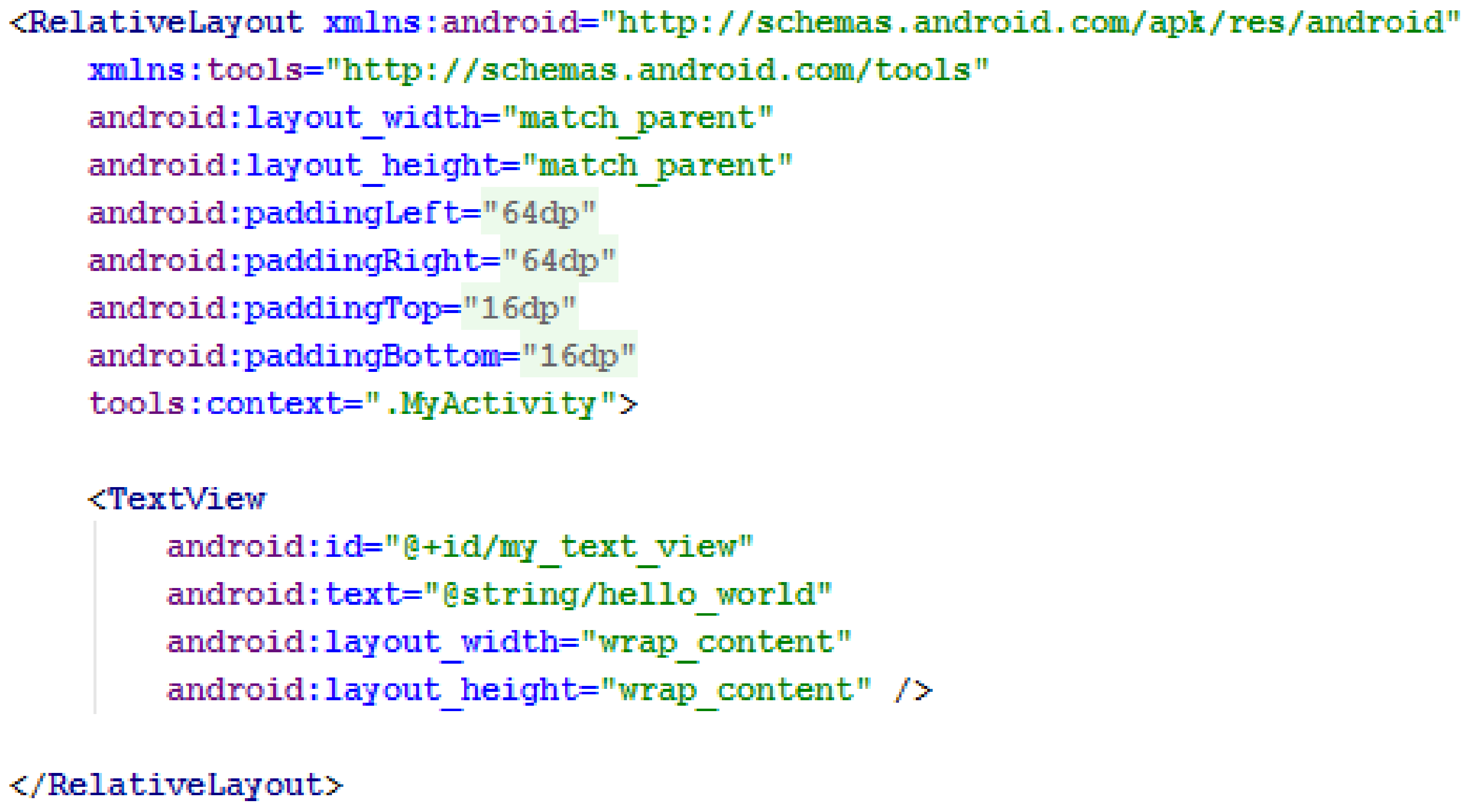


```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapplication" >

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MyActivity"
            android:label="My Application" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Layout Interaction

- Activity Layout



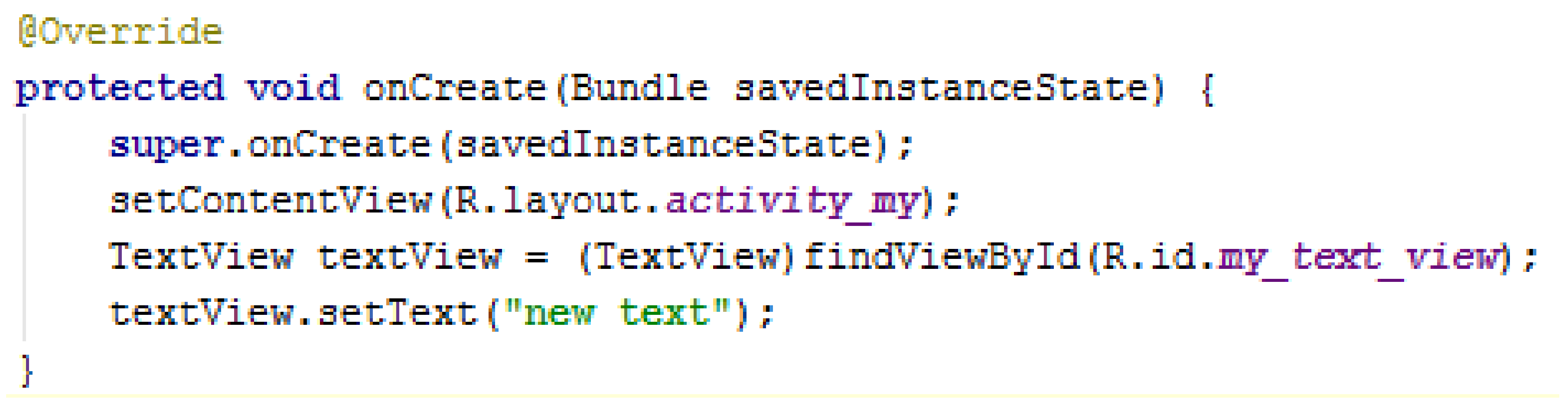
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="64dp"
    android:paddingRight="64dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp"
    tools:context=".MyActivity">

    <TextView
        android:id="@+id/my_text_view"
        android:text="@string/hello_world"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</RelativeLayout>
```

Layout Interaction

- Change Text View Text



```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_my);
    TextView textView = (TextView)findViewById(R.id.my_text_view);
    textView.setText("new text");
}
```

Action Bar

The action bar is a window feature that identifies the user location, and provides user actions and navigation modes.

Using the action bar offers your users a familiar interface across applications that the system gracefully adapts for different screen configurations.

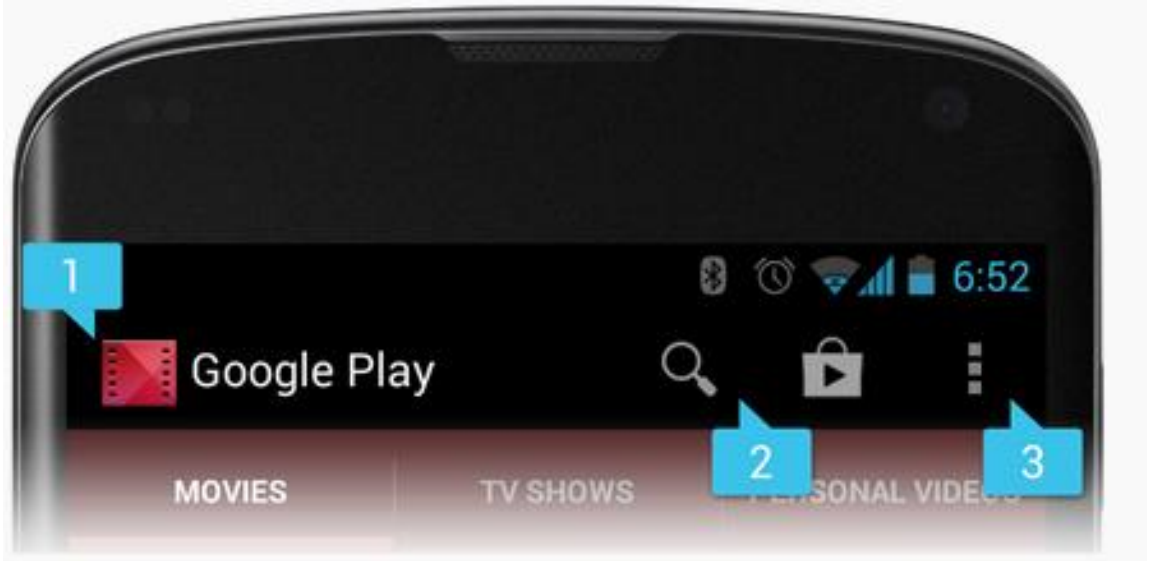


Figure 1. An action bar that includes the [1] app icon, [2] two action items, and [3] action overflow.

Action Bar – Adding Action Items

When your activity starts, the system populates the action items by calling your activity's onCreateOptionsMenu() method. Use this method to inflate a menu resource that defines all the action items. For example, here's a menu resource defining a couple of menu items:

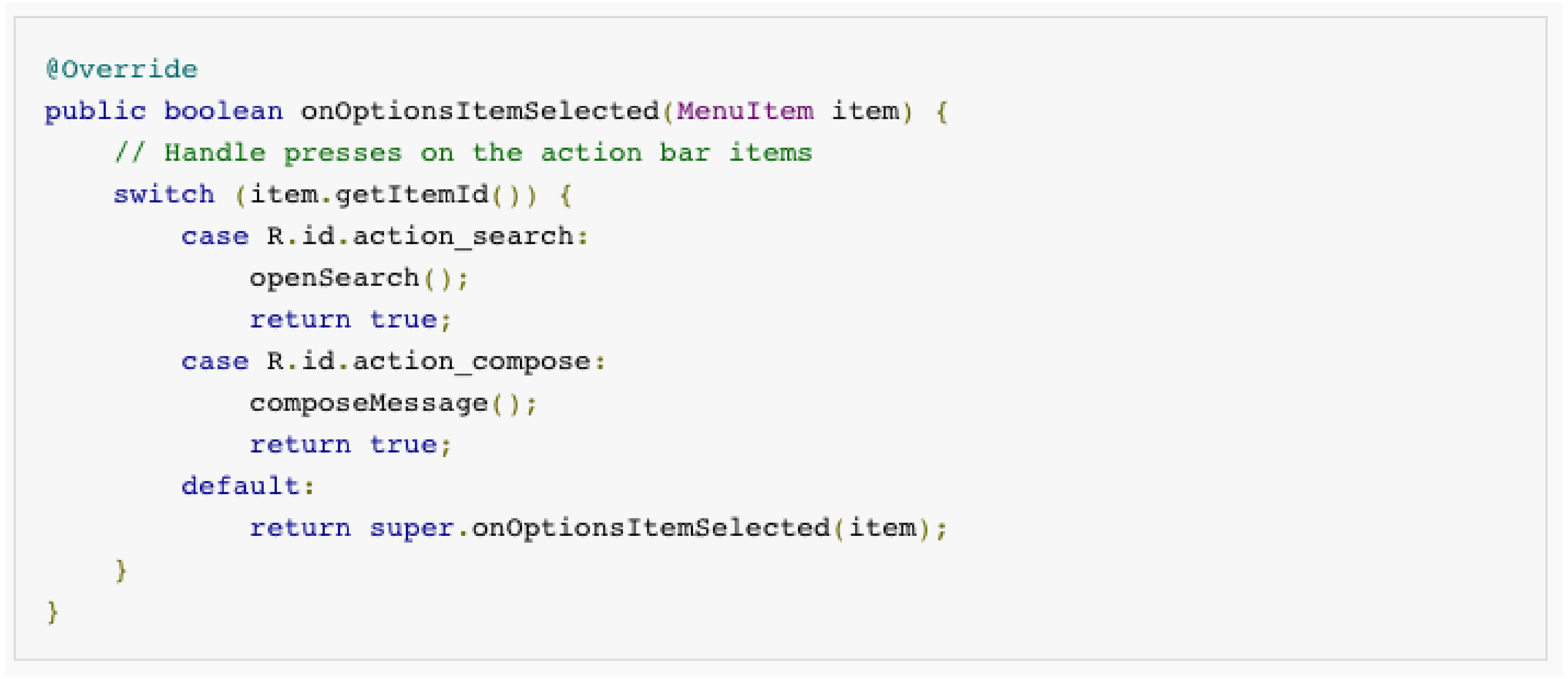
res/menu/main_activity_actions.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:id="@+id/action_search"
          android:icon="@drawable/ic_action_search"
          android:title="@string/action_search"/>
    <item android:id="@+id/action_compose"
          android:icon="@drawable/ic_action_compose"
          android:title="@string/action_compose" />
</menu>
```

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu items for use in the action bar
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.main_activity_actions, menu);
    return super.onCreateOptionsMenu(menu);
}
```

Action Bar – Handling clicks

When the user presses an action, the system calls your activity's onOptionsItemSelected() method. Using the MenuItem passed to this method, you can identify the action by calling getItemId(). This returns the unique ID provided by the <item> tag's id attribute so you can perform the appropriate action.



```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle presses on the action bar items
    switch (item.getItemId()) {
        case R.id.action_search:
            openSearch();
            return true;
        case R.id.action_compose:
            composeMessage();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Resources

- Android Developers - Training

http://developer.android.com/training/index.html

- Genymotion

http://www.genymotion.com/

- Gradle

http://www.gradle.org/

Practice:
Create simple textview clock

Practice:
Count activity visible time

Thank you

Lucio Cossio

Luis Mazoni