SECTION X

NAVIGATION

## PERFORMANCE CRITERIA

ICAPRG409A: Develop mobile applications

x-x xxx xxx xxx

x-x xxx xxx xxx

x-x xxx xxx xxx

x-x xxx xxx xxx

## TOPICS

1. xxx
2. xxx

## Topic: Navigation

The Action Bar is a dedicated space to place important and commonly used navigation items. It has four main sections. If your application has a minimum SDK of API 7, each time a blank activity is added, it will automatically extend the ActionBarActivity class.

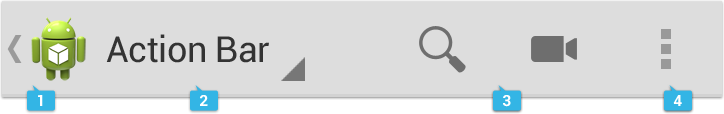


Image source: <http://developer.android.com/design/patterns/actionbar.html>

1. App icon: This area of the action bar can be customised with your app’s icon.
2. View control: This area of the action bar can be used to display your app’s title. You may also provide additional links to views in it’s dropdown menu.
3. Action buttons: Action button provide a way to get to the most important sections of your app, such as settings.
4. Overflow: This area is populated with Actions Buttons which are not used as often.

In this section we will focus on using xml resources to create action buttons and items in the overflow menu.

**Xml menu resource**

Each time a new activity is added to the project, a corresponding xml menu resource file is created in the res/layout directory.

The following is an example of an xml resource called main.xml:

<menu 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"

tools:context="com.cp.dma.actionbarapp.MainActivity" >

<item

android:id="@+id/action\_settings"

android:orderInCategory="100"

android:title="@string/action\_settings"

app:showAsAction="never"/>

</menu>

Notice that there is a single item in this file, and it has the id action\_settings. As you can image, an xml menu resource may contain many items.

Each menu item has an id and a title. These should be defined in strings.xml.

Also specified is the orderInCategory attribute. This allows you to set the order in which multiple items should appear.

The showAsAction attribute specifies where the item should be display. Menu items can be displayed as action buttons, across the top of the action bar, or they can be placed in the overflow menu.

The following showAsAction attribute values allow you to specify where a menu item should be displayed:

* never: This item should always be displayed in the overflow.
* ifRoom: Display the item as an action button if there is room to do so.

**onCreateMenuOptions method**

The menu items are added to the action bar during the onCreateOptionsMenu method. This is known as inflation. This method is an event in the ActionBarActivity class lifecycle and as such, it should not be explicitly called in your own code.

This method is automatically overridden in each activity that extends the ActionBarActivity class.

The method has a single parameter which is the menu instance being created.

The following shows this method:

@Override

**public** **boolean** onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.***main***, menu);

**return** **true**;

}

The menu is inflated using a menu resource. In this case, the menu resource being inflated is main.xml.

Action buttons

As mentioned previously, when a menu item specifies ‘always’ for its showAsAction attribute, it becomes an action button on the action bar.

An action button can also display an icon. Icons can be added to the res/drawable-x directory. They are referenced by name, omitting the file extension (such as png, jpg, etc). The icon is set using the icon attribute.

The following menu item example will create an action button which displays an icon instead of text. Not the inclusion of an icon attribute.

<item

android:id="@+id/action\_settings"

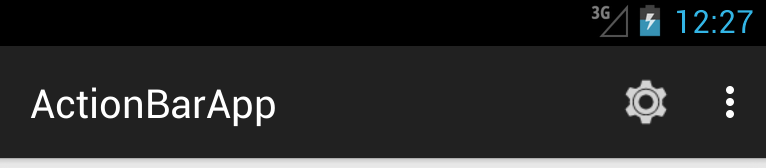
android:icon="@drawable/ic\_action\_settings"

android:orderInCategory="100"

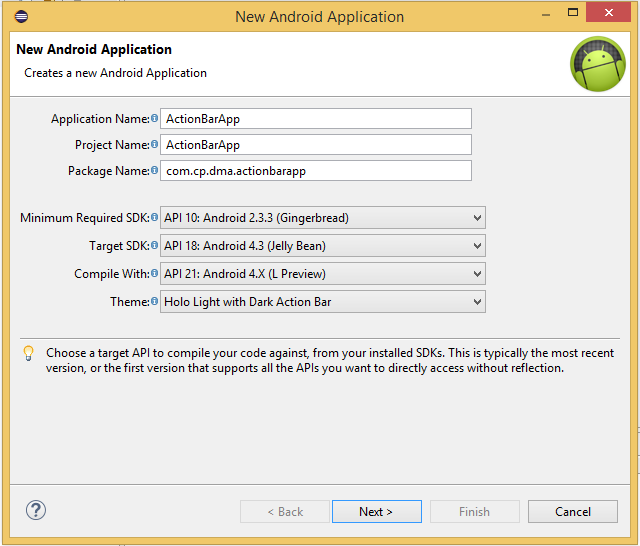
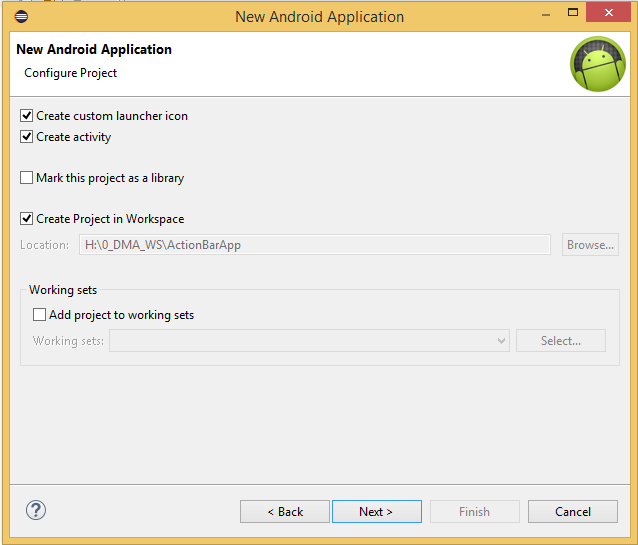
android:title="@string/action\_settings"

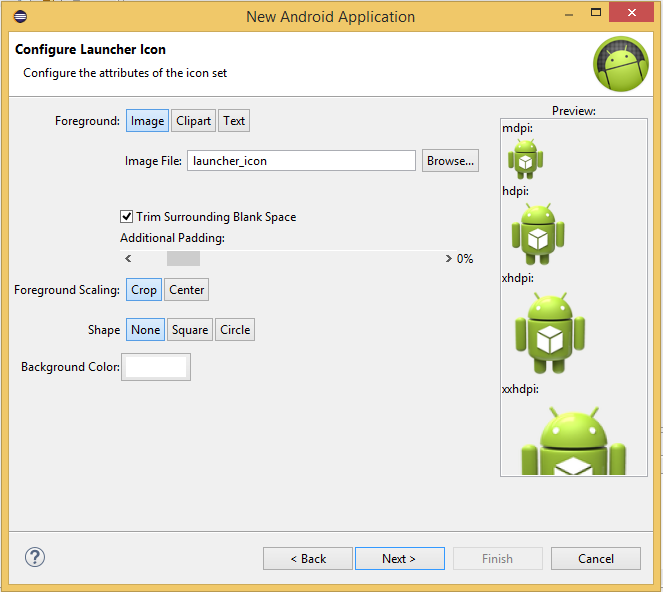
app:showAsAction="ifRoom"/>

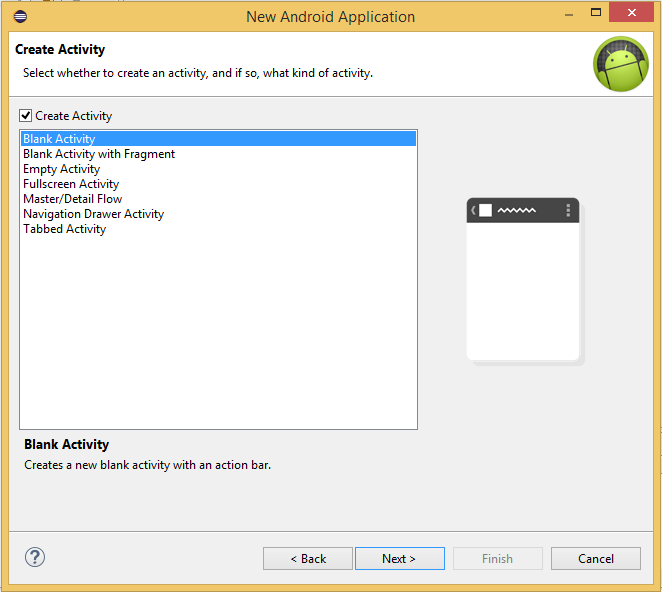
Here is the menu item rendered as an action button on the action bar:

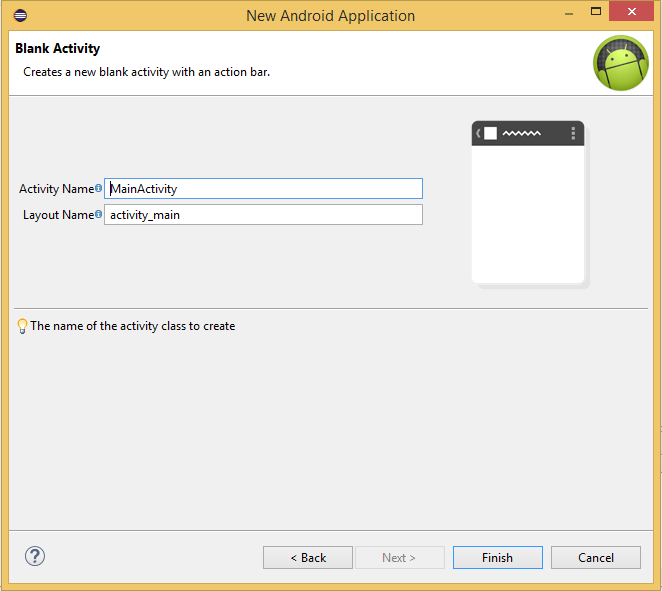


## Activity: Examining the Action Bar.

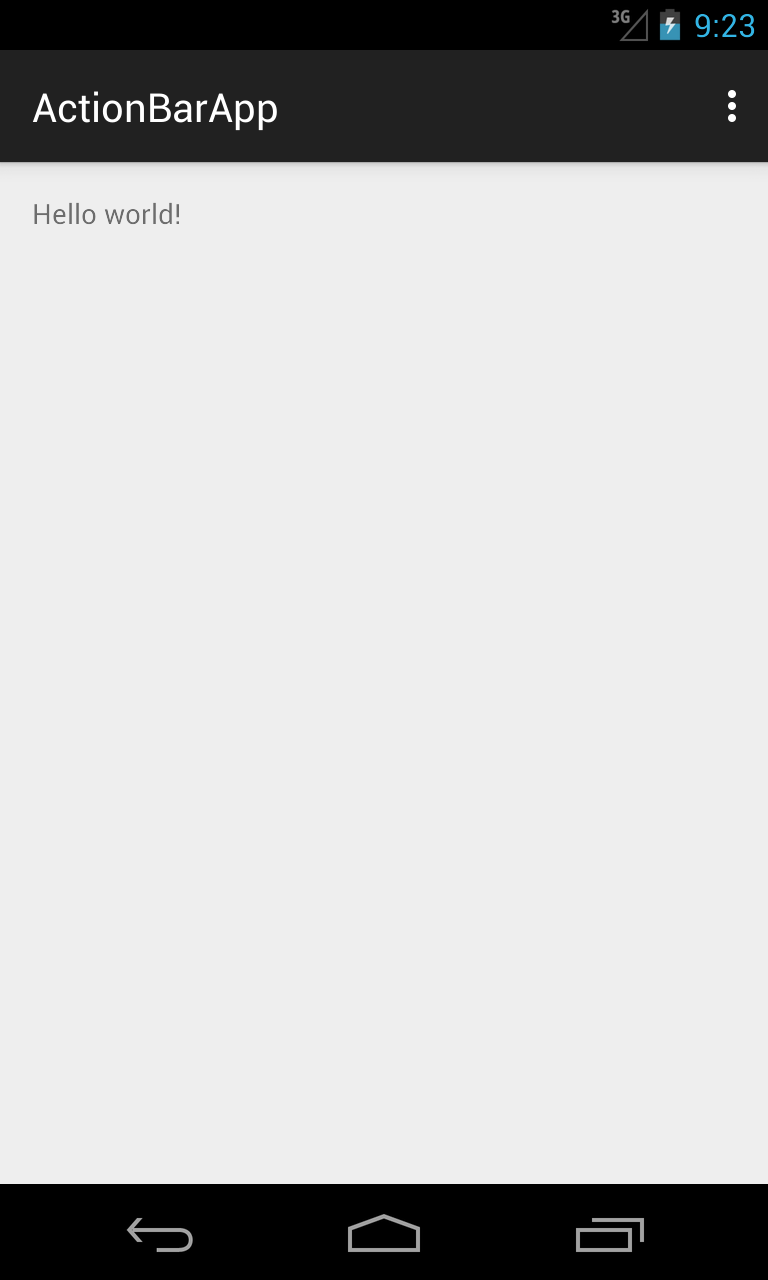
1. Create a new Android Application Project using the Eclipse IDE and enter the following:
   * Application Name: ActionBarApp
   * Project Name: ActionBarApp
   * Package Name: com.cp.dma.actionbarapp
   * Minimum Required SDK: API 10
   * Target SDK: API 18
   * Compile With: API 21
   * Theme: Holo Light with Dark Action Bar
2. Click Next.
3. Leave all options as is and click Next.
4. Leave all options as it and click Next.



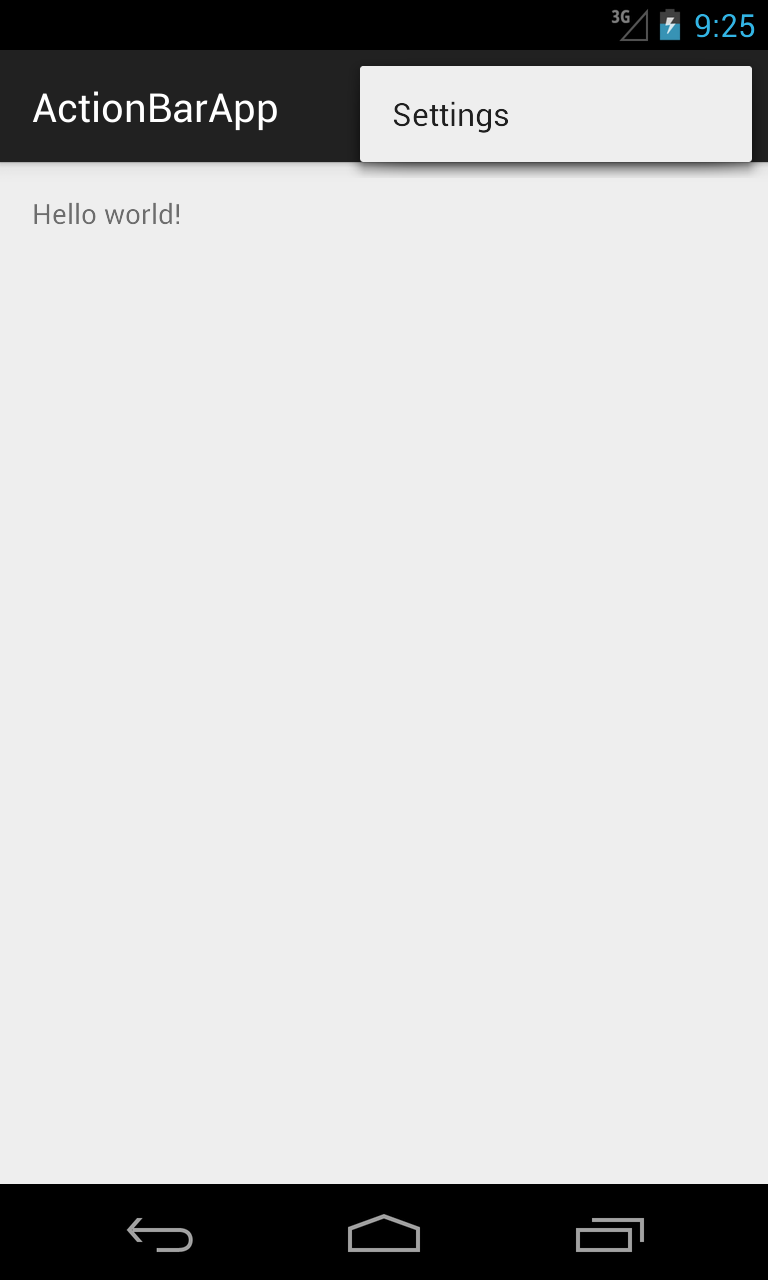
1. Ensure that Blank Activity is selected and click Next.
2. Name the activity MainActivity and click Finish.



1. Open MainActivity.java from the src/com.cp.dma.actionappactivity package and note the following:
   * The class declaration which states that it extends ActionBarActivity.
   * The onCreateOptionsMenu method.
     + The menu resource being inflated is called main.xml.
   * The onOptionsItemSelected method.
2. Open main.xml from the res/menu directory and note the single item with an id of action\_settings. Its showAsAction attribute is set to never and as such, this item will be placed into the overflow menu.
3. Launch the app on an AVD to see the result.



1. Click on the overflow button to see the settings menu item.



At present, clicking on the Settings option is not responsive. We will handle menu selections in the next section.

## Activity: Adding items to the action bar menu

In this activity we will add two more activities to the app and then create menu items which will allow the user to navigate around each of them.

1. In the ActionBarApp project you created in the previous activity, add a new blank activity called ActivityTwo.
2. Open ActivityTwo.java and locate the onCreateMenuOptions method. Note the name of the xml menu resource being inflated.
3. Add a new blank activity called Settings.
4. Open Settings.java and locate the onCreateMenuOptions method. Note the name of the xml menu resource being inflated.
5. Open the res/menu directory. There should now be three xml menu resource files. Notice that each activity has its own xml menu resource file.
6. Open main.xml from the res/menu directory.
7. Replace the current item with the following items:

<item

android:id=*"@+id/action\_activity\_two"*

android:orderInCategory=*"1"*

android:title=*"@string/action\_activity\_two"*

app:showAsAction=*"ifRoom"*/>

<item

android:id=*"@+id/action\_settings"*

android:orderInCategory=*"2"*

android:title=*"@string/action\_settings"*

app:showAsAction=*"never"*/>

1. Open the layout file for ActivityTwo from the res/menu directory.
2. Replace the current item with the following items:

<item

android:id=*"@+id/action\_activity\_main"*

android:orderInCategory=*"1"*

android:title=*"@string/action\_activity\_main"*

app:showAsAction=*"ifRoom"*/>

<item

android:id=*"@+id/action\_settings"*

android:orderInCategory=*"2"*

android:title=*"@string/action\_settings"*

app:showAsAction=*"never"*/>

1. Open the layout file for Settings from the res/menu directory.
2. Replace the current item with the following items:

<item

android:id=*"@+id/action\_activity\_main"*

android:orderInCategory=*"1"*

android:title=*"@string/action\_activity\_main"*

app:showAsAction=*"ifRoom"*/>

<item

android:id=*"@+id/action\_activity\_two"*

android:orderInCategory=*"2"*

android:title=*"@string/action\_activity\_two"*

app:showAsAction=*"never"*/>

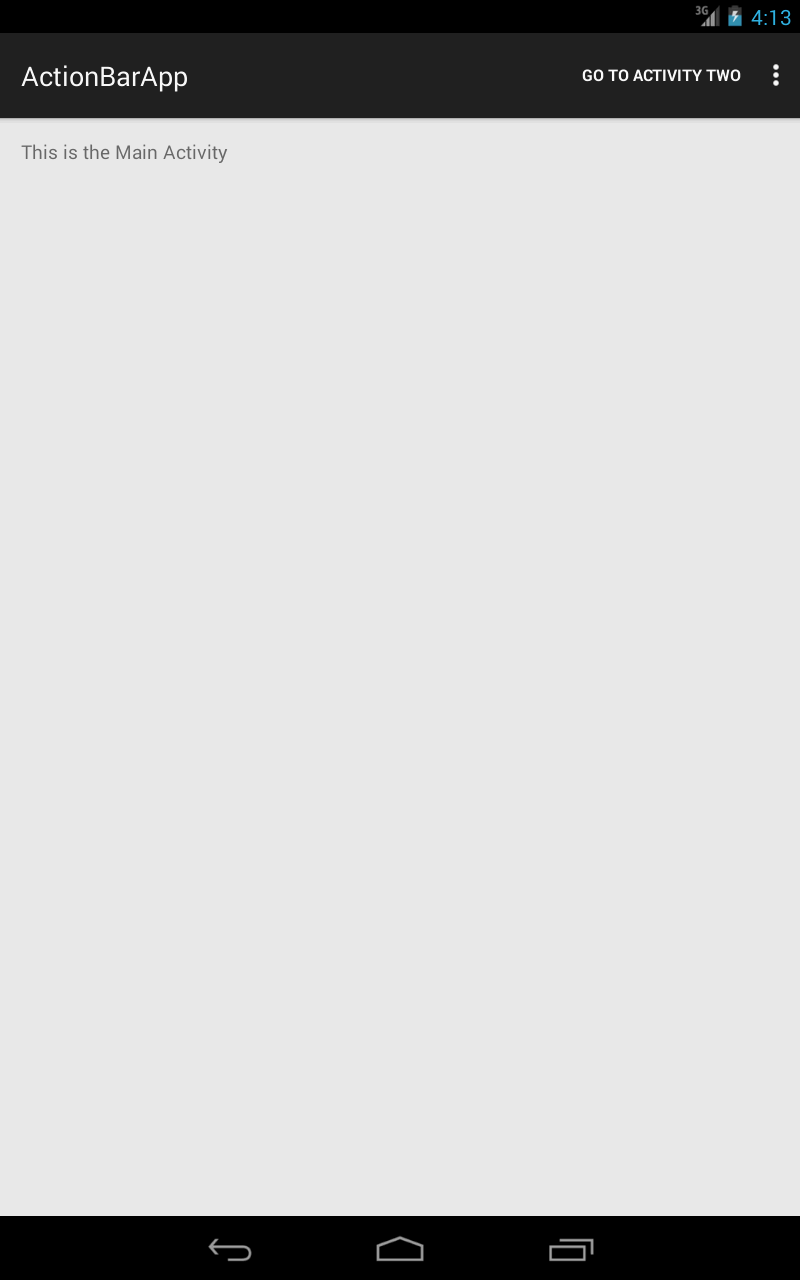
1. Add the following to values/strings.xml:

<string name=*"action\_settings"*>Settings</string>

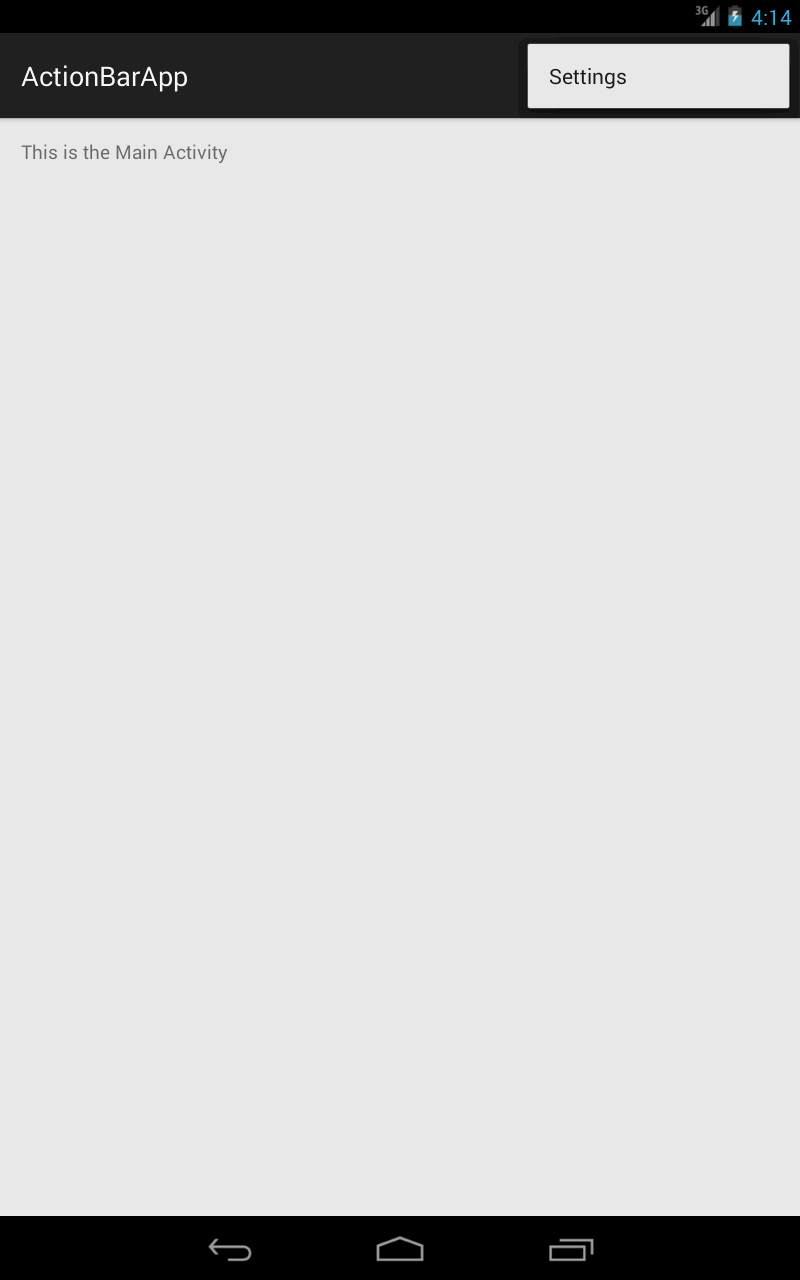
<string name=*"action\_activity\_main"*>Main</string>

<string name=*"action\_activity\_two"*>Go to Activity Two</string>

1. Run the app and view the result on MainActivity, as shown below:



Clicking the overflow button will produce the following result:



Note that you cannot actually select an item as yet. In the next activity we will make each item respond to selections.

**onOptionsItemSelected method**

The onOptionsItemSelected method is automatically overridden in each activity that extends the ActionBarActivity class. This is an event which is executed when a menu option is selected.

The method has a single MenuItem parameter and from this we can determine which menu option was selected and how to respond.

By default, when an activity is first created, the method looks like the following:

@Override

public boolean onOptionsItemSelected(MenuItem item) {

int id = item.getItemId();

if (id == R.id.action\_settings) {

return true;

}

return super.onOptionsItemSelected(item);

}

Calling the getItemId method of MenuItem returns the resource id of the selected item. We can then compare that to the resource ids in the xml menu layout and take the appropriate action.

## Activity: Handling action bar menu selections

In this activity, we will add code to the onOptionsItemSelected method of each activity which will handle menu selections. To do this, we will use a Switch. Each menu option will start a new activity of the appropriate type.

1. Open MainActivity.java and replace the existing code in the onOptionsItemSelected method with the following:

Intent intent = null;

int id = item.getItemId();

switch (id) {

case R.id.action\_activity\_two:

intent = new Intent(this, ActivityTwo.class);

startActivity(intent);

return true;

case R.id.action\_settings:

intent = new Intent(this, SettingsActivity.class);

startActivity(intent);

return true;

}

return super.onOptionsItemSelected(item);

The switch examines the resource id of the selected option and then creates and starts the appropriate intent to start the new activity.

1. Open ActivityTwo.java and replace the existing code in the onOptionsItemSelected method with the following:

Intent intent = null;

int id = item.getItemId();

switch (id) {

case R.id.action\_activity\_main:

intent = new Intent(this, ActivityTwo.class);

startActivity(intent);

return true;

case R.id.action\_settings:

intent = new Intent(this, SettingsActivity.class);

startActivity(intent);

return true;

}

return super.onOptionsItemSelected(item);

This time, we are omitting the ‘action\_activity\_two’ resource id. We are instead including the action\_activity\_main resource id.

1. Open Settings.java and replace the existing code in the onOptionsItemSelected method with the following:

Intent intent = **null**;

**int** id = item.getItemId();

**switch** (id) {

**case** R.id.***action\_activity\_main***:

intent = **new** Intent(**this**, MainActivity.**class**);

startActivity(intent);

**return** **true**;

**case** R.id.***action\_activity\_two***:

intent = **new** Intent(**this**, ActivityTwo.**class**);

startActivity(intent);

**return** **true**;

}

**return** **super**.onOptionsItemSelected(item);

This time, we are omitting the ‘action\_settings’ resource id. We are instead including the action\_activity\_two resource id.

1. Run the app and click each menu item, on each activity. The app should now be able to navigate to each activity.

**Menu icons**

As mentioned previously, we can optionally display an icon for each menu item that is displayed as an action button, rather than the text. An action bar icon set is available from the Android developer site.

Download the action bar icon pack from https://developer.android.com/design/downloads/index.html