LAB 7 – Creating and Managing Overflow Menus on Android

At the end of this lab, students should be able to:

- Demonstrate the creation of Overflow menus both manually via XML and visually using the Android Studio Layout Editor tool.

The Overflow Menu

The overflow menu (also referred to as the options menu) is a menu that is accessible to the user from the device display and allows the developer to include other application options beyond those included in the user interface of the application. The location of the overflow menu is dependent upon the version of Android that is running on the device. With the Android 4.0 release and later, the overflow menu button is located in the top right-hand corner (Figure 13.1) in the action toolbar represented by the stack of three squares:

Figure 13.1

Creating the Project- An Example

a) Creating the Example Application

i. Go to File -> New Project

ii. Choose **Basic Activity** template, then click Next

iii. Enter *MenuExample* into the Name field and specify com.ebookfrenzy.menuexample as the package name.
Change the Minimum API level setting to API26: Android 8.0(Oreo) and the Language menu to Java.
Click Finish.

iv. When the project has been created, navigate to the *app -> res -> layout* folder in the Project tool window and double-click on the *content_main.xml* file to load it into the Android Studio Menu Editor tool. Switch the tool to Design mode, select the ConstraintLayout from the Component Tree panel and enter *layoutView* into the ID field of the Attributes panel.

b) Designing the Menu

Within the Project tool window, locate the project's *app -> res -> menu -> menu_main.xml* file and double-click on it to load it into the Layout Editor tool. Switch to Design mode if necessary and select and delete the default Settings menu item added by Android Studio so that the menu currently has no items.

i. From the palette, click and drag a menu group object onto the title bar of the layout canvas as highlighted in Figure 13.2:

Figure 13.2

ii. Although the group item has been added, it will be invisible within the layout. To verify the presence of the element, refer to the Component Tree panel where the group will be listed as a child of the menu:

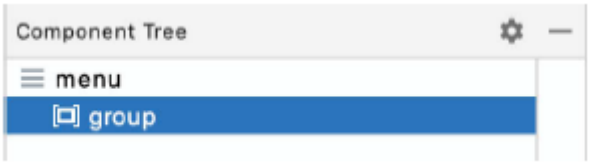

Figure 13.3

iii. Select the *group* entry in the Component Tree and, referring to the Attributes panel, set the *checkableBehavior* property to single so that only one group menu item can be selected at any one time:

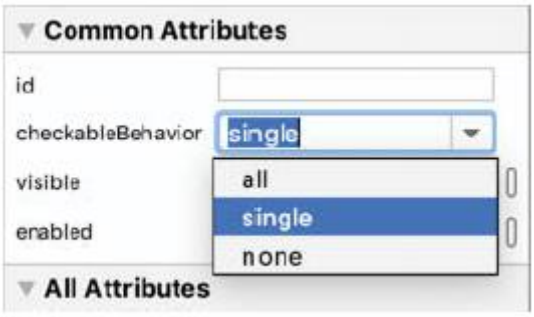

Figure 13.4

iv. Next, drag four *Menu Item* elements from the palette and drop them onto the *group* element in the Component Tree. Select the first item and use the Attributes panel to change the title to "Red" and the ID to *menu_red*:

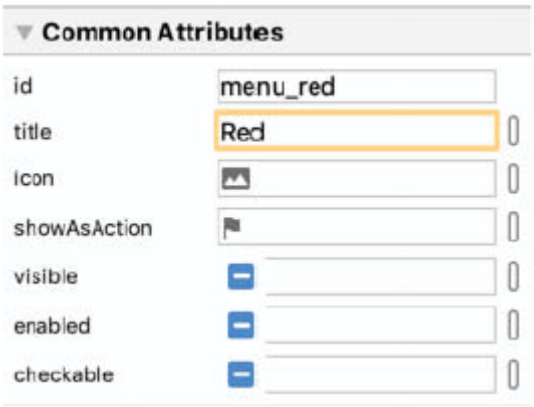

Figure 13.5

v. Repeat these steps for the remaining three menu items setting the titles to "Green","Yellow" and "Blue" with matching IDs of *menu_green, menu_yellow* and *menu_blue*. Use the warning buttons to the right of the menu items in the Component Tree panel to extract the strings to resources:

Figure 13.6

On completion of these steps, the menu layout should match that shown in Figure 13.7 below:

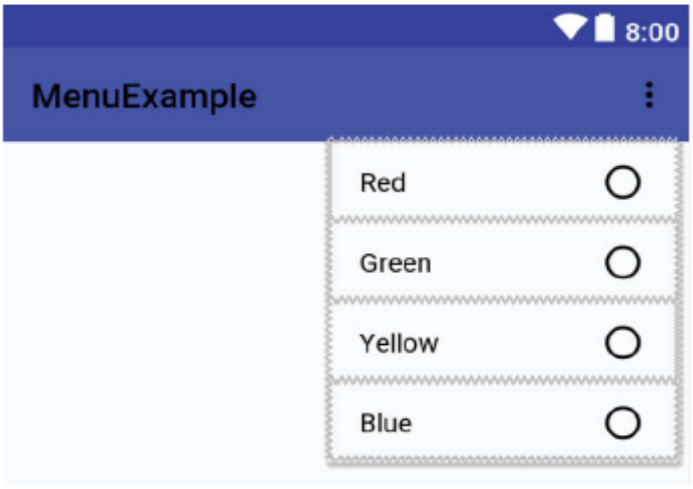

Figure 13.7

Switch the Layout Editor tool to Code mode and review the XML representation of the menu which should match the following listing:

```
<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.ebookfrenzy.menuexample.MainActivity">
<group android:checkableBehavior="single">
<item android:title="@string/red_string"
android:id="@+id/menu_red" />
<item android:title="@string/green_string"
android:id="@+id/menu_green" />
<item android:title="@string/yellow_string"
android:id="@+id/menu_yellow" />
<item android:title="@string/blue_string"
android:id="@+id/menu_blue" />
</group>
</menu>
```

c) Modifying the onOptionsItemSelected() Method

When items are selected from the menu, the overridden *onOptionsItemsSelected()* method of the application's activity will be called. The role of this method will be to identify which item was selected and change the background color of the layout view to the corresponding color.

i. Locate and double-click on the *app -> java -> com.ebookfrenzy.menuexample -> MainActivity* file and modify the method as follows:

```
package com.ebookfrenzy.menuexample;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.android.material.snackbar.Snackbar;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.constraintlayout.widget.ConstraintLayout;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
public class MainActivity extends AppCompatActivity {
..
@Override
public boolean onOptionsItemSelected(MenuItem item) {
ConstraintLayout mainLayout =
findViewById(R.id.layoutView);
switch (item.getItemId()) {
case R.id.menu_red:
if (item.isChecked()) item.setChecked(false);
else item.setChecked(true);
mainLayout.setBackgroundColor(android.graphics.Color.RED);
return true;
case R.id.menu_green:
if (item.isChecked()) item.setChecked(false);
else item.setChecked(true);
mainLayout.setBackgroundColor(android.graphics.Color.GREEN);
return true;
case R.id.menu_yellow:
if (item.isChecked()) item.setChecked(false);
```

```
else item.setChecked(true);
mainLayout.setBackgroundColor(android.graphics.Color.YELLOW);
return true;
case R.id.menu_blue:
if (item.isChecked()) item.setChecked(false);
else item.setChecked(true);
mainLayout.setBackgroundColor(android.graphics.Color.BLUE);
return true;
default:
return super.onOptionsItemSelected(item);
}
}
.
.
}
```

d) Testing the Application

Build and run the application on either an emulator or physical Android device. Using the overflow menu, select menu items and verify that the layout background color changes appropriately. Note that the currently selected color is displayed as the checked item in the menu.