Introduction to Mobile Device Programming

M3G621212

Lab Session 2

In this set of exercises it is assumed that you have Eclipse up and running and that you have created an appropriate Android Virtual Device in the same manner as in lab session 1.

Task 1

Download the zip file containing the HelloAndroidWorld example used in lectures. Unzip this file and import the project into Eclipse. Information on importing projects is given in the supplementary notes.

Deploy the project to an appropriate Android virtual device.

The application just displays a welcome message and a Button object. Pressing the button object results in the application exiting.

Task 2

In the Lab area for this module on GCULearn you will find a zip file called colours.zip which contains 2 xml resources files containing a list of W3C and X11 defined colours respectively. Download and unzip this file and spend a few minutes looking through the contents of each of the resource files. You will be using these files shortly to add some colour to your HelloAndroidWorld application.

Make the following modifications to files in your HelloAndroidWorld project.

Add an id called mainView to main.xml for the top level View in main.xml (The file contained in the layout folder)

android:id="@+id/mainView"

Create and instantiate a View object in the HelloWorldActivity class.

private View mainView; // Add after the definition of the class

mainView = (View)findViewById(R.id.mainView); // Add in the onCreate method

// after setContentView

You should make these steps standard when you create a project because it allows you to get access to the top level view which acts as a container for the components that you add.

Now copy the 2 colour files to the layout folder of your HelloAndroidWorld project. You should be able to copy and paste the files directly into the values folder as shown in the package explorer.

Change the background colour of the application to silver using the definition in the coloursw3c file. You need to do this by adding the line below just after the instantiation of mainView.

// Add towards the end of the onCreate method

mainView.setBackgroundColor(getResources().getColor(R.color.silver));

Now run the application on the emulator as before. You should now have a silver colour for the top level background. You may not be able to see the text in the TextView object at the top of the screen.

If this is the case change the colour of the text in the TextView object to Black by adding an appropriate entry to main.xml. Run the application again and check that the text is now displayed in Black.

Experiment with using colours and text styles with this application. Note the manner in which you change the Typeface for the font in a specific label. You need to make use of the Typeface class. The statements below are entered in the onCreate method.

```
TextView tbox1 = (TextView)findViewById(R.id.textBox1);

tbox1.setTypeface(Typeface.SANS_SERIF);

tbox1.setTypeface(Typeface.MONOSPACE);
```

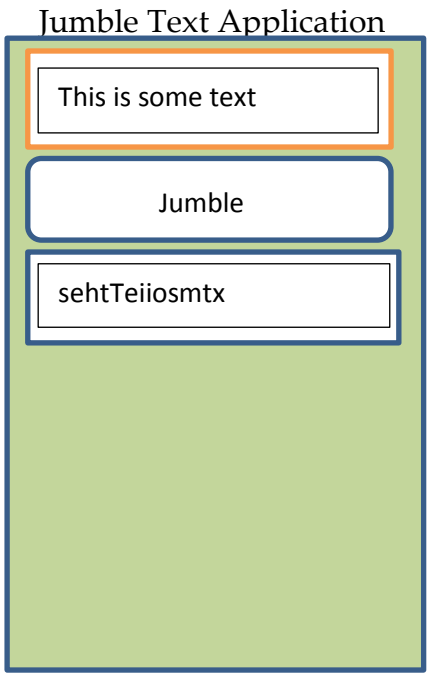
N.B. Make sure that you have include an id in the layout xml file for the TextView object. The id should be called textBox1.

Task 3

Create a new Android project called JumbleTextApplication in the usual way. When this is done you will now have a class called JumbleTextApplicationActivity. Now create a Java class called JumbleText with a single method called jumbleString.

The method jumbleString will take a single String parameter and return a string consisting of the original characters jumbled up.

Create an interface as shown below using appropriate Android widgets.



The top widget should allow the user to enter some text. The widget labelled "Jumble" when clicked will result in the text that was entered appearing in the next widget "jumbled up".

The intention here is for you to develop your own algorithm to jumble the letters. You may find some help on the internet. You probably want to look at generating random numbers between 0 and the length of the line of text being jumbled.

Alternatively you could use the reverse method of the String class in Java to change the text for output into the second TextView box. However, developing a "jumble" letter algorithm should be within your capability. You may find it more straightforward to produce a Java project and use that project to develop the code to jumble up the letters in a word. The neatest solution is to write a method which takes in a String and returns another string with the letters jumbled up. The skeleton of the Java method is shown below.

```
public String jumble(String astring)
{
String jumbledString;
        // Code to jumble up the letters

        return jumbledString

}
```

Using a Java project avoids the need to keep deploying your application to the Android Emulator.

Task 4

Obtain the Android Project ErrorProject and load this project into the Eclipse Workspace so that you can work on the project. The project does not contain any syntax errors but when you deploy the project you will get the dreaded "Unfortunately ErrorProject has stopped" message. Try this for yourself.

You should have found that the project did not fully deploy so it would appear that there are semantic errors in the code that you have been supplied with. Your task is to use investigate what the problem is and use appropriate techniques to track down the error(s).

The purpose of the programme is to provide the user with a button that can be pressed to generate a proverb. The proverb is displayed in a Dialog box. A count of the number of proverbs displayed is displayed in a separate component on the screen.

You may be lucky or able to spot the error(s) quickly by looking through the code but it is more than likely that you will need to "trace" through the program in some way. In order to do this you will need to make use of LogCat to place and view statements as the programme executes. To do this create a tag of your choice and pair this with appropriate messages as you proceed to execute the code.

Log.e(“MyTag” , “At the end of onCreate”);

Etc.

Look though the Java code. You will find that a couple of Log statements have already been added. Add further statements and watch the program execute or not from the Log Cat window. You should get to a point where the programme falls over. Track back from this point to the last displayed Log.e statement.

The Log statements appear in the LogCat tab which is one of the windows on the bottom right of the Eclipse environment.

How many errors did you find?

Task 5

You will revisit this question at a later date using slightly different components to construct a more general application.

Write an Android application to calculate the area of a rectangle. Your application will need to provide 2 “labels” for the length and breadth along with 2 input fields for the data. You will need a further label/output field pair for the result of the calculation.

The layout of the graphical component is at your discretion. There will be scope for revising the layout once we have covered layout managers.

To perform the calculation I suggest that you convert the “entry” string for a piece of data to a Java double type. You will find the code below useful.

double value = new Double(textValue).doubleValue();

You should also catch any number format errors that are thrown.