Session 3 – Tutorial B

The objectives for week one are for you to familiarise yourself with the Android Studio environment and experiment with some basic functionality.

As evidence of completion, upload your projects on SunSpace.

Exercise 1

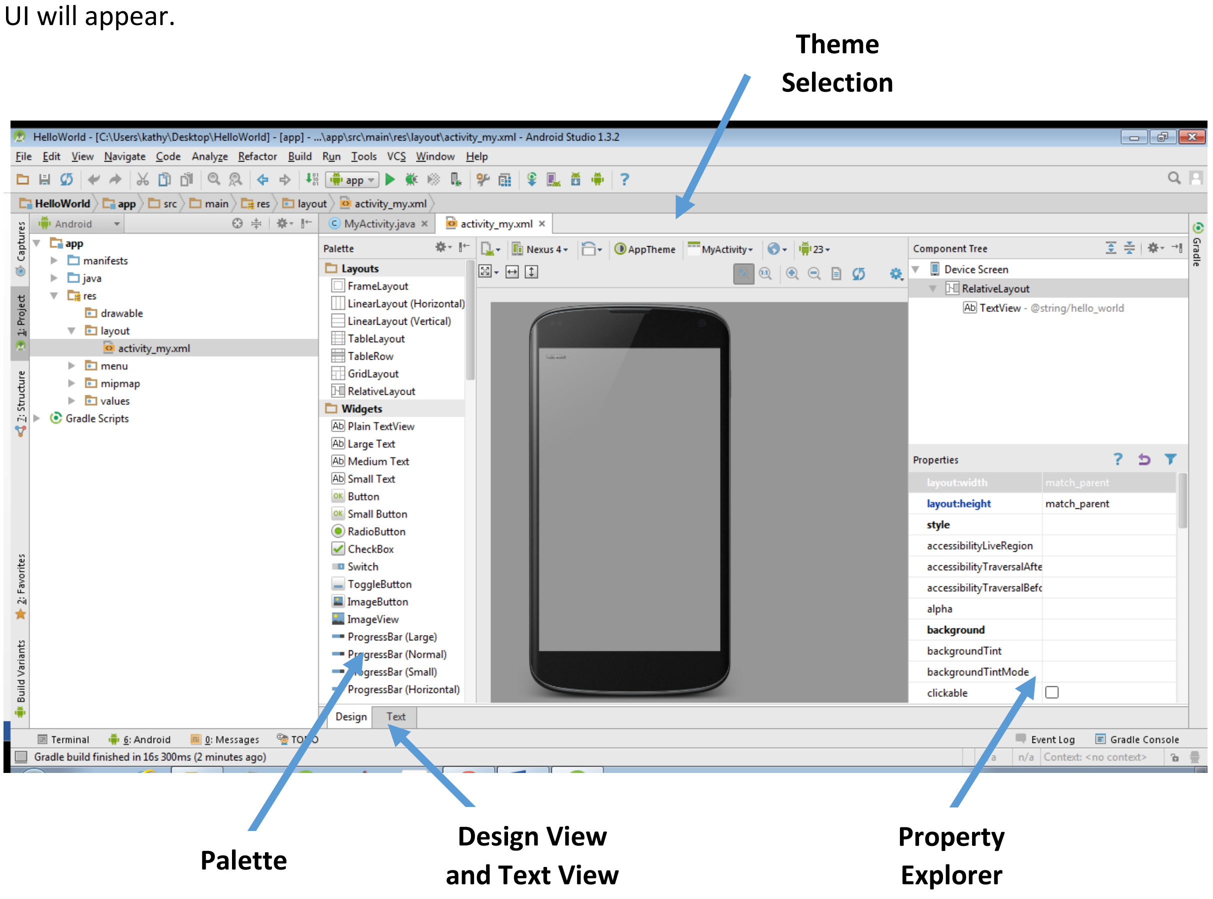
The Android project you have created is a basic "Hello World" app that contains some default files. Explore the Android Studio project, paying particular attention to the following:

- activity_my.xml (app/src/main/res/layout/activity_my.xml)
- MyActivity.java (app/src/main/java/com.mycompany.myfirstapp/MyActivity.java)
- AndroidManifest.xml (app/src/main/AndroidManifest.xml)
- build.gradle (app/build.gradle)
- The /res subdirectories

Using the resources available at developer.android.com, identify and summarise the purpose of each of the files listed above. Add comments to the Java class and XML files, indicating the purpose of each line of code.

Exercise 2

Double click on activity_my.xml. There are two tabs at the bottom of the IDE- one called ‘Design’ and the other called ‘Text’. Look at the ‘Text’ view and inspect the xml code. This defines how your app UI will appear.



HelloWorld - [C:\Users\kathy\Desktop\HelloWorld] - [app] - ...\app\src\main\res\layout\activity_my.xml - Android Studio 1.3.2

Gradle build finished in 16s 300ms (2 minutes ago)

Experiment with changing themes, adding UI components from the Palette, and adjusting their properties in the Property Explorer. After each change check the Text view and evaluate how your XML code has changed. Carry out some independent research online regarding the different types of properties available and what each one means.

Exercise 1

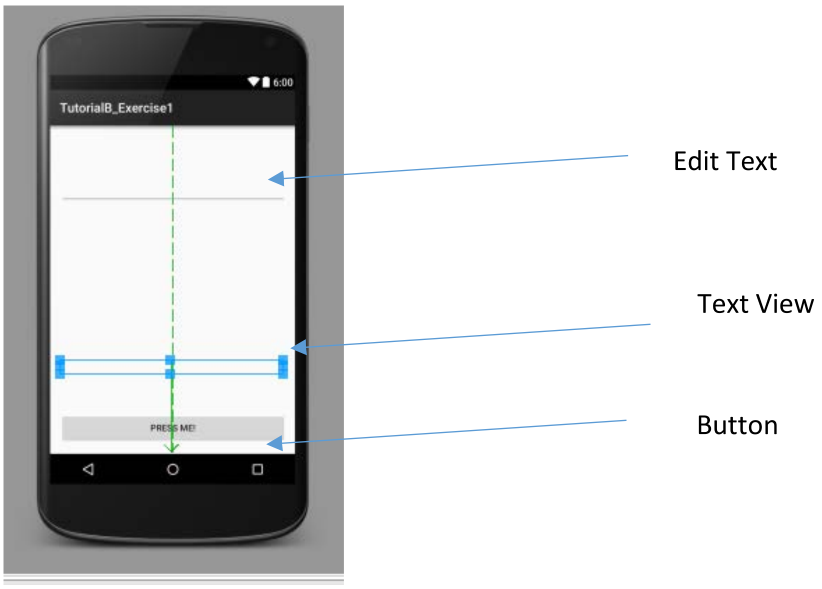
This exercise will help you understand the fundamentals of user interaction introducing how to build Activities which contain basic UI components such as Buttons and EditText widgets.

You will create UI components, reference them programmatically, and respond to user events such as button clicks. Specifically, you will create an application which allows a user to enter text, and then displays that text in a new location whenever a button is clicked.

Step 1 – Create a new Android application, with an empty activity. Give both your application and your activity an appropriate name.

Step 2 – Navigate to the layout file corresponding to your main activity. For example, if your activity is called MainActivity you will have a layout file called activity_main.xml under the res/layout directory.

Using the design view, drag an edit text, a button and a text view component onto your UI as illustrated below. If you wish, you may change the properties so that your application is more visually appealing.



Give each component an ID. You can do this by double clicking on the component and typing the id, or by amending the XML code directly:

```
<EditText

        android:id="@+id/editText1"
```

```
/>


<Button

        android:id="@+id/button1"
/>


<TextView

        android:id="@+id/textView11"
/>
```

Inspect the UI layout code in XML. You will notice that all your UI components are nested within a <RelativeLayout>. This means that the physical position of a component in your activity is relative to other components in that activity. ou can use multiple layouts, including Frame and Table layouts. Each UI component in represented in XML. The property of each component can be set either programmatically, or using the property editor to the right of the design view.

Step 3 – In order to allow user interaction, we have to get references to the components in our view at application runtime. Navigate to MainActivity.java. For each component which you have created in the xml layout file, you must now declare a corresponding member variable in your activity class.

```
EditText editText;
Button button;
TextView textView;
```

When you enter this code, you will notice it is underlined in red. This is because the EditText, Button, and TextView Classes from the Android API have not been imported and are therefore not visible. Hit ALT+Enter, and follow the prompts to import each package.

Step 4 – In step 3 we have simply declared that an EditText, Button and TextView object exist. We have not yet instantiated them. We instantiate them during onCreate().

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

    // Instantiate button to point to button1
    button = (Button) findViewById(R.id.button1);
     //Instantiate edit text
    editText = (EditText) findViewById(R.id.editText1);
    //Instantiate text view
    textView = (TextView) findViewById(R.id.textView1);
}
```

Step 5 – We need to be able to react to when a user clicks the button. We can achieve this by registering an ***event listener*** to the button. The listener will identify when the button has been clicked, and call your implementation of a method called onClick(). We can therefore put our application logic in onClick() to deal with button's user interactions.

Change the Class header so that it implements an onClickListener:

```
public class MainActivity extends AppCompatActivity implements
View.OnClickListener
```

View.OnClickListener is an interface. When implementing an interface you must provide implementations of that interfaces methods. Create a method called onClick(). Within this method we are going to retrieve any text that the user has entered into editText1 and display that text within textBox1:

```
@Override
public void onClick(View v) {

    int id = v.getId();

    if(id == R.id.button1){
        String name = editText.getText().toString();
        textView.setText(name);
    }
}
```

Step 6 - Finally, link the button and the event listener together in onCreate():

```
button.setOnClickListener(this);
```

Your full onCreate() method is shown below:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    button = (Button) findViewById(R.id.button1);
    button.setOnClickListener(this);
    editText = (EditText) findViewById(R.id.editText1);
    textView = (TextView) findViewById(R.id.textView1);
}
```

Step 6 – Execute your code and ensure it operates as expected.

Exercise 4

Modify the app of Exercise 3. The text on the button should be “Copy”. The text should be not hardcoded but should be on an external resource file.

Exercise 5

Modify the app of Exercise 4 in order to accept in input a specific type (and test each one)

- Numbers
- Text with no Numbers
- Password
- Visible Password
- Email address
- Date
- Time