

## Task 1 – Activities and Fragments

Fragments are a sub activity that represents a behaviour or portion of a user interface inside an activity. A fragment offers us a reusable way to combine parts of user interface in an activity. A fragment can be reused in multiple activities as it is the modular section of an activity.

Fragments have their own lifecycle and can be added or removed to activities and can receive its own input events separate from the activity. Fragments are always contained within an activity and the fragment's lifecycle is directly affected by the host's activity lifecycle.

Fragments are designed to support more dynamic and flexible UI designs without managing complex changes. Fragments are also able to modify an activity's visuals at runtime.

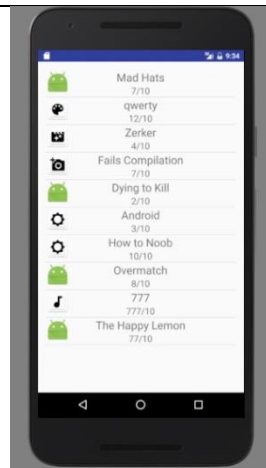
An example of Fragments would be in the tablet version of Gmail where you have the list of emails as one fragment and reading an email as another fragment.

To manage fragments in an activity you need a Fragment Manager which can get existing fragments, pop fragments off the back stack and Register a listener for changes to the back stack.

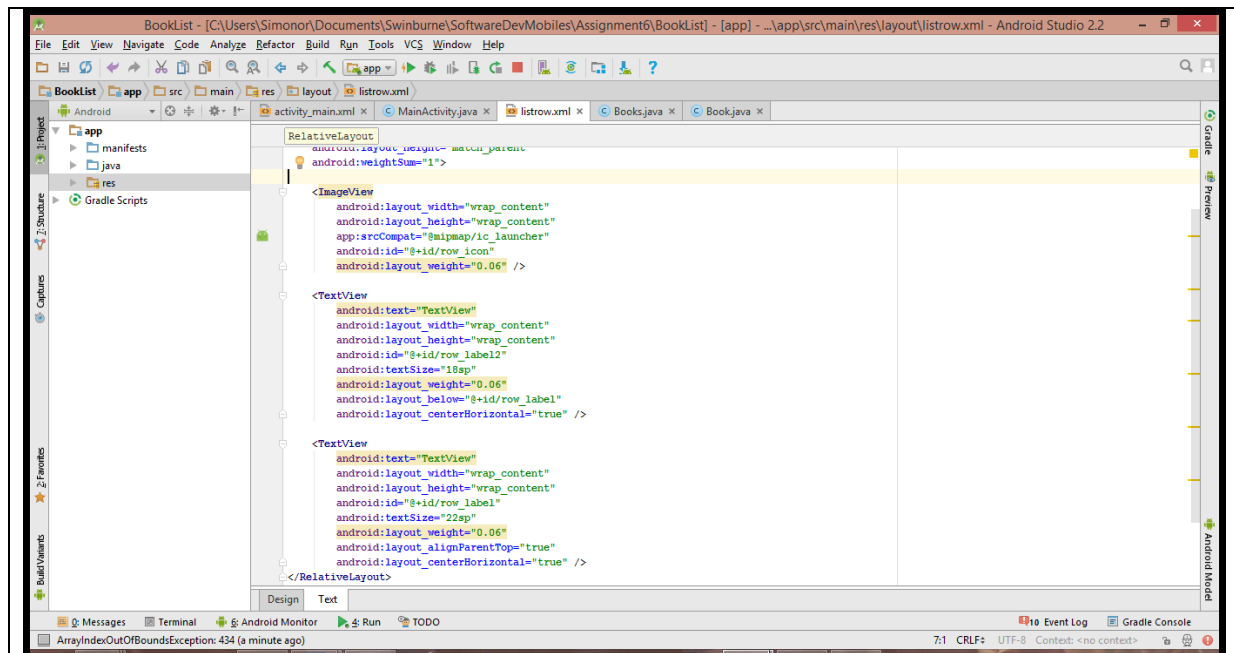
Fragment Transaction gives you the ability to add, remove, replace and perform other actions with the fragments in response to user interaction. This also allows you to save each transaction to a back stack which allows you to navigate backwards through the fragment changes.

## Task 2 – Simple Custom List

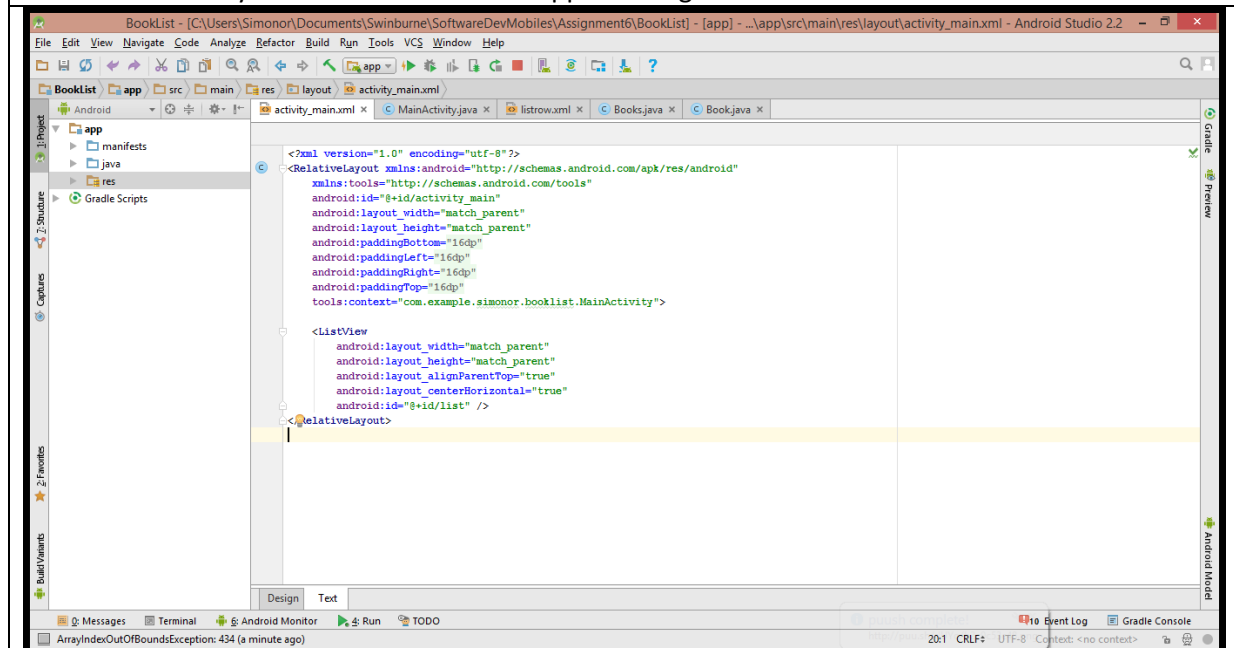
Here is the Picture showing off the list functioning on the emulator



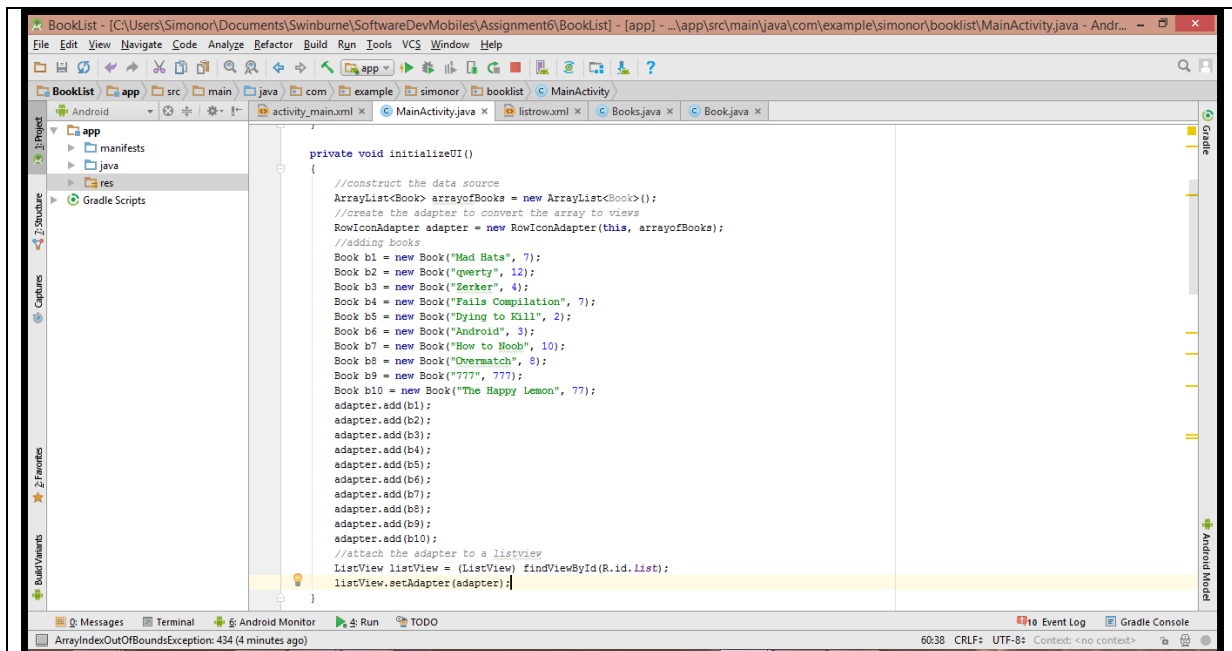
This is the custom layout I created to display each row to meet specification



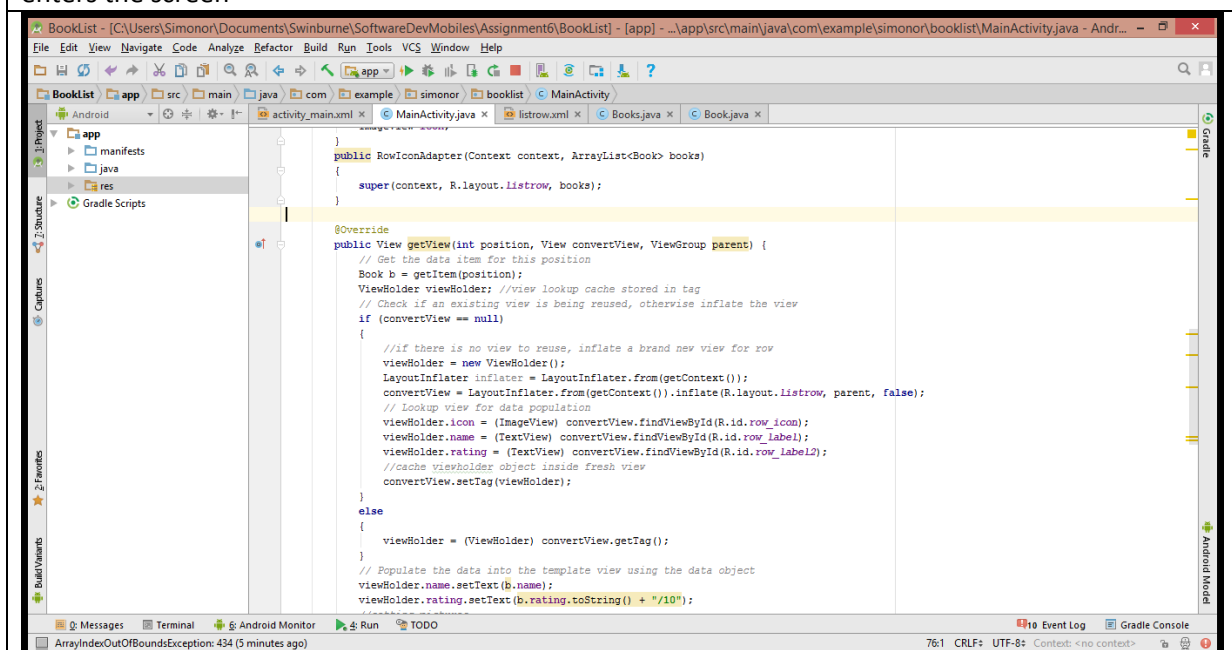
This is the main layout which all the work happens using the listview



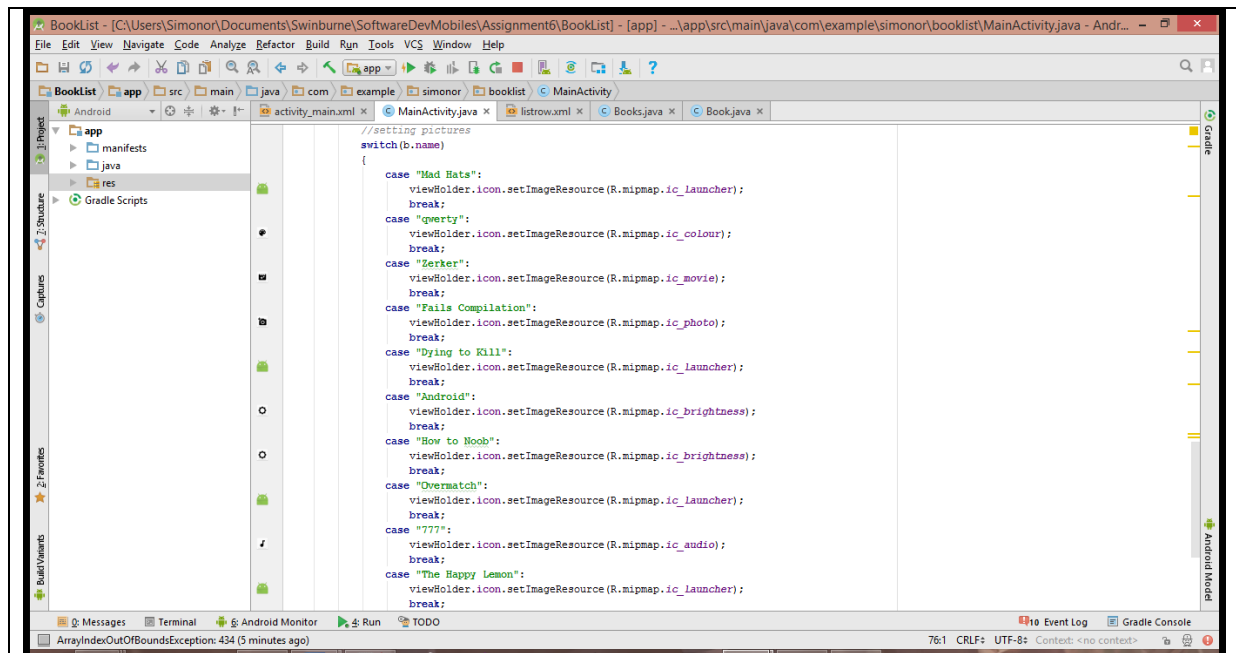
This code shows setting up the adapter in the activity and adds book values to the adapter to be added on the screen.



This is the bulk of the code for the adaptor allowing for the custom layout . This has been optimized so that each time a row goes off the screen it wont need to create a new object as a new object enters the screen



The code to display the icons for the books depending on the name assigns the icon to the imageview



### Task 3 – Action Bar Design Pattern

The Action Bar design pattern is recommended by the Android design guidelines due to a range of features. The action bar is a bar at the top of an application that includes all the actions that one could perform in the app. An example of an action bar would be the google docs.



The action bar contains the most used actions, with in the case of google docs would be on the right the search, layout, sort and open respectively.

There are many advantages of the Action bar over dashboard design. The action bar Provides a visual structure and interactive elements that are consistent with other android apps allowing users to easily understand how to use your app. The action bar is the primary collection of menu items for an activity. This allows for easy access to important actions, support for navigation and view switching and a dedicated space to give your app an identity and inform the user of their location within your app. Having an action bar reduces clutter by providing an action overflow (3 dots) for lesser used actions to store them without filling the screen unnecessarily. A benefit of the action bar is there is no need to have a physical button on the phone to display the options menu and interactivity with the app.

### Task 4 – Add a Custom Geo Location

This is the code for reading in the data from the locally stored file to a list of strings

```
try {
    FileInputStream fis = this.openFileInput("1.txt");
    InputStreamReader isr = new InputStreamReader(fis);

    BufferedReader br = new BufferedReader(isr);
    String l;
    while((l = br.readLine()) != null)
    {
        result.add(l.toString());
        Log.e("line", l.toString());
    }
    br.close();
    Log.e("no more lines", "left to read");
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
```

This converts the list of strings into a list of Cities by breaks apart the line into individual elements and assigning them respectively in the City Constructor

```
void getData(ArrayList<String> s)
{
    for(String l : s)
    {
        Log.e("line", l);
        String[] entry = l.split(",");
        for(int q = 0; q < entry.length; q++)
        {
            Log.e("ec", entry[q]);
        }
        Log.e("entry length", Integer.toString(entry.length));
        if(entry.length == 4)
        {
            City c = new City(entry[0], Double.parseDouble(entry[1]), Double.parseDouble(entry[2]), TimeZone.getTimeZone(entry[3]));
            Cities.add(c);
        }
    }
}
```

This code snippet gets the information from the text boxes and converts it into a line and then writes the line to the file.

```
EditText name = (EditText) findViewById(R.id.name);
EditText lat = (EditText) findViewById(R.id.latitude);
EditText lon = (EditText) findViewById(R.id.longitude);
EditText tz = (EditText) findViewById(R.id.tz);

String n = name.getText().toString();
String la = lat.getText().toString();
String lo = lon.getText().toString();
String t = tz.getText().toString();

String line = "\n" + n + "," + la + "," + lo + "," + t;
byte[] b = line.getBytes();


FileOutputStream out = null;
try {
    out = openFileOutput("1.txt", MODE_APPEND);
    out.write(b);
    out.close();
} catch (Exception e) {
    e.printStackTrace();
}
```

This picture shows that only 2 suburbs that I have already added are listed.



This is the add location screen where you add details about the suburb you want to add including the timezone which can be posted in the offset of the GMT


Android Emulator - Nexus\_5X\_API\_23:5554



The screenshot shows the 'Suntime' app interface on an Android emulator. It features a dark background with white text. The title 'Suntime' is at the top. Below it are four input fields: 'Name' (with 'Bright' entered), 'Latitude' (with '-36.727778' entered), 'Longitude' (with '146.961111' entered), and 'Timezone' (with '+10' entered). A 'SUBMIT' button is at the bottom. The Android navigation bar is visible at the very bottom.

After pressing the submit button and returning to the main page, the new suburb I have added is available to be selected.

Android Emulator - Nexus\_5X\_API\_23:5554



The screenshot shows the 'Suntime' app main screen. It displays the location 'Bright' at the top. Below it, 'Sun Rise' is shown as '19:52' and 'Sun Set' as '08:13' in large orange text. A yellow sun icon is centered below the times. A date bar shows '2016 Wed, 28 Sep'. At the bottom, there is a list of locations: 'Walhalla', 'Orange', and 'Bright'. A 'NEW LOCATION' button is at the bottom of the list. The Android navigation bar is visible at the very bottom.