Project:

Currency Converter

An Android app that converts money from one currency to another

Zachariah Murphy

Contents

1.0 – Analysis

1.1 – Initial ideas

I initially had two ideas for my project, both being smartphone apps but having completely different features and use-cases. One of these was an app that was marketed at tourists visiting my local town that would provide them with local features and sights to see. The users could locate these places on a map and save their 'must see' locations in a database. The local features would range from places to eat and drink, shops and sight-seeing walks.

The second idea was a currency converter app that was marketed at tourists going on holiday abroad. The app would retrieve the latest exchange rates from an external network source and convert the required amount for the user. The user could then save this conversion in a database so that they could view it and compare it to other conversions they had completed.

1.2 – Idea justification

The idea I chose was the currency converter app, this was because I felt that it would provide me with more of a challenge as it contains more advanced concepts, including network requests and persistent data storage.

1.3 – Client identification

My clients for this project will be Mr & Mrs Murphy, they are planning a round-the-world trip where they will need to purchase different currencies during their travel.

1.3.1 – Client requirements

After speaking to my clients over the phone about what they wanted from the app, I sent an email to them so that they could check over the requirements we had discussed. To this, they replied:

> *"Those requirements are fine; but we have also had another couple of ideas we thought could possibly be implemented. One was to have the functionality to convert from multiple currencies, as opposed to just British Pounds. And the other was to have some indication for whether the currency rate has changed from the day before, perhaps with arrows or different symbols to represent this?*
>
> *Hopefully this is okay, and instead of making these additions part of the main requirements they can just be optional, for when everything else is complete."*

To which I responded:

> *"Yes, that is fine. I'll title all the other requirements mandatory, or something like that, and then title these additions optional"*

Therefore, from this communication I had with my clients I was able to produce the client requirements below.

Mandatory:

1. Access different currencies and convert them against GBP
2. Automatically get the conversion rate based on selecting a country
3. Be able to convert real monetary values (i.e. decimal places or large figure values)

4. Use a list to order currencies and place more common ones nearer to the top
5. Use distinguishable flags to help represent currencies in the list
6. Save conversion rates, so that the app can be used when a network connection is not available
7. Have the ability to save converted values for comparisons
8. Be able to delete saved conversions
9. Have a menu that is easy to navigate with symbols, with options including:
    a. Refresh the page
    b. Navigate to view stored data
10. Have a clear and simple layout that makes using the app intuitive
11. Have clear and correctly sized text and headings that are easy to read
12. Have an app icon that has:
    a. A rounded square shape as a background
    b. At least 3 different currencies represented on it
    c. Bold colours
13. Be hardware resource efficient (i.e. not draining battery, consuming large amounts of network traffic (data))

Optional:

14. Be able to choose different base currencies
15. Indicate whether the currency rate has changed from the day before, with arrows to symbolise up or down and equals for has stayed the same

1.4 – Prototype build objectives

Based on these client requirements and speaking with the clients to find out what they want, I created some more build-specific objectives for myself – so that I have a better idea of what I need to do to create a basic version of the app for my client. These are listed below:

1. The user selects a country from a list
2. Based on the country, the app automatically selects the correct currency
3. The user enters how much to convert
4. The conversion process is started
5. Conversion rates are gathered from an external source
6. These values are used to convert the amount the user entered
7. The conversion is shown to the user
8. The user has the option to save the conversion, along with the date
9. The user can view all saved conversions to compare them
10. The user can delete any saved conversions

As well as having these objectives as a starting point, I will be continually referring to the client's requirements in the above section while building the app.

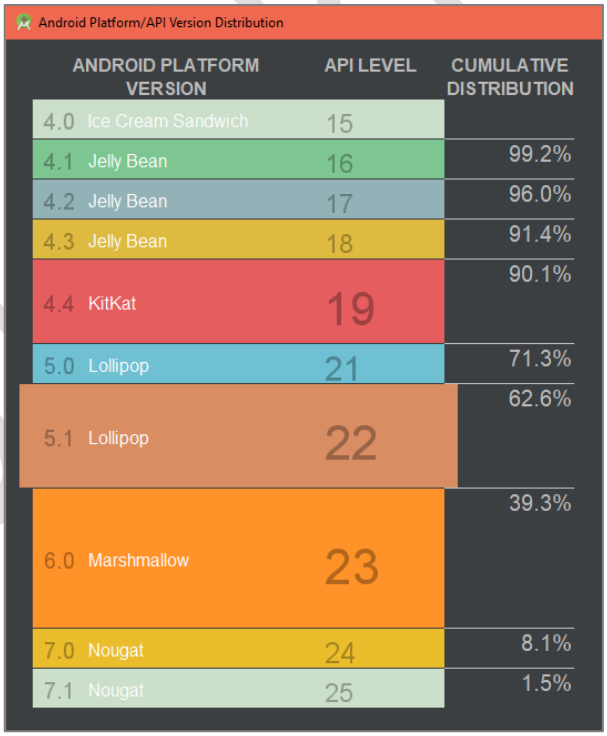
1.5 – Target platform

After doing some research I have decided that the target platform for the app will be Android, this is because the client has an Android powered smartphone and so do I, so I will be able to create and test the app on my phone. In addition, the software development kit for Android application creation, Android Studio, is free to use and runs well on my computer – I know this as I have created some very basic apps using this software before.

Android Studio was chosen to create the app, not only because it is the preferred and most recognised choice for creating Android apps, but because it incorporates multiple languages together to provide a single output of an app. To create a complex app, such as the one I hope to create, it combines object-oriented principles of Java together with XML. More information on these languages is in the next section below.

As with all operating systems, Android is continually developing and updating. Therefore, I have to build my app for a platform version that is newer, so that its features or ways of coding are not obsolete within a year. The latest version I can build for using Android Studio is 7.1 – Nougat and the oldest version is 4.0 – Ice Cream Sandwich (Android uses confectionery names to label version releases). The difference between these two may seem quite small, only being 3 versions – though there are 10 incremental releases between them. To make things more difficult, not every device that is powered by Android is running the latest version, some are still running on the oldest release I can build for. Helpfully, Android Studio provides a chart with a cumulative percentage of devices that are running each release (API level).

Based on this chart I will be using Android version 5.1 – Lollipop (API level 22) to create my app as this covers just over 60% of Android devices on the market. Including my client's device. This enables me to use all the features of 5.0 but use a newer release. Newer releases within versions are better for many reasons, the main ones being improved security measures and efficiency of hardware resources on the device.



My smartphone runs version 6.0, so creating the app in an older version will enable me to test it to make sure it works on future versions. If it successfully works on a later release, then it will most definitely work on an older release (hardware exceptions from different manufacturers do sometimes throw in different complications, leading to varied results across devices).

1.6 – Languages required

After doing some research, I have found out that I will be writing the layout for the app in XML and handling any user interaction or functions in Java. In addition to this, I will need to source an API that lets me access up-to-date currency rates that provide a JSON response that I can parse once submitting a HTTP request. Furthermore, because I want to store permanent data on the device, I can use Android's inbuilt database functionality and use SQLite for data storage.

Java is the official language of Android development and is fully supported by Android Studio, and although I have not used it before I have experience in C# both at a basic level and at an object-oriented level. By applying principles from this language, Java should not be too difficult to pick up. Using object-oriented aspects of Java will be required for my project as the API will be bringing in objects that will have multiple properties attached to them (e.g. exchange rates, currencies etc.).

These objects can then be instantiated within a class and have specific methods to access each part of the object within the class, for instance the exchange rate. In addition, object-oriented principles such as encapsulation and abstraction can be used to make sure that certain parts of the data or program aren't accessible to other parts of the program or user.

In addition to this, XML is very similar to HTML (another language that I do have experience coding with) in that they are both Markup languages (as seen in both of the acronyms: "…ML") and use tags to wrap information. The two languages are different, but like Java and C# I can apply principles that I know from HTML to make learning XML easier.

Submitting a HTTP request and using an API are not things that I have done before but should not require a specific language, as Java has functionality within it to deal with networking and the API is only providing a service to access a set of data (as mentioned in section 1.8 below). In addition, despite JSON looking like a relatively easy data interchange format to understand, it will only be provided by the API and I will not need to code with it myself, just be able to parse it to obtain results.

Finally, SQLite is something that I have a little bit of experience in, and so after recapping the basics it should be familiar to me again.

1.7 – Currently on the market

Searching the Google Play Store (where Android users get their apps) for "currency converter" and filtering to show free, 4-star plus rated apps the top three are "Easy Currency Converter", "XE Currency" and "Currency Converter Plus Free". These apps provide users with features such as:

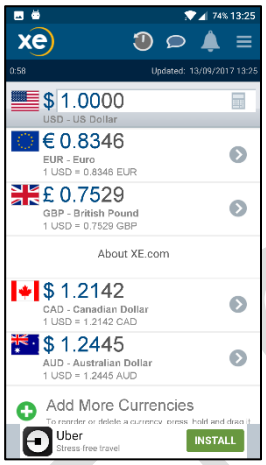
- Update frequencies as short as 1 minute
- Support for up to 5 decimal places on conversions
- 180+ available currencies
- Support for favourite currencies
- Charts showing growth and decline of the currencies
- Offline mode
- Support for multiple currencies at once
- Support to change the base currency

Some of these features I may be able to implement into my app, but my main focus will be to try and get the basics working – as outlined in section 1.3.

1.7.1 – Critiquing the on-market apps

After using the on-market apps for a while I realised that the user experience was quite different for each of them. “XE Currency” felt, in my opinion, cluttered compared to the other two, though it did have a cleaner (more modern) theme. Whereas, “Currency Converter Plus Free” had a very minimalistic interface with large buttons and controls and “Easy Currency Converter” was at a middle-ground between those two.

Easy Currency Converter



XE Currency

Currency Converter Plus Free

A YouTube video that I created shows the use and features of “Easy Currency Converter” and can be found here: http://bit.ly/EasyCurrencyConverter (Make sure to turn on captions).

1.8 – Data source and destination

The get the data for the conversion rates, I require an API to access the results through as this is the most efficient way of obtaining the data from a HTTP request (as opposed to HTML Web Scraping). The two main APIs that I found, currencylayer[1] & Open Exchange Rates[2], were full of features and had exactly what I needed available in the free package. However, both had the base conversion currency set to US dollars, and to change it you had to pay. This was an inconvenience I didn’t want to have to work around so I continued searching. In addition, the free packages had a limit on how many requests to the API you could make a month, something not ideal if lots of testing is required.

1.8.1 – The chosen API

Eventually, I found an API called Fixer.io[3] that uses the European Central Bank as its data source[4] and has everything I need for free. It allows the user to choose the base conversion currency and has unlimited API requests. The only downside is that the conversion rates are updated daily, not hourly/by the minute like the other APIs, however for my target audience and client this will be fine.

[1] https://currencylayer.com/
[2] https://openexchangerates.org/
[3] http://fixer.io/
[4] https://goo.gl/32zdH1

1.8.2 – The data destination

Once the data has been retrieved from the API, it will be stored in a SQLite table within the app, this will allow the client to use the app offline and keep network requests coming from their device to a minimum.

1.9 – Target audience

Though the app is being created as a bespoke solution for my client, there is no reason why other users could not use it. As mentioned in section 1.1, the target audience for my app would be tourists going on holiday abroad. The intended age group for my app would be adults, though anyone of any age could use it. My app would not be suitable for people with a job in global market shares, or someone who used the app because they travelled a lot in the same day. This is because the API that I have found only updates on a daily basis, though this is fine because tourists do not need to know if their amount is changing throughout the day.

1.10 – Feasibility of project

In order to create the app, I will be required to use Android Studio. To help me with this and the learning of the advanced concepts that the app will contain I am going to use Udacity[5] an online teaching resource. I will be following the free courses[6] running parallel to their paid Nanodegree "Android Basics by Google". Within these courses, they will teach me the following concepts:

- User Interface
- User Input
- Multiscreen Apps
- Networking
- Data Storage

These are all the things that I require for my app and the courses may contain elements that I don't require, but I can skip those parts.

In addition, I will use Stack Overflow[7] for questions I have on more specific topics, things that are not covered in the courses or for when errors stop my progress. Most likely, people in the past have had the same or similar problems, so I can use these forums to help me.

Furthermore, Google has published a Material Design guideline[8], for how apps should be created so that all apps give the user a similar feel (e.g. placement of menus within app, how certain buttons should act) so that it is easier for the user when moving between apps (i.e. they don't have to think about using the app). It includes sections such as style, layout, components and patterns amongst many other things. I will be using it as a reference when creating my app so that I can adhere to these material guidelines.

1.11 – Android conventions

When creating an Android app, there are conventions that are best to follow, some are quite basic but can vastly improve the app and others are quite small. I will be trying to stick to them as much as possible when creating my app.

[5] https://eu.udacity.com/
[6] https://goo.gl/3sjze8
[7] https://stackoverflow.com/
[8] https://material.io/guidelines/

1.11.1 – Variable names

All variables must be written in camelCase and have a suitable description of what they do to make referencing them easier.

1.11.2 – Text strings & colour codes

In Android, it is considered a bad practice to hardcode any values that don't have to be, therefore most values are referenced from specific XML resource files. This allows for any text values to be updated in one place in one file and colour codes in another file. This allows for the re-use of values multiple times throughout the app without data redundancy and easy translation to other languages for text strings.

1.11.2.1 – My own conventions

So that I can reference the strings easier, I will give each category of string a prefix to its name. This will act as a filter when the auto-complete pops-up when coding. For instance, buttons may get the prefix "button_" so an example button string variable name would be "button_savedConversions".

1.11.3 – File names

So that XML and Java files are not confused, as they can sometimes have the same name, they are named in different ways. For instance, an XML file may be called "activity_main.xml" but the Java file may be named "MainActivity.java" that way I know that all those with underscores and no capitals must be XML files when the file extensions are not shown.

1.11.4 – Comments & methods

Comments make it easier to follow the code by adding little details of what is happening at each line of code and by setting out the code using methods (i.e. functions built into classes within the object-oriented paradigm) where necessary allows for more efficient code.

2.0 – Documented Design

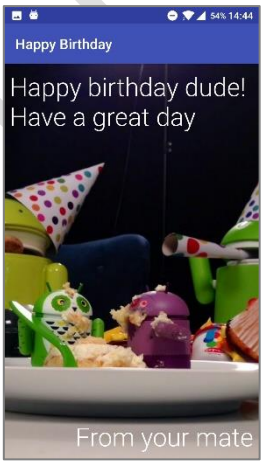
2.1 – Apps created in order to test concepts

Before starting to create or design the main app for my project, I wanted to make sure that the key concepts that I had in mind for it were aspirational to achieve for the project. Therefore, as mentioned in section 1.10, I used the website Udacity to teach myself how to code the concepts of my app as well as to learn how to code in Android Studio so that if anything was not achievable I could re-plan my project. While undertaking the courses, I created many apps that contain the concepts that my project will require. Some of these apps were given to me via the course, with only the core layout so I had to complete the rest of the app, whereas others I created myself from the beginning. Listed below are the apps and what, in creating them, I learned and found out.

The code for each of the apps can be found in the appendices.

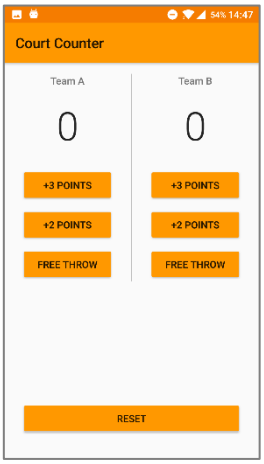
2.1.1 – "Happy Birthday"

This was a very basic, static, app that the user had no functionality with. It was used to show how to layout visual elements on the screen for the user and it covered the basics of different types of views, positioning them and the attributes they had.



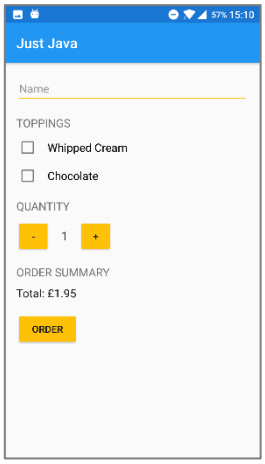
2.1.2 – "Court Counter"

This was a slightly more advanced app that showed me how to use simple Java and work with local and global variables. It also introduced styling views using fonts and colours as well as producing a more visually polished app. The context of this app was to provide a score counter for when refereeing for a sport.

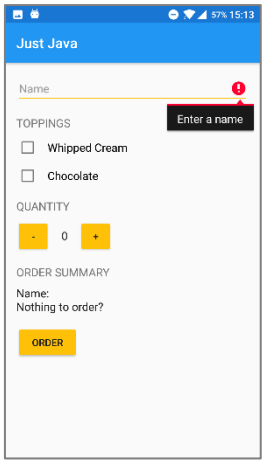


2.1.3 – "Just Java"

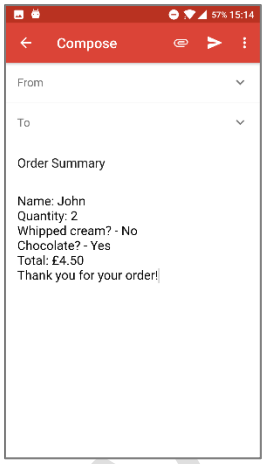
This was a much more advanced app than Court Counter, it used user inputs of textboxes, checkboxes and increment/decrement buttons. In addition, it manipulated entered data to give the user a result (in the context of this app, a price for a number of coffees). Furthermore, I experimented with data validation and prompts. In addition to this, the app used intents (the Android term for using other apps with data from your app) to create an email with the order summary in.



Main screen

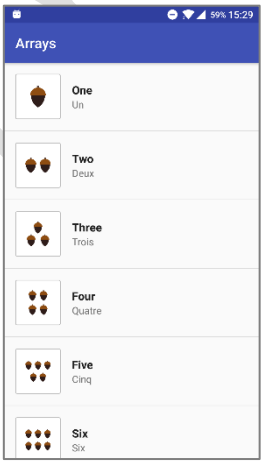


Data validation / prompts



Email intent opened

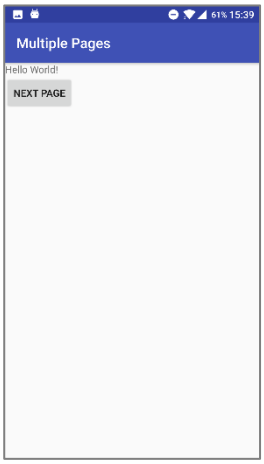
2.1.4 – "Arrays"



This app was used to show how a list of items could be dynamically displayed using a ListView. The dynamic part is important, because it means that you don't have to code a view for each list item – the app automatically detects how many to create based on the size of the array. In addition, if there was a list of 100 items, generating all those views at once would consume a lot of device resources, therefore views are recycled – so that only views visible to the user on screen are populated with data/generated. This allows the array to be updated while the app is running and for the ListView to change also. The context for this app is to provide a list of translations for learning another language. Another thing this app showed was how to implement more than one element into each list item.

2.1.5 – "Multiple Pages"

This app is basic, but utilises the intent function to open another page in the app. After the user has navigated there, they can return to the previous page using the 'up arrow'. The purpose of this app was to work out how, on the most basic level, a user could visit different pages within the app.

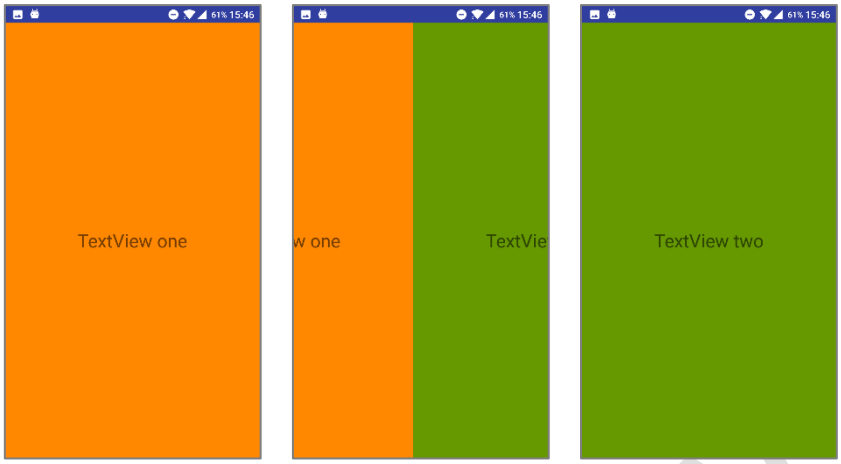


Page 1

Page 2

2.1.6 – "Swipable Pages"

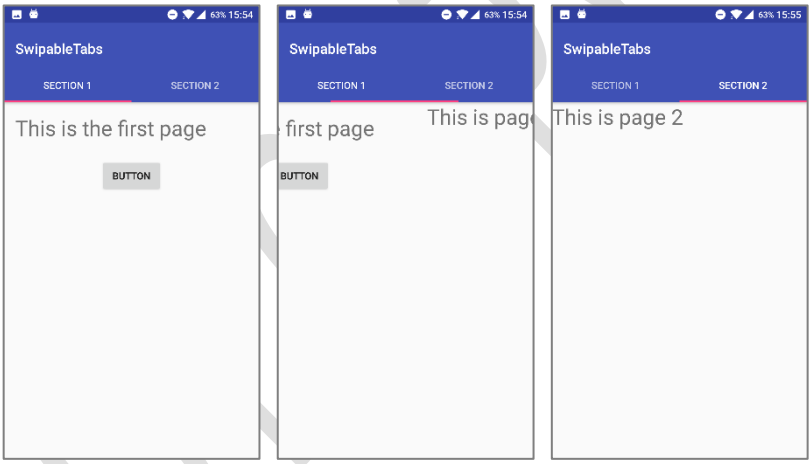
This app was an upgrade to the previous one, and took advantage of user gestures to control the navigation between pages. However, there were no labels outside of what the content showed to let the user know which page they were on.



Page 1 *Page transitioning* *Page 2*

2.1.7 – "Swipable Tabs"

This was another upgrade on the previous app, it implemented a navigation bar into the header of the app, so that the user could click or swipe on which page they require, while giving the feel that the app is all on one page.



Page 1 *Page transitioning* *Page 2*

2.1.8 – "Dropdown boxes"

Part of my main project will require the use of dropdown boxes, so this app shows how a dropdown box can contain a list of items, and then upon selection of one of them – update data elsewhere in the app. Standard dropdown boxes do not look like this, this app also introduces styling a dropdown box.

2.1.9 – "Soonami" and "Did You Feel It?"

These apps were provided by the course I was learning from to show how network retrieval of JSON data through an API is accomplished in Android. They both use a URL which has parameters to specify to the API what data they want returned. Soonami provides the user with an earthquake event, and whether or not it caused a tsunami. Did you feel it, provides the user with an earthquake event and how many people felt it (data obtained through the API).

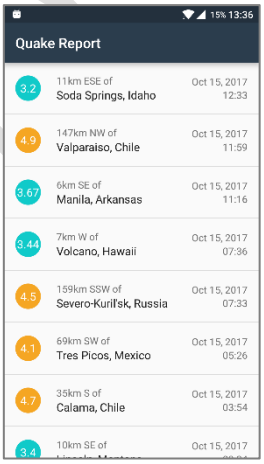
Soonami

Did You Feel It?

2.1.10 – "Quake Report"

This app was also provided by the course I was learning from, though it was only half-completed, so I had to finish the rest of it while learning.

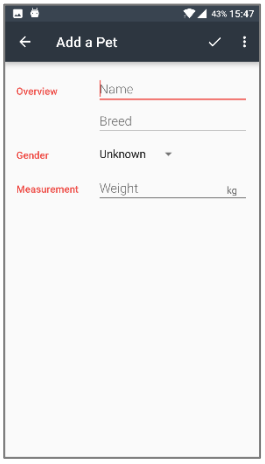
This app was an upgrade to the previous two, whereas they obtained data and displayed one result, this app displays multiple results – this required a more complex array and organising of the data once it had been parsed. In addition, the user can click on the items in the ListView to open information about each list item (using an intent). Also, the magnitude circles on the left automatically change colour based on the severity of the earthquake.



2.1.11 – "Pets"

This app was also provided by the course I was learning from, though it was only half-completed, so I had to finish the rest of it while learning.
The purpose of this app was to learn how SQL is implemented into an Android framework, and how the app can use a database to store and access data.

The main page of the app



The page for adding a new pet to the database

2.2 – Prototype development

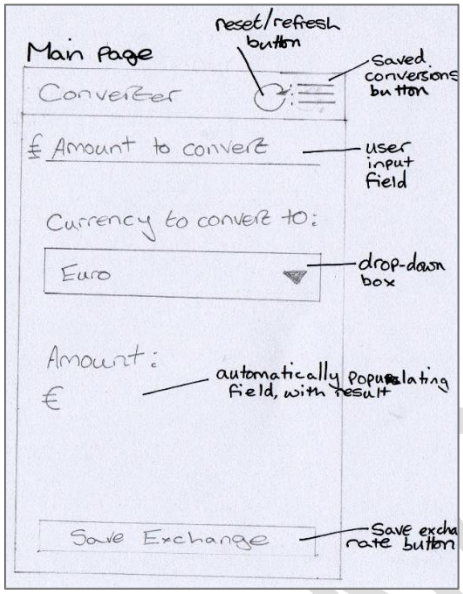
This prototype has been created using the basic objectives in section 1.4, to get a feel of how the app will look on a device. It has also been created so that my client can critique it and help improve it to what they require.

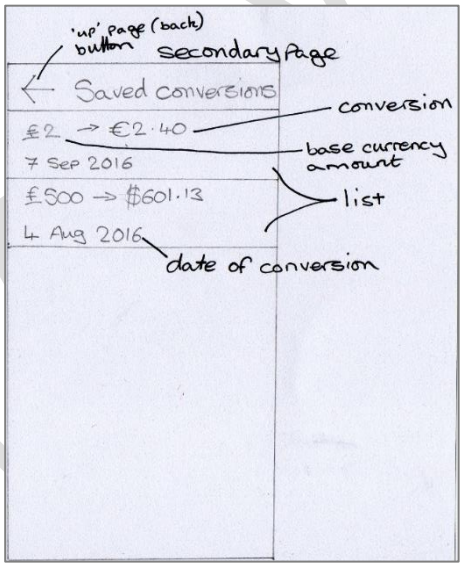
2.2.1 – Design

This is the process where I planned certain elements of the app and how different pages would look or how they were structured.

2.2.1.1 – Sketches

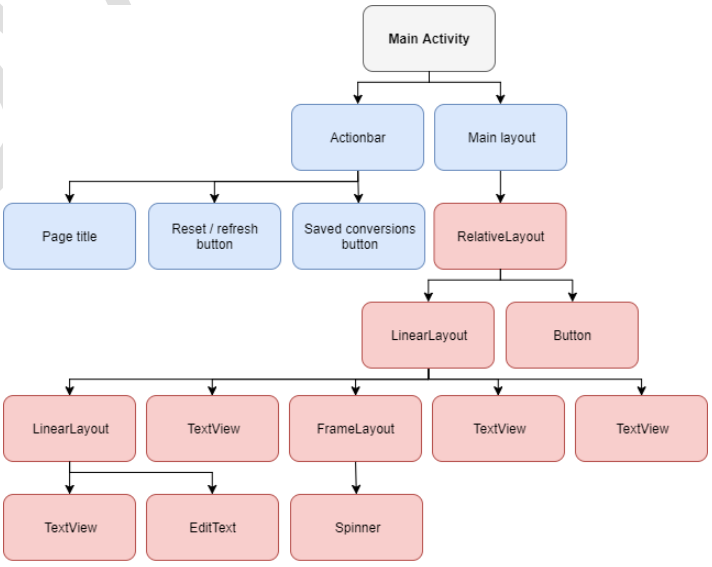
These sketches, which were drafted before any coding occurred, show the initial layout of the app, they include the main and secondary pages.





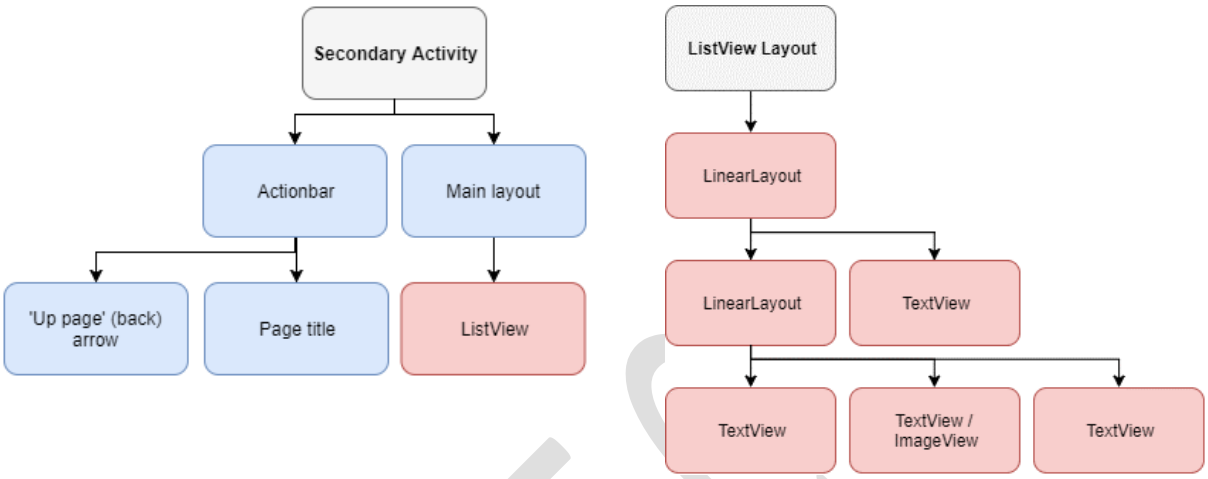
2.2.1.2 – Hierarchy charts

These charts show the break-down of the structure that the two pages will require. The action bar is created in a different way to the main part of the page layout, so the blue represents theoretical ideas or components as I do not know how to code them yet, whereas the red represents the actual structure of the XML code.



A RelativeLayout allows for objects to be positioned relative to their parent (the next level up on the hierarchy chart) or to other objects with the same parent. A LinearLayout positions each object next to the other, either horizontally or vertically.

Android uses the ListView as a parent view and each item is given a set layout style, this means that the array is dynamic and can be changed at any time, even when the app is running because the number of items is not hardcoded. This results in the layout code for the secondary page to be quite simple.



2.2.1.3 – Initial class diagrams

Because I only have the above sketches and hierarchy charts, I cannot fully create any class charts at the moment as I do not know what classes I will need. However, from my previous use of Android Studio I know that when a new project is created, classes are created for any activities that are created. This means that I will have at least two classes: MainActivity.java and SecondaryActivity.java – if I name the classes by the names given in the grey rectangle in the hierarchy charts – that will control the code on each of those respective pages.

Update: *After coding section 2.3, where more classes were involved, I was able to create a proper UML class diagram showing all of the classes involved in the Currency API Test app – this can be seen in section 2.3.5. In addition, after completing the final app in section 2.5, I was able to again create a proper UML class diagram for the final Currency Converter app – this shown in section 2.8.1.*

2.2.2 – Coding

As mentioned above, the app has been initially coded based on the CLIENT REQUIREMENTS in section 1.4, for the client to then critique it later on. Shown below is the process of this initial creation and the code for this app can be found in appendix 13.

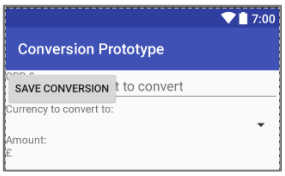
During the coding of this section and in later sections I will be required to override some methods. This allows a class to inherit a method from a superclass where the behaviour is "close enough" to what is required and then be modified as needed. The overridden method has the same name, number and type of parameters, and return type as the method that it overrides.[9] This will allow me to use methods provided by the superclass and adapt them to my need.

[9] https://goo.gl/mpjSo3

2.2.2.1 – Main Activity

This is where I began to create the main page of the app, the first view that the user would see when they loaded the app.

2.2.2.1.1 – A basic layout is created



To begin with, the basic elements for the page were implemented, referencing from the hierarchy chart and sketch. They had no styling or padding attributed to them, resulting in a cramped, conflicting layout. In addition, there were other issues or features missing including setting spinner (drop-down list) options and the action bar functionality.

2.2.2.1.2 – Visual layout improvements

The elements are given padding and the button is positioned correctly. This was done using the attributes of layout margin and aligning objects relative to the parent. This being the reason why the root layout was chosen to be RelativeLayout, so the button could sit apart from the rest of the objects.

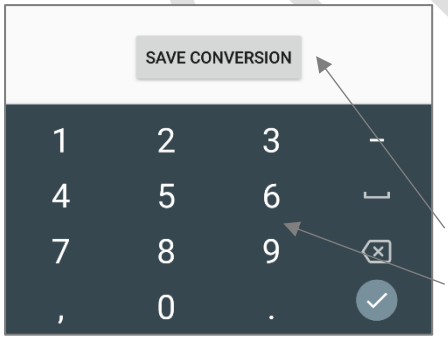
```
android:layout_alignParentBottom="true"
android:layout_centerInParent="true"
```

```
android:layout_marginBottom="8dp"
android:layout_marginLeft="16dp"
android:layout_marginRight="16dp"
android:layout_marginTop="8dp"
```

Note the units in the above screenshot. They are “dp” (pronounced “dips”) or density-independent pixels in comparison to normal pixels (“px”). These are flexible units that scale to uniform dimensions on any screen regardless of the density. This makes sure that a button on a lower density screen appears the same size as on the higher density screen.

2.2.2.1.3 – User-entry field

To stop the user entering data they shouldn’t, a keyboard type is specified. In addition, the behaviour of the keyboard is set so that it remains hidden until prompted to open and that it resizes the layout to keep most of the app functionality. By specifying the input type to be “numberDecimal” this helps to fulfil part of **CLIENT REQUIREMENT 3**, by allowing decimal numbers.




```
android:windowSoftInputMode="stateAlwaysHidden|adjustResize">
android:inputType="numberDecimal" />
```

The button is moved up when the keyboard is opened.

Number-only keypad, symbols (except decimal point) and letters do not enter.

2.2.2.1.4 – Action bar is created

To create the icons on the action bar for **CLIENT REQUIREMENT 9**, a menu file was created, this holds information about each icon such as what it’s icon image should be, it’s title and whether it should appear in the drop-down list or as a button.

Following this, Java code has to be implemented to create the action bar during runtime of the page.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <!--Refresh icon, retrieves latest conversion figures-->
    <item
        android:id="@+id/action_refresh"
        android:icon="@drawable/ic_refresh_white_24dp"
        android:title="Update conversion rates"
        app:showAsAction="ifRoom" />

    <!--"Saved Conversions" page navigation button, should appear
    <item
        android:id="@+id/action_conversionNav"
        android:icon="@drawable/ic_list_white_24dp"
        android:title="Saved conversions"
        app:showAsAction="ifRoom" />
```

```
/**
 * Create the options menu in the action bar
 * using {@param menu} as a resource
 */
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.main_menu, menu);
    return true;
}
```

To obtain the graphics for the buttons, I used the Google Design: Material icons resource page.[10] This results in the action bar looking like the above.

2.2.2.1.5 – Page title is changed

As shown in the above image, the title of the main page (which is taken from the app name) is too long for the action bar with the icons, so a more concise name needs to be assigned to help fit with **CLIENT REQUIREMENT 11**.

Usually, to name an activity all you have to do is state it as an attribute in the Manifest file, like so:

```
<activity
    android:name=".SecondActivity"
    android:label="@string/page_2"
    android:parentActivityName=".MainActivity" />
```

However, the main page of the app doesn't like to follow this convention and whatever value is set in the manifest file for the main activity will be the name of the app on the home screen. Because the longer name is wanted for the home screen, the page title has to be dynamically set in the Java code in the OnCreate method.

```
//Set the title of the title bar to be more appropriate
setTitle("Converter");
```

2.2.2.1.6 – Spinner is given style

At the moment, the spinner is very basic looking and there is no indication to the user where to click, other than the drop-down arrow.

To get around this, a XML drawable file is created. This contains styling information that can be scaled to any size. This is then set as the background of the FrameLayout. If it was set as the background on the actual spinner, it would not size correctly.

```
<FrameLayout
    android:id="@+id/spinnerBox"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:background="@drawable/bg_spinner">

    <Spinner
        android:id="@+id/currency_picker"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</FrameLayout>
```

[10] https://material.io/icons/

```xml
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:bottom="8dp"
        android:top="8dp">
        <shape>
            <!--Use this as the background colour of the Spinner-->
            <solid android:color="@android:color/white" />

            <!--This controls how rounded the corners are-->
            <corners android:radius="2dp" />

            <!--This creates the border width and colour-->
            <stroke
                android:width="1dp"
                android:color="@color/divider" />

            <!--Use this to control how large the box should be-->
            <!--'Top/Bottom' control height, should be equal for text to be centered-->
            <!--'Left' controls how far right the text is pushed-->
            <!--'Right' controls how far left the arrow is pushed-->
            <!--NOTE: with no right padding, the drop-arrow will resume default padding-->
            <padding
                android:bottom="16dp"
                android:left="8dp"
                android:right="8dp"
                android:top="16dp" />
        </shape>
    </item>
</layer-list>
```

2.2.2.1.7 – Spinner options created

To populate the spinner with options, an array has to be created in the strings file. The order in which the elements appear there, are the order in which they appear in the spinner.

```xml
<!--Array for the list of items in the spinner-->
<string-array name="array_currencies">
    <item>European Euro</item>
    <item>US Dollar</item>
    <item>Japanese Yen</item>
</string-array>
```

For now, this simple array is being used to populate the spinner, but later on countries and flags will be shown in it.

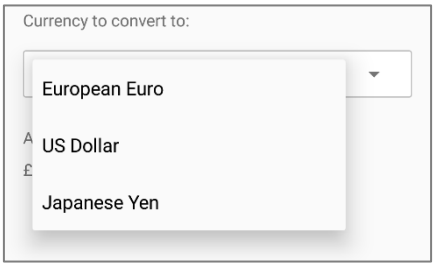
Following this, the spinner has to be created through Java – called from the OnCreate method.

```java
/**
 * The method that creates the spinner options
 */
public void createSpinnerOptions() {
    //Create the spinner as an object
    Spinner spinner = (Spinner) findViewById(R.id.currency_picker);

    // Create an ArrayAdapter using the the array of options and the style of the text before selection layout
    ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this, R.array.array_currencies, R.layout.spinner_item);

    // Specify the layout to use when the list of choices appears
    adapter.setDropDownViewResource(R.layout.spinner_dropdown_item);

    // Apply the adapter to the spinner
    spinner.setAdapter(adapter);
}
```

```xml
<!--This XML layout file controls the styling for the text once selected from the spinner-->
<!--Using a TextView works fine for this XML file-->
<!--Make sure that attribute 'id' is included-->
<!--Everything else can be customised-->
<!--Keep height to 'wrap_content' to keep the height defined by the background XML-->
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/text1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="14sp" />
```

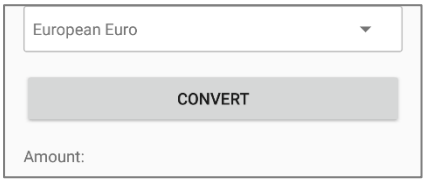
The two layouts, ‘spinner_item’ & ‘spinner_dropdown_item’ are XML layout files that specify the style of the text when the spinner is unselected or selected.

```xml
<?xml version="1.0" encoding="utf-8"?>
<!--This XML layout file controls the styling for the text inside the dropdown menu-->
<!--Use a CheckedTextView for this XML file-->
<!--Make sure that attributes 'id, style, ellipsize, maxLines' are all included-->
<!--Everything else can be customised-->
<!--48dp is the recommended height for a button/selection-->
<CheckedTextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/text1"
    style="?android:attr/spinnerDropDownItemStyle"
    android:layout_width="match_parent"
    android:layout_height="48dp"
    android:ellipsize="marquee"
    android:maxLines="1" />
```



2.2.2.1.8 – Convert button is added

A convert button is added to make ease-of-usability greater. The conversions (in the future) will occur automatically, but there should still be a convert button to allow the user to manually convert the values.




```
<Button
    android:id="@+id/button_convert"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="8dp"
    android:text="@string/button_convert" />
```

The convert button takes the value from the user-entry field and based on the value in the spinner, displays the correct currency and the amount the user entered. Future versions will have actual conversions.

```
//Create a new button object constructor
Button button = (Button) findViewById(R.id.button_convert);
//Set a listener to it and create a new method
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        //Do this when the button is clicked
          Log.i("TEST MESSAGE:", "Convert button click was registered");
        onConvertClick();
    }
});
```

This code shows what is called a 'listener' in the OnCreate method that listens for the click of the specified button.

```
/**
 * Custom method that handles what happens when the convert button is clicked
 */
private void onConvertClick() {
    //Get the value of the current spinner option
      Log.i("TEST MESSAGE:", "onConvertClick was called");
    String currentSpinnerOption = getSpinnerOption();
    //Get the relevant symbol for the currency
    String relevantCurrencySymbol = getCurrencySymbol(currentSpinnerOption);

    //Change the text to display the correct values
    setResultFieldText(relevantCurrencySymbol);
}
```

This code shows the method that the listener calls when the button is clicked.

It gets the current spinner option and then uses that value to determine the correct currency symbol. Following this, it sets the TextView to the correct values. It uses parameter passing and return statements to make the code easier to follow.

```
/**
 * Custom method
 *
 * @return the currently selected option of the spinner
 */
private String getSpinnerOption() {
      Log.i("TEST MESSAGE:", "getSpinnerOption was called");
    //Create a spinner object constructor
    Spinner spinner = (Spinner) findViewById(R.id.currency_picker);
    //Get the current item that is selected in the spinner
      Log.i("TEST MESSAGE:", "Value to return: " + spinner.getSelectedItem().toString());
    return spinner.getSelectedItem().toString();
}
```

This code shows how the current spinner option is retrieved.

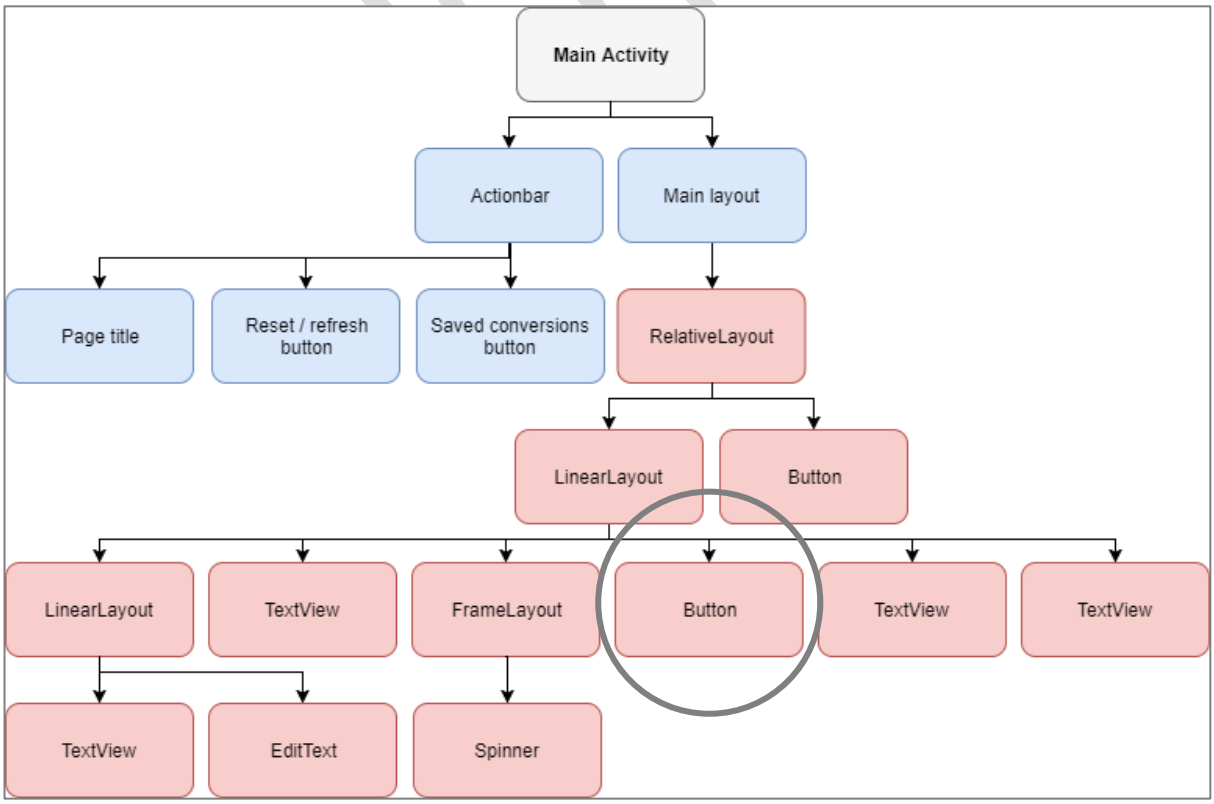
```
private String getCurrencySymbol(String country) {
    //Use switch statement to determine the correct symbol for the
    // currency and return it
    switch (country) {
        //Euros
        case "European Euro":
            return "€";

        //US Dollars
        case "US Dollar":
            return "$";

        //Japanese Yen
        case "Japanese Yen":
            return "¥";

        default:
            return "£";
    }
}
```

This code shows how, using a switch statement, the correct currency symbol is obtained.

```
/**
 * Custom method that outputs the result from the conversion
 *
 * @param relevantCurrencySymbol input of the relevant currency symbol to use
 */
private void setResultFieldText(String relevantCurrencySymbol) {
    //Create an object constructor for the text-field that is changed
    TextView conversionResultField = (TextView) findViewById(R.id.conversionResultField);

    //Create an object constructor for the EditText
    EditText amountToConvertEntryField = (EditText) findViewById(R.id.amountToConvertEntryField);
    //Get the values from the EditText
    String userEnteredQuantity = amountToConvertEntryField.getText().toString();

    //TODO: implement proper calculations into where the text-field is changed
    //Set the text-field text
    conversionResultField.setText(relevantCurrencySymbol + userEnteredQuantity);
}
```

This code shows how the result field is updated.

Now that another element has been added to the layout, the structure hierarchy chart needs to be updated so it reflects the new layout (circled).



2.2.2.1.9 – Spinner advancement

Currently, when there is a value in the user-entry field and a different currency is selected the user has to click the convert button for the currency to change symbol. To make it so that this is automatic, another listener can be set – this time on the spinner.

This code in OnCreate initialises the spinner listener.

```
//Get the spinner as an object and call the OnItemSelectedListener on it to create the listener
Spinner spinner = (Spinner) findViewById(R.id.currency_picker);
spinner.setOnItemSelectedListener(new MyOnItemSelectedListener());
```

This code is the listener, there are two methods within the class because the spinner can be selected or not selected.

```
/**
 * This sub-class controls what happens when the listener detects something has been selected
 */
private class MyOnItemSelectedListener implements AdapterView.OnItemSelectedListener {
    public void onItemSelected(AdapterView<?> parent, View view, int pos, long id) {
        //When an item is selected, do this:
        onConvertClick();
    }

    public void onNothingSelected(AdapterView parent) {
        // Do nothing when nothing is selected.
    }
}
```

2.2.2.1.10 – Action bar icons are given uses

To handle the action bar clicks a method has to be overridden from the superclass. This uses a switch statement based on each menu item.

```
/**
 * Handle item selections of the action bar
 */
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    //Switch statement
    switch (item.getItemId()) {
        //When the list icon is selected:
        case R.id.action_conversionNav:
            //Open the second page
            changeToSavedConversions();
            return true;

        //When the refresh icon is selected:
        case R.id.action_refresh:
            //Refresh the app
            refreshApp();
            return true;

        default:
            // The user's action was not recognized.
            // Invoke the superclass to handle it.
            return super.onOptionsItemSelected(item);
    }
}
```

```
/**
 * Custom method, refresh the conversion rates and reset the changed fields
 */
private void refreshApp() {
    //Create an object constructor for the EditText
    EditText amountToConvertEntryField = (EditText) findViewById(R.id.amountToConvertEntryField);
    //Set the field to be empty
    amountToConvertEntryField.setText(null);

    //Create a spinner object constructor
    Spinner spinner = (Spinner) findViewById(R.id.currency_picker);
    //Reset the value to Euro (default)
    spinner.setSelection(0);

    //Reset the values in the results field
    setResultFieldText(getResources().getString(R.string.symbol_EUR));

    //TODO: Implement refreshing of conversion rates
}
```

Reset/refresh

This method clears the user-entry field, resets the spinner and resets the result field. Future versions will have this method also update the conversion rates.

Saved Conversions

```
/**
 * Method that navigates the user to the second page, using an intent
 */
private void changeToSavedConversions() {
    //Create a new Intent object constructor, populate it with SecondActivity.class
    Intent intent = new Intent(this, SecondActivity.class);
    //Start that new intent
    startActivity(intent);
}
```

This method creates an intent (what Android uses to change instances of pages or apps) of the second page and launches it.

By implementing these two methods, CLIENT REQUIREMENT 9 is fulfilled.

2.2.2.2 – Secondary activity

The second activity, saved conversions, is navigated to by the user via the action bar icon on the main activity.

2.2.2.2.1 – Return navigation and title

For the user to have a clear idea of how to return to the previous page – as opposed to pressing the back button on the phone – an 'up arrow' is required, this is simply declared in the manifest as which activity the action should return the user to.

As mentioned above, to change the title of an activity (not being the main activity) a value is declared in the manifest file.

```
<activity
    android:name=".SecondActivity"
    android:label="Saved Conversions"
    android:parentActivityName=".MainActivity" />
```

This results in the action bar looking like this.

2.2.2.2.2 – Setting up the ArrayList

Android uses a ListView as the best solution to present an array of items to the user and it uses an adapter to help with memory optimisation.

2.2.2.2.2.1 – The main activity layout

In the second activity layout file (saved conversions page) a ListView is set so that Android knows what type of content to display on that page.

```
<?xml version="1.0" encoding="utf-8"?>
<ListView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/root_list_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.example.android.conversionprototype.SecondActivity" />
```

2.2.2.2.2.2 – The view that the list contents will take

Currently, the ArrayList only has support for a single TextView, this can be changed by defining a custom element layout for it. This is an XML layout file that each element of the ArrayList will populate its data into. This layout is designed from the structure hierarchy chart in section 2.2.1.2.


```
tools:text="£2" />
```

The items of text here are not hard-coded, they are placeholder text from a tool, that lets you visualise what the layout will look like while you are coding the rest of the layout.

2.2.2.2.2.3 – The ArrayList class

To provide methods and variables for each part of the ArrayList (i.e. base amount, converted amount, date) an additional class has to be created where methods can be used by the other classes to create the ArrayList.

Within the class, private variables are created that each part of the ArrayList will use – they are private because they are not used outside of this class.

```
/**
 * Declare private variables for this class to use
 */
private String mBaseCurrencyValue;
private String mConversionCurrencyValue;
private String mDate;
```

Following this, a constructor is created for the class, it is public as it can be accessed outside of the class and it has input parameters, which will input the current contents of the ArrayList.

```
/**
 * Create the constructor for this class, a constructor creates an instance of a class
 * This constructor will create an instance of two Strings
 */
public SavedCurrencies(String baseCurrencyValue, String conversionCurrencyValue, String date) {
    mBaseCurrencyValue = baseCurrencyValue;
    mConversionCurrencyValue = conversionCurrencyValue;
    mDate = date;
}
```

Finally, methods that return the private variables are created, these are public as they will be accessed from outside this class later on.

```
public String getDate() { return mDate; }
public String getConversionCurrencyValue() { return mConversionCurrencyValue; }
public String getBaseCurrencyValue() { return mBaseCurrencyValue; }
```

2.2.2.2.2.4 – The SecondActivity class

In the second activity, the ArrayList has to be initialised and populated with data. As there are multiple input parameters in the constructor in the ArrayList class, multiple inputs are required here (per item in the ArrayList).

```
//Create the ArrayList, so it can be used with objects
List<SavedCurrencies> savedCurrencies = new ArrayList<>();
//Populate the ArrayList
savedCurrencies.add(new SavedCurrencies("£1", "$1.20", "04 Sep 2015"));
savedCurrencies.add(new SavedCurrencies("£12", "€15,30", "21 Aug 2016"));
savedCurrencies.add(new SavedCurrencies("£510", "¥1683", "14 Jan 2017"));
```

An ArrayList class inherits the same functions as the List class, but with more specifics. The List class is used as the base to give more flexibility and functionality. The data entered here is dummy data, just to show the ArrayList is working as it should be.

Following this, an adapter has to be created for the elements of the ArrayList to use and is responsible for giving a view to each item in the dataset. In addition, an object constructor is created for the ListView, so that the adapter can be set to it.

2.2.2.2.2.5 – The Adapter class

This is the class that populates each view with the correct value when creating the ArrayList.

```
/**
 * A custom constructor
 *
 * @param context        is used to inflate the layout file
 * @param savedCurrencies is the data we want to populate into the lists
 */
public SavedCurrenciesAdapter(Activity context, List<SavedCurrencies> savedCurrencies) {
    // This initialises the ArrayAdapter's internal storage for the context and the list.
    // The second argument is used when the ArrayAdapter is populating a single TextView.
    // Because this is a custom adapter for three TextViews the adapter is not
    // going to use this second argument, so it can be any value.
    super(context, 0, savedCurrencies);
}
```

Firstly, another constructor is created to create the views that are required and populate data into them.

Following this, the *getView()* method is created which provides a view for the adapter. Within this method:

Each data item is retrieved, for the current position that the adapter is creating.

```
//Get the data item for this position
SavedCurrencies savedCurrencies = getItem(position);
```

A check is undertaken to see if the view is being reused (as they can be, due to helping to reduce memory usage when scrolling up or down the list in Android) and if not, the view is initialised.

```
//Check if an existing view is being reused, otherwise inflate the view
if (convertView == null) {
    convertView = LayoutInflater.from(getContext()).inflate(R.layout.custom_list_view, parent, false);
}
```

Object constructors for each of the views in the custom ListView are created and the data is populated into these.

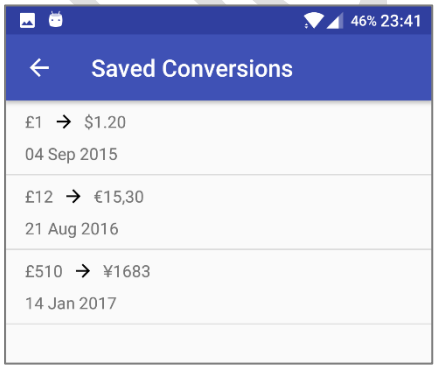
```
TextView dateField = (TextView) convertView.findViewById(R.id.date);

//Populate the data into the template view using the data object
baseCurrencyValueField.setText(savedCurrencies.getBaseCurrencyValue());
```

One of the methods from the ArrayList class.

Finally, the completed view is returned and displayed to the user.

```
//Return the completed view to render on-screen
return convertView;
```

This results in the Saved conversions page looking like this:



2.2.2.3 – Help & About

After finishing the previous two pages of the app, I sat down with my client and showed them what I had produced. One of their suggestions was that a help/about page be implemented for the user. I decided that this was a good idea, and so implemented it. The page will contain content about the app and how to use it.

2.2.2.3.1 – Navigation

As per the theme of this app, other pages are accessed through the action bar on the main activity. The help icon was created by adding another menu item and another case to the switch statement to handle the selection.

```
<!--Help button-->
<item
    android:id="@+id/action_help"
    android:icon="@drawable/ic_help_white_24dp"
    android:title="@string/action_help"
    app:showAsAction="always" />
```

```
//When the help icon is selected:
case R.id.action_help:
    //Open the help page
    changeToHelpPage();
    return true;
```

```
/**
 * Method that navigates the user to the help page, using an intent
 */
private void changeToHelpPage() {
    //Create a new Intent object constructor, populate it with ThirdActivity.class
    Intent intent = new Intent(this, ThirdActivity.class);
    //Start that new intent
    startActivity(intent);
}
```

This results in the main action bar looking like this:

2.2.2.3.2 – Title and return navigation

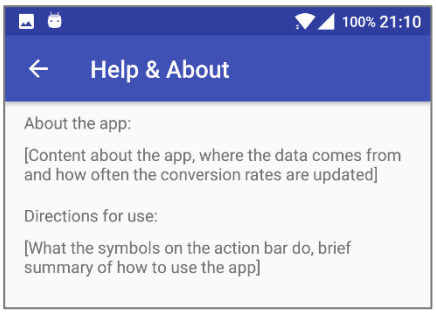
Again, simply declared in the manifest are the title and parent activity.

```
<activity
    android:name=".ThirdActivity"
    android:label="Help & About"
    android:parentActivityName=".MainActivity" />
```

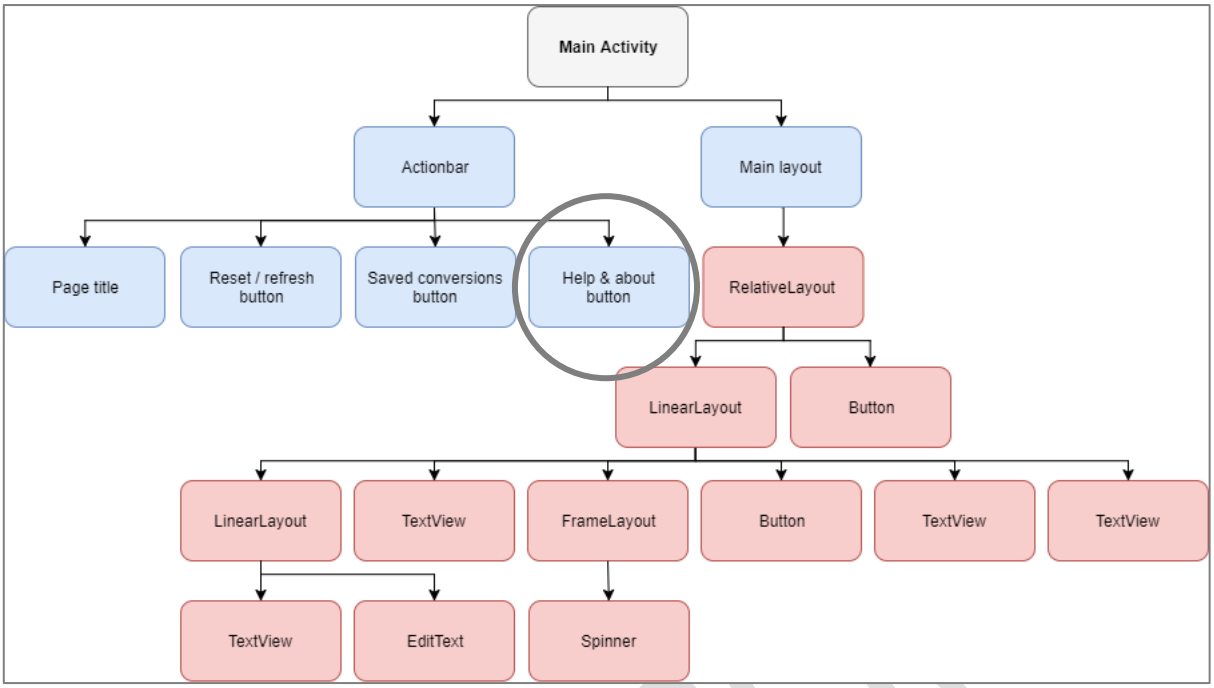
2.2.2.3.3 – Page content

For now, a simple layout consisting of TextViews with place-holder text has been used, this achieved using an XML layout.

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp"
    android:layout_marginTop="8dp"
    android:orientation="vertical">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="About the app:" />

    <!--TODO: Fill out content based on heading-->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="[Content about the app, where
```



Because the structure of the main page has changed, another element has to be added to the structure hierarchy chart (circled).



2.3 – Main focus of the app: API

Up to now, the main app has been designed visually and the basic parts of the app have been tested using the sample apps in section 2.1. However, the main parts of the app are now going to be tested – these are connecting to Fixer.io (the currency API I chose) in this section and getting the SQL to integrate into it in section 2.4.

2.3.1 – Fixer.io

Using the sample app from 2.1.10 I attempted to create an app that connects to Fixer.io, gets the JSON results, and displays them in a list on the app. My process was as follows, and the code for the CurrencyAPITest app can be found in appendix 14.

2.3.1.1 – Currency.java

Because the API returns several variables, including the base currency, exchange date, converted currency and exchange rate, all with different data types – a class is needed to store data about each object being created for the ArrayList. The code below shows the constructor which creates an object when it is called from another class, with parameters being passed in to form the data about the object. In addition, one of the 'getter' methods is shown – these methods return each part of the ArrayList when called by other parts of the app.

```
public Currency(String date, String base, String currencyKey, double currencyValue) {
    Log.d(LOG_TAG, "Constructor - called");
    mDate = date;
    mBase = base;
    mCurrencyKey = currencyKey;
    mCurrencyValue = currencyValue;
}

/**
 * @return the date that is associated with the results
 */
public String getDateOfExchange() {
    return mDate;
}
```

2.3.1.2 – QueryUtils.java

This class contains all of the methods that undertake the detailed instructions on how to connect to the URL, the protocols to follow and the retrieval of the JSON response. Once this had been retrieved, it is then parsed to get the relevant data and added to the ArrayList.

fetchCurrencyData()
This method is the public method that is called from the other classes and it ties all of the other methods together from this class and runs them at the correct times.

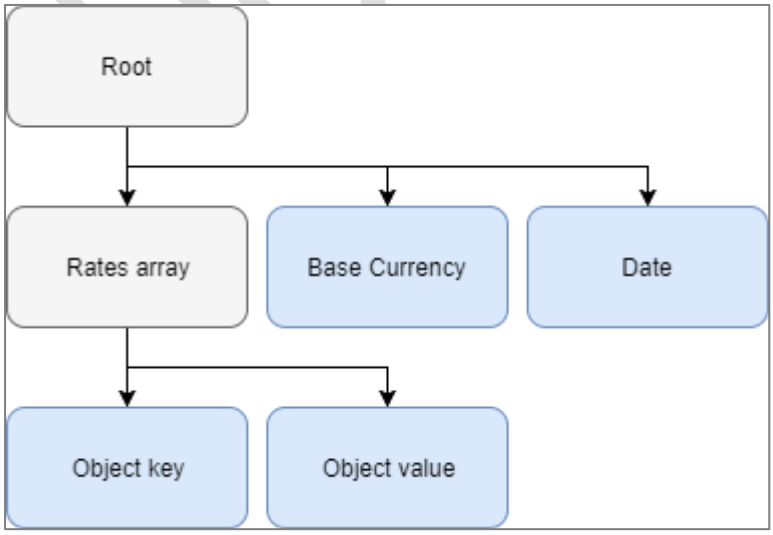
createUrl()
This method takes a string variable and creates a URL object from it, so that the following methods can function properly.

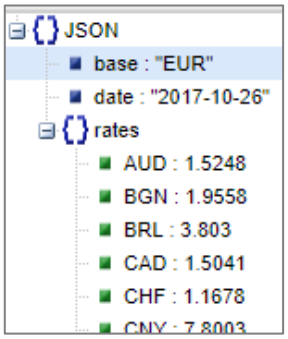
makeHttpRequest()
This method does the following to make a HTTP request:

- Checks that the URL passed in is not null
- Opens the connection to the URL
- Sets timeouts for if an error is thrown
- Sets the request method to “get” (as data is being retrieved from the URL source)
- Makes the connection to the URL
- Monitors response codes, checking for success or failure of the connection
- Upon success, getting the input stream (the data from the URL) and using a Stream Reader to build a string of the JSON results, via *readFromStream()*
- Closing the URL connection and input stream
- Returning the JSON response

readFromStream()
This method takes the content that is on the webpage line by line and converts it from byte characters into one string to form the JSON response.

extractFeaturesFromJson()
This method takes the JSON response in as a parameter and attempts to parse the results. JSON parsing is extracting the required information from the response as a whole. The method checks that the response if not empty, and then parses it.





The screen capture above shows the JSON response in a JSON viewer[11], which allowed me to visualise the levels that the response contained.

The flow diagram above shows the various levels that the response contains, with grey representing objects and arrays, and blue representing values that can be extracted. In this case, the object keys are the country codes, and the object values are the exchange rates. These could have been stored using a HashMap (which is a collection class for storing key and value pairs), but they are instead being stored directly into an object created within the Currency class.

2.3.1.3 – CurrencyLoader.java

This class handles the loading of resource heavy tasks that are in QueryUtils.java and when to start them. This is because in Android, resource heavy tasks have to be executed on a background thread, so that the app UI (user interface) does not freeze, this allows some resources to be dedicated to showing loading icons such as a progress bar. The class has several methods which include a constructor, an overridden method that forces the load to start and an overridden load in background method which returns the obtained data (as an ArrayList) to the method that deals with the load once it has finished (this being in another class).

2.3.1.4 – Activity_main.xml

This layout file is used to structure the layout for the main view, consisting of a ListView, ProgressBar and a TextView which will be used as an empty view. The empty view is responsible for letting the user know if no results have been obtained or there is no Internet connection and can be dynamically changed or hidden to suit the situation.

2.3.1.5 – Element_layout.xml

This layout file is used to structure how each of the array elements look in the list. For the time being, as this is only a test app, to test whether the connection to the API works or not, there is a simple layout of four TextViews: one for the date, one for the base currency, one for the target/destination currency, and one for the exchange value between those two currencies. The screen capture on the right shows this layout using dummy data.

2.3.1.6 – CurrencyAdapter.java

This class handles the creation of the views for the ListView and uses the 'getter' methods from the Currency class to obtain the data. It overrides the method "GetView" from its superclass.

2.3.1.7 – MainActivity.java

This is the main class where everything is tied together and structured, this is also the class that is automatically initiated when the app opens – so all instructions branch from here. In addition, this class holds information for what the loader should do on creation, during loading and after loading.

All of the loader methods in this class are overridden from its superclass. On loader creation, the CurrencyLoader class is run, which initiates the constructor within that class and starts the HTTP request. Then, when the loader has finished, the progress bar is hidden, and the ArrayList (passed to it from the CurrencyLoader class) is added to the adapter. Then, if the loader is reset (e.g. through orientation change) then the adapter is cleared.

[11] http://jsonviewer.stack.hu/

2.3.1.8 – Manifest.xml

This file holds the permissions that grant the app access to use the user's Internet connection to download data when necessary. Without these permissions, the app would not run.

2.3.2 – The problem

I have run into a problem, I have managed to get the app to run, connect to the API and successfully obtain the JSON response. However, it is when parsing the response, that I get this error:

com.example.android.currencyapitest E/QueryUtils: Problems parsing the currency JSON results […] at rates of type org.json.JSONObject cannot be converted to JSONArray

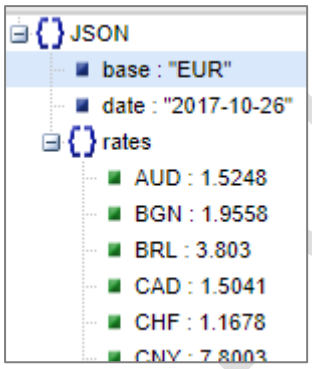
The original error message was much longer, though this is all the information I need. The first part of the message that is highlighted comes from a Log tag I have put in the code. Tracing the error back to this specific message tells me roughly where in the code the error is. Following this, I can use the second part of the error message to find exactly where the issue is.

```
Log.e(LOG_TAG, "Problems parsing the currency JSON results", e);
```

```
JSONArray rates = root.getJSONArray("rates");

for (int i = 0; i < rates.length(); i++) {
    double exchangeRate = rates.getDouble(i);
    currencies.add(new Currency(exchangeDate, baseC
}
```

What it is telling me, is that in my code I have wrongly assumed the datatype of that part of the JSON response to be an Array, when in fact it is an object. That means that I can't use an iteration cycle on it in the same way that I am at the moment.

Having a look at the initial JSON response in a JSON viewer I now see that this would be correct. Because the keys are not numbered within "rates", they are just a series of objects within another object.



2.3.3 – Fixing the problem

It would be possible to parse this section of JSON with the following pseudocode:

arrayKey = "AUD"
exchangeRate = rates.getDouble(arrayKey)
currencies.add(new Currency(x, y, z, exchangeRate)
arrayKey = "BGN"
exchangeRate = rates.getDouble(arrayKey)
currencies.add(new Currency(x, y, z, exchangeRate)

However, completing this for all of the currencies in the JSON would be very inefficient and take up too much code. Therefore, I broke the problem down and wrote a list of what I needed to do to fix this issue:

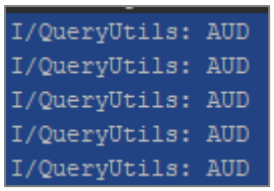
- Cycle through each element of the rates object
- Get its key name
- Using that key name - get the data
- Add the data to a new ArrayList element

Using these points, I searched online for answers and while researching I found this[12] Stack Overflow question that gave this code in one of the answers:

```
for (int i = 0; i < jsonArr.length(); i++) {
    JSONObject jsonObj = jsonArr.getJSONObject(i);
    String k = jsonObj.keys().next();
    Log.i("Info", "Key: " + k + ", value: " + jsonObj.getString(k));
  }
```

From this, I implemented *jsonObj.keys().next()* into my code to see what it did.

I used Log messages to test the concept, so that I don't have to worry about displaying it in a list in the app at the moment.

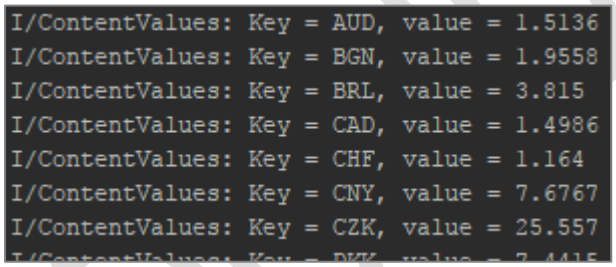


What the code provided, is 31 lines of the first object key (AUD) – there are 31 objects inside rates, so the FOR loop is working correctly, I just need to find a way of getting the loop to go to the next key.

After more research, I found another Stack Overflow question[13] that had a helpful answer, this was code that cycled through each JSON object, got its key name, and then its value – exactly what I needed.

```
for(int i = 0; i<jsonobject.length(); i++){
   Log.e(TAG, "Key = " + jsonobject.names().getString(i) + " value = " +
jsonobject.get(jsonobject.names().getString(i)));
}
```

I implemented this into my code and displayed in the Logcat was this, for all 31 currencies:



The keys and values for each exchange.

Now, using this piece of code, I can adapt it so each of these values is added to the ArrayList.

Finally, the connection to the API works. Using the code below, it produces a list of the currencies and their exchange rate in the app.


```
//For loop that cycles through all of the rates objects
        for (int i = 0; i < rates.length(); i++) {
           //Get the object key (country code)
           String objectKey = rates.names().getString(i);
           //Log.i(LOG_TAG, "Key: " + objectKey);

           //Get the exchange rate value associated with that object
           double keyValue = rates.getDouble(objectKey);
           //Log.i(LOG_TAG, "Value: " + keyValue);

           //Add those details along with the above to the array
           currencies.add(new Currency(exchangeDate, baseCurrency,
objectKey, keyValue));}
```

[12] https://goo.gl/3fmK4r
[13] https://goo.gl/2KdcXH

2.3.4 – Checking configurations

So far, I have been using the basic URL that the API provides, https://api.fixer.io/latest (using the secure HTTPS endpoint). However, it is a definite likelihood that parameters will be added onto this URL in the future – and so this should be tested now.

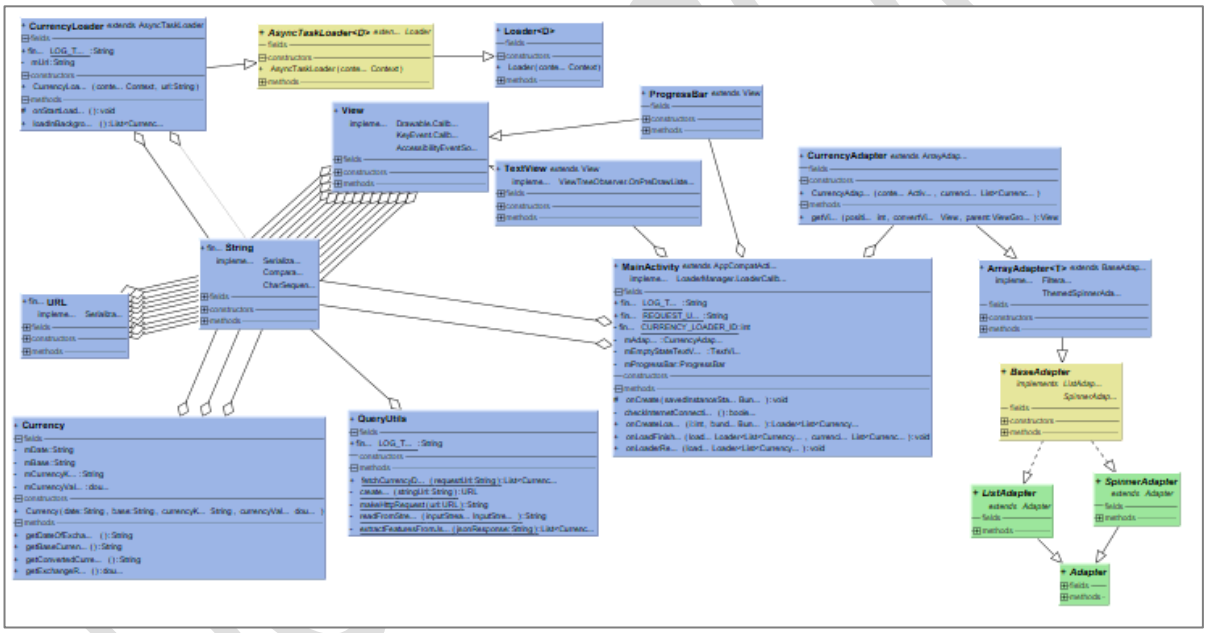
Some of the varying URLs I chose to test were:

- https://api.fixer.io/latest?base=USD – changing the base currency
- https://api.fixer.io/latest?symbols=USD,GBP – specifying which currencies to convert to (from Euros)
- http://api.fixer.io/latest?base=GBP&symbols=EUR,USD – combining the two parameters together

All of these were handled by the code and app as expected and worked correctly.

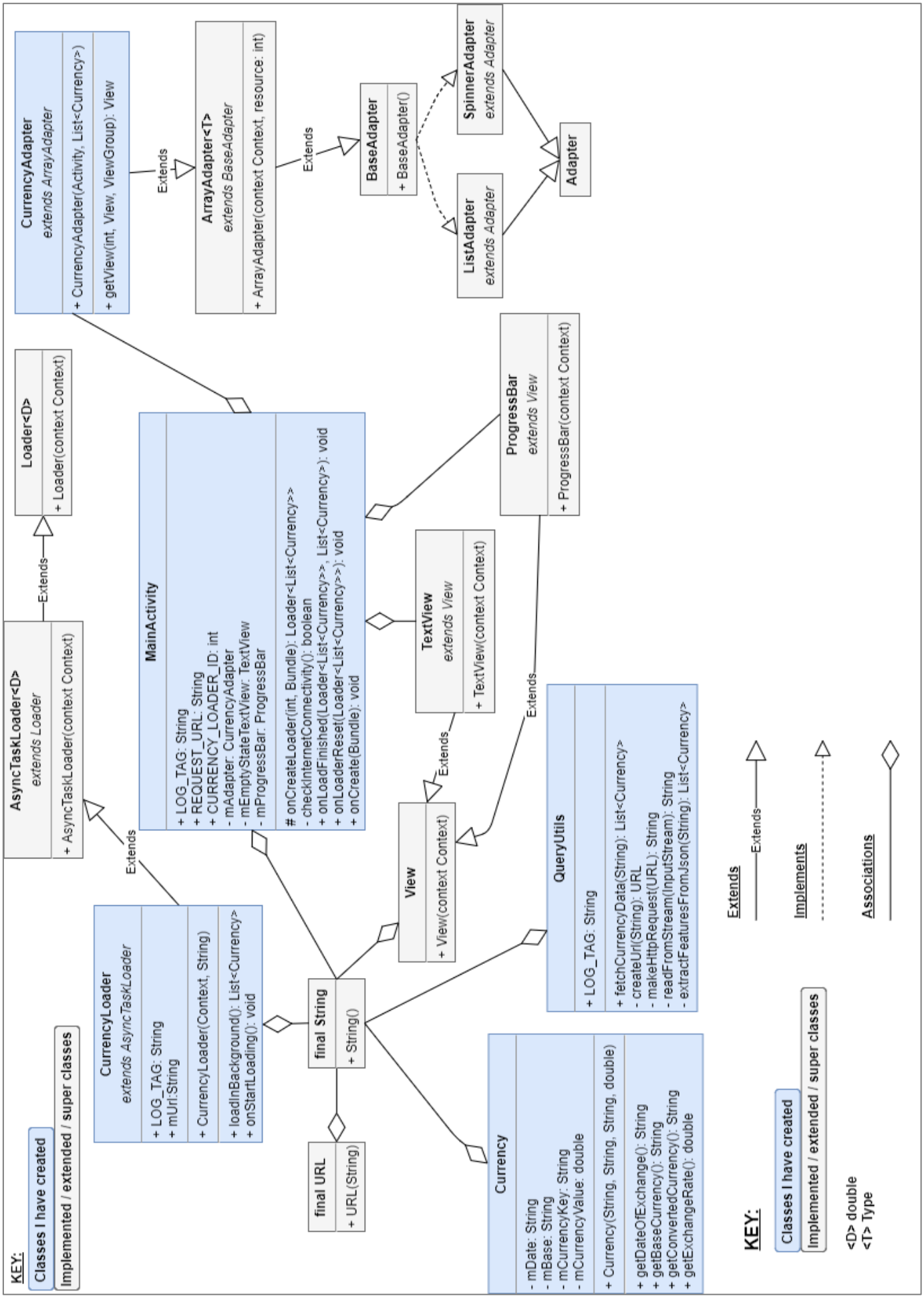
2.3.5 – UML Class Diagram

To create the class diagram for the API test app, I installed a plugin to Android Studio that adds the classes to the diagram automatically and lets you click on "implements" and "extends to" to then bring super classes and interfaces in with the correct UML arrows.



However, this had too much information for each class and interface, so I created my own diagram where only the classes, properties, constructors and methods that I used were included.

Seen below is my Class Diagram that represents this API test app.


CurrencyAdapter
extends ArrayAdapter
+ CurrencyAdapter(Activity, List<Currency>)
+ getView(int, View, ViewGroup): View
Extends
ArrayAdapter<T>
extends BaseAdapter
+ ArrayAdapter(context Context, resource: int)
Extends
BaseAdapter
+ BaseAdapter()
SpinnerAdapter
extends Adapter
ListAdapter
extends Adapter
Adapter
Loader<D>
+ Loader(context Context)
Extends
AsyncTaskLoader<D>
extends Loader
+ AsyncTaskLoader(context Context)
Extends
MainActivity
+ LOG_TAG: String
+ REQUEST_URL: String
+ CURRENCY_LOADER_ID: int
- mAdapter: CurrencyAdapter
- mEmptyStateTextView: TextView
- mProgressBar: ProgressBar
onCreateLoader(int, Bundle): Loader<List<Currency>>
- checkInternetConnectivity(): boolean
+ onLoadFinished(Loader<List<Currency>>, List<Currency>): void
+ onLoaderReset(Loader<List<Currency>>): void
+ onCreate(Bundle): void
ProgressBar
extends View
+ ProgressBar(context Context)
TextView
extends View
+ TextView(context Context)
Extends
Extends
CurrencyLoader
extends AsyncTaskLoader
+ LOG_TAG: String
+ mUrl:String
+ CurrencyLoader(Context, String)
+ loadInBackground(): List<Currency>
+ onStartLoading(): void
final String
+ String()
final URL
+ URL(String)
View
+ View(context Context)
QueryUtils
+ LOG_TAG: String
+ fetchCurrencyData(String): List<Currency>
- createUrl(String): URL
- makeHttpRequest(URL): String
- readFromStream(InputStream): String
- extractFeaturesFromJson(String): List<Currency>
Currency
- mDate: String
- mBase: String
- mCurrencyKey: String
- mCurrencyValue: double
+ Currency(String, String, String, double)
+ getDateOfExchange(): String
+ getBaseCurrency(): String
+ getConvertedCurrency(): String
+ getExchangeRate(): double
KEY:
Classes I have created
Implemented / extended / super classes
Extends
Implements
Associations
KEY:
Classes I have created
Implemented / extended / super classes
<D> double
<T> Type

2.4 – Main focus of the app: SQL – currencyData.db

Update: *It was at the point of writing section 2.4.2.3 that I realised that the SQL functionality would not be implemented into the final app (this is discussed in detail later in that section), that is why section 2.4 as a whole may lack detail in some areas but seem more complete in others, notably more complete up to section 2.4.2.3. Despite this, all the work has been left as it was written (in present tense) – so that the thought and design processes that I had at the time can be understood.*

As mentioned at the beginning of section 2.3, up to now the UI of the app has been designed as documented in section 2.2 and the basic parts of the app have been tested using the sample apps in section 2.1. In the previous section (2.3), the connection to the API was successfully established, and now in this section the SQL will be integrated into the app.

Before implementing a database into Android, I need to know how my database is going to be structured, for this I need to plan tables and various queries, each noted below. In addition, Android uses a variant of SQL called SQLite, which is lighter database framework optimised for smaller devices such as smartphones, in my writing I may refer to SQLite where specific principles apply to only that variant, or I may refer to SQL itself when broader principles can be applied to the language as a whole.

2.4.1 – Planned tables

Within an SQL database, tables are the fundamental to holding the data that you want to store, below are the tables that I plan on including in my database.

2.4.1.1 – apiData

This table's purpose will be to hold all of the data that comes from the API, so that it can be retrieved by the app when there is no Internet connection.

I already have data that is being brought into the app from the API, this is:

- Base currency, "GBP" – String
- Date, "2017-11-06" – String
- Converted currency, "EUR" – String
- Conversion rate, "1.1313" – Double

Using this data, I can plan the attribute names for this table that the data from the API will go into:

_id	**baseCurrency**	**exchangeDate**	**convertedCurrency**	**conversionRate**
1	GBP	2017-11-07	EUR	1.1313

The first attribute is required as all tables in SQL require a unique identifier so that they can be normalised.

With information from the SQLite documentation[14] on supported datatypes I can now select the most appropriate data types for each attribute, as they will need to be declared when creating the database table. Other requirements that each attribute will require are also seen below.

Attribute:	**Datatype:**	**Other requirements:**
_id	INTEGER	Primary key, automatically increment
baseCurrency	TEXT	Not null

[14] https://goo.gl/v14gDG

exchangeDate	TEXT (SQLite does not have a supported Date datatype)	Not null
convertedCurrency	TEXT	Not null
conversionRate	REAL (See note below)	Not null

A note on datatype Real: the range of number types that SQLite has is very limited compared to SQL or Java, with the options being Integer and Real. Therefore, as the data coming in has decimal values the only option is to choose this datatype despite the storage/memory requirements it may use. The datatype of the values coming in from the API are currently Double, so an explicit conversion will need to take place – to go from Double to Real – before the data enters the table, unless when entering the values to the table they are implicitly converted.

To create this table, based on the above data, the following SQLite syntax can be used:

CREATE TABLE apiData (_id INTEGER PRIMARY KEY AUTOINCREMENT, baseCurrency TEXT NOT NULL, exchangeDate TEXT NOT NULL, convertedCurrency TEXT NOT NULL, conversionRate REAL NOT NULL);

2.4.1.2 – savedConversion

This table's purpose will be to hold all of the user's saved conversions that they make.

The data that needs to be saved into this table is:

- The original amount that the user entered,
- The currency of the original amount
- The value of the converted amount
- The currency of the converted amount
- The date of the conversion

I obtained these requirements from the saved conversions page that I built earlier on (section 2.2.2.2) and from speaking to my client about part 7 of their requirements in section 1.3.1.

Therefore, using the data above I can plan the attribute names for this table that the saved conversions will go into:

_id	**baseCurrency**	**amountToConvert**	**convertedCurrency**	**convertedAmount**	**conversionDate**
1	GBP	10.50	EUR	11.77365	2017-11-16

Again, the first attribute "_id" is required so that the table can be normalised and using the SQLite documentation on datatypes I can select the most appropriate data types for each attribute.

Attribute:	**Datatype:**	**Other requirements:**
_id	INTEGER	Primary key, automatically increment
baseCurrency	TEXT	Not null
amountToConvert	REAL	Not null
convertedCurrency	TEXT	Not null
convertedAmount	REAL	Not null
conversionDate	TEXT (SQLite does not have a supported Date datatype)	Not null

To create this table, based on the above data, the following SQLite syntax can be used:

CREATE TABLE savedConversion (_id INTEGER PRIMARY KEY AUTOINCREMENT, baseCurrency TEXT NOT NULL, amountToConvert REAL NOT NULL, convertedCurrency TEXT NOT NULL, convertedAmount REAL NOT NULL, conversionDate TEXT NOT NULL);

2.4.2 – Planned queries

To interact with the data within the above tables, queries will be required, below are the planned queries I intend to use.

2.4.2.1 – INSERT

For the tables to contain data, they need to have a statement telling them what data should be put where in the table. Getting information from the SQLite documentation on INSERT statements[15] I discovered that as long as the number of values inserted is equal to the number of columns in the table, there is no need to specify the column names before the values in the query – this helps to keep the query simple and less error-prone. The INSERT function in SQLite has the form of:

INSERT INTO table VALUES(…);

And so, for the apiData table the query would be the following:

INSERT INTO apiData VALUES(null, [baseCurrency_value], [exchangeDate_value], [convertedCurrency_value], [conversionRate_value]);

And, for the savedConversion table:

INSERT INTO savedConversion VALUES(null, [baseCurrency_value], [amountToConvert_value], [convertedCurrency_value], [convertedAmount_value], [conversionDate_value]);

Note that the value NULL is passed in for the _id field as this is an automatically incrementing field – which the database will fill in.

2.4.2.2 – SELECT

In addition to the above, data will need to be retrieved from the table – this can be completed using a SELECT statement, which has one of the following forms:

*SELECT * FROM table WHERE expression [ORDER BY expression];*

SELECT attributes FROM table WHERE expression [ORDER BY expression];

One of the contexts in which data will need to be retrieved from the **apiData table** is to get the conversion rate when making a conversion, this can be done using the second form from above expressed in the following way:

SELECT conversionRate FROM apiData WHERE convertedCurrency = [currency chosen to convert to];

For the **savedConversion table,** all the data for each record will need to be retrieved so that it can be displayed to the user in the saved conversions page – this being done using the following expression:

*SELECT * FROM savedConversion;*

[15] https://goo.gl/87aoCQ

2.4.2.3 – Other queries, and the dilemma…

Some of the other queries that may be implemented into the app include the following, though they are not discussed in detail due to the reasons below:

- REPLACE
- DELETE
- DROP

Before learning about the implementation of SQL into Android, I had underestimated the amount of time and work it would take to get a fully functioning database within an Android app. The entirety of section 2.4 and its subsections was being completed at the same time as learning about SQL via the Pets app (section 2.1.11), the idea being that I could apply the ideas from that app to this project as I went along – so I did not forget what I had learned. This was working well, except getting through the online course was taking a longer time than I had initially anticipated because of the amount of content. So far, I only had one app to represent the visuals of the final project, and another that brought in the data – to the user and my client, both of these apps are useless.

Due to this, I sat down with my clients and spoke to them about the issues I was facing and why the creation of the app may overrun. They said to me they needed a working app for when they go on their round-the-world trip and that at a most basic level, completes the currency conversions. Therefore, we agreed that I should put the data storage side of the app (**CLIENT REQUIREMENTS 6, 7 AND 8**) on hold and focus on merging the two existing apps I have to create a single fully functioning one. This itself, could possibly throw many errors, and so I will need to give myself time to solve those. Following this, re-assessing time, if the above has gone successfully, quickly, then some minor implementation of data storage could be added.

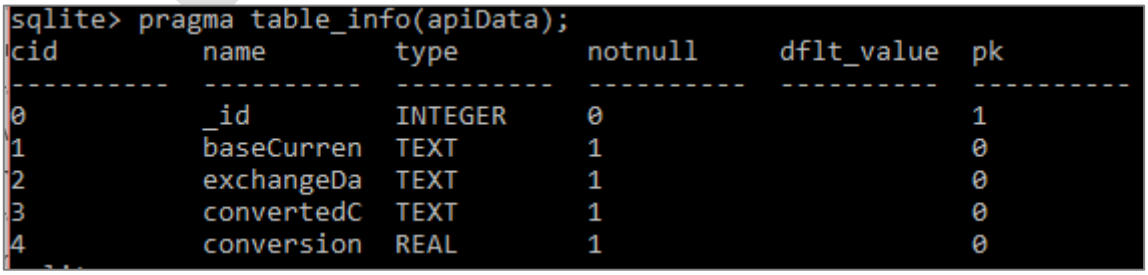
All of the previous notes I made on the SQL remain, and the following subsection 2.4.3 notes some testing I did for the above SQL statements. Then, the notes on creating the final app from the two existing ones will begin in section 2.5.

2.4.3 – Testing planned tables and queries

2.4.3.1 – Tables

Before using the above tables and queries in my app, I will need to test them to make sure that they work in a database. To do this, I downloaded SQLite 3 and created a sample database on my computer – currencyData.db.

First, I created the apiData table, then ran the command “pragma table_info(table_name)” to get the information about the table. This gave the following results, showing that the information I had entered in the CREATE statement had been interpreted correctly.



```
sqlite> pragma table_info(apiData);
cid         name        type        notnull     dflt_value  pk
----------  ----------  ----------  ----------  ----------  ----------
0           _id         INTEGER     0                       1
1           baseCurren  TEXT        1                       0
2           exchangeDa  TEXT        1                       0
3           convertedC  TEXT        1                       0
4           conversion  REAL        1                       0
```

Following this, I created the savedConversion table, then ran the same pragma command, but with the different table name. This gave me the following, showing again that the CREATE statement had been interpreted correctly.

```
sqlite> pragma table_info(savedConversion);
cid         name        type        notnull     dflt_value  pk
----------  ----------  ----------  ----------  ----------  ----------
0           _id         INTEGER     0                       1
1           baseCurren  TEXT        1                       0
2           amountToCo  REAL        1                       0
3           convertedC  TEXT        1                       0
4           convertedA  REAL        1                       0
5           conversion  TEXT        1                       0
```

Some notes on the column headers for the pragma command:

- **cid** – column ID
- **name** – the attribute name
- **type** – the datatype of the attribute
- **notnull** – whether the attribute can be left empty when inserting a new record (Boolean value – 0: false, 1: true). _id is left as 0 as it is the primary key which will automatically be assigned by SQLite when inserting a new record.
- **dflt_value** – the default value that the attribute should be assigned if nothing else is. These are left blank as the Java code will deal with all the data validation and verification.
- **pk** – primary key, the unique identifier for the table (Boolean value – 0: false, 1: true)

Another note, in both images the attribute names are not complete, this is only due to the column width cutting some characters off and not a misinterpretation of the CREATE statement.

2.4.3.2 – Queries

To test the that the INSERT queries were working correctly, I entered two lots of data into each table. The second INSERT statement checking that the auto-increment function is working. Following this, I ran the select all from [table] statement to view the results.

```
sqlite> INSERT INTO apiData VALUES(null, "GBP", "2017-11-17", "EUR", "1.1213");
sqlite> INSERT INTO apiData VALUES(null, "EUR", "2018-11-17", "GBP", "0.9761");
sqlite> SELECT * FROM apiData;
_id         baseCurrency  exchangeDate  convertedCurrency  conversionRate
----------  ------------  ------------  -----------------  --------------
1           GBP           2017-11-17    EUR                1.1213
2           EUR           2018-11-17    GBP                0.9761
```

As the above screenshot shows, the data was interpreted correctly.
I repeated this for the savedConversion table, again the data being interpreted correctly:

```
sqlite> INSERT INTO savedConversion VALUES(null, "GBP", "10.50", "EUR", "11.77365", "2017-11-16");
sqlite> INSERT INTO savedConversion VALUES(null, "EUR", "11.20", "AUR", "20.3340", "2057-31-16");
sqlite> SELECT * FROM savedConversion;
_id         baseCurrency  amountToConvert  convertedCurrency  convertedAmount  conversionDate
----------  ------------  --------------  -----------------  ---------------  --------------
1           GBP           10.5            EUR                11.77365         2017-11-16
2           EUR           11.2            AUR                20.334           2057-31-16
```

In displaying the data for the screen-capture above, I have already tested the SELECT query for the savedConversion table (*SELECT * FROM savedConversion;*) and so, only need to test the SELECT query for the apiData table (*SELECT conversionRate FROM apiData WHERE convertedCurrency = [currency chosen to convert to];*). For this example, I will be using the currency "EUR" as it is in the table already, and should return the value "1.1213".

```
sqlite> SELECT conversionRate FROM apiData WHERE convertedCurrency = "EUR";
conversionRate
--------------
1.1213
sqlite>
```

As the above screen capture shows, the query works correctly.

2.5 – Creating the final app

This section documents the process of creating the final app, up to the point where it is presented to my client as a fully functioning application for their smartphone based on the **CLIENT REQUIREMENTS**.

Because this app is created from the two prototype apps "Conversion Prototype" & "CurrencyAPITest" which have already had their design and creation discussed in detail in sections 2.2 and 2.3, this section resultingly has less detailed information on the parts that have already been discussed. Though, anything new or changed is noted below.

2.5.1 – Creating the new project

As so to preserve the previous two apps as prototypes, I created a new app entirely for the currency converter and going through the Android Studio app setup for this presented no difficulties. When Android Studio creates a new project, it requires a company domain to uniquely identify the package (if the app was ever to be uploaded to the Google Play Store) and up until this point I have been using the default *com.example.android*. Now, because this app is the final version I am using my own domain (though, you don't actually have to own the web domain) to mark the app as my own: *zacmurphy.com.* This gives the package the following name: *com.zacmurphy.currencyconverter.*

In addition, as discussed in section 1.5 I am using API level 22 as a minimum to build the project on and using an Empty Activity as the starting point of the app (an Empty Activity is an activity that Android Studio creates for you which doesn't contain any content – it allows for the most freedom when creating an app). Furthermore, the main page of the app remains named as MainActivity so that I remember which Java file corresponds to the main page the user will be interacting with (as there will be a lot of other Java files, this will make it easier to find).

2.5.2 – Copying across the layout code

As I know that the layout code works, it can be copied across from the Conversion Prototype app. This includes the XML files for the activities, the resources for the spinner and any string files or other assets.

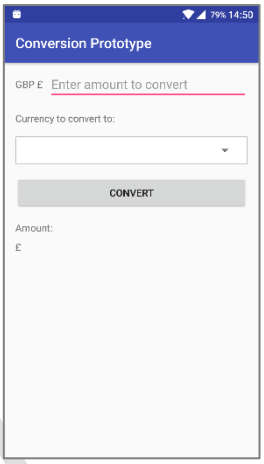
Due to the reasons discussed above in section 2.4.2.3, I am not including the saved conversions XML layout or any of the data storage features in the main app. Though, as there are buttons or menu items in some of the other files that are being included, these data storage functions will be commented out (as opposed to not including them in the code) so that I can choose to implement them or remove them entirely, dependent on how much time I have later on.

In addition, the ProgressBar from the main activity from the CurrencyAPITest app is copied into the new main activity, the other assets (e.g. string files) are copied across, the manifest file is updated with all of the correct information about each activity and the permissions are added.

2.5.2.1 – Testing the app runs

After implementing all of the above code, it is necessary to check that the app will still run, and that the layout looks as it is supposed to. As shown in the picture, the menu items are not present, nor is

the spinner populated. However, this is expected because these functions require Java code which has not yet been implemented. In addition, the Save Conversion button at the bottom of the page is not present, which is what is expected/wanted.



2.5.3 – Copying across the Java code

Following on from above, the corresponding Java files to the layout files are implemented from the Conversion Prototype app, any unneeded code (that links to data storage) commented out and the app is run again to test that it still works as intended – with the results being as expected.

Next, the Java files from the CurrencyAPITest app are copied, with the main activities being combined. Section 2.3 goes into more detail about what these files contain.

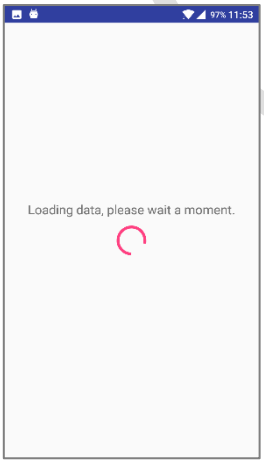
Obviously, as I had anticipated, after combining all of the code together the app doesn't completely work – there are a lot of bugs and a few errors. One of the ones I can understand, is that the test app was written to gather the data then display it in a list, but the new context does not require a list and the data needs to be stored in variables (due to the lack of a database). Therefore, to separate the loading / storage (in variables, probably as an ArrayList) and gathering of the data from the main user interface, I will create a loading activity. This also helps to separate the two layouts, and not have to worry about hiding the main user interface when the app starts, it can be treated as another page which is just switched to when the app is ready to move on.

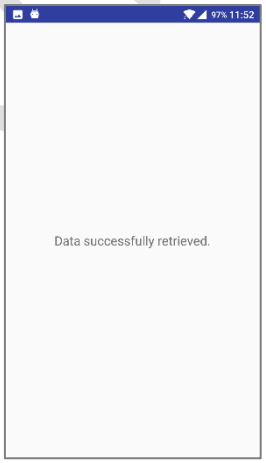
2.5.4 – Creating the loading activity

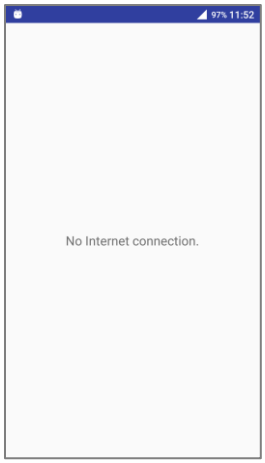
After creating the LoadingActivity files, I moved all of the code that dealt with retrieving the API results from the MainActvity to the Loading Activity and the ProgressBar XML element from the main activity layout file to the LoadingActivity layout file.

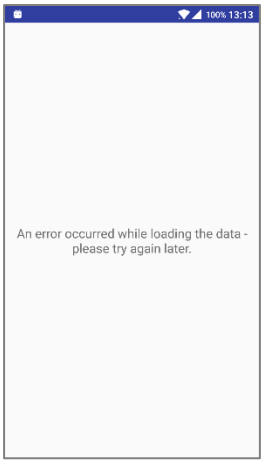
2.5.4.1 – Loading messages

Following the above, I added a TextView to the layout which would appear alongside the ProgressBar on the loading screen. It displays different messages to the user based on different circumstances.









As shown in the pictures above, I disabled the app's loading screen title bar giving the page an almost full screen look. To do this, I created a theme in styles.xml that used a parent theme of one with no title bar, and then set that theme to the activity in the Manifest. However, if there is time at

the end I would like to implement the removal of the notification bar, giving the user an immersive full screen.

The various messages shown in the picture above are determined by:

- Loading data: this is the default value of the TextView
- Lack of an Internet connection: the method in LoadingActivity which uses the Connectivity manager to test for an active connection.
- An error or success in loading the data: a public Boolean called ERROR_OCCURRED which is declared in QueryUtils, and set initially to false. When a try:catch block fails, the Boolean value is changed. Then, an if statement is used once the data loading process has finished to determine if an error occurred or not based on the Boolean value.

I then ran the app to test it, and after a few tweaks it was working – the app successfully connected to the API, retrieved the data and stored it in an ArrayList (see section below), but did not take the user any further than the Loading activity.

2.5.4.2 – Where the data is being stored

Despite **CLIENT REQUIREMENTS 6-8** not needing to be fulfilled anymore, and the app not have any permanent storage – the data still needs to be stored somewhere so that it can be used by the other parts of the app. Therefore, it is stored within the Currency class with each item being an object in the class to be accessed as part of an ArrayList. This turns out to be the same as initially coded in section 2.3.1.1. Though, because the CurrencyAdapter is not being used (as mentioned in the next section), to access this data a call to the public ArrayList variable has to be used, such as shown in the code below:

```
public static List<Currency> currenciesList = new ArrayList<>();
```

```
currenciesList.get(i).getBaseCurrency() + "
currenciesList.get(i).getConvertedCurrency()
currenciesList.get(i).getDateOfExchange() + "
currenciesList.get(i).getExchangeRate()
```

Where 'i' is the item position that is to be accessed.

```
V/LoadingActivity: 31
V/LoadingActivity: GBP, AUD, 2017-12-27, 1.7288
V/LoadingActivity: GBP, BGN, 2017-12-27, 2.2076
V/LoadingActivity: GBP, BRL, 2017-12-27, 4.4243
```

This code is from a FOR loop that outputs the size of the ArrayList into the Logcat and then each of the attributes separated by a comma, line by line for each object.

2.5.4.3 – Using the Android Studio Logcat

After testing the app at the end of section 2.5.4.1, I decided to track the flow of data through the app so far. This was because I couldn't understand when each part of the program was running, notably – when the loader methods were being called and if the CurrencyAdapter was being used at all. To solve this, I added debug Log comments with notes to let me know when each method was run and to return some of the variables at certain points to make sure they were correct.

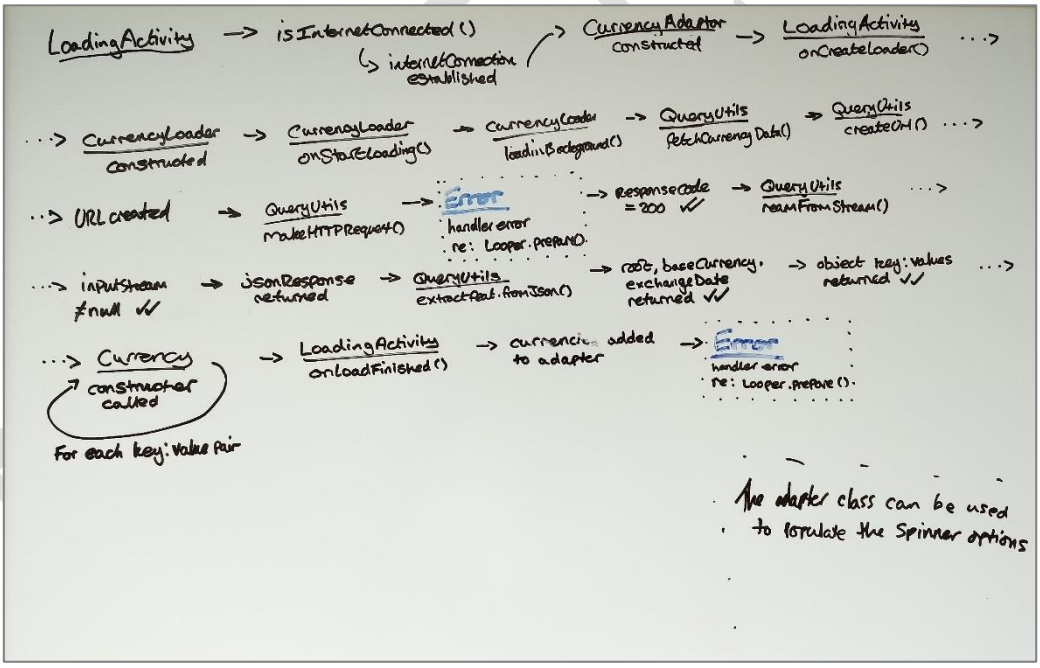
In addition, I altered the Android Logcat UI colours, so that the different types of message would stand out from one another. This will help me to ignore some messages and find others easier, for example errors.

```
VERBOSE/ProtocolEngine(24): DownloadRate 104166 bytes
DEBUG/dalvikvm(2227): HeapWorker thread shutting down
INFO/dalvikvm(2234): Debugger is active
WARN/ActivityManager(564): Launch timeout has expired,
ERROR/AndroidRuntime(4687): Uncaught handler: thread m
ASSERT/Assertion(4687): Expected true but was false
```

This resulted in part of the Logcat, at this point of where the app is in development, to look like the following:

```
12-28 11:04:30.712 ? W/System: ClassLoader referenced unknown path: /data/app/com.zacmurphy.currencyconverter-2/lib/
12-28 11:04:30.722 ? D/LoadingActivity: LoadingActivity - started
12-28 11:04:30.726 ? W/art: Before Android 4.1, method android.graphics.PorterDuffColorFilter android.support.graphi
12-28 11:04:30.747 ? D/LoadingActivity: isInternetConnected() - called
12-28 11:04:30.748 ? V/LoadingActivity: Internet connection established
12-28 11:04:30.749 ? D/CurrencyAdapter: Constructor - called
12-28 11:04:30.749 ? D/LoadingActivity: onCreateLoader() - called
12-28 11:04:30.749 ? D/CurrencyLoader: constructor - called
12-28 11:04:30.750 ? D/CurrencyLoader: onStartLoading - called
12-28 11:04:30.751 ? D/CurrencyLoader: loadInBackground - called
12-28 11:04:30.751 ? D/QueryUtils: fetchCurrencyData - called
12-28 11:04:30.751 ? D/QueryUtils: createUrl - called
12-28 11:04:30.752 ? V/QueryUtils: url: https://api.fixer.io/latest?base=GBP
12-28 11:04:30.752 ? V/QueryUtils: url: https://api.fixer.io/latest?base=GBP
12-28 11:04:30.752 ? D/QueryUtils: makeHttpRequest - called
12-28 11:04:30.752 ? D/QueryUtils: before if statement
12-28 11:04:30.752 ? D/QueryUtils: before variables initialised
12-28 11:04:30.753 ? E/AbstractTracker: Can't create handler inside thread that has not called Looper.prepare()
12-28 11:04:30.753 ? D/AppTracker: App Event: start
12-28 11:04:30.756 ? I/DpmTcmClient: RegisterTcmMonitor from: com.android.okhttp.TcmIdleTimerMonitor
12-28 11:04:30.757 ? D/QueryUtils: urlConnection opened
12-28 11:04:30.758 ? D/QueryUtils: timeouts set
12-28 11:04:30.758 ? D/QueryUtils: request method set
12-28 11:04:30.760 ? D/OpenGLRenderer: Use EGL_SWAP_BEHAVIOR_PRESERVED: true
```

From the entire Logcat response I was able to draw out which methods and classes were run at which point, and for the app so far, came up with a rough diagram showing this.



As the diagram shows, the CurrencyAdapter class is never called, this is because this class deals with the creation of views, which isn't used in the app as of yet. However, as the note in the bottom right corner says – the CurrencyAdapter will be able to be used to populate the spinner.

Furthermore, an error appears at two points within the Logcat, though it doesn't crash the app. When repeatedly running the app and checking the Logcat, the error does not always appear in the same place, or it sometimes only appears once (at the beginning and not at the end). Therefore, this has led me to believe that the error is somehow created at an earlier point in the app and due to the speed at which responses are output in the Logcat, it only manages to output the error message in

amongst all of the other messages, and not in its true location. In addition, looking into it, it seems that the error is something to do with completing tasks on specific threads (e.g. background vs UI thread) and is not critically endangering the app. So, I will leave it for now, as the implementation to fix it is very lengthy and complicated – and I need to focus on getting the app fully working first.

The next stage of coding will be to navigate the user to the main activity once the data has been loaded successfully.

2.5.4.4 – Linking to the MainActivity

This is simply done using an intent, which I know how to use from the Multiple Pages app in section 2.1.5. However, this time it will be important that the user cannot navigate back to the previous activity – as it is only a loading activity, and so must be closed. This can be achieved using the following piece of code obtained from a Stack Overflow question[16]:

```
//Create a new Intent object constructor, populate it with MainActivity.class
Intent intent = new Intent(this, MainActivity.class);
//Make sure the user cannot navigate back to this activity by using the back button
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
//Start that new intent
startActivity(intent);
//Close the current activity
finish();
```

2.5.5 – Populating the spinner with the ArrayList data

Once the user has been changed to the MainActivity, the spinner needs to be populated with data, so that the user can interact with it. To deal with this, data from the ArrayList can be used with a custom Adapter – helpfully, I already have one created: *CurrencyAdapter.* To use this on the spinner a new adapter has to be created using the CurrencyAdapter class and have the ArrayList passed in – then, have this adapter set onto the spinner. The following code illustrates this:

```
public void createSpinnerOptions() {
    Log.d(LOG_TAG, "createSpinnerOptions - called");
    //Create an ArrayAdapter based on the CurrencyAdapter using the li
    ArrayAdapter adapter = new CurrencyAdapter(this, currenciesList);

    //Apply the adapter to the spinner
    mSpinner.setAdapter(adapter);
}
```

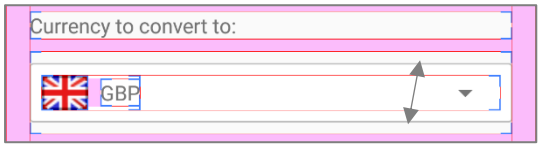
The variable *currenciesList* is a public variable declared in the Currency class that allows access to the ArrayList, as mentioned in section 2.5.4.2.

Following this, the methods *getView()* and *getDropDownView()* are overridden in CurrencyAdapter to provide views for the spinner when nothing is selected and when the options are to be shown.

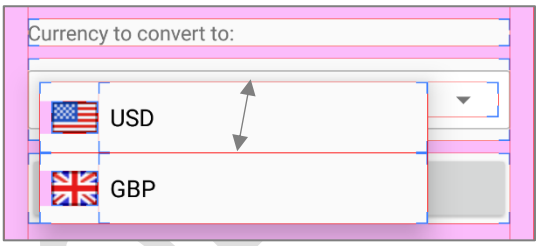
[16] https://goo.gl/mTqDwh

Within these methods the code follows the following steps:

1. Get the current item in the ArrayList
2. Check if an existing view is being reused, otherwise inflate (create) the view from the specified resource
    - Each of the above methods has a different layout resource file, though to the user the elements will be laid-out in very similar ways. They have to be different, so that when selecting the item (when the spinner is dropped-down) there is enough space for the user to easily select the item, but then it can go back to a smaller size once it has been selected. The diagrams below help to explain this.



This is using getView()



This is using getDropDownView()

For these diagrams, I enabled Android's Developer option on my phone – show layout bounds. From this, you can see that the area for each element on the right is slightly taller than on the left. Having two separate layout files also means that in the future either of the situations could have a completely different look to the other.

3. Lookup the views that need to be populated within the layout
    - These being an ImageView and a TextView
4. Use the method *getConvertedCurrency()* provided by the Currency class to return the country code associated with the current ArrayList item, and set that value to the TextView
5. Use a custom method to return the resource ID for the flag
    - Where the country code is passed in and formed into part of the ID
    - If there is no resource matching the ID generated, 0 is returned
6. As long as the resource ID exists (i.e. isn't 0) set the resource for this ArrayList item to the ImageView. Else, get rid of the view so that it doesn't take up unnecessary space
    - This fulfils **CLIENT REQUIREMENT 5** in section 1.3.1
7. Return the completed view to appear within the spinner on-screen
8. Repeat for each item in the ArrayList

The code below shows the steps described above for *getDropDownView()*.

```
@Override
public View getDropDownView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
    Log.d(LOG_TAG, "getDropDownView - called");
    //Get the data item for this position
    Currency currentExchange = getItem(position);

    //Check if an existing view is being reused, otherwise inflate the view
    if (convertView == null) {
        convertView = LayoutInflater.from(getContext()).inflate(R.layout.spinner_dropdown_item, parent, false);
    }

    //Lookup view for data population
    TextView country = convertView.findViewById(R.id.spinnerDropDownText);
    ImageView icon = convertView.findViewById(R.id.spinnerDropDownImage);

    //Set the correct text to the TextView
    country.setText(currentExchange.getConvertedCurrency());

    //Get the resource ID
    int resId = getObjectResourceId(currentExchange.getConvertedCurrency());

    //If there is no resource for the item, 0 will be returned
    if (resId != 0) {
        //Set the resource to the ImageView for this item
        icon.setImageResource(resId);
    } else {
        //Get rid of the view
        icon.setVisibility(View.GONE);
    }

    //Return the completed view to render on-screen
    return convertView;
}
```

For the above, I have been using the URL "https://api.fixer.io/latest?symbols=USD,GBP" so that the app only brings in two currencies, so that testing it is manageable, but I will use one of the URLs mentioned in section 2.3.4 at a later point and for when the app is completed. This shouldn't change any of the above as the adapter and ArrayList are designed to work with any number of items.

2.5.5.1 – Dynamically creating the resource IDs

As mentioned in the above section, step 5 of the sequence uses a custom method to return the resource ID for the flag, where the country code is passed in and formed into part of the ID. This is all done dynamically, instead of using a switch statement, so that the code is easy to read and takes up less space. If a switch statement were to be used, using the following format, then there would be a lot of lines of code for all 30 currencies provided by the API.

Switch(countryCode) {
Case "GBP":
Return [resource ID];
Case "USD":
Return [resource ID];
Default:
Return countryCode;
}

So instead, I named all of the flag assets "flag_" followed by their country code in lowercase, this allowing me to use the following code to dynamically create their ID, specifying the ID to get and the type of resource.

return getContext().getResources().getIdentifier("flag_" + countryCode.toLowerCase(), "drawable", getContext().getPackageName());

This piece of code above is also good, because it returns a zero if the resource doesn't exist instead of throwing an error or crashing the app. This allows me to do a check before trying to assign a resource that doesn't exist to the ImageView – so no errors are thrown.

```
//Get the resource ID
int resId = getObjectResourceId(currentExchange.getConvertedCurrency());

//If there is no resource for the item, 0 will be returned
if (resId != 0) {
    //Set the resource to the ImageView for this item
    icon.setImageResource(resId);
} else {
    //Get rid of the view
    icon.setVisibility(View.GONE);
}

//Return the completed view to render on-screen
return convertView;
```

Overall, this piece of code is a more efficient way of completing the same task that a switch statement would.

2.5.5.2 – Ordering the ArrayList

To help complete **CLIENT REQUIREMENT 4** from section 1.3.1 the spinner options need to be re-ordered. At present, the spinner displays the items in the order that they are fetched from the API – alphabetical (I know this from section 2.5.4.2). However, to make the app more user-friendly it would make sense to include the more commonly used worldwide currencies at the top of the list.

To sort the items, each one needed to have a priority assigned to it, so I went back and altered the Currency class to include a priority attribute and a *getPriority()* method. Following this, I was able to use a custom method using a switch statement in the QueryUtils class to assign a priority based on the country code. For this I searched online and used a website[17] to determine which currencies should have precedence over others, then leaving all others with a default value of 10. These priority values are then added to each item in the ArrayList along with all of the other data.

```
private static int getPriority(String currencyCode) {
    Log.d(LOG_TAG, "getPriority - called");
    switch (currencyCode){
        case "USD":
            return 0;
        case "EUR":
            return 1;
        case "JPY":
            return 2;
        case "AUD":
            return 3;
        case "CAD":
            return 4;
        case "CHF":
            return 5;
        case "CNY":
            return 6;
        default:
            return 10;
    }
}
```

Next, I used this[18] Stack Overflow question and answer combined with this one[19] on how to order items in an ArrayList using an integer property, which gave me the following code:

[17] https://goo.gl/6ejw7T
[18] https://goo.gl/SrsoRQ
[19] https://goo.gl/4wmp63

```
private class CurrencyComparator implements Comparator<Currency> {
    public int compare(Currency left, Currency right){
        Log.d(LOG_TAG, "compare class - called");
        return Integer.compare(left.getPriority(), right.getPriority());
    }
}
```

Which takes two objects (left and right) of Currency, and compares the values of the methods given.

This can then be called, passing in the ArrayList to be sorted:

```
//Re-order the ArrayList, to prioritise the more frequently used currencies to the beginning
Collections.sort(currenciesList, new CurrencyComparator());
```

This results in the ArrayList being sorted and populated into the spinner in the following order:

```
USD, 2017-12-29, 1.3517, 0
EUR, 2017-12-29, 1.1271, 1
JPY, 2017-12-29, 152.17, 2
AUD, 2017-12-29, 1.7297, 3
CAD, 2017-12-29, 1.6951, 4
CHF, 2017-12-29, 1.3189, 5
CNY, 2017-12-29, 8.7964, 6
BGN, 2017-12-29, 2.2044, 10
BRL, 2017-12-29, 4.4779, 10
CZK, 2017-12-29, 28.781, 10
DKK, 2017-12-29, 8.3912, 10
HKD, 2017-12-29, 10.563, 10
```

Notice the priority values on the right-hand side – they are incremental, until 10, and then the objects are left in alphabetical order.

This fulfils CLIENT REQUIREMENT 4.

2.5.6 – Getting the converted value for the user

Now that the user can enter a value and select the currency they wish to convert to, functionality will need to be implemented to convert the value for them.

To help fulfil CLIENT REQUIREMENTS 1, 2 AND 3 to complete the main, primary, function of the app I began by looking at the method *onConvertClick()*. This is the function that is responsible for gathering the required data and performing various calculations via calls to other methods within it.

2.5.6.1 – Get the selected spinner option

The first thing that needed to be done was to get the currently selected spinner option, as this would provide the country code needed for the currency symbol, and without a symbol the user may be left unsure as to whether their value had been converted correctly or not. This method simply gets the position of the selected item within the spinner, and then uses this to return the country code from the ArrayList that corresponds to it.

```
//Get the currently selected item's position
int position = mSpinner.getSelectedItemPosition();

Log.v(LOG_TAG, "Selected item: " + currenciesList.get(position).getConvertedCurrency());
//Use that position to look up the respective item in the list and return the corresponding country code
return currenciesList.get(position).getConvertedCurrency();
```

2.5.6.2 – Get the currency symbol

Following this, the country code is used to get the relevant currency symbol from strings.xml using the same dynamic method as the code in section 2.5.5.1, except this time the resource is named "symbol_" followed by the country code, and the resource type is a string.

symbolId = getApplicationContext().getResources().getIdentifier("symbol_" + countryCode, "string", getPackageName());

In addition, as mentioned in section 2.5.5.1, a zero is returned if the resource doesn't exist, so another check can be made to prevent any errors occurring.

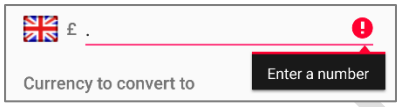
```
//As long as there is an existing string resource available
if (symbolId != 0) {
    //Get the resource
    symbol = getResources().getString(symbolId);
} else {
    //Assign the country code
    symbol = countryCode;
}
```

To know which countryCode corresponded to which currency symbol I used the currency website XE, in which they have a page[20] dedicated to matching currencies to symbols.

2.5.6.3 – Calculating the value

The next thing that is required, is for the exchanged value to be calculated. For this, two main variables are needed: the user's input and the exchange rate for the selected currency.

To get the user's input, the combined *getText().toString()* operation can be used on the EditText to retrieve a string. Following this, a check is undertaken to make sure the string is not empty, or just a decimal point, as when parsing the string to a double datatype these conditions would throw an error. In both cases, a value of zero is assigned to the double variable, as a value is needed for calculations further on, but if a decimal point is entered by itself (".4" is registered as "0.4" and "4." is registered as "4.0") then a prompt will appear for the user. Any other characters and symbols are restricted by the keyboard input, as mentioned in section 2.2.2.1.3, helping to enforce data validation on the user input.



Following this, the exchange rate is obtained for the correct currency by getting the selected spinner item position, and using it to get the particular currency from the ArrayList and calling the *getExchangeRate()* method on it.

Then the calculation *valueToConvert * exchangeRate* is returned by the method to provide the calculated value for the user.

2.5.6.4 – Updating the user interface

Finally, the user interface is updated using a method that takes in the two parameters of the relevant currency symbol and the converted value. It then checks if the value is zero and if so only displays the symbol, otherwise it displays the symbol and quantity. To display the two variables a xliff tag is used, which Android Studio recommends (as it doesn't like two variables appended together to form the text in a TextView). This is simply done using two tags in this case (1 for either variable), with each specifying the ID, datatype and giving an example. Then to use the string, *getString()* is called with the string resource specified and the variables passed in, in the order they should appear.

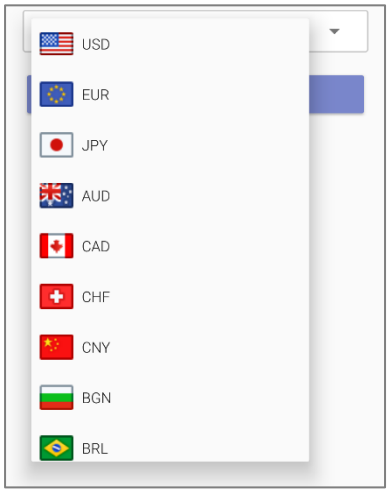
<string name="content_convertedValueText"><xliff:g example="£" id="relevantCurrencySymbol">%s</xliff:g> <xliff:g example="1" id="exchangedValue">%f</xliff:g></string>

[20] https://goo.gl/GEBPXT

mConversionResultField.setText(getString(R.string.content_convertedValueText, relevantCurrencySymbol, exchangedValue));

2.5.7 – Adding all image assets and currency symbols

Now that the app's functionality is complete I can change the request URL to get all exchange rates from the base currency GBP, add all the currency symbols to strings.xml and add all of the currency flags to the project. This now gives the user a wider range of currencies to choose from, and as in section 2.5.5.2 the spinner places the more common currencies nearer to the top.



To obtain the resources for the flags, I searched online and found a website that provided free icon sets. On this website I was able to find a set of world flags[21] which were already optimal size for my project – so no editing of the files was necessary.

2.5.8 – The menu and its options

Now that the main page of the app has been created alongside the loading screen, the menu and its options (which are already implemented) can be looked at more closely to help make sure **CLIENT REQUIREMENT 9** is completely fulfilled.

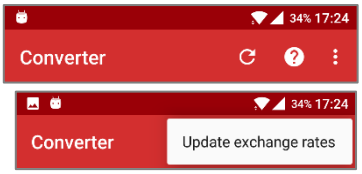
Part A of the requirement says that the page needs to be refreshed – this would be for the user entry-field to be cleared, the spinner reset and converted value field to be cleared. This is something that the Conversion Prototype app already did, and as such the functionality was copied over to this app when copying the Java and XML code in sections 2.5.2 and 2.5.3. Though, part B cannot be fulfilled because there will not be a saved conversions page, due to the reasons in section 2.4.2.3. However, also copied across from the Conversion Prototype app is the help page, and as such its menu option.

When speaking to my client about these menu options they said that another functionality, not included in their original requirements, would be to have an option to manually check for the latest exchange rates while the app is running. Then, to help clarify this for the user, include what date the results are from within the Help & About page.

To do this, I added another option to the menu.xml file and another case to the switch statement for when the option is selected. Following this, a method that would be run when the user clicked on the option. Within this method, the ArrayList is cleared so that the list does not have any duplicate data, then an intent is run in the same way as section 2.5.4.4, but calling the Loading activity instead. This basically restarts the app, and the latest exchange rates are retrieved from the API.

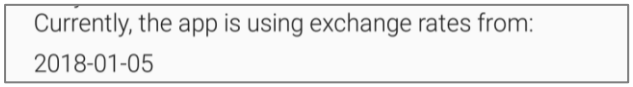
[21] https://goo.gl/VtLcBZ

Because the new menu option is not a primary function, it is located within the overflow menu, which when selected will show the option.



Note: *these screen captures were taken after styling had been applied in section 2.6.3 below*.

The screen capture below shows the Help & About page section.

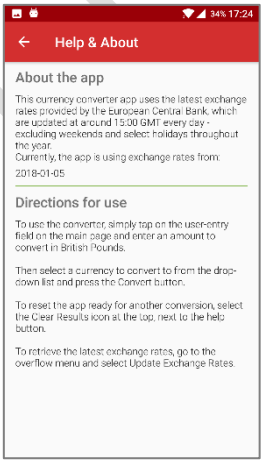


2.6 – Giving the app a visual overhaul and adding small features

Now the app has its main functionality, more effort can be put into styling it and making it look nicer as well as giving it some additional features that weren't the focus previously.

2.6.1 – Giving the Help & About page content

To give this page content, all I did was to change the string values within the strings.xml file that corresponded to those in the layout file. I also added a divider to separate the two parts of the page.



Note: *this screen capture was taken after styling had been applied in section 2.6.3 below.*

2.6.2 – Adding the framework for styling

In Android, to apply a theme or style to something you can set a specific tag to the element within the XML, such as *android:textColor="@color/black"*. However, to do this for each element would be more hassle than necessary, especially with many tags for each element (text style, colour, size etc.).

Instead, the styles.xml file can be utilised, where a general style can be created and changed in one place and then applied to many elements, giving the same results. This is a much more code efficient way to complete the process, especially for when you are using trial and error to choose a particular colour or font for an element and keep having to change values. Therefore, before sitting down with my client to implement the styles, I implemented the framework for the following:

- Smaller headings
- Larger headings
- Smaller content
- Larger content
- Notification style messages
- Dropdown lists
- Buttons

The following piece of code shows an example of the small heading style and where it is applied.

```
<!--Smaller Heading text style-->
<style name="HeadingSmall" parent="@android:style/TextAppearance.Large">
    <item name="android:textSize">16sp</item>
    <item name="android:textColor">@color/secondaryText</item>
    <item name="android:fontFamily">sans-serif-medium</item>
</style>
```

```
<!--Label for the spinner-->
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:text="@string/heading_currencyToConvert"
    style="@style/HeadingSmall"/>
```

2.6.3 – Styling/selecting themes

As mentioned in section 1.10, I am adhering to the Material Design guidelines to make my app look and feel like other apps from the Play Store, while keeping to **CLIENT REQUIREMENT 11**. Using these guidelines, I sat down with my client and they chose the various fonts[22] and colours[23] for the app from the Material Design with the help of two sites.[24] These model the colours in various situations before implementing them into the app. Though, we went through many variations before deciding on the final implementations – these were:

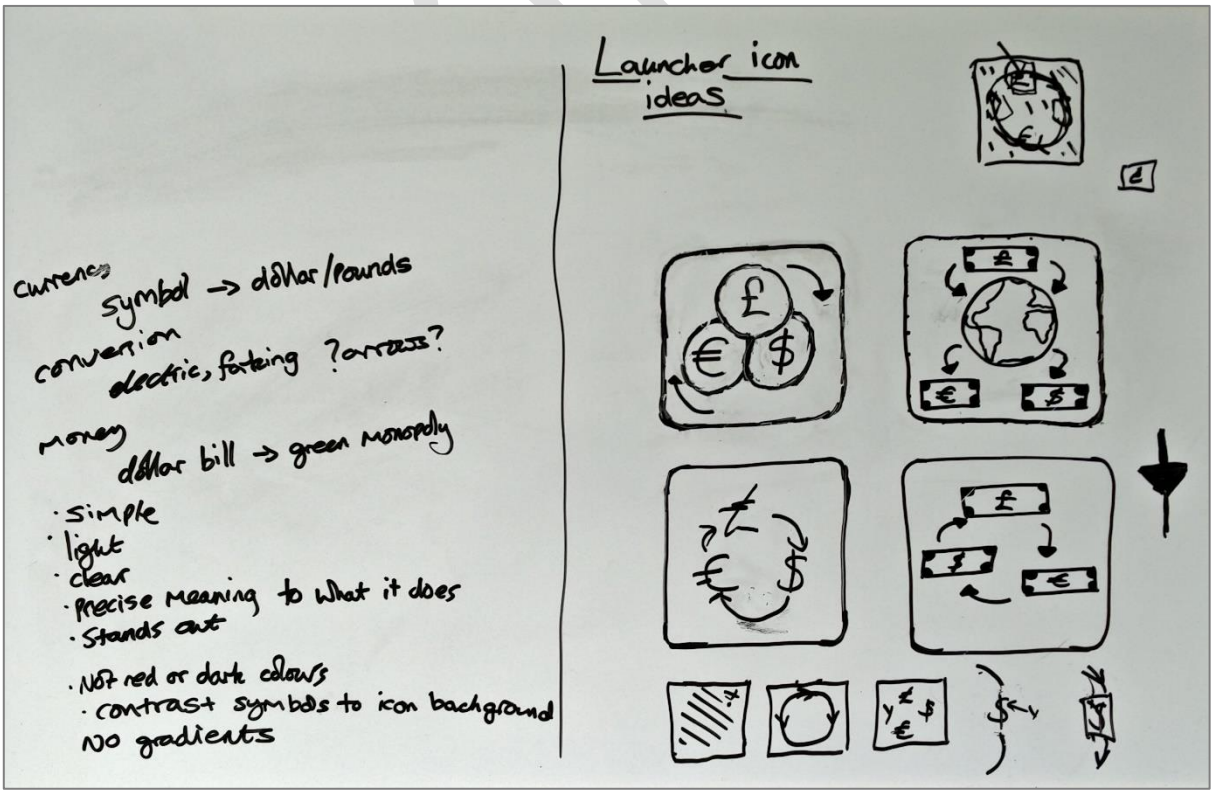
- To use Sans Serif as the font family across the app
- To have a shade of red as the primary colour, and a shade of green as the secondary colour
- To use the secondary colour for the divider in help & about, the loading spinner, the convert button and the underline on the user-entry field
- To have the loading screen have a theme of grey

In addition, my client decided instead of having the letters "GBP" at the top of the main page, they wanted the UK flag, so I implemented that by changing the TextView to an ImageView and setting the background to the correct resource.

The final chosen colours and themes within the app can be seen in section 2.7.

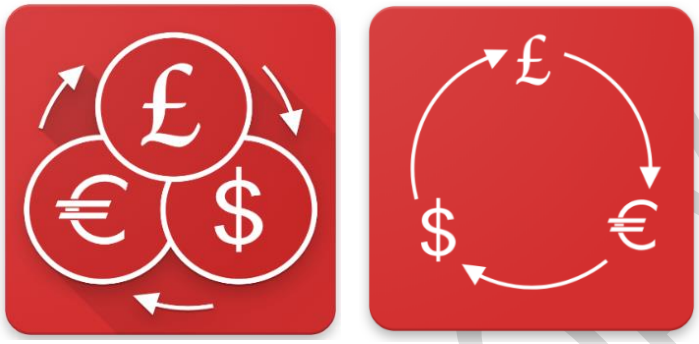
2.6.4 – Creating the launcher icon

To keep within the colour scheme of the app that my client had now chosen, the launcher icon would also be the same shade of red and to design the icon I did some initial sketches based on some ideas and thoughts from my client, while keeping in mind **CLIENT REQUIREMENT 12**.



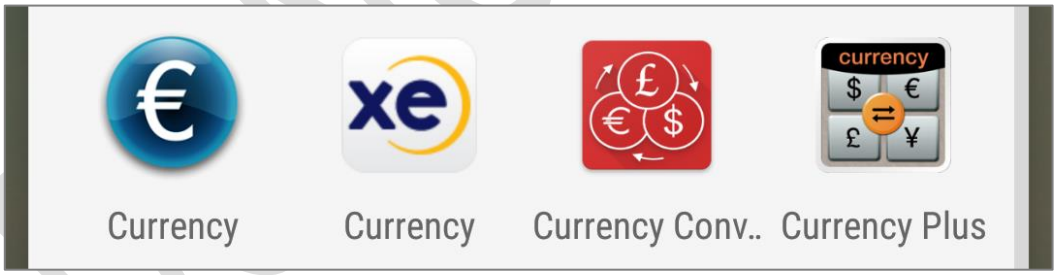
[22] https://goo.gl/pNcjXP
[23] https://goo.gl/vLW6Zk
[24] https://goo.gl/ptgUrq / https://goo.gl/1ze5sm

From these I created some vector graphics, and uploaded them to the Android Asset Studio: Launcher icon generator[25]. This online tool takes the uploaded image and scales it down correctly to the different sizes that Android requires for different sized screens. In addition, it allows you to choose the background shape and colour, to have any shadows on the design, and the amount of padding the image uploaded has around the edge. Using this tool, I came up with two variants and based on the clarity of the icons at smaller sizes I chose the first icon.



Android also requires a circular icon, so using the online tool I created one which is shown below.

This is what my icon looks like next to the other currency converters, mentioned in section 1.7.



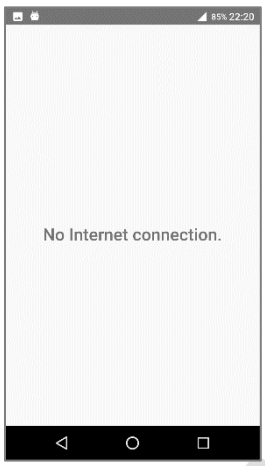
2.6.5 – Giving the loading activity an immersive full-screen look

To complete this, I used a piece of sample code from the Android Developer's documentation[26] which adds another method to the code which can be called in *onCreate()* when the activity starts. It uses system flags to tell the app which system bars to hide.

Then when the app changes to the next activity, the flags are reset, and the system UI bars are present again. Though, if one of the error messages is displayed and the activity is not changed, the method *showSystemUI()* is called which shows the system UI for the user to navigate away from the app.

[25] https://goo.gl/fKGeY3
[26] https://goo.gl/YkS8ZV

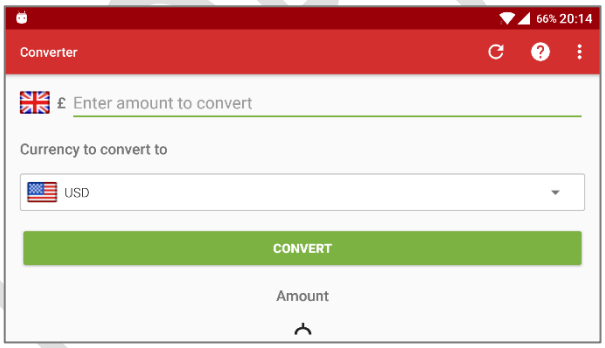
Showing full-screen



Showing full-screen with system UI bars

2.6.6 – Forcing orientation

While designing and creating the app, my smartphone would occasionally change orientation. When this happened the layout of the app would change accordingly, and I realised that the app would be very difficult to use in this state due to the layout and positioning of elements. For instance, the result field is cut off and can't be scrolled to. To fix this I added two lines of code in the Manifest file to stop the app from rotating, locking it in portrait mode. Helping to enforce **CLIENT REQUIREMENT 10**.



android:screenOrientation="portrait"
android: configChanges="orientation|keyboardHidden"

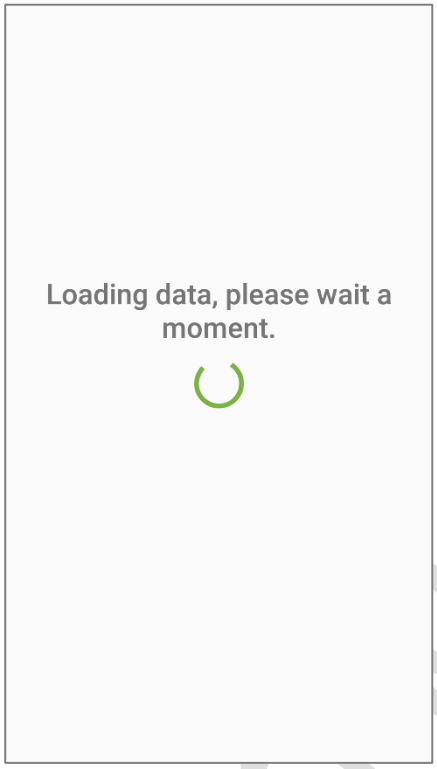
2.6.7 – Hiding the blinking cursor

Again, while designing and creating the app, I noticed that when the user is switched to the main activity – the user-entry field would already have its cursor blinking, waiting for a user input and once the user had entered their value, it would continue to blink. This would draw the user's attention away from other elements and possibly confuse them, making them think what they had entered wasn't correct or needed changing. Therefore, the line of code *android:cursorVisible="false"* was set in the XML for the element and a listener for the keyboard was set, so that when the keyboard was opened, the cursor was re-enabled – allowing the user to see where they were typing. Then, a command to disable the cursor was placed in various situations, such as when the keyboard is closed. This again, helped to fulfil **CLIENT REQUIREMENT 10**.

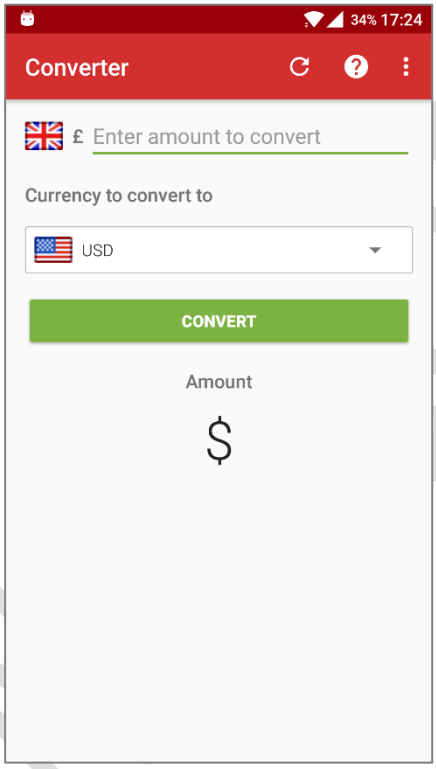
2.7 – The completed app

Shown below are screenshots of the final, completed, app. A YouTube video that I created shows the use and features of my final app and can be found here: http://bit.ly/CurrencyAppShowcase (Make sure to turn captions on).

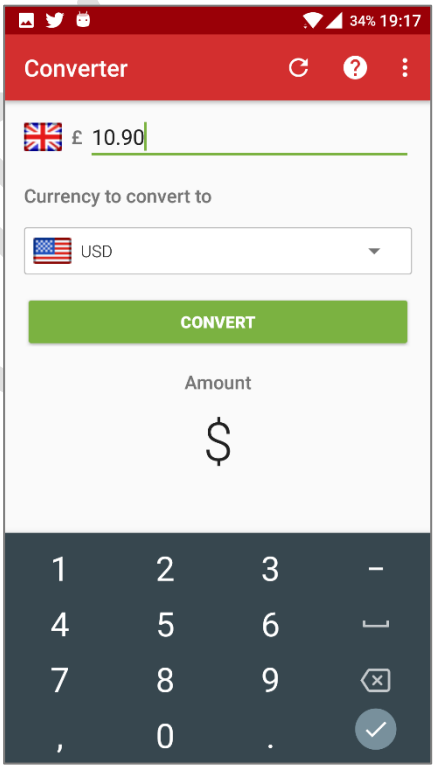
The app is also available to download and install on your own smartphone, with all the instructions here: http://bit.ly/APKdownloadGuide



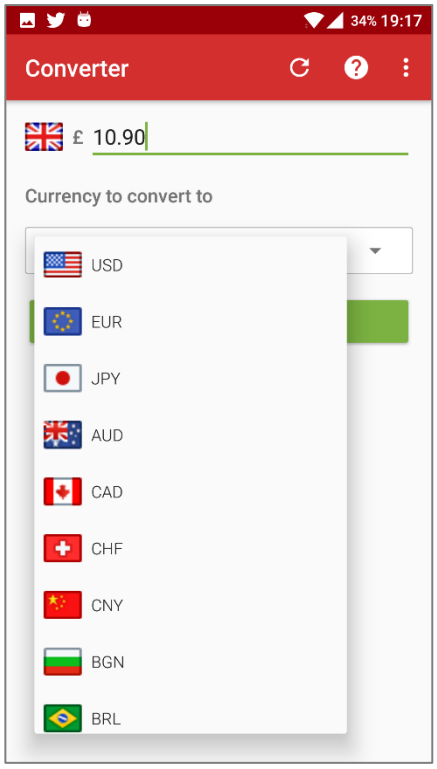
The loading screen



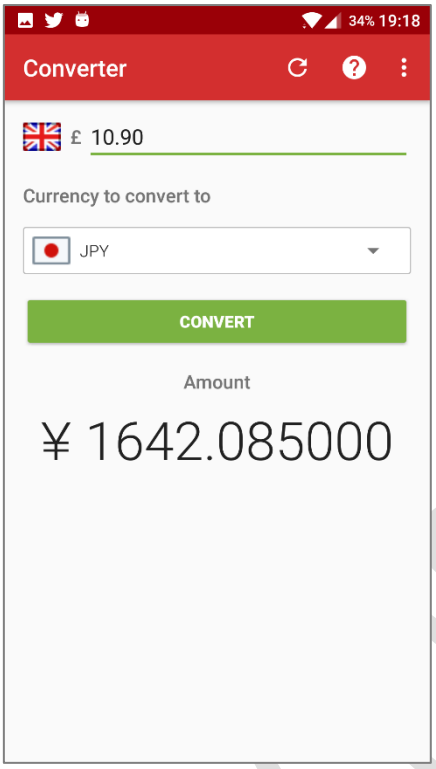
The page the user is taken to once the data has loaded



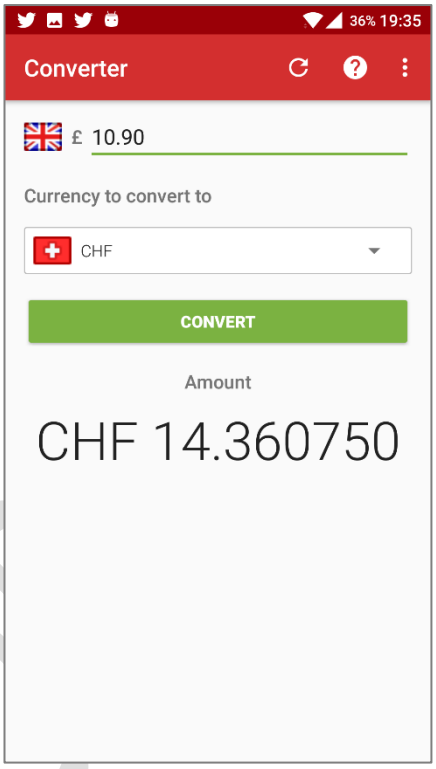
The user entering a figure to convert



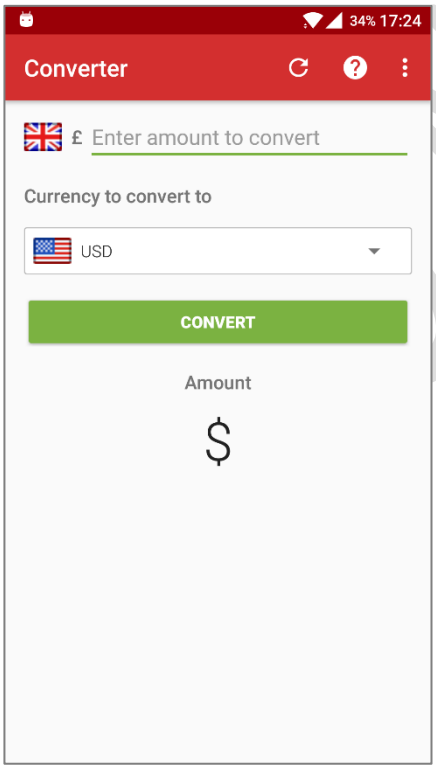
The user selecting a currency to convert to



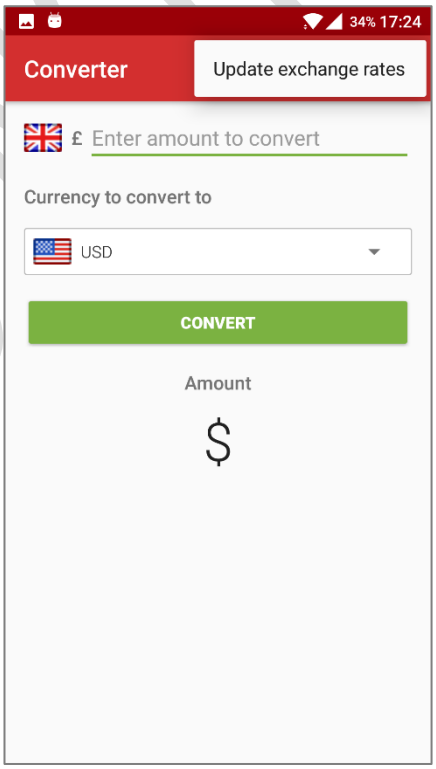
The value is displayed on-screen once the user has selected the currency



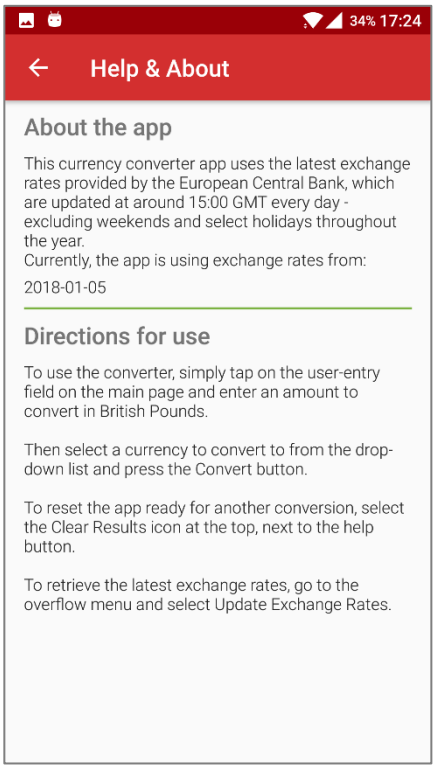
Keeping the initial value, the user has changed the currency to convert to



The user selected the Clear results menu option, and is now presented with a refreshed interface



The user has selected the overflow menu, and the option is displayed



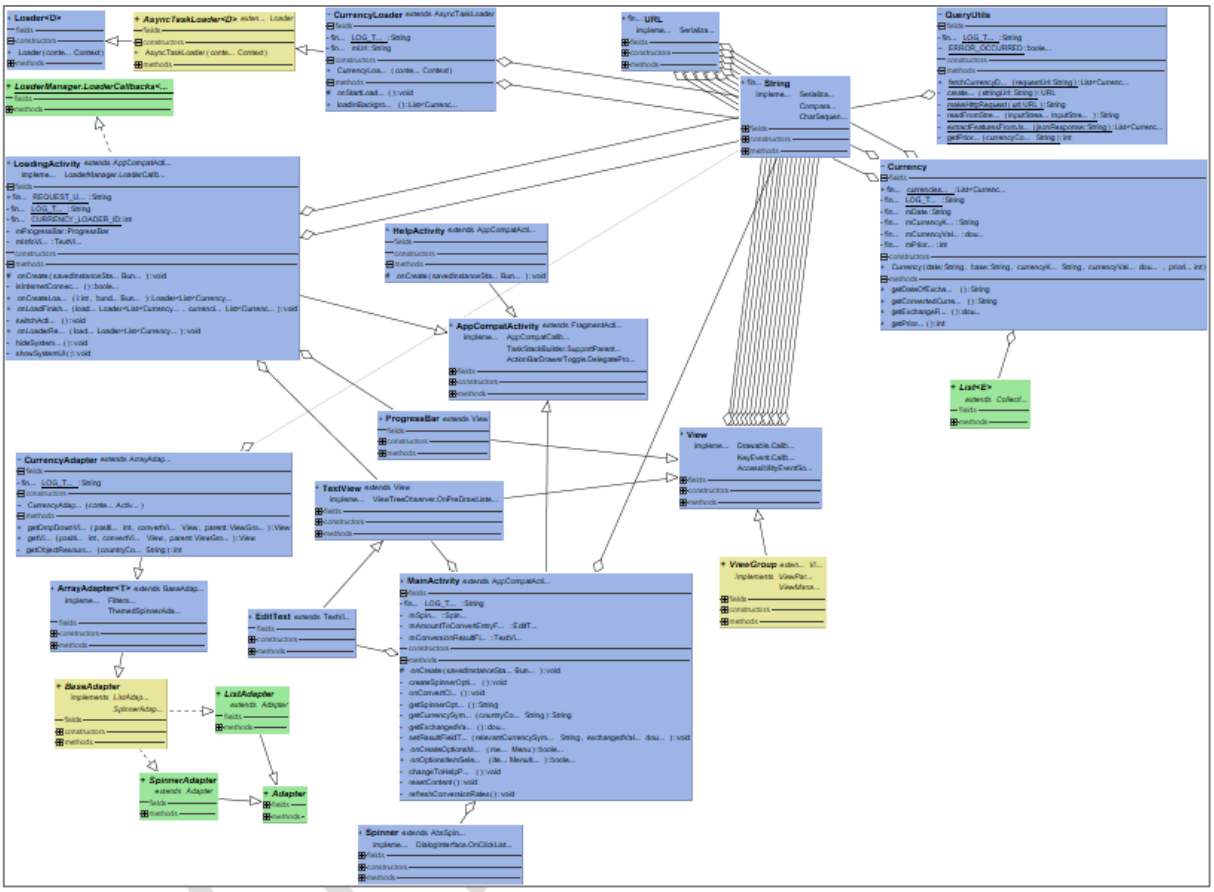
The user selected the Help & about menu option, and has been taken to that page

2.8 – Updated charts

Throughout the design and creation process the app has changed from the original in many ways, for instance the exclusion of the saved conversions page or the inclusion of the help page. Seen below are the final versions of these charts for the Currency Converter app.

2.8.1 – Java class UML diagram

As in section 2.3.5, I used the Android Studio plugin to create an initial diagram of the classes and interfaces, this shown below.



From this, I was able to create my own diagram where only the classes, properties, constructors and methods that I used were included.

Seen on the next page is my Class Diagram that represents the final Currency Converter app.

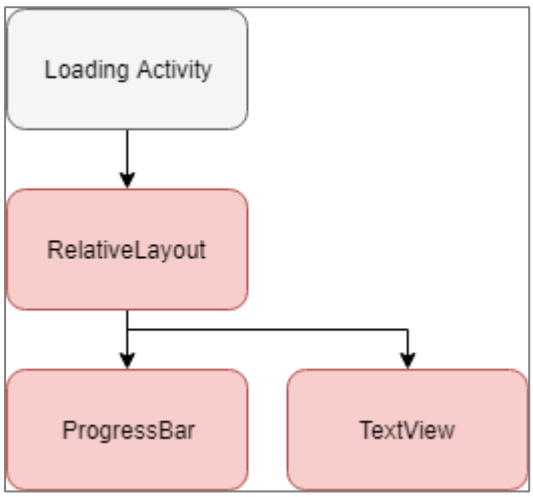
This page is a placeholder for the A3 UML class diagram (appendix 16).

2.8.2 – XML hierarchy charts

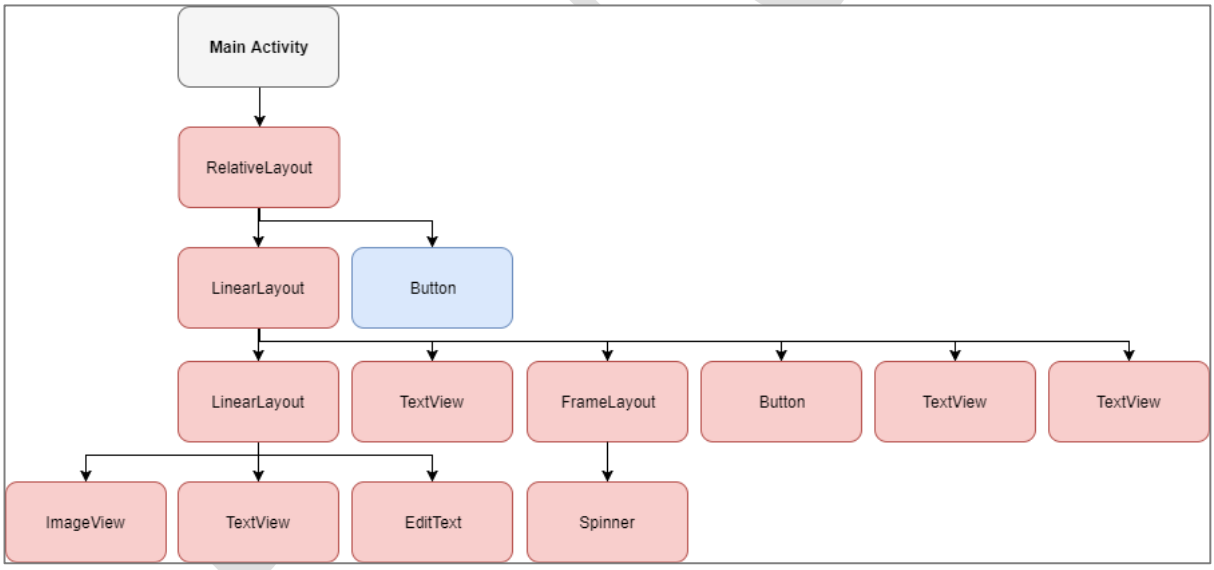
These hierarchy charts show the final structures of the main XML layout files.

Key:
Red = Implemented elements
Blue = commented-out elements

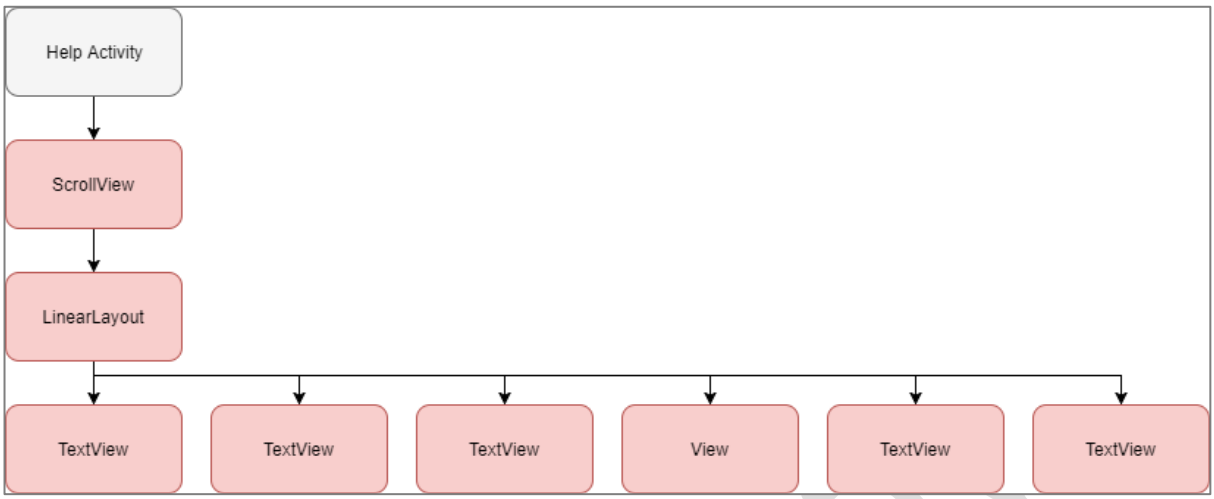
2.8.2.1 – activity_loading.xml



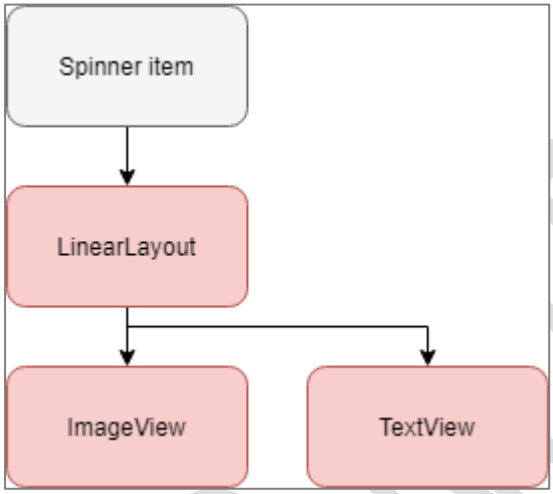
2.8.2.2 – activity_main.xml



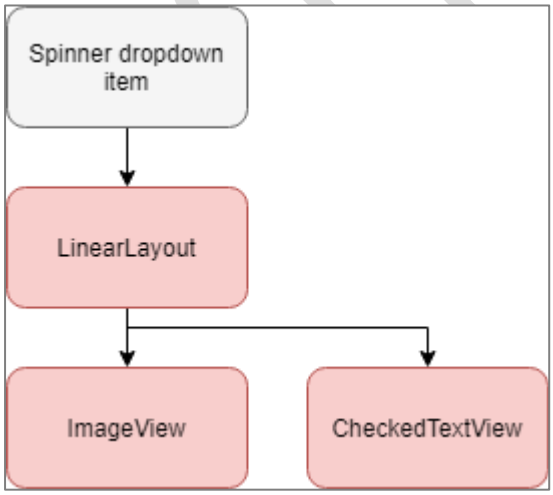
2.8.2.3 – activity_help.xml



2.8.2.4 – spinner_item.xml

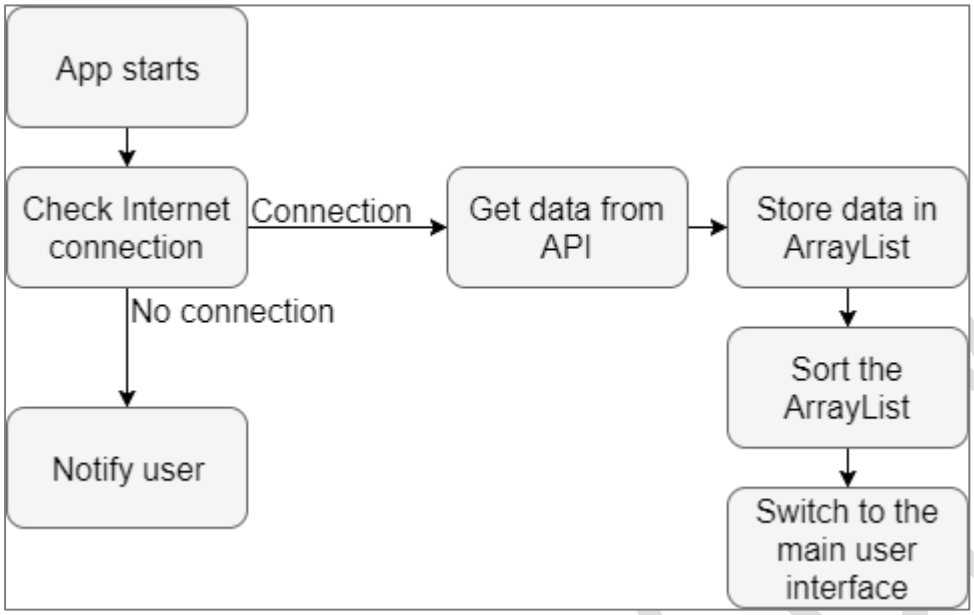


2.8.2.5 – spinner_dropdown_item.xml

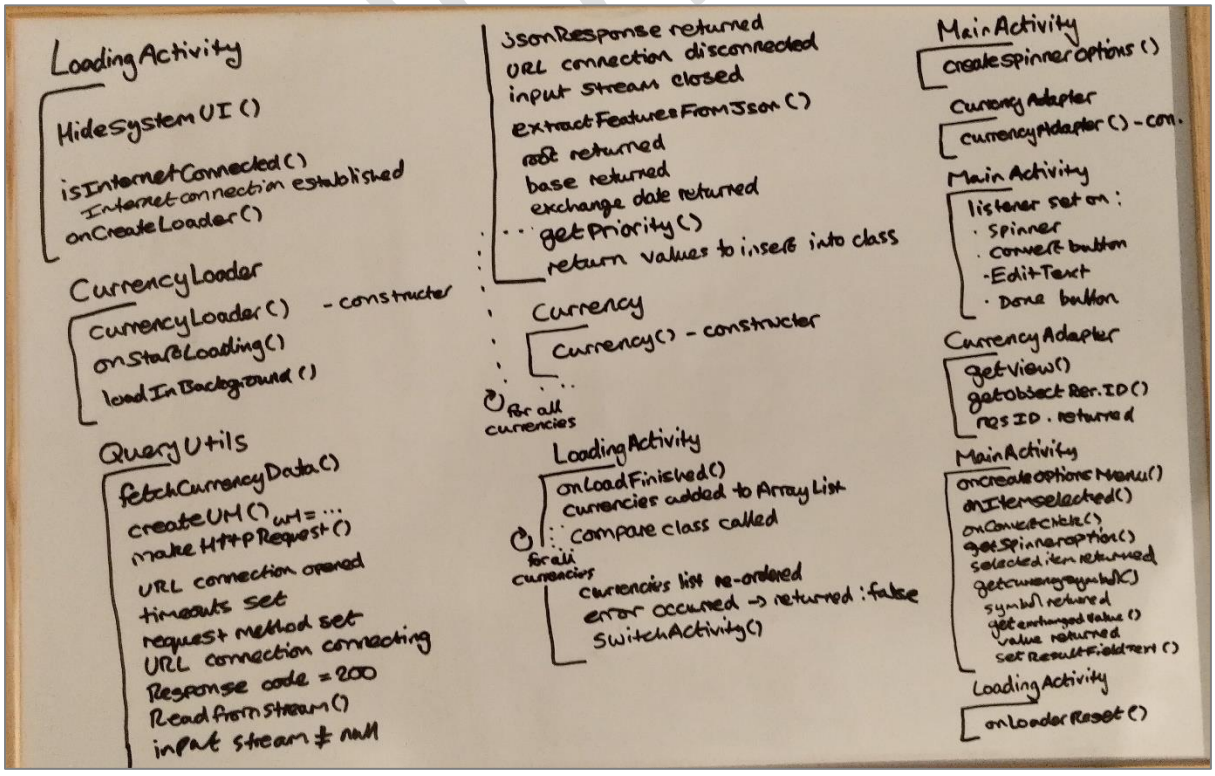


2.8.3 – Data flow diagram

To begin with, to show the very basic idea of the flow of data through the app, I created this flowchart:

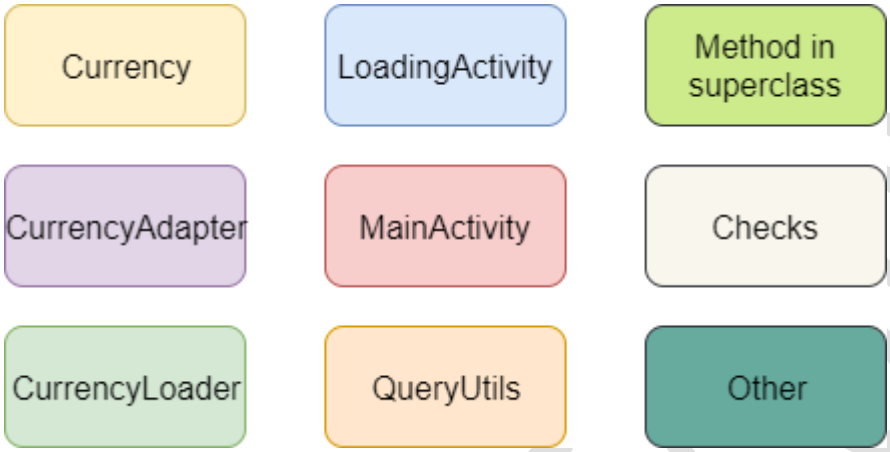


This gave me a good starting point, but it was too basic. So, to properly show the flow of data through the app, I used the debug messages in the Logcat (like in section 2.5.4.3) to show the order that the methods were called in and when the key variables were returned. This gave me something that looked like the following, based on the Logcat output in appendix 15. (The content of this image reads downwards, in columns).



Then, using the above as a reference I created a proper flowchart that shows the order of the methods as they are called, the key decisions the code makes and the key variables that are returned by those methods. The flowchart depicts these things from the moment the app is started to the point where the user is left to interact with the user interface after the loading of data. It shows the main flow of the loading/storing of data and leaves anything afterwards out. This is because the flowchart needs to stop eventually, to keep it simple and understandable, and not all eventualities of the program can be included on the flowchart.

The key to the flowchart is as follows and the flowchart itself can be seen on the next page.



This page is a placeholder for the A3 Data Flow Diagram (appendix 17).

3.0 – Technical Solution

3.1 – System overview

The following files show the code that I have created for the completed project, written in Java and XML. They use the charts in the previous section (2.8) as a basis for the structure of the code and the layouts. The code is commented thoroughly throughout and includes `blue Javadoc comments that annotate methods and some classes` and `general green comments`, both of which have been written by me. Furthermore, to make formatting and readability easier, each file is started at the top of a new page – this being the reason why in some cases there may be half a page or more before the next file.

The files and their corresponding sections are:

Java – 3.2

These files are used to control the actions of the program, whether those actions be automatically called or triggered by the user.

File	Description	Section ref.
Currency.java	The class that creates the currency objects and stores them in an ArrayList	3.2.1
CurrencyAdapter.java	The class that is responsible for creating the views of the spinner	3.2.2
CurrencyLoader.java	The class that deals with loading the processes in the background	3.2.3
HelpActivity.java	The class for the help and about page	3.2.4
LoadingActivity.java	The class for the loading page	3.2.5
MainActivity.java	The class for the main page	3.2.6
QueryUtils.java	The class that makes the connection and gets the data from the API	3.2.7

XML – 3.3

These files are used to provide information to the app in a variety of forms, whether that be layouts, resources or general information.

File	Description	Section ref.
activity_help.xml	The layout for the help and about page	3.3.1
activity_loading.xml	The layout for the loading page	3.3.2
activity_main.xml	The layout for the main page	3.3.3
AndroidManifest.xml	The Android Manifest for information about the app	3.3.4
bg_spinner.xml	The background style for the spinner	3.3.5
colors.xml	The colour palette of the app	3.3.6
main_menu.xml	The layout for the main menu	3.3.7
spinner_dropdown_item.xml	A layout for the spinner	3.3.8
spinner_item.xml	Another layout for the spinner	3.3.9
strings.xml	All of the text values	3.3.10
styles.xml	The styles for the headings and text	3.3.11

3.2 – Java files

3.2.1 – Currency.java

```
1   package com.zacmurphy.currencyconverter;
2
3   import android.util.Log;
4
5   import java.util.ArrayList;
6   import java.util.List;
7
8   /**
9    * The Currency class that creates and provides methods that return parts
10  of the ArrayList when called
11   * by other parts of the app
12   */
13  class Currency {
14
15      public static final List<Currency> currenciesList = new ArrayList<>();
16      //Tag for the log messages
17      private static final String LOG_TAG = Currency.class.getSimpleName();
18      //Declare private variables
19      private final String mDate;
20      private final String mCurrencyKey;
21      private final double mCurrencyValue;
22      private final int mPriority;
23
24      /**
25       * Create the constructor for this class, a constructor creates an
26  instance of a class
27       * This constructor will create an instance of three Strings, a double
28  and an int
29       *
30       * @param date          - the date the conversion rates are from
31       * @param base          - the base currency
32       * @param currencyKey   - the currency being converted to
33       * @param currencyValue - the exchange rate
34       */
35      public Currency(String date, String base, String currencyKey, double
36  currencyValue, int priority) {
37          Log.d(LOG_TAG, "Constructor - called");
38          mDate = date;
39          String mBase = base;
40          mCurrencyKey = currencyKey;
41          mCurrencyValue = currencyValue;
42          mPriority = priority;
43      }
44
45      /**
46       * @return the date that is associated with the results
47       */
48      public String getDateOfExchange() {
49          return mDate;
50      }
51
52  // --Commented out by Inspection START (05/01/2018 22:02):
53  //    /**
54  //     * @return the base currency
55  //     */
56  //    public String getBaseCurrency() {
57  //        return mBase;
```

```
58  //    }
59  // --Commented out by Inspection STOP (05/01/2018 22:02)
60
61      /**
62       * @return the currency that the conversion rate applies to
63       */
64      public String getConvertedCurrency() {
65          return mCurrencyKey;
66      }
67
68      /**
69       * @return the exchange rate that the country is providing
70       */
71      public double getExchangeRate() {
72          return mCurrencyValue;
73      }
74
75      /**
76       * @return the priority of the currency, determined in QueryUtils
77       */
78      public int getPriority() {
79          return mPriority;
80      }
81  }
```

3.2.2 – CurrencyAdapter.java

```
1   package com.zacmurphy.currencyconverter;
2
3   import android.app.Activity;
4   import android.support.annotation.NonNull;
5   import android.support.annotation.Nullable;
6   import android.util.Log;
7   import android.view.LayoutInflater;
8   import android.view.View;
9   import android.view.ViewGroup;
10  import android.widget.ArrayAdapter;
11  import android.widget.ImageView;
12  import android.widget.TextView;
13
14  class CurrencyAdapter extends ArrayAdapter<Currency> {
15
16      //Tag for the log messages
17      private static final String LOG_TAG =
18  CurrencyAdapter.class.getSimpleName();
19
20      /**
21       * A custom constructor.
22       *
23       * @param context is used to inflate the layout file
24       */
25      CurrencyAdapter(Activity context) {
26          // This initialises the ArrayAdapter's internal storage for the
27  context and the list.
28          // The second argument is used when the ArrayAdapter is populating
29  a single TextView.
30          // Because this is a custom adapter for a TextView and an ImageView
31  the adapter is not
32          // going to use this second argument, so it can be any value.
33          super(context, 0, Currency.currenciesList);
34          Log.d(LOG_TAG, "Constructor - called");
35      }
36
37      /**
38       * Overridden method to produce the contents of the spinner when
39  clicked on
40       *
41       * @param position    - the specific item in the list
42       * @param convertView - the view to change
43       * @param parent      - the parent activity
44       * @return the completed view
45       */
46      @Override
47      public View getDropDownView(int position, @Nullable View convertView,
48  @NonNull ViewGroup parent) {
49          Log.d(LOG_TAG, "getDropDownView - called");
50          //Get the data item for this position
51          Currency currentExchange = getItem(position);
52
53          //Check if an existing view is being reused, otherwise inflate the
54  view
55          if (convertView == null) {
56              convertView =
57  LayoutInflater.from(getContext()).inflate(R.layout.spinner_dropdown_item,
58  parent, false);
59          }
```

```
60
61          //Lookup view for data population
62          TextView country =
63  convertView.findViewById(R.id.spinnerDropDownText);
64          ImageView icon =
65  convertView.findViewById(R.id.spinnerDropDownImage);
66
67          //Set the correct text to the TextView
68          assert currentExchange != null;
69          country.setText(currentExchange.getConvertedCurrency());
70
71          //Get the resource ID
72          int resId =
73  getObjectResourceId(currentExchange.getConvertedCurrency());
74
75          //If there is no resource for the item, 0 will be returned
76          if (resId != 0) {
77              //Set the resource to the ImageView for this item
78              icon.setImageResource(resId);
79          } else {
80              //Get rid of the view
81              icon.setVisibility(View.GONE);
82          }
83
84          //Return the completed view to render on-screen
85          return convertView;
86      }
87
88      /**
89       * Overridden method to produce the contents of the spinner before it's
90  clicked on
91       *
92       * @param position    - the specific item in the list
93       * @param convertView - the view to change
94       * @param parent      - the parent activity
95       * @return the completed view
96       */
97      @NonNull
98      @Override
99      public View getView(int position, @Nullable View convertView, @NonNull
100 ViewGroup parent) {
101         Log.d(LOG_TAG, "getView() - called");
102         //Get the data item for this position
103         Currency currentExchange = getItem(position);
104
105         //Check if an existing view is being reused, otherwise inflate the
106 view
107         if (convertView == null) {
108             convertView =
109 LayoutInflater.from(getContext()).inflate(R.layout.spinner_item, parent,
110 false);
111         }
112
113         //Lookup view for data population
114         TextView country = convertView.findViewById(R.id.spinnerText);
115         ImageView icon = convertView.findViewById(R.id.spinnerImage);
116
117         //Set the correct text to the TextView
118         assert currentExchange != null;
119         country.setText(currentExchange.getConvertedCurrency());
120
```

```
121             //Get the resource ID
122             int resId =
123     getObjectResourceId(currentExchange.getConvertedCurrency());
124
125             //If there is no resource for the item, 0 will be returned
126             if (resId != 0) {
127                 //Set the resource to the ImageView for this item
128                 icon.setImageResource(resId);
129             } else {
130                 //Get rid of the view
131                 icon.setVisibility(View.GONE);
132             }
133
134             //Return the completed view to render on-screen
135             return convertView;
136         }
137
138         /**
139          * @return the resource ID based on the {@param countryCode}
140          */
141         private int getObjectResourceId(String countryCode) {
142             Log.d(LOG_TAG, "getObjectResourceId - called");
143             Log.v(LOG_TAG, "resourceId: " +
144     getContext().getResources().getIdentifier("flag_" +
145     countryCode.toLowerCase(), "drawable", getContext().getPackageName()));
146             //Append the country code to the prefix "flag_" to get the file
147     name, specify the resource type and package name - which return the
148     drawable resource
149             return getContext().getResources().getIdentifier("flag_" +
150     countryCode.toLowerCase(), "drawable", getContext().getPackageName());
151         }
152     }
```

3.2.3 – CurrencyLoader.java

```
1   package com.zacmurphy.currencyconverter;
2
3   import android.content.AsyncTaskLoader;
4   import android.content.Context;
5   import android.util.Log;
6
7   import java.util.List;
8
9   /**
10   * Currency loader task
11   */
12  class CurrencyLoader extends AsyncTaskLoader<List<Currency>> {
13
14      //Log tag that returns the package name for errors
15      private static final String LOG_TAG =
16  CurrencyLoader.class.getSimpleName();
17
18      //Global instance of the String URL, so it can be used in multiple
19  methods in this class
20      private final String mUrl;
21
22      /**
23       * The constructor for this class
24       */
25      public CurrencyLoader(Context context) {
26          super(context);
27          mUrl = LoadingActivity.REQUEST_URL;
28          Log.d(LOG_TAG, "constructor - called");
29      }
30
31      /**
32       * When the loader is instructed to load
33       */
34      @Override
35      protected void onStartLoading() {
36          //Force the load
37          forceLoad();
38          Log.d(LOG_TAG, "onStartLoading - called");
39      }
40
41      /**
42       * The background process for the loader
43       */
44      @Override
45      public List<Currency> loadInBackground() {
46          Log.d(LOG_TAG, "loadInBackground - called");
47          //Don't perform the request if there are no URLs, or the URL is
48  null
49          if (mUrl == null) {
50              Log.v(LOG_TAG, "mUrl is null");
51              return null;
52          }
53
54          //Get the data for the URL provided & Return the result
55          return QueryUtils.fetchCurrencyData(mUrl);
56      }
57  }
```

3.2.4 – HelpActivity.java

```
1  package com.zacmurphy.currencyconverter;
2
3  import android.os.Bundle;
4  import android.support.v7.app.AppCompatActivity;
5  import android.widget.TextView;
6
7  import static com.zacmurphy.currencyconverter.Currency.currenciesList;
8
9  public class HelpActivity extends AppCompatActivity {
10
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         setContentView(R.layout.activity_help);
15
16         //Create a constructor for the TextView
17         TextView exchangeRateDateView = (TextView)
18 findViewById(R.id.exchangeRateDate);
19
20         //String to Date conversion in Java with dd-MM-yyyy format e.g.
21 "14-09-2011"
22         String dateOfExchange = currenciesList.get(0).getDateOfExchange();
23
24         //Set the date on the TextView
25         exchangeRateDateView.setText(dateOfExchange);
26     }
27 }
```

3.2.5 – LoadingActivity.java

```
1   package com.zacmurphy.currencyconverter;
2
3   import android.app.LoaderManager;
4   import android.content.Context;
5   import android.content.Intent;
6   import android.content.Loader;
7   import android.net.ConnectivityManager;
8   import android.net.NetworkInfo;
9   import android.os.Bundle;
10  import android.support.v7.app.AppCompatActivity;
11  import android.util.Log;
12  import android.view.View;
13  import android.widget.ProgressBar;
14  import android.widget.TextView;
15
16  import java.util.Collections;
17  import java.util.Comparator;
18  import java.util.List;
19
20  import static com.zacmurphy.currencyconverter.Currency.currenciesList;
21
22  public class LoadingActivity extends AppCompatActivity implements
23  LoaderManager.LoaderCallbacks<List<Currency>> {
24
25      //URL that retrieves the JSON response from the API
26      public static final String REQUEST_URL =
27  "https://api.fixer.io/latest?base=GBP";
28      //Tag for the log messages
29      private static final String LOG_TAG =
30  LoadingActivity.class.getSimpleName();
31      //https://api.fixer.io/latest?base=GBP
32      //https://api.fixer.io/latest?symbols=USD,GBP
33      //Constant value for the currency loader ID. This is only really used
34  if you're using multiple loaders.
35      private static final int CURRENCY_LOADER_ID = 1;
36      //Global instance of the ProgressBar, so it can be used in multiple
37  methods in this class
38      private ProgressBar mProgressBar;
39      //Global instance of the information TextView
40      private TextView mInfoView;
41
42      @Override
43      protected void onCreate(Bundle savedInstanceState) {
44          Log.d(LOG_TAG, "LoadingActivity - started");
45          super.onCreate(savedInstanceState);
46          setContentView(R.layout.activity_loading);
47
48          //Hide the system UI
49          hideSystemUI();
50
51          //Initialise the global variables
52          mProgressBar = (ProgressBar) findViewById(R.id.loading_spinner);
53          mInfoView = (TextView) findViewById(R.id.info_view);
54
55          //Check if the user is connected to the Internet
56          if (!isInternetConnected()) {
57              Log.v(LOG_TAG, "No Internet connection");
58              //Hide the loading spinner
59              mProgressBar.setVisibility(View.GONE);
```

```
60
61              //Let the user know there is no connection
62
63  mInfoView.setText(getResources().getText(R.string.error_noConnectivity));
64
65              //Show the system UI
66              showSystemUI();
67          } else {
68              Log.v(LOG_TAG, "Internet connection established");
69
70              //Get a reference to the LoaderManager, in order to interact
71  with loaders.
72              LoaderManager loaderManager = getLoaderManager();
73
74              //Initialize the loader. Pass in the int ID constant defined
75  above and pass in null for
76              //the bundle. Pass in this activity for the LoaderCallbacks
77  parameter (which is valid
78              //because this activity implements the LoaderCallbacks
79  interface).
80              loaderManager.initLoader(CURRENCY_LOADER_ID, null, this);
81          }
82      }
83
84      /**
85       * Method that checks if the user is connected to the Internet
86       *
87       * @return true or false
88       */
89      private boolean isInternetConnected() {
90          Log.d(LOG_TAG, "isInternetConnected() - called");
91          //Create a connectivity manager
92          ConnectivityManager cm = (ConnectivityManager)
93  getSystemService(Context.CONNECTIVITY_SERVICE);
94
95          //Get the active network's network info, and return a boolean value
96          NetworkInfo activeNetwork = cm.getActiveNetworkInfo();
97          return activeNetwork != null &&
98  activeNetwork.isConnectedOrConnecting();
99      }
100
101     /**
102      * Method for the creation of the loader, pass in:
103      * The Loader {@param id} to be used
104      * The {@param args} (arguments) the Loader should take
105      *
106      * @return a new instance of the Loader
107      */
108     @Override
109     public Loader<List<Currency>> onCreateLoader(int i, Bundle bundle) {
110         Log.d(LOG_TAG, "onCreateLoader() - called");
111         return new CurrencyLoader(this);
112     }
113
114     /**
115      * Once the loader has finished, execute this method with:
116      * The {@param loader} to be used
117      * The {@param result} from the Loader creation
118      */
119     @Override
```

```
120         public void onLoadFinished(Loader<List<Currency>> loader,
121     List<Currency> currencies) {
122             Log.d(LOG_TAG, "onLoadFinished() - called");
123             //If there is a valid list of Currency's, then add them to the
124     ArrayList
125             if (currencies != null && !currencies.isEmpty()) {
126                 currenciesList.addAll(currencies);
127                 Log.v(LOG_TAG, "currencies added to ArrayList");
128 
129                 //Re-order the ArrayList, to prioritise the more frequently
130     used currencies to the beginning
131                 Collections.sort(currenciesList, new CurrencyComparator());
132                 Log.v(LOG_TAG, "currenciesList re-ordered");
133             }
134 
135             //Remove the loading spinner and change the on-screen text
136             mProgressBar.setVisibility(View.GONE);
137             Log.d(LOG_TAG, "Error occurred?: " + QueryUtils.ERROR_OCCURRED);
138             if (QueryUtils.ERROR_OCCURRED) {
139 
140     mInfoView.setText(getResources().getText(R.string.error_dataLoading));
141 
142                 //Show the system UI
143                 showSystemUI();
144             } else {
145 
146     mInfoView.setText(getResources().getText(R.string.message_dataLoaded));
147                 //Switch to the next activity
148                 switchActivity();
149             }
150         }
151 
152         /**
153          * Custom method that switches the user to the next activity and closes
154     the main
155          */
156         private void switchActivity() {
157             Log.d(LOG_TAG, "Activity switching");
158             //Create a new Intent object constructor, populate it with
159     MainActivity.class
160             Intent intent = new Intent(this, MainActivity.class);
161             //Make sure the user cannot navigate back to this activity by using
162     the back button
163             intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
164             //Start that new intent
165             startActivity(intent);
166             //Close the current activity
167             finish();
168         }
169 
170         /**
171          * If the Loader is reset (i.e. through orientation change), handle
172     that in this method
173          * The {@param loader} to be used
174          */
175         @Override
176         public void onLoaderReset(Loader<List<Currency>> loader) {
177             Log.d(LOG_TAG, "onLoaderReset() - called");
178         }
179 
180         /**
```

```
181         * Method that hides all system UI, making the activity truly
182     fullscreen
183         * Uses code sample from the Android Dev. documentation
184         */
185        private void hideSystemUI() {
186            Log.d(LOG_TAG, "hideSystemUI - called");
187            // Set the IMMERSIVE flag.
188            // Set the content to appear under the system bars so that the
189     content
190            // doesn't resize when the system bars hide and show.
191            getWindow().getDecorView().setSystemUiVisibility(
192                    View.SYSTEM_UI_FLAG_LAYOUT_STABLE
193                            | View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION
194                            | View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN
195                            | View.SYSTEM_UI_FLAG_HIDE_NAVIGATION // hide nav
196     bar
197                            | View.SYSTEM_UI_FLAG_FULLSCREEN // hide status bar
198                            | View.SYSTEM_UI_FLAG_IMMERSIVE);
199        }
200
201        /**
202         * Method that shows all system UI, returning the activity from
203     fullscreen
204         * Uses code sample from the Android Dev. documentation
205         */
206        private void showSystemUI() {
207            Log.d(LOG_TAG, "showSystemUI - called");
208            // This snippet shows the system bars. It does this by removing all
209     the flags
210            // except for the ones that make the content appear under the
211     system bars.
212            getWindow().getDecorView().setSystemUiVisibility(
213                    View.SYSTEM_UI_FLAG_LAYOUT_STABLE
214                            | View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION
215                            | View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN);
216        }
217
218        /**
219         * Sub-class that sorts the ArrayList into the order specified by the
220     item's priority,
221         * when the items have the same priority, they are left in alphabetical
222     order by country code
223         */
224        private class CurrencyComparator implements Comparator<Currency> {
225            public int compare(Currency left, Currency right) {
226                Log.d(LOG_TAG, "compare class - called");
227                return Integer.compare(left.getPriority(),
228     right.getPriority());
229            }
230        }
231    }
```

3.2.6 – MainActivity.java

```
1   package com.zacmurphy.currencyconverter;
2
3   import android.content.Intent;
4   import android.graphics.Typeface;
5   import android.os.Bundle;
6   import android.support.v7.app.AppCompatActivity;
7   import android.util.Log;
8   import android.view.KeyEvent;
9   import android.view.Menu;
10  import android.view.MenuInflater;
11  import android.view.MenuItem;
12  import android.view.View;
13  import android.view.inputmethod.EditorInfo;
14  import android.widget.AdapterView;
15  import android.widget.ArrayAdapter;
16  import android.widget.Button;
17  import android.widget.EditText;
18  import android.widget.Spinner;
19  import android.widget.TextView;
20
21  import static com.zacmurphy.currencyconverter.Currency.currenciesList;
22  import static
23  com.zacmurphy.currencyconverter.R.id.amountToConvertEntryField;
24  import static com.zacmurphy.currencyconverter.R.id.conversionResultField;
25
26  public class MainActivity extends AppCompatActivity {
27
28      //Tag for the log messages
29      private static final String LOG_TAG =
30  MainActivity.class.getSimpleName();
31
32      //Global instance of the Spinner, so it can be used in multiple methods
33  in this class
34      private Spinner mSpinner;
35
36      //Global instance of the EditText, so it can be used in multiple
37  methods in this class
38      private EditText mAmountToConvertEntryField;
39
40      //Global instance of the results field TextView, so it can be used in
41  multiple methods in this class
42      private TextView mConversionResultField;
43
44      @Override
45      protected void onCreate(Bundle savedInstanceState) {
46          super.onCreate(savedInstanceState);
47          setContentView(R.layout.activity_main);
48
49          //Initialise global variables
50          mSpinner = (Spinner) findViewById(R.id.currency_picker);
51          mAmountToConvertEntryField = (EditText)
52  findViewById(amountToConvertEntryField);
53          mConversionResultField = (TextView)
54  findViewById(conversionResultField);
55
56          //Set the title of the title bar to be more appropriate
57          setTitle(getResources().getText(R.string.page_1));
58
59          //Set the typeface to the EditText
```

```
60          mAmountToConvertEntryField.setTypeface(Typeface.SANS_SERIF);
61
62          //Call this method to initiate the creation of the Spinner
63          createSpinnerOptions();
64
65          //Get the spinner as an object and call the OnItemSelectedListener
66  on it to create the listener
67          mSpinner.setOnItemSelectedListener(new MyOnItemSelectedListener());
68          Log.v(LOG_TAG, "Listener set on spinner");
69
70          //Create a new button object constructor
71          Button button = (Button) findViewById(R.id.button_convert);
72          //Set a listener to it and create a new method
73          button.setOnClickListener(new View.OnClickListener() {
74              public void onClick(View v) {
75                  //Do this when the button is clicked
76                  Log.v(LOG_TAG, "Convert button click was registered");
77                  onConvertClick();
78              }
79          });
80          Log.v(LOG_TAG, "Listener set on 'Convert' button");
81
82          //Set a listener on the EditText
83          mAmountToConvertEntryField.setOnClickListener(new
84  View.OnClickListener() {
85              public void onClick(View v) {
86                  if (v.getId() == mAmountToConvertEntryField.getId()) {
87                      //Make the cursor visible
88                      mAmountToConvertEntryField.setCursorVisible(true);
89                  }
90              }
91          });
92          Log.v(LOG_TAG, "Listener set on EditText");
93
94          //Set a listener on the done button on the keyboard
95          mAmountToConvertEntryField.setOnEditorActionListener(new
96  TextView.OnEditorActionListener() {
97              @Override
98              public boolean onEditorAction(TextView v, int actionId,
99  KeyEvent event) {
100                 if (actionId == EditorInfo.IME_ACTION_DONE) {
101                     //Run the calculations process
102                     Log.v(LOG_TAG, "Done button click was registered");
103                     onConvertClick();
104
105                     //Hide the cursor
106                     mAmountToConvertEntryField.setCursorVisible(false);
107                 }
108                 return false;
109             }
110         });
111         Log.v(LOG_TAG, "Listener set on 'Done' button");
112     }
113
114     /**
115      * The method that creates the spinner options
116      */
117     private void createSpinnerOptions() {
118         Log.d(LOG_TAG, "createSpinnerOptions - called");
119         //Create an ArrayAdapter based on the CurrencyAdapter using the
120 list of currencies
```

```
121             ArrayAdapter adapter = new CurrencyAdapter(this);
122
123             //Apply the adapter to the spinner
124             mSpinner.setAdapter(adapter);
125         }
126
127         /**
128          * Custom method that handles what happens when the convert button is
129     clicked
130          */
131         private void onConvertClick() {
132             Log.d(LOG_TAG, "onConvertClick - called");
133             //Hide the cursor
134             mAmountToConvertEntryField.setCursorVisible(false);
135
136             //Get the country code of the currently selected spinner option
137             String currentCountryCode = getSpinnerOption();
138
139             //Get the relevant symbol for the currency
140             String relevantCurrencySymbol =
141     getCurrencySymbol(currentCountryCode);
142
143             //Perform the required calculations
144             double exchangedValue = getExchangedValue();
145
146             //Change the text to display the correct values
147             setResultFieldText(relevantCurrencySymbol, exchangedValue);
148         }
149
150         /**
151          * Custom method
152          *
153          * @return the currently selected option of the spinner
154          */
155         private String getSpinnerOption() {
156             Log.d(LOG_TAG, "getSpinnerOption - called");
157             //Get the currently selected item's position
158             int position = mSpinner.getSelectedItemPosition();
159
160             Log.v(LOG_TAG, "Selected item: " +
161     currenciesList.get(position).getConvertedCurrency());
162             //Use that position to look up the respective item in the list and
163     return the corresponding country code
164             return currenciesList.get(position).getConvertedCurrency();
165         }
166
167         /**
168          * Custom method that takes in the {@param countryCode}
169          *
170          * @return the relevant currency symbol associated with the countryCode
171          */
172         private String getCurrencySymbol(String countryCode) {
173             Log.d(LOG_TAG, "getCurrencySymbol - called");
174             //Declare private variables
175             String symbol;
176             int symbolId;
177
178             //Append the country code to the prefix "symbol_" to get the string
179     name, specify the resource type and package name - which give the string
180     resource
```

```
181             symbolId =
182     getApplicationContext().getResources().getIdentifier("symbol_" +
183     countryCode, "string", getPackageName());
184
185             //As long as there is an existing string resource available
186             if (symbolId != 0) {
187                 //Get the resource
188                 symbol = getResources().getString(symbolId);
189             } else {
190                 //Assign the country code
191                 symbol = countryCode;
192             }
193
194             Log.v(LOG_TAG, "symbol: " + symbol);
195             return symbol;
196         }
197
198         /**
199          * Custom method, using the exchange rate and user entered value
200          *
201          * @return the exchanged value for the user
202          */
203         private double getExchangedValue() {
204             Log.d(LOG_TAG, "getExchangedValue - called");
205             //Get the values from the EditText
206             String userEnteredQuantity =
207     mAmountToConvertEntryField.getText().toString();
208             Log.v(LOG_TAG, "userEnteredQuantity: " + userEnteredQuantity +
209     "|END");
210
211             double valueToConvert;
212             //Check if the user input is blank
213             if (userEnteredQuantity.isEmpty()) {
214                 valueToConvert = 0;
215             } else {
216                 if (userEnteredQuantity.equals(".")) {
217                     valueToConvert = 0;
218                     //Notify the user of their error
219
220     mAmountToConvertEntryField.setError(getString(R.string.error_invalidChar));
221                 } else {
222                     valueToConvert = Double.parseDouble(userEnteredQuantity);
223                 }
224             }
225             Log.v(LOG_TAG, "valueToConvert: " + valueToConvert);
226
227             //Get the currently selected item's position
228             int position = mSpinner.getSelectedItemPosition();
229
230             //Get the exchange rate for the particular currency
231             double exchangeRate =
232     currenciesList.get(position).getExchangeRate();
233             Log.v(LOG_TAG, "exchangeRate: " + exchangeRate);
234
235             Log.v(LOG_TAG, "exchangedValue: " + valueToConvert * exchangeRate);
236             return valueToConvert * exchangeRate;
237         }
238
239         /**
240          * Custom method that outputs the result from the conversion
241          *
```

```
242         * @param relevantCurrencySymbol input of the relevant currency symbol
243     to use
244         * @param exchangedValue         input of the exchanged value to show
245     the user
246         */
247        private void setResultFieldText(String relevantCurrencySymbol, double
248     exchangedValue) {
249             Log.d(LOG_TAG, "setResultFieldText - called");
250 
251             //Set the text based on the value being input
252             if (exchangedValue == 0.0) {
253                 mConversionResultField.setText(relevantCurrencySymbol);
254             } else {
255 
256     mConversionResultField.setText(getString(R.string.content_convertedValueTex
257     t, relevantCurrencySymbol, exchangedValue));
258             }
259         }
260 
261         /**
262          * Create the options menu in the action bar
263          * using {@param menu} as a resource
264          */
265         @Override
266         public boolean onCreateOptionsMenu(Menu menu) {
267             Log.d(LOG_TAG, "onCreateOptionsMenu - called");
268             MenuInflater inflater = getMenuInflater();
269             inflater.inflate(R.menu.main_menu, menu);
270             return true;
271         }
272 
273         /**
274          * Handle item selections of the action bar
275          */
276         @Override
277         public boolean onOptionsItemSelected(MenuItem item) {
278             Log.d(LOG_TAG, "onOptionsItemSelected - called");
279             //Switch statement
280             switch (item.getItemId()) {
281     //              //When the list icon is selected:
282     //              case R.id.action_conversionNav:
283     //                  //Open the second page
284     //                  changeToSavedConversions();
285     //                  return true;
286 
287                 //When the refresh icon is selected:
288                 case R.id.action_resetContent:
289                     Log.v(LOG_TAG, "Reset the page contents");
290                     //Refresh the app
291                     resetContent();
292                     return true;
293 
294                 //When the help icon is selected:
295                 case R.id.action_help:
296                     Log.v(LOG_TAG, "Nav to help");
297                     //Open the help page
298                     changeToHelpPage();
299                     return true;
300 
301                 //When the refresh option is selected:
302                 case R.id.action_refresh:
```

```
303                 Log.v(LOG_TAG, "Update conversion rates");
304                 //Refresh the conversion rates
305                 refreshConversionRates();
306                 return true;
307 
308             default:
309                 // The user's action was not recognized. Invoke the
310 superclass to handle it.
311                 return super.onOptionsItemSelected(item);
312         }
313     }
314 
315     /**
316      * Method that navigates the user to the help page, using an intent
317      */
318     private void changeToHelpPage() {
319         Log.d(LOG_TAG, "changeToHelpPage - called");
320         //Create a new Intent object constructor, populate it with
321 HelpActivity.class
322         Intent intent = new Intent(this, HelpActivity.class);
323         //Start that new intent
324         startActivity(intent);
325     }
326 
327 //    /**
328 //     * Method that navigates the user to the second page, using an intent
329 //     */
330 //    private void changeToSavedConversions() {
331 //        //Create a new Intent object constructor, populate it with
332 SecondActivity.class
333 //        Intent intent = new Intent(this, SecondActivity.class);
334 //        //Start that new intent
335 //        startActivity(intent);
336 //    }
337 
338     /**
339      * Custom method, reset the changed fields
340      */
341     private void resetContent() {
342         Log.d(LOG_TAG, "refreshApp - called");
343         //Set the EditText field to be empty
344         mAmountToConvertEntryField.setText(null);
345 
346         //Reset the Spinner value the default currency
347         mSpinner.setSelection(0);
348 
349         //Reset the values in the results field
350 
351 mConversionResultField.setText(getCurrencySymbol(getSpinnerOption()));
352 
353         //Hide the cursor
354         mAmountToConvertEntryField.setCursorVisible(false);
355     }
356 
357     /**
358      * Custom method that refreshes the currency rates
359      */
360     private void refreshConversionRates() {
361         Log.d(LOG_TAG, "refreshConversionRates - called");
362         //Clear the existing currencies, so duplicates are not produced
363         currenciesList.clear();
```

```
364
365             //Create a new Intent object constructor, populate it with
366     MainActivity.class
367             Intent intent = new Intent(this, LoadingActivity.class);
368             //Make sure the user cannot navigate back to this activity by using
369     the back button
370             intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
371             //Start that new intent
372             startActivity(intent);
373             //Close the current activity
374             finish();
375         }
376
377         /**
378          * This sub-class controls what happens when the listener detects
379     something has been selected
380          */
381         private class MyOnItemSelectedListener implements
382     AdapterView.OnItemSelectedListener {
383             public void onItemSelected(AdapterView<?> parent, View view, int
384     pos, long id) {
385                 Log.d(LOG_TAG, "Spinner item selected");
386                 //When an item is selected, do this:
387                 onConvertClick();
388             }
389
390             public void onNothingSelected(AdapterView parent) {
391                 Log.d(LOG_TAG, "no spinner options selected");
392                 // Do nothing when nothing is selected.
393             }
394         }
395     }
```

3.2.7 – QueryUtils.java

```
1  package com.zacmurphy.currencyconverter;
2
3  import android.text.TextUtils;
4  import android.util.Log;
5
6  import org.json.JSONException;
7  import org.json.JSONObject;
8
9  import java.io.BufferedReader;
10 import java.io.IOException;
11 import java.io.InputStream;
12 import java.io.InputStreamReader;
13 import java.net.HttpURLConnection;
14 import java.net.MalformedURLException;
15 import java.net.URL;
16 import java.nio.charset.Charset;
17 import java.util.ArrayList;
18 import java.util.List;
19
20 /**
21  * Helper methods related to requesting and receiving currency data from
22 Fixer.IO
23  */
24 class QueryUtils {
25
26     //Tag for the log messages
27     private static final String LOG_TAG = QueryUtils.class.getSimpleName();
28
29     //Public Boolean for error detection during the data loading process
30     static boolean ERROR_OCCURRED;
31
32     /**
33      * Query Fixer.IO and return an {@link List<Currency>} object to
34 represent a single exchange rate.
35      */
36     public static List<Currency> fetchCurrencyData(String requestUrl) {
37         Log.d(LOG_TAG, "fetchCurrencyData - called");
38         //Set the variable to false, until an error occurs and it is set to
39 true
40         ERROR_OCCURRED = false;
41
42         //Create a URL object
43         URL url = createUrl(requestUrl);
44         Log.v(LOG_TAG, "url: " + url);
45
46         //Perform a HTTP request to the URL and receive a JSON response
47 back
48         //Initialise variable
49         String jsonResponse = null;
50
51         //Try the creation of a JSON response from the URL, if not catch
52 the exception
53         try {
54             jsonResponse = makeHttpRequest(url);
55         } catch (IOException e) {
56             ERROR_OCCURRED = true;
57             //Log the error
58             Log.e(LOG_TAG, "Error closing input stream", e);
59         }
```

```
60
61          //Extract and return the relevant fields from the JSON response and
62  create a List<Currency> object
63          return extractFeaturesFromJson(jsonResponse);
64      }
65
66      /**
67       * @return a new URL object from the given string URL.
68       */
69      private static URL createUrl(String stringUrl) {
70          Log.d(LOG_TAG, "createUrl - called");
71          //Initialise variable
72          URL url = null;
73
74          //Try the creation of a URL, if not catch the exception
75          try {
76              url = new URL(stringUrl);
77          } catch (MalformedURLException e) {
78              ERROR_OCCURRED = true;
79              //Log the error
80              Log.e(LOG_TAG, "Error with creating URL", e);
81          }
82          Log.v(LOG_TAG, "url: " + url);
83          //Return the successful URL
84          return url;
85      }
86
87      /**
88       * Make an HTTP request to the given URL and return a String as the
89  response.
90       *
91       * @throws IOException if something goes wrong
92       */
93      private static String makeHttpRequest(URL url) throws IOException {
94          Log.d(LOG_TAG, "makeHttpRequest - called");
95          //Initialise variable
96          String jsonResponse = "";
97
98          //If the URL is null, then return early with an empty string
99          if (url == null) {
100             ERROR_OCCURRED = true;
101             Log.v(LOG_TAG, "URL is null");
102             return jsonResponse;
103         }
104
105         //Initialise variables
106         HttpURLConnection urlConnection = null;
107         InputStream inputStream = null;
108
109         //Try connecting to the URL and reading the response, else catch
110 the exception
111         try {
112             //Open the connection to the URL
113             urlConnection = (HttpURLConnection) url.openConnection();
114             Log.d(LOG_TAG, "urlConnection opened");
115
116             //Set timeouts so if the connections expire before data is
117 available, an error is thrown
118             //Makes ure the user is never waiting for long periods of time
119 if there is no data
120             urlConnection.setReadTimeout(10000 /* milliseconds */);
```

```
121             urlConnection.setConnectTimeout(15000 /* milliseconds */);
122             Log.d(LOG_TAG, "timeouts set");
123 
124             //Set the request method, as data wants to be retrieved
125             urlConnection.setRequestMethod("GET");
126             Log.d(LOG_TAG, "request method set");
127 
128             //Make the connection
129             urlConnection.connect();
130             Log.d(LOG_TAG, "url connection connecting");
131 
132             //If the request was successful (response code 200), then read
133 the input and parse the response
134             if (urlConnection.getResponseCode() == 200) {
135                 Log.v(LOG_TAG, "Response code was 200");
136                 inputStream = urlConnection.getInputStream();
137                 jsonResponse = readFromStream(inputStream);
138                 Log.v(LOG_TAG, "jsonResponse: " + jsonResponse);
139             } else {
140                 ERROR_OCCURRED = true;
141                 //Log the error
142                 Log.e(LOG_TAG, "Error response code: " +
143 urlConnection.getResponseCode());
144             }
145         } catch (IOException e) {
146             ERROR_OCCURRED = true;
147             //Log the error
148             Log.e(LOG_TAG, "Problem retrieving the JSON results", e);
149         } finally {
150             //As long as there is an open connection
151             if (urlConnection != null) {
152                 //Close the URL connection
153                 urlConnection.disconnect();
154                 Log.v(LOG_TAG, "urlConnection disconnected");
155             }
156             //As long as the inputStream is open
157             if (inputStream != null) {
158                 //Close the inputStream
159                 inputStream.close();
160                 Log.v(LOG_TAG, "inputStream closed");
161             }
162         }
163         //Return the JSON response
164         Log.v(LOG_TAG, "jsonResponse: " + jsonResponse);
165         return jsonResponse;
166     }
167 
168     /**
169      * Convert the {@link InputStream} into a String which contains the
170      * whole JSON response from the server.
171      */
172     private static String readFromStream(InputStream inputStream) throws
173 IOException {
174         Log.d(LOG_TAG, "readFromStream - called");
175         //Initialise variables / create a new String Builder (S.B.)
176         //S.B. builds one String bit by bit instead of appending more
177 information each time through other variables
178         StringBuilder output = new StringBuilder();
179 
180         //As long as the inputStream is not empty
181         if (inputStream != null) {
```

```
182             Log.v(LOG_TAG, "inputStream is not null");
183             //Create a new InputStreamReader, using the inputStream and
184 "UTF-8" character set
185             InputStreamReader inputStreamReader = new
186 InputStreamReader(inputStream, Charset.forName("UTF-8"));
187 
188             //Create a buffered reader, which stores information about the
189 characters around each character
190             BufferedReader reader = new BufferedReader(inputStreamReader);
191 
192             //Get the line of text
193             String line = reader.readLine();
194 
195             //While the line isn't empty
196             while (line != null) {
197                 //Append the line to the output
198                 output.append(line);
199                 //Goto the next line
200                 line = reader.readLine();
201             }
202         }
203         //Return the completed output in String format
204         return output.toString();
205     }
206 
207     /**
208      * @return an {@link List<Currency>} item by parsing out information
209      * about the exchange rates from the input jsonResponse string.
210      */
211     private static List<Currency> extractFeaturesFromJson(String
212 jsonResponse) {
213         Log.d(LOG_TAG, "extractFeaturesFromJson - called");
214         //If the JSON String is empty or null, then return early
215         if (TextUtils.isEmpty(jsonResponse)) {
216             ERROR_OCCURRED = true;
217             Log.v(LOG_TAG, "JSON string is empty");
218             return null;
219         }
220 
221         //Create an empty ArrayList that we can start adding exchange rates
222 to
223         List<Currency> currencies = new ArrayList<>();
224         //List<Currency> currencies = Currency.getArrayList();
225 
226         //Try to parse the jsonResponse, else catch the JSONException
227         try {
228             //Get the root of the JSON string
229             JSONObject root = new JSONObject(jsonResponse);
230             Log.v(LOG_TAG, "root: " + root.toString());
231 
232             //Get the base currency
233             String baseCurrency = root.getString("base");
234             Log.v(LOG_TAG, "baseCurrency: " + baseCurrency);
235 
236             //Get the date of the exchange rate
237             String exchangeDate = root.getString("date");
238             Log.v(LOG_TAG, "exchangeDate: " + exchangeDate);
239 
240             //Get the "rates" array as an object
241             JSONObject rates = root.getJSONObject("rates");
242 
```

```
243                 //For loop that cycles through all of the rates objects
244                 for (int i = 0; i < rates.length(); i++) {
245                     //Get the object key (country code)
246                     String objectKey = rates.names().getString(i);
247 
248                     //Get the exchange rate value associated with that object
249                     double keyValue = rates.getDouble(objectKey);
250 
251                     //Get the priority for each object
252                     int priority = getPriority(objectKey);
253 
254                     //Add those details along with the above date and base to
255 the array
256                     Log.v(LOG_TAG, exchangeDate + ", " + baseCurrency + ", " +
257 objectKey + ", " + keyValue + ", " + priority);
258                     currencies.add(new Currency(exchangeDate, baseCurrency,
259 objectKey, keyValue, priority));
260                 }
261             } catch (JSONException e) {
262                 ERROR_OCCURRED = true;
263                 //Log the error
264                 Log.e(LOG_TAG, "Problems parsing the currency JSON results",
265 e);
266             }
267             return currencies;
268         }
269 
270         /**
271          * Custom method that assigns a priority to a currency based on how
272 used it is worldwide,
273          * this I determined using data from the Internet
274          *
275          * @param currencyCode, the country to be checked
276          * @return the priority level
277          */
278         private static int getPriority(String currencyCode) {
279             Log.d(LOG_TAG, "getPriority - called");
280             switch (currencyCode) {
281                 case "USD":
282                     return 0;
283                 case "EUR":
284                     return 1;
285                 case "JPY":
286                     return 2;
287                 case "AUD":
288                     return 3;
289                 case "CAD":
290                     return 4;
291                 case "CHF":
292                     return 5;
293                 case "CNY":
294                     return 6;
295                 default:
296                     return 10;
297             }
298         }
299 }
```

3.3 – XML files

3.3.1 – activity_help.xml

```
<?xml version="1.0" encoding="utf-8"?><!--Having the main layout as a
ScrollView allows the app the automatically provide the user with
scrolling capabilities if the contents overflow the screen.-->
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.zacmurphy.currencyconverter.HelpActivity">

    <!--The main structure for the content on the page, to be structured in
a vertical order-->
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginLeft="16dp"
        android:layout_marginRight="16dp"
        android:layout_marginTop="8dp"
        android:orientation="vertical">

        <!--First heading-->
        <TextView
            style="@style/HeadingLarge"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/heading_aboutApp" />

        <!--First content-->
        <TextView
            style="@style/Content"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:text="@string/content_aboutApp" />

        <!--Date of exchange rates being used-->
        <TextView
            android:id="@+id/exchangeRateDate"
            style="@style/Content"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="4dp" />

        <View
            android:layout_width="match_parent"
            android:layout_height="2dp"
            android:layout_marginTop="8dp"
            android:background="@color/colorAccent" />

        <!--Second heading-->
        <TextView
            style="@style/HeadingLarge"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
```

```
57              android:text="@string/heading_useOfApp" />
58
59          <!--Second content-->
60          <TextView
61              style="@style/Content"
62              android:layout_width="match_parent"
63              android:layout_height="wrap_content"
64              android:layout_marginTop="8dp"
65              android:text="@string/content_directionsOfUse" />
66      </LinearLayout>
67  </ScrollView>
```

3.3.2 – activity_loading.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:tools="http://schemas.android.com/tools"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      tools:context="com.zacmurphy.currencyconverter.LoadingActivity">
7
8      <!--Progress bar to let the user know that data is being fetched-->
9      <ProgressBar
10         android:id="@+id/loading_spinner"
11         style="@style/Widget.AppCompat.ProgressBar"
12         android:layout_width="wrap_content"
13         android:layout_height="wrap_content"
14         android:layout_centerInParent="true" />
15
16     <!--TextView to display information to the user-->
17     <TextView
18         android:id="@+id/info_view"
19         style="@style/Notification"
20         android:layout_width="match_parent"
21         android:layout_height="wrap_content"
22         android:layout_above="@id/loading_spinner"
23         android:layout_centerHorizontal="true"
24         android:layout_centerInParent="true"
25         android:layout_margin="8dp"
26         android:text="@string/message_loadingData"
27         android:textAlignment="center" />
28 </RelativeLayout>
```

3.3.3 – activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?><!--This relative layout allows for
the main functions to be at the top and the save conversion
button at the bottom-->
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/mainLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.zacmurphy.currencyconverter.MainActivity">

    <!--This linear layout structures the page contents in a vertical
order-->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginLeft="16dp"
        android:layout_marginRight="16dp"
        android:layout_marginTop="8dp"
        android:orientation="vertical">

        <!--This linear layout structures the top line of contents in a
horizontal order-->
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal">

            <!--Image label for the EditText-->
            <ImageView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:background="@drawable/flag_gbp"
                android:contentDescription="@string/accessibility_gbpFlag"
/>

            <!--Label for the EditText-->
            <TextView
                style="@style/HeadingSmall"
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:layout_marginLeft="8dp"
                android:gravity="center_vertical"
                android:text="@string/symbol_GBP" />

            <!--The user input field-->
            <EditText
                android:id="@+id/amountToConvertEntryField"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginLeft="4dp"
                android:cursorVisible="false"
                android:hint="@string/prompt_amountEntry"
                android:inputType="numberDecimal" />
        </LinearLayout>

        <!--Label for the spinner-->
        <TextView
```

```
            style="@style/HeadingSmall"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="16dp"
            android:text="@string/heading_currencyToConvert" />

        <!--The spinner requires a frame, so that a style can be applied to
it-->
        <FrameLayout
            android:id="@+id/spinnerBox"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:background="@drawable/bg_spinner">

            <!--Drop-down box-->
            <Spinner
                android:id="@+id/currency_picker"
                android:layout_width="match_parent"
                android:layout_height="wrap_content" />
        </FrameLayout>

        <!--The convert button-->
        <Button
            android:id="@+id/button_convert"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="8dp"
            android:text="@string/button_convert"
            android:theme="@style/AppTheme.Button" />

        <!--Heading for the amount converted text-->
        <TextView
            style="@style/HeadingSmall"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="16dp"
            android:gravity="center_horizontal"
            android:text="@string/heading_amountConverted" />

        <!--Text-box that displays converted amount-->
        <TextView
            android:id="@+id/conversionResultField"
            style="@style/ContentLarge"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:gravity="center_horizontal"
            android:text="@string/symbol_GBP" />
    </LinearLayout>

    <!--Save the current conversion button-->
    <!--<Button-->
    <!--android:id="@+id/button_saveConversion"-->
    <!--android:layout_width="wrap_content"-->
    <!--android:layout_height="wrap_content"-->
    <!--android:layout_alignParentBottom="true"-->
    <!--android:layout_centerInParent="true"-->
    <!--android:layout_marginBottom="16dp"-->
    <!--android:text="@string/button_saveConversion" />-->
```

```
</RelativeLayout>
```

3.3.4 – AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.zacmurphy.currencyconverter">

    <!-- Permissions -->
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"
/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:configChanges="orientation|keyboardHidden"
            android:screenOrientation="portrait"
            android:windowSoftInputMode="stateAlwaysHidden|adjustResize" />
        <!-- <activity -->
        <!-- android:name=".SecondActivity" -->
        <!-- android:label="@string/page_2" -->
        <!-- android:parentActivityName=".MainActivity" /> -->
        <activity
            android:name=".HelpActivity"
            android:configChanges="orientation|keyboardHidden"
            android:label="@string/page_3"
            android:parentActivityName=".MainActivity"
            android:screenOrientation="portrait" />
        <activity
            android:name=".LoadingActivity"
            android:theme="@style/FullScreen">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```

3.3.5 – bg_spinner.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:bottom="8dp"
        android:top="8dp">
        <shape>
            <!--Use this as the background colour of the Spinner-->
            <solid android:color="@android:color/white" />

            <!--This controls how rounded the corners are-->
            <corners android:radius="2dp" />

            <!--This creates the border width and colour-->
            <stroke
                android:width="1dp"
                android:color="@color/divider" />

            <!--Use this to control how large the box should be-->
            <!--'Top/Bottom' control height, should be equal for text to be
centered-->
            <!--'Left' controls how far right the text is pushed-->
            <!--'Right' controls how far left the arrow is pushed-->
            <!--NOTE: with no right padding, the drop-arrow will resume
default padding-->
            <padding
                android:bottom="16dp"
                android:left="8dp"
                android:right="8dp"
                android:top="16dp" />
        </shape>
    </item>
</layer-list>
```

3.3.6 – colors.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <color name="colorPrimary">#d32f2f</color>
4      <color name="colorPrimaryDark">#9a0007</color>
5      <color name="colorAccent">#7bb241</color>
6      <color name="primaryText">#212121</color>
7      <color name="secondaryText">#757575</color>
8      <color name="divider">#BDBDBD</color>
9  </resources>
```

3.3.7 – main_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <!--Refresh icon, resets the layout for the user-->
    <item
        android:id="@+id/action_resetContent"
        android:icon="@drawable/ic_refresh_white_24dp"
        android:title="@string/action_resetContent"
        app:showAsAction="ifRoom" />

    <!--"Saved Conversions" page navigation button, should appear as button
if possible-->
    <!--<item-->
    <!--android:id="@+id/action_conversionNav"-->
    <!--android:icon="@drawable/ic_list_white_24dp"-->
    <!--android:title="@string/action_navigate"-->
    <!--app:showAsAction="ifRoom" />-->

    <!--Help button-->
    <item
        android:id="@+id/action_help"
        android:icon="@drawable/ic_help_white_24dp"
        android:title="@string/action_help"
        app:showAsAction="always" />

    <!--Refresh exchange rates action, retrieves latest figures-->
    <item
        android:id="@+id/action_refresh"
        android:title="@string/action_refresh"
        app:showAsAction="never" />
</menu>
```

3.3.8 – spinner_dropdown_item.xml

```
<?xml version="1.0" encoding="utf-8"?><!--This XML layout file controls the
styling for the text inside the dropdown menu--><!--Use a CheckedTextView
for this XML file--><!--Make sure that attributes 'id, style, ellipsize,
maxLines' are all included--><!--Everything else can be customised--><!--
48dp is the recommended height for a button/selection-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <ImageView
        android:id="@+id/spinnerDropDownImage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginLeft="8dp"
        android:contentDescription="@string/accessibility_flagIcon"
        tools:background="@drawable/flag_gbp" />

    <CheckedTextView
xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/spinnerDropDownText"
        style="@style/Dropdown"
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:layout_marginLeft="8dp"
        android:ellipsize="marquee"
        android:maxLines="1"
        tools:text="America" />
</LinearLayout>
```

3.3.9 – spinner_item.xml

```
<?xml version="1.0" encoding="utf-8"?><!--This XML layout file controls the
styling for the text once selected from the spinner--><!--Using a TextView
works fine for this XML file--><!--Make sure that attribute 'id' is
included--><!--Everything else can be customised--><!--Keep height to
'wrap_content' to keep the height defined by the background XML-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="4dp"
    android:layout_marginRight="8dp"
    android:layout_marginTop="4dp"
    android:orientation="horizontal">

    <ImageView
        android:id="@+id/spinnerImage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:contentDescription="@string/accessibility_flagIcon"
        tools:background="@drawable/flag_gbp" />

    <TextView
        android:id="@+id/spinnerText"
        style="@style/Content"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginLeft="8dp"
        android:textSize="14sp"
        tools:text="GBP" />

</LinearLayout>
```

3.3.10 – strings.xml

```
<resources xmlns:xliff="urn:oasis:names:tc:xliff:document:1.2">
    <!--Page names-->
    <string name="app_name">Currency Converter</string>
    <string name="page_1">Converter</string>
    <!--<string name="page_2">Saved Conversions</string>-->
    <string name="page_3">Help &amp; About</string>

    <!--Content headings-->
    <string name="heading_currencyToConvert">Currency to convert
to</string>
    <string name="heading_amountConverted">Amount</string>
    <string name="heading_aboutApp">About the app</string>
    <string name="heading_useOfApp">Directions for use</string>
    <string name="heading_baseCurrency">GBP £</string>

    <!--Page content-->
    <string name="content_aboutApp">This currency converter app uses the
latest exchange rates provided by the European Central Bank, which are
updated at around 15:00 GMT every day - excluding weekends and select
holidays throughout the year.\nCurrently, the app is using exchange rates
from:</string>
    <string name="content_directionsOfUse">To use the converter, simply tap
on the user-entry field on the main page and enter an amount to convert in
British Pounds.\n\nThen select a currency to convert to from the drop-down
list and press the Convert button.\n\nTo reset the app ready for another
conversion, select the Clear Results icon at the top, next to the help
button.\n\nTo retrieve the latest exchange rates, go to the overflow menu
and select Update Exchange Rates. </string>
    <string name="content_convertedValueText"><xliff:g example="£"
id="relevantCurrencySymbol">%s</xliff:g> <xliff:g example="1"
id="exchangedValue">%f</xliff:g></string>

    <!--Actions, button & prompt labels-->
    <string name="action_refresh">Update exchange rates</string>
    <!--<string name="action_navigate">Saved conversions</string>-->
    <string name="action_help">Help &amp; about</string>
    <string name="action_resetContent">Clear results</string>
    <string name="button_convert">Convert</string>
    <!--<string name="button_saveConversion">Save Conversion</string>-->
    <string name="prompt_amountEntry">Enter amount to convert</string>

    <!--Error messages & notifications-->
    <string name="error_noConnectivity">No Internet connection.</string>
    <string name="error_noDataResults">No exchange rates found.</string>
    <string name="error_dataLoading">An error occurred - please try again
later.</string>
    <string name="error_invalidChar">Enter a number</string>
    <string name="message_loadingData">Loading data, please wait a
moment.</string>
    <string name="message_dataLoaded">Data successfully retrieved.</string>

    <!--Symbols for currencies-->
    <string name="symbol_AUD">$</string>
    <string name="symbol_BGN">лв</string>
    <string name="symbol_BRL">R$</string>
    <string name="symbol_CAD">$</string>
    <string name="symbol_CNY">¥</string>
    <string name="symbol_CZK">Kč</string>
    <string name="symbol_DKK">kr</string>
```

```
60        <string name="symbol_EUR">€</string>
61        <string name="symbol_GBP">£</string>
62        <string name="symbol_HKD">$</string>
63        <string name="symbol_HRK">kn</string>
64        <string name="symbol_HUF">Ft</string>
65        <string name="symbol_IDR">Rp</string>
66        <string name="symbol_ILS">₪</string>
67        <string name="symbol_INR">₹</string>
68        <string name="symbol_ISK">kr</string>
69        <string name="symbol_JPY">¥</string>
70        <string name="symbol_KRW">₩</string>
71        <string name="symbol_MXN">$</string>
72        <string name="symbol_MYR">RM</string>
73        <string name="symbol_NOK">kr</string>
74        <string name="symbol_NZD">$</string>
75        <string name="symbol_PHP">₱</string>
76        <string name="symbol_PLN">zł</string>
77        <string name="symbol_RON">lei</string>
78        <string name="symbol_RUB">₽</string>
79        <string name="symbol_SEK">kr</string>
80        <string name="symbol_SGD">$</string>
81        <string name="symbol_THB">฿</string>
82        <string name="symbol_TRY">₺</string>
83        <string name="symbol_USD">$</string>
84        <string name="symbol_ZAR">R</string>
85
86        <string name="accessibility_gbpFlag">GBP Flag</string>
87        <string name="accessibility_flagIcon">Currency flag icon</string>
88    </resources>
```

3.3.11 – styles.xml

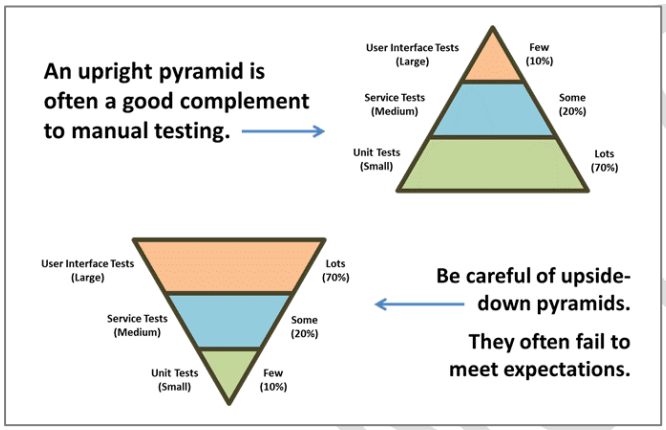
```
<resources>

    <!--Base application theme-->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

    <!--Full screen theme-->
    <style name="FullScreen" parent="Theme.AppCompat.Light.NoActionBar">
        <!--Colour for loading spinner-->
        <item name="colorPrimary">@color/divider</item>
        <item name="colorPrimaryDark">@color/secondaryText</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

    <!--Smaller Heading text style-->
    <style name="HeadingSmall"
parent="@android:style/TextAppearance.Large">
        <item name="android:textSize">16sp</item>
        <item name="android:textColor">@color/secondaryText</item>
        <item name="android:fontFamily">sans-serif-medium</item>
    </style>

    <!--Larger Heading text style-->
    <style name="HeadingLarge"
parent="@android:style/TextAppearance.Large">
        <item name="android:textSize">20sp</item>
        <item name="android:textColor">@color/secondaryText</item>
        <item name="android:fontFamily">sans-serif-medium</item>
    </style>

    <!--Content text style-->
    <style name="Content" parent="@android:style/TextAppearance.Medium">
        <item name="android:textSize">14sp</item>
        <item name="android:textColor">@color/primaryText</item>
        <item name="android:fontFamily">sans-serif-light</item>
    </style>

    <!--Large Content text style-->
    <style name="ContentLarge"
parent="@android:style/TextAppearance.Medium">
        <item name="android:textSize">45sp</item>
        <item name="android:textColor">@color/primaryText</item>
        <item name="android:fontFamily">sans-serif-light</item>
    </style>

    <!--Notification text style-->
    <style name="Notification"
parent="@android:style/TextAppearance.Large">
        <item name="android:textSize">24sp</item>
        <item name="android:textColor">@color/secondaryText</item>
        <item name="android:fontFamily">sans-serif-medium</item>
    </style>

    <!--Dropdown text style-->
    <style name="Dropdown" parent="@android:style/TextAppearance.Medium">
```

```
60            <item name="android:textSize">14sp</item>
61            <item name="android:textColor">@color/primaryText</item>
62            <item name="android:fontFamily">sans-serif-light</item>
63            <item name="android:gravity">center_vertical</item>
64        </style>
65
66        <!--Button text style-->
67        <style name="AppTheme.Button" parent="Widget.AppCompat.Button.Colored">
68            <item name="colorButtonNormal">@color/colorAccent</item>
69            <item name="android:textColor">@android:color/white</item>
70            <item name="android:textStyle">bold</item>
71        </style>
72
73    </resources>
```

4.0 – Testing

4.1 – Testing methods

With agile software development techniques increasing in popularity over the years, testing has had to change to keep up with them. It used to be the case that testing was something that was separated and done through the user interface at the end of the development process, once the system had been fully built. However, now, because of continuous integration of elements into a system, testing is completed throughout the development stage at a code level as well, to check that each implemented unit functions as it should. This helps to identify bugs and errors faster, as the pieces of code just written can be used to find the bug, instead of trying to find it from behind a user interface[27]. Furthermore, with a focus on unit testing while developing, adopting the 'testing pyramid' approach can help prevent bugs within the code, as opposed to finding them then fixing them in the 'testing cone' approach[28], as shown in the diagram[29] below.



Therefore, I have followed the 'testing pyramid' approach throughout the development of my app and shown in section 4.2 are the results from the unit and service tests. In addition, I will also complete user Interface tests to check that the app works as it should for the user – the results from these tests can be seen in section 4.3 below.

Following the 'testing pyramid' approach meant that during the development of the app I was continually coding then testing the app, this most often being done by attempting to run the app on a smartphone – this in itself, being a form of testing. Therefore, with this being a common test method, I ended up running the app hundreds of times to test it. This is obviously far too many times to document, and so not all of these instances have been documented.

Furthermore, during the development of my app I not only created one final app, but two prototype apps beforehand as well (in addition to the smaller test apps). These were tested with the same 'testing pyramid' approach as the final app, and so the testing of the two prototype apps led me nicely into the testing of the final app. The results from the prototype tests are at the beginning of the tests in section 4.2 below.

In the tables below, I will be using 4 different test data types – these are:

- Correct – data that the system should expect
- Erroneous – data I know is incorrect, designed to break the system
- Extreme – data that is either very large or very small
- Boundary – data that is on the boundary of a limit

[27] Pastusiak, I. (2016). How testing has changed. [online] Kainos. Available at: https://goo.gl/m1fYkg
[28] Brown, R. (2014). The Agile Testing Pyramid. [online] Agile Coach Journal. Available at: https://goo.gl/EA563E
[29] Image source: https://goo.gl/wULFzM

4.2 – Testing within the Documented Design

Throughout the development section of my app (section 2.0 – Documented Design) I undertook many unit and service tests to make sure that the various components of the app were working as they should. Unit tests make sure that individual sections of the app are working such as buttons or pages, whereas service tests test things such as the connection to the API, or the storage of data. All of these tests were documented within the relevant sections above, but they are also displayed below with more details.

Test no.	Section ref.	Purpose of test	Test data used	Expected outcome	Actual outcome	Type of test data	Date of test
1	2.2.2.1.2	To check that the padding attributes added to the elements are correct	*The attributes code for each XML element*	Display all of the elements in the correct position, with ample amount of padding	As expected	Correct	13/10/17
2	2.2.2.1.3	To check the datatypes that the keyboard allows, based on the type of keyboard set	"1"	Allow the input	As expected	Correct	15/10/17
3			","	Do not allow the input	As expected	Erroneous	
4			"-"	Do not allow the input	As expected	Erroneous	
5			"."	Allow the input	As expected	Erroneous	
6			".."	Do not allow the input	As expected	Erroneous	
7	2.2.2.1.5	Check that the Java code to override the page title set in the manifest works	setTitle("Converter");	The title bar to be named "Converter"	As expected	Correct	18/10/17
8	2.2.2.1.7	Check that the spinner displays the correct data in the correct order	"array_currencies" array from strings.xml	The spinner to display these options: "European Euro" "US Dollar" "Japanese Yen"	As expected	Correct	23/10/17
9	2.2.2.1.8	Check that the convert button does what it is supposed to	Spinner option: "European Euro" User input: "10"	"€10"	As expected	Correct	27/10/17
10			Spinner option: "US Dollar" User input: "10"	"$10"	As expected	Correct	

Test no.	Section ref.	Purpose of test	Test data used	Expected outcome	Actual outcome	Type of test data	Date of test
11			Spinner option: "Japanese Yen" User input: "10"	"¥10"	As expected	Correct	
12	2.2.2.1.9	To check that when an option is selected from the spinner onConvertClick() is called	*The user selects a spinner option*	The conversion process to occur	As expected	Correct	28/10/17
13	2.2.2.1.10	To check clicking on the action bar icons work properly	*The user selects the Reset/refresh icon*	The user input field to be cleared The spinner to be reset to the first option The result field to be cleared	As expected	Correct	30/10/17
14			*The user selects the Saved Conversions icon*	Change to the Saved Conversions page	As expected	Correct	
15	2.2.2.2.1	To check that the up arrow works on the saved conversions page	*The user selects the up arrow*	Return the user to the main page	As expected	Correct	1/11/17
16	2.2.2.2.2.5	To check the ListView displays the data correctly	*The dummy data entered in section 2.2.2.2.2.4*	The list to display the data in the correct layout and order	As expected	Correct	5/11/17
17	2.2.2.3.1	To check clicking on the help icon on the action bar works properly	*The user selects the Help icon*	Change to the Help & About page	As expected	Correct	7/11/17
18	2.2.2.3.2	To check that the up arrow works on the help and about page	*The user selects the up arrow*	Return the user to the main page	As expected	Correct	
19	2.3.2	To check that the JSON results are parsed correctly	*Data retrieved from the API*	To add each currency to the ArrayList correctly	The app crashes and produces an error	Correct	8/11/17

Test no.	Section ref.	Purpose of test	Test data used	Expected outcome	Actual outcome	Type of test data	Date of test
20	2.3.3	To check the correct data is retrieved from the API	https://api.fixer.io/latest	For the Android Logcat to display the key and value pairs for all the currencies.	The Android Logcat displays 31 lines of the first object key (AUD)	Correct	8/11/17
21	2.3.4	To check varying parameters added to the request URL work as intended	https://api.fixer.io/latest?base=USD	Return currencies and conversion rates against the US dollar	As expected	Correct	10/11/17
22			https://api.fixer.io/latest?symbols=USD,GBP	Return conversion rates for US dollars and Pounds Sterling with a base of Euros	As expected	Correct	
23			http://api.fixer.io/latest?base=GBP&symbols=EUR,USD	Return conversion rates for Euros and US dollars with a base of Pounds Sterling	As expected	Correct	
24	2.4.3.1	To check the tables have been created correctly	pragma table_info(apiData);	Return each attribute with its properties	As expected	Correct	24/11/17
25			pragma table_info(savedConversion);	Return each attribute with its properties	As expected	Correct	
26	2.4.3.2	To check the SQL INSERT queries for apiData work correctly	INSERT INTO apiData VALUES(null, “GBP”, “2017-11-17”, “EUR”, “1.1213”);	Insert the values into record 1 of the table.	As expected	Correct	25/11/17
27			INSERT INTO apiData VALUES(null, “EUR”, “2018-11-17”, “GBP”, “0.9761”);	Insert the values into record 2 of the table.	As expected	Correct	

Test no.	Section ref.	Purpose of test	Test data used	Expected outcome	Actual outcome	Type of test data	Date of test
28	2.4.3.2	To check the SQL INSERT queries for savedConversion work correctly	INSERT INTO savedConversion VALUES(null, “GBP”, “10.50”, “EUR”, “11.77365”, “2017-11-16”);	Insert the values into record 1 of the table.	As expected	Correct	26/11/17
29			INSERT INTO savedConversion VALUES(null, “EUR”, “11.20”, “AUR”, “20.3340”, “2057-31-16”);	Insert the values into record 2 of the table.	As expected	Correct	
30		To check that the SELECT query for apiData works correctly	SELECT conversionRate FROM apiData WHERE convertedCurrency = “EUR”;	Return only the value “1.1213”	As expected	Correct	27/11/17
31		To check that the SELECT query for savedConversion works correctly	SELECT * FROM savedConversion;	Return the contents of the entire table	As expected	Correct	
32	2.5.2.1	To check that the app still runs after copying across the layout code	*Running the app on a smartphone*	For the main features of the layout to be present in the correct places with the correct amount of padding, the action bar menu items not to be present, the spinner not populated and save conversion button not to be present.	As expected	Correct	29/11/17
33	2.5.3	To check that the app still runs after copying across the Java code	*Running the app on a smartphone*	For the app to crash	As expected	Erroneous	29/11/17

Test no.	Section ref.	Purpose of test	Test data used	Expected outcome	Actual outcome	Type of test data	Date of test
34	2.5.4.1	To check that the correct message is displayed to the user on the loading screen	*The smartphone to have an Internet connection*	The message "Loading data, please wait a moment" appear above a loading spinner. Following this, display the message "Data successfully retrieved" once the data has been loaded	As expected	Correct	29/11/17
35			*The smartphone to have no Internet connection*	The message "No Internet connection" to appear, with no loading spinner	As expected	Erroneous	29/11/17
36			*The smartphone to have an Internet connection AND* Set the request URL to something that would not work, e.g. "https://www.google.com/"	The message "An error occurred while loading the data – please try again later" to appear, with no loading spinner	As expected	Erroneous	29/11/17
37	2.5.4.2	To check that the data has been stored in the ArrayList correctly	*FOR loop in code, that outputs to Logcat*	Output each object on a new line, with it's attributes organised in the following order: baseCurrency, convertedCurrency, dateOfExchange, exchangeRate	As expected	Correct	29/11/17
38	2.5.4.4	To check that the intent works as intended, including the set flags	*switchActivity() is called once the data has loaded successfully*	Change the user to the main page of the app	As expected	Correct	1/12/17
39			*The user presses the back button on their smartphone*	The app should close instead of taking the user to the loading page	As expected	Erroneous	
40	2.5.5	To check that the spinner displays the correct options	https://api.fixer.io/latest?symbols=USD,GBP	Show the options USD and GBP with their correct flag resources	As expected	Correct	4/12/17

Test no.	Section ref.	Purpose of test	Test data used	Expected outcome	Actual outcome	Type of test data	Date of test
41	2.5.5.1	To check that the app doesn't crash when there is no resource for a given currency	https://api.fixer.io/latest?symbols=USD,GBP,AUD – which brings in AUD which doesn't yet a flag resource	For the app to handle the lack of an icon, and display just the currencyCode Both of the other currencies should be present with their flags	As expected	Erroneous	4/12/17
42	2.5.5.2	To check that the ArrayList is ordered correctly	*Each currency is given a priority, and then sorted based on that priority*	The first 7 currencies in the ArrayList to be, in this order: USD, EUR, JPY, AUD, CAD, CHF, CNY	As expected	Correct	6/12/17
43	2.5.6.3	To check the value the user entered can be used in a conversion	"4.1"	"4.1" to be used as the user input for the conversion	As expected	Correct	8/12/17
44			""	"0.0" to be used as the user input for the conversion	As expected	Erroneous	
45			"."	"0.0" to be used as the user input for the conversion AND The user to be notified to enter a number	As expected	Erroneous	
46			".4"	"0.4" to be used as the user input for the conversion	As expected	Correct	
47			"4."	"4.0" to be used as the user input for the conversion	As expected	Correct	
48	2.5.6.4	To check that the app displays the result in the correct format	User input: "5.4" Currency: "EUR"	"€ x" to be returned to the user, where x is the converted value	As expected	Correct	9/12/17
49			User input: "5.4" Currency: "CHF"	"CHF x" to be returned to the user, where x is the converted value	As expected	Correct	
Note: *for tests 48 & 49, making sure the calculated value is correct is not a priority of the test. This will come later, in the testing of the completed app in section 4.3 below.*							

Test no.	Section ref.	Purpose of test	Test data used	Expected outcome	Actual outcome	Type of test data	Date of test
50	2.5.7	To check that all of the currencies match with a flag resource and are displayed in the correct order in the spinner	https://api.fixer.io/latest?base=GBP *AND running the app on a smartphone*	Each currency to have a flag resource. The top 7 currencies be ordered: USD, EUR, JPY, AUD, CAD, CHF, CNY Any following currencies to be ordered alphabetically	As expected	Correct	10/12/17
51	2.5.8	To check the update exchange rates button works properly	*The user selects the Update exchange rates button*	Clear the ArrayList Call the loading activity to retrieve the latest results Change the date in the help & about page	As expected	Correct	12/12/17

4.3 – Testing the completed app

Now that the app is completed and that various units of the app have been tested while in development, user interface tests can be completed. These were all done after the above tests and focus on things that affect the user, e.g. button clicks, making sure calculations are correct etc.

All of the below tests/calculations were completed on 03/02/18, which meant that because it was a weekend, the app was using exchange rates from 02/02/18 (due to the way the API works). In addition, the expected conversions needed to come from a source that wasn't my app, so XE.com was used to provide the latest values from 03/02/18 – this is why there may be a couple of pence difference between the expected outcome and the actual outcome. Also, all values entered by the user are always in GBP.

To provide evidence for the actual outcome of the tests in the table below, I created a video that shows most of the tests being undertaken. A timestamp can be found for those applicable tests, matching to this YouTube video: http://bit.ly/AppTestResults. (Make sure to turn on captions).

Test no.	Purpose of test	Test data used	Expected outcome	Actual outcome	Timestamp	Type of test data
1	To check that the user can enter a value into the text input field	*User selects input field*	Cursor should start blinking; numeric keyboard should show	As expected	0:00	Correct
2		*User enters "4"*	"4" should be shown in the input field, blinking cursor following it	As expected		Correct
3	To check that the conversion process works / that the convert button works	User entered value: "4" Chosen currency: "USD"	$ 5.64884	$ 5.688000	0:05	Correct
4	To check that the conversion process works / that the convert button works	User entered value: "4.1" Chosen currency: "USD"	$ 5.79002	$ 5.830200	0:10	Correct
5		User entered value: "4." Chosen currency: "USD"	$ 5.64884	$ 5.688000	0:17	Correct
6		User entered value: ".4" Chosen currency: "USD"	$ 0.564884	$ 0.568800	0:24	Correct
7		User entered value: "4,200.90" Chosen currency: "USD"	Comma to be ignored; $ 5932.55	$ 5973.679800	0:30	Erroneous

Test no.	Purpose of test	Test data used	Expected outcome	Actual outcome	Timestamp	Type of test data
8		User entered value: "5..3" Chosen currency: "USD"	Second decimal point to be ignored; $ 7.48471	$ 7.536600	0:41	Erroneous
9		User entered value: "9-5" Chosen currency: "USD"	The second number be taken from the first, then converted; $ 5.64884	The operand is ignored; input becomes "95" and is converted; $ 135.090000	0:50	Erroneous
10		User entered value: "-3.7" Chosen currency: "USD"	The negative sign to be ignored; $ 5.22507	$ 5.261400	0:56	Erroneous
11		User entered value: "0.1" Chosen currency: "USD"	$ 0.141221	$ 0.14220	1:04	Extreme/ boundary
12		User entered value: "100000" Chosen currency: "USD"	$ 141221.43	$ 142200.000000	1:09	Extreme/ boundary
Note: the datatype of the user input can cope with values much larger and smaller than the above two tests, though the average user of the app will rarely, if ever, go outside of these boundaries (0.10p through to £100,000)						
13	To check that the spinner shows all of the available currencies	*User selects the spinner*	The top 7 currencies be ordered: USD, EUR, JPY, AUD, CAD, CHF, CNY; Any following currencies to be ordered alphabetically	As expected	1:20	Correct
14	To check the conversion process works with other currencies other than USD	User entered value: "500" Chosen currency: "CZK"	Kč 14,286.76	Kč 14338.000000	1:29	Correct
15	To check the conversion process works with currencies that don't have a symbol	User entered value: "750" Chosen currency: "CHF"	CHF 986.383	CHF 990.000000	1:42	Correct

Test no.	Purpose of test	Test data used	Expected outcome	Actual outcome	Timestamp	Type of test data
16	To check that the user can change the currency while keeping the input value the same	User entered value: "1000" Initial currency: "NOK" Changed currency: "TRY"	The result to change from: "kr 10924.58" to: "₺ 5328.09"	"kr 10889.000000" to "₺ 5326.700000"	1:57	Correct
17	To check that the user can change the input value while keeping the currency the same	Currency: "TRY" Initial entered value: "1000" Changed value: "500"	The result to change from: "₺ 5328.09" to: "₺ 2664.05"	"₺ 5326.700000" to: "₺ 2663.350000"	2:27	Correct
18	To check that the clear results button works properly	*User selects the Clear results button*	The user input field to be reset; The spinner to be reset to USD; The result field to be reset;	As expected	2:37	Correct
19	To check that the Help and about button works properly	*User selects the Help and about button*	Change the user to the Help & About page	As expected	2:42	Correct
20	To check that the date displayed on the help and about page is correct	Date taken from raw JSON at: https://api.fixer.io/latest?base=GBP	Date from raw JSON: "2018-01-26"	Date shown on app page: "2018-01-26"	2:44	Correct
21	To check that the up arrow works on the help and about page	*The user selects the up arrow*	Take the user to the main page of the app	As expected	2:46	Correct
22	To check the update exchange rates button works properly	*The user selects the Update exchange rates button*	Clear the ArrayList Call the loading activity to retrieve the latest results Change the date in the help & about page	As expected	2:49	Correct

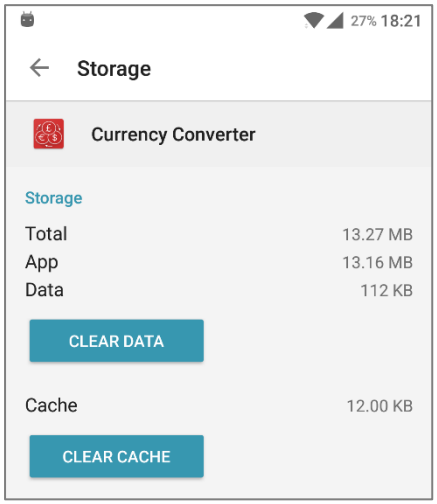
Test no.	Purpose of test	Test data used	Expected outcome	Actual outcome	Timestamp	Type of test data
23	To check that the loading page uses full screen immersion at the correct times	*Loading the app with an Internet connection*	The app to enter full screen while loading the data; Return to a normal view once the user is taken to the main page	As expected	2:52 Pause, then select the time	Correct
24		*Loading the app with no Internet connection*	The app to enter full screen while loading the data; Return to a normal view once the lack of connection is established	As expected	N/A	Correct
25	To check that the layout of the app does not change when the orientation of the smartphone is changed	*Tilting the smartphone to a horizontal orientation*	The app layout to remain as it was	As expected	N/A	Correct
26	To check the cursor only appears in the user input field when the user is typing	*The user input field to be unselected*	The keyboard to be hidden; No cursor shown	As expected	2:55	Correct
27		*The user input field to be selected*	The keyboard to open; A blinking cursor shown	As expected	2:59	Correct
28		*The done button on the keyboard to be selected*	The cursor to hide; The keyboard to hide	As expected	3:02	Correct
29		*The keyboard shown; The convert button selected*	Keyboard to remain shown; Cursor remain visible	The cursor hides	3:06	Correct
30		*The keyboard shown; A spinner option selected*	The keyboard and the cursor to hide	As expected	3:11	Correct

Test no.	Purpose of test	Test data used	Expected outcome	Actual outcome	Timestamp	Type of test data
31		*The keyboard shown;* *The spinner selected;* *The user selects an area outside of the spinner/the back button is pressed*	The spinner to hide; Keyboard and cursor remain visible	Spinner and keyboard hide; Cursor remains visible	N/A	Erroneous

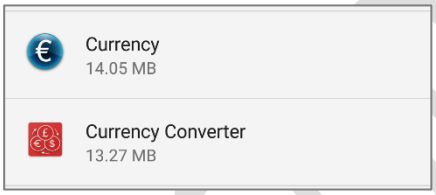
4.4 – Monitoring resource levels

CLIENT REQUIREMENT 13 stated that the final app should be hardware resource efficient (i.e. not draining battery, consuming large amounts of network traffic (data) or taking up too much storage space) and so these will now be tested.

To see how much storage space the app takes up on a smartphone, I was able to go to the settings on my smartphone and view its total size, which was around 13MB.

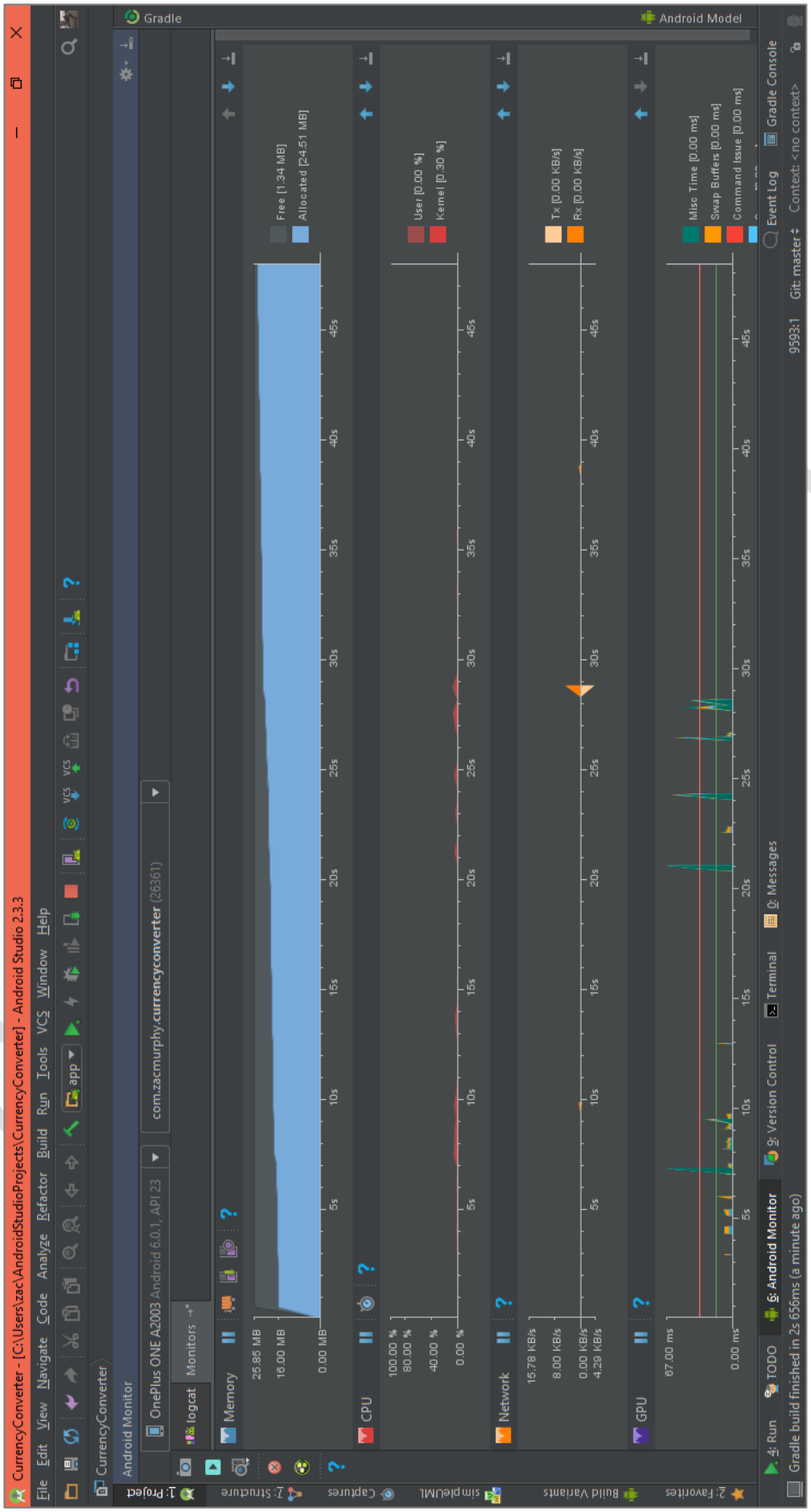


For this kind of app, with no data storage capabilities on it, this value is very good. In addition, it is ever so slightly smaller than "Easy Currency Converter" – one of the on-market apps I looked at in section 1.7 – showing that my app is competitive with the on-market apps.



To test the memory, CPU, network and GPU usage, I used Android Studio's resource monitor while running the app to monitor the usage of these resources. While running the app, I used all of the functionalities on it and seen below are the events that match with the timestamps on the image of the next page.

Timestamp (seconds)	Event
0-5	The app starts; the UI is loaded
5-10	The user input field is selected; a value is entered; the convert button is pressed
10-15	The 'Clear results' button is selected
15-20	The app is idle
20-25	The help page is loaded and then exited; the spinner is selected, and the list is scrolled through
25-30	'Update exchange rates' menu option is selected
30+	The app is idle


CurrencyConverter - [C:\Users\zac\AndroidStudioProjects\CurrencyConverter] - Android Studio 2.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
CurrencyConverter
Gradle
Android Model
Android Monitor
OnePlus ONE A2003 Android 6.0.1, API 23
com.zacmurphy.currencyconverter (26361)
logcat
Monitors
Memory
26.85 MB
16.00 MB
0.00 MB
Free [1.34 MB]
Allocated [24.51 MB]
CPU
100.00 %
80.00 %
40.00 %
0.00 %
User [0.00 %]
Kernel [0.30 %]
Network
16.78 KB/s
8.00 KB/s
0.00 KB/s
4.29 KB/s
Tx [0.00 KB/s]
Rx [0.00 KB/s]
GPU
67.00 ms
0.00 ms
Misc Time [0.00 ms]
Swap Buffers [0.00 ms]
Command Issue [0.00 ms]
5s 10s 15s 20s 25s 30s 35s 40s 45s
Run TODO Android Monitor Version Control Terminal Messages
Event Log Gradle Console
Gradle build finished in 2s 656ms (a minute ago)
9593:1 Git: master Context: <no context>
1: Project 7: Structure Captures simpleUML Build Variants 2: Favorites

4.5 – Stress testing

Stress testing is a way of pushing the capabilities of a system, often beyond what it was designed for. Though this is not part of the client requirements, I felt that the value of this kind of test could help to add to the overall testing of my app.

Android Studio has a feature called the "UI/Application Exerciser Monkey" or Monkey for short "that runs on your emulator or device and generates pseudo-random streams of user events such as clicks, touches, or gestures, as well as a number of system-level events." Its purpose being to "stress-test applications that you are developing, in a random yet repeatable manner."[30]

Using the Android Developer's documentation on the tool, I used the following command to stress test my app:

adb shell monkey -p com.zacmurphy.currencyconverter --pct-touch 25 -v 10000

As the above command is quite unusual, the table below breaks down what the various parts mean.

Part of command	Meaning
adb shell monkey	Call the Android Monkey tool
-p com.zacmurphy.currencyconverter	Specify the only package (app) that the tool is allowed to access
--pct-touch 25	Specify the percentage of touch events. (Touch events are a down-up event in a single place on the screen.) These were prioritised as there are few swipe-gestures enabled within my app.
-v	Specify the verbosity (detail) level of the output in the shell (see picture below for example)
10000	The number of events that should be simulated

```
:Sending Trackball (ACTION_MOVE): 0:(1.0,1.0)
:Switch: #Intent;action=android.intent.action.MAIN;category=andr
    // Allowing start of Intent { act=android.intent.action.MAIN
:Sending Touch (ACTION_DOWN): 0:(220.0,226.0)
:Sending Touch (ACTION_UP): 0:(218.86307,222.32733)
:Sending Touch (ACTION_DOWN): 0:(907.0,1159.0)
:Sending Touch (ACTION_UP): 0:(918.8327,1148.3967)
:Sending Touch (ACTION_DOWN): 0:(14.0,1797.0)
Events injected: 5000
:Sending rotation degree=0, persist=false
:Dropped: keys=1 pointers=18 trackballs=0 flips=15 rotations=0
## Network stats: elapsed time=14526ms (0ms mobile, 0ms wifi, 14
// Monkey finished
```

From this, the simulations that the tool created are shown in this YouTube video that I created: http://bit.ly/MonkeyTestResults (Make sure to turn on captions).

As the video shows, the app did not crash, therefore I would say that the app has passed stress testing. However, as the Android Monkey simulations are completely pseudo-random, another set of events under different parameters could cause the app to crash.

[30] https://goo.gl/wN4edz

4.6 – Future testing

Documented above is all of the testing that I have done, and will do, for my app. However, there is more advanced and thorough testing that can be done with Android apps, as detailed in the book "Learn Android Studio: Build Android Apps Quickly and Effectively" written by Clifton Craig & Adam Gerber.

This method is called Instrumentation testing, and it involves writing tests within separate Java classes of the app that when run "perform operations on a device as if a human user were operating it".[31] When writing the tests, you have to plan quite a lot of detail into them, including dealing with set-up and tear-down methods and testing the state of the app after various activities on different threads. Therefore, because this is quite complex and due to time constraints, I have decided to not include this type of testing on my app. But, I wanted to note, that if in the future I were to revisit this app to improve it, then creating and running Instrumentation tests would be one of the things I would do.

[31] Page 298: Gerber, A. and Craig, C. (2015). Learn Android Studio. [New York, NY]: Apress.

5.0 – Evaluation

5.1 – Feedback from the clients

Now that the app is complete, I am able to get feedback from my both of my clients on what they think of it, seen below are their views.

Mr Murphy

Even before the app has loaded, Mr Murphy liked that the home-screen icon is very clear and precise to the features that the app has and upon selection, the app loads very quickly to the page of the main function on the app. Following this, he thought that the user input field is intuitive and clearly placed and felt that restricting the user input to only numeric values is an extremely good thing. In addition, he liked the convenience of the done button on the keyboard automatically initiating the conversion process, as this made the app easier to use for him. Furthermore, the fixed base currency of GBP is appreciated, as this is the currency that he would be converting from and he felt that the flag icons are placed well and fit into the theme of the app. Finally, the inclusion of the date provided in the about the app section was welcomed, as it provides confirmation of when the exchange rates are from.

However, with no Internet connection, he thought that it would have been good for the app to be able to be used with previously collected data. Also, that the lack of data storage functions, that were originally included in the client requirements, but later not implemented, limit the comprehensiveness of the app.

Though, overall, he thought that "the whole app is easy to use and navigate" and was *pleasantly surprised by the "simplicity of the look and use of the app, considering the complexity that was required to create it."*

Mrs Murphy

Firstly, Mrs Murphy was very appreciative that throughout the creation process, I had involved her in some of the decision making – notably, consulting her about the colour schemes, the design of the app icon and which certain features to implement (e.g. the help page). This meant that she was very pleased that these things had been implemented and was very happy overall with how the app turned out to be. Mrs Murphy also felt that the inclusion of the flags is very useful, for those that operate visually, as well as giving the app some more colour and was impressed that the exchange rates update every 24 hours, instead of having fixed rates. She also felt that the headings and text sizes "are ideal, especially for someone that usually needs glasses" and that the app icon on the home screen stands-out well against the other icons.

However, Mrs Murphy did suggest that it would have been nice to have the functionality to convert from multiple currencies, as opposed to just GBP. And, that more information could be provided on each currency, for example its name, or the country of origin.

Though, overall, she thought that the app was "simple, concise and easy to use – being something that you can put in your pocket to easily get the latest exchange rates."

5.2 – Evaluating the client requirements

The **CLIENT REQUIREMENTS** in section 1.3 were given to me at the beginning of the project as a basis to build the app on and throughout the design and creation process, I constantly referred to them to

make sure that what I was building suited the client's needs. Now that the app has been fully built, I can analyse each of the requirements to make sure they were met.

1. **Access different currencies and convert them against GBP**
    - This was met by the inclusion of a drop-down box for the user to select from a list of 31 different currencies to convert to, from GBP.
    - See section 2.5.5 that documents the implementation of this.
2. **Automatically get the conversion rate based on selecting a country**
    - This was met by the automatic functions that initiated the conversion process, when the convert button was selected, the done button on the keyboard was selected, or a currency to convert to is selected.
    - See section 2.5.6 that documents the implementation of this.
3. **Be able to convert real monetary values (i.e. decimal places or large figure values)**
    - This was met by allowing the user to enter a decimal value from smaller than £0.10p to larger than £100,000
    - See section 4.3 that shows the testing of this.
4. **Use a list to order currencies and place more common ones nearer to the top**
    - This was met by the inclusion of a drop-down box and the implementation of a priority system for the currencies, in order to rank them.
    - See section 2.5.5.2 that documents the implementation of this.
5. **Use distinguishable flags to help represent currencies in the list**
    - This was met by including an icon for each currency in the list, and by including a flag on the main page of the app.
    - See section 2.5.5.1 that documents the implementation of this.
6. **Save conversion rates, so that the app can be used when a network connection is not available**
    - This was not met due to the discussion in section 2.4.2.3.
7. **Have the ability to save converted values for comparisons**
    - This was not met due to the discussion in section 2.4.2.3.
8. **Be able to delete saved conversions**
    - This was not met due to the discussion in section 2.4.2.3.
9. **Have a menu that is easy to navigate with symbols, with options including:**
    - **Refresh the page**
        - This was met by including a refresh button on the action bar.
        - See section 2.5.8 that documents the implementation of this.
    - **Navigate to view stored data**
        - This was not met due to the discussion in section 2.4.2.3.
10. **Have a clear and simple layout that makes using the app intuitive**
    - This was met as my client, Mr Murphy, said "the whole app is easy to use and navigate"
11. **Have clear and correctly sized text and headings that are easy to read**
    - This was met as my client, Mrs Murphy, said that the heading and text sizes "*are ideal, especially for someone that usually needs glasses*"
12. **Have an app icon that has:**
    - **A rounded square shape as a background**
    - **At least 3 different currencies represented on it**
    - **Bold colours**

- All three of these are met from the design and implementation of the launcher icon in section 2.6.4.

13. **Be hardware resource efficient (i.e. not draining battery, consuming large amounts of network traffic (data))**
    - These conditions were met by the successful test results in section 4.4.
14. **Be able to choose different base currencies**
    - This was not met due to the limitations of time.
15. **Have whether the currency rate has changed from the day before, with arrows to symbolise up or down and equals for has stayed the same**
    - This was not met due to the limitations of time.

5.3 – Future refinements to the project

When completing the project, there wasn't time for me to do everything that I had either hoped to do or had planned on doing. In this section I will be discussing the ideas that I had, and how they could be implemented in any future versions of the app.

Firstly, the main features that would be implemented in future versions of the app would include CLIENT REQUIREMENTS 6 TO 8. These were the data storage features that part way through the project (section 2.4.2.3), I decided to not include. While creating the app, I kept any code to do with data storage that was in the files that I needed commented out and said that at the end of building the app I would decide on whether or not to continue learning and implementing them if I had time. Upon reflection, once finishing the app, I realised that I had in fact made a sensible decision, as I did not have enough time to continue with them, with testing and other things to complete for the project as well. Therefore, saving conversion rates and being able to save / delete converted values for comparisons would be the first things I would look into implementing into future versions of the app.

Secondly, the optional CLIENT REQUIREMENTS 14 AND 15 could be implemented into future versions as they were not included in the final app. These were, the ability to choose the base currency to convert from and having whether the exchange rate for a particular currency had risen, fallen, or stayed the same – in comparison to the day before. My client, Mrs Murphy, also suggested that support for additional base currencies would have been nice and Mr Murphy suggested that it would have been good for the app to be able to be used with previously collected data, if there was no current Internet connection – so the app remained functioning as much as possible. In addition, I thought that a 'last three conversions' list could be implemented into future versions, where the data from the last three conversions appeared on the main screen (where there is currently an empty space), but not permanently saved. This would allow the user to instantly compare a couple of values, without using the saved conversions feature (this being designed for permanent storage of currency exchanges).

Finally, as mentioned in section 4.6, instrumentation testing could be something that is implemented into the testing of the future versions of the app, in order to help test it more thoroughly. This would help to make the app more robust and bring it up to Google Play Store standards, if I was to decide to publish it online. In addition, the minor issues with the cursor not hiding when it should (found during testing) could also be fixed.

Overall, I feel that this project has gone successfully considering the timeframe that I had, and despite not being able to include some of the features that I had wanted – I am pleased with the outcome of the final app.

6.0 – Appendices

Appendices

App code note

Due to the large number of files and size of each file for each app, the code for each app has been uploaded to its own repository on GitHub – this prevents this appendix becoming excessively long and difficult to navigate. A link to each repository also allows me to organise the code better by making sure that the latest version on this appendix is always correct – this will be done by regularly uploading any changes made in Android Studio. In addition, using GitHub and other similar repository services is something that is used in the technology industry – and so it makes sense to adopt this convention.

The file structure tree for an Android app can be quite confusing, therefore **I have provided the code of the two main file types** (XML & Java) for easy finding of my code and in addition a link to the main source of the app, where the manifest file can be found[32]. In addition, within the main source is a "res" resources folder **which contains all of the app's assets** including XML files for menus, values and drawable resources; and image files, if any. These resources, **I will have created or placed there**. However, going backwards in the tree from the main source will only show files **generated by Android Studio.**

[32] The manifest file provides essential information about the app to the Android system, which the system must have before it can run any of the app's code

Appendix 1 – Happy Birthday app code

https://goo.gl/V3tK5r

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="com.example.android.happybirthday.MainActivity"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:src="@drawable/android_party"
        android:scaleType="centerCrop"/>

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Happy birthday dude! Have a great day"
        android:textSize="36sp"
        android:fontFamily="sans-serif-light"
        android:textColor="#FFFFFF"
        android:padding="8dp"/>

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="From your mate"
        android:textSize="36sp"
        android:fontFamily="sans-serif-light"
        android:textColor="#FFFFFF"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:padding="8dp"/>

</RelativeLayout>
```

MainActivity.java

```
package com.example.android.happybirthday;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

Appendix 2 – Court Counter app code

https://goo.gl/77yfac

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.android.courtcounter.MainActivity">

    <LinearLayout
        android:id="@+id/scoresLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="24dp"
            android:layout_marginRight="24dp"
            android:layout_marginTop="16dp"
            android:layout_weight="1"
            android:orientation="vertical">

            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginBottom="16dp"
                android:fontFamily="sans-serif-medium"
                android:gravity="center_horizontal"
                android:text="Team A"
                android:textColor="@color/secondaryText"
                android:textSize="14sp" />

            <TextView
                android:id="@+id/teamAScore"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginBottom="24dp"
                android:fontFamily="sans-serif-light"
                android:gravity="center_horizontal"
                android:text="0"
                android:textColor="@color/primaryText"
                android:textSize="56sp" />

            <Button
                android:layout_width="match_parent"
                android:layout_height="48dp"
                android:layout_marginBottom="8dp"
                android:onClick="increaseBy3TeamA"
                android:text="+3 points" />

            <Button
                android:layout_width="match_parent"
                android:layout_height="48dp"
                android:layout_marginBottom="8dp"
                android:onClick="increaseBy2TeamA"
                android:text="+2 points" />
```

```
        <Button
            android:layout_width="match_parent"
            android:layout_height="48dp"
            android:onClick="increaseBy1TeamA"
            android:text="Free Throw" />
    </LinearLayout>

    <View
        android:layout_width="1dp"
        android:layout_height="match_parent"
        android:layout_marginTop="16dp"
        android:background="@color/divider"></View>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="24dp"
        android:layout_marginRight="24dp"
        android:layout_marginTop="16dp"
        android:layout_weight="1"
        android:orientation="vertical">

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="16dp"
            android:fontFamily="sans-serif-medium"
            android:gravity="center_horizontal"
            android:text="Team B"
            android:textColor="@color/secondaryText"
            android:textSize="14sp" />

        <TextView
            android:id="@+id/teamBScore"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="24dp"
            android:fontFamily="sans-serif-light"
            android:gravity="center_horizontal"
            android:text="0"
            android:textColor="@color/primaryText"
            android:textSize="56sp" />

        <Button
            android:layout_width="match_parent"
            android:layout_height="48dp"
            android:layout_marginBottom="8dp"
            android:onClick="increaseBy3TeamB"
            android:text="+3 points" />

        <Button
            android:layout_width="match_parent"
            android:layout_height="48dp"
            android:layout_marginBottom="8dp"
            android:onClick="increaseBy2TeamB"
            android:text="+2 points" />

        <Button
            android:layout_width="match_parent"
            android:layout_height="48dp"
```

```
                android:onClick="increaseBy1TeamB"
                android:text="Free Throw" />
        </LinearLayout>
    </LinearLayout>

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_marginBottom="32dp"
        android:layout_marginLeft="24dp"
        android:layout_marginRight="24dp"
        android:onClick="resetScore"
        android:text="Reset" />
</RelativeLayout>
```

MainActivity.java

```
package com.example.android.courtcounter;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    /**
     * Declare global variables
     */
    int teamAScore = 0;
    int teamBScore = 0;

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

    /**
     * Increase the score by 1
     */
    public void increaseBy1TeamA(View view) {
        teamAScore += 1;
        displayScoreTeamA(teamAScore);
    }

    /**
     * Increase the score by 2
     */
    public void increaseBy2TeamA(View view) {
        teamAScore += 2;
        displayScoreTeamA(teamAScore);
    }

    /**
     * Increase the score by 3
     */
    public void increaseBy3TeamA(View view) {
        teamAScore += 3;
        displayScoreTeamA(teamAScore);
```

```
    }

    /**
     * Displays the score for Team A
     */
    public void displayScoreTeamA(int score) {
        TextView scoreView = (TextView) findViewById(R.id.teamAScore);
        scoreView.setText(String.valueOf(score));
    }

    /**
     * Increase the score by 1
     */
    public void increaseBy1TeamB(View view) {
        teamBScore += 1;
        displayScoreTeamB(teamBScore);
    }

    /**
     * Increase the score by 2
     */
    public void increaseBy2TeamB(View view) {
        teamBScore += 2;
        displayScoreTeamB(teamBScore);
    }

    /**
     * Increase the score by 3
     */
    public void increaseBy3TeamB(View view) {
        teamBScore += 3;
        displayScoreTeamB(teamBScore);
    }

    /**
     * Displays the score for Team B
     */
    public void displayScoreTeamB(int score) {
        TextView scoreView = (TextView) findViewById(R.id.teamBScore);
        scoreView.setText(String.valueOf(score));
    }

    /**
     * Resets the score for both teams
     */
    public void resetScore(View view) {
        teamAScore = 0;
        teamBScore = 0;
        TextView teamA = (TextView) findViewById(R.id.teamAScore);
        TextView teamB = (TextView) findViewById(R.id.teamBScore);
        teamA.setText(String.valueOf(teamAScore));
        teamB.setText(String.valueOf(teamBScore));
    }
}
```

Appendix 3 – Just Java app code

https://goo.gl/cXahQk

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fillViewport="true"
    tools:context="com.example.android.justjava.MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <EditText
            android:id="@+id/name_view"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="16dp"
            android:layout_marginRight="16dp"
            android:layout_marginTop="16dp"
            android:hint="@string/nameEntry"
            android:inputType="textCapWords"
            android:theme="@style/mainTextTheme" />

        <TextView
            android:id="@+id/textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="16dp"
            android:layout_marginTop="16dp"
            android:text="@string/toppingsHeader"
            android:theme="@style/headerTheme" />

        <CheckBox
            android:id="@+id/whipped_cream_checkbox"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="16dp"
            android:layout_marginTop="8dp"
            android:paddingLeft="12dp"
            android:text="@string/whippCreamOpt"
            android:theme="@style/mainTextTheme" />

        <CheckBox
            android:id="@+id/chocolate_checkbox"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="16dp"
            android:layout_marginTop="8dp"
            android:paddingLeft="12dp"
            android:text="@string/chocOpt"
            android:theme="@style/mainTextTheme" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
        android:layout_marginLeft="16dp"
        android:layout_marginTop="16dp"
        android:text="@string/quantityHeader"
        android:theme="@style/headerTheme" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:orientation="horizontal">

        <Button
            android:layout_width="48dp"
            android:layout_height="48dp"
            android:layout_marginLeft="16dp"
            android:onClick="decrement"
            android:text="-"
            android:theme="@style/AppTheme.Button" />

        <TextView
            android:id="@+id/quantity_text_view"
            android:layout_width="24dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="8dp"
            android:gravity="center"
            android:text="1"
            android:theme="@style/mainTextTheme" />

        <Button
            android:layout_width="48dp"
            android:layout_height="48dp"
            android:layout_marginLeft="8dp"
            android:onClick="increment"
            android:text="+"
            android:theme="@style/AppTheme.Button" />
    </LinearLayout>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="16dp"
        android:layout_marginTop="16dp"
        android:text="@string/orderSummaryHeader"
        android:theme="@style/headerTheme" />

    <TextView
        android:id="@+id/orderSummaryTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="16dp"
        android:layout_marginTop="8dp"
        android:text="@string/orderSummaryText"
        android:theme="@style/mainTextTheme" />

    <Button
        android:id="@+id/next_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="16dp"
        android:layout_marginTop="16dp"
        android:onClick="submitOrder"
```

```
            android:text="@string/orderButton"
            android:theme="@style/AppTheme.Button" />
    </LinearLayout>

</ScrollView>
```

MainActivity.java

```
/**
 * IMPORTANT: Add your package below. Package name can be found in the
project's AndroidManifest.xml file.
 * This is the package name our example uses:
 * <p>
 * package com.example.android.justjava;
 */
package com.example.android.justjava;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.text.TextUtils;
import android.view.View;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.TextView;

import java.text.NumberFormat;

/**
 * This app displays an order form to order coffee.
 */
public class MainActivity extends AppCompatActivity {

    /**
     * Declare global variables
     */
    int quantity = 1;
    double costOfCoffee = 1.95;
    double costOfCream = 0.20;
    double costOfChocolate = 0.30;
    double priceOfOrder;

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

    /**
     * This method is called when the order button is clicked.
     */
    public void submitOrder(View view) {
        String customerName = getNameFromField();

        if (customerName == null) {
            String orderSummary =
createOrderSummary(getString(R.string.orderSummaryNoName));
            displaySummary(orderSummary);
        } else {
            String orderSummary = createOrderSummary(customerName);
```

```
            if (quantity == 0) {
                displaySummary(orderSummary);
            } else {
                displaySummary(getString(R.string.processingOrder));
                composeEmail(orderSummary);
            }
        }
    }

    /**
     * Creates an order summary for the user
     **/
    private String createOrderSummary(String customerName) {
        String quantityText = getString(R.string.quantity, quantity);
        String creamText = getString(R.string.cream, isCreamChecked());
        String chocolateText = getString(R.string.chocolate,
isChocolateChecked());
        String totalPriceText = getString(R.string.total, formatPrice());
        String thankYou = getString(R.string.thankYouMessage);

        if (quantity == 0) {
            return customerName + "\n" + getString(R.string.noOrderText);
        } else {
            return customerName + "\n" + quantityText + "\n" + creamText +
"\n" + chocolateText + "\n" + totalPriceText + "\n" + thankYou;
        }
    }

    /**
     * Gets the  contents of the Name entry field
     */
    private String getNameFromField() {
        EditText nameField = (EditText) findViewById(R.id.name_view);
        String userName = nameField.getText().toString();

        if (TextUtils.isEmpty(userName)) {
            nameField.setError(getString(R.string.errorText));
            return null;
        } else {
            return getString(R.string.orderSummaryName, userName);
        }
    }

    /**
     * Gets the status of the Whipped Cream checkbox
     */
    private String isCreamChecked() {
        CheckBox whippedCreamBox = (CheckBox)
findViewById(R.id.whipped_cream_checkbox);
        if (whippedCreamBox.isChecked()) {
            return getString(R.string.choiceYes);
        } else {
            return getString(R.string.choiceNo);
        }
    }

    /**
     * Gets the status of the Chocolate checkbox
     */
    private String isChocolateChecked() {
```

```
        CheckBox chocolateBox = (CheckBox)
findViewById(R.id.chocolate_checkbox);
        if (chocolateBox.isChecked()) {
            return getString(R.string.choiceYes);
        } else {
            return getString(R.string.choiceNo);
        }
    }

    /**
     * This method displays the summary of the order to the user
     */
    private void displaySummary(String message) {
        TextView orderSummaryTextView = (TextView)
findViewById(R.id.orderSummaryTextView);
        orderSummaryTextView.setText(message);
    }

    /**
     * Formats the price for the user using their local currency
     */
    private String formatPrice() {
        return NumberFormat.getCurrencyInstance().format(calculatePrice());
    }

    /**
     * Calculates the price for the user
     */
    private double calculatePrice() {
        priceOfOrder = costOfCoffee;
        if (isCreamChecked() == getString(R.string.choiceYes)) {
            priceOfOrder += costOfCream;
        }
        if (isChocolateChecked() == getString(R.string.choiceYes)) {
            priceOfOrder += costOfChocolate;
        }
        priceOfOrder = priceOfOrder * quantity;
        return priceOfOrder;
    }

    /**
     * Increments the value of quantity by 1
     */
    public void increment(View view) {
        if (quantity == 99) {
            quantity = 99;
        } else {
            quantity = quantity + 1;
        }
        displayQuantity(quantity);
    }

    /**
     * Decrements the value of quantity by 1
     */
    public void decrement(View view) {
        if (quantity == 0) {
            quantity = 0;
        } else {
            quantity = quantity - 1;
        }
```

```
        displayQuantity(quantity);
    }

    /**
     * This method displays the given quantity value on the screen.
     */
    private void displayQuantity(int number) {
        TextView quantityTextView = (TextView)
findViewById(R.id.quantity_text_view);
        quantityTextView.setText("" + number);
    }

    /**
     * Calls the email intent an creates an email with the order summary
     */
    public void composeEmail(String emailContents) {
        Intent intent = new Intent(Intent.ACTION_SENDTO);
        intent.setData(Uri.parse("mailto:")); // only email apps should
handle this
        intent.putExtra(Intent.EXTRA_SUBJECT,
getString(R.string.orderSummaryHeader));
        intent.putExtra(Intent.EXTRA_TEXT, emailContents);
        if (intent.resolveActivity(getPackageManager()) != null) {
            startActivity(intent);
        }
    }
}
```

Appendix 4 – Arrays app code

https://goo.gl/eawzb9

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--In this, the main activity, is where the list view is defined.-->
<!--It is important to give the view an ID so it can be referenced later-->
<ListView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/rootView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.example.android.arrays.MainActivity" />
```

custom_list_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--Use this XML layout to define what each element of the list should look
like.-->
<!--Use 'xmlns:tools="http://schemas.android.com/tools"' in the parent
view-->
<!--so that 'tools:text' can be used to show sample text-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:paddingBottom="8dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="8dp">

    <LinearLayout
        android:id="@+id/iconLayout"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/icon_bg">

        <ImageView
            android:id="@+id/imageView"
            android:layout_width="48dp"
            android:layout_height="48dp"
            android:src="@mipmap/ic_launcher" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginLeft="16dp"
        android:orientation="vertical">

        <TextView
xmlns:android="http://schemas.android.com/apk/res/android"
            android:id="@+id/mainText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="bottom"
```

```
android:textAppearance="?android:attr/textAppearanceListItemSmall"
            android:textColor="@color/primaryText"
            android:textStyle="bold"
            tools:text="MainText" />

        <TextView
            android:id="@+id/subText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="top"

android:textAppearance="?android:attr/textAppearanceListItemSecondary"
            android:textColor="@color/secondaryText"
            tools:text="SubText" />
    </LinearLayout>
</LinearLayout>
```

MainActivity.java

```
package com.example.android.arrays;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.widget.ListView;

import java.util.ArrayList;
import java.util.List;

public class MainActivity extends AppCompatActivity {

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

        //Create the ArrayList, so it can be used with objects
        List<Word> words = new ArrayList<>();
        //Populate the ArrayList
        words.add(new Word("One", "Un", R.drawable.number_one));
        words.add(new Word("Two", "Deux", R.drawable.number_two));
        words.add(new Word("Three", "Trois", R.drawable.number_three));
        words.add(new Word("Four", "Quatre", R.drawable.number_four));
        words.add(new Word("Five", "Cinq", R.drawable.number_five));
        words.add(new Word("Six", "Six", R.drawable.number_six));
        words.add(new Word("Seven", "Sept", R.drawable.number_seven));
        words.add(new Word("Eight", "Huit", R.drawable.number_eight));
        words.add(new Word("Nine", "Neuf", R.drawable.number_nine));
        words.add(new Word("Ten", "Dix", R.drawable.number_ten));
        words.add(new Word("Eleven", "Onze"));
        words.add(new Word("Twelve", "Douze"));
        words.add(new Word("Thirteen", "Treize"));
        words.add(new Word("Fourteen", "Quatorze"));
        words.add(new Word("Fifteen", "Quinze"));
        words.add(new Word("Sixteen", "Seize"));
        words.add(new Word("Seventeen", "Dix-sept"));
        words.add(new Word("Eighteen", "Dix-huit"));
        words.add(new Word("Nineteen", "Dix-neuf"));
        words.add(new Word("Twenty", "Vingt"));
```

```
        //Creates an adapter for the words to use, appends the array of
words to the adapter,
        //the adapter is responsible for making a View for each item in the
data set
        WordAdapter adapter = new WordAdapter(this, words);
        //Creates a object constructor for the ListView
        ListView listView = (ListView) findViewById(R.id.rootView);
        //Sets the adapter method on the ListView
        listView.setAdapter(adapter);
    }
}
```

Word.java

```
package com.example.android.arrays;

/**
 * The Word class that creates and provides methods that return parts of
the ArrayList when called
 * by other parts of the app
 */
public class Word {

    private static final int NO_IMAGE_PROVIDED = -1;
    /**
     * Declare private variables for this class to use
     */
    private String mEnglishWord;
    private String mFrenchWord;
    private int mResourceId = NO_IMAGE_PROVIDED;

    /**
     * Create the constructor for this class, a constructor creates an
instance of a class
     * This constructor will create an instance of two Strings
     *
     * @param englishWord
     * @param frenchWord
     */
    public Word(String englishWord, String frenchWord) {
        mEnglishWord = englishWord;
        mFrenchWord = frenchWord;
    }

    /**
     * Create a second constructor for this class
     * This constructor will create an instance of two Strings and an int
     *
     * @param englishWord
     * @param frenchWord
     * @param resourceId  for Images use int as the resource ID is an int
type
     */
    public Word(String englishWord, String frenchWord, int resourceId) {
        mEnglishWord = englishWord;
        mFrenchWord = frenchWord;
        mResourceId = resourceId;
    }

    /**
     * Custom method
```

```
     *
     * @return the English version of the word
     */
    public String getEnglishWord() {
        return mEnglishWord;
    }

    /**
     * Custom method
     *
     * @return the French version of the word
     */
    public String getFrenchWord() {
        return mFrenchWord;
    }

    /**
     * Custom method
     *
     * @return the resource ID of the image
     */
    public int getImageResourceId() {
        return mResourceId;
    }

    /**
     * Custom method
     *
     * @return a boolean value on whether the image has a resource ID or
not
     */
    public boolean hasImage() {
        return mResourceId != NO_IMAGE_PROVIDED;
    }
}
```

WordAdapter.java

```
package com.example.android.arrays;

import android.app.Activity;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;

import java.util.List;

/**
 * Word adapter class that handles the multiple TextViews and ImageView for
the ListView
 */
public class WordAdapter extends ArrayAdapter<Word> {

    /**
     * A custom constructor.
```

```
     *
     * @param context is used to inflate the layout file
     * @param words   is the data we want to populate into the lists
     */
    public WordAdapter(Activity context, List<Word> words) {
        // This initialises the ArrayAdapter's internal storage for the
context and the list.
        // The second argument is used when the ArrayAdapter is populating
a single TextView.
        // Because this is a custom adapter for two TextViews the adapter
is not
        // going to use this second argument, so it can be any value.
        super(context, 0, words);
    }

    /**
     * Provides a view for an AdapterView (ListView, GridView, etc.)
     *
     * @param position    is the position in the list of data that should
be displayed in the list item view.
     * @param convertView The recycled view to populate.
     * @param parent      The parent ViewGroup that is used for inflation.
     * @return The View for the position in the AdapterView.
     */
    @NonNull
    @Override
    public View getView(final int position, @Nullable View convertView,
@NonNull ViewGroup parent) {
        //Get the data item for this position
        Word word = getItem(position);

        //Check if an existing view is being reused, otherwise inflate the
view
        if (convertView == null) {
            convertView =
LayoutInflater.from(getContext()).inflate(R.layout.custom_list_view,
parent, false);
        }

        //Lookup view for data population
        TextView mainText = (TextView)
convertView.findViewById(R.id.mainText);
        TextView subText = (TextView)
convertView.findViewById(R.id.subText);
        ImageView iconImage = (ImageView)
convertView.findViewById(R.id.imageView);
        LinearLayout iconLayout = (LinearLayout)
convertView.findViewById(R.id.iconLayout);

        //Populate the data into the template view using the data object
        mainText.setText(word.getEnglishWord());
        subText.setText(word.getFrenchWord());

        //If statement to check if the list item has an image associated
with it
        if (word.hasImage()) {
            //If it does, set the the ImageView to the resource
            iconImage.setImageResource(word.getImageResourceId());
            //Make sure it is visible, as views get recycled
            iconImage.setVisibility(View.VISIBLE);
            iconLayout.setVisibility(View.VISIBLE);
```

```
            } else {
                //Hide the view
                iconImage.setVisibility(View.GONE);
                iconLayout.setVisibility(View.GONE);
            }

            //Return the completed view to render on-screen
            return convertView;
        }
    }
```

Appendix 5 – Multiple Pages app code

https://goo.gl/Q8t38c

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.android.multiplepages.MainActivity">

    <TextView
        android:id="@+id/textField"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />

    <!--Use this button to navigate to the next page-->
    <Button
        android:id="@+id/navigate"
        android:layout_width="wrap_content"
        android:layout_height="48dp"
        android:layout_below="@id/textField"
        android:text="Next page" />
</RelativeLayout>
```

activity_secondary.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.example.android.multiplepages.SecondaryActivity">
    <!--xmlns:app="http://schemas.android.com/apk/res-auto"-->

    <TextView
        android:id="@+id/numbers"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="string/category_numbers" />

    <TextView
        android:id="@+id/family"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="string/category_family" />

    <TextView
        android:id="@+id/colors"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="string/category_colors" />

    <TextView
        android:id="@+id/phrases"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="string/category_phrases" />
```

```
    <!--Use this button to navigate to the previous page-->
    <Button
        android:id="@+id/backButton"
        android:layout_width="wrap_content"
        android:layout_height="48dp"
        android:text="Main page" />

</LinearLayout>
```

MainActivity.java

```
package com.example.android.multiplepages;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

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

        //Create a new button object constructor
        Button button = (Button) findViewById(R.id.navigate);
        //Set a listener to it and create a new method
        button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                //Do this when the button is clicked
                navigateNext();
            }
        });
    }

    /**
     * Method that creates an intent and launches the next activity for the
app
     */
    public void navigateNext() {
        //Create a new Intent object constructor, populate it with [the
activity you want to call].class
        Intent intent = new Intent(this, SecondaryActivity.class);
        //Start that new intent
        startActivity(intent);
    }
}
```

SecondaryActivity.java

```
package com.example.android.multiplepages;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
```

```
public class SecondaryActivity extends AppCompatActivity {

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

        //Create a new button object constructor
        Button button = (Button) findViewById(R.id.backButton);
        //Set a listener to it and create a new method
        button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                //Do this when the button is clicked
                navigateNext(null);
            }
        });
    }

    /**
     * Method that creates an intent and launches the next activity for the
app
     * @param view
     */
    public void navigateNext(View view) {
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
    }
}
```

Appendix 6 – Swipable Pages app code

https://goo.gl/eifJmx

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--XML layout for the main page-->
<!--Contains the view pager-->
<RelativeLayout 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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.android.tabbedpages.MainActivity">

    <android.support.v4.view.ViewPager
        android:id="@+id/viewPager"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" />
</RelativeLayout>
```

first_frag.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--XML layout for the first page-->
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/holo_orange_dark">

    <TextView
        android:id="@+id/tvFragFirst"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="TextView one"
        android:textSize="26sp" />
</RelativeLayout>
```

second_frag.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--XML layout for the second page-->
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/holo_green_dark">

    <TextView
        android:id="@+id/tvFragSecond"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="TextView two"
        android:textSize="26sp" />

</RelativeLayout>
```

MainActivity.java

```
package com.example.android.tabbedpages;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentActivity;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;
import android.support.v4.view.ViewPager;

public class MainActivity extends FragmentActivity {

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

        //Create the pager adapter and specify which method to control it
        ViewPager pager = findViewById(R.id.viewPager);
        pager.setAdapter(new MyPagerAdapter(getSupportFragmentManager()));
    }

    /**
     * Method for the pager adapter
     */
    private class MyPagerAdapter extends FragmentPagerAdapter {

        public MyPagerAdapter(FragmentManager fm) {
            super(fm);
        }

        /**
         * The switch statement that provides which fragment to switch to
when swiping across
         * views in the app
         *
         * @param pos
         * @return
         */
        @Override
        public Fragment getItem(int pos) {
            switch (pos) {
                case 0:
                    return FirstFragment.newInstance("FirstFragment");
                case 1:
                    return SecondFragment.newInstance("SecondFragment");
                default:
                    return SecondFragment.newInstance("Default");
            }
        }

        /**
         * Method that returns how many pages there will be
         *
         * @return
         */
        @Override
        public int getCount() {
            return 2;
        }
    }
```

```
}
```

FirstFragment.java

```
package com.example.android.tabbedpages;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

/**
 * Class that controls the creation of the first fragment
 */
public class FirstFragment extends Fragment {

    public static FirstFragment newInstance(String text) {
        FirstFragment f = new FirstFragment();
        return f;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.first_frag, container, false);
        return v;
    }
}
```

SecondFragment.java

```
package com.example.android.tabbedpages;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

/**
 * Class that controls the creation of the second fragment
 */
public class SecondFragment extends Fragment {

    public static SecondFragment newInstance(String text) {
        SecondFragment f = new SecondFragment();
        return f;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.second_frag, container, false);
        return v;
    }
}
```

Appendix 7 – Swipable Tabs app code

https://goo.gl/zU8qHC

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?><!--Keep the contents of this layout
file how they were when they were created.--><!--This makes sure that the
swipable actions and tabs remain functional. -->
<android.support.design.widget.CoordinatorLayout
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"
    android:id="@+id/main_content"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context="com.example.android.swipabletabs.MainActivity">

    <android.support.design.widget.AppBarLayout
        android:id="@+id/appbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingTop="@dimen/appbar_padding_top"
        android:theme="@style/AppTheme.AppBarOverlay">

        <!--Remove the following 'Toolbar' widget to remove the 'title
bar'-->
        <!--If doing so, remove the 'paddingTop' in the above
'AppBarLayout' declarations-->
        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:layout_scrollFlags="scroll|enterAlways"
            app:popupTheme="@style/AppTheme.PopupOverlay">

        </android.support.v7.widget.Toolbar>

        <android.support.design.widget.TabLayout
            android:id="@+id/tabs"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />

    </android.support.design.widget.AppBarLayout>

    <android.support.v4.view.ViewPager
        android:id="@+id/container"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior" />

</android.support.design.widget.CoordinatorLayout>
```

first_frag.xml

```
<?xml version="1.0" encoding="utf-8"?><!--Use this as a normal XML layout
file.-->
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.android.swipabletabs.MainActivity">

    <TextView
        android:id="@+id/section_label"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is the first page"
        android:textSize="30sp" />

    <Button
        android:id="@+id/toast"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/section_label"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="24dp"
        android:elevation="0dp"
        android:text="Button" />
</RelativeLayout>
```

second_frag.xml

```
<?xml version="1.0" encoding="utf-8"?><!--Use this as a normal XML layout
file.-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="This is page 2"
        android:textSize="30sp" />
</LinearLayout>
```

MainActivity.java

```
package com.example.android.swipabletabs;

import android.os.Bundle;
import android.support.design.widget.TabLayout;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;
import android.support.v4.view.ViewPager;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;

public class MainActivity extends AppCompatActivity {

    /**
     * The {@link android.support.v4.view.PagerAdapter} that will provide
     * fragments for each of the sections. We use a
     * {@link FragmentPagerAdapter} derivative, which will keep every
```

```
     * loaded fragment in memory. If this becomes too memory intensive, it
     * may be best to switch to a
     * {@link android.support.v4.app.FragmentStatePagerAdapter}.
     */
    private SectionsPagerAdapter mSectionsPagerAdapter;

    /**
     * The {@link ViewPager} that will host the section contents.
     * i.e the 'white-space' on the page that changes (not the header)
     */
    private ViewPager mViewPager;

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

        //These two lines populate the 'title bar' of the app
        //Removing them along with the specified code in
'activity_main.xml' will remove
        //the 'title bar' completely
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        // Create the adapter that will return a fragment for each of the
        // primary sections of the activity.
        mSectionsPagerAdapter = new
SectionsPagerAdapter(getSupportFragmentManager());

        // Set up the ViewPager with the sections adapter.
        mViewPager = (ViewPager) findViewById(R.id.container);
        mViewPager.setAdapter(mSectionsPagerAdapter);

        //Create the tabs at the top of the page
        TabLayout tabLayout = (TabLayout) findViewById(R.id.tabs);
        tabLayout.setupWithViewPager(mViewPager);
    }

    /**
     * A {@link FragmentPagerAdapter} that returns a fragment corresponding
to
     * one of the sections/tabs/pages.
     */
    public class SectionsPagerAdapter extends FragmentPagerAdapter {

        public SectionsPagerAdapter(FragmentManager fm) {
            super(fm);
        }

        /**
         * For each page (in order of navigation) return which class should
be loaded.
         * Where there are more pages created than layouts for (shouldn't
happen) the
         * second fragment will be duplicated.
         */
        @Override
        public Fragment getItem(int position) {
            switch (position) {
                case 0:
                    return FirstFragment.newInstance("FirstFragment");
```

```
                case 1:
                    return SecondFragment.newInstance("SecondFragment");
            }
            return null;
        }

        /**
         * Controls number of pages shown to user
         *
         * @return the number of pages there should be
         */
        @Override
        public int getCount() {
            return 2;
        }

        /**
         * Method to control the tab titles
         *
         * @return the titles for each of the tabs
         */
        @Override
        public CharSequence getPageTitle(int position) {
            switch (position) {
                case 0:
                    return "SECTION 1";
                case 1:
                    return "SECTION 2";
            }
            return null;
        }
    }
}
```

FirstFragment.java

```
package com.example.android.swipabletabs;

import android.content.Context;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.Toast;

/**
 * Class that controls the creation of the first fragment.
 * Everything can be put inside of it, like a normal class.
 */
public class FirstFragment extends Fragment {

    public static FirstFragment newInstance(String text) {
        FirstFragment f = new FirstFragment();
        return f;
    }

    /**
     * Handle this method as a normal 'onCreate' method, but make sure that
all commands are
```

```
     * inserted before 'return v'.
     * <p>
     * In addition, make sure that referencing 'findViewById' has a 'v.'
before it so it knows
     * to look in this specific view.
     */
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.first_frag, container, false);

        //Create a new button object constructor
        Button button = v.findViewById(R.id.toast);
        //Set a listener to it and create a new method
        button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                //Do this when the button is clicked
                toaster();
            }
        });

        return v;
    }

    /**
     * Method that creates a simple toast
     */
    public void toaster() {
        Context context = getActivity();
        CharSequence text = "The button worked!";
        int duration = Toast.LENGTH_SHORT;

        Toast toast = Toast.makeText(context, text, duration);
        toast.show();
    }
}
```

SecondFragment.java

```
package com.example.android.swipabletabs;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

/**
 * Class that controls the creation of the second fragment.
 * Everything can be put inside of it, like a normal class.
 */
public class SecondFragment extends Fragment {

    public static SecondFragment newInstance(String text) {
        SecondFragment f = new SecondFragment();
        return f;
    }

    /**
     * Handle this method as a normal 'onCreate' method, but make sure that
all commands are
```

```
     * inserted before 'return v'.
     */
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.second_frag, container, false);
        return v;
    }
}
```

Appendix 8 – Dropdown Boxes app code

https://goo.gl/EzqpLv

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.android.dropdownboxes.MainActivity">

    <!--Use the frame layout to encompass the Spinner-->
    <!--Set the height to wrap_content as the height of the text is
controlled elsewhere-->
    <!--Use the 'android:background' attribute to set the background-->
    <FrameLayout
        android:id="@+id/spinnerBox"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        android:background="@drawable/bg_spinner">

        <!--Keep the Spinner simple, give it an ID, width & height-->
        <Spinner
            android:id="@+id/spinner"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />
    </FrameLayout>

    <!--This TextView is only used to represent the text that is retrieved
from the Spinner-->
    <!--It is not necessarily needed-->
    <TextView
        android:id="@+id/spinnerText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/spinnerBox"
        android:layout_marginLeft="16dp"
        android:layout_marginRight="16dp" />
</RelativeLayout>
```

spinner_dropdown_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--This XML layout file controls the styling for the text inside the
dropdown menu-->
<!--Use a CheckedTextView for this XML file-->
<!--Make sure that attributes 'id, style, ellipsize, maxLines' are all
included-->
<!--Everything else can be customised-->
<!--48dp is the recommended height for a button/selection-->
<CheckedTextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/text1"
    style="?android:attr/spinnerDropDownItemStyle"
    android:layout_width="match_parent"
    android:layout_height="48dp"
    android:ellipsize="marquee"
    android:maxLines="1"
    android:textColor="#aa66cc"/>
```

spinner_item.xml

```
<!--This XML layout file controls the styling for the text once selected
from the spinner-->
<!--Using a TExtView works fine for this XML file-->
<!--Make sure that attribute 'id' is included-->
<!--Everything else can be customised-->
<!--Keep height to 'wrap_content' to keep the height defined by the
background XML-->
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/text1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="14sp"
    android:textColor="#ff0000" />
```

MainActivity.java

```
package com.example.android.dropdownboxes;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.TextView;

/**
 * If it isn't automatically added, make sure:
 * import android.widget.AdapterView.OnItemSelectedListener;
 * is imported
 */

public class MainActivity extends AppCompatActivity {

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

        //Call this method to initiate the creation of the Spinner
        createSpinnerOptions();

        //Get the spinner as an object and call the OnItemSelectedListener
on it to create the listener
        Spinner spinner = (Spinner) findViewById(R.id.spinner);
        spinner.setOnItemSelectedListener(new MyOnItemSelectedListener());
    }

    /**
     * The method that creates the spinner options
     */
    public void createSpinnerOptions() {
        //Create the spinner as an object
        Spinner spinner = (Spinner) findViewById(R.id.spinner);

        // Create an ArrayAdapter using the the array of options and the
style of the text before selection layout
```

```
        ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(this, R.array.planets_array,
R.layout.spinner_item);

        // Specify the layout to use when the list of choices appears
        adapter.setDropDownViewResource(R.layout.spinner_dropdown_item);

        // Apply the adapter to the spinner
        spinner.setAdapter(adapter);
    }

    /**
     * This sub-class controls what happens when the listener detects
something has been selected
     */
    public class MyOnItemSelectedListener implements OnItemSelectedListener
{
        public void onItemSelected(AdapterView<?> parent, View view, int
pos, long id) {
            //When an item is selected, do this:
            TextView spinnerStringOut = (TextView)
findViewById(R.id.spinnerText);

spinnerStringOut.setText(parent.getItemAtPosition(pos).toString());
        }

        public void onNothingSelected(AdapterView parent) {
            // Do nothing when nothing is selected.
        }
    }
}
```

Appendix 9 – Soonami app code

As mentioned in section 2.1.9 – this app was provided by the course. I may have made changes to it, so included is the repository of the original, as well as my own.

Original: https://goo.gl/Uba6LB

Mine: https://goo.gl/edKDjE

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    tools:context="com.example.android.soonami.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/header_event"
        android:textAllCaps="true"
        android:textAppearance="?android:textAppearanceSmall" />

    <TextView
        android:id="@+id/title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:textAppearanceLarge" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="16dp"
        android:text="@string/header_date"
        android:textAllCaps="true"
        android:textAppearance="?android:textAppearanceSmall" />

    <TextView
        android:id="@+id/date"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:textAppearanceLarge" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="16dp"
        android:text="@string/header_tsunami_alert"
        android:textAllCaps="true"
        android:textAppearance="?android:textAppearanceSmall" />

    <TextView
        android:id="@+id/tsunami_alert"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:textAppearanceLarge" />
```

```
</LinearLayout>
```

MainActivity.java

```
package com.example.android.soonami;

import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.text.TextUtils;
import android.util.Log;
import android.widget.TextView;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.nio.charset.Charset;
import java.text.SimpleDateFormat;

/**
 * Displays information about a single earthquake.
 */
public class MainActivity extends AppCompatActivity {

    /**
     * Tag for the log messages
     */
    public static final String LOG_TAG =
MainActivity.class.getSimpleName();

    /**
     * URL to query the USGS dataset for earthquake information
     */
    private static final String USGS_REQUEST_URL =

"https://earthquake.usgs.gov/fdsnws/event/1/query?format=geojson&starttime=
2014-01-01&endtime=2014-12-01&minmagnitude=7";

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

        // Kick off an {@link AsyncTask} to perform the network request
        TsunamiAsyncTask task = new TsunamiAsyncTask();
        task.execute();
    }

    /**
     * Update the screen to display information from the given {@link
Event}.
     */
    private void updateUi(Event earthquake) {
```

```
        // Display the earthquake title in the UI
        TextView titleTextView = (TextView) findViewById(R.id.title);
        titleTextView.setText(earthquake.title);

        // Display the earthquake date in the UI
        TextView dateTextView = (TextView) findViewById(R.id.date);
        dateTextView.setText(getDateString(earthquake.time));

        // Display whether or not there was a tsunami alert in the UI
        TextView tsunamiTextView = (TextView)
findViewById(R.id.tsunami_alert);

tsunamiTextView.setText(getTsunamiAlertString(earthquake.tsunamiAlert));
    }

    /**
     * Returns a formatted date and time string for when the earthquake
happened.
     */
    private String getDateString(long timeInMilliseconds) {
        SimpleDateFormat formatter = new SimpleDateFormat("EEE, d MMM yyyy
'at' HH:mm:ss z");
        return formatter.format(timeInMilliseconds);
    }

    /**
     * Return the display string for whether or not there was a tsunami
alert for an earthquake.
     */
    private String getTsunamiAlertString(int tsunamiAlert) {
        switch (tsunamiAlert) {
            case 0:
                return getString(R.string.alert_no);
            case 1:
                return getString(R.string.alert_yes);
            default:
                return getString(R.string.alert_not_available);
        }
    }

    /**
     * {@link AsyncTask} to perform the network request on a background
thread, and then
     * update the UI with the first earthquake in the response.
     */
    private class TsunamiAsyncTask extends AsyncTask<URL, Void, Event> {

        @Override
        protected Event doInBackground(URL... urls) {
            // Create URL object
            URL url = createUrl(USGS_REQUEST_URL);

            // Perform HTTP request to the URL and receive a JSON response
back
            String jsonResponse = "";
            try {
                jsonResponse = makeHttpRequest(url);
            } catch (IOException e) {
                // TODO Handle the IOException
            }
```

```
            // Extract relevant fields from the JSON response and create an
{@link Event} object
            Event earthquake = extractFeatureFromJson(jsonResponse);

            // Return the {@link Event} object as the result fo the {@link
TsunamiAsyncTask}
            return earthquake;
        }

        /**
         * Update the screen with the given earthquake (which was the
result of the
         * {@link TsunamiAsyncTask}).
         */
        @Override
        protected void onPostExecute(Event earthquake) {
            if (earthquake == null) {
                return;
            }

            updateUi(earthquake);
        }

        /**
         * Returns new URL object from the given string URL.
         */
        private URL createUrl(String stringUrl) {
            URL url = null;
            try {
                url = new URL(stringUrl);
            } catch (MalformedURLException exception) {
                Log.e(LOG_TAG, "Error with creating URL", exception);
                return null;
            }
            return url;
        }

        /**
         * Make an HTTP request to the given URL and return a String as the
response.
         */
        private String makeHttpRequest(URL url) throws IOException {
            String jsonResponse = "";

            //If statement to check that the URL is valid
            if (url == null) {
                return jsonResponse;
            }

            HttpURLConnection urlConnection = null;
            InputStream inputStream = null;
            try {
                urlConnection = (HttpURLConnection) url.openConnection();
                urlConnection.setRequestMethod("GET");
                urlConnection.setReadTimeout(10000 /* milliseconds */);
                urlConnection.setConnectTimeout(15000 /* milliseconds */);
                urlConnection.connect();

                if (urlConnection.getResponseCode() == 200) {
                    inputStream = urlConnection.getInputStream();
                    jsonResponse = readFromStream(inputStream);
```

```
                } else {
                    Log.e(LOG_TAG, "Error with URL response - Response
Code: " + urlConnection.getResponseCode());
                }
            } catch (IOException e) {
                Log.e(LOG_TAG, "IOException thrown: " + e);
            } finally {
                if (urlConnection != null) {
                    urlConnection.disconnect();
                }
                if (inputStream != null) {
                    // function must handle java.io.IOException here
                    inputStream.close();
                }
            }
            return jsonResponse;
        }

        /**
         * Convert the {@link InputStream} into a String which contains the
         * whole JSON response from the server.
         */
        private String readFromStream(InputStream inputStream) throws
IOException {
            StringBuilder output = new StringBuilder();
            if (inputStream != null) {
                InputStreamReader inputStreamReader = new
InputStreamReader(inputStream, Charset.forName("UTF-8"));
                BufferedReader reader = new
BufferedReader(inputStreamReader);
                String line = reader.readLine();
                while (line != null) {
                    output.append(line);
                    line = reader.readLine();
                }
            }
            return output.toString();
        }

        /**
         * Return an {@link Event} object by parsing out information
         * about the first earthquake from the input earthquakeJSON string.
         */
        private Event extractFeatureFromJson(String earthquakeJSON) {
            //If the JSOn string is empty or null, then return early
            if (TextUtils.isEmpty(earthquakeJSON)) {
                return null;
            }

            try {
                JSONObject baseJsonResponse = new
JSONObject(earthquakeJSON);
                JSONArray featureArray =
baseJsonResponse.getJSONArray("features");

                // If there are results in the features array
                if (featureArray.length() > 0) {
                    // Extract out the first feature (which is an
earthquake)
                    JSONObject firstFeature =
featureArray.getJSONObject(0);
```

```
                    JSONObject properties =
firstFeature.getJSONObject("properties");

                    // Extract out the title, time, and tsunami values
                    String title = properties.getString("title");
                    long time = properties.getLong("time");
                    int tsunamiAlert = properties.getInt("tsunami");

                    // Create a new {@link Event} object
                    return new Event(title, time, tsunamiAlert);
                }
            } catch (JSONException e) {
                Log.e(LOG_TAG, "Problem parsing the earthquake JSON
results", e);
            }
            return null;
        }
    }
}
```

Event.java

```
package com.example.android.soonami;

/**
 * {@Event} represents an earthquake event. It holds the details
 * of that event such as title (which contains magnitude and location
 * of the earthquake), as well as time, and whether or not a tsunami
 * alert was issued during the earthquake.
 */
public class Event {

    /** Title of the earthquake event */
    public final String title;

    /** Time that the earthquake happened (in milliseconds) */
    public final long time;

    /** Whether or not a tsunami alert was issued (1 if it was issued, 0 if
no alert was issued) */
    public final int tsunamiAlert;

    /**
     * Constructs a new {@link Event}.
     *
     * @param eventTitle is the title of the earthquake event
     * @param eventTime is the time the earhtquake happened
     * @param eventTsunamiAlert is whether or not a tsunami alert was
issued
     */
    public Event(String eventTitle, long eventTime, int eventTsunamiAlert)
{
        title = eventTitle;
        time = eventTime;
        tsunamiAlert = eventTsunamiAlert;
    }
}
```

Appendix 10 – Did You Feel It app code

As mentioned in section 2.1.9 – this app was provided by the course. I may have made changes to it, so included is the repository of the original, as well as my own.

Original: https://goo.gl/8RBwWh

Mine: https://goo.gl/mhw76R

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Layout for the main screen -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    tools:context="com.example.android.didyoufeelit.MainActivity">

    <TextView
        android:id="@+id/title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:paddingBottom="16dp"
        android:paddingTop="16dp"
        android:textAppearance="?android:textAppearanceLarge" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"
        android:paddingBottom="16dp"
        android:paddingTop="16dp"
        android:text="@string/perceived_strength"
        android:textAllCaps="true"
        android:textAppearance="?android:textAppearanceMedium" />

    <TextView
        android:id="@+id/perceived_magnitude"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:background="@drawable/magnitude_circle"
        android:gravity="center"
        android:textColor="@android:color/white"
        android:textSize="100sp" />

    <TextView
        android:id="@+id/number_of_people"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:paddingTop="16dp"
        android:textAllCaps="true"
        android:textAppearance="?android:textAppearanceMedium" />
</LinearLayout>
```

MainActivity.java

```
package com.example.android.didyoufeelit;

import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.widget.TextView;

/**
 * Displays the perceived strength of a single earthquake event based on
responses from people who
 * felt the earthquake.
 */
public class MainActivity extends AppCompatActivity {

    /** URL for earthquake data from the USGS dataset */
    private static final String USGS_REQUEST_URL =

"https://earthquake.usgs.gov/fdsnws/event/1/query?format=geojson&starttime=
2016-01-01&endtime=2016-05-02&minfelt=50&minmagnitude=5";

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

        // Perform the HTTP request for earthquake data and process the
response.
        //Event earthquake = Utils.fetchEarthquakeData(USGS_REQUEST_URL);
        new DownloadDataTask().execute(USGS_REQUEST_URL);

        // Update the information displayed to the user.
        //updateUi(earthquake);
    }

    /**
     * Update the UI with the given earthquake information.
     */
    private void updateUi(Event earthquake) {
        TextView titleTextView = (TextView) findViewById(R.id.title);
        titleTextView.setText(earthquake.title);

        TextView tsunamiTextView = (TextView)
findViewById(R.id.number_of_people);
        tsunamiTextView.setText(getString(R.string.num_people_felt_it,
earthquake.numOfPeople));

        TextView magnitudeTextView = (TextView)
findViewById(R.id.perceived_magnitude);
        magnitudeTextView.setText(earthquake.perceivedStrength);
    }

    /**
     * Inner Async class
     * Async task that handles gathering data in the background
     */
    private class DownloadDataTask extends AsyncTask<String, Void, Event> {

        /**
         * Execute the contents of this method in the background
         * @param urls input argument
```

```
         * @return the earthquake event
         */
        protected Event doInBackground(String... urls) {
            // Don't perform the request if there are no URLs, or the first
URL is null.
            if (urls.length < 1 || urls[0] == null) {
                return null;
            }

            Event result = Utils.fetchEarthquakeData(urls[0]);
            return result;
        }

        /**
         * Once the data has downloaded in the background, execute this
method
         * @param result as input
         */
        protected void onPostExecute(Event result) {
            // If there is no result, do nothing.
            if (result == null) {
                return;
            }

            updateUi(result);
        }
    }
}
```

Event.java

```
package com.example.android.didyoufeelit;

/**
 * {@Event} represents an earthquake event.
 */
public class Event {

    /** Title of the earthquake event */
    public final String title;

    /** Number of people who felt the earthquake and reported how strong it
was */
    public final String numOfPeople;

    /** Perceived strength of the earthquake from the people's responses */
    public final String perceivedStrength;

    /**
     * Constructs a new {@link Event}.
     *
     * @param eventTitle is the title of the earthquake event
     * @param eventNumOfPeople is the number of people who felt the
earthquake and reported how
     *                         strong it was
     * @param eventPerceivedStrength is the perceived strength of the
earthquake from the responses
     */
    public Event(String eventTitle, String eventNumOfPeople, String
eventPerceivedStrength) {
        title = eventTitle;
```

```
            numOfPeople = eventNumOfPeople;
            perceivedStrength = eventPerceivedStrength;
        }
    }
```

Utils.java

```
package com.example.android.didyoufeelit;

import android.text.TextUtils;
import android.util.Log;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.nio.charset.Charset;

/**
 * Utility class with methods to help perform the HTTP request and
 * parse the response.
 */
public final class Utils {

    /**
     * Tag for the log messages
     */
    public static final String LOG_TAG = Utils.class.getSimpleName();

    /**
     * Query the USGS dataset and return an {@link Event} object to
represent a single earthquake.
     */
    public static Event fetchEarthquakeData(String requestUrl) {
        // Create URL object
        URL url = createUrl(requestUrl);

        // Perform HTTP request to the URL and receive a JSON response back
        String jsonResponse = null;
        try {
            jsonResponse = makeHttpRequest(url);
        } catch (IOException e) {
            Log.e(LOG_TAG, "Error closing input stream", e);
        }

        // Extract relevant fields from the JSON response and create an
{@link Event} object
        Event earthquake = extractFeatureFromJson(jsonResponse);

        // Return the {@link Event}
        return earthquake;
    }

    /**
```

```
     * Returns new URL object from the given string URL.
     */
    private static URL createUrl(String stringUrl) {
        URL url = null;
        try {
            url = new URL(stringUrl);
        } catch (MalformedURLException e) {
            Log.e(LOG_TAG, "Error with creating URL ", e);
        }
        return url;
    }

    /**
     * Make an HTTP request to the given URL and return a String as the
response.
     */
    private static String makeHttpRequest(URL url) throws IOException {
        String jsonResponse = "";

        // If the URL is null, then return early.
        if (url == null) {
            return jsonResponse;
        }

        HttpURLConnection urlConnection = null;
        InputStream inputStream = null;
        try {
            urlConnection = (HttpURLConnection) url.openConnection();
            urlConnection.setReadTimeout(10000 /* milliseconds */);
            urlConnection.setConnectTimeout(15000 /* milliseconds */);
            urlConnection.setRequestMethod("GET");
            urlConnection.connect();

            // If the request was successful (response code 200),
            // then read the input stream and parse the response.
            if (urlConnection.getResponseCode() == 200) {
                inputStream = urlConnection.getInputStream();
                jsonResponse = readFromStream(inputStream);
            } else {
                Log.e(LOG_TAG, "Error response code: " +
urlConnection.getResponseCode());
            }
        } catch (IOException e) {
            Log.e(LOG_TAG, "Problem retrieving the earthquake JSON
results.", e);
        } finally {
            if (urlConnection != null) {
                urlConnection.disconnect();
            }
            if (inputStream != null) {
                inputStream.close();
            }
        }
        return jsonResponse;
    }

    /**
     * Convert the {@link InputStream} into a String which contains the
     * whole JSON response from the server.
     */
```

```
    private static String readFromStream(InputStream inputStream) throws
IOException {
        StringBuilder output = new StringBuilder();
        if (inputStream != null) {
            InputStreamReader inputStreamReader = new
InputStreamReader(inputStream, Charset.forName("UTF-8"));
            BufferedReader reader = new BufferedReader(inputStreamReader);
            String line = reader.readLine();
            while (line != null) {
                output.append(line);
                line = reader.readLine();
            }
        }
        return output.toString();
    }

    /**
     * Return an {@link Event} object by parsing out information
     * about the first earthquake from the input earthquakeJSON string.
     */
    private static Event extractFeatureFromJson(String earthquakeJSON) {
        // If the JSON string is empty or null, then return early.
        if (TextUtils.isEmpty(earthquakeJSON)) {
            return null;
        }

        try {
            JSONObject baseJsonResponse = new JSONObject(earthquakeJSON);
            JSONArray featureArray =
baseJsonResponse.getJSONArray("features");

            // If there are results in the features array
            if (featureArray.length() > 0) {
                // Extract out the first feature (which is an earthquake)
                JSONObject firstFeature = featureArray.getJSONObject(0);
                JSONObject properties =
firstFeature.getJSONObject("properties");

                // Extract out the title, number of people, and perceived
strength values
                String title = properties.getString("title");
                String numberOfPeople = properties.getString("felt");
                String perceivedStrength = properties.getString("cdi");

                // Create a new {@link Event} object
                return new Event(title, numberOfPeople, perceivedStrength);
            }
        } catch (JSONException e) {
            Log.e(LOG_TAG, "Problem parsing the earthquake JSON results",
e);
        }
        return null;
    }
}
```

Appendix 11 – Quake Report app code

As mentioned in section 2.1.10 – this app was provided by the course. I may have made changes to it, so included is the repository of the original, as well as my own.

Original: https://goo.gl/48GKsT

Mine: https://goo.gl/az9Hyf

earthquake_activity.xml

```
<?xml version="1.0" encoding="utf-8"?><!-- Layout for a list of earthquakes
-->
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ListView
        android:id="@+id/list"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" />

    <!-- Empty view is only visible when the list has no items. -->
    <TextView
        android:id="@+id/empty_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:textAppearance="?android:textAppearanceMedium" />

    <!--Progress bar to let the user know that data is being fetched-->
    <ProgressBar
        android:id="@+id/loading_spinner"
        style="@style/Widget.AppCompat.ProgressBar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true" />
</RelativeLayout>
```

element_layout.xml

```
<?xml version="1.0" encoding="utf-8"?><!--The custom list layout-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="16dp">

    <LinearLayout
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_weight="1">

        <TextView
            android:id="@+id/magnitude"
            android:layout_width="36dp"
            android:layout_height="36dp"
            android:background="@drawable/magnitude_circle"
            android:gravity="center"
```

```
            android:textAppearance="?android:attr/textAppearanceListItem"
            android:textColor="@android:color/white"
            tools:text="7.2" />
    </LinearLayout>

    <LinearLayout
        android:id="@+id/locationViews"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_marginLeft="16dp"
        android:layout_marginStart="16dp"
        android:layout_weight="4"
        android:orientation="vertical">

        <TextView
            android:id="@+id/locationOffset"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"

android:textAppearance="?android:attr/textAppearanceListItemSecondary"
            android:textColor="@color/secondaryText"
            tools:text="74km NW of" />

        <TextView
            android:id="@+id/primaryLocation"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ellipsize="marquee"
            android:maxLines="2"

android:textAppearance="?android:attr/textAppearanceListItemSmall"
            android:textColor="@color/primaryText"
            tools:text="San Francisco" />
    </LinearLayout>

    <LinearLayout
        android:id="@+id/dateViews"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_marginLeft="16dp"
        android:layout_weight="2"
        android:orientation="vertical">

        <TextView
            android:id="@+id/date"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="right"

android:textAppearance="?android:attr/textAppearanceListItemSecondary"
            android:textColor="@color/secondaryText"
            tools:text="Jan 31, 2013" />

        <TextView
            android:id="@+id/time"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="right"
```

```
android:textAppearance="?android:attr/textAppearanceListItemSecondary"
            android:textColor="@color/secondaryText"
            tools:text="3:00 PM" />
    </LinearLayout>
</LinearLayout>
```

EarthquakeActivity.java

```
package com.example.android.quakereport;

import android.app.LoaderManager;
import android.content.Context;
import android.content.Intent;
import android.content.Loader;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.Uri;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.ProgressBar;
import android.widget.TextView;

import java.util.ArrayList;
import java.util.List;

/**
 * Implement the Loader Manager, so that that background tasks can be
completed in a resource efficient way
 */
public class EarthquakeActivity extends AppCompatActivity implements
LoaderManager.LoaderCallbacks<List<Earthquake>> {

    //Log tag that returns the package name for errors
    public static final String LOG_TAG =
EarthquakeActivity.class.getName();

    //URL that provides the JSON response from the USGS site
    private static final String REQUEST_URL =

"https://earthquake.usgs.gov/fdsnws/event/1/query?format=geojson&eventtype=
earthquake&orderby=time&minmag=3&limit=1000";

    //Constant value for the earthquake loader ID. This really only comes
into play if you're using multiple loaders.
    private static final int EARTHQUAKE_LOADER_ID = 1;

    //Global instance of the EarthquakeAdapter, so it can be used in
multiple methods in this class
    private EarthquakeAdapter mAdapter;

    //Global instance of the TextView that is displayed when the list is
empty
    private TextView mEmptyStateTextView;

    //Global instance of the ProgressBar, so it can be used in multiple
methods in this class
    private ProgressBar mProgressBar;
```

```
    /**
     * OnCreate method for this activity
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.earthquake_activity);

        //Check if the user is connected to the Internet
        if (!checkInternetConnectivity()) {
            //Hide the loading spinner
            mProgressBar = (ProgressBar)
findViewById(R.id.loading_spinner);
            mProgressBar.setVisibility(View.GONE);

            // Set empty state text to display "No earthquakes found."
            mEmptyStateTextView = (TextView) findViewById(R.id.empty_view);
            mEmptyStateTextView.setText(R.string.no_connectivity);
        } else {
            //Creates a object constructor for the ListView
            ListView earthquakeListView = (ListView)
findViewById(R.id.list);

            //Find and set the empty text view as the empty view for the
layout
            mEmptyStateTextView = (TextView) findViewById(R.id.empty_view);
            earthquakeListView.setEmptyView(mEmptyStateTextView);

            //Creates an adapter for the words to use, appends the array of
words to the adapter,
            //the adapter is responsible for making a View for each item in
the data set
            mAdapter = new EarthquakeAdapter(this, new
ArrayList<Earthquake>());

            //Sets the adapter method on the ListView
            //so the list can be populated in the user interface
            earthquakeListView.setAdapter(mAdapter);

            //On click listener for each item
            earthquakeListView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
                @Override
                public void onItemClick(AdapterView<?> adapterView, View
view, int position, long l) {
                    //Find the current earthquake that was clicked on
                    Earthquake currentEarthquake =
mAdapter.getItem(position);

                    //Convert the String URL into a URI object (to pass
into the Intent constructor)
                    Uri earthquakeUri =
Uri.parse(currentEarthquake.getDetailUrl());

                    //Load the webpage
                    openWebPage(earthquakeUri);
                }
            });
```

```
            //Get a reference to the LoaderManager, in order to interact
with loaders.
            LoaderManager loaderManager = getLoaderManager();

            //Initialize the loader. Pass in the int ID constant defined
above and pass in null for
            //the bundle. Pass in this activity for the LoaderCallbacks
parameter (which is valid
            //because this activity implements the LoaderCallbacks
interface).
            loaderManager.initLoader(EARTHQUAKE_LOADER_ID, null, this);
        }
    }

    /**
     * Method that checks if the user is connected to the Internet
     *
     * @return true or false
     */
    private boolean checkInternetConnectivity() {
        //Create a connectivity manager
        ConnectivityManager cm = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);

        //Get the active network's network info, and return a boolean value
        NetworkInfo activeNetwork = cm.getActiveNetworkInfo();
        return activeNetwork != null &&
activeNetwork.isConnectedOrConnecting();
    }

    /**
     * Method for the creation of the loader, pass in:
     * The Loader {@param id} to be used
     * The {@param args} (arguments) the Loader should take
     *
     * @return a new instance of the Loader
     */
    @Override
    public Loader<List<Earthquake>> onCreateLoader(int id, Bundle args) {
        return new EarthquakeLoader(this, REQUEST_URL);
    }

    /**
     * Once the loader has finished, execute this method with:
     * The {@param loader} to be used
     * The {@param result} from the Loader creation
     */
    @Override
    public void onLoadFinished(Loader<List<Earthquake>> loader,
List<Earthquake> result) {
        //Hide the loading spinner
        mProgressBar = (ProgressBar) findViewById(R.id.loading_spinner);
        mProgressBar.setVisibility(View.GONE);

        // Set empty state text to display "No earthquakes found."
        mEmptyStateTextView.setText(R.string.no_earthquakes);

        //Clear the adapter of previous data
        mAdapter.clear();
```

```
        //If there is a valid list of {@Link Earthquake}'s, then add them
to the adapter's dataset
        //This triggers the ListView to update
        if (result != null && !result.isEmpty()) {
            mAdapter.addAll(result);
        }
    }

    /**
     * If the Loader is reset (i.e. through orientation change), handle
that in this method
     * The {@param loader} to be used
     */
    @Override
    public void onLoaderReset(Loader<List<Earthquake>> loader) {
        // Loader reset, so we can clear out our existing data.
        mAdapter.clear();
    }

    /**
     * Opens an intent using the URL passed in a parameter
     * The {@param url} to be used
     */
    public void openWebPage(Uri url) {
        //Create an intent
        Intent intent = new Intent(Intent.ACTION_VIEW, url);
        if (intent.resolveActivity(getPackageManager()) != null) {
            //If possible, open the intent
            startActivity(intent);
        }
    }
}
```

Earthquake.java

```
package com.example.android.quakereport;

/**
 * The Earthquake class that creates and provides methods that return parts
of the ArrayList when called
 * by other parts of the app
 */
public class Earthquake {

    //Declare private variables for this class to use
    private double mMagnitude;
    private String mLocation;
    private long mTimeInMilliseconds;
    private String mDetailUrl;

    /**
     * Create the constructor for this class, a constructor creates an
instance of a class
     * This constructor will create an instance of a double, long and
string
     *
     * @param magnitude
     * @param location
     * @param timeInMilliseconds
     * @param detailUrl
     */
```

```
    public Earthquake(double magnitude, String location, long
timeInMilliseconds, String detailUrl) {
        mMagnitude = magnitude;
        mLocation = location;
        mTimeInMilliseconds = timeInMilliseconds;
        mDetailUrl = detailUrl;
    }

    /**
     * @return the magnitude of the Earthquake
     */
    public double getMagnitude() {
        return mMagnitude;
    }

    /**
     * @return the location of the Earthquake
     */
    public String getLocation() {
        return mLocation;
    }

    /**
     * @return the date of the Earthquake
     */
    public long getTimeInMilliseconds() {
        return mTimeInMilliseconds;
    }

    /**
     * @return the URL associated with the Earthquake
     */
    public String getDetailUrl() {
        return mDetailUrl;
    }
}
```

EarthquakeAdaptor.java

```
package com.example.android.quakereport;

import android.app.Activity;
import android.graphics.drawable.GradientDrawable;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.content.ContextCompat;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;

import static com.example.android.quakereport.R.id.magnitude;

/**
 * Earthquake adapter class that handles the multiple TextViews for the
ListView
```

```
 */
public class EarthquakeAdapter extends ArrayAdapter<Earthquake> {

    /**
     * A custom constructor.
     *
     * @param context     is used to inflate the layout file
     * @param earthquakes is the data we want to populate into the lists
     */
    public EarthquakeAdapter(Activity context, List<Earthquake>
earthquakes) {
        // This initialises the ArrayAdapter's internal storage for the
context and the list.
        // The second argument is used when the ArrayAdapter is populating
a single TextView.
        // Because this is a custom adapter for two TextViews the adapter
is not
        // going to use this second argument, so it can be any value.
        super(context, 0, earthquakes);
    }

    /**
     * Provides a view for an AdapterView (ListView, GridView, etc.)
     *
     * @param position    is the position in the list of data that should
be displayed in the list item view.
     * @param convertView The recycled view to populate.
     * @param parent      The parent ViewGroup that is used for inflation.
     * @return The View for the position in the AdapterView.
     */
    @NonNull
    @Override
    public View getView(int position, @Nullable View convertView, @NonNull
ViewGroup parent) {
        //Get the data item for this position
        Earthquake currentEarthquake = getItem(position);

        //Check if an existing view is being reused, otherwise inflate the
view
        if (convertView == null) {
            convertView =
LayoutInflater.from(getContext()).inflate(R.layout.element_layout, parent,
false);
        }

        //Lookup the views for the data population
        TextView magnitudeView = (TextView)
convertView.findViewById(magnitude);
        TextView locationView = (TextView)
convertView.findViewById(R.id.primaryLocation);
        TextView offsetView = (TextView)
convertView.findViewById(R.id.locationOffset);
        TextView dateView = (TextView) convertView.findViewById(R.id.date);
        TextView timeView = (TextView) convertView.findViewById(R.id.time);

        //Populate the data into the template view using the data object
        magnitudeView.setText("" + currentEarthquake.getMagnitude());

        // Set the proper background colour on the magnitude circle.
        // Fetch the background from the TextView, which is a
GradientDrawable.
```

```
        GradientDrawable magnitudeCircle = (GradientDrawable)
magnitudeView.getBackground();
        // Get the appropriate background colour based on the current
earthquake magnitude
        int magnitudeColour =
getMagnitudeColour(currentEarthquake.getMagnitude());
        // Set the colour on the magnitude circle
        magnitudeCircle.setColor(magnitudeColour);

        //Create a date from the milliseconds provided in the time
        Date dateObject = new
Date(currentEarthquake.getTimeInMilliseconds());
        //Format the date & time
        String formattedDate = formatDate(dateObject);
        String formattedTime = formatTime(dateObject);
        //Populate that date & time into the respective TextViews
        dateView.setText(formattedDate);
        timeView.setText(formattedTime);

        //Get string of original location
        String ogLocation = currentEarthquake.getLocation();
        //Get index location of "of" in string
        int ofIndex = ogLocation.indexOf("of");

        //If statement to check if the string has an offset or not
        if (ogLocation.contains("of")) {
            //Create the relevant substrings using "of" as a separator
location
            String offset = ogLocation.substring(0, ofIndex + 3);
            String location = ogLocation.substring(ofIndex + 3,
ogLocation.length());
            //Populate the views with the correct data
            offsetView.setText(offset);
            locationView.setText(location);
        } else {
            //Substitute some text for the offset and set the data to the
TextViews
            offsetView.setText("Near to the");
            locationView.setText(ogLocation);
        }

        //Return the completed view to render on-screen
        return convertView;
    }

    /**
     * Return the formatted date string (i.e. "Mar 3, 1984") from a Date
object.
     */
    private String formatDate(Date dateObject) {
        SimpleDateFormat dateFormat = new SimpleDateFormat("MMM dd, yyyy");
        return dateFormat.format(dateObject);
    }

    /**
     * Return the formatted date string (i.e. "4:30 PM") from a Date
object.
     */
    private String formatTime(Date dateObject) {
        SimpleDateFormat timeFormat = new SimpleDateFormat("HH:mm");
        return timeFormat.format(dateObject);
```

```
    }

    /**
     * @param magnitude - based on the input
     * @return the correct colour ID for the magnitude circle
     */
    private int getMagnitudeColour(double magnitude) {
        int magColourID;
        //Convert double to int, so switch can work
        int magFloor = (int) Math.floor(magnitude);
        switch (magFloor) {
            case 0:
            case 1:
                magColourID = R.color.magnitude1;
                break;
            case 2:
                magColourID = R.color.magnitude2;
                break;
            case 3:
                magColourID = R.color.magnitude3;
                break;
            case 4:
                magColourID = R.color.magnitude4;
                break;
            case 5:
                magColourID = R.color.magnitude5;
                break;
            case 6:
                magColourID = R.color.magnitude6;
                break;
            case 7:
                magColourID = R.color.magnitude7;
                break;
            case 8:
                magColourID = R.color.magnitude8;
                break;
            case 9:
                magColourID = R.color.magnitude9;
                break;
            default:
                magColourID = R.color.magnitude10plus;
                break;
        }
        //Return the colour, instead of the colour ID
        return ContextCompat.getColor(getContext(), magColourID);
    }
}
```

EarthquakeLoader.java

```
package com.example.android.quakereport;

import android.content.AsyncTaskLoader;
import android.content.Context;

import java.util.List;

/**
 * Earthquake loader task
 */
public class EarthquakeLoader extends AsyncTaskLoader<List<Earthquake>> {
```

```
    //Log tag that returns the package name for errors
    public static final String LOG_TAG =
EarthquakeActivity.class.getName();

    //Global instance of the String Url, so it can be used in multiple
methods in this class
    private String mUrl;

    /**
     * The constructor for this class
     */
    public EarthquakeLoader(Context context, String url) {
        super(context);
        mUrl = url;
    }

    /**
     * When the Loader is instructed to load
     */
    @Override
    protected void onStartLoading() {
        //Force the load
        forceLoad();
    }

    /**
     * The background process for the Loader
     *
     * @return
     */
    @Override
    public List<Earthquake> loadInBackground() {
        // Don't perform the request if there are no URLs, or the URL is
null.
        if (mUrl == null) {
            return null;
        }

        //Get the data for the URL provided
        List<Earthquake> result = QueryUtils.fetchEarthquakeData(mUrl);

        //Return the result
        return result;
    }
}
```

QueryUtils.java

```
package com.example.android.quakereport;

import android.text.TextUtils;
import android.util.Log;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
```

```
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.nio.charset.Charset;
import java.util.ArrayList;
import java.util.List;

/**
 * Helper methods related to requesting and receiving earthquake data from
USGS.
 */
public final class QueryUtils {

    //Tag for the log messages
    public static final String LOG_TAG = QueryUtils.class.getSimpleName();

    /**
     * Query the USGS dataset and return an {@link List<Earthquake>} object
to represent a single earthquake.
     */
    public static List<Earthquake> fetchEarthquakeData(String requestUrl) {
        //Create a URL object
        URL url = createUrl(requestUrl);

        //Perform a HTTP request to the URL and receive a JSON response
back
        //Initialise variable
        String jsonResponse = null;

        //Try the creation of a JSON response from the URL, if not catch
the exception
        try {
            jsonResponse = makeHttpRequest(url);
        } catch (IOException e) {
            //Log the error
            Log.e(LOG_TAG, "Error closing input stream", e);
        }

        //Extract the relevant fields from the JSON response and create an
List<Earthquake> object
        List<Earthquake> earthquake =
extractFeaturesFromJson(jsonResponse);

        //Return the List<Earthquake>
        return earthquake;
    }

    /**
     * Returns new URL object from the given string URL.
     */
    private static URL createUrl(String stringUrl) {
        //Initialise variable
        URL url = null;

        //Try the creation of a URL, if not catch the exception
        try {
            url = new URL(stringUrl);
        } catch (MalformedURLException e) {
            //Log the error
            Log.e(LOG_TAG, "Error with creating URL", e);
```

```
        }
        //Return the successful URL
        return url;
    }

    /**
     * Make an HTTP request to the given URL and return a String as the
response.
     */
    private static String makeHttpRequest(URL url) throws IOException {
        //Initialise variable
        String jsonResponse = "";

        //If the URL is null, then return early with an empty String
        if (url == null) {
            return jsonResponse;
        }

        //Initialise variables
        HttpURLConnection urlConnection = null;
        InputStream inputStream = null;

        //Try the connecting to the URL and reading the response, else
catch the exception
        try {
            //Open the connection to the URL
            urlConnection = (HttpURLConnection) url.openConnection();

            //Set timeouts so if they expire before there is data available
an error is thrown
            //Makes sure the user is never waiting for long periods of time
if there is no data
            urlConnection.setReadTimeout(10000 /* milliseconds */);
            urlConnection.setConnectTimeout(15000 /* milliseconds */);

            //Set the request method, as data is wanted to be received
            urlConnection.setRequestMethod("GET");

            //Make the connection
            urlConnection.connect();

            //If the request was successful (response code 200), then read
the input and parse the response
            if (urlConnection.getResponseCode() == 200) {
                inputStream = urlConnection.getInputStream();
                jsonResponse = readFromStream(inputStream);
            } else {
                //Log the error
                Log.e(LOG_TAG, "Error response code: " +
urlConnection.getResponseCode());
            }
        } catch (IOException e) {
            //Log the error
            Log.e(LOG_TAG, "Problem retrieving the JSON results", e);
        } finally {
            //As long there is an open connection
            if (urlConnection != null) {
                //Close the URL connection
                urlConnection.disconnect();
            }
            //As long as the inputStream is open
```

```
            if (inputStream != null) {
                //Close the inputStream
                inputStream.close();
            }
        }
        //Return the JSON response
        return jsonResponse;
    }

    /**
     * Convert the {@link InputStream} into a String which contains the
     * whole JSON response from the server.
     */
    private static String readFromStream(InputStream inputStream) throws
IOException {
        //Initialise variable / create a new String Builder (S.B.)
        //S.B. builds one String bit by bit instead of appending more
information each time through other variables
        StringBuilder output = new StringBuilder();

        //As long as the inputStream is not empty
        if (inputStream != null) {
            //Create a new InputStreamReader, using the inputStream and
"UTF-8" character set
            InputStreamReader inputStreamReader = new
InputStreamReader(inputStream, Charset.forName("UTF-8"));

            //Create a buffered reader, which stores information about the
characters around each character
            BufferedReader reader = new BufferedReader(inputStreamReader);

            //Get the line of text
            String line = reader.readLine();

            //While the line isn't empty
            while (line != null) {
                //Append the line to the output
                output.append(line);
                //Goto the next line
                line = reader.readLine();
            }
        }
        //Return the completed output in String format
        return output.toString();
    }

    /**
     * Return an {@link List<Earthquake>} item by parsing out information
     * about the earthquake from the input jsonResponse string.
     */
    private static List<Earthquake> extractFeaturesFromJson(String
jsonResponse) {
        //If the JSON String is empty or null, then return early
        if (TextUtils.isEmpty(jsonResponse)) {
            return null;
        }

        //Create an empty ArrayList that we can start adding earthquakes to
        List<Earthquake> earthquakes = new ArrayList<>();

        // Try to parse the jsonResponse, else catch the JSONException
```

```
        try {
            //Get the root of the JSON string
            JSONObject root = new JSONObject(jsonResponse);

            //Get the "features" array
            JSONArray features = root.getJSONArray("features");

            //Loop through each element of features
            for (int i = 0; i < features.length(); i++) {
                //Get the relevant element
                JSONObject element = features.getJSONObject(i);

                //Get the "properties" object
                JSONObject properties =
element.getJSONObject("properties");

                //Get the data that is needed
                //Declare variable out of try-catch
                double magnitude;

                //Sometimes magnitude is given as null on USGS, try-catch
block prevents crashing
                try {
                    magnitude = properties.getDouble("mag");
                } catch (JSONException e) {
                    //If the error is thrown, then set a default value of 0
                    magnitude = 0;
                }

                String location = properties.getString("place");
                long timeInMilliseconds = properties.getLong("time");
                String detailUrl = properties.getString("url");

                //Add the data to a new ArrayList element
                earthquakes.add(new Earthquake(magnitude, location,
timeInMilliseconds, detailUrl));
            }
        } catch (JSONException e) {
            //Log the error
            Log.e("QueryUtils", "Problem parsing the earthquake JSON
results", e);
        }
        return earthquakes;
    }
}
```

Appendix 12 – Pets app code

As mentioned in section 2.1.11 – this app was provided by the course. I may have made changes to it, so included is the repository of the original, as well as my own.

Original: https://goo.gl/HwMPaU

Mine: https://goo.gl/uFTkm2

activity_catalog.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Layout for the list of pets -->
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".CatalogActivity">

    <TextView
        android:id="@+id/text_view_pet"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:padding="@dimen/activity_margin"/>

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_margin="@dimen/fab_margin"
        android:src="@drawable/ic_add_pet"/>
</RelativeLayout>
```

activity_editor.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Layout for the editor -->
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="@dimen/activity_margin"
    tools:context=".EditorActivity">

    <!-- Overview category -->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <!-- Label -->
        <TextView
            android:text="@string/category_overview"
            style="@style/CategoryStyle" />

        <!-- Input fields -->
        <LinearLayout
```

```
            android:layout_height="wrap_content"
            android:layout_width="0dp"
            android:layout_weight="2"
            android:paddingLeft="4dp"
            android:orientation="vertical">

            <!-- Name field -->
            <EditText
                android:id="@+id/edit_pet_name"
                android:hint="@string/hint_pet_name"
                android:inputType="textCapWords"
                style="@style/EditorFieldStyle" />

            <!-- Breed field -->
            <EditText
                android:id="@+id/edit_pet_breed"
                android:hint="@string/hint_pet_breed"
                android:inputType="textCapWords"
                style="@style/EditorFieldStyle" />
        </LinearLayout>
    </LinearLayout>

    <!-- Gender category -->
    <LinearLayout
        android:id="@+id/container_gender"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <!-- Label -->
        <TextView
            android:text="@string/category_gender"
            style="@style/CategoryStyle" />

        <!-- Input field -->
        <LinearLayout
            android:layout_height="wrap_content"
            android:layout_width="0dp"
            android:layout_weight="2"
            android:orientation="vertical">

            <!-- Gender drop-down spinner -->
            <Spinner
                android:id="@+id/spinner_gender"
                android:layout_height="48dp"
                android:layout_width="wrap_content"
                android:paddingRight="16dp"
                android:spinnerMode="dropdown"/>
        </LinearLayout>
    </LinearLayout>

    <!-- Measurement category -->
    <LinearLayout
        android:id="@+id/container_measurement"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <!-- Label -->
        <TextView
            android:text="@string/category_measurement"
```

```
            style="@style/CategoryStyle" />

        <!-- Input fields -->
        <RelativeLayout
            android:layout_height="wrap_content"
            android:layout_width="0dp"
            android:layout_weight="2"
            android:paddingLeft="4dp">

            <!-- Weight field -->
            <EditText
                android:id="@+id/edit_pet_weight"
                android:hint="@string/hint_pet_weight"
                android:inputType="number"
                style="@style/EditorFieldStyle" />

            <!-- Units for weight (kg) -->
            <TextView
                android:id="@+id/label_weight_units"
                android:text="@string/unit_pet_weight"
                style="@style/EditorUnitsStyle"/>
        </RelativeLayout>
    </LinearLayout>
</LinearLayout>
```

CatalogActivity.java

```
package com.example.android.pets;

import android.content.ContentValues;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.TextView;

import com.example.android.pets.data.PetContract.PetsEntry;
import com.example.android.pets.data.PetDBHelper;

/**
 * This is the MainActivity, it displays a list of pets that were entered
and stored in the app.
 */
public class CatalogActivity extends AppCompatActivity {

    //Class-wide instance variable of the DB helper class
    private PetDBHelper mDbHelper;

    /**
     * Override method that deals with the creation of this activity
     * @param savedInstanceState - if any non-persistent data is stored,
this brings it back in if
     * the activity needs to be recreated (e.g. an orientation change)
     */
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_catalog);

        // Setup FAB to open EditorActivity
        FloatingActionButton fab = (FloatingActionButton)
findViewById(R.id.fab);
        //Add a listener to the FAB
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //Open the EditorActivity
                Intent intent = new Intent(CatalogActivity.this,
EditorActivity.class);
                startActivity(intent);
            }
        });

        //Create an instance of the PetDBHelper class
        mDbHelper = new PetDBHelper(this);
        //Refresh the page with the latest DB results
        displayDatabaseInfo();
    }

    /**
     * Override method for when the activity starts, but has already been
created
     * e.g. returning from another activity
     */
    @Override
    protected void onStart() {
        super.onStart();
        //Show the latest DB results
        displayDatabaseInfo();
    }

    /**
     * Temporary helper method to display information in the onscreen
TextView about the state of the DB
     */
    private void displayDatabaseInfo() {
        //To access the database, the subclass of SQLiteOpenHelper is
instantiated and is passed the context: the current activity
        PetDBHelper mDbHelper = new PetDBHelper(this);

        //Create and/or open a database object to read from it
        //If the database already exists a connection to it will be
established, as opposed to overwriting it
        SQLiteDatabase db = mDbHelper.getReadableDatabase();

        //The projection (the columns to return)
        String[] projection = {
                PetsEntry._ID,
                PetsEntry.COLUMN_PET_NAME,
                PetsEntry.COLUMN_PET_BREED,
                PetsEntry.COLUMN_PET_GENDER,
                PetsEntry.COLUMN_PET_WEIGHT
        };

        //An example selection (the WHERE part of the SQL query)
        //String selection = PetsEntry._ID + "=?";
```

```
        //The selection arguments, separated to prevent an SQL injection
        //String[] selectionArgs = {"1"};

        //Get a Cursor that returns all columns and rows from the pets
table
        Cursor cursor = db.query(
                PetsEntry.TABLE_NAME, //The table to query
                projection, //The columns to return
                null, //The columns for the WHERE clause
                null, //The values for the WHERE clause
                null, //Group by
                null, //Filter by
                null //Sort/order by
        );

        //Get the TextView as an object
        TextView displayView = (TextView) findViewById(R.id.text_view_pet);

        //Use a try:finally block so that the cursor can be closed
        try {
            //Create a header in the Text View that looks like this:
            //
            //The pets table contains <number of rows in Cursor> pets.
            // _id - name - breed - gender - weight
            //
            //In the while loop below, iterate through the rows of the
cursor and display
            //the information from each column in this order.
            displayView.setText("The pets table contains " +
cursor.getCount() + " pets.\n\n");
            //Use append so that more text can be added without the need
for extra variables
            displayView.append(
                    PetsEntry._ID + " - " +
                            PetsEntry.COLUMN_PET_NAME + " - " +
                            PetsEntry.COLUMN_PET_BREED + " - " +
                            PetsEntry.COLUMN_PET_GENDER + " - " +
                            PetsEntry.COLUMN_PET_WEIGHT + "\n"
            );

            //Figure out the index of each column
            int idColumnIndex = cursor.getColumnIndex(PetsEntry._ID);
            int nameColumnIndex =
cursor.getColumnIndex(PetsEntry.COLUMN_PET_NAME);
            int breedColumnIndex =
cursor.getColumnIndex(PetsEntry.COLUMN_PET_BREED);
            int genderColumnIndex =
cursor.getColumnIndex(PetsEntry.COLUMN_PET_GENDER);
            int weightColumnIndex =
cursor.getColumnIndex(PetsEntry.COLUMN_PET_WEIGHT);

            //Iterate through each returned row in the cursor, one at a
time
            while (cursor.moveToNext()) {
                // Use the indexes above to extract the String or Int value
of the word at the current row the cursor is on.
                int currentID = cursor.getInt(idColumnIndex);
                String currentName = cursor.getString(nameColumnIndex);
                String currentBreed = cursor.getString(breedColumnIndex);
                int currentGender = cursor.getInt(genderColumnIndex);
                int currentWeight = cursor.getInt(weightColumnIndex);
```

```
                //Display the values from each column of the current row in
the cursor in the TextView
                displayView.append(("\n" +
                        currentID + " - " +
                        currentName + " - " +
                        currentBreed + " - " +
                        currentGender + " - " +
                        currentWeight
                ));
            }
        } finally {
            //Close the cursor now that's it finished with. This releases
all its resources and makes it invalid.
            cursor.close();
        }
    }

    /**
     * Method that inserts a new pet into the database
     */
    private void insertPet() {
        //Create an instance of the writable database
        SQLiteDatabase db = mDbHelper.getWritableDatabase();

        //Create the key:value pairs
        ContentValues values = new ContentValues();
        values.put(PetsEntry.COLUMN_PET_NAME, "Toto");
        values.put(PetsEntry.COLUMN_PET_BREED, "Terrier");
        values.put(PetsEntry.COLUMN_PET_GENDER, PetsEntry.GENDER_MALE);
        values.put(PetsEntry.COLUMN_PET_WEIGHT, 7);

        //Insert the data into a new row, with the row ID being returned
        long newRowId = db.insert(PetsEntry.TABLE_NAME, null, values);

        Log.i("Catalog activity", "Row ID: " + newRowId);
    }

    /**
     * Override method that creates the menu options
     *
     * @param menu - the menu that should be passed in
     * @return - the inflated menu
     */
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu options from the res/menu/menu_catalog.xml
file.
        // This adds menu items to the app bar.
        getMenuInflater().inflate(R.menu.menu_catalog, menu);
        return true;
    }

    /**
     * Override method that handles the selection of the menu items
     *
     * @param item - the particular menu item that the user has selected
     * @return - the actions that clicking on the menu item triggers
     */
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
```

```
        //Switch statement for when a user clicks on an overflow menu item
        switch (item.getItemId()) {
            //Respond to a click on the "Insert dummy data" menu option
            case R.id.action_insert_dummy_data:
                insertPet();
                displayDatabaseInfo();
                return true;
            //Respond to a click on the "Delete all entries" menu option
            case R.id.action_delete_all_entries:
                //Do nothing for now
                return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

EditorActivity.java

```
package com.example.android.pets;

import android.content.ContentValues;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.support.v4.app.NavUtils;
import android.support.v7.app.AppCompatActivity;
import android.text.TextUtils;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.Toast;

import com.example.android.pets.data.PetContract.PetsEntry;
import com.example.android.pets.data.PetDBHelper;

/**
 * This is the EditorActivity class, it allows the user to create a new pet
or edit an existing one.
 */
public class EditorActivity extends AppCompatActivity {

    //Class-wide instance variables
    private EditText mNameEditText; //EditText field to enter the pet's
name
    private EditText mBreedEditText; //EditText field to enter the pet's
breed
    private EditText mWeightEditText; //EditText field to enter the pet's
weight
    private Spinner mGenderSpinner; //EditText field to enter the pet's
gender
    private PetDBHelper mDbHelper; //The DB helper class
    private int mGender = 0; //Gender of the pet. The possible values are:
0-unknown, 1-male, 2-female

    /**
     * Override method for onCreate
     */
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_editor);

        // Find all relevant views that we will need to read user input
from
        mNameEditText = (EditText) findViewById(R.id.edit_pet_name);
        mBreedEditText = (EditText) findViewById(R.id.edit_pet_breed);
        mWeightEditText = (EditText) findViewById(R.id.edit_pet_weight);
        mGenderSpinner = (Spinner) findViewById(R.id.spinner_gender);

        //Set-up the spinner with its relevant options
        setupSpinner();

        //Create an instance of the PetDBHelper class
        mDbHelper = new PetDBHelper(this);
    }

    /**
     * Setup the dropdown spinner that allows the user to select the gender
of the pet.
     */
    private void setupSpinner() {
        //Create an adapter for the spinner. The list options it will use
are from the String array, it will use the default layout
        ArrayAdapter genderSpinnerAdapter =
ArrayAdapter.createFromResource(this, R.array.array_gender_options,
android.R.layout.simple_spinner_item);

        //Specify the dropdown layout style - a simple list view with 1
item per line

genderSpinnerAdapter.setDropDownViewResource(android.R.layout.simple_dropdo
wn_item_1line);

        //Apply the adapter to the spinner
        mGenderSpinner.setAdapter(genderSpinnerAdapter);

        //Set the integer mSelected to the constant values
        mGenderSpinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view,
int position, long id) {
                String selection = (String)
parent.getItemAtPosition(position);
                if (!TextUtils.isEmpty(selection)) {
                    if (selection.equals(getString(R.string.gender_male)))
{
                        mGender = PetsEntry.GENDER_MALE; // Male
                    } else if
(selection.equals(getString(R.string.gender_female))) {
                        mGender = PetsEntry.GENDER_FEMALE; // Female
                    } else {
                        mGender = PetsEntry.GENDER_UNKNOWN; // Unknown
                    }
                }
            }

            //Because AdapterView is an abstract class, onNothingSelected
must be defined
```

```
            @Override
            public void onNothingSelected(AdapterView<?> parent) {
                mGender = 0; // Unknown
            }
        });
    }

    /**
     * Gets user input from the editor interface and save into the database
     */
    private void insertNewPet() {
        //Create an instance of the writable database
        SQLiteDatabase db = mDbHelper.getWritableDatabase();
        //Create the key:value pairs
        ContentValues values = new ContentValues();
        values.put(PetsEntry.COLUMN_PET_NAME,
mNameEditText.getText().toString().trim());
        values.put(PetsEntry.COLUMN_PET_BREED,
mBreedEditText.getText().toString().trim());
        values.put(PetsEntry.COLUMN_PET_GENDER, mGender);
        values.put(PetsEntry.COLUMN_PET_WEIGHT,
Integer.parseInt(mWeightEditText.getText().toString().trim()));

        //Insert the data into a new row, with the row ID being returned
        long newRowId = db.insert(PetsEntry.TABLE_NAME, null, values);

        //Notify the user of the success/or not of the query
        if (newRowId != -1) {
            Toast.makeText(getApplicationContext(), "Pet saved with ID: " +
newRowId, Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(getApplicationContext(), "Error with saving
pet", Toast.LENGTH_SHORT).show();
        }
    }

    /**
     * Override method that creates the menu options
     *
     * @param menu
     * @return
     */
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        //Inflate the menu options from the res/menu/menu_editor.xml file.
        //This adds menu items to the app bar.
        getMenuInflater().inflate(R.menu.menu_editor, menu);
        return true;
    }

    /**
     * Override method that handles the selection of the menu items
     *
     * @param item
     * @return
     */
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        //Switch statement for when a user clicks on an overflow menu item
in the app bar
        switch (item.getItemId()) {
```

```
            //Respond to a click on the "Save" menu option
            case R.id.action_save:
                //Insert a new pet into the database
                insertNewPet();
                //Return to the previous page
                finish();
                return true;
            //Respond to a click on the "Delete" menu option
            case R.id.action_delete:
                //Do nothing for now
                return true;
            //Respond to a click on the "Up" arrow button in the app bar
            case android.R.id.home:
                //Navigate back to parent activity (CatalogActivity)
                NavUtils.navigateUpFromSameTask(this);
                return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

PetDBHelper.java

```
package com.example.android.pets.data;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import com.example.android.pets.data.PetContract.PetsEntry;

/**
 * This is the schema (how the DB is created) for the database, where the
SQLite commands are created
 * It extends the SQLiteOpenHelper which helps to manage the database CRUD
commands
 */
public class PetDBHelper extends SQLiteOpenHelper {

    //Constants for the class, if the database schema is changed the
version must be updated
    private static final int DATABASE_VERSION = 1;
    private static final String DATABASE_NAME = "shelter.db";

    //Public constructor for the class
    public PetDBHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    /**
     * Override method that deals with the creation of the database, if it
isn't already created
     * @param db - the database being dealt with and returned
     */
    @Override
    public void onCreate(SQLiteDatabase db) {
        //SQL statement to CREATE the pets table, using the constants from
the Contract class
        String SQL_CREATE_PETS_TABLE = "CREATE TABLE " +
PetsEntry.TABLE_NAME + "("
                + PetsEntry._ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
```

```
                + PetsEntry.COLUMN_PET_NAME + " TEXT NOT NULL, "
                + PetsEntry.COLUMN_PET_BREED + " TEXT, "
                + PetsEntry.COLUMN_PET_GENDER + " INTEGER NOT NULL, "
                + PetsEntry.COLUMN_PET_WEIGHT + " INTEGER NOT NULL DEFAULT
0);";

        //Execute the create statement
        db.execSQL(SQL_CREATE_PETS_TABLE);
    }

    /**
     * Override method that deals with the upgrading (editing, dropping,
updating etc.) of tables
     * @param sqLiteDatabase - the DB being dealt with
     * @param oldVersion - the previous version of the DB
     * @param newVersion - the new version of the DB if it's changed
     */
    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int oldVersion,
int newVersion) {

    }
}
```

PetContract.java

```
package com.example.android.pets.data;

import android.net.Uri;
import android.provider.BaseColumns;

/**
 * The contract class holds all of the constants that will be used for the
SQLite DB throughout the other classes
 * These include table & column names, as well as other constants
 */
public final class PetContract {
    //Private constructor
    private PetContract() {
    }

    //The content authority that helps identify the content provider
    public static final String CONTENT_AUTHORITY =
"com.example.android.pets";

    //This is the base content Uri that will be shared with every Uri
associated with PetContract
    public static final Uri BASE_CONTENT_URI = Uri.parse("content://" +
CONTENT_AUTHORITY);

    //This is the table name that will be appended to the base content Uri
    public static final String PATH_PETS = "pets";

    /**
     * Inner class that defines the table contents
     */
    public static class PetsEntry implements BaseColumns {
        //Constants for table name, headings & pet genders
        public static final String TABLE_NAME = "pets";
        public static final String _ID = BaseColumns._ID;
        public static final String COLUMN_PET_NAME = "name";
```

```
        public static final String COLUMN_PET_BREED = "breed";
        public static final String COLUMN_PET_GENDER = "gender";
        public static final String COLUMN_PET_WEIGHT = "weight";
        public static final int GENDER_UNKNOWN = 0;
        public static final int GENDER_MALE = 1;
        public static final int GENDER_FEMALE = 2;

        //The full Uri using the constants declared above
        public static final Uri CONTENT_URI =
Uri.withAppendedPath(BASE_CONTENT_URI, PATH_PETS);
    }
}
```

PetProvider.java

```
package com.example.android.pets.data;

import android.content.ContentProvider;
import android.content.ContentValues;
import android.content.UriMatcher;
import android.database.Cursor;
import android.net.Uri;

/**
 * This class is the Content Provider class for the app
 */
public class PetProvider extends ContentProvider {

    //Log tag
    public static final String LOG_TAG = PetProvider.class.getSimpleName();

    //Class-wide instance variable of the DB helper class
    private PetDBHelper mDbHelper;

    //Global variables for match case integers for Uri matcher
    private static final int PETS = 100;
    private static final int PET_ID = 101;

    //Global variable for the Uri matcher
    private static final UriMatcher sUriMatcher = new
UriMatcher(UriMatcher.NO_MATCH);

    /**
     * The calls to addURI() go here, for all of the content URI patterns
that the provider should recognize
     */
    static {
        //Adds the URI match for the whole pets table
        sUriMatcher.addURI(PetContract.CONTENT_AUTHORITY,
PetContract.PATH_PETS, PETS);

        //Adds the URI match for a single row of the pets table
        // # is used as a wildcard to support any integer for the specified
row number
        sUriMatcher.addURI(PetContract.CONTENT_AUTHORITY,
PetContract.PATH_PETS + "/#", PET_ID);
    }

    /**
     * Initialise the provider and the database helper object.
     */
```

```
    @Override
    public boolean onCreate() {
        //Create and initialise a PetDBHelper object to gain access to the
pets database
        mDbHelper = new PetDBHelper(getContext());
        return true;
    }

    /**
     * Perform the query for the given URI. Use the given projection,
selection, selection arguments, and sort order
     */
    @Override
    public Cursor query(Uri uri, String[] projection, String selection,
String[] selectionArgs, String sortOrder) {
        return null;
    }

    /**
     * Insert new data into the provider with the given ContentValues
     */
    @Override
    public Uri insert(Uri uri, ContentValues contentValues) {
        return null;
    }

    /**
     * Updates the data at the given selection and selection arguments,
with the new ContentValues
     */
    @Override
    public int update(Uri uri, ContentValues contentValues, String
selection, String[] selectionArgs) {
        return 0;
    }

    /**
     * Delete the data at the given selection and selection arguments
     */
    @Override public int delete(Uri uri, String selection, String[]
selectionArgs) {
        return 0;
    }

    /**
     * Returns the MIME type of data for the content URI
     */
    @Override
    public String getType(Uri uri) {
        return null;
    }
}
```

Appendix 13 – Conversion Prototype app code

https://goo.gl/BpU5uC

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.android.conversionprototype.MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginLeft="16dp"
        android:layout_marginRight="16dp"
        android:layout_marginTop="8dp"
        android:orientation="vertical">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:gravity="center_vertical"
                android:text="@string/heading_baseCurrency" />

            <EditText
                android:id="@+id/amountToConvertEntryField"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginLeft="8dp"
                android:hint="@string/prompt_amountEntry"
                android:inputType="numberDecimal" />
        </LinearLayout>

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="16dp"
            android:text="@string/heading_currencyToConvert" />

        <FrameLayout
            android:id="@+id/spinnerBox"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:background="@drawable/bg_spinner">

            <Spinner
                android:id="@+id/currency_picker"
                android:layout_width="match_parent"
                android:layout_height="wrap_content" />
        </FrameLayout>

        <Button
```

```
            android:id="@+id/button_convert"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="8dp"
            android:text="@string/button_convert" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="16dp"
            android:text="@string/heading_amountConverted" />

        <TextView
            android:id="@+id/conversionResultField"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:text="@string/symbol_GBP" />
    </LinearLayout>

    <Button
        android:id="@+id/button_saveConversion"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerInParent="true"
        android:layout_marginBottom="16dp"
        android:text="@string/button_saveConversion" />
</RelativeLayout>
```

activity_second.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ListView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/root_list_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.example.android.conversionprototype.SecondActivity"
/>
```

activity_third.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.example.android.conversionprototype.ThirdActivity">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginLeft="16dp"
        android:layout_marginRight="16dp"
        android:layout_marginTop="8dp"
        android:orientation="vertical">
```

```
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/heading_aboutApp" />

        <!--TODO: Fill out content based on heading-->
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:text="@string/content_aboutApp" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="16dp"
            android:text="@string/heading_useOfApp" />

        <!--TODO: Fill out content based on heading-->
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:text="@string/content_directionsOfUse" />

    </LinearLayout>

</ScrollView>
```

custom_list_view.xml

```
<?xml version="1.0" encoding="utf-8"?><!--Use this XML layout to define
what each element of the list should look like.--><!--Use
'xmlns:tools="http://schemas.android.com/tools"' in the parent view--><!--
so that 'tools:text' can be used to show sample text-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:paddingBottom="8dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="8dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/baseCurrencyValue"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            tools:text="£2" />

        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_marginLeft="8dp"
            android:src="@drawable/ic_arrow_forward_black_18dp" />
```

```
        <TextView
            android:id="@+id/conversionCurrencyValue"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_marginLeft="8dp"
            tools:text="€2,40" />
    </LinearLayout>

    <TextView
        android:id="@+id/date"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        tools:text="12 Apr 2016" />

    <!--TODO: Formatting for primary & secondary text-->
    <!--Primary text formatting-->
    <!--android:textAppearance="?android:attr/textAppearanceListItemSmall"-
->
    <!--android:textColor="@color/primaryText"-->

    <!--Secondary text formatting-->
    <!--
android:textAppearance="?android:attr/textAppearanceListItemSecondary"-->
    <!--android:textColor="@color/secondaryText"-->

</LinearLayout>
```

spinner_dropdown_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--This XML layout file controls the styling for the text inside the
dropdown menu-->
<!--Use a CheckedTextView for this XML file-->
<!--Make sure that attributes 'id, style, ellipsize, maxLines' are all
included-->
<!--Everything else can be customised-->
<!--48dp is the recommended height for a button/selection-->
<CheckedTextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/text1"
    style="?android:attr/spinnerDropDownItemStyle"
    android:layout_width="match_parent"
    android:layout_height="48dp"
    android:ellipsize="marquee"
    android:maxLines="1" />
```

spinner_item.xml

```
<!--This XML layout file controls the styling for the text once selected
from the spinner--><!--Using a TextView works fine for this XML file--><!--
Make sure that attribute 'id' is included--><!--Everything else can be
customised--><!--Keep height to 'wrap_content' to keep the height defined
by the background XML-->
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/text1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="14sp" />
```

MainActivity.java

```
package com.example.android.conversionprototype;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

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

        //Set the title of the title bar to be more appropriate
        setTitle(getResources().getText(R.string.page_1));

        //Call this method to initiate the creation of the Spinner
        createSpinnerOptions();

        //Get the spinner as an object and call the OnItemSelectedListener
on it to create the listener
        Spinner spinner = (Spinner) findViewById(R.id.currency_picker);
        spinner.setOnItemSelectedListener(new MyOnItemSelectedListener());

        //Create a new button object constructor
        Button button = (Button) findViewById(R.id.button_convert);
        //Set a listener to it and create a new method
        button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                //Do this when the button is clicked
//                Log.i("TEST MESSAGE:", "Convert button click was
registered");
                onConvertClick();
            }
        });
    }

    /**
     * Custom method that handles what happens when the convert button is
clicked
     */
    private void onConvertClick() {
        //Get the value of the current spinner option
//        Log.i("TEST MESSAGE:", "onConvertClick was called");
        String currentSpinnerOption = getSpinnerOption();
        //Get the relevant symbol for the currency
        String relevantCurrencySymbol =
getCurrencySymbol(currentSpinnerOption);

        //Change the text to display the correct values
```

```
            setResultFieldText(relevantCurrencySymbol);
        }

        /**
         * Custom method
         *
         * @return the currently selected option of the spinner
         */
        private String getSpinnerOption() {
//          Log.i("TEST MESSAGE:", "getSpinnerOption was called");
            //Create a spinner object constructor
            Spinner spinner = (Spinner) findViewById(R.id.currency_picker);
            //Get the current item that is selected in the spinner
//          Log.i("TEST MESSAGE:", "Value to return: " +
spinner.getSelectedItem().toString());
            return spinner.getSelectedItem().toString();
        }

        /**
         * Custom method
         *
         * @param country input of the country
         * @return the relevant currency symbol associated with the country
         */
        private String getCurrencySymbol(String country) {
            //Use switch statement to determine the correct symbol for the
            // currency and return it
            switch (country) {
                //Euros
                case "European Euro":
                    return getResources().getString(R.string.symbol_EUR);

                //US Dollars
                case "US Dollar":
                    return getResources().getString(R.string.symbol_USD);

                //Japanese Yen
                case "Japanese Yen":
                    return getResources().getString(R.string.symbol_JPY);

                default:
                    return getResources().getString(R.string.symbol_GBP);
            }
        }

        /**
         * Custom method that outputs the result from the conversion
         *
         * @param relevantCurrencySymbol input of the relevant currency symbol
to use
         */
        private void setResultFieldText(String relevantCurrencySymbol) {
            //Create an object constructor for the text-field that is changed
            TextView conversionResultField = (TextView)
findViewById(R.id.conversionResultField);

            //Create an object constructor for the EditText
            EditText amountToConvertEntryField = (EditText)
findViewById(R.id.amountToConvertEntryField);
            //Get the values from the EditText
```

```
        String userEnteredQuantity =
amountToConvertEntryField.getText().toString();

        //TODO: implement proper calculations into where the text-field is
changed
        //Set the text-field text
        conversionResultField.setText(relevantCurrencySymbol +
userEnteredQuantity);
    }

    /**
     * Create the options menu in the action bar
     * using {@param menu} as a resource
     */
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.main_menu, menu);
        return true;
    }

    /**
     * Handle item selections of the action bar
     */
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        //Switch statement
        switch (item.getItemId()) {
            //When the list icon is selected:
            case R.id.action_conversionNav:
                //Open the second page
                changeToSavedConversions();
                return true;

            //When the refresh icon is selected:
            case R.id.action_refresh:
                //Refresh the app
                refreshApp();
                return true;

            //When the help icon is selected:
            case R.id.action_help:
                //Open the help page
                changeToHelpPage();
                return true;

            default:
                // The user's action was not recognized. Invoke the
superclass to handle it.
                return super.onOptionsItemSelected(item);
        }
    }

    /**
     * Method that navigates the user to the second page, using an intent
     */
    private void changeToSavedConversions() {
        //Create a new Intent object constructor, populate it with
SecondActivity.class
        Intent intent = new Intent(this, SecondActivity.class);
        //Start that new intent
```

```
        startActivity(intent);
    }

    /**
     * Method that navigates the user to the help page, using an intent
     */
    private void changeToHelpPage() {
        //Create a new Intent object constructor, populate it with
ThirdActivity.class
        Intent intent = new Intent(this, ThirdActivity.class);
        //Start that new intent
        startActivity(intent);
    }

    /**
     * Custom method, refresh the conversion rates and reset the changed
fields
     */
    private void refreshApp() {
        //Create an object constructor for the EditText
        EditText amountToConvertEntryField = (EditText)
findViewById(R.id.amountToConvertEntryField);
        //Set the field to be empty
        amountToConvertEntryField.setText(null);

        //Create a spinner object constructor
        Spinner spinner = (Spinner) findViewById(R.id.currency_picker);
        //Reset the value to Euro (default)
        spinner.setSelection(0);

        //Reset the values in the results field
        setResultFieldText(getResources().getString(R.string.symbol_EUR));

        //TODO: Implement refreshing of conversion rates
    }

    /**
     * The method that creates the spinner options
     */
    public void createSpinnerOptions() {
        //Create the spinner as an object
        Spinner spinner = (Spinner) findViewById(R.id.currency_picker);

        // Create an ArrayAdapter using the the array of options and the
style of the text before selection layout
        ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(this, R.array.array_currencies,
R.layout.spinner_item);

        // Specify the layout to use when the list of choices appears
        adapter.setDropDownViewResource(R.layout.spinner_dropdown_item);

        // Apply the adapter to the spinner
        spinner.setAdapter(adapter);
    }

    /**
     * This sub-class controls what happens when the listener detects
something has been selected
     */
```

```
    private class MyOnItemSelectedListener implements
AdapterView.OnItemSelectedListener {
        public void onItemSelected(AdapterView<?> parent, View view, int
pos, long id) {
            //When an item is selected, do this:
            onConvertClick();
        }

        public void onNothingSelected(AdapterView parent) {
            // Do nothing when nothing is selected.
        }
    }
}
```

SecondActivity.java

```
package com.example.android.conversionprototype;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.widget.ListView;

import java.util.ArrayList;
import java.util.List;

public class SecondActivity extends AppCompatActivity {

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

        //Create the ArrayList, so it can be used with objects
        List<SavedCurrencies> savedCurrencies = new ArrayList<>();
        //Populate the ArrayList
        savedCurrencies.add(new SavedCurrencies("£1", "$1.20", "04 Sep
2015"));
        savedCurrencies.add(new SavedCurrencies("£12", "€15,30", "21 Aug
2016"));
        savedCurrencies.add(new SavedCurrencies("£510", "¥1683", "14 Jan
2017"));

        //Creates an adapter for the values to use, appends the array of
values to the adapter,
        //the adapter is responsible for making a View for each item in the
data set
        SavedCurrenciesAdapter adapter = new SavedCurrenciesAdapter(this,
savedCurrencies);
        //Creates a object constructor for the ListView
        ListView listView = (ListView) findViewById(R.id.root_list_view);
        //Sets the adapter method on the ListView
        listView.setAdapter(adapter);
    }
}
```

ThirdActivity.java

```
package com.example.android.conversionprototype;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
```

```
public class ThirdActivity extends AppCompatActivity {

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

SavedCurrencies.java

```
package com.example.android.conversionprototype;

public class SavedCurrencies {

    /**
     * Declare private variables for this class to use
     */
    private String mBaseCurrencyValue;
    private String mConversionCurrencyValue;
    private String mDate;

    /**
     * Create the constructor for this class, a constructor creates an
instance of a class
     * This constructor will create an instance of two Strings
     */
    public SavedCurrencies(String baseCurrencyValue, String
conversionCurrencyValue, String date) {
        mBaseCurrencyValue = baseCurrencyValue;
        mConversionCurrencyValue = conversionCurrencyValue;
        mDate = date;
    }

    /**
     * Custom method
     *
     * @return the user-entered value to convert
     */
    public String getBaseCurrencyValue() {
        return mBaseCurrencyValue;
    }

    /**
     * Custom method
     *
     * @return the converted value
     */
    public String getConversionCurrencyValue() {
        return mConversionCurrencyValue;
    }

    /**
     * Custom method
     *
     * @return the date
     */
    public String getDate() {
        return mDate;
    }
```

```
}
```

SavedCurrenciesAdapter.java

```
package com.example.android.conversionprototype;

import android.app.Activity;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

import java.util.List;

public class SavedCurrenciesAdapter extends ArrayAdapter<SavedCurrencies> {

    /**
     * A custom constructor
     *
     * @param context         is used to inflate the layout file
     * @param savedCurrencies is the data we want to populate into the
lists
     */
    public SavedCurrenciesAdapter(Activity context, List<SavedCurrencies>
savedCurrencies) {
        // This initialises the ArrayAdapter's internal storage for the
context and the list.
        // The second argument is used when the ArrayAdapter is populating
a single TextView.
        // Because this is a custom adapter for three TextViews the adapter
is not
        // going to use this second argument, so it can be any value.
        super(context, 0, savedCurrencies);
    }

    /**
     * Provides a view for an AdapterView (ListView, GridView, etc.)
     *
     * @param position    is the position in the list of data that should
be displayed in the list item view.
     * @param convertView The recycled view to populate.
     * @param parent      The parent ViewGroup that is used for inflation.
     * @return The View for the position in the AdapterView.
     */
    @NonNull
    @Override
    public View getView(int position, @Nullable View convertView, @NonNull
ViewGroup parent) {
        //Get the data item for this position
        SavedCurrencies savedCurrencies = getItem(position);

        //Check if an existing view is being reused, otherwise inflate the
view
        if (convertView == null) {
            convertView =
LayoutInflater.from(getContext()).inflate(R.layout.custom_list_view,
parent, false);
        }
```

```
        //Lookup view for data population
        TextView baseCurrencyValueField = (TextView)
convertView.findViewById(R.id.baseCurrencyValue);
        TextView conversionCurrencyValueField = (TextView)
convertView.findViewById(R.id.conversionCurrencyValue);
        TextView dateField = (TextView)
convertView.findViewById(R.id.date);

        //Populate the data into the template view using the data object

baseCurrencyValueField.setText(savedCurrencies.getBaseCurrencyValue());

conversionCurrencyValueField.setText(savedCurrencies.getConversionCurrencyV
alue());
        dateField.setText(savedCurrencies.getDate());

        //Return the completed view to render on-screen
        return convertView;
    }
}
```

Appendix 14 – Currency API Test app code

https://goo.gl/numD4G

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.android.currencyapitest.MainActivity">

    <ListView
        android:id="@+id/list"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" />

    <!-- Empty view is only visible when the list has no items. -->
    <TextView
        android:id="@+id/empty_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:textAppearance="?android:textAppearanceMedium" />

    <!--Progress bar to let the user know that data is being fetched-->
    <ProgressBar
        android:id="@+id/loading_spinner"
        style="@style/Widget.AppCompat.ProgressBar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true" />

</RelativeLayout>
```

element_layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="8dp"
    android:orientation="vertical">

    <TextView
        android:id="@+id/dateView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        tools:text="26/10/13" />

    <TextView
        android:id="@+id/baseCurrencyView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        tools:text="EUR" />

    <TextView
        android:id="@+id/exchangeCurrencyView"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        tools:text="GBP" />

    <TextView
        android:id="@+id/exchangeCurrencyRateView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        tools:text="1.2345" />
</LinearLayout>
```

MainActivity.java

```
package com.example.android.currencyapitest;

import android.app.LoaderManager;
import android.content.Context;
import android.content.Loader;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.ListView;
import android.widget.ProgressBar;
import android.widget.TextView;

import java.util.ArrayList;
import java.util.List;

public class MainActivity extends AppCompatActivity implements
LoaderManager.LoaderCallbacks<List<Currency>> {

    //URL that retrieves the JSON response from the API
    public static final String REQUEST_URL =
"https://api.fixer.io/latest?base=GBP";

    //Constant value for the currency loader ID. This is only really used
if you're using multiple loaders.
    private static final int CURRENCY_LOADER_ID = 1;

    //Global instance of the CurrencyAdapter, so it can be used in multiple
methods in this class
    private CurrencyAdapter mAdapter;

    //Global instance of the TextView that is displayed when the list is
empty
    private TextView mEmptyStateTextView;

    //Global instance of the ProgressBar, so it can be used in multiple
methods in this class
    private ProgressBar mProgressBar;

    /**
     * OnCreate method
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //Check if the user is connected to the Internet
```

```
        if (!checkInternetConnectivity()) {
            //Hide the loading spinner
            mProgressBar = (ProgressBar)
findViewById(R.id.loading_spinner);
            mProgressBar.setVisibility(View.GONE);

            //Let the user know there is no Internet connection
            mEmptyStateTextView = (TextView) findViewById(R.id.empty_view);
            mEmptyStateTextView.setText(R.string.no_connectivity);
        } else {
            //Create an object constructor for the ListView
            ListView currencyListView = (ListView) findViewById(R.id.list);

            //Find and set the empty text view as the empty view for the
layout
            mEmptyStateTextView = (TextView) findViewById(R.id.empty_view);
            currencyListView.setEmptyView(mEmptyStateTextView);

            //Creates an adapter for the words to use, appends the array of
words to the adapter,
            //the adapter is responsible for making a View for each item in
the data set
            mAdapter = new CurrencyAdapter(this, new
ArrayList<Currency>());

            //Sets the adapter method on the ListView
            //so the list can be populated in the user interface
            currencyListView.setAdapter(mAdapter);

            //Get a reference to the LoaderManager, in order to interact
with loaders.
            LoaderManager loaderManager = getLoaderManager();

            //Initialize the loader. Pass in the int ID constant defined
above and pass in null for
            //the bundle. Pass in this activity for the LoaderCallbacks
parameter (which is valid
            //because this activity implements the LoaderCallbacks
interface).
            loaderManager.initLoader(CURRENCY_LOADER_ID, null, this);
        }
    }

    /**
     * Method that checks if the user is connected to the Internet
     *
     * @return true or false
     */
    private boolean checkInternetConnectivity() {
        //Create a connectivity manager
        ConnectivityManager cm = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);

        //Get the active network's network info, and return a boolean value
        NetworkInfo activeNetwork = cm.getActiveNetworkInfo();
        return activeNetwork != null &&
activeNetwork.isConnectedOrConnecting();
    }

    /**
     * Method for the creation of the loader, pass in:
```

```
     * The Loader {@param id} to be used
     * The {@param args} (arguments) the Loader should take
     *
     * @return a new instance of the Loader
     */
    @Override
    public Loader<List<Currency>> onCreateLoader(int i, Bundle bundle) {
        return new CurrencyLoader(this);
    }

    /**
     * Once the loader has finished, execute this method with:
     * The {@param loader} to be used
     * The {@param result} from the Loader creation
     */
    @Override
    public void onLoadFinished(Loader<List<Currency>> loader,
List<Currency> currencies) {
        //Hide the loading spinner
        mProgressBar = (ProgressBar) findViewById(R.id.loading_spinner);
        mProgressBar.setVisibility(View.GONE);

        //Set the empty state text to display "No exchange rates found."
        mEmptyStateTextView.setText(R.string.no_dataResults);

        //Clear the adapter of previous results
        mAdapter.clear();

        //If there is a valid list of Currency's, then add them to the
adapter's dataset
        //This triggers the ListView to update
        if (currencies != null && !currencies.isEmpty()) {
            mAdapter.addAll(currencies);
        }
    }

    /**
     * If the Loader is reset (i.e. through orientation change), handle
that in this method
     * The {@param loader} to be used
     */
    @Override
    public void onLoaderReset(Loader<List<Currency>> loader) {
        //Loader reset, so we can clear out our existing data.
        mAdapter.clear();
    }
}
```

Currency.java

```
package com.example.android.currencyapitest;

/**
 * The Currency class that creates and provides methods that return parts
of the ArrayList when called
 * by other parts of the app
 */
class Currency {

    //Declare private variables
    private String mDate;
```

```
    private String mBase;
    private String mCurrencyKey;
    private double mCurrencyValue;

    /**
     * Create the constructor for this class, a constructor creates an
instance of a class
     * This constructor will create an instance of two Strings and a double
     *
     * @param date
     * @param base
     * @param currencyKey
     * @param currencyValue
     */
    public Currency(String date, String base, String currencyKey, double
currencyValue) {
        mDate = date;
        mBase = base;
        mCurrencyKey = currencyKey;
        mCurrencyValue = currencyValue;
    }

    /**
     * @return the date that is associated with the results
     */
    public String getDateOfExchange() {
        return mDate;
    }

    /**
     * @return the base currency
     */
    public String getBaseCurrency() {
        return mBase;
    }

    /**
     * @return the currency that the conversion rate applies to
     */
    public String getConvertedCurrency() {
        return mCurrencyKey;
    }

    /**
     * @return the exchange rate that the country is providing
     */
    public double getExchangeRate() {
        return mCurrencyValue;
    }
}
```

CurrencyAdapter.java

```
package com.example.android.currencyapitest;

import android.app.Activity;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
```

```
import android.widget.ArrayAdapter;
import android.widget.TextView;

import java.util.List;

class CurrencyAdapter extends ArrayAdapter<Currency> {

    /**
     * A custom constructor.
     *
     * @param context    is used to inflate the layout file
     * @param currencies is the data we want to populate into the lists
     */
    public CurrencyAdapter(Activity context, List<Currency> currencies) {
        // This initialises the ArrayAdapter's internal storage for the
context and the list.
        // The second argument is used when the ArrayAdapter is populating
a single TextView.
        // Because this is a custom adapter for four TextViews the adapter
is not
        // going to use this second argument, so it can be any value.
        super(context, 0, currencies);
    }

    @NonNull
    @Override
    public View getView(int position, @Nullable View convertView, @NonNull
ViewGroup parent) {
        //Get the data item for this position
        Currency currentExchange = getItem(position);

        //Check if an existing view is being reused, otherwise inflate the
view
        if (convertView == null) {
            convertView =
LayoutInflater.from(getContext()).inflate(R.layout.element_layout, parent,
false);
        }

        //Lookup the views for the data population
        TextView dateView = convertView.findViewById(R.id.dateView);
        TextView baseCurrencyView =
convertView.findViewById(R.id.baseCurrencyView);
        TextView exchangeCurrencyView =
convertView.findViewById(R.id.exchangeCurrencyView);
        TextView exchangeCurrencyRateView =
convertView.findViewById(R.id.exchangeCurrencyRateView);

        //Populate data into the template view using the data object
        dateView.setText(currentExchange.getDateOfExchange());
        baseCurrencyView.setText(currentExchange.getBaseCurrency());

exchangeCurrencyView.setText(currentExchange.getConvertedCurrency());
        exchangeCurrencyRateView.setText("" +
currentExchange.getExchangeRate());

        //Return the completed view to render on-screen
        return convertView;
    }
}
```

CurrencyLoader.java

```
package com.example.android.currencyapitest;

import android.content.AsyncTaskLoader;
import android.content.Context;

import java.util.List;

/**
 * Currency loader task
 */
class CurrencyLoader extends AsyncTaskLoader<List<Currency>> {

    //Log tag that returns the package name for errors
    public static final String LOG_TAG =
CurrencyLoader.class.getSimpleName();

    //Global instance of the String URL, so it can be used in multiple
methods in this class
    private String mUrl;

    /**
     * The constructor for this class
     */
    public CurrencyLoader(Context context) {
        super(context);
        mUrl = MainActivity.REQUEST_URL;
    }

    /**
     * When the loader is instructed to load
     */
    @Override
    protected void onStartLoading() {
        //Force the load
        forceLoad();
    }

    /**
     * The background process for the loader
     */
    @Override
    public List<Currency> loadInBackground() {
        //Don't perform the request is there are no URLs, or the URL is
null
        if (mUrl == null) {
            return null;
        }

        //Get the data for the URL provided & Return the result
        return QueryUtils.fetchCurrencyData(mUrl);
    }
}
```

QueryUtils.java

```
package com.example.android.currencyapitest;

import android.text.TextUtils;
import android.util.Log;
```

```
import org.json.JSONException;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.nio.charset.Charset;
import java.util.ArrayList;
import java.util.List;

/**
 * Helper methods related to requesting and receiving currency data from
Fixer.IO
 */
class QueryUtils {

    //Tag for the log messages
    private static final String LOG_TAG = QueryUtils.class.getSimpleName();

    /**
     * Query Fixer.IO and return an {@link List<Currency>} object to
represent a single exchange rate.
     */
    public static List<Currency> fetchCurrencyData(String requestUrl) {
        //Create a URL object
        URL url = createUrl(requestUrl);

        //Perform a HTTP request to the URL and receive a JSON response
back
        //Initialise variable
        String jsonResponse = null;

        //Try the creation of a JSON response from the URL, if not catch
the exception
        try {
            jsonResponse = makeHttpRequest(url);
        } catch (IOException e) {
            //Log the error
            Log.e(LOG_TAG, "Error closing input stream", e);
        }

        //Extract and return the relevant fields from the JSON response and
create a List<Currency> object
        return extractFeaturesFromJson(jsonResponse);
    }

    /**
     * @return a new URL object from the given string URL.
     */
    private static URL createUrl(String stringUrl) {
        //Initialise variable
        URL url = null;

        //Try the creation of a URL, if not catch the exception
        try {
            url = new URL(stringUrl);
```

```
        } catch (MalformedURLException e) {
            //Log the error
            Log.e(LOG_TAG, "Error with creating URL", e);
        }
        //Return the successful URL
        return url;
    }

    /**
     * Make an HTTP request to the given URL and return a String as the
response.
     *
     * @throws IOException if something goes wrong
     */
    private static String makeHttpRequest(URL url) throws IOException {
        //Initialise variable
        String jsonResponse = "";

        //If the URL is null, then return early with an empty string
        if (url == null) {
            return jsonResponse;
        }

        //Initialise variables
        HttpURLConnection urlConnection = null;
        InputStream inputStream = null;

        //Try connecting to the URL and reading the response, else catch
the exception
        try {
            //Open the connection to the URL
            urlConnection = (HttpURLConnection) url.openConnection();

            //Set timeouts so if the connections expire before data is
available, an error is thrown
            //Makes ure the user is never waiting for long periods of time
if there is no data
            urlConnection.setReadTimeout(10000 /* milliseconds */);
            urlConnection.setConnectTimeout(15000 /* milliseconds */);

            //Set the request method, as data wants to be retrieved
            urlConnection.setRequestMethod("GET");

            //Make the connection
            urlConnection.connect();

            //If the request was successful (response code 200), then read
the input and parse the response
            if (urlConnection.getResponseCode() == 200) {
                inputStream = urlConnection.getInputStream();
                jsonResponse = readFromStream(inputStream);
            } else {
                //Log the error
                Log.e(LOG_TAG, "Error response code: " +
urlConnection.getResponseCode());
            }
        } catch (IOException e) {
            //Log the error
            Log.e(LOG_TAG, "Problem retrieving the JSON results", e);
        } finally {
            //As long as there is an open connection
```

```
            if (urlConnection != null) {
                //Close the URL connection
                urlConnection.disconnect();
            }
            //As long as the inputStream is open
            if (inputStream != null) {
                //Close the inputStream
                inputStream.close();
            }
        }
        //Return the JSON response
        return jsonResponse;
    }

    /**
     * Convert the {@link InputStream} into a String which contains the
     * whole JSON response from the server.
     */
    private static String readFromStream(InputStream inputStream) throws
IOException {
        //Initialise variables / create a new String Builder (S.B.)
        //S.B. builds one String bit by bit instead of appending more
information each time through other variables
        StringBuilder output = new StringBuilder();

        //As long as the inputStream is not empty
        if (inputStream != null) {
            //Create a new InputStreamReader, using the inputStream and
"UTF-8" character set
            InputStreamReader inputStreamReader = new
InputStreamReader(inputStream, Charset.forName("UTF-8"));

            //Create a buffered reader, which stores information about the
characters around each character
            BufferedReader reader = new BufferedReader(inputStreamReader);

            //Get the line of text
            String line = reader.readLine();

            //While the line isn't empty
            while (line != null) {
                //Append the line to the output
                output.append(line);
                //Goto the next line
                line = reader.readLine();
            }
        }
        //Return the completed output in String format
        return output.toString();
    }

    /**
     * @return an {@link List<Currency>} item by parsing out information
     * about the exchange rates from the input jsonResponse string.
     */
    private static List<Currency> extractFeaturesFromJson(String
jsonResponse) {
        //If the JSON String is empty or null, then return early
        if (TextUtils.isEmpty(jsonResponse)) {
            return null;
        }
```

```
        //Create an empty ArrayList that we can start adding exchange rates
to
        List<Currency> currencies = new ArrayList<>();

        //Try to parse the jsonResponse, else catch the JSONException
        try {
            //Get the root of the JSON string
            JSONObject root = new JSONObject(jsonResponse);

            //Get the base currency
            String baseCurrency = root.getString("base");
            Log.i("base currency", baseCurrency);

            //Get the date of the exchange rate
            String exchangeDate = root.getString("date");
            Log.i("exchange date", exchangeDate);

            //Get the "rates" array as an object
            JSONObject rates = root.getJSONObject("rates");

            //For loop that cycles through all of the rates objects
            for (int i = 0; i < rates.length(); i++) {
                //Get the object key (country code)
                String objectKey = rates.names().getString(i);

                //Get the exchange rate value associated with that object
                double keyValue = rates.getDouble(objectKey);

                //Add those details along with the above date and base to
the array
                currencies.add(new Currency(exchangeDate, baseCurrency,
objectKey, keyValue));
            }
        } catch (JSONException e) {
            //Log the error
            Log.e(LOG_TAG, "Problems parsing the currency JSON results",
e);
        }
        return currencies;
    }
}
```

Appendix 15 – Logcat output for DFD

```
01-12 17:14:33.108 ? I/art: Late-enabling -Xcheck:jni
01-12 17:14:33.157 com.zacmurphy.currencyconverter V/Theme:
ColorMode:com.zacmurphy.currencyconverter is false
01-12 17:14:33.162 com.zacmurphy.currencyconverter W/System: ClassLoader
referenced unknown path: /data/app/com.zacmurphy.currencyconverter-
1/lib/arm64
01-12 17:14:33.177 com.zacmurphy.currencyconverter D/LoadingActivity:
LoadingActivity - started
01-12 17:14:33.185 com.zacmurphy.currencyconverter W/art: Before Android
4.1, method android.graphics.PorterDuffColorFilter
android.support.graphics.drawable.VectorDrawableCompat.updateTintFilter(and
roid.graphics.PorterDuffColorFilter, android.content.res.ColorStateList,
android.graphics.PorterDuff$Mode) would have incorrectly overridden the
package-private method in android.graphics.drawable.Drawable
01-12 17:14:33.219 com.zacmurphy.currencyconverter D/LoadingActivity:
hideSystemUI - called
01-12 17:14:33.219 com.zacmurphy.currencyconverter D/LoadingActivity:
isInternetConnected() - called
01-12 17:14:33.220 com.zacmurphy.currencyconverter V/LoadingActivity:
Internet connection established
01-12 17:14:33.221 com.zacmurphy.currencyconverter D/LoadingActivity:
onCreateLoader() - called
01-12 17:14:33.222 com.zacmurphy.currencyconverter D/CurrencyLoader:
constructor - called
01-12 17:14:33.223 com.zacmurphy.currencyconverter D/CurrencyLoader:
onStartLoading - called
01-12 17:14:33.224 com.zacmurphy.currencyconverter D/CurrencyLoader:
loadInBackground - called
01-12 17:14:33.224 com.zacmurphy.currencyconverter D/QueryUtils:
fetchCurrencyData - called
01-12 17:14:33.224 com.zacmurphy.currencyconverter D/QueryUtils: createUrl
- called
01-12 17:14:33.225 com.zacmurphy.currencyconverter E/AbstractTracker: Can't
create handler inside thread that has not called Looper.prepare()
01-12 17:14:33.225 com.zacmurphy.currencyconverter D/AppTracker: App Event:
start
01-12 17:14:33.227 com.zacmurphy.currencyconverter V/QueryUtils: url:
https://api.fixer.io/latest?base=GBP
01-12 17:14:33.227 com.zacmurphy.currencyconverter V/QueryUtils: url:
https://api.fixer.io/latest?base=GBP
01-12 17:14:33.227 com.zacmurphy.currencyconverter D/QueryUtils:
makeHttpRequest - called
01-12 17:14:33.230 com.zacmurphy.currencyconverter I/DpmTcmClient:
RegisterTcmMonitor from: com.android.okhttp.TcmIdleTimerMonitor
01-12 17:14:33.233 com.zacmurphy.currencyconverter D/QueryUtils:
urlConnection opened
01-12 17:14:33.233 com.zacmurphy.currencyconverter D/QueryUtils: timeouts
set
01-12 17:14:33.233 com.zacmurphy.currencyconverter D/QueryUtils: request
method set
01-12 17:14:33.237 com.zacmurphy.currencyconverter D/OpenGLRenderer: Use
EGL_SWAP_BEHAVIOR_PRESERVED: true
01-12 17:14:33.264 com.zacmurphy.currencyconverter I/Adreno: QUALCOMM build
: 183c040, Iff84fb1103
                                                    Build Date
: 03/18/16
                                                    OpenGL ES
Shader Compiler Version: XE031.06.00.02
```

```
                                                                Local Branch
:
                                                                Remote Branch
: refs/tags/AU_LINUX_ANDROID_LA.BF64.1.2.2_RB4.06.00.01.180.031
                                                                Remote Branch
: NONE
                                                                Reconstruct
Branch                : NOTHING
01-12 17:14:33.267 com.zacmurphy.currencyconverter I/OpenGLRenderer:
Initialized EGL, version 1.4
01-12 17:15:05.092 com.zacmurphy.currencyconverter D/QueryUtils: url
connection connecting
01-12 17:15:08.477 com.zacmurphy.currencyconverter V/QueryUtils: Response
code was 200
01-12 17:15:08.477 com.zacmurphy.currencyconverter D/QueryUtils:
readFromStream - called
01-12 17:15:08.477 com.zacmurphy.currencyconverter V/QueryUtils:
inputStream is not null
01-12 17:15:08.479 com.zacmurphy.currencyconverter V/QueryUtils:
jsonResponse: {"base":"GBP","date":"2018-01-
12","rates":{"AUD":1.7326,"BGN":2.1979,"BRL":4.3783,"CAD":1.7075,"CHF":1.32
46,"CNY":8.8126,"CZK":28.681,"DKK":8.3709,"HKD":10.672,"HRK":8.3672,"HUF":3
46.97,"IDR":18183.0,"ILS":4.6382,"INR":86.749,"JPY":151.58,"KRW":1449.2,"MX
N":26.068,"MYR":5.4229,"NOK":10.855,"NZD":1.8797,"PHP":68.714,"PLN":4.6888,
"RON":5.2103,"RUB":77.214,"SEK":11.053,"SGD":1.8097,"THB":43.606,"TRY":5.12
31,"USD":1.364,"ZAR":16.911,"EUR":1.1238}}
01-12 17:15:08.479 com.zacmurphy.currencyconverter V/QueryUtils:
urlConnection disconnected
01-12 17:15:08.479 com.zacmurphy.currencyconverter V/QueryUtils:
inputStream closed
01-12 17:15:08.479 com.zacmurphy.currencyconverter V/QueryUtils:
jsonResponse: {"base":"GBP","date":"2018-01-
12","rates":{"AUD":1.7326,"BGN":2.1979,"BRL":4.3783,"CAD":1.7075,"CHF":1.32
46,"CNY":8.8126,"CZK":28.681,"DKK":8.3709,"HKD":10.672,"HRK":8.3672,"HUF":3
46.97,"IDR":18183.0,"ILS":4.6382,"INR":86.749,"JPY":151.58,"KRW":1449.2,"MX
N":26.068,"MYR":5.4229,"NOK":10.855,"NZD":1.8797,"PHP":68.714,"PLN":4.6888,
"RON":5.2103,"RUB":77.214,"SEK":11.053,"SGD":1.8097,"THB":43.606,"TRY":5.12
31,"USD":1.364,"ZAR":16.911,"EUR":1.1238}}
01-12 17:15:08.479 com.zacmurphy.currencyconverter D/QueryUtils:
extractFeaturesFromJson - called
01-12 17:15:08.482 com.zacmurphy.currencyconverter V/QueryUtils: root:
{"base":"GBP","date":"2018-01-
12","rates":{"AUD":1.7326,"BGN":2.1979,"BRL":4.3783,"CAD":1.7075,"CHF":1.32
46,"CNY":8.8126,"CZK":28.681,"DKK":8.3709,"HKD":10.672,"HRK":8.3672,"HUF":3
46.97,"IDR":18183,"ILS":4.6382,"INR":86.749,"JPY":151.58,"KRW":1449.2,"MXN"
:26.068,"MYR":5.4229,"NOK":10.855,"NZD":1.8797,"PHP":68.714,"PLN":4.6888,"R
ON":5.2103,"RUB":77.214,"SEK":11.053,"SGD":1.8097,"THB":43.606,"TRY":5.1231
,"USD":1.364,"ZAR":16.911,"EUR":1.1238}}
01-12 17:15:08.482 com.zacmurphy.currencyconverter V/QueryUtils:
baseCurrency: GBP
01-12 17:15:08.482 com.zacmurphy.currencyconverter V/QueryUtils:
exchangeDate: 2018-01-12
01-12 17:15:08.482 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.483 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, AUD, 1.7326, 3
01-12 17:15:08.483 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.484 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
```

```
01-12 17:15:08.484 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, BGN, 2.1979, 10
01-12 17:15:08.484 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.484 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.485 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, BRL, 4.3783, 10
01-12 17:15:08.485 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.485 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.485 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, CAD, 1.7075, 4
01-12 17:15:08.485 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.485 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.485 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, CHF, 1.3246, 5
01-12 17:15:08.485 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.485 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.485 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, CNY, 8.8126, 6
01-12 17:15:08.485 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.485 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.485 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, CZK, 28.681, 10
01-12 17:15:08.485 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.485 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.485 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, DKK, 8.3709, 10
01-12 17:15:08.486 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.486 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.486 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, HKD, 10.672, 10
01-12 17:15:08.486 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.486 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.486 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, HRK, 8.3672, 10
01-12 17:15:08.486 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.486 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.486 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, HUF, 346.97, 10
01-12 17:15:08.486 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.486 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
```

```
01-12 17:15:08.486 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, IDR, 18183.0, 10
01-12 17:15:08.486 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.486 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.486 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, ILS, 4.6382, 10
01-12 17:15:08.486 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.486 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.486 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, INR, 86.749, 10
01-12 17:15:08.486 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.486 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.486 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, JPY, 151.58, 2
01-12 17:15:08.486 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.486 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.486 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, KRW, 1449.2, 10
01-12 17:15:08.487 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.487 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.487 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, MXN, 26.068, 10
01-12 17:15:08.487 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.487 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.487 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, MYR, 5.4229, 10
01-12 17:15:08.487 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.487 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.487 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, NOK, 10.855, 10
01-12 17:15:08.487 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.487 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.487 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, NZD, 1.8797, 10
01-12 17:15:08.487 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.487 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.487 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, PHP, 68.714, 10
01-12 17:15:08.487 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.487 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
```

```
01-12 17:15:08.487 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, PLN, 4.6888, 10
01-12 17:15:08.487 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.487 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.487 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, RON, 5.2103, 10
01-12 17:15:08.487 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.487 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.487 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, RUB, 77.214, 10
01-12 17:15:08.487 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.487 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.488 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, SEK, 11.053, 10
01-12 17:15:08.488 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.488 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.488 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, SGD, 1.8097, 10
01-12 17:15:08.488 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.488 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.488 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, THB, 43.606, 10
01-12 17:15:08.488 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.488 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.488 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, TRY, 5.1231, 10
01-12 17:15:08.488 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.488 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.488 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, USD, 1.364, 0
01-12 17:15:08.488 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.488 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.488 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, ZAR, 16.911, 10
01-12 17:15:08.488 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.488 com.zacmurphy.currencyconverter D/QueryUtils:
getPriority - called
01-12 17:15:08.488 com.zacmurphy.currencyconverter V/QueryUtils: 2018-01-
12, GBP, EUR, 1.1238, 1
01-12 17:15:08.488 com.zacmurphy.currencyconverter D/Currency: Constructor
- called
01-12 17:15:08.498 com.zacmurphy.currencyconverter D/LoadingActivity:
onLoadFinished() - called
```

```
01-12 17:15:08.499 com.zacmurphy.currencyconverter V/LoadingActivity:
currencies added to ArrayList
01-12 17:15:08.499 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.499 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.499 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.499 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.499 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.499 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.499 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.499 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.499 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.499 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.499 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.499 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.499 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.499 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.499 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.499 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.499 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.499 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.499 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.499 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
```

```
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
```

```
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.500 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.501 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.501 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.501 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.501 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.501 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.501 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.501 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
```

```
01-12 17:15:08.501 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.501 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.501 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.501 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.501 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.501 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.501 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.501 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.501 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.501 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.501 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.501 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.501 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.501 com.zacmurphy.currencyconverter D/LoadingActivity:
compare class - called
01-12 17:15:08.501 com.zacmurphy.currencyconverter V/LoadingActivity:
currenciesList re-ordered
01-12 17:15:08.501 com.zacmurphy.currencyconverter D/LoadingActivity: Error
occurred?: false
01-12 17:15:08.505 com.zacmurphy.currencyconverter D/LoadingActivity:
Activity switching
01-12 17:15:08.524 com.zacmurphy.currencyconverter E/AbstractTracker: Can't
create handler inside thread that has not called Looper.prepare()
01-12 17:15:08.524 com.zacmurphy.currencyconverter D/AppTracker: App Event:
stop
01-12 17:15:08.594 com.zacmurphy.currencyconverter D/MainActivity:
createSpinnerOptions - called
01-12 17:15:08.595 com.zacmurphy.currencyconverter D/CurrencyAdapter:
Constructor - called
01-12 17:15:08.595 com.zacmurphy.currencyconverter V/MainActivity: Listener
set on spinner
01-12 17:15:08.595 com.zacmurphy.currencyconverter V/MainActivity: Listener
set on 'Convert' button
01-12 17:15:08.595 com.zacmurphy.currencyconverter V/MainActivity: Listener
set on EditText
01-12 17:15:08.596 com.zacmurphy.currencyconverter V/MainActivity: Listener
set on 'Done' button
01-12 17:15:08.600 com.zacmurphy.currencyconverter E/AbstractTracker: Can't
create handler inside thread that has not called Looper.prepare()
01-12 17:15:08.600 com.zacmurphy.currencyconverter D/AppTracker: App Event:
start
01-12 17:15:08.628 com.zacmurphy.currencyconverter D/CurrencyAdapter:
getView() - called
01-12 17:15:08.632 com.zacmurphy.currencyconverter D/CurrencyAdapter:
getObjectResourceId - called
01-12 17:15:08.632 com.zacmurphy.currencyconverter V/CurrencyAdapter:
resourceId: 2130837619
```

```
01-12 17:15:08.677 com.zacmurphy.currencyconverter D/MainActivity:
onCreateOptionsMenu - called
01-12 17:15:08.903 com.zacmurphy.currencyconverter D/MainActivity: Spinner
item selected
01-12 17:15:08.903 com.zacmurphy.currencyconverter D/MainActivity:
onConvertClick - called
01-12 17:15:08.904 com.zacmurphy.currencyconverter D/MainActivity:
getSpinnerOption - called
01-12 17:15:08.904 com.zacmurphy.currencyconverter V/MainActivity: Selected
item: USD
01-12 17:15:08.904 com.zacmurphy.currencyconverter D/MainActivity:
getCurrencySymbol - called
01-12 17:15:08.904 com.zacmurphy.currencyconverter V/MainActivity: symbol:
$
01-12 17:15:08.904 com.zacmurphy.currencyconverter D/MainActivity:
getExchangedValue - called
01-12 17:15:08.905 com.zacmurphy.currencyconverter V/MainActivity:
userEnteredQuantity: |END
01-12 17:15:08.905 com.zacmurphy.currencyconverter V/MainActivity:
valueToConvert: 0.0
01-12 17:15:08.905 com.zacmurphy.currencyconverter V/MainActivity:
exchangeRate: 1.364
01-12 17:15:08.905 com.zacmurphy.currencyconverter V/MainActivity:
exchangedValue: 0.0
01-12 17:15:08.905 com.zacmurphy.currencyconverter D/MainActivity:
setResultFieldText - called
01-12 17:15:08.907 com.zacmurphy.currencyconverter D/CurrencyAdapter:
getView() - called
01-12 17:15:08.909 com.zacmurphy.currencyconverter D/CurrencyAdapter:
getObjectResourceId - called
01-12 17:15:08.910 com.zacmurphy.currencyconverter V/CurrencyAdapter:
resourceId: 2130837619
01-12 17:15:08.944 com.zacmurphy.currencyconverter D/CurrencyAdapter:
getView() - called
01-12 17:15:08.946 com.zacmurphy.currencyconverter D/CurrencyAdapter:
getObjectResourceId - called
01-12 17:15:08.946 com.zacmurphy.currencyconverter V/CurrencyAdapter:
resourceId: 2130837619
01-12 17:15:09.202 com.zacmurphy.currencyconverter D/LoadingActivity:
onLoaderReset() - called
```