



# The Android Application: Currency Exchange Rate

---

By David Sajdl

BSc Computing Project Report, Birkbeck College,  
University of London, 2015

This report is the result of my own work except where explicitly stated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged.

## Abstract

This project is aimed to develop an Android application which is used to provide an efficient and friendly way of identifying the best source of the currency exchange with respect to the most favourable exchange rate. The currency and the corresponding amount for exchange is defined by the user.

The application is developed on an Android platform and uses its built-in features. Moreover, with the use of the Android platform, the application aims to provide a simpler and faster way for selecting the online provider of the currency exchange.

## Table of Contents

<b>1.</b>	<b>Introduction .....</b>	<b>4</b>
1.1	Interest of the topic .....	5
<b>2.</b>	<b>Background .....</b>	<b>7</b>
2.1	Android History .....	7
2.2	Android Popularity Worldwide .....	7
2.3	Competition of the Currency Rate Exchange app on the Market .....	8
<b>3.</b>	<b>Requirements .....</b>	<b>9</b>
<b>4.</b>	<b>Design .....</b>	<b>10</b>
4.1	Visual Perspective of the System to Data Sources .....	10
4.2	Use Case .....	11
4.3	Activity Flow Chart Diagram .....	12
4.4	Classes diagram .....	17
4.5	Android Concept of Designing User Interface .....	24
<b>5.</b>	<b>Methods .....</b>	<b>26</b>
5.1	Research and Familiarization .....	26
5.2	Download, Archive and Process Data from RBS .....	26
5.3	Connecting CER app to x-Rate bank .....	27
5.4	Connecting Application to the Barclays Web Server .....	27
<b>6.</b>	<b>Graphical User Interface .....</b>	<b>29</b>
6.1	Information Section .....	29
6.2	Preference section .....	31
6.3	Spinner Section .....	32
6.4	Currency Conversion Calculator Section .....	32
<b>7.</b>	<b>Implementation .....</b>	<b>35</b>
7.1	Downloading Data from Web Server .....	35
7.2	SQLite Database .....	36
7.3	Spinner .....	38

7.4	Conversion Calculator .....	39
7.5	Setting Preference .....	39
7.6	Managing Toast .....	41
7.7	Notification .....	41
7.8	Layout XML .....	42
<b>8.</b>	<b>Testing .....</b>	<b>45</b>
8.1	Unit Testing.....	45
8.2	SQLite Database Testing .....	47
8.3	Preference Testing.....	48
8.4	Testing Other Activity Classes.....	49
<b>9.</b>	<b>Conclusions .....</b>	<b>50</b>
9.1	Enhancements .....	50
	<b>Appendix A - Android Issues .....</b>	<b>53</b>
	<b>Appendix B – Java Classes .....</b>	<b>58</b>
	<b>Appendix C - XML resource .....</b>	<b>104</b>
	<b>Appendix D – Unit Testing.....</b>	<b>116</b>
	<b>References .....</b>	<b>127</b>

# 1. Introduction

In this section I introduce the technology which is used to develop an IT environment and a rationale of choosing to design such an application.

As part of my BSc degree, I have chosen to carry out a project which encompasses utilizing an Eclipse. The Eclipse is an open source software called Integrate Development Environment (IDE) for Java project. According to [developer](#) 2015<sup>R.6</sup>, an Android offers a custom plugin for the Eclipse IDE, called Android Development Tools (ADT). This plugin provides a powerful, integrated environment in which to develop Android apps. It extends the capabilities of Eclipse to let you quickly set up new Android.

Moreover, the ADT plugin for the Eclipse integrates with Android Virtual Device (AVD) Management<sup>1</sup> which has been created by the google team. The Android project utilises an Eclipse which includes multiple component projects and libraries compiled into .jar file<sup>2</sup>.

Generally, Android is an operating system and runs in a similar way as PCs run Microsoft Windows as their operating system. There exist many mobile platforms on the market, including iOS “iPod Operation system for mobile”, Windows mobile, BlackBerry, Symbian, WebOS etc.

Android has also a built-in functionality for downloading data from different sources in web servers and is able to archive and update the data in a local SQLite<sup>3</sup> database.

---

1 The AVD Manager “provides a graphical user interface in which can be created and managed Android Virtual Devices (AVDs), which are required by the Android Emulator. Emulator is a virtual mobile device that is run on your computer. The emulator lets you develop and test Android applications without using a physical device([developer](#), 2015<sup>R.6</sup>)”.

2 “In software, [JAR \(Java Archive\)](#) is a package file format typically used to aggregate many Java class files and associated metadata and resources (text, images, etc.) into one file to distribute application software or libraries on the Java platform ([wikipedia-JAR\(file format\)](#), 18 February 2015<sup>R.18</sup>)”.

3 “SQLite is an in-process library that implements a self-contained, zero-configuration, server less, transactional SQL database engine. The source code for SQLite exists in the public domain and is free for both private and commercial purposes. SQLite has bindings to several programming languages such as C, C++, BASIC, Python ([techopedia.com](#), 2015<sup>R14</sup>)”.

The reasons for choosing this project for developing the Currency Exchange Rate (CER) application in Android are:

- Gaining knowledge of the Android framework.
- Expanding existing knowledge of the Java.
- Developing an application that would be able to communicate with other web services.
- Understanding of developing and designing Android applications.

The CER application allows to select foreign currency name via spinners and set banks or online conversion providers in the preference setting. Upon the spinner's value and preference's value a data, a currency rate value, is automatically retrieved from the database. The retrieved data is calculated upon the selected spinner's value and the result is displayed.

Additionally, the amount of money can be entered into a text editor and by pressing an equal button the entered data is calculated and displayed.

## 1.1 Interest of the topic

Here I summarise the most challenging and interesting areas of this project:

- Challenge to learn a new platform which I have not studied before.
- Learn and understand the Android framework which is one of the most popular mobile platforms worldwide.
- Challenge to fetch the data from web services into the application.
- Validate the data to ensure that the data contain correct information before it is stored into the database.
- Challenge to archive, update and retrieve data in the SQLite database.
- Process the data in terms of its representation in different currencies, for example, £1.00 in EURO, £1.00 in US Dollar, 1 EURO in Hong Kong Dollar etc.
- Challenge for designing a Graphical User interface (GUI).

The report contains further eight chapters. Chapter 2 discusses the background material of the project which includes a history of the Android platform and a competition for the CER app. Chapter 3 lists the main requirement for the project. Chapter 4 presents an application design which includes use cases, activity flow chart diagrams and class diagrams. Chapter 5 shows how the project was broken into sub-parts to achieve the final goal. Chapter 6 presents the Graphical User Interface (GUI) of the CER app. Chapter 7 discusses the implementation of the CER app. Chapter 8 shows the testing process of the application, including the unit test. Chapter 9 discusses a review of the final product and list potential enhancements of the CER app.

## 2. Background

In this chapter, I discuss the history of the Android mobile phone platform. I also highlight the popularity and selling figures of the Android mobiles' platform worldwide. Lastly, I discuss existing competitions for the Currency Exchange Rate app in the current market.

### 2.1 Android History

Android began in November 2007 with the release of the Android beta. The first commercial version Android 1.0 was released in September 2008. Android is under an ongoing development by Google and it is designed mainly for touchscreen mobile devices such as smartphones and tablet computers. The most recent Android update is Android 5.0 Lollipop which was released on the 3<sup>rd</sup> of November 2014.<sup>4</sup> Since Android has introduced its product on the market, the sell's figure of the Android platform has grown by approximately 70% as shown in Figure 1.

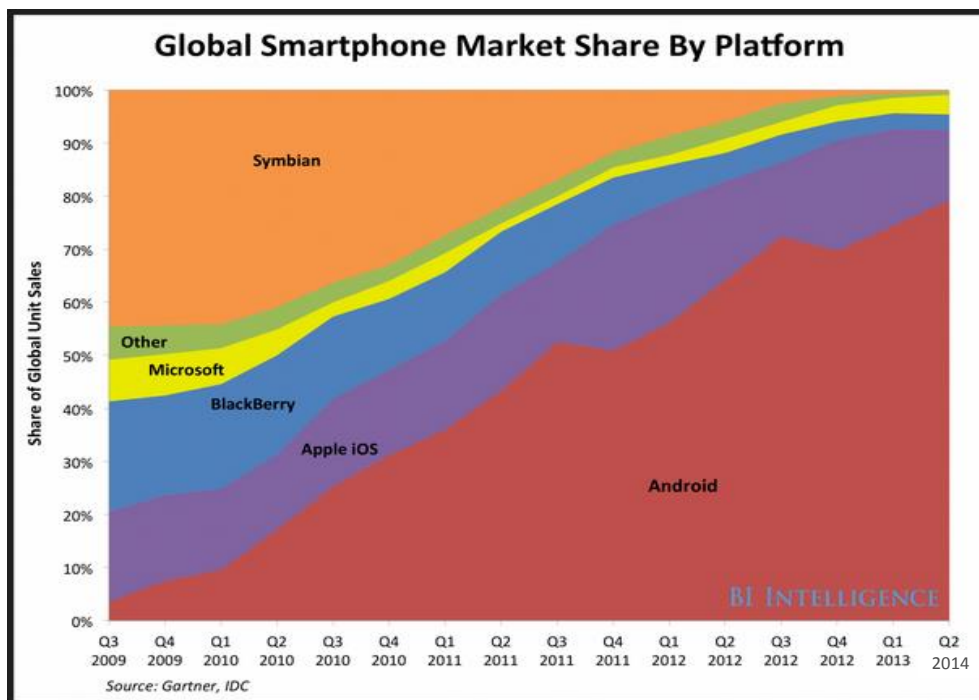


Figure 1: Represents sell unit of mobile phone platform worldwide form year 2009 to 2014.

Source: [go-Digital Blog on Digital Marketing \(2014\)](#) <sup>R.8</sup>

### 2.2 Android Popularity Worldwide

A worldwide smartphone market grows annually by approximately 27.2% which is just over a third of billion shipments of 335 million units. It is estimated that the market with Smartphone

<sup>4</sup> Source: [Wikipedia \(2015\) Android \(operating system\)](#) <sup>R.17</sup>



will continue to grow<sup>5</sup>. Table 1 shows a market share and unit growth of Android and other smartphones.

Worldwide Smartphone OS market Share in 2014 (Share in Unit Shipments)					
Period	Android	iOS	Windows Phone	BlackBerry OS	Other
2014	76.6%	19.7%	2.8%	0.4%	0.5%
2013	78.2%	17.5%	3.0%	0.6%	0.8%
2012	70.4%	20.9%	2.6%	3.2%	2.9%
2011	52.8%	23.0%	1.5%	8.1%	14.6%

Table 1: Source: [IDC Analyze the Future \(2014\)](#) <sup>R.10</sup>

Table 1 shows that the Android platform is the most selling smartphone platform worldwide with 76.6% in unit shipments. The Android platform takes a major share on the market and each year Android popularity grows<sup>5</sup>.

According to [developer](#) 2015 <sup>R.6</sup>, there are hundreds of millions of mobile devices in more than 190 countries around the world. It's the largest installed base of any mobile platform and growing fast—every day another million users power up their Android devices for the first time.

## 2.3 Competition of the Currency Rate Exchange app on the Market

The current market offers a couple of applications that provide foreign currency exchange. One of the most downloaded exchange app, with around 25million download, is XE Currency which is adaptable for any mobile platform including Android, iPhone and Blackberry. Another major player in the current market that offers a foreign currency exchange rate application is Google.

However, Google and XE Currency provide their currency exchange rate application with their own exchange rate source. On the other hand, the CER app provides a comparison of foreign currency exchange rates from three different sources. The CER application is unique in a way that offers the user to make a choice between different foreign exchange sources with more favourable exchange rate.

---

<sup>5</sup> Source: [IDC Analyze the Future \(2014\)](#) <sup>R.10</sup>

### 3. Requirements

In this chapter I list the main requirements which the CER app must have. The following requirements have been originally identified in the initial proposal of my BSc project, Currency Exchange Rate - Android application.

- The CER app is able to process data from different UK's banks, such as Royal Bank of Scotland, or different online currency converters, such as x-Rate, and offer the best three rating results. (For example, if RBS has the best rating offer, the app displays the first row as RBS has 1 GBP to 1 USD with a rate of 1.6089:1, or 1 USD to 1 GBP with a rate of 0.6215:1.
- The downloaded data is saved in the CER app. When the CER app is offline, the user is informed that the CER app is currently offline. The user is also informed which date the data is used from.
- The user is able to select different types of currencies from the spinner menu, such as the GBP to USD and the best exchange rate is automatically displayed.
- The user is able to input amount that he/she desires to exchange and the application automatically processes and displays results.
- The CER app is able to provide a list of all banks and online currency converters in a drop-down menu. If the user wants to use a particular bank or online currency converter, the app is able to process the user's request and display a particular bank with exchange rate offers.

## 4. Design

In this chapter, I present different stages of the Android CER app design. First, I discuss where the data for the Android CER app are downloaded from. The main functionalities are captured in the use case diagram which shows the user interaction with the Android CER app interface. The next discussion is on a logic and usage of the application, specifically, when the application starts running and follow to explain more permanent aspects of the application domain. Finally, I explain the main components used in designing a User Interface (UI) in Android.

### 4.1 Visual Perspective of the System to Data Sources

The CER app is able to download data from three different websites. One is taken from the RBS website<sup>6</sup>. Another downloaded data is taken from the x-Rate website<sup>7</sup> and the last downloaded data is taken from the Barclays website<sup>8</sup>.

An example of a visual observation of the RBS website is presented in Figure 2 on the right side. The application is able to connect to the RBS website and start downloading data from the website's table. The application archives the downloaded data into a database. More information how the application downloads and archive data into the database is presented in Chapter 7 Implementation. The other two websites follow the same principle of downloading data as the RBS website.

---

<sup>6</sup> Access for RBS website where the table is available if on the following URL:

<http://www.rbs.co.uk/personal/travel/g1/money/exchange-rates.ashx>

<sup>7</sup> Access for x-Rate website where the table is available if on the following URL:

<https://www.barclayscorporate.com/foreign-exchange-rates.html>

<sup>8</sup> Access for Barclay website where the table is available if on the following URL: <http://www.x-rates.com/table/?from=GBP>

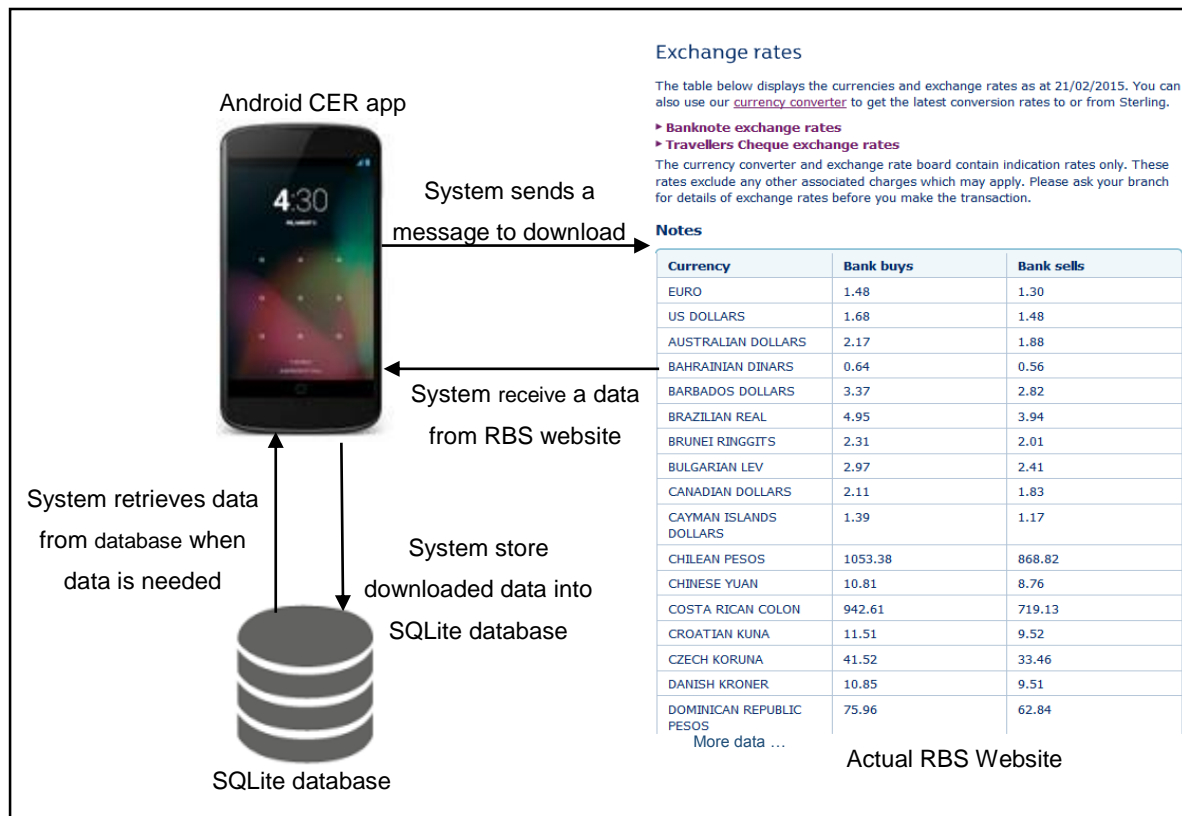


Figure 2: Source: [RBS \(2015\)](#)

## 4.2 Use Case

The use case diagram describes the user interaction with the system. Figure 3 presents four use cases which interact with the user (actor). The first use case on the top “Select bank preference” describes how the user is able to select the bank or online conversion in the preference option. The use case “Select bank preference” have a stereotype <<include>> to the use case “Display selected preference”, which means that the selected bank in the preference section is displayed.

The next two use cases “Select currency that user has” and “Select currency that user wants” describe the list of different currency names that the user can select. One which user has and the second one which the user wants. These two use cases have a stereotype <<include>> to the use case “Get rate from the SQLite and display rate”. The use case “Get rate from the SQLite and display rate” interact with the use case “Select bank preference”. Once the new currency has been selected in spinners, the system checks what preference value has been selected. The system then retrieves a data from the database upon the selected preference and spinner’s value. Finally, the system displays the result.

In addition, these two use cases have stereotype <<extend>> condition for use case “Download data and update SQLite database”, which means that it provides an additional condition to check whether the database of the system has used current updates. If yes, the application does not do anything. If not, the system sends a message to download and update the database.

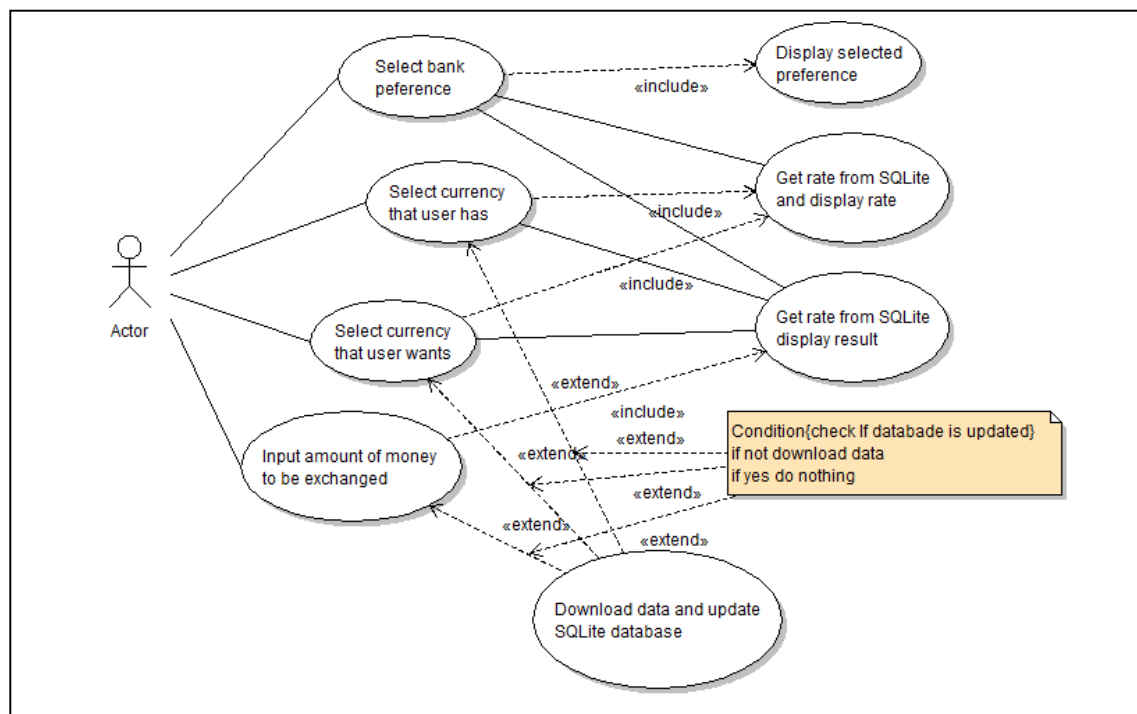


Figure 3: Use case diagram describes the user interaction with the application system.

The last use case that interacts with the user, Figure 3 on the bottom, “Input amount of money to be exchanged”, has <<include>> stereotype to the use case “Get rate from SQLite and display result”. The use case “Get rate from SQLite and display result” has interaction with three use cases: “Select currency that user has”, “Select currency that user wants” and “Select bank preference”. This means that the system allows the user to enter the amount of money that is required to exchange by the user. The system then checks which bank preference is selected and which currencies are selected and upon the selection it processes the user’s input, calculate the amount of money and display results.

### 4.3 Activity Flow Chart Diagram

The logic of the system, when the application gets active and the user starts interacting with the system, are described with three activity flow chart diagrams. First activity flow chart diagram describes downloading data and second diagram describes spinners’ process. The

last flow chart diagram describes the user interaction when the amount of money is inserted into the edit view and the equal button is pressed.

#### 4.3.1 Activity Downloading Data Process

Figure 4 shows the activity diagram for downloading data into the system when the application gets active.

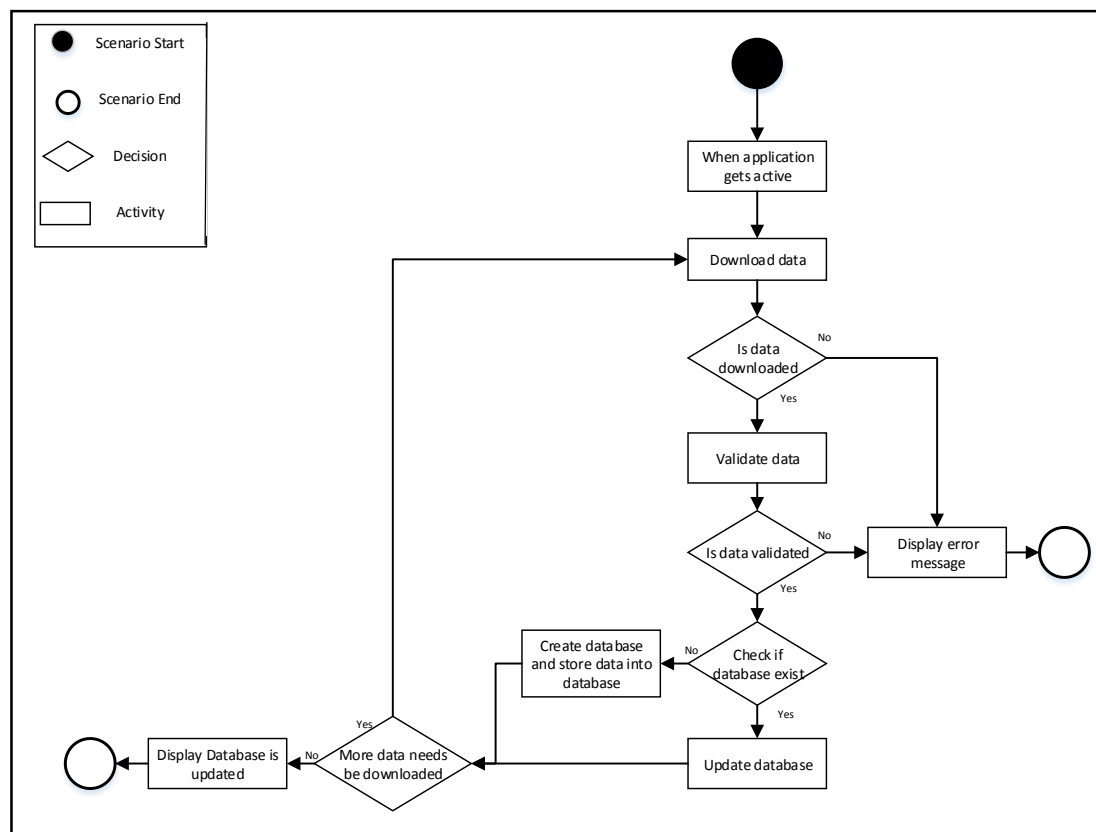


Figure 4: Activity flow chart diagram describes downloading data process of the CER app.

#### Downloading data process

- D.1. When the application gets active, it sends a message to do the next step.
- D2. The system starts downloading data from the particular web server, e.g. RBS web site.
- D3. The system checks whether the data is downloaded. If the data is downloaded, the system sends a message to do the next step. If the data is NOT downloaded, the system displays an error message D6.
- D4. When the data is downloaded to the system, the system validates the correct format of data.

- D5. The system checks whether the data is validated. If yes, the system sends the data for the next step. If not, the system displays an error message D6.
- D6. The system's pop up message, informs the user about the current state of the system e.g. no access connection to the Internet.
- D7. When the data is validated, the system checks whether the database exists in the system.
- D8. If the database exists in the system, the database is updated with downloaded data and the system sends a message to do the next step D10.
- D9. If the database does NOT exist, the system creates a new database and inserts downloaded data into the created database. The system then sends a message to do the next step.
- D10. The system checks whether other data needs to be downloaded for updating the database. If yes, the system sends a message to download a data (back to step D2). If not, the system sends a message for the next step.
- D11. The system displays a message that all database tables are updated.

#### 4.3.2 Activity Spinners Process

Figure 5 shows the activity diagram for processing spinners when the application gets active.

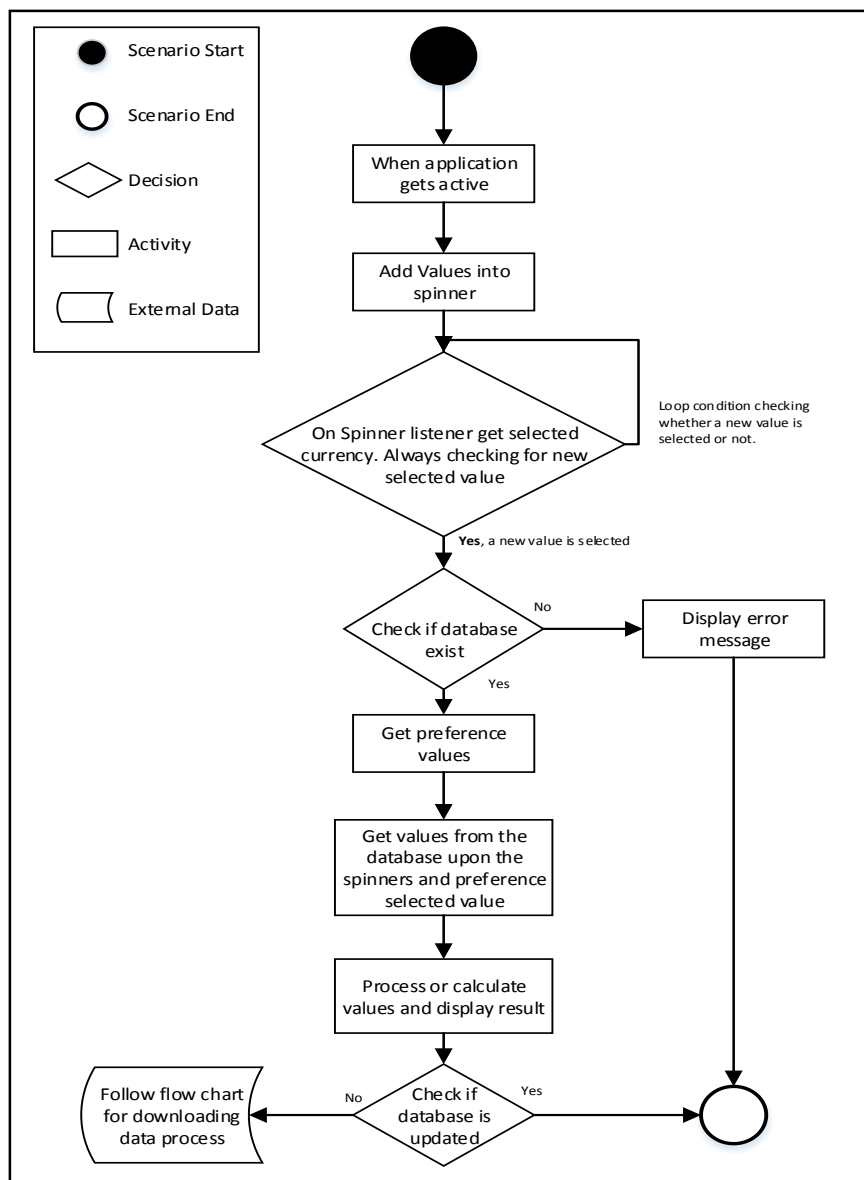


Figure 5: Activity flow chatr diagram describes spinner process in the CER app.

### Spinner process

- S1. When the application gets active, it sends a message to do the next step.
- S2. The system fills up the foreign currency's name value into the spinners.
- S3. The system constantly checks if a new value inside the spinners has been selected.
- S4. If a new value is selected, the system checks whether the database exists in the system.
- S5. If the database does NOT exist in the system, an error message is displayed.
- S6. If the database exists in the system, the system then retrieves the preference value (from the selected source).



- S7. The system retrieves the currency value from the database upon the selected preference value and upon the selected currency value. (for instance, if RBS is selected in the preference and USD is selected in the spinner then the value from the RBS database table, where the currency name is equal to USD, will be retrieved).
- S8. The system processes retrieved database values and display results.
- S9. The system checks whether the database is updated. If yes, the system does not do anything. If not, the system sends a message to download a data (see activity description Activity Downloading Data Process, step D2).

### 4.3.3 Activity Calculate the Input

Figure 6 presents the activity diagram for processing the user input when the equal button is pressed.

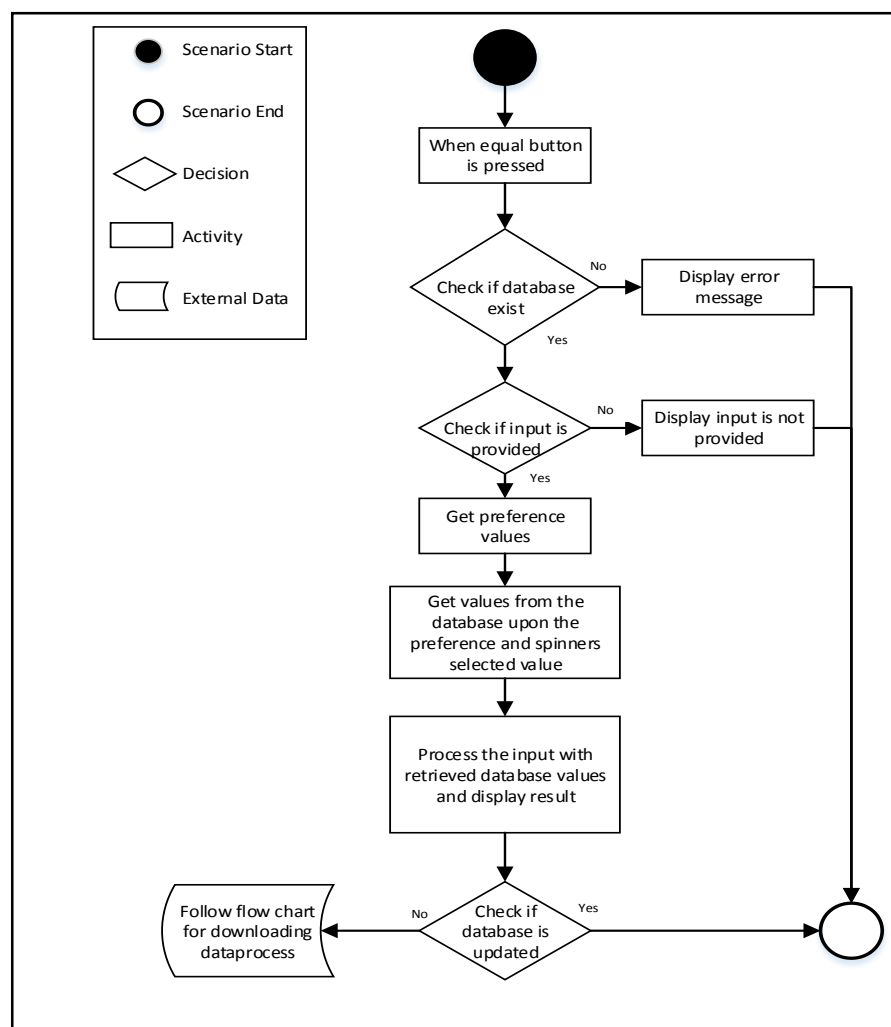


Figure 6: Activity flow chart diagram describes calculation of the input in the CER app.

### Calculation of the input

- CI1. When the user presses the equal button, the system sends a message to go to the next step.
- CI2. The system checks whether the database exists in the system.
- CI3. If the database does NOT exist, the system displays an error message.
- CI4. If the database exists, the system checks whether the input is provided by the user.
- CI5. If the user does not enter any input, the system displays a message “value is not entered”.
- CI6. If the value is entered by the user, the system then retrieves selected preference value (from the selected source).
- CI7. The system retrieves the currency value from the database upon the selected preference value and upon the selected currency value. (for instance, if RBS is selected in the preference and USD is selected in the spinner, then the value from the RBS database table, where the currency name is equal to USD, will be retrieved).
- CI8. The system processes and calculates the user input with retrieved database values and display the result.
- S9. The system checks whether the database is updated. If yes, the system does not do anything. If not, the system sends a message to download data (see activity description Activity Downloading Data Process, step D2).

### **4.4 Classes diagram**

The structure of the system is described by classes diagram showing the CER app’s classes and their attributes, methods and the relationship among objects. The structure of the system is divided into four categories. The first category consists of classes that allow opening a new activity. The second category consists of classes that manage a preference. The third category consists of classes that manage the SQLite database. The fourth category consists of classes that help the system to process data. Each category also contains the main class called CurrenciesrateExchangeMainActivity.

#### 4.4.1 Opening Activity Classes

Classes that enable to open a layout activity, or open a new layout activity on the top of the main activity are presented in Figure 7. The main class, which is a sub-class of the Activity class<sup>9</sup>, is called when the CER app gets active.

One of the method in the main class is onCreate() method which is called when the class gets active. It enables to load main\_activity.xml file to open the main layout activity.

Each class in Figure 7 has onCreate() method which is an override method from the super-class Activity. All classes enable to load a new xml file by using onCreate() method. The main Activity class is executed when the application gets active. Other classes are called from the main Activity class. For example, btnAbout() method calls About class, btnToUse() method calls Used Class, and NotificationView class is called by method displayNotification().

In addition, a notification text can be updated with different text by using method updateNotification(), or destroyed by using method cancelNotification().

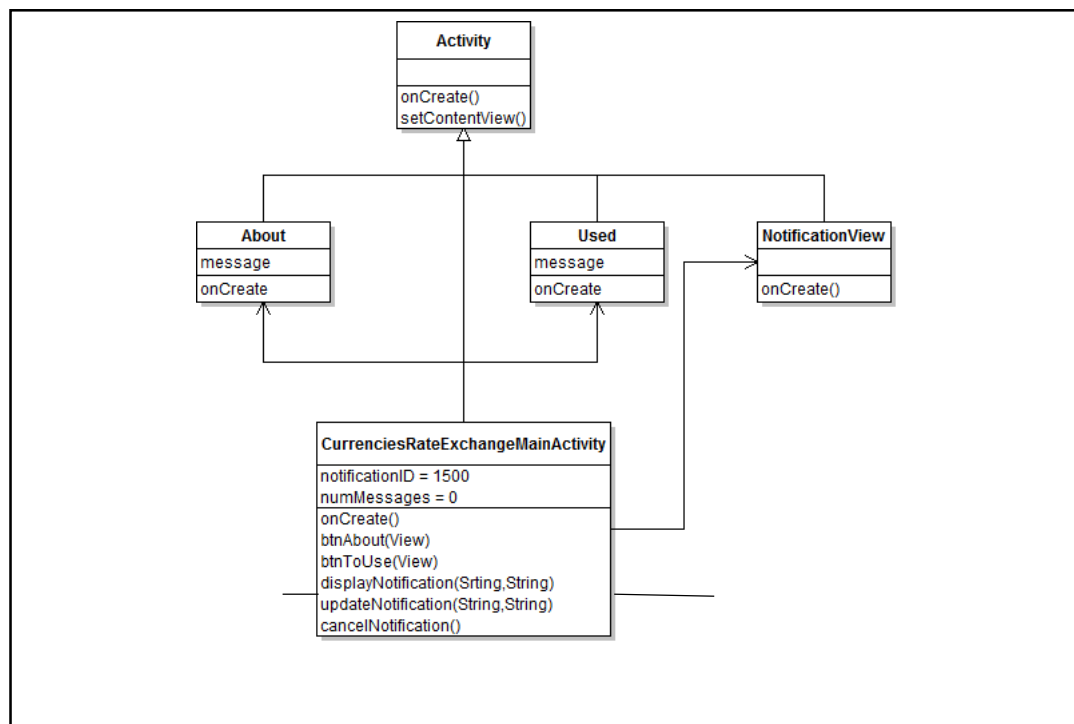


Figure 7: Classes diagram describes classes that enable to open a new xml layout.

<sup>9</sup> “The Activity class is an important part of an application's overall lifecycle, and the way activities are launched and put together is a fundamental part of the platform's application model (developer, 2015<sup>R.6</sup>)”.

#### 4.4.2 Manage Preference Classes

Classes that manage to select and save preferences are presented in Figure 8. The main class has two extra methods. The first method is `reloadPreferences()` which enable to load a preference value from `SharedPreference` and save it into an instance variable called a preference. The second method `setPreferences()`, which gets executed when the preference button is pressed, is called a `SetPreferenceActivity` class.

`SetPreferenceActivity` is a sub-class of `Activity` class and it is a composition of the `SetPreference` class, which is also a sub-class of the `PreferenceFragment` class<sup>10</sup>. The `SetPreference` class is able to open the `preference.xml` file by using the `addPreferencesFromResource()` method, which has the `preference.xml` file as a passing parameter. The `AddPreferencesFromResource()` method is an override method from the inherit `PreferenceFragment` Class.

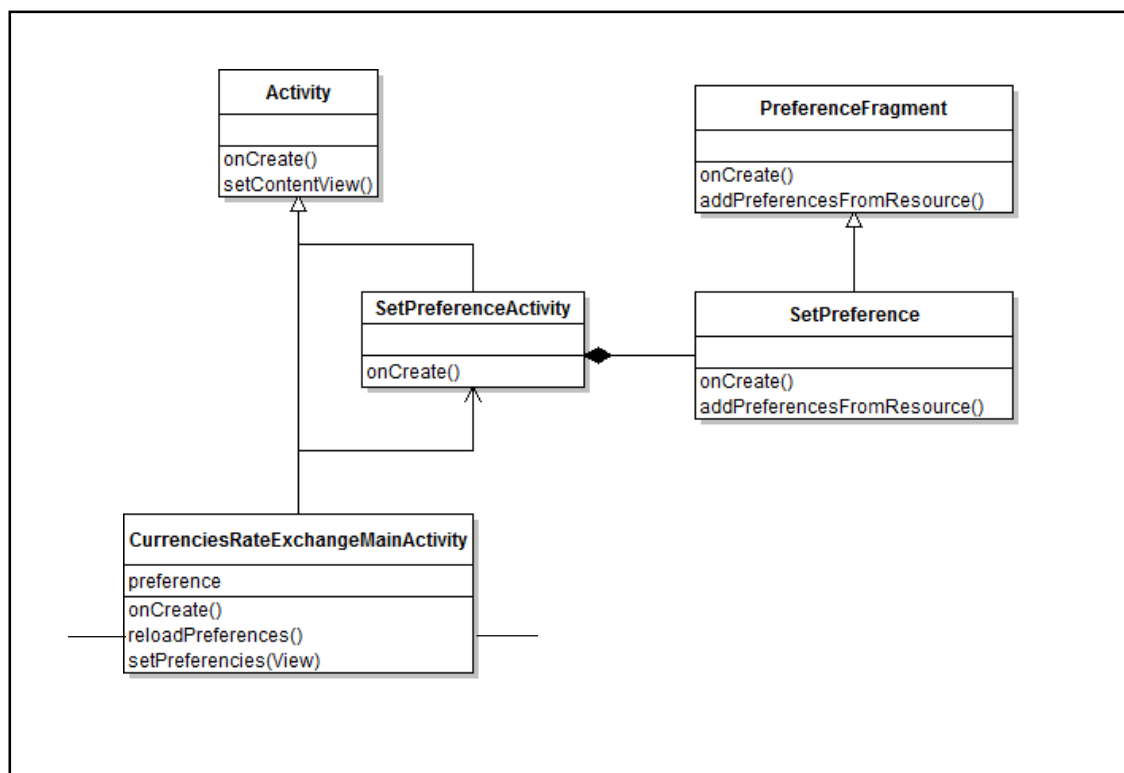


Figure 8: Class diagram describes classes that enable to manage preference.

<sup>10</sup> PreferenceFragmant shows a hierarchy of Preference objects as lists. These preferences will automatically save to `SharedPreferences` as the user interacts with them. To retrieve an instance of `SharedPreferences` that the preference hierarchy in this fragment will use, call `getDefaultSharedPreferences(android.content.Context)` with a context in the same package as this fragment ([developer](#), 2015<sup>R.6</sup>).

### 4.4.3 Manage Database Classes

The next category is classes that are able to create, archive and retrieve data from the database, show in Figure 9.

To create a SQLite database into an application the class MySQLiteHelper, which is a sub-class of SQLiteOpenHelper, needs to define a database name and SQL query statments to create database tables as instance variables. When the object of the class MySQLiteHelper is created, it creates SQLite database into the application.

To archive, update or retrieve data from tables, each table has its own source class and each source class is a composition of the MySQLileHelper class. Each source class communicates with one database table, for instance, XRateSource class is able to add or update values from X-Rate table by using addValueIntoTable() and updateValue() methods.

In addition, BarclaysDataSurce and XRateSource classes have interface class Source. It means that these two classes are forced to override five interface methods: open(), close(), addValuesIntoTable(), updateValue() and getCurrencyValue().

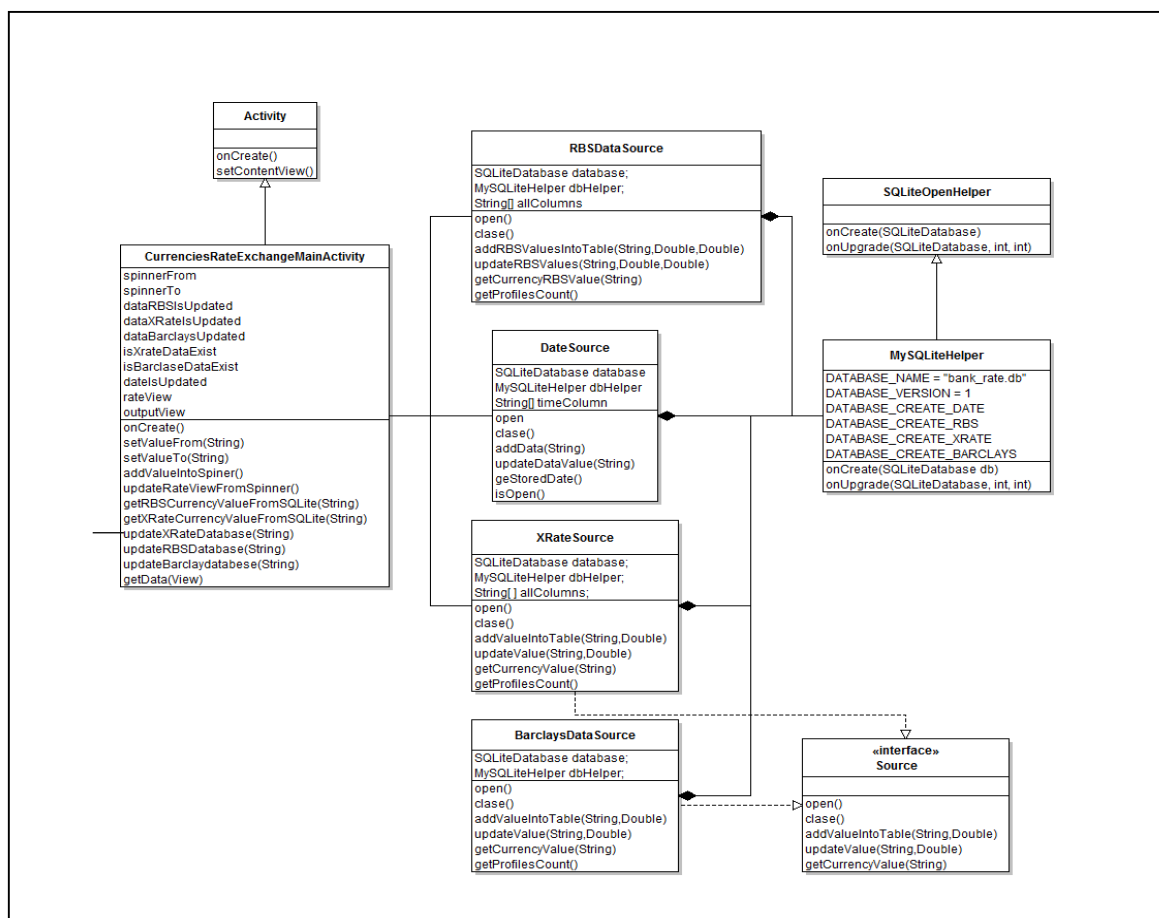


Figure 9: Class diagram describes classes that enable communication with database.

#### 4.4.4 Processing Data Classes

The last category is classes that help the system to process data by validating downloaded data, converting currency name, calculating data and getting current day as displayed in Figure 10.

The system downloads data from three different websites. The classes ValidateBarclaysData, ValidateRBSData and ValidateXRateData help to clear and validate data before the data is archived inside the database. All three classes implement interface class Validator forcing them to override five interface methods `getValid()`, `getErrorMessage()`, `findData(String)`, `getCurrency(String)`, `validateData(String, String)`.

The spinners hold different currency name than the currency name that are stored in the database tables. For instance, inside the spinners USD is written as USD – US Dollar and inside RBS table USD is written as US DOLLARS. The class SwitchCurrencyName enables to get the correct currency name that is stored inside database tables. When the foreign currency name is selected inside the spinners, by using the methods `switchXRateCurrencyName()`, `switchRBSCurrencyName()`, `switchBarclaysCurrencyName()`, it gets the right currency name for the particular database table.

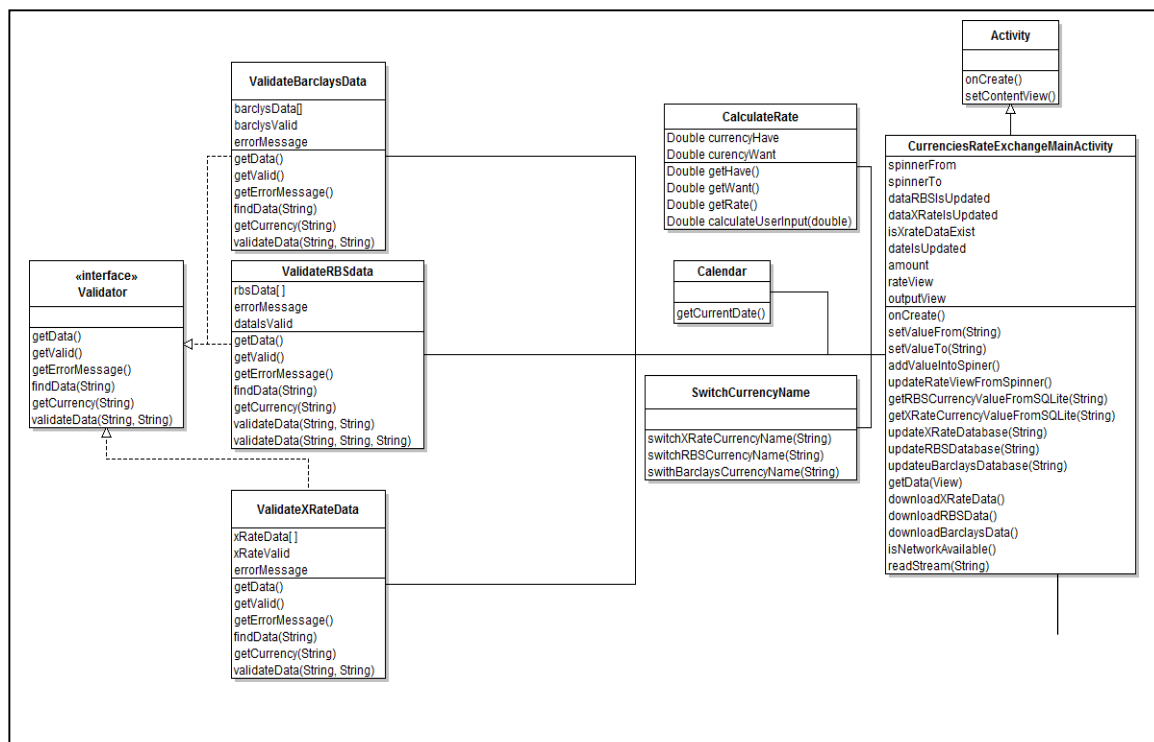


Figure 10: Class diagram describes classes that help to process data.

The CalculateRate class enables to calculate the conversion rate of the selected currency name by using method `getRate()`. The CalculateRate class has three more methods. Methods `getHave()` and `getWant()` return selected currency value. The `calculateUserInput(double)` method enables to calculate an amount that is specified by the user. Description how to calculate an exchange rate is described in the Appendix – A 3 Calculate Rate Exchange.

The last class in the process data diagram section is a Calendar class that enables to get, or creates a current date and time by using the `getCurrentDate()` method. The system uses a current date to be stored inside the Date table when all other three tables are updated to keep a record of the last table's update.

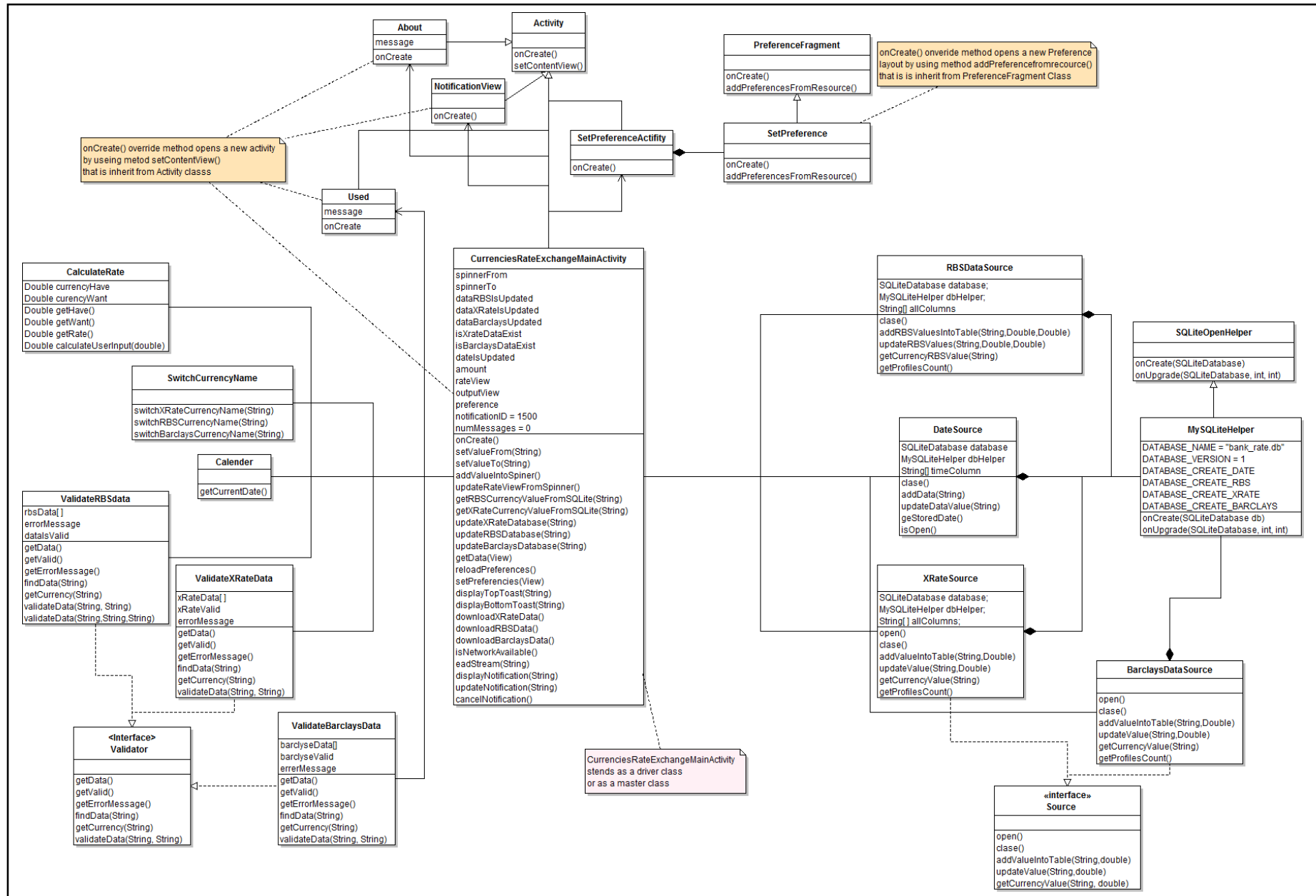


Figure 11: Class diagram describes classes for the whole CER app system.



## 4.5 Android Concept of Designing User Interface

According to [Skill Guru](#) 2014 <sup>R.12</sup>, good UI (user interface) is very essential for the success of any application in this competitive market. UI acts as a bridge between the user and the device. It provides all components required for communication with the device. In Android, there are three components which are required to build the visual UI:

- 1 Activity
- 2 View (widgets)
- 3 ViewGroup (layout)

### 1 Activity

Everything that can be seen on the Android screen is an Activity, for example, email messaging, photo gallery, etc. An activity creates a window, or certain container where classes (views) can be placed such as buttons and spinners. The activity base class is implemented as displayed in Figure 12.

```
public class ClassName extends Activity{  
    /* extend the base class Activity and implement the onCreate() method  
    * which is the first executed method when the application gets active. */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
    }  
}
```

Figure 12: Source: [Skill Gure \(2014\)](#) <sup>R.12</sup>

### 2 View

The View Class represents the basic building block for UI components, where each class view has its own attribute. The view's attribute can be modified by declaring attribute in xml file, or by activity using Java code at run time. View is the base class for Widgets, which are used to create interactive UI components such as:

- TextView
- Button
- EditText
- CheckBox
- ListView
- RadioButton
- Spinner
- WebView

### 3 ViewGroup

The ViewGroup sub-class is the base for layout classes, also called invisible containers, which hold other Views, or other ViewGroups and layout properties. Figure 14 shows a ViewGroup collection of different views. The implementation of the main layout ViewGroup for the CER app is presented in Chapter 7.8 Layout XML.

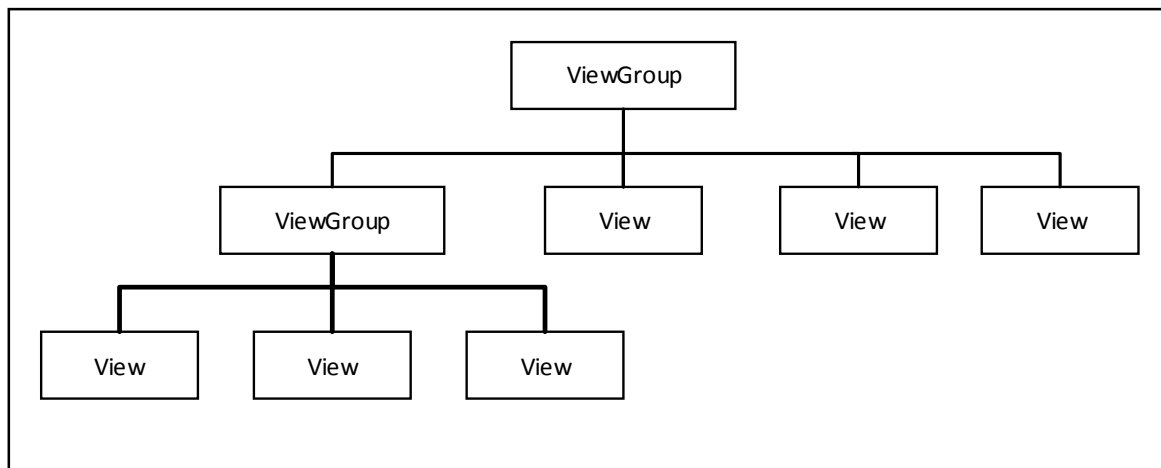


Figure 14: Source: [Skill Gure \(2014\)](#) R.12

## 5. Methods

In this chapter, I discuss how the development of the application was broken into four main categories. The first category was a research and familiarization with the Android platform. The second category was to download, archive and process data from the web server, of RBS website. The third category was to download data from the second web server of x-Rate, compare both downloaded data and implement preferences. The last category was to download the data from Barclays.

### 5.1 Research and Familiarization

The first step for developing the CER application was to get familiar with the Android development platform by following two books: Hello Android<sup>R.3</sup> and Beginning Android Development<sup>R.1</sup>. These two books introduced me to the Android framework. However, some of the exercises was not updated to the current day, which required further research from the following online sources: [developer](#)<sup>R.6</sup>, [stack overflow](#) websites.

### 5.2 Download, Archive and Process Data from RBS

The next step was to download data from the first website, RBS (Royal Bank of Scotland), by using ReadStreamTask class that inherits AsyncTack class and creates a connectivity manager object that allows to connect applications with the Internet. When data is downloaded, it is sent for a validation into the ValidateRBSdata class. After the validation, data is stored in the instance variable as well as in the SQLite database.

Next step was to display a message that the database was updated via Toast method. In addition, the spinners check whether downloaded data is stored in an instance variable. If yes, data is displayed. If not, data is taken from the database and message is sent to download data.

When the implementation described above was implemented, a better design was found out of downloading, archiving and processing data in order to reduce passing messages and make codes more efficient. Therefore, all created and implemented codes were destroyed and a new design was implemented.

A new design considers that spinners take the currency's value from the database and check if the database is updated. If yes, spinners do not do anything. If not, spinners send a message to download RBS data.

It works in a same way when the equal button is pressed. The button checks if the database is updated and if the input is provided. For more details about the design of downloading and processing data go to Chapter 4.3 Activity Flow Chart Diagram.

### 5.3 Connecting CER app to x-Rate bank

When the CER app is able to download, archive and process data from the RBS server, the application is then connected to another web server, x-Rate. To download, validate and archive x-Rate data into the database, the same tactic is used as for downloading RBS data. Then next step is to implement a preference.

The Preference means that the application includes a setting that allows the user to modify application features and behaviours. In the CER app, the user can modify which bank is displayed to calculate conversions. More information about Preference go to Chapter 6.2 Preference section. In this stage, the CER application provides three preference options:

- Best Result preference – the CER application will compare the currency's value and which currencies offer the best rate is displayed on the first line.
- RBS preference - means that only RBS data is displayed.
- X-Rate preference - means that only x-Rate data is displayed.

### 5.4 Connecting Application to the Barclays Web Server

The last step is to connect the application with the last web service Barclays. To download and archive data, the same technique is used as for downloading and archiving RBS data. Into the preference list, the item Barclays is added.

When the best result is selected in preference, the application displays the best result on the first line. On the second line, it displays the second best result and the worst result on the last line.

To implement this, a new method was implemented that receive three parameters as arguments which represent a value rate of three banks. This method compares which of three

sources have the best result and display it. Figure 15 presents a method with an if statement, which simulate a method of displaying the best result in the application.

```
public static void printBestResult(int a, int b, int c){
    if(a < b){
        if(a < c){
            if(b < c){
                System.out.println("hi = " + c + ", mid = " + b + ", lo = " + a);
            } else{
                System.out.println("hi = " + b + ", mid = " + c + ", lo = " + a);
            }
        } else if(c < b){
            System.out.println("hi = " + b + ", mid = " + a + ", lo = " + c);
        }
    }
    else if (c < b){
        System.out.println("hi = " + a + ", mid = " + b + ", lo = " + c);
    }else if(a < c){
        System.out.println("hi = " + c + ", mid = " + a + ", lo = " + b);
    }
    else{
        System.out.println("hi = " + a + ", mid = " + c + ", lo = " + b);
    }
}
```

**Figure 15.**

Additionally, how to calculate a currency conversion and what is mean by the best rating offers is discussed in Appendix A - Android Issues.

## 6. Graphical User Interface

In this chapter, I discuss and show The Graphical User Interface (GUI) of the CER app. The whole interface is divided into four sections. The first section appears on the top of the screen, which is called an information section. The second section is a preference, or setting section. The third section bellows the preference section is a spinners section. The fourth section appears on the bottom of the screen, called a currency conversion calculator section.

### 6.1 Information Section

The Information section in Figure 16 has three parts. On the top of the screen is a notification bar that allows displaying notification messages. Notification messages describe the application state such as when the database gets updated as displayed in Figure 16 in the image on the right side. In addition, a notification can be dragged down and its layout appears on the top of the main layout. On a new notification layout, further information of the current state of the application is described, see Figure 17.

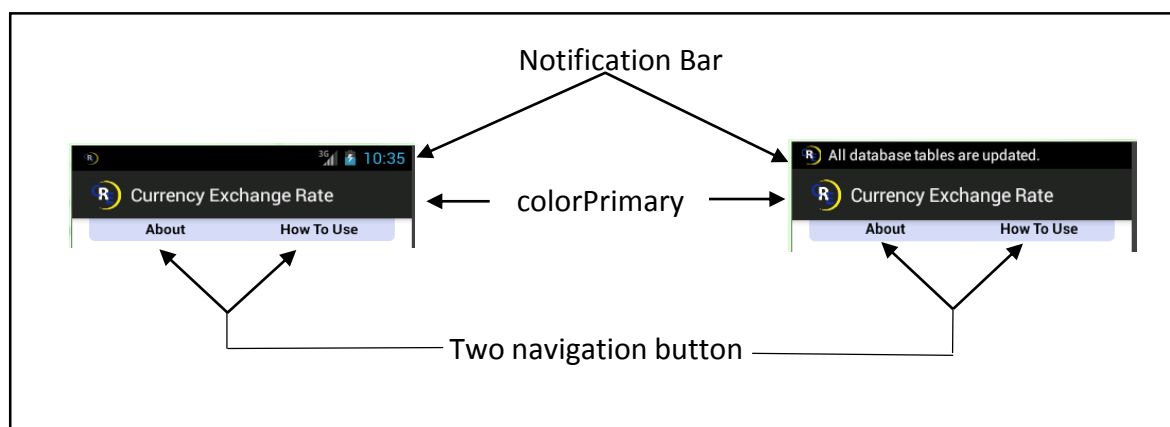


Figure 16: Represents the CER app interface of the information section.

In the middle of the information section is the colorPrimary bar. The colorPrimary bar displays logo of the CER with a title 'Currency Exchange Rate'. The last part in the information section is two views button. One button, on the left side, is called 'About', which opens a new activity, shown in Figure 18 image on the left, where a short brief describes what the CER app is about. The second button, on the right side, is called 'How to Use', opens a new activity, shown in Figure 18 image on the right, where a short brief describes how to use the CER app.

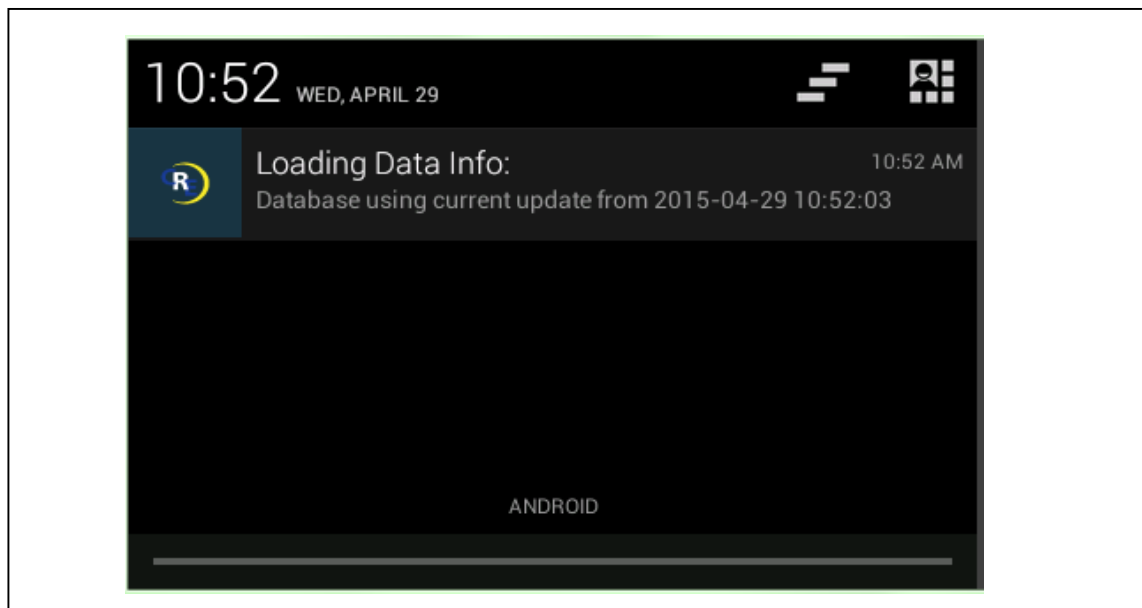


Figure 17: Represents notification layout activity in the CER app. It inform users about current state of the application.

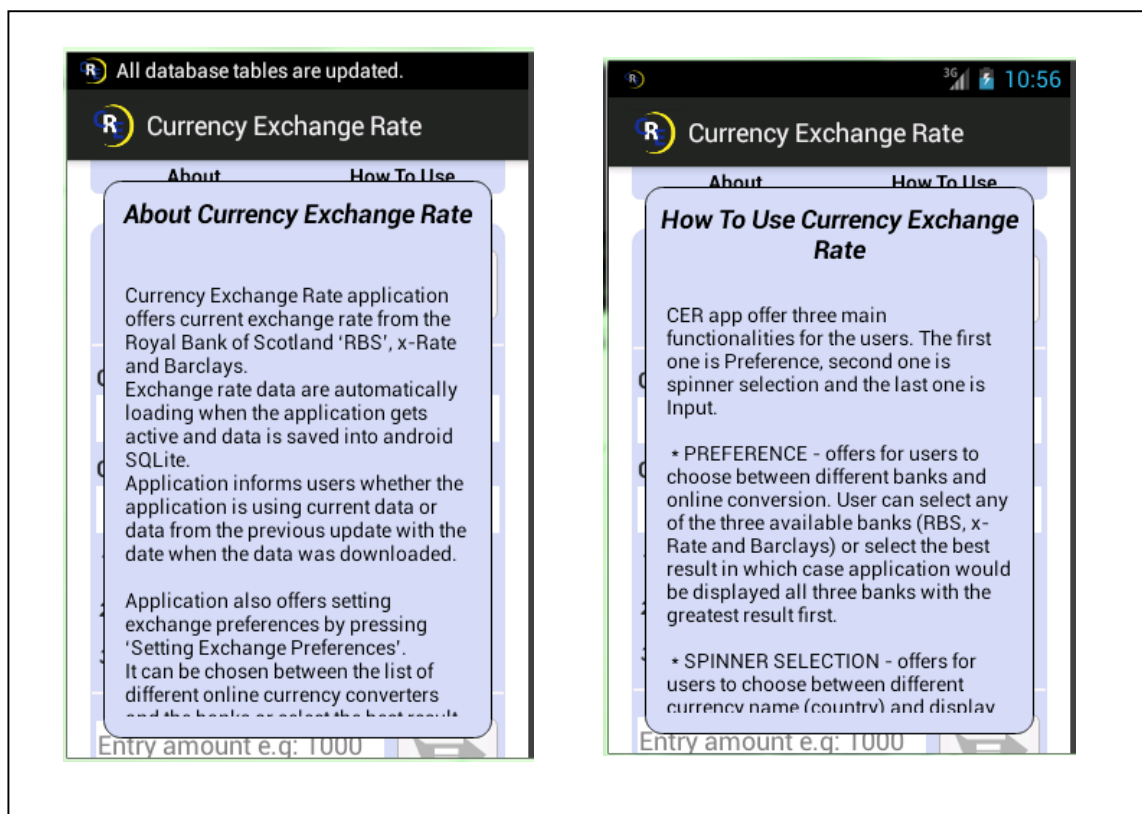


Figure 18: Represents screen shot of two activities in the CER app. The image on the left describes About activity and the image on the right describes How To User Currency Exchange Rate activity.

## 6.2 Preference section

The preference, or setting section contains two views. One on the right side is a text view which displays a bank, or an online conversion source that is currently selected. The second part on the left side is a button called Setting Exchange Preference, shown in Figure 19.

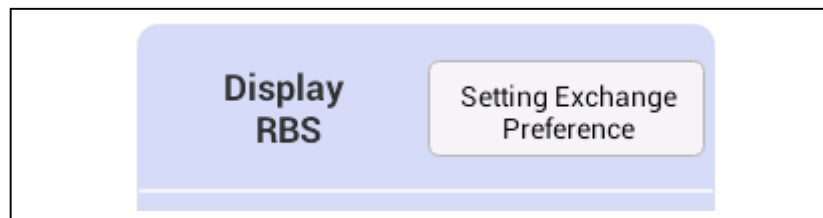


Figure 19: Represents the CER app interface of the preference section.

When the button is pressed, it opens a preference activity layout called Preference, Figure 20 image on the left. A Preference layout allows the user to set which bank is used to calculate the foreign currency rate by pressing “Select Rate Preferences” which opens a radio dialog box where the list of available banks is displayed, Figure 20 image on the right.

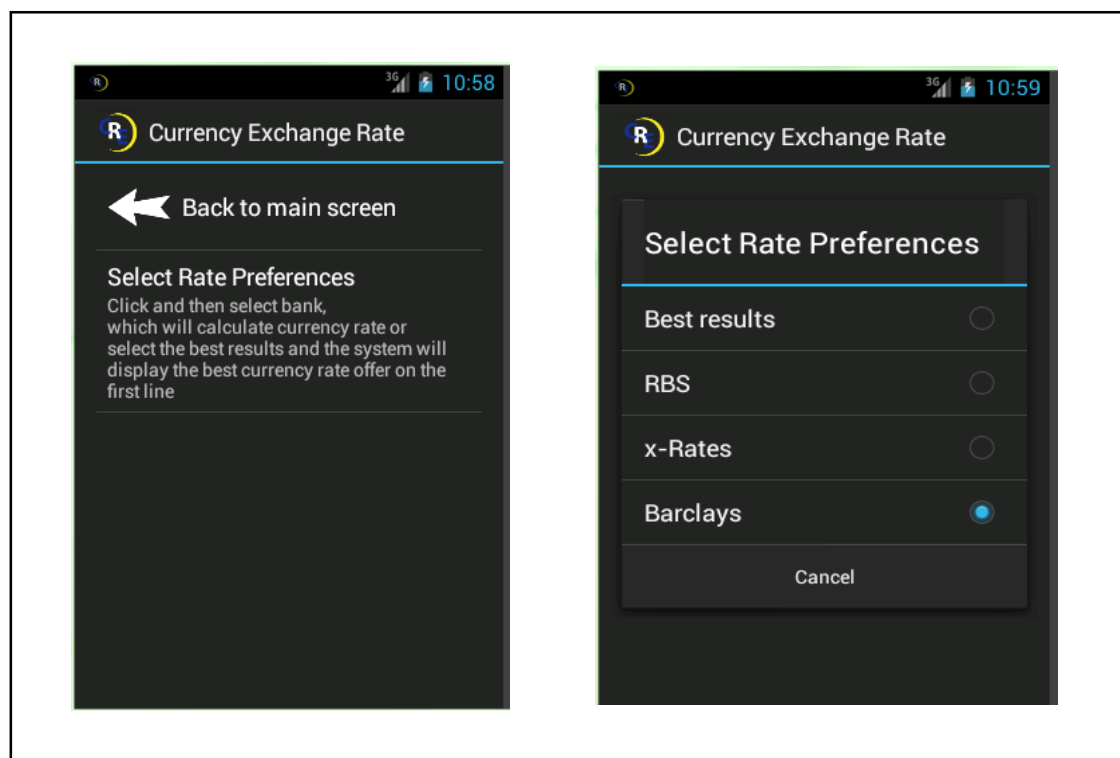


Figure 20: Represents preference layout screen.



### 6.3 Spinner Section

The spinner section has five views, two of them are spinners and three of them are text views, Figure 21.

When the CER app gets active, the first spinner has a selected value British Pound that represent currency the user has. The second spinner has a selected value Euro that represents the currency the user wishes to exchange for. Above both of spinners is a text view that describes the spinner states.



Figure 21. Represents the CER app interface of the spinner section.

On the bottom of the spinner section is a text view which automatically displays a current exchange rate of the selected spinners' value.

### 6.4 Currency Conversion Calculator Section

The currency conversion calculator section contains three views, Figure 22. On the right side is a text editor, where only numbers can be inserted. The maximum number that can be entered into the text editor is 7 digits "9 999 999" or 6 digits and one dot "9 999.99". The reason for limiting the user input for 7 digits is that the maximum number of double in Java is  $2^{32}$ . If a greater number is entered, then it may produce undefined numbers such as 9.9999999E7. Another reason is that it is unlikely that any user will hold a greater amount than 10 million.

On the left side, next to the text editor, is an equal button and below the text editor is a text view. When the equal button is pressed, it enables to calculate the input and display the result. If the equal button is pressed and the input was not entered, toast message appears on the bottom "Value is not entered", as displayed in Figure 22 image on the left.

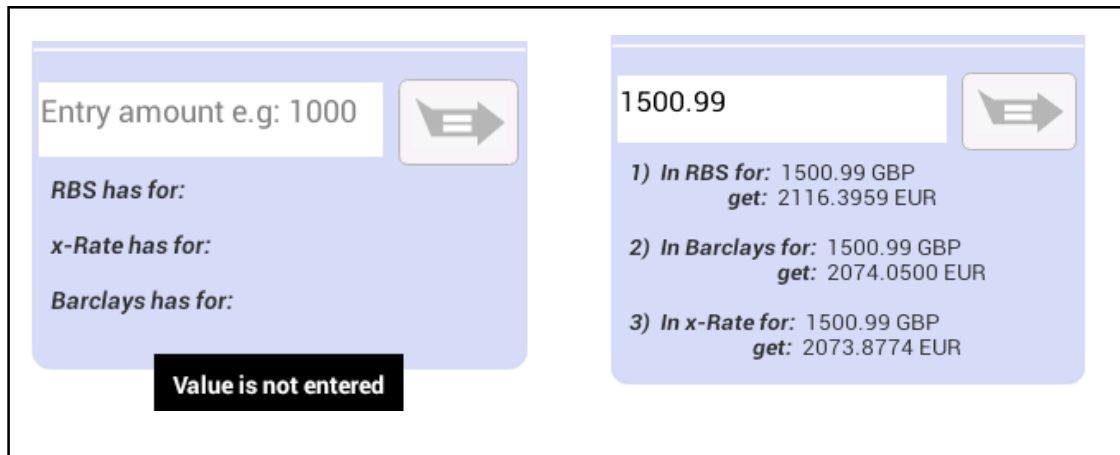


Figure 22: Represents the CER app interface of the currency conversion calculator section.

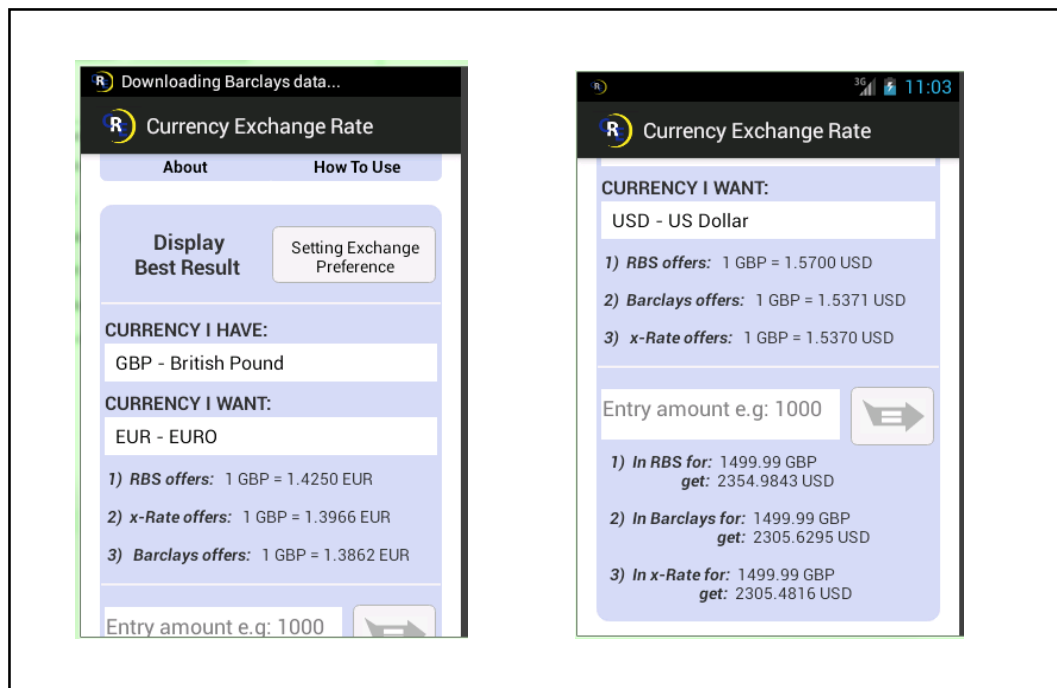


Figure 23: Represents the CER app interfaces.

- The image on the left represents the CER interface of information section, preference section and spinner section. Note a small notification message in the notification bar that inform the user of downloading Barclays data.
- The image on the right represents the CER interface of the spinner section and currency conversion calculator section.

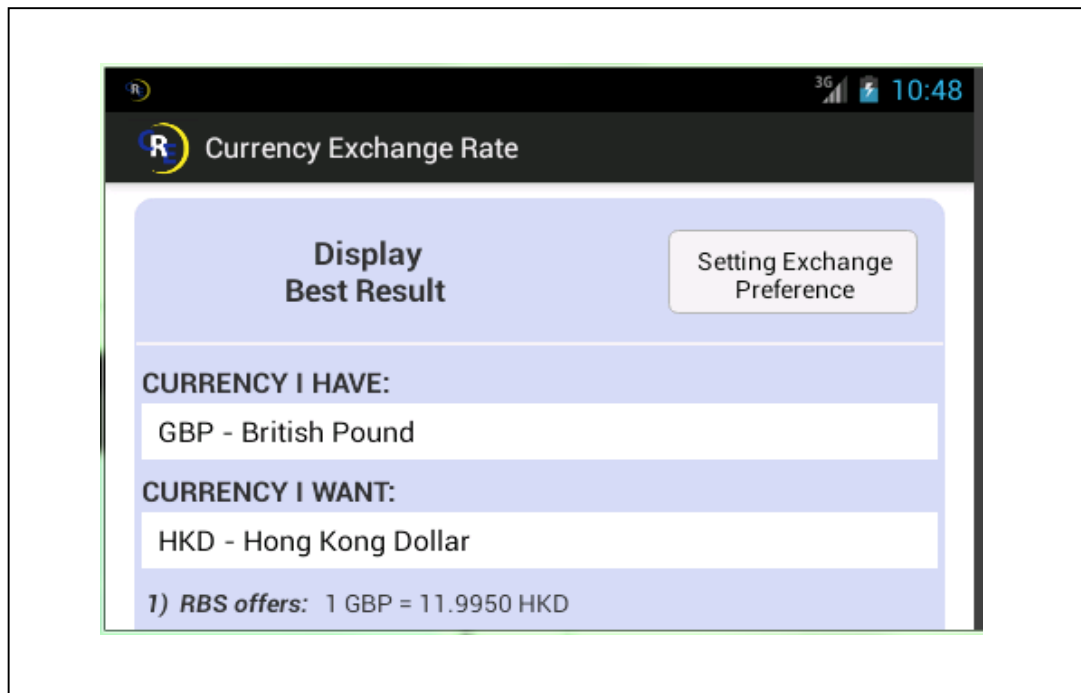


Figure 24: Represents the CER app interfaces when android device screen is in landscape orientation.

## 7. Implementation

In this chapter, I present the implementation for the CER app. I describe an overview of Java in the Android application and follow by giving a detailed account of downloading data from the web server, including its validation. I outline the following details:

- How Java manages to store and retrieve data from database.
- How Java uploads values into spinners.
- How Java manages to store and display preference.
- How Java manages to display notification and toast pop up message.
- An overview of the layout XML file, including an implementation of the main layout.

Java is a computer object-oriented programming language which provides the main operational logic in the Android application. The Java code enables connectivity between database SQLite and application, downloads data from a web server and changes states of the view at run time.

### 7.1 Downloading Data from Web Server

There are two ways for downloading a data from the web server. One way is to download binary data (text data) by using HTTP and the second way is by using a JSON web service (JavaScript Object Nation). For this project, I use a binary data downloaded by using HTTP.

The HTTP stands for Hyper Text Transfer Protocol which is most commonly used protocol to communicate over the Internet. In the Android application, the HTTP protocol can be used to perform tasks such as downloading web pages via binary data.

The way the application starts downloading data from the website is by checking if the Android device has network connectivity via the method `isNetworkAvailable()`. If the connectivity is available, a message is sent to the inner class which is a subclass of [`AsyncTask` class](#). The message to inner class carry a URL string that represents the URL of the particular web page from which a data will be downloaded.

The `AsyncTask` class uses three override methods. The first override method, called `onPreExecute()`, is executed before the data starts to be downloaded. The method sends a message to display a notification that it is downloading data. The second override method,

called `dolnBackground()`, Figure 25, receives a string URL from the parameter and starts downloading the HTML source code from the website and store it onto the string variable. The last override method is `onPostExecute()` which receives an output string that contains downloaded HTML source data from the method `dolnBackground()` as a parameter string and sends data to update the database. The `onPostExecute()` method also sends a message to cancel notification.

```
@Override
protected String dolnBackground(String... urls) {
    String response = "";
    for (String url : urls) {
        // create HttpClient
        DefaultHttpClient client = new DefaultHttpClient();
        HttpGet httpGet = new HttpGet(url);
        try {
            // make GET request to the given URL
            HttpResponse execute = client.execute(httpGet);
            // receive response as inputStream
            InputStream content = execute.getEntity().getContent();
            // creates a BufferedReader on the input stream and reads from it.
            BufferedReader buffer = new BufferedReader(new InputStreamReader(content));
            String s = "";
            while ((s = buffer.readLine()) != null) { response += s; }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    return response;
}
```

Figure 25.

## 7.2 SQLite Database

The Android application is able to create its own database called SQLite which takes approximately 150KB within a memory budget of the Android device. The SQLite database can archive structured data much more efficiently and it does not require setup, administration, or a config file. It is free and the Android store the database into the `data/data/package_name/database` directory.

```
// Constant that represent database
protected static final String DATABASE_NAME = "bank_rate.db";
private static final int DATABASE_VERSION = 1;

// Constants that represent table informations for RBS
protected static final String TABLE_RBS_NAME = "rbs_data";
protected static final String COLUMN_ID = "id";
protected static final String COLUMN_CURRENCY_NAME = "currency_name";
protected static final String COLUMN_BUYS = "buy";
protected static final String COLUMN_SELLS = "sell";
```

Figure 26.

## MySQLiteHelper Class

To create a database in the application, the MySQLiteHelper class is implemented which is subclass of the SQLiteOpenHelper class. The MySQLiteHelper class encapsulates all complexities of handling database operations and overrides two methods of its super class called onCreate() and onUpgrade().

Inside the MySQLiteHelper class, a bunch of string constants is initialed, Figure 26, that represents different information about the database, e.g., database version, name, etc. Some of the string constants are declared for SQL syntax to create tables such as RBS table, Figure 27. The MySQLiteHelper class contains four constrains for creating four tables; RBS, x-Rate, Barclays and Date. Java code for MySQLiteHelper class is presented in Appendix - B MySQLiteHelper.java.

```
// table creation as sql statement for RBS table
private static final String DATABASE_CREATE_RBS = "create table " + TABLE_RBS_NAME + "("
    + COLUMN_ID + "integer primary key autoincrement," + COLUMN_CURRENCY_NAME
    + " text not null, " + COLUMN_BUYS + " REAL, " + COLUMN_SELLS + " REAL)";
```

Figure 27.

When a new instance of the MySQLiteHelper class is created, a Context object is passed as a parameter argument. The MySQLiteHelper class constructor calls the super class constructor and passes three arguments: the Context object, database name and database version. It then creates a new database by calling onCreate() method. In the case that the database is already created, it checks a current database version against the supplied database version. If both versions are the same nothing happens. If the supplied database version is a greater than the currently installed database version, the onUpgrade() method is called that deletes a current database and creates a new database.

## Managing Database

To manage retrieving, inserting or updating data from database tables, I create four Name\_Source classes. Each source class belongs to one database table. All Source classes work in the same way. Here is how the RBSDataSource class works.

When a new instance of the class RBSDataSource is created, it has a Context object as a parameter argument. The constructor of RBSDataSource creates a new object of the instance MySQLiteHelper class. The RBSDataSource class has five methods.

The first and second method open and close database by using open() and close() method. The third method, addRBSValuesIntoDatabase(), manages to add value into the database table by using the insert() method. The fourth method, updateRBSValues(), manages to update RBS values in the database table by using the update() method. The fifth method getCurrencyRBSValue() enables to retrieve currency values from columns buy and sell and returns the average of these two currency values by using the rawQuery() method.

### 7.3 Spinner

According to [developer](#) 2015<sup>R.6</sup>, spinners provide a quick way to select one value from a set. A Spinners' value in the CER app represents the currency's name. In the default state, it shows its currently selected value. By pressing the spinner, a drop-down menu is displayed with all other available currencies' name, where a new currency name can be selected.

```
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(  
    this, R.array.countries_name, android.R.layout.simple_spinner_item);  
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);  
spinner_from.setAdapter(adapter);
```

Figure 28.

When the main activity gets active, the addValuesIntoSpinner() method is called that enables to load values from a string array by creating an instance of the ArrayAdapter class that load array values into itself by calling the createFromResource() method. The values from the instance ArrayAdapter are displayed into the spinner by using the setAdapter() method, Figure 28.

The addValuesIntoSpinner() method has the inner class called setOnItemSelectedListener(). The setOnItemSelectedListener() class gets executed when the spinner's value is changed. The class enables to retrieve a new selected spinner's value and store the value in an instance variable. It then sends a message to update the text view by calling the updateRateViewFromSpinner() method.

The updateRateViewFromSpinner() method enables to check which preference is selected. Upon the preference and spinner's value, the data is retrieved from the database and by using

the `getRate()` method of the class `CalculateRate`, the data is calculated. The calculated data is then displayed via a text view.

## 7.4 Conversion Calculator

By pressing the equal button, the `getData ()` method is executed. This method checks if the database exists. If the database does not exist, it displays a toast message “Android device needs to be connected to the Internet to load a data. Please make sure that Android device has a connection to the Internet.” If the database exists the method checks whether a data values are entered. If not, a toast message “Value is not entered” is displayed. If yes, the method checks which preference is selected. Through the instance of the class `CalculateRate`, the `calculateUserInput()` method is called. The `calculateUserInput()` method has the user input as a passing argument parameter, which enables to calculate how much money the user gets from the input and return the value. Finally, the `getData()` method displays the returned result from the `calculateUserInput()` method via a text view.

## 7.5 Setting Preference

[Preference](#) is a way how the user can interact with application’s features. In the CER app, the setting preference can be used to select which bank, or online conversion source is used to calculate and display currency exchange rate. To select a new bank, the Setting Exchange Preference button needs to be pressed.

When the Setting Exchange Preference button is pressed, the `setPreferences()` method gets executed. The `SetPreferences()` method, called the `SetPreferenceActivity` class, allows to open a preference layout by calling the `PreferenceFragment` class, which uses the `addPreferenceFromresource()` method. The `addPreferenceFromresource()` method passes a parameter called `preferences.xml` or preference layout. In addition, every new activity class needs to be declared in the `AndroidManifest.xml` file<sup>11</sup>.

The preference layout has Select Rate Preferences which behaves like a button. When Select Rate Preferences is pressed, a ratio dialog box appears, where a list of available banks as radio

---

<sup>11</sup> “Every application must have an `AndroidManifest.xml` file (with precisely that name) in its root directory. The manifest file presents essential information about app to the Android system, information the system must have before it can run any of the app's code. ([developer](#), 2015<sup>R.6</sup>)”.



buttons can be selected. More information and visual presentation of the preference is given in Chapter 6.2 Preference section

The selected bank's value is stored inside the SharedPreferences file. Then by using the reloadPreference() method in the main activity, the bank's value from the SharedPreferences file is retrieved, assigned into an instance variable, called preference, and displayed.

### XML File Preferences

To use a preferences.xml file inside the res directory, a new sub-directory needs to be created, called XML, which creates a preferences.xml file. Preferences.xml is presented in Appendix – C, preferences.xml and it has two preferences classes. One is called Preference class and the second is called ListPreference class.

The Preference class behaves like a button and it has three items:

- Icon – allows displaying image calls black\_arrow.png from the directory drawable.
- Title – provides a user-visibility name for setting “Back to main screen”.
- Key – specifies the unique key (string), which the system uses when saving the value in SharedPreferences.

The ListPreference class opens a dialog box with a list of radio buttons. These radio buttons allow the user to select a list of the bank, or online conversion source. The ListPreference class has six items:

- Title and key – have the same meaning as in the Preference class.
- Summary – gives a description of the ListPreference selection.
- DefaultValue – specifies the initial value that the system should set in the SharedPreferences file.
- Entries – display names of the bank and online conversion source as radio buttons. The banks' name is loaded from an array called arrayListOfBanks. The array is initialized in the string.xml file.
- EntryValues – represent values of the bank and online conversion source name. The entryValue is used to identify which bank is selected. The values for the entryValues item are loaded from an array, called arraysValuesOfBanks, which is also initialized in the string.xml file.

## 7.6 Managing Toast

According to [developer](#) 2015<sup>R.6</sup>, a toast provides a feedback about a current operation in a small popup message. A Toast class can be imported and used via `makeText()` method, which displays a toast popup message, Figure 29.

```
Toast.makeText(this,"message" ,Toast.LENGTH_LONG).show();
```

**Figure 29.**

The disadvantage of using the Android build-in toast is that the background and the text colour cannot be modified. Therefore, the CER app defines its own custom toast. To build a custom toast, a customized layout needs to be created in the XML file for toast notification in the layout folder. The customized layout needs to be loaded from the layout folder which create a new Toast class and set properties by using a Java code. The detail of creating the customized layout called a `toast_layout.xml` is presented in Appendix – C, `toast_layout.xml`.

To Manage Displaying Custom Toast Layout, the main Class has two methods, which create a new Toast object, that enables to load the custom layout. One is called `displayBottomToast()` and the second is called `displayTopToast()`. Both methods load the custom toast layout, called `notification.xml`, and create a new toast object. They have the same execution, except setting the properties for a gravity. The `displayTopToast()` method sets the gravity to the top (`TOP|Gravity.CENTER,0,49`) such that the pop up message appears on the top of the screen and the `displayBottomToast()` method sets the gravity to the bottom (`BOTTOM|Gravity.CENTER,0,0`) such that the pop up message appears on the bottom of the screen.

## 7.7 Notification

According to [developer](#) 2015<sup>R.6</sup>, the notification allows the user to keep informed about relevant and timely events in the apps such as new chat messages from friends. In the CER app, the notification keeps the user informed about what data is being currently downloaded and when data is updated in the database.

To provide the notification message in the CER app, a new xml layout is created called `notification.xml`. Three methods, `displayNotification()`, `updateNotification()` and `cancelNotification()`, are implemented in the main class to control and display notification

messages. To help methods displaying a new xml layout, a new class called NotificationView is created, that enables to load a notification.xml layout. Additionally, the NotificationView class needs to be declared in the AndroidManifest.xml file<sup>12</sup>.

### Notification XML Layout

I create a Notification.xml file in the layout folder, which has the root element LinearLayout, see Appendix – C, notification.xml. Notification.xml has one child called TextView, where notification messages are displayed. The TextView file has three attributes. Two of them describe the text view layout, height and width, and the third attribute describes a text which is set empty. The text of the notification is declared in run time by using Java code.

To manage Notification, the Activity class has one method that enables to create a notification called displayNotification() and one method that enables to update notification called updateNotification(). Both of these methods have two string arguments as parameters. The first string represents a short message that appears in the notification bar (see Figure 16 on page 29) and the second string is the message that appears in the notification layout.

Both methods create the object of the class Notification.Builder (Context), where messages and image are inserted by using the following methods:

- setTicker() – enables to display a notification message in the notification bar
- setSmallIcon() – enables to display the CER logo (image in .png file)
- setTitle() – enables to set the title of the notification
- setStyle().bigText() – enables to display a message notification layout and to set multiple line text below the title.

Finally, the notification object which set the text and image is passed to the system by calling NotificationManager.notify().

## 7.8 Layout XML

According to [developer](#) 2015<sup>R.6</sup>, a layout defines the visual structure for a User Interface (UI), such as the UI for an activity or app widget. The layout can be declared in two ways. The first

---

<sup>12</sup> “Every application must have an AndroidManifest.xml file (with precisely that name) in its root directory. The manifest file presents essential information about app to the Android system, information the system must have before it can run any of the app's code.([developer](#), 2015<sup>R.6</sup>)”.

way is using UI elements in XML and the second way is instantiating layout elements at runtime via Java code. In the CER application, all layouts are declared in XML.

The XML layout (or group view) is a container with one or more child objects (view), or with more group views, refer to Chapters 2 View and 3 ViewGroup. The most common layouts in Android are:

- [FrameLayout](#) - arranges its children so they all start at the top left of the screen. This is used for tabbed views and image switchers (Marko Gargenta, 2013, Hello Android, page 35<sup>R.3</sup>).
- [LinearLayout](#) - arranges its children in a single column or row. It is the most used layout” (Marko Gargenta, 2013, Hello Android, page 35<sup>R.3</sup>).
- [RelativeLayout](#) - arranges its children in relation to each other or to the parent. This is often used in forms (Marko Gargenta, 2013, Hello Android, page 35<sup>R.3</sup>).
- [TableLayout](#) - arranges its children in rows and columns. It behaves like an HTML table (Marko Gargenta, 2013, Hello Android, page 35<sup>R.3</sup>).

According to the website [developer-declaring-layout.html](#) [developer](#) 2015<sup>R.6</sup>, the advantage of using the XML layout is that it enables one to better separate the presentation of the application from code that controls its behaviour. It can be easily modified, or adapted without having to modify the source code. Additionally, declaring the layout in XML makes it easier to visualize the structure of the UI, hence, it is easier to debug problems.

The CER app has five XML layouts which are placed in the res/layout folder. Later in the Activity class, the following layout resource is loaded by using the onCreate() method.

- about.xml – is used when “About” button is pressed.
- activity\_main.xml – is used when the CER application gets active.
- how\_to\_use.xml – is used when the “How To Use” button is pressed.
- notification.xml – is used when data is getting downloaded and when all database tables gets updated.
- toast\_layout.xml – is used for popping up a small message.

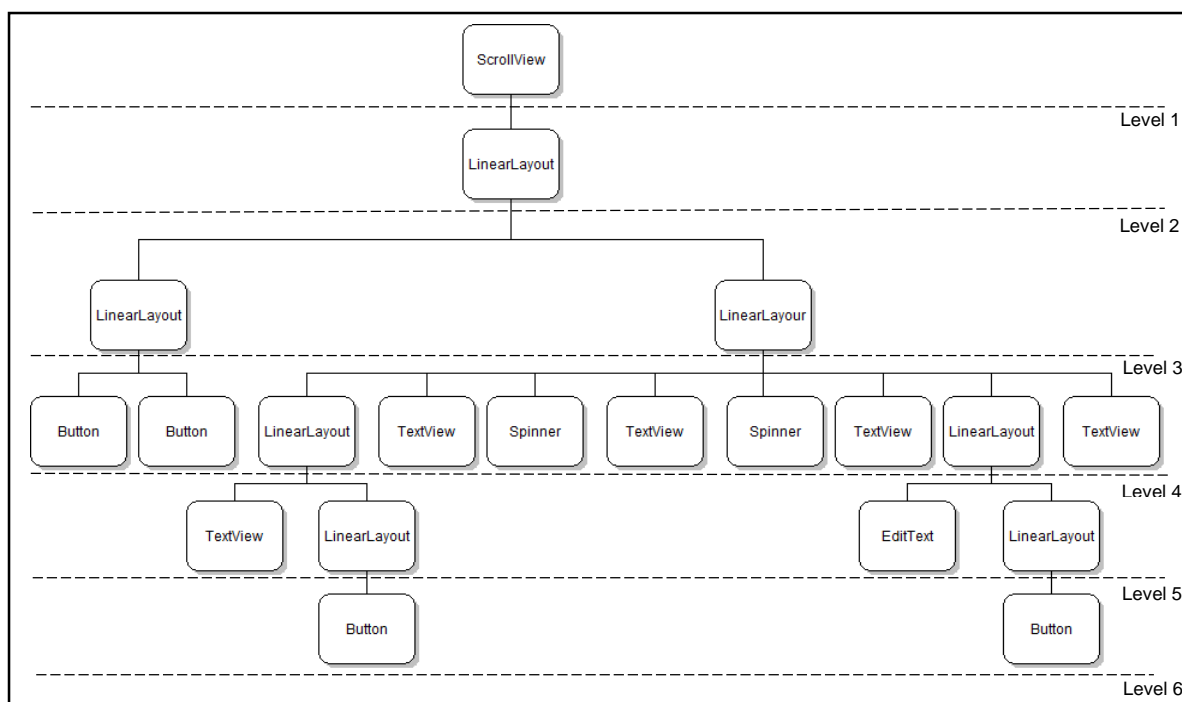
Additionally, the activity\_main.xml file appears in the res/layout-land folder, which is used for the landscape mode, and the res/layout-small, which is used for a small screen. The activity\_main.xml file in the res/layout folder is used in a portrait mode (adjusting screen).

All three layouts have the exact same child object and same elements, only values of their properties are set differently.

The Activity\_main layout has the top ViewGroup ScrollView<sup>13</sup> which is the first level of the hierarchy. The ScrollView class contains one child LinearLayout (ViewGroup) as the second level, Figure 30. The LinearLayout (ViewGroup) class on the second level describes whole Layout of the application, where the background colour property is set to white and it has two children LinearLayouts that are on the third level.

The child LinearLayout on the third level left side has two buttons view called “About” and “How To Use”. The second LinearLayout on the right side is where all other children’s views are placed such as spinners, buttons, edit text, etc. It has the background colour property set to light blue.

In addition, the LinearLayout child on the third level right side has two ViewGroups (LinearLayouts) as children. The views in the LinearLayout appear one below another. To have two views next to the other view, the LinearLayouts classes are implemented, where one has the property for the gravity set to the left and the second one has the property for the gravity set to the right.



**Figure 30: Describes designing hierarchy of the view group collection of the main layout for the CER app.**

<sup>13</sup> “ScrollView Layout container for a view hierarchy that can be scrolled by the user, allowing it to be larger than the physical display. A ScrollView is a FrameLayout, meaning you should place one child in it containing the entire contents to scroll; this child may itself be a layout manager with a complex hierarchy of objects. ([developer](#), 2015<sup>R.6</sup>)”.

## 8. Testing

This chapter discusses the testing of the CER app. I first present the unit test which was written for the application. Second, I show database tables tests, that is, whether data is inserted into database tables and currency values are updated by using the emulator<sup>9</sup>. In the last part, I explain the test of preferences via the emulator.

### 8.1 Unit Testing

According to [Lars Vogel](#) 2014<sup>R.16</sup>, a unit test is a piece of code written by a developer that executes a specific functionality in the code to be tested. The percentage of code which is tested by unit tests is typically called test coverage. The unit code ensures that the implemented code works as planned, including a correct delivery. It also helps developers to debug errors within methods, after the method is modified, or extend functionality.

The unit testing of the CER app concentrates on all non-activity classes. These classes provide additional functionalities to the system that could be unit tested. The classes that include the unit test are:

- CalculateRate
- Calendar
- SwitchCurrencyName
- ValidateBarclaysData
- ValidateRBSdata
- ValidateXRateData

Unit test codes were created to test all methods of these classes in a separate Android project file called CurrenciesRateExchangeTest as the Android Test project. The unit test result of all methods in non-activity classes can be seen in Figure 31.

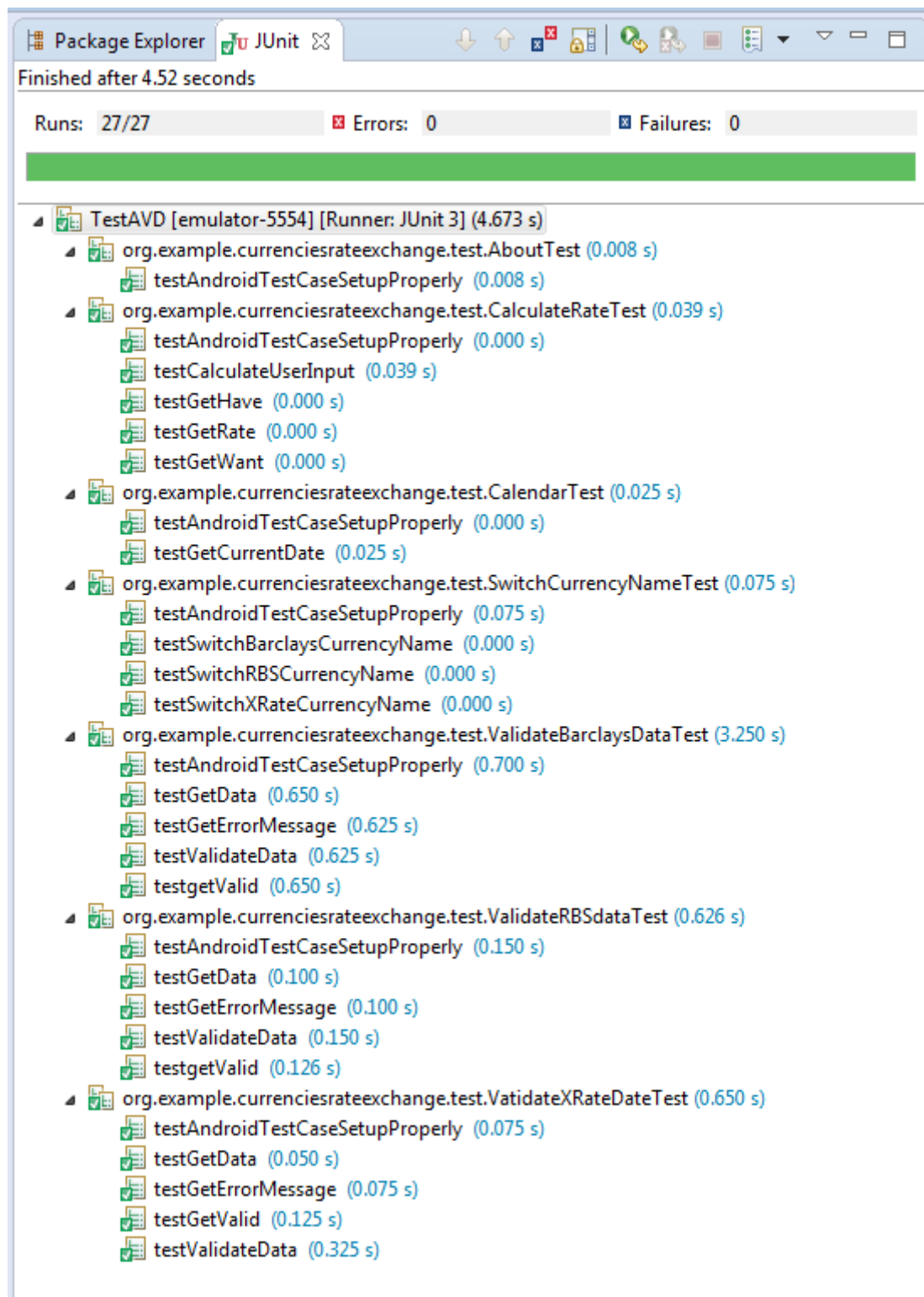
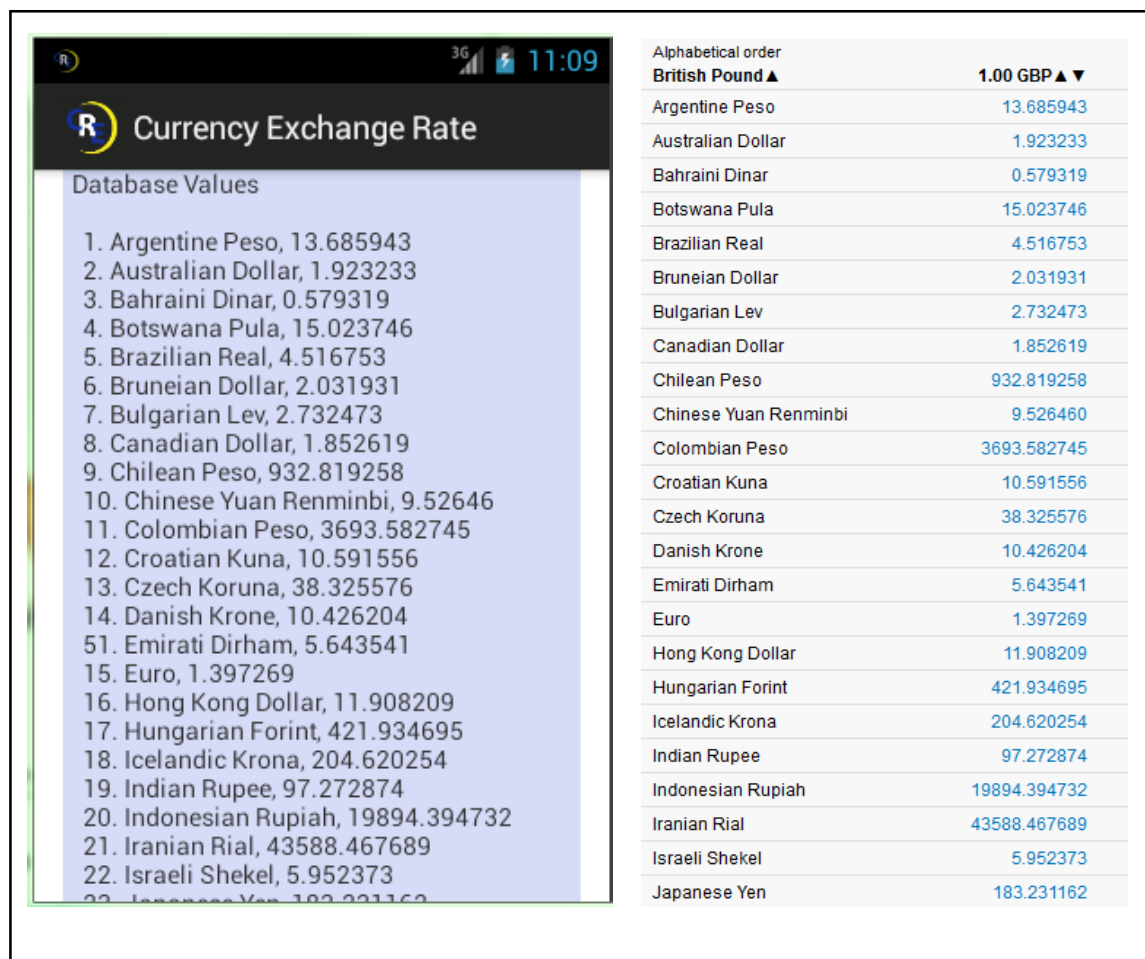


Figure 31: Describes the unit test result of all non\_activity classes and their methods.

Additionally, the unit test codes of all non-activity classes are available in Appendix D – Unit Testing.

## 8.2 SQLite Database Testing

Once the data has been downloaded and the values are inserted into the database table, through the `getAllTuples()` method, all data are retrieved and printed by using the emulator<sup>14</sup> as seen in Figure 32 on the left which represents the data from the x-Rate table. The data is also compared such that the printed data contain the same values as presented in the x-Rate website, see Figure 32 on the right. Next day, the data is printed again and database values are checked. By using this tactic, all tables in the SQLite database is tested.



Database Values	Alphabetical order
1. Argentine Peso, 13.685943	British Pound ▲ 1.00 GBP ▲ ▼
2. Australian Dollar, 1.923233	Argentine Peso 13.685943
3. Bahraini Dinar, 0.579319	Australian Dollar 1.923233
4. Botswana Pula, 15.023746	Bahraini Dinar 0.579319
5. Brazilian Real, 4.516753	Botswana Pula 15.023746
6. Bruneian Dollar, 2.031931	Brazilian Real 4.516753
7. Bulgarian Lev, 2.732473	Bruneian Dollar 2.031931
8. Canadian Dollar, 1.852619	Bulgarian Lev 2.732473
9. Chilean Peso, 932.819258	Canadian Dollar 1.852619
10. Chinese Yuan Renminbi, 9.52646	Chilean Peso 932.819258
11. Colombian Peso, 3693.582745	Chinese Yuan Renminbi 9.526460
12. Croatian Kuna, 10.591556	Colombian Peso 3693.582745
13. Czech Koruna, 38.325576	Croatian Kuna 10.591556
14. Danish Krone, 10.426204	Czech Koruna 38.325576
15. Emirati Dirham, 5.643541	Danish Krone 10.426204
16. Euro, 1.397269	Emirati Dirham 5.643541
17. Hong Kong Dollar, 11.908209	Euro 1.397269
18. Hungarian Forint, 421.934695	Hong Kong Dollar 11.908209
19. Icelandic Krona, 204.620254	Hungarian Forint 421.934695
20. Indonesian Rupiah, 19894.394732	Icelandic Krona 204.620254
21. Iranian Rial, 43588.467689	Indian Rupee 97.272874
22. Israeli Shekel, 5.952373	Indonesian Rupiah 19894.394732
23. Japanese Yen, 183.231162	Iranian Rial 43588.467689
	Israeli Shekel 5.952373
	Japanese Yen 183.231162

**Figure 32:** Represents a simulation of comparing data from x-Rate table in the database with data in the x-Rate website

- The image on the left represents printed data from the x-Rate database table of the CER app.
- The image on the right represents x-Rate table from the [x-Rate website](#).

<sup>14</sup> “The emulator lets you develop and test Android applications without using a physical device ([developer](#), 2015<sup>R.6</sup>).”



When the CER app manages to update all three tables in the database, the database is dropped down by using the code in Figure 33. The application is then run via the emulator again and checked whether it manages to create a database and insert the data into tables. The data from database tables are printed and compared with data in website tables as described in the previous paragraph.

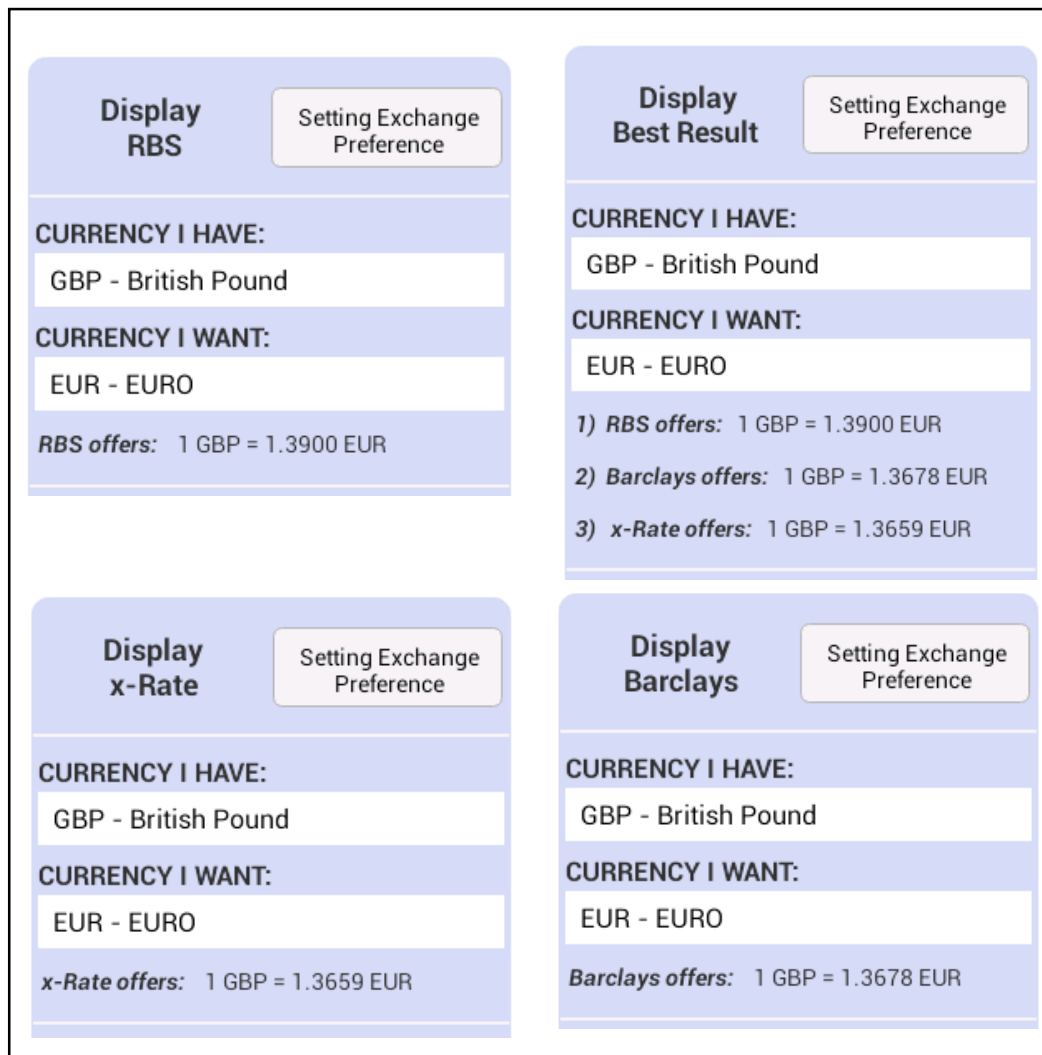
```
// delete database
this.deleteDatabase("bank_rate.db");
//display pop up message of deleting database success
displayBottomToast("database was deleted " + dtb.exists());
// set all database tables are updated to true to ensure that the database
// is not created again

dataRBSIsUpdated = true;
dataXRateIsUpdated = true;
dataBarclaysUpdated = true;
```

Figure 33: Describes codes that manage to drop down the database in the CER app .

### 8.3 Preference Testing

Each bank source is selected and tested such that the application displays a correct data. For instance, when the RBS bank is selected in the preference list, only the RBS data is displayed, Figure 34 the image on the top. The preference setting is changed to the best result and checked if the application displays what is expected which includes the best result on the first line, Figure 34 the image on the top right side. The other two banks Barclays and x-Rate are similarly checked in the preference list.



**Figure 34: Represents the CER app user interface of all selected banks**

- The image on the top on the right is when RBS is selected .
- The image on the top on the left is when the best result is selected. Notice that the best result is displayed on the first line.
- The image on the bottom on the right is when the x-Rate is selected.
- The image on the bottom on the left is when the Barclays is selected.

## 8.4 Testing Other Activity Classes

Throughout the implementation stage, I tested and debugged each method. For example, when the `btnAbout()` method is implemented, which opens a new layout activity, the CER app is run via the emulator and the About button is tested several times to ensure the system opens a new activity. In the same way, the other activity classes are tested. The CER application is yet to be tested on the actual Android device.

## 9. Conclusions

In the final chapter, I discuss the overall success of the CER app project and further recommendations.

Throughout the implementation of the CER app, all functionalities mentioned in the initial BSc project proposal were tested and successfully delivered. The CER app offers to the user the choice of the most favourable exchange rate as provided by three different market sources.

The implementation of the CER app has been successfully achieved. The original requirements, which were written in the initial BSc project proposal, Currency Exchange Rate - Android application, are satisfactory and the features in the CER app are as per requirements. The design of the CER application has been completed and the application was unit tested as well as tested via the emulator.

### 9.1 Enhancements

There are several ways how the CER app could be enhanced. In the following three paragraphs I discuss the main problems which could be implemented for further enhancements of the application.

#### 1) Non—functional improvement

The CER app downloads data from three different websites. To download all data require approximately 5 minutes. The Validator class cleans downloaded data which follows a certain pattern. If the provider of the website changes the HTML structure of the HTML source, the data may not be validated and the database will never be updated.

The suggestion to this issue is to create a database, e.g. in SQL Server, or PHP website. The PHP website would communicate with three different banks' websites, download and validate data. The PHP website would then store downloaded data into the database.

The middleware would be performed to communicate between the database and the application as presented in Figure 35. The middleware would manage a queue if several applications would require downloading data at the same time and it would ensure that 'dirty' data would not be read.

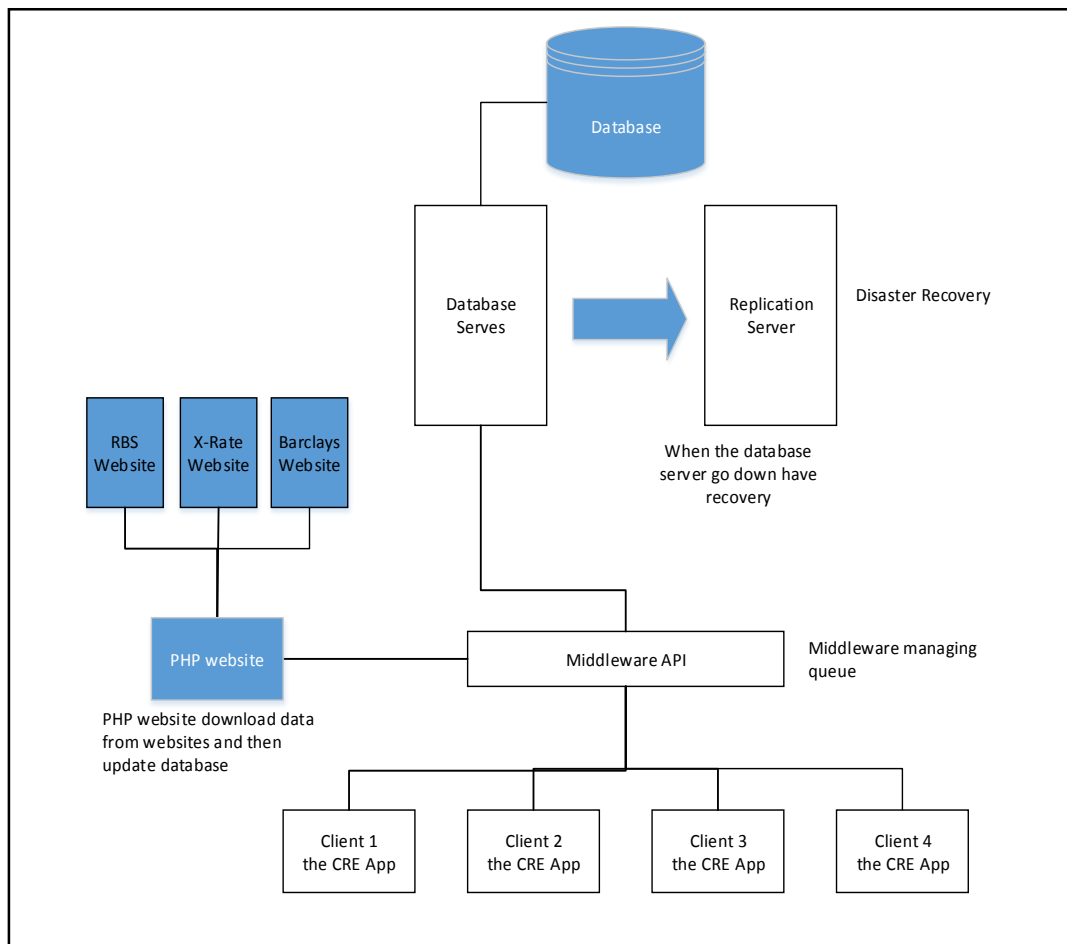


Figure 35: Represent conceptual enterprise diagram for downloading data from a database in the CER apps.

By downloading the data into the CER app from the database, instead of web-servers, the amount of time would be reduced significantly. The issue of clearing data would be omitted as data would be checked before it is inserted into the database.

In addition, the SQL server database would be able to hold a huge amount of data. The CER app can also be extended to other functionality such as graph representations of certain currencies for a certain period of time, or displaying history tables of any currencies.

## 2) Supporting Different Screen Sizes

The Android platform runs on several devices such as smart phones or tablets. Each device has a different screen size. The CER app is adaptable for middle and small size screens. The main layout could be extended for large and extra-large screens. This could be done by creating two extra folders, called layout-large and layout-xlarge in the res folder. Inside these two folders, the activity\_main.xml file would describe layouts properties for the particular screen. In the same way, layout-xlarge-land could be created that would describe the layout for extra-large screen size when the screen is in the landscape orientation.

### **3) Different Platforms**

The CER app runs only on the Android platform, but there are other major players on the market such as iOS, Windows or BlackBerry. Therefore, the last suggestion is that the CER app could be built for other platforms.

## Appendix A - Android Issues

In this appendix, I present the main problems that occurred throughout the development of the Android CER app. Firstly, I show how to ensure that the data is in the right format and is validated before it is archived in the database. Furthermore, I discuss the Android technology environment and its rapid changes that required to do additional research for the most current updates. I then present the challenge of calculating currency conversion followed by presenting what is meant by the best rating offer.

### 1 Validation Data Form Web Server Issue

To download a data into the application from the web server is done by using HTTP, refer to Chapter 7.1 Downloading Data from Web Server. When the HTTP string is downloaded from the web server, the data contains HTML elements that consist of tags enclosed in angle brackets, such as (<html> or <table>). The problem which arises is that the data needs to be cleared such that a string contains only currencies' name and currencies' value.

Once the data has been cleared, and the string contains only currency name and corresponding values, the data needs to be validated. The reason for validating data is the following: if the HTTP of any bank server would be changed, then the clear data may contain a different value which is not required and, therefore, the validation system is required to ensure that the currency's name contains only the currency name and the currency's value contains only numbers.

The solution for this issue was found by debugging HTTP data. The data string is split into an array by using regular expression `split("<>")`, shown in Figure A.1 on the second line. By using for loop, each value in the array is accessed. When "td" is found, the next index's value of the array contains either currency name or currency value, which is saved into a string variable called data and data string is then returned back.

```

private String getCurrency(String urlStr){
    String[] urlArr = urlStr.split("<>");
    String data = "";
    int count = 0;
    for(int i = 0; i < urlArr.length; i++){
        if(urlArr[i].equals("Travellers Cheques"))
            break;
        if(urlArr[i].equals("td")){
            data = data + urlArr[i + 1] + ", ";
            count++;
            if(count == 3){
                count = 0;
                data = data + "\n";
            }
        }
    }
    return data;
}

```

**Figure A.1:** Describes method that clear HTML source code of the RBS website to find out the currency's name and the currency's value.

To ensure that the data is in a correct format the validateData() method is implemented that contains a regular expression to check if the data is in the correct format, displayed in Figure A.2.

```

private boolean validateData(String name, String value1, String value2){
    final String DigitsRegExp = "^\\d+\\.\\d{1,5}|^\\d+";
    final String LettersRegExp = "[a-zA-Z]+|[a-zA-Z]+\\&[a-zA-Z]+\\;[a-zA-Z]+";
    return name.replaceAll("\\s+", "").matches(LettersRegExp) &&
           value1.replaceAll("\\s+", "").matches(DigitsRegExp) &&
           value2.replaceAll("\\s+", "").matches(DigitsRegExp);
}

```

**Figure A.2:** Represents method that checks data if they are in the correct format where string name represent currency's name and value 1 & 2 represent currency's value.

## 2 Using Old Android Implementation

Other issues appear in the research development. When I was going through the book's exercises, some of methods that were described in the book were out of the date. That is, some imported classes did not use the methods that were described in the books. Below are described two of those methods which are required to be used for the CER app.

### Retrieving Data from SQLite database

The books recommendation of retrieving data from the SQLite was by using a method startManageCursor(), which is the method from imported class [SQLiteDatabase](#). However, when the method was implemented into the Android project, it gets cut through as is shown on Figure A.3 with an error message; "The method startManageCustor(cursor) is deprecated".

The method is deprecated because it does operations on the main thread which can freeze up the UI and deliver a poor user experience. The Solution was found in Android [developer](#), (2014) <sup>R.6</sup>. Instead of using startManagingCursor() method, I use the following methods: moveToFirst(), isAfterLast() and moveToNext().

```
private Cursor getEvents() {  
    // Perform a managed query. The Activity will handle closing  
    // and re-querying the cursor when needed.  
    SQLiteDatabase db = events.getReadableDatabase();  
    Cursor cursor = db.query(TABLE_NAME, FROM, null, null, null,  
        null, ORDER_BY);  
    startManagingCursor(cursor) //  
    return cursor;  
}
```

Figure A.3.

The database query output is inside a cursor object that needs to be moved to the first row position by using the moveToFirst() method. Then a while loop is implemented which goes through all rows inside the cursor object. The loop condition uses isAfterLast() which returns boolean condition and ensures that the cursor points to the position after the last row.

By using the moveToFirst() method, the cursor is in the first row 'position'. To retrieve a data from the first position of the cursor object, I use getString(int) and getDouble(int) methods with passing integer as a parameter that represents the column's position. Zero is the first column, one is the second column and so on. The getString(int) method returns the value of the requested column in that particular row as a string. I then use the moveToNext() method, which move from one position to another position and checks while loop condition again. This is going on until the cursor is moved to the last position, see Figure A.4.

```
String allTuples = "";  
cursor.moveToFirst();  
while(!cursor.isAfterLast()){  
    allTuples = allTuples + " , " + cursor.getString(0) + " , " + cursor.getString(1) + "\n ";  
    cursor.moveToNext();  
}  
cursor.close();  
return allTuples;
```

Figure A.4: Represent java code that manage to retrieve data from Cursor object.



## Displaying Preference.xml file

Another books' recommendations were to use a preference for storing data into the SharedPreferences file by using a super-class PreferenceActivity that inherit a method addPreferencesFromResource(), which loads the preference.xml file. However, when these steps were implemented into the activity, the addPreferencesFromResource() method gets cut through, as shown in Figure A.5, with an error message "The method addPreferencesFromResource(int) from the type PreferenceActivity is deprecated".

```
public class Prefs extends PreferenceActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        addPreferencesFromResource(R.xml.settings);  
    }  
}
```

Figure A.5.

The addPreferencesFromResource() method was deprecated from the inherit class PreferenceActivity. The solution was found in [stack overflow](#)<sup>R.13</sup> and it occurs that instead of inheriting class PreferenceActivity, the PreferenceFragment class needs to be inherited. In addition, the PreferenceFragment class is supported for the app in API Level 11 and higher<sup>15</sup>. If the app uses API level 10 or below the PreferenceActivity class needs to be used for managing the load preference .xml file.

### 3 Calculate Rate Exchange

The data conversion is stored in the database for the British pound, for example, 1 British pound for USD or 1 British pound for Euro. The problem arises when none of the spinner's value is selected to the British pound, for instance, USD to Euro. How to get currency conversion 1 USD to the euro, if it is known that £1.00 is \$1.51 and £1.00 is €1.31?

First step is to divide British pound by a USD to get 1 USD to British pound and to divide the British pound by Euro to get 1 Euro to British pound, "£1.00/\$1.51 and £1.00/€1.31". Then the result of 1 British pound to USD is divided by the result of 1 Euro to British pound. With this calculation, the calculator gets the conversion of 1 USD to Euro:

---

<sup>15</sup> "API Level is an integer value that uniquely identifies the framework API revision offered by a version of the Android platform. The Android platform provides a framework API that applications can use to interact with the underlying Android system. ([developer](#), 2015<sup>R.6</sup>)".

	Pound	US Dollar	EURO		Pound	US Dollar	EURO
Pound	£1.00	£0.662252	£0.763359	Pound	£1.00	£0.662252	£0.763359
US Dollar	\$1.51	\$1	?	US Dollar	\$1.51	\$1.00	\$1.152672
EURO	€1.31	?	€1.00	EURO	€1.31	€0.86755	€1.00
(£/\$)/(£/€) is 1 US dollar to Euro		$(1/1.51)/(1/1.31) = 0.662252/0.763352 = 0.86755$		\$1.00 = €0.86755			
(£/€)/(£/\$) is 1 EURO to US dollar		$(1/1.31)/(1/1.51) = 0.763352/0.662252 = 1.152672$		€1.00 = \$1.152672			

**Table 2: Describes calculation of the conversion 1 USD to Euro and 1 Euro to USD from British pound.**

However, I found a better and faster way to get the result of the conversion 1 USD to Euro. Table 3 shows the calculation.

One Pound to	Division	Conversion
€ 1.31	$€1.31/\$1.51 = 0.86755$	$\$1.00 = €0.86755$
\$ 1.51	$\$1.51/€1.31 = 1.152672$	$€1.00 = \$1.152672$

**Table 3: Describes simple calculation of the conversion of 1 USD to Euro and 1 Euro to USD from British pound.**

Finally, to calculate the user input, Euro is divided by USD and the result is multiplied by the input. For instance, if the input is 1500 USD to Euro then Euro is divided by USD  $€1.31/\$1.51 = \$0.86755$  and the result is multiplied by the input  $0.86755 * 1500 = 1301.325$ . So 1500 USD is equal to 1301.325 Euro.

#### 4 Best Rate

When the best rate is selected in the preference section, the CER app displays the best rate. The issue is what to consider as the best rate, whether when 1 British pound is 1.51 USD in RBS or when 1 British pound is 1.53 USD in Barclays.

RBS bank has a lower offer, it seems that the RBS bank offers the best rate. However, after considering that the user wants for their money as much as they can get. The user exchanges 1000 British pound in RBS and gets 1510.00 USD, but if the user exchange 1000 British pounds in Barclays, he/she gets 1530.00 USD. At the end the user gets 20 USD more in Barclays bank than in RBS bank, and therefore, the best result is selected in the bank which has the greater exchange rate.

## Appendix B – Java Classes

The program consists with 17 classes and 2 interface classes. This chapter presents Java codes of the each program's class. The first class is the main Activity class called CurrenciesRateExchangeMainActivity. The rest of Java classes are listed in an alphabetical order. The last two classes are Interface classes.

### CurrenciesRateExchangeMainActivity.java

```
package org.example.currenciesrateexchange;

import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;

import android.app.Activity;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.app.TaskStackBuilder;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.res.Configuration;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.AsyncTask;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.text.Html;
import android.util.Log;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.WindowManager;
import android.view.inputmethod.InputMethodManager;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

/**
 * CurrenciesRateExchangeMainActivity class is master or driver class that
 * communicate with other classes
 * When the application gets active the first method that is executed is onCreate().
 * Class enable to download data from the web servers and send them to validate
 * them and then store them into database.
 * Class fill spinner with values (currency name) and display
 * selected spinners value with rate.
 * Class is also able to deal with user input when the button is pressed class.
 *
 * @author dsajdl01 (David Sajdl)

```

```

* @version (01/03/2015)
*/

public class CurrenciesRateExchangeMainActivity extends Activity {

    // initialize instance variables
    private Spinner spinner_from, spinner_to;
    private EditText amount;
    private TextView rateView, txtPreference, output;
    private Button btn_preference, btn_about, btn_used;
    private File dtb;
    private String valueFrom, valueTo, preference, currentDate;
    private boolean finalUpdatedAllData, dataRBSIsUpdated, dataXRateIsUpdated,
        dataBarclaysUpdated, isXrateDataExist, dateIsUpdated, isBarclaysDataExist;
    private Calendar c;
    private ValidateRBSdata rbsData;
    private DataSource dataSource;
    private RBSDataSource rbsSource;
    private XRateSource xrateSource;
    private BarclaysDataSource barclaysSource;
    private CalculateRate rbsCr1 = null;
    private CalculateRate xrateCr1 = null;
    private CalculateRate barclaysCr1 = null;
    private NotificationManager mNotificationManager;
    private int notificationID = 1500;
    private int numMessages = 0;
    private InputMethodManager imm;
    /**
     * onCreate() method "main method" is the first method that is called
     * when the application gets active
     * Firstly it calls inherit method follow by loading xml file
     * called activity_main.xml layout.
     * Then give a value to instance variables and check if database exist.
     * If database exist it opens database as writable
     * Finally method calls the other methods. One to load a data
     * from web server by downloadRBSData().
     * Other one to add value into spinner by addValuesIntoSpinner()
     * and the last method that is called is to get stare data
     * from preferences by reloadPreferences();
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        dtb = getApplicationContext().getDatabasePath("bank_rate.db");
        rbsSource = new RBSDataSource(this);
        dataSource = new DataSource(this);
        xrateSource = new XRateSource(this);
        barclaysSource = new BarclaysDataSource(this);
        c = new Calendar();
        valueFrom = null;
        valueTo = null;
        currentDate = null;
        dataRBSIsUpdated = false;
        dataXRateIsUpdated = false;
        dataBarclaysUpdated = false;
        isXrateDataExist = false;
        isBarclaysDataExist = false;
        dateIsUpdated = false;
        finalUpdatedAllData = false;
        preference = "0";

        imm = (InputMethodManager) getSystemService(Context.INPUT_METHOD_SERVICE);

```

```

spinner_from = (Spinner) findViewById(R.id.spinner_From);
spinner_to = (Spinner) findViewById(R.id.spinner_to);
txtPreference = (TextView) findViewById(R.id.txt_preference);
rateView = (TextView) findViewById(R.id.txtview_inputRate);
output = (TextView) findViewById(R.id.txtview_output_user_result);
btn_preference = (Button) findViewById(R.id.btn_press);
btn_about = (Button) findViewById(R.id.btn_about);
btn_used = (Button) findViewById(R.id.btn_used);
btn_about.setGravity(Gravity.CENTER);
btn_used.setGravity(Gravity.CENTER);

amount = (EditText) findViewById(R.id.txtEdit_amount);
amount.setFocusable(false);

/**
 * setOnClickListener() inner class or Interface definition for a call-back
 * to be invoked when a edit text is clicked.
 */
amount.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // set focus true
        amount.requestFocus();
        amount.setFocusableInTouchMode(true);
        // display keyboard
        imm.showSoftInput(amount, InputMethodManager.SHOW_IMPLICIT);
        getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_VISIBLE);
    }
});

amount.setRawInputType(Configuration.KEYBOARD_12KEY);
// check if database exist if yes open database
if(dtb.exists()){
    barclaysSource.open();
    rbsSource.open();
    dateSource.open();
    xrateSource.open();
    barclaysSource.open();
    // check if x-Rate data was inserted
    if(xrateSource.getProfilesCount() > 1){
        isXrateDataExist = true;
    }
    // check if Barclays data was inserted
    if(barclaysSource.getProfilesCount() > 1){
        isBarclaysDataExist = true;
    }
}

downloadRBSData();
addValuesIntoSpinner();
reloadPreferences();
}

/**
 * onResume() methods is called when the activity start interacting with the user
 * Method sets data updated as false and send message to download data and
 * reload preference
 */
@Override
public void onResume(){
    super.onResume();
    dataRBSIsUpdated = false;
    dataXRateIsUpdated = false;
    dataBarclaysUpdated = false;
    dateIsUpdated = false;
}

```

```

        finalUpdatedAllData = false;
        downloadRBSData();
        reloadPreferences();
    }

    /**
     * addValuesIntoSpinner() method allows to read a data from array call
     * countries_name in string xml file
     * and add arrays value into spinners. Method also has inner class
     * setOnItemSelectedListener,
     * when the spinner change selected value setOnItemSelectedListener
     * get execute and send
     * message to the method call updateRateViewFromSpinner() to update for current value
     */
    private void addValuesIntoSpinner(){
        // loading data from string array into spinner that currency have
        ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(
            this, R.array.countries_name, android.R.layout.simple_spinner_item);
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        spinner_from.setAdapter(adapter);
        // loading data from string array into spinner that currency want
        ArrayAdapter<CharSequence> adapterTwo = ArrayAdapter.createFromResource(this,
            R.array.countries_name_two, android.R.layout.simple_spinner_item);
        adapterTwo.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        spinner_to.setAdapter(adapterTwo);

        //listener when spinner is move to update value
        spinner_from.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            public void onItemSelected(AdapterView<?> parent, View view, int pos, long id) {
                Object item = parent.getItemAtPosition(pos);
                setValueFrom(String.valueOf(item));
                updateRateViewFromSpinner();
            }
            public void onNothingSelected(AdapterView<?> parent) {
            }
        });
        //listener when spinner is move to update value
        spinner_to.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            public void onItemSelected(AdapterView<?> parent, View view, int pos, long id) {
                Object item = parent.getItemAtPosition(pos);
                setValueTo(String.valueOf(item));
                updateRateViewFromSpinner();
            }
            public void onNothingSelected(AdapterView<?> parent) {
            }
        });
    }

    /**
     * setValueFrom() private method sets currency name into instance
     * variable called valueFrom
     * that represent currency name that user has.
     */
    private void setValueFrom(String name){
        this.valueFrom = name;
    }

    /**
     * setValueTo private method sets currency name into instance variable called valueTo
     * that represent currency name that user wants.
     */
    private void setValueTo(String name){
        this.valueTo = name;
    }

    /**

```

```

* updateRateViewFromSpinner() private method update text view.
* The value of the view is the value
* of the spinners with currency rates.
* Method check is database exist if yes it update view by comparing
* spinner value and preference value,
* if the database does not exist it displays message that "Device needs
* to be connected to the Internet!"
*/
private void updateRateViewFromSpinner(){
    try{
        // check if database exist
        if(dtb.exists()){
            // get currencies name that are stored in database
            if(valueFrom == null || valueTo == null) addValuesIntoSpinner();
            SwitchCurrencyName scn = new SwitchCurrencyName();
            String rbsCurrencyName1 = scn.switchRBSCurrencyName(valueFrom);
            String rbsCurrencyName2 = scn.switchRBSCurrencyName(valueTo);
            String xrateCurName = scn.switchXRateCurrencyName(valueFrom);
            String xrateCurName2 = scn.switchXRateCurrencyName(valueTo);
            String barclaysName1 = scn.switchBarclaysCurrencyName(valueFrom);
            String barclaysName2 = scn.switchBarclaysCurrencyName(valueTo);
            // display message if database using current data
            if(finalUpdatedAllData){
                // update database with current date
                if(!dateIsUpdated){
                    currentDate = c.getCurrentDate();
                    dateSource.updateDateValue(currentDate);
                    dateIsUpdated = true;
                }
                // displayTopToast("Database using current update");
            }else{
                // display message which date data was last time updated
                String lastUpdate = dateSource.getStoredDate();
                displayTopToast("Database using data from: " + lastUpdate);
            }
            // if the best result is selected

            if(preference.equals("0")){
                // get currency value
                rateView.setText(preference.toString() + " preference");
                Double rbsHave = getRBSCurrencyValueFromSQLite(rbsCurrencyName1);
                Double rbsWant = getRBSCurrencyValueFromSQLite(rbsCurrencyName2);
                Double xrateHave = getXRateCurrencyValueFromSQLite(xrateCurName);
                Double xrateWant = getXRateCurrencyValueFromSQLite(xrateCurName2);
                Double barclaysHave = getBarclaysCurrencyValueFromSQLite(barclaysName1);
                Double barclaysWant = getBarclaysCurrencyValueFromSQLite(barclaysName2);
                rateView.setText(preference.toString() + " preference 1..");
                barclaysCr1 = new CalculateRate(barclaysHave,barclaysWant);
                rbsCr1 = new CalculateRate(rbsHave,rbsWant);
                xrateCr1 = new CalculateRate(xrateHave,xrateWant);
                // display the best result
                rateView.setText(preference.toString() + " preference comming here");
                printBestResult(rbsCr1.getRate(),xrateCr1.getRate(),barclaysCr1.getRate());
            }
            // if RBS is selected
            else if (preference.equals("1")){
                // get RBS value
                Double rbsHave = getRBSCurrencyValueFromSQLite(rbsCurrencyName1);
                Double rbsWant = getRBSCurrencyValueFromSQLite(rbsCurrencyName2);
                rbsCr1 = new CalculateRate(rbsHave,rbsWant);

                // display RBS value
                rateView.setText( Html.fromHtml("<b><i>&nbsp; RBS offers:</i></b>&nbsp;");

```

```

        &nbsp; 1 " + valueFrom.substring(0,3) + " = " + String.format( "%.4f",
        rbsCrl.getRate()) + " " + valueTo.substring(0,3));
    }
    // if xRate is selected
    else if(preference.equals("2")){
        // get xRate value
        Double xrateHave = getXRateCurrencyValueFromSQLite(xrateCurName);
        Double xrateWant = getXRateCurrencyValueFromSQLite(xrateCurName2);
        xrateCrl = new CalculateRate(xrateHave,xrateWant);
        // display xRate value
        rateView.setText(Html.fromHtml("<b><i>&nbsp; x-Rate offers:</i></b>&nbsp;
        &nbsp; 1 " + valueFrom.substring(0,3) + " = " + String.format(
        "%.4f", xrateCrl.getRate()) + " " + valueTo.substring(0,3));
    }
    else if(preference.equals("3")){
        Double barclaysHave = getBarclaysCurrencyValueFromSQLite(barclaysName1);
        Double barclaysWant = getBarclaysCurrencyValueFromSQLite(barclaysName2);
        barclaysCrl = new CalculateRate(barclaysHave,barclaysWant);
        rateView.setText(Html.fromHtml("<b><i>&nbsp; Barclays
        offers:</i></b>&nbsp; &nbsp; 1 " + valueFrom.substring(0,3) + " = "
        + String.format( "%.4f", barclaysCrl.getRate()) + " " +
        valueTo.substring(0,3));
    }
}

// check if database is updates if not download data
if(!dataRBSIsUpdated) downloadRBSData();
if(dataRBSIsUpdated && !dataXRateIsUpdated) downloadXRateData();
if(dataRBSIsUpdated && dataXRateIsUpdated && !dataBarclaysUpdated)
    downloadBarclaysData();
}
else{
    // if database does not exist display message
    displayBottomToast("Device needs to be conected to the Internet to load a data!
    \nPlease make sure that Android divide has a connection to the Internet.");
}
} catch(Exception e){
    Log.d("DatabaseDemo ", e.toString());
}
}

/**
 * printBestResult() private method compare currency rate value
 * from the bank RBS, Barclays and x-Rate
 * and display values in ascending order. The rate value with higher
 * rates it appears on the first line.
 * Then second higher rate value is displayed on the second line and on the
 * third line is displayed the lower rate value.
 *
 * @param rbs double that represent currency rate from the selected currency name inside
the spinners.
 * @param xrate double that represent currency rate from the selected currency name inside
the spinners.
 * @param barclays double that represent currency rate from the selected currency name
inside the spinners.
 */
private void printBestResult(double rbs, double xRate, double barclays){
    if(rbs < xRate){
        if(rbs < barclays){
            if(xRate < barclays){
                rateView.setText( Html.fromHtml("<p><b><i>&nbsp; 1)&nbsp; Barclays
                offers: </i></b>&nbsp; 1 " + valueFrom.substring(0,3) + " = " +
                String.format( "%.4f", barclays) + " " + valueTo.substring(0,3)

```



```

        + "</p><p><b><i>&nbsp;   2)&nbsp;   x-Rate offers: </i></b>&nbsp;   1 " +
        valueFrom.substring(0,3) + " = " + String.format( "%.4f", xRate) +
        " " + valueTo.substring(0,3) + "</p><b><i>&nbsp;   3)&nbsp;   RBS offers:
        </i></b>&nbsp;   1 " + valueFrom.substring(0,3) + " = " + String.format(
        "%.4f", rbs) + " " + valueTo.substring(0,3));
    } else{
        rateView.setText(Html.fromHtml( "<p><b><i>&nbsp;   1)&nbsp;   x-Rate
        offers: </i></b>&nbsp;   1 " + valueFrom.substring(0,3) + " = " +
        String.format( "%.4f", xRate) + " " + valueTo.substring(0,3)
        + "<p><p><b><i>&nbsp;   2)&nbsp;   Barclays offers: </i></b>&nbsp;   1 "
        + valueFrom.substring(0,3) + " = " + String.format( "%.4f",
        barclays) + " " + valueTo.substring(0,3)+"<p><b><i>&nbsp;   3) &nbsp;  
        RBS offers: </i></b>&nbsp;   1 " + valueFrom.substring(0,3) + " = " +
        String.format( "%.4f", rbs) + " " + valueTo.substring(0,3));
    }
} else if(barclays < xRate){
    rateView.setText(Html.fromHtml("<p><b><i>&nbsp;   1)&nbsp;   x-Rate offers:
    </i></b>&nbsp;   1 " + valueFrom.substring(0,3) + " = " +
    String.format( "%.4f", xRate) + " " + valueTo.substring(0,3) +
    "<p><p><b><i>&nbsp;   2)&nbsp;   RBS offers: </i></b>&nbsp;   1 " +
    valueFrom.substring(0,3) + " = " + String.format( "%.4f", rbs) + " "
    + valueTo.substring(0,3)+"<p><b><i>&nbsp;   3) &nbsp;   Barclays offers:
    </i></b>&nbsp;   1 " + valueFrom.substring(0,3) + " = " +
    String.format( "%.4f", barclays) + " " + valueTo.substring(0,3));
}
}
else if (barclays < xRate){
    rateView.setText(Html.fromHtml("<p><b><i>&nbsp;   1)&nbsp;   RBS offers:
    </i></b>&nbsp;   1 " + valueFrom.substring(0,3) + " = " + String.format(
    "%.4f", rbs) + " " + valueTo.substring(0,3)+"<p><p><b><i>&nbsp;   2)&nbsp;  
    x-Rate offers: </i></b>&nbsp;   1 " + valueFrom.substring(0,3) + " = " +
    String.format( "%.4f", xRate) + " " + valueTo.substring(0,3)+
    "<p><b><i>&nbsp;   3) &nbsp;   Barclays offers: </i></b>&nbsp;   1 " +
    valueFrom.substring(0,3) + " = " + String.format( "%.4f", barclays) + " "
    + valueTo.substring(0,3));
} else if(rbs < barclays){
    rateView.setText(Html.fromHtml("<p><b><i>&nbsp;   1)&nbsp;   Barclays offers:
    </i></b>&nbsp;   1 " + valueFrom.substring(0,3) + " = " + String.format(
    "%.4f", barclays) + " " + valueTo.substring(0,3)+ "<p><p><b><i>&nbsp;  
    2)&nbsp;   RBS offers: </i></b>&nbsp;   1 " + valueFrom.substring(0,3) + " = "
    + String.format( "%.4f", rbs) + " " + valueTo.substring(0,3)+
    "<p><b><i>&nbsp;   3) &nbsp;   x-Rate offers: </i></b>&nbsp;   1 " +
    valueFrom.substring(0,3) + " = " + String.format( "%.4f", xRate) + " " +
    valueTo.substring(0,3));
}
} else{
    rateView.setText(Html.fromHtml("<p><b><i>&nbsp;   1)&nbsp;   RBS offers: </i></b>&nbsp;  
    1 " + valueFrom.substring(0,3) + " = " + String.format( "%.4f", rbs) + " "
    + valueTo.substring(0,3)+ "<p><p><b><i>&nbsp;   2)&nbsp;   Barclays offers:
    </i></b>&nbsp;   1 " + valueFrom.substring(0,3) + " = " + String.format(
    "%.4f", barclays) + " " + valueTo.substring(0,3)+"<p><b><i>&nbsp;   3) &nbsp;  
    x-Rate offers: </i></b>&nbsp;   1 " + valueFrom.substring(0,3) + " = " +
    String.format( "%.4f", xRate) + " " + valueTo.substring(0,3));
}
}
/**
 * getBarclaysCurrencyValueFromSQLite() private method retrieve Barclays currency value from
 the database.
 *
 * @param String that represent currency name that value would be retrieved.
 * @param double that represent currency value.
 */
private double getBarclaysCurrencyValueFromSQLite(String curName){
    if(curName.equals("Euro")) return 1.0;
    return barclaysSource.getCurrencyValue(curName);
}

```

```

/**
 * getRBSCurrencyValufromSqlite() method retrieve RBS currency's value from the database.
 *
 * @param String that represent currency name that value would be retrieved.
 * @param double that represent currency value.
 */
private double getRBSCurrencyValueFromSQLite(String curName){
    if(curName.equals("GBP - BRITISH POUND")) return 1.0;
    return rbsSource.getCurrencyRBSValue(curName);
}
/**
 * getXRateCurrencyValufromSqlite() method retrieves xRate currency's value from the
database.
 *
 * @param String that represent currency name that value would be retrieved.
 * @param double that represent currency value.
 */
private double getXRateCurrencyValueFromSQLite(String curName){
    if(curName.equals("GBP - BRITISH POUND")) return 1.0;
    return xrateSource.getCurrencyValue(curName);
}
/**
 * updateBarclaysDatabase() method validates Barclays data and
 * it either creates a new database and store data or it updates database.
 *
 * @param String that represent html downloaded data from Barclays server
 */
public void updateBarclaysDatabase(String data){
    try{
        // checking if database is already update if not
        if(!dataBarclaysUpdated){
            ValidateBarclaysData barclays = new ValidateBarclaysData(data);
            // checking if data are validate if yes
            if(barclays.getValid()){
                String[] barArray = barclays.getData();
                // check if Barclays database table has value if not insert values
                if(!isBarclaysDataExist){
                    barclaysSource.open();
                    for(int i = 0; i < barArray.length - 1; i = i + 2){
                        double value = Double.parseDouble(barArray[i+1]);
                        barclaysSource.addValueIntoTable(barArray[i], value);
                    }
                    dataBarclaysUpdated = true;
                    isBarclaysDataExist = true;
                    finalUpdatedAllData = true;
                    updateNotification("Database has been just created & all data is
                        inserted.", "Database using current update from " + c.getCurrentDate());
                    // if Barclays table has values then update values
                } else {
                    for(int i = 0; i < barArray.length - 1; i = i + 2){
                        double value = Double.parseDouble(barArray[i+1]);
                        barclaysSource.updateTable(barArray[i], value);
                    }
                    dataBarclaysUpdated = true;
                    finalUpdatedAllData = true;
                    // display that database is updated
                    updateNotification("All database tables are updated.", "Database using
                        current update from " + c.getCurrentDate());
                }
            }

            // if data is not validated display message
        } else {
            displayBottomToast(barclays.getErrorMessage());
        }
    }
}

```

```

    }
} catch (Exception e){
    Log.d("DatabaseDemo ", e.toString());
}
}

/**
 * updateXRateDatabase() method validates xRate data and either create new
 * database and store data or update database if database exist
 *
 * @param String that represent html downloaded data from xRate server
 */
public void updateXRateDatabase(String data){
    try{
        // check if database is updated if not then
        if(!dataXRateIsUpdated){
            // get validate data
            ValidateXRateData xRate = new ValidateXRateData(data);
            // if data is validated update or store data
            if(xRate.getValid()){
                String[] xrateArr = xRate.getData();
                // check if database exist if not open & create database and add data to database
                if(!isXrateDataExist){
                    xrateSource.open();
                    for(int i = 0; i < xrateArr.length-1; i = i + 2){
                        double value = Double.parseDouble(xrateArr[i + 1]);
                        xrateSource.addValueIntoTable(xrateArr[i], value);
                    }
                    dataXRateIsUpdated = true;
                    isXrateDataExist = true; }
                else{
                    // if database exist update database
                    for(int j = 0; j < xrateArr.length-1; j = j + 2){
                        double value = Double.parseDouble(xrateArr[j + 1]);
                        xrateSource.updateTable(xrateArr[j], value);
                    }
                    dataXRateIsUpdated = true;
                }
            }
        } else{
            //if data is not validated display message
            displayBottomToast(xRate.getErrorMessage());
        }
    } catch (Exception e){
        Log.d("DatabaseDemo ", e.toString());
    }
}

/**
 * updateRBSDatabase() method validates RBS data and either create a new database and store
 * data
 * or update database if database exist
 *
 * @param String that represent html downloaded data from RBS server
 */
public void updateRBSDatabase(String data){
    try{
        // checking if database is updated if not then update database
        if(!dataRBSIsUpdated){
            // get validate data
            rbsData = new ValidateRBSdata(data);
            // if data is validated then
            if(rbsData.getValid()){
                String[] rbsArrData = rbsData.getData();
                // checking if database exist? if does not:
                if(!dtb.exists()){
                    //it creates and opens database then add value into database

```

```

        rbsSource.open();
        dateSource.open();
        for(int i = 0; i < rbsArrData.length - 1; i = i + 3){
            double buy = Double.parseDouble(rbsArrData[i + 1]);
            double sell = Double.parseDouble(rbsArrData[i + 2]);
            rbsSource.addRBSValuesIntoTable(rbsArrData[i], buy, sell);
        }
        currentDate = c.getCurrentDate();
        dateSource.addDate(currentDate);
        dataRBSIsUpdated = true;
    }
    else{
        // if database exist then update data
        for(int i = 0; i < rbsArrData.length - 1; i = i + 3){
            double buy = Double.parseDouble(rbsArrData[i + 1]);
            double sell = Double.parseDouble(rbsArrData[i + 2]);
            rbsSource.updateRBSValues(rbsArrData[i], buy, sell);
        }
        dataRBSIsUpdated = true;
    }
}
else{
    // if data is not validated
    displayBottomToast(rbsData.getErrorMessage());
}
}
}
}
catch (Exception e){
    Log.d("DatabaseDemo ", e.toString());
}
}
/**
 * btnToUse() method is executed when the How To Use button is pressed
 * and method starts new activity (Used Class).
 */
public void btnToUse(View v){
    Intent i = new Intent(this, Used.class);
    startActivity(i);
}
/**
 * btnAbout() method is executed when the About button is pressed
 * and method starts new activity (About Class).
 */
public void btnAbout(View v){
    Intent i = new Intent(this, About.class);
    startActivity(i);
}
/**
 * getData() method is execute when = 'equal' button is pressed and
 * method checks if database exist if yes; then method checks if value was insert into edit
text.
 * if value is insert then method process value and display result if value
 * was not insert does not do anything. If database does not exist
 * it display message that database does not exist.
 */
public void getData(View v){
    try{
        // checking if database exist if yes;
        if(dtb.exists()){
            // getting input data from edit text
            String strInput = amount.getText().toString().trim();
            // if value is not entered

```









```

    }
}

/**
 * reloadPreferences() method retrieves data from preference resource
 * and the method is executed when the application is called
 */
private void reloadPreferences(){
    SharedPreferences sp = PreferenceManager.getDefaultSharedPreferences(this);
    preference = sp.getString("dispalRate","");
    btn_preference.setText("Setting Exchange Preference");
    btn_preference.setGravity(Gravity.CENTER);
    // checking which preference is selected
    // if the best preference is selected display the best result
    if(preference.equals("0")){
        txtPreference.setGravity(Gravity.CENTER);
        txtPreference.setText(" Display \nBest Result");
        output.setText( Html.fromHtml("<b><i>&nbsp; RBS has for:</b></i><br>&nbsp;
        <br><b><i>&nbsp; x-Rate has for:</b></i><br>&nbsp; <br><b><i>&nbsp; Barclays
        has for:</b></i><br>&nbsp;"));
    }
    // if RBS preference is selected display RBS
    else if(preference.equals("1")){
        txtPreference.setGravity(Gravity.CENTER);
        txtPreference.setText("Display\n RBS ");
        output.setText( Html.fromHtml("<b><i>&nbsp; RBS has for:</b></i><br>&nbsp; " ));
    }
    // if xRate preference is selected display xRate
    else if(preference.equals("2")){
        txtPreference.setGravity(Gravity.CENTER);
        txtPreference.setText("Display\nx-Rate");
        output.setText( Html.fromHtml("<b><i>&nbsp; x-Rate has for:</b></i><br>&nbsp;"));
    }
    // if Barclays preference is selected display Barclays
    else if(preference.equals("3")){
        txtPreference.setGravity(Gravity.CENTER);
        txtPreference.setText("Display\nBarclays");
        output.setText( Html.fromHtml("<b><i>&nbsp; Barclays has for:</b>
        </i><br>&nbsp;"));
    }
    else{
        displayBottomToast("Preference was not selected.");
    }
}

/**
 * setPreferences method is executed when the button Setting Exchange Preference is pressed
 * It called class SetPreferenceActivity that show preference activity form
 */
public void setPreferences(View v){
    Intent i = new Intent();
    i.setClass(this, SetPreferenceActivity.class);
    startActivityForResult(i,0);
}

/**
 * onActivityResult() method is override from Activity class and
 * method is called when new activity is closed and going back to main activity
 * Method is call when preference activity in closed and go back
 * to main activity and it sends message to reload preference
 */
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    reloadPreferences();
}

```



```

/**
 * onPause() method is called when the current "main" activity is being paused
 * this happen when some other activity start overpaying the current "main" activity
 */
@Override
protected void onPause() {
    super.onPause();
    setResult(RESULT_OK, new Intent(this, CurrenciesRateExchangeMainActivity.class));
}
/**
 * displayTopToast() method displays pop up message on the top of the screen which
 * will be display approximately for 20 second.
 *
 * @param string message that represent a message that will be displayed
 */
public void displayTopToast(String message){
    LayoutInflater inflater = getLayoutInflater();
    View layout = inflater.inflate(R.layout.toast_layout, (ViewGroup)
                                findViewById(R.id.toast_layout_root));

    TextView text = (TextView) layout.findViewById(R.id.text);
    text.setText(message);
    Toast toast = new Toast(getApplicationContext());
    toast.setGravity(Gravity.TOP|Gravity.CENTER, 0, 49);
    toast.setDuration(Toast.LENGTH_LONG);
    toast.setView(layout);
    toast.show();
}
/**
 * displayBottomToast() method displays pop up message on the bottom of the screen which
 * will be display approximately for 20 second.
 *
 * @param string message that represent a message that will be displayed
 */
public void displayBottomToast(String message){
    LayoutInflater inflater = getLayoutInflater();
    View layout = inflater.inflate(R.layout.toast_layout, (ViewGroup)
                                findViewById(R.id.toast_layout_root));

    TextView text = (TextView) layout.findViewById(R.id.text);
    text.setText(message);
    Toast toast = new Toast(getApplicationContext());
    toast.setGravity(Gravity.BOTTOM|Gravity.CENTER, 0, 0);
    toast.setDuration(Toast.LENGTH_LONG);
    toast.setView(layout);
    toast.show();
}

/**
 * downloadBarclysData() method checks if the connection with network is available
 * if yes it starts loading data by calling DownloadBarclysWebPageTask
 * inner class that have Barclays url as the argument
 * if no, it displays message that network in not available
 */
public void downloadBarclysData(){
    if(isNetworkAvailable()){
        DownloadBarclysWebPageTask task = new DownloadBarclysWebPageTask();
        task.execute(new String[] { "https://www.barclayscorporate.com/foreign-exchange-
                                   rates.html" });
    }
    else{
        displayBottomToast("NETWORK IS NOT AVAILABLE");
    }
}

```

```

/**
 * downloadXRateData() method checks if the connection with network is available
 * if yes it starts loading data by calling DownloadWebPageTask inner class that have xRate
url as the argument
 * if no, it displays message that network in not available
 */
public void downloadXRateData() {
    if(isNetworkAvailable()){
        DownloadWebPageTask task = new DownloadWebPageTask();
        task.execute(new String[] { "http://www.x-rates.com/table/?from=GBP" });
    }
    else{
        displayBottomToast("NETWORK IS NOT AVAILABLE");
    }
}

/**
 * downloadRBSData() method checks if the connection with network is available
 * if yes it starts loading data by calling ReadStreamTask inner class that have RBS url
argument
 * if no, it displays message that network in not available
 */
public void downloadRBSData(){
    if(isNetworkAvailable()){
        new ReadStreamTask().execute("http://www.rbs.co.uk/personal/travel/g1/money/
exchange-rates.ashx");
    }
    else{
        displayBottomToast("NETWORK IS NOT AVAILABLE");
    }
}

/**
 * isNetworkAvailable() check if android device is connected to the Internet
 *
 * @return boolean condition that is either true if there is a connection
 * available or false there is not connection
 */
private boolean isNetworkAvailable(){
    boolean available = false;
    ConnectivityManager mng =
        (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo netInfo = mng.getActiveNetworkInfo();
    if(netInfo != null && netInfo.isAvailable()){
        available = true;
    }
    return available;
}

/**
 * readStream method takes an URL as argument and return HTML text context
 *
 * @param urlStr string url from where data will be loaded
 * @return string HTML text content
 * @throws IOException if instance of HttpURLConnection class is not delivered by method
 * openConnection it throws exception
 */

private String readStream(String urlStr) throws IOException{
    String str = "";
    InputStream inputstr = null;
    BufferedReader reader = null;
    try{
        URL url = new URL(urlStr);
        HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();
        inputstr = urlConn.getInputStream();

```

```

        reader = new BufferedReader(new InputStreamReader(inputstr));
        String line = "";
        while((line = reader.readLine()) != null){
            str +=line;
        }
    }catch(Exception ex){
        Log.d("Networking demo", ex.toString());
    } finally{
        inputstr.close();
        reader.close();
    }
    return str;
}

/**
 * displayNotification() method enable to create notification
 * and display small notes on the top of the screen
 *
 * @param string message that represent small note on the top of the screen
 * @param string message2 that represent text in the notification
 */
protected void displayNotification(String message, String message2) {
    Log.i("Start", "notification");

    /* Invoking the default notification service */
    Notification.Builder mBuilder = new Notification.Builder(this);
    mBuilder.setSmallIcon(R.drawable.Logo_notification).setTicker(message); //.setWhen(0);
    mBuilder.setAutoCancel(false).setContentTitle("Loading Data Info:");
    mBuilder.setStyle(new Notification.BigTextStyle().bigText(message2));

    /* Increase notification number every time a new notification arrives */
    mBuilder.setNumber(++numMessages);

    /* Creates an explicit intent for an Activity in your app */
    Intent resultIntent = new Intent(this, NotificationView.class);

    TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
    stackBuilder.addParentStack(NotificationView.class);

    /* Adds the Intent that starts the Activity to the top of the stack */
    stackBuilder.addNextIntent(resultIntent);
    PendingIntent resultPendingIntent = stackBuilder.getPendingIntent(0,
        PendingIntent.FLAG_UPDATE_CURRENT);

    mBuilder.setContentIntent(resultPendingIntent);

    mNotificationManager =
        (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);

    /* notificationID allows you to update the notification later on. */
    mNotificationManager.notify(notificationID, mBuilder.build());
}

/**
 * updateNotification() method enable to update notification text
 * and pop up small note on the top of the screen
 *
 * @param string message that represent small note on the top of the screen.
 * @param string message2 that represent text in the notification
 */
protected void updateNotification(String message, String message2) {
    Log.i("Update", "notification");

    /* Invoking the default notification service */
    Notification.Builder mBuilder = new Notification.Builder(this);

    mBuilder.setSmallIcon(R.drawable.Logo_notification).setTicker(message); //.setWhen(0);

```

```

mBuilder.setContentTitle("Loading Data Info:");
mBuilder.setStyle(new Notification.BigTextStyle().bigText(message2));

/* Increase notification number every time a new notification arrives */
mBuilder.setNumber(++numMessages);

/* Creates an explicit intent for an Activity in your app */
Intent resultIntent = new Intent(this, NotificationView.class);

TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
stackBuilder.addParentStack(NotificationView.class);

/* Adds the Intent that starts the Activity to the top of the stack */
stackBuilder.addNextIntent(resultIntent);
PendingIntent resultPendingIntent = stackBuilder.getPendingIntent(0,
    PendingIntent.FLAG_UPDATE_CURRENT);

mBuilder.setContentIntent(resultPendingIntent);

mNotificationManager =
    (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);

/* Update the existing notification using same notification ID */
mNotificationManager.notify(notificationID, mBuilder.build());
}
/**
 * cancelNotification() method destroyed or cancel notification
 */
protected void cancelNotification() {
    Log.i("Cancel", "notification");
    mNotificationManager.cancel(notificationID);
}

/**
 * ReadStreamTask class or inner class is a subclass of AsyncTask class
 * to not allowed to perform network operations in a UI thread.
 * Class allows to download data from the web server.
 */
private class ReadStreamTask extends AsyncTask<String, Void, String>{
    String str = "";
    /**
     * onPreExecute() method is execute before data is loaded.
     */
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        // while RBS data is downloading do not send to download data again
        dataRBSIsUpdated = true;
        displayNotification("Downloading RBS data...", "RBS data just being loaded.");
    }

    /**
     * doInBackground method download data from the server
     */

    @Override
    protected String doInBackground(String... params) {
        // TODO Auto-generated method stub
        try{
            str = readStream(params[0]);
        }catch(Exception ex){
            // if downloading fail then try to download data again
            dataRBSIsUpdated = false;
            Log.d("NetworkingDemo ", ex.toString());
        }
    }
}

```

```

        return str;
    }

    /**
     * onPostExecute method receives web server data and sent data to update database.
     */
    @Override
    protected void onPostExecute(String result) {
        cancelNotification();
        // before RBS data is validated set dataRBSIsUpdated to false
        dataRBSIsUpdated = false;
        updateRBSDatabase(result.toString());
        downloadXRateData();
    }
}

/**
 * DownloadBarclaysWebPageTask class or inner class is a subclass of AsyncTask class
 * Class allows to download Barclays data from the web server.
 */
private class DownloadBarclaysWebPageTask extends AsyncTask<String, Void, String> {
    /**
     * onPreExecute() method is execute before data is loaded.
     */
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        // while Barclays data is downloading set dataBarclaysUpdated to true
        // to not sending download data again
        dataBarclaysUpdated = true;
        displayNotification("Downloading Barclays data...", "RBS and x-Rate Data is
            downloaded and database has been updated \nBarclays data just being
            downloaded. \nTo download all data may take a couple second.
            Please wait...");
    }

    /**
     * doInBackground method download data from the server
     */
    @Override
    protected String doInBackground(String... urls) {
        String response = "";
        for (String url : urls) {
            DefaultHttpClient client = new DefaultHttpClient();
            HttpGet httpGet = new HttpGet(url);
            try {
                HttpResponse execute = client.execute(httpGet);
                InputStream content = execute.getEntity().getContent();
                BufferedReader buffer = new BufferedReader(
                    new InputStreamReader(content));

                String s = "";
                while ((s = buffer.readLine()) != null) {
                    response += s;
                }
            } catch (Exception e) {
                // if downloading data fail set dataBarclaysUpdated to false to download data again.
                dataBarclaysUpdated = false;
                e.printStackTrace();
            }
        }
        return response;
    }

    /**
     * onPostExecute method receives web server data and sent data to update database.
     */
    @Override

```

```

protected void onPostExecute(String result) {
    cancelNotification();
    // before the data is validated set dataBarclaysUpdated to fail
    dataBarclaysUpdated = false;
    updateBarclaysDatabase(result.toString());
}
}

/**
 * DownloadBarclaysWebPageTask class or inner class is a subclass of AsyncTask class
 * Class allows to download barclays data from the web server.
 */
private class DownloadWebPageTask extends AsyncTask<String, Void, String> {
    /**
     * onPreExecute() method is execute before data is loaded.
     */
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        //while data is downloading do not send message to download data again.
        dataXRateIsUpdated = true;
        displayNotification("Downloading x-Rate data...", "RBS Data is downloaded \n
            x-Rate data just being loaded. \nTo download all data
            may take a couple second.");
    }
    /**
     * doInBackground method download data from the server
     */
    @Override
    protected String doInBackground(String... urls) {
        String response = "";
        for (String url : urls) {
            DefaultHttpClient client = new DefaultHttpClient();
            HttpGet httpGet = new HttpGet(url);
            try {
                HttpResponse execute = client.execute(httpGet);
                InputStream content = execute.getEntity().getContent();

                BufferedReader buffer = new BufferedReader(
                    new InputStreamReader(content));
                String s = "";
                while ((s = buffer.readLine()) != null) {
                    response += s;
                }
            } catch (Exception e) {
                //if downloading fail set dataXRateIsUpdated to fail to try load data again
                dataXRateIsUpdated = false;
                e.printStackTrace();
            }
        }
        return response;
    }

    /**
     * onPostExecute method receives web server data and sent data to update database.
     */
    @Override
    protected void onPostExecute(String result) {
        cancelNotification();
        // before the data is validated set dataXRateUpdate to fail
        dataXRateIsUpdated = false;
        updateXRateDatabase(result.toString());
        downloadBarclaysData();
    }
}
}

```

## About.java

```
package org.example.currenciesrateexchange;

import android.app.Activity;
import android.os.Bundle;
import android.view.WindowManager;

/**
 * About class enable to open new xml layout activity called about.xml
 * where layout is set with exact width and height.
 *
 * @author dsajdl01 (David Sajdl)
 * @version (01/03/2015)
 */
public class About extends Activity {

    /**
     * onCreate() method allows to open new xml layout called about.xml
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.about);

        WindowManager.LayoutParams params = getWindow().getAttributes();
        params.x = 0;
        params.height = 380;
        params.width = 270;
        params.y = 25;

        this.getWindow().setAttributes(params);
    }
}
```

## BarclaysDataSource.java

```
package org.example.currenciesrateexchange;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;

/**
 * BarclaysDataSource class is a middle class or communication class between database tables
 * called TABLE_BARCLAYS_NAME and CurrenciesRateExchangeMainActivity class
 * Class enable to open and close database
 * create new rows into table update table and retrieve data from the table
 *
 * @author dsajdl01 (David Sajdl)
 * @version (01/14/2015)
 */
public class BarclaysDataSource implements Source {

    //declaring instance variables
    private SQLiteDatabase database;
    private MySQLiteHelper dbHelper;

    /**
     * constructor that takes Context object as argument
     *
     * @param Context that in object of the calls CurrenciesRateExchangeMainActivity
     */
}
```

```

public BarclaysDataSource(Context context){
    dbHelper = new MySQLiteHelper(context);
}

/**
 * open method opens database
 * if method does not manage to open database it throws SQLException
 *
 * @throws SQLException if does not manage to open database
 */
@Override
public void open() throws SQLException{
    database = dbHelper.getReadableDatabase();
}

/**
 * close method closes database
 */
@Override
public void close(){
    dbHelper.close();
}

/**
 * addValuesIntoTable() method increase arguments values into
 * database table called TABLE_BARCLAYS_NAME
 *
 * @param currencyName string currency name (country)
 * @param value double the value of currency
 */
@Override
public void addValueIntoTable(String currencyName, Double value) {
    // TODO Auto-generated method stub
    ContentValues val= new ContentValues();
    currencyName = currencyName.trim();
    val.put(MySQLiteHelper.COLUMN_BARCLAYS_CURRENCY_NAME, currencyName);
    val.put(MySQLiteHelper.COLUMN_BARCLAYS_VALUE, value);
    database.insert(MySQLiteHelper.TABLE_BARCLAYS_NAME, null, val);
}

/**
 * updateTable() methods manages to update database table called TABLE_BARCLAYS_NAME
 * Method updates only currency value by using WHERE cause where COLUMN_CURRENCY_NAME
 * currency name
 *
 * @param currencyName string currency name (country)
 * @param value double the value of currency
 */
@Override
public void updateTable(String currencyName, Double value) {
    // TODO Auto-generated method stub
    currencyName = "" +currencyName.trim() +"";
    ContentValues val = new ContentValues();
    val.put(MySQLiteHelper.COLUMN_BARCLAYS_VALUE, value);
    database.update(MySQLiteHelper.TABLE_BARCLAYS_NAME, val,
        MySQLiteHelper.COLUMN_BARCLAYS_CURRENCY_NAME + "=" + currencyName, null);
}

/**
 * getCurrencyValue() method enables to retrieve currency' value where currency
 * name is equal to currency' name which is passed by the parameter
 *
 * @return double that represent currency rate value to Euro
 */
@Override
public double getCurrencyValue(String name) {
    // TODO Auto-generated method stub
    Cursor cr = database.rawQuery("select * from " + MySQLiteHelper.TABLE_BARCLAYS_NAME

```



```

        + " where " + MySQLiteHelper.COLUMN_BARCLAYS_CURRENCY_NAME + " = " +
        "" + name + "", null);
    cr.moveToFirst();
    return cr.getDouble(2);
}
/**
 * getProfilesCount() method enables to count row in the database table
 * called TABLE_BARCLAYS_NAME
 *
 * @return integer that represent number of row in the table called TABLE_BARCLAYS_NAME
 */
public int getProfilesCount() {
    String countQuery = "SELECT * FROM " + MySQLiteHelper.TABLE_BARCLAYS_NAME;
    Cursor cursor = database.rawQuery(countQuery, null);
    int cnt = cursor.getCount();
    cursor.close();
    return cnt;
}

//*****
//**          METHODS BELOW DO NOT NEED IT FOR CER APP          **//
//**          However result of the method was print it out to check it if all          **//
//**          data is inserted and then updated.          **//
//*****
/**
 * getAllTableTuples method loads all tuples value from the table
 * called TABLE_BARCLAYS_NAME into string and return it
 *
 * @return string allTuples that hold all values from the table called TABLE_BARCLAYS_NAME
 */

public String getAllTuples(){
    String[] allColumns =
        {MySQLiteHelper.COLUMN_BARCLAYS_ID,MySQLiteHelper.COLUMN_BARCLAYS_CURRENCY_NAME,
        MySQLiteHelper.COLUMN_BARCLAYS_VALUE};
    String allTuples = "Database Values\n\n";
    Cursor cr = database.query(MySQLiteHelper.TABLE_BARCLAYS_NAME, allColumns, null,
        null, null, null, null);
    cr.moveToFirst();
    while(!cr.isAfterLast()){
        allTuples = allTuples + getRowValues(cr);
        cr.moveToNext();
    }
    cr.close();
    return allTuples;
}
/**
 * getRowValues() private method convert Cursor object into string
 * Cursor is output of the database queries that is casting into string
 *
 * @return string that contains currency's id, currency's name and currency's value
 */
private String getRowValues(Cursor cr){
    long id = (cr.getLong(0));
    String name = (cr.getString(1));
    Double value = (cr.getDouble(2));
    return " " + id + ". " + name + ", " + value + "\n";
}
}

```

## CalculateRate.java

```
package org.example.currenciesrateexchange;

/**
 * CalculateRate class calculate currency rate exchange.
 *
 * @author dsajdl01 (David Sajdl)
 * @version (01/14/2015)
 */
public class CalculateRate {
    //declaring instance variables
    private Double have;
    private Double want;
    /**
     * Constructor that takes two double as argument
     * that represent currency that user has e.g. US Dollar and
     * currency that user wants e.g. EURO
     *
     * @param double have that represent currency that user has e.g. US Dollar
     * @param double want that represent currency that user wants e.g. EURO
     */
    public CalculateRate(Double have, Double want){
        this.have = have;
        this.want = want;
    }
    /**
     * getHave() method returns currency that user has
     *
     * @return double currency that user has
     */
    public Double getHave(){
        return have;
    }
    /**
     * getWant() method returns currency that user wants
     *
     * @return currency that user wants
     */
    public Double getWant(){
        return want;
    }
    /**
     * getRate method calculates rate that user would get for 1 currency
     * that has to currency user wants e.g. how much EURO gets for 1 US Dollar.
     *
     * @return double rates to currency has to currency wants
     */
    public Double getRate(){
        return want/have;
    }
    /**
     * calculateUserInput() method calculate how much user gets for money that user has
     *
     * @param double that represents amount of currency that user has.
     * @return double that represent amount of currency that user gets.
     */
    public Double calculateUserInput(double input){
        return (want/have) * input;
    }
}
```

## Calendar.java

```
package org.example.currenciesrateexchange;
// import java classes
import android.annotation.SuppressLint;
import java.util.Date;
import java.text.SimpleDateFormat;
/**
 * Calendar class represent current day
 * Calendar class is used to get current day when that
 * database is updated to keep record of last updated database.
 *
 * @author dsajdl01 (David Sajdl)
 * @version (01/14/2015)
 */
public class Calendar {
    /**
     * getCurrentDate() methods gets and returns current day and hour
     *
     * @return string current day in format year, month, day, hour, minutes and second.
     */
    @SuppressWarnings("SimpleDateFormat")
    public String getCurrentDate(){
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        return sdf.format(new Date());
    }
}
```

## DataSource.java

```
package org.example.currenciesrateexchange;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
/**
 * DataSource class is a middle class or communication class between database tables
 * called TABLE_DATE_NAME and CurrenciesRateExchangeMainActivity class
 * Class enable to open and close database
 * insert data into table, update table and retrieve data from the table.
 *
 * @author dsajdl01 (David Sajdl)
 * @version (01/14/2015)
 */
public class DataSource {
    //declaring instance variables
    private SQLiteDatabase database;
    private MySQLiteHelper dbHelper;
    private String[] timeColumn = {MySQLiteHelper.COLUMN_DATE_ID,
                                    MySQLiteHelper.COLUMN_DATE,};

    /**
     * Constructor that has one argument called Context object
     *
     * @param Context that in object of the calls CurrenciesRateExchangeMainActivity
     */
    public DataSource(Context context){
        dbHelper = new MySQLiteHelper(context);
    }
    /**
     * open() method opens database if method does not manage to open database
     * it throws SQLException
     */
}
```

```

*
* @throws SQLException if does not manage to open database or find database
*/
public void open() throws SQLException{
    database = dbHelper.getReadableDatabase();
}
/**
* close() method closes database
*/
public void close(){
    dbHelper.close();
}
/**
* addDate() method allows to add current data into database table
* called TABLE_DATE_NAME
*
* @param string date that represents current day
*/
public void addDate(String date){
    ContentValues values = new ContentValues();
    values.put(MySQLiteHelper.COLUMN_DATE, date);
    database.insert(MySQLiteHelper.TABLE_DATE_NAME, null, values);
}

/**
* updateDateValue() method manages to update database table called TABLE_DATE_NAME
* In the table 'TABLE_DATE_NAME' is just one row therefore only the first row is update
* by using WHERE cause where COLUMN_DATE_ID is equal to 1.
*
* @param string date that represents current day
*
*/
public void updateDateValue(String date){
    ContentValues val = new ContentValues();
    val.put(MySQLiteHelper.COLUMN_DATE, date);
    database.update(MySQLiteHelper.TABLE_DATE_NAME, val,
        MySQLiteHelper.COLUMN_DATE_ID + "=" + "'1'", null);
}
/**
* getStoredDate() method manages to get data from database table
* called TABLE_DATE_NAME and return them as a string
* In the table 'TABLE_DATE_NAME' is just one row therefore only the first is retrieved
*
* @return String that represents day, which day was whole database updated
*/
public String getStoredDate(){
    Cursor crs = database.query(MySQLiteHelper.TABLE_DATE_NAME,
        timeColumn, null, null, null, null, null);

    crs.moveToFirst();
    return (crs.getString(1));
}
/**
* isOpen() method check if database table called TABLE_DATE_NAME is open
*
* @return boolean either true if database table is open or false if database table is
                                close
*/
public boolean isOpen(){
    return database.isOpen();
}
}

```

## MySQLiteHelper.java

```
package org.example.currenciesrateexchange;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;
/**
 * MySQLiteHelper class manages to create database called bank_rate and
 * tables called TABLE_DATE_NAME, TABLE_RBS_NAME, TABLE_XRATE_NAME and TABLE_BARCLAYS_NAME
 *
 * @author dsajdl01 (David Sajdl)
 * @version (01/14/2015)
 */
public class MySQLiteHelper extends SQLiteOpenHelper{

    // Constant that represent database
    protected static final String DATABASE_NAME = "bank_rate.db";
    private static final int DATABASE_VERSION = 1;

    // Constants that represent table informations for RBS
    protected static final String TABLE_RBS_NAME = "rbs_data";
    protected static final String COLUMN_ID = "id";
    protected static final String COLUMN_CURRENCY_NAME = "currency_name";
    protected static final String COLUMN_BUYS = "buy";
    protected static final String COLUMN_SELLS = "sell";

    // table creation as sql statement for RBS table
    private static final String DATABASE_CREATE_RBS = "create table " + TABLE_RBS_NAME +
        "(" + COLUMN_ID + " integer primary key autoincrement, " +
        COLUMN_CURRENCY_NAME + " text not null, " + COLUMN_BUYS + " REAL, " +
        COLUMN_SELLS + " REAL)";

    // Constants that represent table informations for date
    protected static final String TABLE_DATE_NAME = "DATE";
    protected static final String COLUMN_DATE_ID = "date_id";
    protected static final String COLUMN_DATE = "date";
    // table creation as SQL statement for date table
    private static final String DATABASE_CREATE_DATE = "create table " + TABLE_DATE_NAME +
        "(" + COLUMN_DATE_ID + " integer primary key autoincrement, " + COLUMN_DATE +
        " text)";

    // Constants that represent table informations for xRate
    protected static final String TABLE_XRATE_NAME = "xrate_data";
    protected static final String COLUMN_XRATE_ID = "xrate_id";
    protected static final String COLUMN_XRATE_CURRENCY_NAME = "xrate_name";
    protected static final String COLUMN_XRATE_VALUE = "xrate_value";
    // table creation as SQL statement for xRate table
    private static final String DATABASE_CREATE_XRATE = "create table " + TABLE_XRATE_NAME
        + "(" + COLUMN_XRATE_ID + " integer primary key autoincrement, " +
        COLUMN_XRATE_CURRENCY_NAME + " text not null, " + COLUMN_XRATE_VALUE + "
        REAL)";

    // Constants that represent table informations for Barclays
    protected static final String TABLE_BARCLAYS_NAME = "barclays_data";
    protected static final String COLUMN_BARCLAYS_ID = "barclays_id";
    protected static final String COLUMN_BARCLAYS_CURRENCY_NAME = "barclays_name";
    protected static final String COLUMN_BARCLAYS_VALUE = "barclays_value";
    // table creation as SQL statement for barclays table
    private static final String DATABASE_CREATE_BARCLAYS = "create table " +
        TABLE_BARCLAYS_NAME + "(" + COLUMN_BARCLAYS_ID + " integer primary key
        autoincrement, " + COLUMN_BARCLAYS_CURRENCY_NAME + " text not null, " +
        COLUMN_BARCLAYS_VALUE + " REAL)";
```

```

/**
 * Constructor that has one argument called Context object
 *
 * @param Context that in object of the calls CurrenciesRateExchangeMainActivity
 */
public MySQLiteHelper(Context context){
    super(context, DATABASE_NAME, null, DATABASE_VERSION);
}

/**
 * onCreate() method create database and their tables.
 * However if the database isn't created, it will create a new database
 * if database is already created then it check the current database version against
 * supplied version. if both version matches nothing happens.
 */
@Override
public void onCreate(SQLiteDatabase db) {
    // TODO Auto-generated method stub
    db.execSQL(DATABASE_CREATE_RBS);
    db.execSQL(DATABASE_CREATE_DATE);
    db.execSQL(DATABASE_CREATE_XRATE);
    db.execSQL(DATABASE_CREATE_BARCLAYS);
}

/**
 * onUpgrade method handle the upgrade logic for database schema
 */
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // TODO Auto-generated method stub
    Log.w(MySQLiteHelper.class.getName(), "Upgrading database from version " +
        oldVersion + " to " + newVersion + ", which destroy all data");
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_RBS_NAME);
    Log.w(MySQLiteHelper.class.getName(), "Upgrading database from version " +
        oldVersion + " to " + newVersion + ", which destroy all data");
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_DATE_NAME);
    Log.w(MySQLiteHelper.class.getName(), "Upgrading database from version " +
        oldVersion + " to " + newVersion + ", which destroy all data");
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_XRATE_NAME);
    Log.w(MySQLiteHelper.class.getName(), "Upgrading database from version " +
        oldVersion + " to " + newVersion + ", which destroy all data");
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_BARCLAYS_NAME);
    onCreate(db);
}
}

```

## NotificationView.java

```

package org.example.currenciesrateexchange;
import android.os.Bundle;
import android.app.Activity;
/**
 * NotificationView class manages to open notification xml layout
 * that could be display on the top of the main layout
 *
 * @author dsajdl01 (David Sajdl)
 * @version (01/14/2015)
 */
public class NotificationView extends Activity{

    /**
     * onCreate method is the first method that is called when the class is called
     * Firstly it calls inherit method follow by loading xml file called notification
     */
}

```

```

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

```

## RBSDDataSource.java

```

package org.example.currenciesrateexchange;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
/**
 * RBSDDataSource class is a middle class or communication class between database tables
 * called TABLE_RBS_NAME and CurrenciesRateExchangeMainActivity class
 * Class enable to open and close database
 * create new rows into table update table and retrieve data from the table
 *
 * @author dsajdl01 (David Sajdl)
 * @version (01/14/2015)
 */
public class RBSDDataSource {

    //declaring instance variables
    private SQLiteDatabase database;
    private MySQLiteHelper dbHelper;
    private String[] allColumns =
        {MySQLiteHelper.COLUMN_ID,MySQLiteHelper.COLUMN_CURRENCY_NAME,MySQLiteHelper.COLUMN
        _BUYS,MySQLiteHelper.COLUMN_SELLS};
    /**
     * constructor that takes Context object as argument
     *
     * @param Context that in object of the calls CurrenciesRateExchangeMainActivity
     */
    public RBSDDataSource(Context context){
        dbHelper = new MySQLiteHelper(context);
    }

    /**
     * open() method opens database if method does not manage to open database
     * it throws SQLException or if the database cannot be found
     *
     * @throws SQLException if does not manage to open database
     */
    public void open() throws SQLException{
        database = dbHelper.getReadableDatabase();
    }

    /**
     * close() method closes database
     */
    public void close(){
        dbHelper.close();
    }

    /**
     * addRBSValuesIntoTable() method increase arguments values into database table
     * called TABLE_RBS_NAME
     *
     * @param currencyName string currency name (country)
     * @param buys double the value of currency that is buy
     * @param sells double the value of the currency that is sell
     */
}

```

```

public void addRBSValuesIntoTable(String currencyName, double buys, double sells){
    ContentValues value = new ContentValues();
    currencyName = currencyName.trim();
    value.put(MySQLiteHelper.COLUMN_CURRENCY_NAME, currencyName);
    value.put(MySQLiteHelper.COLUMN_BUYS, buys);
    value.put(MySQLiteHelper.COLUMN_SELLS, sells);
    database.insert(MySQLiteHelper.TABLE_RBS_NAME, null, value);
}
/**
 * updateRBSValues() methods manages to update database table called TABLE_RBS_NAME
 * updateRBSValues updates only currency value buy and sell by using
 * cause WHERE COLUMN_CURRENCY_NAME = currency name
 *
 * @param currencyName string currency name (country)
 * @param buys double the value of currency that is being buy
 * @param sells double the value of the currency that is being sell
 */
public void updateRBSValues(String currencyName, double buys, double sells){
    currencyName = "" + currencyName.trim() + "";
    ContentValues value = new ContentValues();
    value.put(MySQLiteHelper.COLUMN_BUYS, buys);
    value.put(MySQLiteHelper.COLUMN_SELLS, sells);
    database.update(MySQLiteHelper.TABLE_RBS_NAME, value,
        MySQLiteHelper.COLUMN_CURRENCY_NAME + "=" + currencyName, null);
}
/**
 * getCurrencyRBSValue() method manages to retrieve currency value that is equal to
 * currency's name that is passed through parameter
 *
 * @return double that is average of currency buy and sell
 */
public double getCurrencyRBSValue(String name){
    Cursor cr = database.rawQuery("select * from " + MySQLiteHelper.TABLE_RBS_NAME
        + " where " + MySQLiteHelper.COLUMN_CURRENCY_NAME + " = " + "'" + name +
        "'", null);
    cr.moveToFirst();
    double buy = cr.getDouble(2);
    double sell = cr.getDouble(3);
    return (buy+sell)/2;
}

/**
 * getProfilesCount() method manage to count how many rows are in the table
 * called TABLE_RBS_NAME
 *
 * @return integer the represents numbers of rows in the table called TABLE_RBS_NAME
 */
public int getProfilesCount(){
    String countQuery = "SELECT * FROM " + MySQLiteHelper.TABLE_RBS_NAME;
    Cursor cursor = database.rawQuery(countQuery, null);
    int cnt = cursor.getCount();
    cursor.close();
    return cnt;
}
//*****
/**
 * METHODS BELOW DO NOT NEED IT FOR CER APP
 * however result of the method was print it out to check it if all data
 * is inserted and then updated.
 */
//*****
/**
 * getAllTableTuples method loads all tuples value from the table
 * called TABLE_RBS_NAME into string and return it
 *
 * @return string allTuples that hold all values from the table called TABLE_RBS_NAME
 */

```



```

public String getAllTableTuples(){
    String allTuples = "Database Values\n\n";
    Cursor cr = database.query(MySQLiteHelper.TABLE_RBS_NAME, allColumns,
                                null, null, null, null, null);

    cr.moveToFirst();
    while(!cr.isAfterLast()){
        allTuples = allTuples + getRowValues(cr);
        cr.moveToNext();
    }
    cr.close();
    return allTuples;
}
/**
 * getRowValues() private method manages to cast object 'Cursor' into string
 * Cursor object holds row value that are converted into string and then return it.
 *
 * @param cursor object that holds rows values from database table
 *                                called TABLE_RBS_NAME.
 * @return string that hold currency's id, currency's name currency's
 *                                buy and currency's sell
 */
private String getRowValues(Cursor cursor){
    long id = (cursor.getLong(0));
    String name = (cursor.getString(1));
    Double buy = (cursor.getDouble(2));
    Double sell = (cursor.getDouble(3));
    return id + ". " + name + ", " + buy + ", " + sell + "\n";
}
}

```

## SetPreference.java

```

package org.example.currenciesrateexchange;
import android.content.Intent;
import android.os.Bundle;
import android.preference.Preference;
import android.preference.PreferenceFragment;
/**
 * SetPreference class manages to open preferences xml layout that
 * is displayed on the top of the main layout (activity_main xml)
 *
 * @author dsajdl01 (David Sajdl)
 * @version (01/14/2015)
 */
public class SetPreference extends PreferenceFragment{
    /**
     * onCreate() method is the first method that is called when SetPreference is called
     * Firstly it calls inherit method follow by loading xml file called activity_main
     * onCreate method also has on button listener and when the button back is pressed
     * setOnPreferenceClickListener class gets executed that
     * returns back to CurrenciesRateExchangeMainActivity class or to main layout
     */
    /**
     public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         // Load the preferences from an XML resource
         addPreferencesFromResource(R.xml.preferences);
         //on button listener
         Preference button = (Preference)getPreferenceManager().findPreference("button");
         if (button != null) {
             button.setOnPreferenceClickListener(new Preference.OnPreferenceClickListener(){
                 @Override
                 public boolean onPreferenceClick(Preference arg0){
                     Intent intent = new Intent(getActivity(),

```

```

        CurrenciesRateExchangeMainActivity.class);
        startActivity(intent);
        return false;
    }
    });
}
}
}

```

## SetPreferenceActivity.java

```

package org.example.currenciesrateexchange;
import android.app.Activity;
import android.os.Bundle;
/**
 * SetPreferenceAcivity class manages sending message SetPreference class
 *
 * @author dsajdl01 (David Sajdl)
 * @version (01/14/2015)
 */
public class SetPreferenceActivity extends Activity{
    /**
     * onCreate() method is the first method that is called when SetPreference is called
     * Firstly it calls inherit method follow by calling SetPreference class
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        getFragmentManager().beginTransaction().replace(android.R.id.content,
            new SetPreference()).commit();
    }
}

```

## SwithCurrencyName.java

```

package org.example.currenciesrateexchange;
/**
 * SwitchCurrencyName class enable switching currency Name that is inside spinner
 * to currency name that is inside database tables
 *
 * @author dsajdl01 (David Sajdl)
 * @version (01/14/2015)
 */
public class SwitchCurrencyName {
    /**
     * switchBarclaysCurrencyName() method converts currency name that is inside spinner
     * to currency name that is inside database table called TABLE_BARCLAYS_NAME.
     *
     * @param string currencyName that represent currency name inside spinners.
     * @return string that represent currency name inside database
     *         table TABLE_BARCLAYS_NAME.
     */
    public String switchBarclaysCurrencyName(String currencyName){
        if(currencyName.equals("GBP - British Pound")) return "British Pound";
        else if(currencyName.equals("EUR - EURO")) return "Euro";
        else if(currencyName.equals("USD - US Dollar")) return "US Dollar";
        else if(currencyName.equals("AUD - Australian Dollar")) return "Australian Dollar";
        else if(currencyName.equals("BHD - Bahrain Dinar")) return "Bahraini Dinar";
        else if(currencyName.equals("BBD - Barbados Dollar")) return "Barbados Dollar";
        else if(currencyName.equals("BRL - Brazilian Real")) return "Brazilian Real";
    }
}

```

```

else if(currencyName.equals("BND - Brunei Ringgit")) return "";
else if(currencyName.equals("BGN - Bulgarian Lev")) return "Bulgarian Lev";
else if(currencyName.equals("CAD - Canadian Dollar")) return "Canadian Dollar";
else if(currencyName.equals("KYD - Cayman Island Dollar")) return "";
else if(currencyName.equals("CLP - Chilean Peso"))return "Chilean Peso";
else if(currencyName.equals("CNY - Chinese Yuan")) return "Chinese Renminbi";
else if(currencyName.equals("CRC - Costa Rican Colon")) return "";
else if(currencyName.equals("HRK - Croatian Kuna")) return "Croatian Kuna";
else if(currencyName.equals("CZK - Czech Koruna")) return "Czech Koruna";
else if(currencyName.equals("DKK - Danish Kroner")) return "Danish Krona";
else if(currencyName.equals("DOP - Dominican Rep. Peso")) return "";
else if(currencyName.equals("XCD - East Caribbean Dollar"))
    return "East Carribbean Dollar";
else if(currencyName.equals("EGP - Egyptian Pound")) return "Egyptian Pound";
else if(currencyName.equals("FJD - Fiji Dollar")) return "";
else if(currencyName.equals("HKD - Hong Kong Dollar")) return "Hong Kong Dollar";
else if(currencyName.equals("HUF - Hungarian Forint")) return "Hungarian Forint";
else if(currencyName.equals("ISK - Icelandic Krona")) return "Iceland Krona";
else if(currencyName.equals("IDR - Indonesian Rupiah")) return "Indian Rupee";
else if(currencyName.equals("ILS - Israeli New Sheqel")) return "Israeli Shekel";
else if(currencyName.equals("JMD - Jamaican Dollar")) return "Jamaican Dollar";
else if(currencyName.equals("JPY - Japanese Yen")) return "Japanese Yen";
else if(currencyName.equals("JOD - Jordanian Dinar")) return "Jordanian Dinar";
else if(currencyName.equals("KES - Kenyan Shilling")) return "Kenyan Shilling";
else if(currencyName.equals("KWD - Kuwaiti Dinar")) return "Kuwaiti Dinar";
else if(currencyName.equals("LBP - Lebanon Pound")) return "";
else if(currencyName.equals("MYR - Malaysian Ringgit")) return "Malaysian Ringgit";
else if(currencyName.equals("MUR - Mauritian Rupee")) return "Mauritius Rupee";
else if(currencyName.equals("MXN - Mexican Pesos")) return "Mexican Peso";
else if(currencyName.equals("TWD - New Taiwan Dollar")) return "Taiwanese Dollar";
else if(currencyName.equals("TRY - Turkish Lira"))return "New Turkish Lira";
else if(currencyName.equals("NZD - New Zealand Dollar"))
    return "New Zealand Dollar";
else if(currencyName.equals("NOK - Norwegian Kroner")) return "Norwegian Krone";
else if(currencyName.equals("OMR - Omani Rial"))return "";
else if(currencyName.equals("PGK - Papua New Guinea Kina")) return "";
else if(currencyName.equals("PEN - Peru New Sol")) return "Peruvian Nuevo Sol";
else if(currencyName.equals("PHP - Philippines Piso")) return "Phillipine Peso";
else if(currencyName.equals("PLN - Polish Zloty")) return "Polish Zloty";
else if(currencyName.equals("QAR - Qatar Riyal")) return "Qatari Rial";
else if(currencyName.equals("RUB - Russian Ruble")) return "Russian Ruble";
else if(currencyName.equals("SAR - Saudi Arabian Riyal")) return "Saudi Riyal";
else if(currencyName.equals("SGD - Singapore Dollar")) return "Singapore Dollar";
else if(currencyName.equals("ZAR - South African Rand"))
    return "South African Rand";
else if(currencyName.equals("KRW - South Korean Won"))return "Korean Wong";
else if(currencyName.equals("SEK - Swedish Krona")) return "Swedish Krona";
else if(currencyName.equals("CHF - Swiss Franc")) return "Swiss Franc";
else if(currencyName.equals("THB - Thai Bath")) return "Thai Baht";
else if(currencyName.equals("TTD - Trinidad & Tobago Dollar"))
    return "Trinidad and Tobago Dollar";
else if(currencyName.equals("AED - Uae Dirham")) return "UAE Dirham";
else return null;
}

/**
 * switchXRateCurrencyName() method converts currency name that is inside spinner
 * to currency name that is inside database table called TABLE_XRATE_NAME
 *
 * @param string currencyName that represent currency name inside spinners.
 * @return string that represent currency name inside database table TABLE_XRATE_NAME.
 */

```

```

public String switchXRateCurrencyName(String name){
    if(name.equals("GBP - British Pound"))return "GBP - BRITISH POUND";
    else if("EUR - EURO".equals(name)) return "Euro";
    else if("USD - US Dollar".equals(name)) return "US Dollar";
    else if("AUD - Australian Dollar".equals(name)) return "Australian Dollar";
    else if("BHD - Bahrain Dinar".equals(name)) return "Bahraini Dinar";
    else if("BBD - Barbados Dollar".equals(name)) return "";
    else if("BRL - Brazilian Real".equals(name)) return "Brazilian Real";
    else if("BND - Brunei Ringgit".equals(name)) return "Bruneian Dollar";
    else if("BGN - Bulgarian Lev".equals(name)) return "Bulgarian Lev";
    else if("CAD - Canadian Dollar".equals(name)) return "Canadian Dollar";
    else if("KYD - Cayman Island Dollar".equals(name)) return "";
    else if("CLP - Chilean Peso".equals(name)) return "Chilean Peso";
    else if("CNY - Chinese Yuan".equals(name)) return "Chinese Yuan Renminbi";
    else if("CRC - Costa Rican Colon".equals(name)) return "";
    else if("HRK - Croatian Kuna".equals(name))return "Croatian Kuna";
    else if("CZK - Czech Koruna".equals(name))return "Czech Koruna";
    else if("DKK - Danish Kroner".equals(name))return "Danish Krone";
    else if("DOP - Dominican Rep. Peso".equals(name))return "";
    else if("XCD - East Caribbean Dollar".equals(name))return "";
    else if("EGP - Egyptian Pound".equals(name))return "";
    else if("FJD - Fiji Dollar".equals(name))return "";
    else if("HKD - Hong Kong Dollar".equals(name))return "Hong Kong Dollar";
    else if("HUF - Hungarian Forint".equals(name))return "Hungarian Forint";
    else if("ISK - Icelandic Krona".equals(name)) return "Icelandic Krona";
    else if("IDR - Indonesian Rupiah".equals(name))return "Indonesian Rupiah";
    else if("ILS - Israeli New Sheqel".equals(name))return "Israeli Shekel";
    else if("JMD - Jamaican Dollar".equals(name))return "";
    else if("JPY - Japanese Yen".equals(name))return "Japanese Yen";
    else if("JOD - Jordanian Dinar".equals(name))return "";
    else if("KES - Kenyan Shilling".equals(name))return "";
    else if("KWD - Kuwaiti Dinar".equals(name))return "Kuwaiti Dinar";
    else if("LBP - Lebanon Pound".equals(name))return "";
    else if("MYR - Malaysian Ringgit".equals(name))return "Malaysian Ringgit";
    else if("MUR - Mauritian Rupee".equals(name))return "Mauritian Rupee";
    else if("MXN - Mexican Pesos".equals(name))return "Mexican Peso";
    else if("TWD - New Taiwan Dollar".equals(name))return "Taiwan New Dollar";
    else if("TRY - Turkish Lira".equals(name))return "Turkish Lira";
    else if("NZD - New Zealand Dollar".equals(name))return "New Zealand Dollar";
    else if("NOK - Norwegian Kroner".equals(name))return "Norwegian Krone";
    else if("OMR - Omani Rial".equals(name))return "Omani Rial";
    else if("PGK - Papua New Guinea Kina".equals(name))return "";
    else if("PEN - Peru New Sol".equals(name))return "";
    else if("PHP - Philippines Piso".equals(name))return "Philippine Peso";
    else if("PLN - Polish Zloty".equals(name))return "Polish Zloty";
    else if("QAR - Qatar Riyal".equals(name))return "Qatari Riyal";
    else if("RUB - Russian Ruble".equals(name))return "Russian Ruble";
    else if("SAR - Saudi Arabian Riyal".equals(name))return "Saudi Arabian Riyal";
    else if("SGD - Singapore Dollar".equals(name))return "Singapore Dollar";
    else if("ZAR - South African Rand".equals(name))return "South African Rand";
    else if("KRW - South Korean Won".equals(name))return "South Korean Won";
    else if("SEK - Swedish Krona".equals(name))return "Swedish Krona";
    else if("CHF - Swiss Franc".equals(name))return "Swiss Franc";
    else if("THB - Thai Bath".equals(name))return "Thai Baht";
    else if("TTD - Trinidad & Tobago Dollar".equals(name))return "Trinidadian Dollar";
    else if("AED - Uae Dirham".equals(name))return "Emirati Dirham";
    else return null;
}
}
/**
 * switchRBSCurrencyName() method converts currency name that is inside spinner
 * to currency name that is inside database table called TABLE_RBS_NAME.
 *
 * @param string currencyName that represent currency name inside spinners.
 * @return string that represent currency name inside database table TABLE_RBS_NAME.
 */

```

```

public String switchRBSCurrencyName(String currencyName){

    if(currencyName.equals("GBP - British Pound"))
        return "GBP - BRITISH POUND";
    else if(currencyName.equals("EUR - EURO"))
        return "EURO";
    else if(currencyName.equals("USD - US Dollar"))
        return "US DOLLARS";
    else if(currencyName.equals("AUD - Australian Dollar"))
        return "AUSTRALIAN DOLLARS";
    else if(currencyName.equals("BHD - Bahrain Dinar"))
        return "BAHRAINIAN DINARS";
    else if(currencyName.equals("BBD - Barbados Dollar"))
        return "BARBADOS DOLLARS";
    else if(currencyName.equals("BRL - Brazilian Real"))
        return "BRAZILIAN REAL";
    else if(currencyName.equals("BND - Brunei Ringgit"))
        return "BRUNEI RINGGITS";
    else if(currencyName.equals("BGN - Bulgarian Lev"))
        return "BULGARIAN LEV";
    else if(currencyName.equals("CAD - Canadian Dollar"))
        return "CANADIAN DOLLARS";
    else if(currencyName.equals("KYD - Cayman Island Dollar"))
        return "CAYMAN ISLANDS DOLLARS";
    else if(currencyName.equals("CLP - Chilean Peso"))
        return "CHILEAN PESOS";
    else if(currencyName.equals("CNY - Chinese Yuan"))
        return "CHINESE YUAN";
    else if(currencyName.equals("CRC - Costa Rican Colon"))
        return "COSTA RICAN COLON";
    else if(currencyName.equals("HRK - Croatian Kuna"))
        return "CROATIAN KUNA";
    else if(currencyName.equals("CZK - Czech Koruna"))
        return "CZECH KORUNA";
    else if(currencyName.equals("DKK - Danish Kroner"))
        return "DANISH KRONER";
    else if(currencyName.equals("DOP - Dominican Rep. Peso"))
        return "DOMINICAN REPUBLIC PESOS";
    else if(currencyName.equals("XCD - East Caribbean Dollar"))
        return "EAST CARIBBEAN DOLLAR";
    else if(currencyName.equals("EGP - Egyptian Pound"))
        return "EGYPTIAN POUNDS";
    else if(currencyName.equals("FJD - Fiji Dollar"))
        return "FIJI DOLLARS";
    else if(currencyName.equals("HKD - Hong Kong Dollar"))
        return "HONG KONG DOLLARS";
    else if(currencyName.equals("HUF - Hungarian Forint"))
        return "HUNGARIAN FORINTS";
    else if(currencyName.equals("ISK - Icelandic Krona"))
        return "ICELANDIC KRONA";
    else if(currencyName.equals("IDR - Indonesian Rupiah"))
        return "INDONESIAN RUPIAH";
    else if(currencyName.equals("ILS - Israeli New Sheqel"))
        return "ISRAELI NEW SHEQELS";
    else if(currencyName.equals("JMD - Jamaican Dollar"))
        return "JAMAICAN DOLLARS";
    else if(currencyName.equals("JPY - Japanese Yen"))
        return "JAPANESE YEN";
    else if(currencyName.equals("JOD - Jordanian Dinar"))
        return "JORDANIAN DINARS";
    else if(currencyName.equals("KES - Kenyan Shilling"))
        return "KENYAN SHILLINGS";
    else if(currencyName.equals("KWD - Kuwaiti Dinar"))
        return "KUWAITI DINARS";
    else if(currencyName.equals("LBP - Lebanon Pound"))
        return "LEBANON POUND";
}

```

```

else if(currencyName.equals("MYR - Malaysian Ringgit"))
    return "MALAYSIAN RINGGITS";
else if(currencyName.equals("MUR - Mauritian Rupee"))
    return "MAURITIAN RUPEES";
else if(currencyName.equals("MXN - Mexican Pesos"))
    return "MEXICAN PESOS";
else if(currencyName.equals("TWD - New Taiwan Dollar"))
    return "NEW TAIWAN DOLLARS";
else if(currencyName.equals("TRY - Turkish Lira"))
    return "NEW TURKISH LIRA";
else if(currencyName.equals("NDZ - New Zealand Dollar"))
    return "NEW ZEALAND DOLLARS";
else if(currencyName.equals("NOK - Norwegian Kroner"))
    return "NORWEGIAN KRONER";
else if(currencyName.equals("OMR - Omani Rial"))
    return "OMANI RIALS";
else if(currencyName.equals("PGK - Papua New Guinea Kina"))
    return "PAPUA NEW GUINEA KINA";
else if(currencyName.equals("PEN - Peru New Sol"))
    return "PERU NEW SOL";
else if(currencyName.equals("PHP - Philippines Piso"))
    return "PHILIPPINES PISO";
else if(currencyName.equals("PLN - Polish Zloty"))
    return "POLISH ZLOTY";
else if(currencyName.equals("QAR - Qatar Riyal"))
    return "QATAR RIYALS";
else if(currencyName.equals("RUB - Russian Ruble"))
    return "RUSSIAN RUBLE";
else if(currencyName.equals("SAR - Saudi Arabian Riyal"))
    return "SAUDI ARABIAN RIYALS";
else if(currencyName.equals("SGD - Singapore Dollar"))
    return "SINGAPORE DOLLARS";
else if(currencyName.equals("ZAR - South African Rand"))
    return "SOUTH AFRICAN RAND";
else if(currencyName.equals("KRW - South Korean Won"))
    return "SOUTH KOREAN WON";
else if(currencyName.equals("SEK - Swedish Krona"))
    return "SWEDISH KRONA";
else if(currencyName.equals("CHF - Swiss Franc"))
    return "SWISS FRANCS";
else if(currencyName.equals("THB - Thai Bath"))
    return "THAI BAHTS";
else if(currencyName.equals("TTD - Trinidad & Tobago Dollar"))
    return "TRINIDAD & TOBAGO DOLLAR";
else if(currencyName.equals("AED - Uae Dirham"))
    return "UAE DIRHAMS";
else
    return null;
    }
}

```

### Used.java

```

package org.example.currenciesrateexchange;
import android.app.Activity;
import android.os.Bundle;
import android.view.WindowManager;

/**
 * Used class enable to open new xml layout activity called how_to_use.xml
 * where layout is set with exact width and height.
 *
 * @author dsajdl01 (David Sajdl)
 * @version (01/03/2015)
 */

```

```

public class Used extends Activity{

    /**
     * onCreate() method allows to open new xml layout called about.xml
     */

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

        WindowManager.LayoutParams params = getWindow().getAttributes();
        params.x = 0;
        params.height = 380;
        params.width = 270;
        params.y = 25;
        this.getWindow().setAttributes(params);
    }
}

```

### ValidateBarclaysData.java

```

package org.example.currenciesrateexchange;

/**
 * ValidateBarclaysData class enable to clear and validate html data
 * that is loaded from Barclays web site
 *
 * @author dsajdl01 (David Sajdl)
 * @version (01/14/2015)
 */
public class ValidateBarclaysData implements Validator {

    private String[] barclaysData;
    private boolean BarclaysValid;
    private String errorMessage;

    /**
     * constructor takes one argument as a parameter that is a
     * string data that represent HTML source from Barclays web site
     */
    public ValidateBarclaysData(String url) {
        BarclaysValid = false;
        errorMessage = null;
        findData(url);
    }

    /**
     * getData() method returns array string that
     * contain currency name and currency value
     *
     * @return array string that contain currency's name and currency's value
     */
    @Override
    public String[] getData() {
        // TODO Auto-generated method stub
        return barclaysData;
    }

    /**
     * isValid() method returns true if data is validated or false if not
     *
     * @return boolean either true if data is validated or false if data is not validated
     */
}

```



```

@Override
public boolean getValid() {
    // TODO Auto-generated method stub
    return BarclaysValid;
}
/**
 * getErrorMessage() method returns error message if data is not validated
 * if data is validated method return null.
 *
 * @return string that contain either error message or null
 */
@Override
public String getErrorMessage() {
    // TODO Auto-generated method stub
    return errorMessage;
}
/**
 * findData() method send a message to split HTML data to data that is needed
 * and check if data contain what it should "validate data"
 * if data is validated then it assigns instance variable: barclaysData with data that
 * is needed, BarclaysValid assigns to true and errorMessage to null.
 * if data is NOT validate then it assigns instance variable: errorMessage with
 * validation error message, dataIsValid to false and barclaysData vith null.
 */
@Override
public void findData(String data){
    // TODO Auto-generated method stub
    boolean error = true;
    String barData = getCurrency(data);
    String[] barArrdata = barData.split(",");
    for(int i = 0; i < barArrdata.length-1; i = i + 2){
        if(!validateData(barArrdata[i],barArrdata[i+1])){
            error = false;
            break;
        }
    }
    if(error){
        barclaysData = barArrdata;
        BarclaysValid = true;
        errorMessage = null;
    }else{
        barclaysData = null;
        BarclaysValid = false;
        errorMessage = "Barclays data was not validated!";
    }
}
/**
 * getCurrency() method split HTML data to data that are needed
 * that represent currency name and currency value
 *
 * @return string that contain currency name and currency value
 */
@Override
public String getCurrency(String url) {
    // TODO Auto-generated method stub
    String[] dataArr = url.split("<>");
    int count = 0;
    String barData = "";
    boolean toAdd = false;
    boolean toBreak = false;
    for(int i = 0; i < dataArr.length; i++){
        if(dataArr[i].equals("British Pound")){
            toAdd = true;
        }
        if(toAdd){
            if(dataArr[i].equals("span")||dataArr[i].equals("/div")){

```





```

/**
 * getData() method returns data of BRS currency name,
 * BRS currency buy value and BRS currency sell value.
 *
 * @return string array that contain BRS data in format as BRS currency name,
 * BRS currency buy value and BRS currency sell value
 */
@Override
public String[] getData() {
    // TODO Auto-generated method stub
    return rbsData;
}
/**
 * getValid() method checks if data has been successful valid
 *
 * @return boolean either true if data is valid or false if data is not valid
 */
@Override
public boolean getValid() {
    // TODO Auto-generated method stub
    return dataIsValid;
}
/**
 * getErrorMessage() method return error validation message
 * if data is not validated or null if data if validated.
 *
 * @return string that contain either validation error message or null
 */
@Override
public String getErrorMessage() {
    // TODO Auto-generated method stub
    return errorMessage;
}
/**
 * findData() method send a message to split HTML data to get data that is needed
 * and check if data contain what it should "validate data"
 * if data is validated then it assigns instance variable: rdsData with data
 * that is needed, dataIsValid assigns to true and errorMessage to null
 * if data is NOT validate then it assigns instance variable: errorMessage with
 * validation error message, dataIsValid to false and rbsData to null.
 */
@Override
public void findData(String url) {
    // TODO Auto-generated method stub
    String values = getCurrency(url);
    String[] dataArray = values.split(",");
    int lastIndex = dataArray.length - 1;
    if(validateData(dataArray[0],dataArray[1],dataArray[2])
        && validateData(dataArray[lastIndex-3],dataArray[lastIndex-2],
            dataArray[lastIndex-1])){

        rbsData = dataArray;
        dataIsValid = true;
        errorMessage = null;
    }
    else{
        errorMessage = "RBS data is not valiaded!";
        dataIsValid = false;
        rbsData = null;
    }
}
/**
 * getCurrency() method takes download data as an argument
 * and clear data for a data that is needed (currency name and value).
 *
 * @param urlStr string as HTML data

```

```

        * @return data string that contain only data that is needed
        */
    @Override
    public String getCurrency(String url) {
        // TODO Auto-generated method stub
        String[] urlArr = url.split("<>");
        String data = "";
        int count = 0;
        for(int i = 0; i < urlArr.length; i++){
            if(urlArr[i].equals("Travellers Cheques"))
                break;
            if(urlArr[i].equals("td")){
                data = data + urlArr[i + 1] + ", ";
                count++;
                if(count == 3){
                    count = 0;
                    data = data + "\n";
                }
            }
        }
        return data;
    }

    /**
     * validateData() methods is implemented from interface class but method
     * its not needed! Instead is used validateData method with three parameter arguments.
     */
    @Override
    public boolean validateData(String name, String value) {
        // TODO Auto-generated method stub
        return false;
    }

    /**
     * validateData method take three arguments as string that are data from web server
     * Method validate date if data are contain what we are interested
     *
     * @param d1 string that is currency name
     * @param d2 string that contain currency value of buying
     * @param d3 string that contain currency value of selling
     * @return boolean condition true if the values contain correct data otherwise false
     */
    public boolean validateData(String d1, String d2, String d3){
        final String DigitsRegExp = "^\\d{0,6}\\.\\d{0,4}|^\\d{0,6}$";
        final String LettersRegExp = "[a-zA-Z]+|[a-zA-Z]+\\&[a-zA-Z]+\\;[a-zA-Z]+&";
        return d1.replaceAll("\\s+", "").matches(LettersRegExp) &&
            d2.replaceAll("\\s+", "").matches(DigitsRegExp) &&
            d3.replaceAll("\\s+", "").matches(DigitsRegExp);
    }
}

```

## ValidateXRateData.java

```

package org.example.currenciesrateexchange;

/**
 * ValidateXRateData class enable to clear and validate HTML data
 * that is loaded from x-Rate web site.
 *
 * @author dsajdl01 (David Sajdl)
 * @version (01/14/2015)
 */
public class ValidateXRateData implements Validator{

    private String[] xRateData;

```

```

private boolean xRateValid;
private String errorMessage;

/**
 * constructor takes one argument as a parameter that is
 * string data that represent HTML source of x-Rate web site.
 */
public ValidateXRateData(String data){
    this.xRateValid = false;
    this.errorMessage = null;
    this.findData(data);
}

/**
 * getData() method returns array string that
 * contain currency name and currency value if data is validated
 * or null if data is not validated.
 *
 * @return array string that contain currency's name and currency's value or null.
 */

@Override
public String[] getData() {
    // TODO Auto-generated method stub
    return xRateData;
}

/**
 * getValid() method returns true if data is validated or false if not.
 *
 * @return boolean either true if data is validated or false if data is not validated.
 */
@Override
public boolean getValid() {
    // TODO Auto-generated method stub
    return xRateValid;
}

/**
 * getErrorMessage() method returns error message if data is not validated
 * or null if data is validate.
 *
 * @return string that contain either error message if data is not valid or
 * null if data is valid.
 */

@Override
public String getErrorMessage(){
    return errorMessage;
}

/**
 * findData() method send a message to split HTML data to get data that is needed
 * and check if data contain what it should "validate data"
 * if data is validated then it assigns instance variable:
 * xRateData with data that is needed,
 * aRateValid assigns to true and errorMessage to null.
 * if data is NOT validate then it assigns instance variable: xRateData with null
 * errorMessage with validation error message and dataIsValid to false.
 */

```

```

@Override
public void findData(String data){
    // TODO Auto-generated method stub
    boolean error = true;
    String dataXRate = getCurrency(data);
    String[] xRateArray = dataXRate.split(",");
    for(int i =0; i < xRateArray.length-1; i = i + 2){
        if(!validateData(xRateArray[i],xRateArray[i+1])){
            error = false;
            break;
        }
    }

    if(error){
        xRateData = xRateArray;
        xRateValid = true;
        errorMessage = null;
    }
    else{
        xRateData = null;
        xRateValid = false;
        errorMessage = "x-Rate data was NOT validated!";
    }
}
/**
 * getCurrency() method split HTML data to data that are needed
 * that represent currency name and currency value.
 *
 * @return string that contain currency name and currency value.
 */
@Override
public String getCurrency(String url) {
    String[] arrData = url.split("<>");
    int count =0;
    String message = "";
    for(int i = 0; i < arrData.length; i++){
        if("tbody".equals(arrData[i])){
            count ++;
        }
        if(count > 1){
            if("td".equals(arrData[i]) &&
                (!("class='rtRates'".equals(arrData[i].trim())))){
                if("Venezuelan Bolivar".equals(arrData[i+1].trim())){
                    message = message + arrData[i+1] + "," + arrData[i + 7];
                    break;
                }
                else{
                    message = message + arrData[i+1] + "," + arrData[i + 7]+",";
                }
            }
        }
    }
    return message;
}
/**
 * validateData() method checks if strings from parameter contain what it should.
 * First parameter name only alphabetical letters and second one
 * value only digits with one dots.
 *
 * @param string name that represent currency's name.
 * @param string value that represent currency's value.
 * @return boolean that contain either true if data is validated or
 *         false if data is not validated.
 */

```

```

@Override
public boolean validateData(String name, String value){
    final String DigitsRegExp = "^\\d{0,6}\\.|\\.\\d{0,7}|^\\d{0,6}$";
    final String LettersRegExp = "[a-zA-Z]+$";
    return name.replaceAll("\\s+", "").matches(LettersRegExp) &&
           value.replaceAll("\\s+", "").matches(DigitsRegExp);
}
}

```

## XRateSource.java

```

package org.example.currenciesrateexchange;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
/**
 * XRateSource class is a middle class or communication class between database tables
 * called TABLE_XRATE_NAME and CurrenciesRateExchangeMainActivity class
 * Class enable to open and close database
 * insert data into table, update table and retrieve data from the table.
 *
 * @author dsajdl01 (David Sajdl)
 * @version (01/14/2015)
 */
public class XRateSource implements Source {
    //declaring instance variables
    private SQLiteDatabase database;
    private MySQLiteHelper dbHelper;
    private String[] allColumns =
        {MySQLiteHelper.COLUMN_XRATE_ID, MySQLiteHelper.COLUMN_XRATE_CURRENCY_NAME,
        MySQLiteHelper.COLUMN_XRATE_VALUE};

    /**
     * constructor that has one argument called Context object
     *
     * @param Context that in object of the calls CurrenciesRateExchangeMainActivity
     */
    protected XRateSource(Context context){
        dbHelper = new MySQLiteHelper(context);
    }

    /**
     * open() method opens database if method does not manage to open
     * database it throws SQLException
     *
     * @throws SQLException if does not manage to open database
     */
    @Override
    public void open() throws SQLException{
        database = dbHelper.getReadableDatabase();
    }

    /**
     * close() method closes database
     */
    @Override
    public void close(){
        dbHelper.close();
    }

    /**
     * addValueIntoTable() method insert arguments values into table called TABLE_XRATE_NAME
     *
     * @param currencyName string currency name (country)
     * @param value double that represent value of the currency name
     */
}

```

```

@Override
public void addValueIntoTable(String currencyName, Double value) {
    // TODO Auto-generated method stub
    ContentValues val= new ContentValues();
    currencyName = currencyName.trim();
    val.put(MySQLiteHelper.COLUMN_XRATE_CURRENCY_NAME, currencyName);
    val.put(MySQLiteHelper.COLUMN_XRATE_VALUE, value);
    database.insert(MySQLiteHelper.TABLE_XRATE_NAME, null, val);
}
/**
 * updateTable() method enables to update database table called TABLE_XRATE_NAME
 * Method update column COLUMN_XRATE_VALUE using WHERE clause where
 * column name COLUMN_XRATE_CURRENCY_NAME name is equal to name
 */
@Override
public void updateTable(String currencyName, Double value) {
    // TODO Auto-generated method stub
    currencyName = "" +currencyName.trim() +"";
    ContentValues val = new ContentValues();
    val.put(MySQLiteHelper.COLUMN_XRATE_VALUE, value);
    database.update(MySQLiteHelper.TABLE_XRATE_NAME, val,
        MySQLiteHelper.COLUMN_XRATE_CURRENCY_NAME + "=" + currencyName, null);
}
/**
 * getCurrencyValue() method enables to retrieve currency' value where currency
 * name equal to currency' name passed by parameter argument.
 */
@Override
public double getCurrencyValue(String name) {
    // TODO Auto-generated method stub
    Cursor cr = database.rawQuery("select * from " + MySQLiteHelper.TABLE_XRATE_NAME +
        " where " + MySQLiteHelper.COLUMN_XRATE_CURRENCY_NAME + " = " + "" +
        name + "", null);
    cr.moveToFirst();
    return cr.getDouble(2);
}
/**
 * getProfilesCount() method enables to count raw in the database
 * table called TABLE_XRATE_NAME
 */
@Override
public int getProfilesCount() {
    String countQuery = "SELECT * FROM " + MySQLiteHelper.TABLE_XRATE_NAME;
    Cursor cursor = database.rawQuery(countQuery, null);
    int cnt = cursor.getCount();
    cursor.close();
    return cnt;
}

//*****
//**          METHODS BELOW DO NOT NEED IT FOR CER APP          **//
//**          however result of the method was print it out to check it          **//
//**          if all data is inserted and then updated.          **//
//*****

/**
 * getAllTuples() method retrieves all data from the database
 * table called TABLE_XRATE_NAME
 */
@Override
public String getAllTuples() {
    // TODO Auto-generated method stub
    Cursor cursor = database.rawQuery("select * from " + MySQLiteHelper.TABLE_XRATE_NAME, null);
    StringBuilder sb = new StringBuilder();
    while (cursor.moveToNext()) {
        sb.append(cursor.getString(0) + " " + cursor.getString(1) + " " + cursor.getString(2) + "\n");
    }
    return sb.toString();
}

```

```

public String getAllTuples(){
    String allTuples = "Database Values\n\n";
    Cursor cr = database.query(MySQLiteHelper.TABLE_XRATE_NAME, allColumns,
                                null, null, null, null, null);

    cr.moveToFirst();
    while(!cr.isAfterLast()){
        allTuples = allTuples + getRowValues(cr);
        cr.moveToNext();
    }
    cr.close();
    return allTuples;
}
/**
 * getRowValues() private method convert Cursor object into string
 * Cursor is output of the database queries that is casting into string
 *
 * @return string that contains currency's id, currency's name and currency's value
 */
private String getRowValues(Cursor cr){
    long id = (cr.getLong(0));
    String name = (cr.getString(1));
    Double value = (cr.getDouble(2));
    return " " + id + ". " + name + ", " + value + "\n";
}
}

```

### Source.java – Interface

```

package org.example.currenciesrateexchange;
/**
 * Source interface class set methods that must be used for
 * implemented class.
 *
 * @author dsajdl01 (David Sajdl)
 * @version (01/14/2015)
 */
public interface Source {
    public void open();
    public void close();
    public void addValueIntoTable(String name, Double value);
    public void updateTable(String name, Double value);
    public double getCurrencyValue(String name);
}

```

### Valicator.java – Interface

```

package org.example.currenciesrateexchange;
/**
 * Validator interface class set methods that must be used for
 * implemented class.
 *
 * @author dsajdl01 (David Sajdl)
 * @version (01/14/2015)
 */
public interface Validator {
    public String[] getData();
    public boolean getValid();
    public String getErrorMessage();
    public void findData(String url);
    public String getCurrency(String url);
    public boolean validateData(String name, String value);
}

```



## Appendix C - XML resource

The program consists with five XML layouts and one a preference XML file. The Application has eight XML files describing the background of the layouts or buttons. The program also consists with: string.xml file where all localizable strings are archived, color.xml file where localizable colour are stored and AndroidManifest.xml contains essential information about the CER app. All these files are presented in this Appendix C.

### activity\_main.xml

```
<!--
    activity_main.xml is main layout for CER app
    It is used when the application get active and android screen device is in the vertical
    position. -->
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:background="@color/white"
        android:paddingBottom="@dimen/activity_horizontal_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="0dip" >

        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:id="@+id/LinearLayoutSecond"
            android:paddingTop="-120dip"
            android:paddingBottom="20dip"
            android:layout_marginTop="-26dip"
            android:layout_marginLeft="0dip"
            android:layout_marginRight="0dip"
            android:background="@color/white">

            <Button
                android:id="@+id/btn_about"
                android:layout_width="144dip"
                android:layout_height="wrap_content"
                android:layout_marginTop="0dip"
                android:layout_marginBottom="0dip"
                android:layout_marginRight="0dip"
                android:paddingBottom="0dip"
                android:paddingTop="24dip"
                android:paddingLeft="0dip"
                android:paddingRight="0dip"
                android:layout_gravity="left"
                android:textSize="14sp"
                android:text="@string/btn_About"
                android:onClick="btnAbout"
                android:textStyle="bold"
                android:gravity="left|center_vertical"
                android:background="@drawable/about_button" />

            <Button
                android:id="@+id/btn_used"
                android:layout_width="144dip"
                android:layout_height="wrap_content"
                android:layout_marginTop="0dip"
```

```

        android:layout_marginRight="0dip"
        android:paddingBottom="0dip"
        android:paddingTop="24dip"
        android:paddingLeft="0dip"
        android:paddingRight="0dip"
        android:layout_gravity="right"
        android:textSize="14sp"
        android:textStyle="bold"
        android:text="@string/btn_used"
        android:onClick="btnToUse"
        android:gravity="left/center_vertical"
        android:background="@drawable/used_button" />

</LinearLayout>

<LinearLayout
    android:layout_width="fill_parent"
    android:orientation="vertical"
    android:id="@+id/LinearLayoutFirst"
    android:paddingTop="12dip"
    android:paddingBottom="10dip"
    android:layout_marginBottom="10dip"
    android:background="@drawable/currency_layout"
    android:layout_height="wrap_content">

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:id="@+id/LinearLayout2"
        android:paddingTop="6dip"
        android:layout_gravity="left/right"
        android:gravity="left"
        android:paddingBottom="15dip"
        android:layout_marginLeft="1dip"
        android:layout_marginRight="1dip"
        android:background="@drawable/preference_layout">

        <TextView
            android:id="@+id/txt_preference"
            android:layout_width="138dip"
            android:paddingTop="0dip"
            android:paddingBottom="5dip"
            android:paddingLeft="8dip"
            android:layout_marginRight="3dip"
            android:paddingRight="1dip"
            android:layout_height="wrap_content"
            android:textStyle="bold"
            android:textSize="18sp"
            android:layout_gravity="left"
            android:text="@string/txt_preference" />

        <LinearLayout
            android:orientation="horizontal"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:gravity="right" >

            <Button
                android:id="@+id/btn_press"
                android:layout_width="138dip"
                android:layout_height="wrap_content"
                android:layout_marginTop="0dip"
                android:layout_marginLeft="4dip"
                android:layout_marginRight="4dip"

```

```

        android:paddingBottom="1dip"
        android:paddingTop="4dip"
        android:paddingLeft="4dip"
        android:paddingRight="1dip"
        android:layout_gravity="right"
        android:textSize="14sp"
        android:onClick="setPreferences"
        android:background="@drawable/button" />
    </LinearLayout>
</LinearLayout>
<TextView
    android:id="@+id/txt_from"
    android:layout_width="fill_parent"
    android:paddingTop="10dip"
    android:paddingBottom="1dip"
    android:paddingLeft="4dip"
    android:paddingRight="1dip"
    android:layout_height="wrap_content"
    android:textStyle="bold"
    android:textSize="16sp"
    android:text="@string/from" />
<Spinner
    android:id="@+id/spinner_From"
    android:layout_width="fill_parent"
    android:paddingTop="5dip"
    android:layout_marginLeft="4dip"
    android:layout_marginBottom="2dip"
    android:layout_marginRight="4dip"
    android:layout_marginTop="0dip"
    android:paddingBottom="5dip"
    android:paddingLeft="1dip"
    android:layout_height="wrap_content"
    android:background="@color/white"
    android:popupBackground="@color/popup_background"
    android:textSize="12sp"
    android:drawSelectorOnTop="false" />
<TextView
    android:id="@+id/txt_to"
    android:layout_width="fill_parent"
    android:paddingTop="5dip"
    android:paddingBottom="1dip"
    android:paddingLeft="4dip"
    android:paddingRight="1dip"
    android:layout_height="wrap_content"
    android:textStyle="bold"
    android:textSize="16sp"
    android:text="@string/to" />
<Spinner
    android:id="@+id/spinner_to"
    android:layout_width="fill_parent"
    android:layout_marginLeft="4dip"
    android:layout_marginBottom="10dip"
    android:layout_marginRight="4dip"
    android:layout_marginTop="0dip"
    android:paddingTop="5dip"
    android:paddingBottom="5dip"
    android:paddingLeft="1dip"
    android:layout_height="wrap_content"
    android:background="@color/white"
    android:popupBackground="@color/popup_background"
    android:textSize="12sp"
    android:layout_gravity="right"
    android:drawSelectorOnTop="false" />
<TextView
    android:id="@+id/txtview_inputRate"

```

```

        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:paddingTop="0dip"
        android:paddingBottom="15dip"
        android:maxLines="6"
        android:singleLine="false"
        android:background="@color/main_background" />
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:id="@+id/LinearLayout3"
    android:paddingTop="10dip"
    android:paddingBottom="4dip"
    android:layout_marginLeft="1dip"
    android:layout_marginRight="1dip"
    android:background="@drawable/input_layout" >
    <EditText
        android:id="@+id/txtEdit_amount"
        android:layout_width="200dip"
        android:layout_height="match_parent"
        android:layout_marginLeft="3dip"
        android:layout_marginBottom="5dip"
        android:layout_marginRight="8dip"
        android:layout_marginTop="10dip"
        android:paddingTop="3dip"
        android:inputType="numberDecimal"
        android:maxLength="7"
        android:paddingBottom="3dip"
        android:paddingLeft="1dip"
        android:gravity="left"
        android:hint="@string/hint_amount"
        android:background="@color/white" />
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dip"
        android:gravity="right" >
        <Button
            android:layout_width="70dip"
            android:layout_height="wrap_content"
            android:layout_marginRight="4dip"
            android:onClick="getData"
            android:gravity="center"
            android:layout_gravity="right"
            android:background="@drawable/result_button" />
    </LinearLayout>
</LinearLayout>
<TextView
    android:id="@+id/txtview_output_user_result"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:paddingTop="0dip"
    android:paddingBottom="5dip"
    android:layout_marginLeft="4dip"
    android:layout_marginBottom="0dip"
    android:layout_marginRight="4dip"
    android:layout_marginTop="0dip"
    android:paddingLeft="1dip"
    android:text="@string/txt_output"
    android:background="@color/main_background"/>
</LinearLayout>
</LinearLayout>
</ScrollView>

```

## about.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--
    about.xml file is the layout using for describing about activity.
    It is used when about button is pressed and new activity (about) gets active. -->
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="15dip" >

    <TextView
        android:id="@+id/about_txt1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/about_text1" />
</ScrollView>
```

## how\_to\_use.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--
    how_to_use.xml file is the layout uses for describing how to use activity.
    It is used when How To Use button is pressed and new activity (how_to_use) gets
    active. -->
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="15dip" >
    <TextView
        android:id="@+id/use_txt1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/used_text1" />
</ScrollView>
```

## notification.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--
    notification.xml is layout to describe notification layout
    It is used when a new notification object is created in runtime
    to inform user about current state of the application. -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="400dp"
        android:text="" />
</LinearLayout>
```

## toast\_layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--
    toast_layout.xml file is a layout for popping up short messages. -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/toast_layout_root"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="8dp"
    android:background="@color/toast_background" >
```

```

        <TextView android:id="@+id/text"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textStyle="bold"
            android:textColor="@color/white" />
    </LinearLayout>

```

## about\_button.xml

```

<?xml version="1.0" encoding="utf-8"?>
<!-- about_button.xml file gives a shape and colour of the background About button -->
<selector xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:state_pressed="true">
        <shape android:shape="rectangle">
            <corners android:bottomLeftRadius="5dp"/>
            <solid android:color="#d4dbf6"/>
            <padding android:left="10.0dip" android:top="10.0dip"
                android:right="10.0dip" android:bottom="10.0dip"/>
            <stroke android:width="0.5dip" android:color="#c5ceed"/>
        </shape>
    </item>
    <item android:state_pressed="false" >
        <shape android:shape="rectangle">
            <corners android:bottomLeftRadius="5dp"/>
            <solid android:color="#d4dbf6"/>
            <padding android:left="10.0dip" android:top="10dip"
                android:right="10.0dip" android:bottom="10.0dip"/>
            <stroke android:width="0.5dip" android:color="#d4dbf6"/>
        </shape>
    </item>
</selector>

```

## preferences.xml

```

<?xml version="1.0" encoding="utf-8"?>
<!--
    preferences.xml file is the preference, which is used to set
    which bank would be calautated currency rate exchange.
    It is used when about button "Setting exchange Preference"
    is pressed and preference gets active. -->
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android"
    android:key="checkbox_pref"
    android:title="Check Rate Preferences" >

    <Preference
        android:icon="@drawable/arrow"
        android:title="Back to main screen"
        android:key="button"/>
    <ListPreference
        android:title="Select Rate Preferences"
        android:summary="Click and then\nselect any kind of bank to display rate or\nselect
            the best result to find the best currency rate offer"

        android:key="dispalYRate"
        android:defaultValue="1"
        android:entries="@array/arrayListOfBanks"
        android:entryValues="@array/arrayVaLuesOfBanks"/>
</PreferenceScreen>

```

## button.xml

```

<?xml version="1.0" encoding="utf-8"?>
<!-- button.xml file gives shape and colour of the background Select Preference Bank button
-->
<selector xmlns:android="http://schemas.android.com/apk/res/android">

```

```

        <item android:state_pressed="true">
            <shape android:shape="rectangle">
                <corners android:radius="5dp"/>
                <solid android:color="#b5b5b5"/>
                <padding android:left="10.0dip" android:top="10.0dip"
                    android:right="10.0dip" android:bottom="10.0dip"/>
                <stroke android:width="0.5dip" android:color="#f1f1f1"/>
            </shape>
        </item>

        <item android:state_pressed="false" >
            <shape android:shape="rectangle">
                <corners android:radius="5dp"/>
                <solid android:color="#f1f1f1"/>
                <padding android:left="10.0dip" android:top="10.0dip"
                    android:right="10.0dip" android:bottom="10.0dip"/>
                <stroke android:width="0.5dip" android:color="#b5b5b5"/>
            </shape>
        </item>
    </selector>

```

### currency\_layout.xml

```

<?xml version="1.0" encoding="utf-8"?>
<!-- currency_layout gives shape and background colour to currency main container -->
<shape xmlns:android="http://schemas.android.com/apk/res/android" >
    <solid android:color="#d4dbf6"/>
    <stroke android:width="0.5dip" android:color="#d4dbf6" />
    <corners android:radius="10dip"/>
    <padding android:left="0dip" android:top="0dip" android:right="0dip"
        android:bottom="0dip" />
</shape>

```

### input\_layout.xml

```

<?xml version="1.0" encoding="utf-8"?>
<!-- input_layout.xml gives background colour and top horizontal line to calculate conversion
section. Calculate conversion section is where the users are allowed to enter amount of money
and press equal button. -->
<layer-list xmlns:android="http://schemas.android.com/apk/res/android" >
    <item>
        <shape android:shape="rectangle" >
            <solid android:color="#d4dbf6" />
        </shape>
    </item>

    <item
        android:top="0dp" android:right="-4dp" android:left="-4dp" android:bottom="-6dp">
        <shape>
            <solid android:color="#d4dbf6"/>
            <stroke
                android:width="2dp"
                android:color="#f1f1f1" />
        </shape>
    </item>
</layer-list>

```

### preference\_layout.xml

```

<?xml version="1.0" encoding="utf-8"?>
<!-- preference_layout.xml file gives background colour and bottom horizontal line to
preference section. Preference section is where selected preference is displayed with Select
preference button -->
<layer-list xmlns:android="http://schemas.android.com/apk/res/android" >

```

```

<item>
    <shape android:shape="rectangle" >
        <solid android:color="#d4dbf6" />
    </shape>
</item>

<item android:top="-8dp" android:right="-3dp" android:left="-3dp">
    <shape>
        <solid android:color="#d4dbf6"/>
        <stroke
            android:width="2dp"
            android:color="#f1f1f1" />
    </shape>
</item>
</layer-list>

```

### result\_button.xml

```

<?xml version="1.0" encoding="utf-8"?>
<!-- result_button.xml file gives shape, colour and image to equal button -->
<selector xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:state_pressed="true">
        <layer-list>
            <item>
                <shape android:shape="rectangle">
                    <corners android:radius="5dp"/>
                    <solid android:color="#b5b5b5"/>
                    <padding
                        android:left="10.0dip"
                        android:top="10.0dip"
                        android:right="10.0dip"
                        android:bottom="10.0dip"/>
                    <stroke
                        android:width="1.0dip"
                        android:color="#f1f1f1"/>
                </shape>
            </item>
            <item>
                <bitmap android:gravity="center"
                    android:src="@drawable/true_button_arrow_output" />
            </item>
        </layer-list>
    </item>
    <item android:state_pressed="false" >
        <layer-list>
            <item>
                <shape android:shape="rectangle">
                    <corners android:radius="5dp"/>
                    <solid android:color="#f1f1f1"/>
                    <padding android:left="10.0dip"
                        android:top="10.0dip"
                        android:right="10.0dip"
                        android:bottom="10.0dip"/>
                    <stroke android:width="1.0dip"
                        android:color="#b5b5b5"/>
                </shape>
            </item>
            <item>
                <bitmap android:gravity="center"
                    android:src="@drawable/false_button_arrow_output" />
            </item>
        </layer-list>
    </item>
</selector>

```



## shapedialogtheme.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- shapedialogtheme.xml gives black border and shape to layouts for About activity and How To Use activity -->
<shape xmlns:android="http://schemas.android.com/apk/res/android" >
    <solid android:color="#d4dbf6" />
    <stroke
        android:width="0.5dp"
        android:color="#000000" />
    <corners android:radius="12dp" />
</shape>
```

## used\_button.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- used_button.xml file gives shape and background colour to How To Used button. -->
<selector xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:state_pressed="true">
        <shape android:shape="rectangle">
            <corners android:bottomRightRadius="5dp"/>
            <solid android:color="#d4dbf6"/>
            <padding android:left="10.0dip" android:top="10.0dip"
                android:right="10.0dip" android:bottom="10.0dip"/>
            <stroke android:width="0.5dip" android:color="#c5ceed"/>
        </shape>
    </item>
    <item android:state_pressed="false" >
        <shape android:shape="rectangle">
            <corners android:bottomRightRadius="5dp"/>
            <solid android:color="#d4dbf6"/>
            <padding android:left="10.0dip" android:top="10dip"
                android:right="10.0dip" android:bottom="10.0dip"/>
            <stroke android:width="0.5dip" android:color="#d4dbf6"/>
        </shape>
    </item>
</selector>
```

## string.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- string.xml describes different text in the CER app. -->
<resources>
    <string name="app_name">"Currency Exchange Rate "</string>
    <string name="action_settings">Settings</string>
    <string name="from">CURRENCY I HAVE:</string>
    <string name="to">CURRENCY I WANT:</string>
    <string name="txt_preference">Preference is not selected</string>
    <string name="btn_About">About</string>
    <string name="btn_used">How To Use</string>
    <string name="txt_output">\n 0</string>
    <string name="hint_amount">Entry amount e.g: 1000</string>
    <string name="about_title">About Currency Exchange Rate</string>
    <string name="about_text1">Currency Rate Exchange application offers current rate exchange from the Royal Bank of Scotland 'RBS', x-Rate and Barclays.\n Exchange rate data are automatically loading when the application gets active and data is saved into android SQLite. \n Application informs users whether the application is using current data or data from the previous update with the date when the data was downloaded.\n\n Application also offers setting exchange preferences by pressing 'Setting Exchange Preferences'. \n It can be chosen between the list of different online currency converters and the banks or select the best result, which application offers the best rating result.
    </string>

    <string name="used_title">How To Use Currency Exchange Rate</string>
```

```

<string name="used_text1">CER app offer three main functionalities for the users. The
first one is Preference, second one is spinner selection and the last one is Input.
\n\n * PREFERENCE - offers for users to choose between different banks and online
conversion. User can select any of the three available banks (RBS, x-Rate and
Barclays) or select the best result in which case application would be displayed all
three banks with the greatest result first.\n\n * SPINNER SELECTION - offers for
users to choose between different currency name (country) and display current rate
offer. \n\n * INPUT - offers for users to enter amount of money that user wish to
exchange and by pressing equal button display result (how much money user get).
</string>

<string-array name="countries_name">
    <item>GBP - British Pound</item>
    <item>EUR - EURO</item>
    <item>USD - US Dollar</item>
    <item>AUD - Australian Dollar</item>
    <item>BHD - Bahrain Dinar</item>
    <item>BRL - Brazilian Real</item>
    <item>BGN - Bulgarian Lev</item>
    <item>CAD - Canadian Dollar</item>
    <item>CLP - Chilean Peso</item>
    <item>CNY - Chinese Yuan</item>
    <item>HRK - Croatian Kuna</item>
    <item>CZK - Czech Koruna</item>
    <item>DKK - Danish Kroner</item>
    <item>HKD - Hong Kong Dollar</item>
    <item>HUF - Hungarian Forint</item>
    <item>ISK - Icelandic Krona</item>
    <item>IDR - Indonesian Rupiah</item>
    <item>ILS - Israeli New Sheqel</item>
    <item>JPY - Japanese Yen</item>
    <item>KWD - Kuwaiti Dinar</item>
    <item>MYR - Malaysian Ringgit</item>
    <item>MUR - Mauritian Rupee</item>
    <item>MXN - Mexican Pesos</item>
    <item>TWD - New Taiwan Dollar</item>
    <item>TRY - Turkish Lira</item>
    <item>NDZ - New Zealand Dollar</item>
    <item>NOK - Norwegian Kroner</item>
    <item>PHP - Philippines Piso</item>
    <item>PLN - Polish Zloty</item>
    <item>QAR - Qatar Riyal</item>
    <item>SAR - Saudi Arabian Riyal</item>
    <item>SGD - Singapore Dollar</item>
    <item>ZAR - South African Rand</item>
    <item>KRW - South Korean Won</item>
    <item>SEK - Swedish Krona</item>
    <item>CHF - Swiss Franc</item>
    <item>THB - Thai Bath</item>
    <item>TTD - Trinidad & Tobago Dollar</item>
    <item>AED - Uae Dirham</item>
</string-array>
<string-array name="countries_name_two">
    <item>EUR - EURO</item>
    <item>GBP - British Pound</item>
    <item>USD - US Dollar</item>
    <item>AUD - Australian Dollar</item>
    <item>BHD - Bahrain Dinar</item>
    <item>BRL - Brazilian Real</item>
    <item>BGN - Bulgarian Lev</item>
    <item>CAD - Canadian Dollar</item>
    <item>CLP - Chilean Peso</item>
    <item>CNY - Chinese Yuan</item>
    <item>CRC - Costa Rican Colon</item>
    <item>HRK - Croatian Kuna</item>
    <item>CZK - Czech Koruna</item>

```

```

        <item>DKK - Danish Kroner</item>
        <item>HKD - Hong Kong Dollar</item>
        <item>HUF - Hungarian Forint</item>
        <item>ISK - Icelandic Krona</item>
        <item>IDR - Indonesian Rupiah</item>
        <item>ILS - Israeli New Sheqel</item>
        <item>JPY - Japanese Yen</item>
        <item>KWD - Kuwaiti Dinar</item>
        <item>MYR - Malaysian Ringgit</item>
        <item>MUR - Mauritian Rupee</item>
        <item>MXN - Mexican Pesos</item>
        <item>TWD - New Taiwan Dollar</item>
        <item>TRY - Turkish Lira</item>
        <item>NDZ - New Zealand Dollar</item>
        <item>NOK - Norwegian Kroner</item>
        <item>PHP - Philippines Piso</item>
        <item>PLN - Polish Zloty</item>
        <item>QAR - Qatar Riyal</item>
        <item>SAR - Saudi Arabian Riyal</item>
        <item>SGD - Singapore Dollar</item>
        <item>ZAR - South African Rand</item>
        <item>KRW - South Korean Won</item>
        <item>SEK - Swedish Krona</item>
        <item>CHF - Swiss Franc</item>
        <item>THB - Thai Bath</item>
        <item>TTD - Trinidad & Tobago Dollar</item>
        <item>AED - Uae Dirham</item>
    </string-array>
    <string-array name="arrayListOfBanks">
        <item>Best results</item>
        <item>RBS</item>
        <item>x-Rates</item>
        <item>Barclays</item>
    </string-array>
    <string-array name="arrayValuesOfBanks">
        <item>0</item>
        <item>1</item>
        <item>2</item>
        <item>3</item>
    </string-array>
</resources>

```

## color.xml

```

<?xml version="1.0" encoding="utf-8"?>
<!-- color.xml describe different colour that is used in the CER app. -->
<resources>
    <color name="white">#ffffff</color>
    <color name="main_background">#d4dbf6</color>
    <color name="txt_output">#f1f1f1</color>
    <color name="popup_background">#EDED</color>
    <color name="textColor">#000000</color>
    <color name="btn_backgroung">#efefef</color>
    <color name="toast_background">#000000</color>
    <color name="colorPrimary_background">#242424</color>
    <color name="black">#000000</color>
    <color name="whitegrey">#d4dbf6</color>
</resources>

```

## AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- AndroidManifest describes essential information about the CER app -->
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.example.currenciesrateexchange"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="17"
        android:targetSdkVersion="21" />

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

    <supports-screens
        android:largeScreens="false"
        android:normalScreens="true"
        android:smallScreens="true"
        android:anyDensity="true" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/Logo_cre"
        android:label="@string/app_name"
        android:theme="@style/CustomActionBarTheme" >
        <activity
            android:name=".CurrenciesRateExchangeMainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".SetPreferenceActivity"
            android:label="@string/app_name"
            android:theme="@style/PreferencesTheme">
        </activity>
        <activity android:name=".About"
            android:label="@string/about_title"
            android:theme="@style/dialog_light">
        </activity>
        <activity android:name=".Used"
            android:label="@string/used_title"
            android:theme="@style/dialog_light">
        </activity>
        <activity android:name=".NotificationView"
            android:label="Details of notification"
            android:parentActivityName=".CurrenciesRateExchangeMainActivity">
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value=".CurrenciesRateExchangeMainActivity"/>
        </activity>
    </application>

</manifest>
```

## Appendix D – Unit Testing

The unit test for the Android CER application is written in the separate Android project called CurrencyRateExchangeTest. The currency exchange rate test file consists of 6 unit test files. The following sub-chapters consist of Java unit test code files in an alphabetical order, which are written for methods that help processing a data in the CER app.

### CaluclateRateTest.java

```
package org.example.currenciesrateexchange.test;
import org.example.currenciesrateexchange.CalculateRate;
import android.test.AndroidTestCase;
/**
 * CalculateRateTest class enable to test all methods in
 * CalculateRate class in CurrenciesRateExchange file
 *
 * @author dsajdl01 (David Sajdl)
 * @version (01/03/2015)
 */
public class CalculateRateTest extends AndroidTestCase {
    private CalculateRate cr1;
    private CalculateRate cr2;
    private double have1, have2, want1, want2;

    public CalculateRateTest(){
    }
    /**
     * setUp() method is called before every single test inside
     * CalculateRateTest class is tested and it sets instance variables
     */
    public void setUp(){
        have1 = 3.0921;
        want1 = 6.932734;
        cr1 = new CalculateRate(have1, want1);
        have2 = 8;
        want2 = 0.896554;
        cr2 = new CalculateRate(have2, want2);
    }
    /**
     * tearDown() is called after every single test inside CalculateRateTest class is
     * tested and it sets instance variables cr1 and cr2 to null
     */
    public void tearDown(){
        cr1 = null;
        cr2 = null;
    }
    /**
     * testing getHave() method.
     */
    public void testGetHave(){
        assertEquals(have1, cr1.getHave());
        assertEquals(have2, cr2.getHave());
        assertNotSame(have1, cr2.getHave());
    }
    /**
     * testing getWant() method.
     */
    public void testGetWant(){
        assertEquals(want1, cr1.getWant());
        assertEquals(want2, cr2.getWant());
    }
}
```

```

        assertNotSame(want1, cr2.getWant());
    }
    /**
     * testing getRate() method.
     */
    public void testGetRate(){
        Double result1 = want1/have1;
        assertEquals(result1, cr1.getRate());
        Double result2 = want2/have2;
        assertEquals(result2, cr2.getRate());
        Double result3 = want1/have2;
        assertNotSame(result3, cr1.getRate());
        assertNotSame(result3, cr2.getRate());
    }
    /**
     * testing calculateUserInput() method.
     */
    public void testCalculateUserInput(){
        Double input = 1500.00;
        Double result1 = (want1/have1) * input;
        assertEquals(result1, cr1.calculateUserInput(input));
        Double result2 = (want2/have2) * input;
        assertEquals(result2, cr2.calculateUserInput(input));
        assertNotSame(result1, cr2.calculateUserInput(input));
        assertNotSame(result2, cr1.calculateUserInput(input));
    }
}
}

```

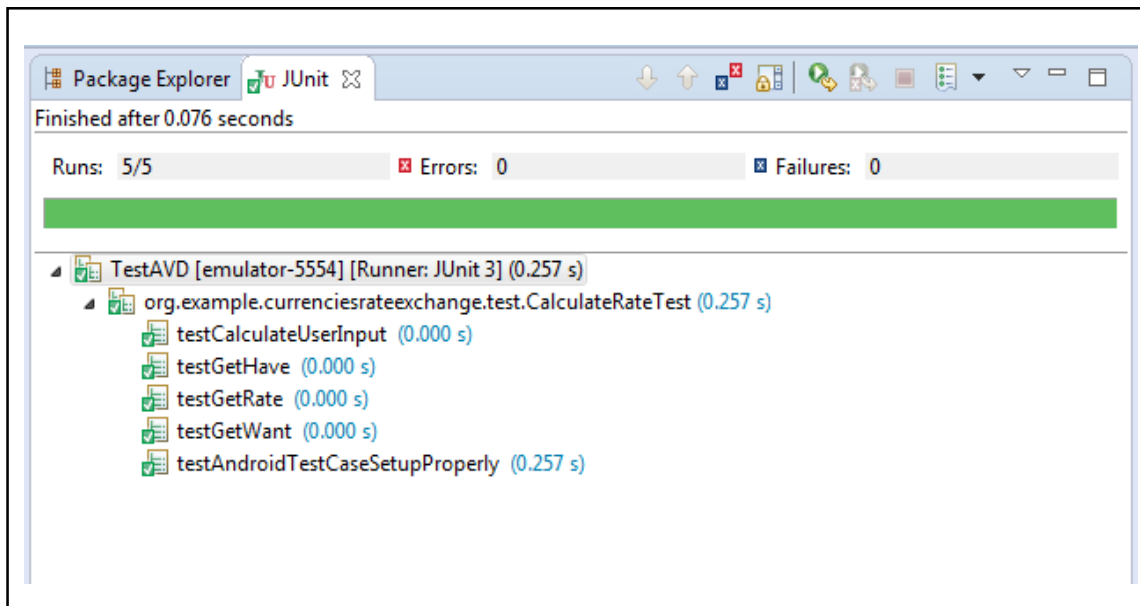


Figure A.6: The running unit test result of CaluclateRateTest class.

## CalendarTest.java

```

package org.example.currenciesrateexchange.test;
import org.example.currenciesrateexchange.Calendar;
import java.util.Date;
import java.text.SimpleDateFormat;
import android.annotation.SuppressLint;
import android.test.AndroidTestCase;
/**
 * CalendarTest class enable to test method in
 * Calendar class in CurrenciesRateExchange file
 *
 * @author dsajdl01 (David Sajdl)

```

```

* @version (01/03/2015)
*
*/
@SuppressLint("SimpleDateFormat")
public class CalendarTest extends AndroidTestCase {

    private Calendar c;

    public CalendarTest(){
    }
    /**
     * setUp() method is called before every single test inside
     * CalendarTest class is tested and it creates instance of the Calendar class
     */
    public void setUp(){
        c = new Calendar();
    }
    /**
     * tearDown() is called after every single test inside
     * CalendarTest class is tested and it sets instance variables c to null
     */
    protected void tearDown() {
        c = null;
    }
    /**
     * testing getCurrentDate() method
     */
    public void testGetCurrentDate(){
        String date = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(new Date());
        assertEquals(date, c.getCurrentDate());
    }
}

```

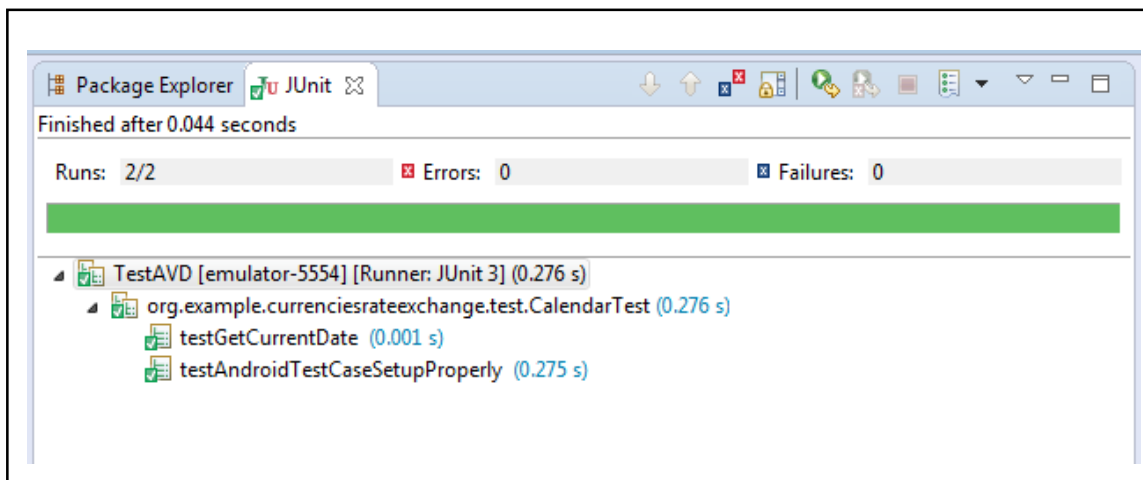


Figure A.7: The running unit test result of CalenderTest class.

### SwitchCurrencyNameTest.java

```

package org.example.currenciesrateexchange.test;

import org.example.currenciesrateexchange.SwitchCurrencyName;
import android.test.AndroidTestCase;
/**
 * SwitchCurrencyNameTest class enable to test all methods in
 * SwitchCurrencyName class in CurrenciesRateExchange file
 *
 * @author dsajdl01 (David Sajdl)

```

```

* @version (01/03/2015)
*
*/

public class SwitchCurrencyNameTest extends AndroidTestCase {

    private SwitchCurrencyName swn;
    private String euro, usa, czech, russia, thai;
    public SwitchCurrencyNameTest(){
    }
    /**
     * setUp() method is called before every single test inside
     * CalculateRateTest class is tested and it creates instance of the
     * SwitchCurrencyName class plus it sets instance variables with values
     */
    public void setUp(){
        swn = new SwitchCurrencyName();
        euro = "EUR - EURO";
        usa = "USD - US Dollar";
        czech = "CZK - Czech Koruna";
        russia = "RUB - Russian Ruble";
        thai = "THB - Thai Bath";

    }
    /**
     * tearDown() is called after every single test inside
     * CalculateRateTest class is tested and it sets instance variable swn to null
     */
    public void tearDown(){
        swn = null;
    }
    /**
     * testing switchXRateCurrencyName() method.
     */
    public void testSwitchXRateCurrencyName(){
        assertEquals("Euro", swn.switchXRateCurrencyName(euro));
        assertEquals("US Dollar", swn.switchXRateCurrencyName(usa));
        assertEquals("Czech Koruna", swn.switchXRateCurrencyName(czech));
        assertEquals("Russian Ruble", swn.switchXRateCurrencyName(russia));
        assertEquals("Thai Baht", swn.switchXRateCurrencyName(thai));
        assertNull(swn.switchXRateCurrencyName("England"));
    }
    /**
     * testing switchRBSCurrencyName() method.
     */
    public void testSwitchRBSCurrencyName(){
        assertEquals("EURO", swn.switchRBSCurrencyName(euro));
        assertEquals("US DOLLARS", swn.switchRBSCurrencyName(usa));
        assertEquals("CZECH KORUNA", swn.switchRBSCurrencyName(czech));
        assertEquals("RUSSIAN RUBLE", swn.switchRBSCurrencyName(russia));
        assertEquals("THAI BAHTS", swn.switchRBSCurrencyName(thai));
        assertNull(swn.switchRBSCurrencyName("England"));
    }
    /**
     * testing switchBarclaysCurrencyName() method.
     */
    public void testSwitchBarclaysCurrencyName(){
        assertEquals("Euro", swn.switchBarclaysCurrencyName(euro));
        assertEquals("US Dollar", swn.switchBarclaysCurrencyName(usa));
        assertEquals("Czech Koruna", swn.switchBarclaysCurrencyName(czech));
        assertEquals("Russian Ruble", swn.switchBarclaysCurrencyName(russia));
        assertEquals("Thai Baht", swn.switchBarclaysCurrencyName(thai));
        assertNull(swn.switchBarclaysCurrencyName("England"));
    }
}

```



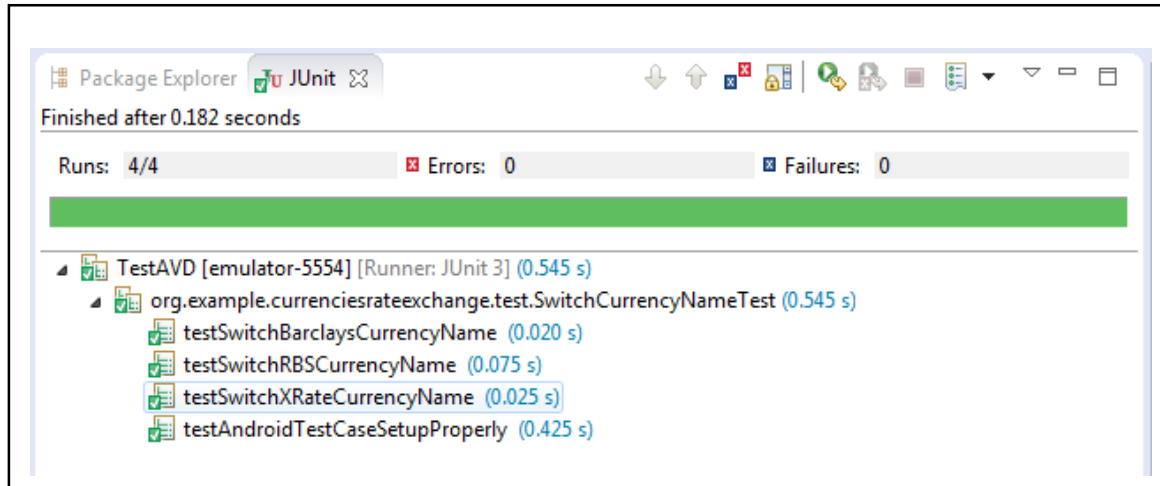


Figure A.8: The running unit test result of SwitchCurrencyNameTest class.

### ValidateBarclaysDataTest.java

```
package org.example.currenciesrateexchange.test;

import org.example.currenciesrateexchange.ValidateBarclaysData;
import android.test.AndroidTestCase;
/**
 * ValidateBarclaysDataTest class enable to test all methods in
 * ValidateBarclaysData class in CurrenciesRateExchange file
 *
 * @author dsajdl01 (David Sajdl)
 * @version (01/03/2015)
 */
public class ValidateBarclaysDataTest extends AndroidTestCase {

    private ValidateBarclaysData barclaysDataV;
    private ValidateBarclaysData barclaysDataNotV;

    public ValidateBarclaysDataTest(){

    }
    /**
     * setUp() method is called before every single test inside
     * ValidateBarclaysDataTest class is tested and it creates two instances of the
     * ValidateBarclaysData class plus it sets instance variables with HTML source value
     */
    protected void setUp() {
        String brcURLData = "<table class=\"fxrates_graph\" width=\"100%\" border=\"0\"><tr><td><br /><b>British Pound - euro</b> <br /><div class=\"fxrates_plot\" id=\"81b813907c47437a9db4011e74d16a27\" > . . . . . tabs();}}</script></div>}}";
        String brcURLData2 = "<table class=\"fxrates_graph\" width=\"100%\" border=\"0\"><tr><td><br /><b>British Pound - euro</b> . . . . ."
        // in front of UAE Dirham is added number 36 so the data will not be validated
+ "<span>UAE 36 Dirham</span><div> Exchange Rate</div> . . . . . script></div>}}";
        barclaysDataV = new ValidateBarclaysData(brcURLData);
        barclaysDataNotV = new ValidateBarclaysData(brcURLData2);

    }

    // . . . . blue dots means that more HTML source is appeared in set up method
```

```

/**
 * tearDown() is called after every single test inside
 * ValidateBarclaysDataTest class is tested and it sets instance variables
 * barclaysDataV and barclaysDataNotV to null
 */
protected void tearDown() {
    barclaysDataV = null;
    barclaysDataNotV = null;
}
/**
 * testing getData() method.
 */
public void testGetData(){
    assertNotNull(barclaysDataV.getData());
    assertNull(barclaysDataNotV.getData());
    String[] currencyData = barclaysDataV.getData();
    assertEquals("British Pound", currencyData[0]);
    assertEquals("0.65640", currencyData[1]);
    int last = currencyData.length - 1;
    assertEquals("5208.00000", currencyData[last]);
    assertEquals("Zambian Kwacha", currencyData[last - 1]);
}
/**
 * testing getValid() method.
 */
public void testgetValid(){
    assertTrue(barclaysDataV.getValid());
    assertFalse(barclaysDataNotV.getValid());
}
/**
 * testing getErrorMessage() method.
 */
public void testGetErrorMessage(){
    assertNotNull(barclaysDataNotV.getErrorMessage());
    assertNull(barclaysDataV.getErrorMessage());
    assertEquals("Barclays data was not validated!", barclaysDataNotV.getErrorMessage());
}
/**
 * testing ValidateData() method.
 */
public void testValidateData(){
    assertEquals(true, barclaysDataV.validateData("Czech Republic", "35.34895"));
    assertEquals(false, barclaysDataV.validateData("Czech Republic", "35,34895"));
    assertEquals(true, barclaysDataV.validateData("Great Britan", "1"));
    assertEquals(false, barclaysDataV.validateData("Great & Britan", "1"));
    assertEquals(false, barclaysDataV.validateData("Great Britan", "1.0000000000"));

    assertEquals(true, barclaysDataV.validateData("Czech Rep", "12345.123456"));
    assertEquals(true, barclaysDataV.validateData("MY Country", "12345"));
    assertEquals(true, barclaysDataV.validateData("towalec potapec", "140.0990"));
    assertEquals(false, barclaysDataV.validateData("Great & Britan", "1"));
    assertEquals(false, barclaysDataV.validateData("Great Britan", "123456"));
    assertEquals(false, barclaysDataV.validateData("Great Britan", "123,1234567"));
}
}
}

```

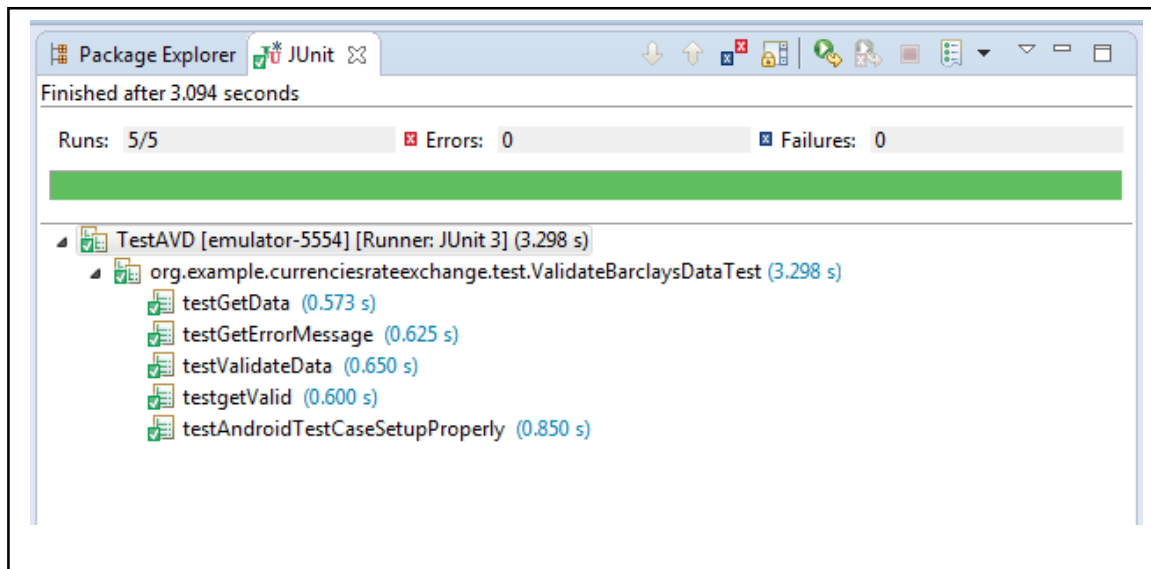


Figure A.9: The running unit test result of ValidateBarclaysDataTest Test class.

## ValidateRBSData.java

```
package org.example.currenciesrateexchange.test;
import org.example.currenciesrateexchange.ValidateRBSdata;
import android.test.AndroidTestCase;
/**
 * ValidateRBSdataTest class enable to test all methods in
 * ValidateRBSData class in CurrenciesRateExchange file
 *
 * @author dsajdl01 (David Sajdl)
 * @version (01/03/2015)
 */
public class ValidateRBSdataTest extends AndroidTestCase {
    private ValidateRBSdata rbsDataV;
    private ValidateRBSdata rbsDataNotV;

    public ValidateRBSdataTest(){
    }
    /**
     * setUp() method is called before every single test inside
     * ValidateRBSdataTest class is tested and it creates two instances of the
     * ValidateRBSData class plus it sets instance variables with HTML RBS source value
     */
    protected void setUp() {
        String url = "<!DOCTYPE html><div id=\"mid\"><h1 . . . . .
        DOLLARS</td><td>2.04</td><td>1.78</td></tr><tr><td>BAHRAINIAN
        DINARS</td><td>0.65</td><td>0.57</td></tr><tr><td>BARBADOS
        DOLLARS</td><td>3.42</td><td>2.86</td></tr><tr><td>BRAZILIAN
        REAL</td><td>4.54</td><td>3.61</td></tr><tr><td>BRUNEI
        RINGGITS</td><td>2.26</td><td>1.97</td></tr><tr><td>BULGARIAN . . . . .
        <script type=\"text/javascript\"> ";
        // in front of EURO is added € sign so the data will not be validated
        String url2 = "<!DOCTYPE html><div id=\"mid\"> . . . . .
        <tr><td>€EURO</td><td>1.38</td><td>1.21</td></tr><tr><td>US
        DOLLARS</td><td>1.71</td><td>1.50</td></tr><tr><td>AUSTRALIAN
        DOLLARS</td><td>2.04</td><td>1.78</td></tr><tr><td>BAHRAINIAN
        DINARS</td><td>0.65</td><td>0.57</td></tr><tr><td>BARBADOS
        DOLLARS</td><td>3.42</td><td>2.86</td></tr><tr><td>BRAZILIAN
        REAL</td><td>4.54</td><td>3.61</td></tr><tr><td>BRUNEI . . . . . <script
        type=\"text/javascript\"> ";
        // . . . . blue dots means that more HTML source is appeared in set up method
    }
```

```

        rbsDataV = new ValidateRBSdata(url1);
        rbsDataNotV = new ValidateRBSdata(url2);
    }
    /**
     * tearDown() is called after every single test inside
     * ValidateRBSdataTest class is tested and it sets instance variables
     * rbsDataV and rbsDataNotV to null
     */
    protected void tearDown() {
        rbsDataV = null;
        rbsDataNotV = null;
    }
    /**
     * testing getData() method.
     */
    public void testGetData(){
        String[] rbsDataArray = rbsDataV.getData();
        assertNotNull(rbsDataArray);
        assertEquals("EURO",rbsDataArray[0].trim());
        assertEquals("1.38",rbsDataArray[1].trim());
        assertEquals("1.21",rbsDataArray[2].trim());
        int lastIndex = rbsDataArray.length -1;
        assertEquals("5.29",rbsDataArray[lastIndex - 1].trim());
        assertEquals("6.36",rbsDataArray[lastIndex - 2].trim());
        assertEquals("UAE DIRHAMS",rbsDataArray[lastIndex - 3].trim());
        assertNull(rbsDataNotV.getData());
    }
    /**
     * testing getValid() method.
     */
    public void testgetValid(){
        assertTrue(rbsDataV.getValid());
        assertFalse(rbsDataNotV.getValid());
    }
    /**
     * testing getErrorMessage() method.
     */
    public void testGetErrorMessage(){
        assertNull(rbsDataV.getErrorMessage());
        assertEquals("RBS data is not valiaded!",rbsDataNotV.getErrorMessage());
    }
    /**
     * testing validateData() method.
     */
    public void testValidateData(){
        assertTrue(rbsDataV.validateData("Fred Shith", "2.89", "19.99"));
        assertTrue(rbsDataV.validateData("USA &iuy; EURO", "123456.8901", "654321"));
        assertTrue(rbsDataV.validateData("woow &tdr; mow", "1", "0.9941"));

        assertFalse(rbsDataV.validateData("Fred Shith", "2.89969", "19.99"));
        assertFalse(rbsDataV.validateData("USA &iuy;", "123456.8901", "654321"));
        assertFalse(rbsDataV.validateData("woow & mow", "1", "0.9941"));
        assertFalse(rbsDataV.validateData("Fred Shith", "2.89", "1,999"));
        assertFalse(rbsDataV.validateData("USA", "12.8.9", "654"));
        assertFalse(rbsDataV.validateData("woow", "1", "0.99412"));
        assertFalse(rbsDataV.validateData("woow &tdr mow", "1", "0.99"));
    }
}

```

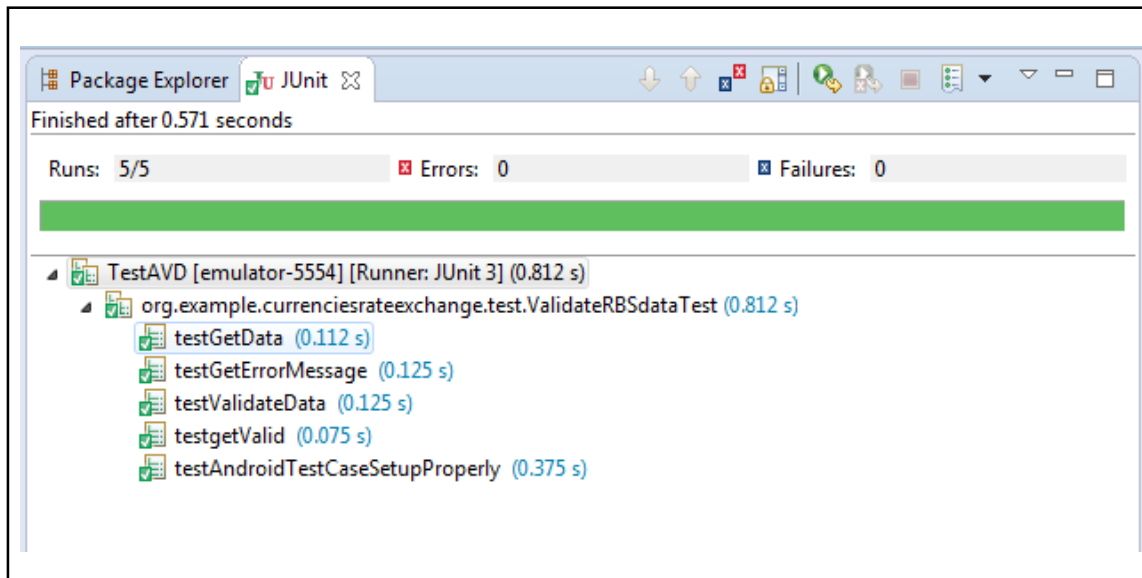


Figure A.10: The running unit test result of ValidateRBSdataTest class class.

### ValidationXRateDate.java

```
package org.example.currenciesrateexchange.test;
import org.example.currenciesrateexchange.ValidateXRateData;
import android.test.AndroidTestCase;
/**
 * ValidateXRateDataTest class enable to test all methods in
 * ValidateXRateData class in CurrenciesRateExchange file
 *
 * @author dsajdl01 (David Sajdl)
 * @version (01/03/2015)
 */
public class VatidateXRateDateTest extends AndroidTestCase {
    private ValidateXRateData vd;
    private ValidateXRateData notVd;

    public VatidateXRateDateTest(){

    }
    /**
     * setUp() method is called before every single test inside
     * ValidateXRateDataTest class is tested and it creates two instances of the
     * ValidateXRateData class plus it sets instance variables with HTML source value
     */
    protected void setUp() {
        String data = " <div class='moduleContent'> <h1 class='basicH1 ratestableH1'>
        <span class='OutputHeader'>Rates Table</span><a href='#converter'
        class='backToConverterButton'>Converter</a> <tbody> <tr> <td>Euro</td> <td
        class='rtRates'><a href='/graph/?from=GBP& to=EUR'>1.277233</a></td> <td
        class='rtRates'><a href='/graph/?from=EUR& to=GBP'>0.782943</a></td> . .
        . . .
        href='/graph/?from=USD& to=GBP'>0.652422</a></td></tr><tr><td> Venezuelan
        Bolivar</td> <td class='rtRates'> <a href='/graph/?from=GBP& to=VEF'>
        9.656708</a></td><td class='rtRates'><a href='/graph/?from=VEF&
        to=GBP'>0.103555</a></td></tr></tbody>
        </table><script
        type='text/javascript'>;
```

```
// Into 'Venezuelan Bolivar' was added extra dots so the data should not be validated!
9.65.6708
```

```
String data2 = " <div class='moduleContent'> <h1 class='basicH1 ratestableH1'>
<span class='OutputHeader'>Rates Table</span><a href='#converter'
class='backToConverterButton'> Converter</a> <tbody> <tr> <td>Euro</td> <td
class='rtRates'><a href='/graph/?from=GBP&to=EUR'>1.277233</a> . . . . .
tr><td>Venezuelan Bolivar</td> <td class='rtRates'><a href='/graph/?
from=GBP&to=VEF'>9.65.6708</a></td><td class='rtRates'> <a
href='/graph/?from=VEF&to=GBP'>0.103555</a></td></tr></tbody></table>
<script type='text/javascript'>";
vd = new ValidateXRateData(data);
notVd = new ValidateXRateData(data2);
}
/**
 * tearDown() is called after every single test inside
 * ValidateXRateDataTest class is tested and
 * it sets instance variables vd and notVd to null
 */
protected void tearDown() {
    vd = null;
    notVd = null;
}
/**
 * testing getValid() method.
 */
public void testGetValid(){
    assertEquals(true,vd.getValid());
    assertEquals(false,notVd.getValid());
}
/**
 * testing getErrorMessage() method.
 */
public void testGetErrorMessage(){
    assertNull(vd.getErrorMessage());
    assertNotNull(notVd.getErrorMessage());
    assertEquals("x-Rate data was NOT validated!",notVd.getErrorMessage());
}
/**
 * testing getData() method.
 */
public void testGetData(){
    String[] dataArray = vd.getData();
    assertEquals("Argentine Peso",dataArray[0]);
    assertEquals("13.105013",dataArray[1]);
    assertEquals("9.656708",dataArray[dataArray.length - 1]);
    assertEquals("Venezuelan Bolivar",dataArray[dataArray.length - 2]);

    String[] notValDataArray = notVd.getData();
    assertNull(notValDataArray);
}
/**
 * testing ValidateDate() method.
 */
public void testValidateData(){
    assertEquals(true,vd.validateData("Czech Republic","35.34895"));
    assertEquals(false,vd.validateData("Czech Republic","35,34895"));
    assertEquals(true,vd.validateData("Great Britan","1"));
    assertEquals(false,vd.validateData("Great & Britan","1"));
    assertEquals(false,vd.validateData("Great Britan","1.0000000000"));
}
}
```

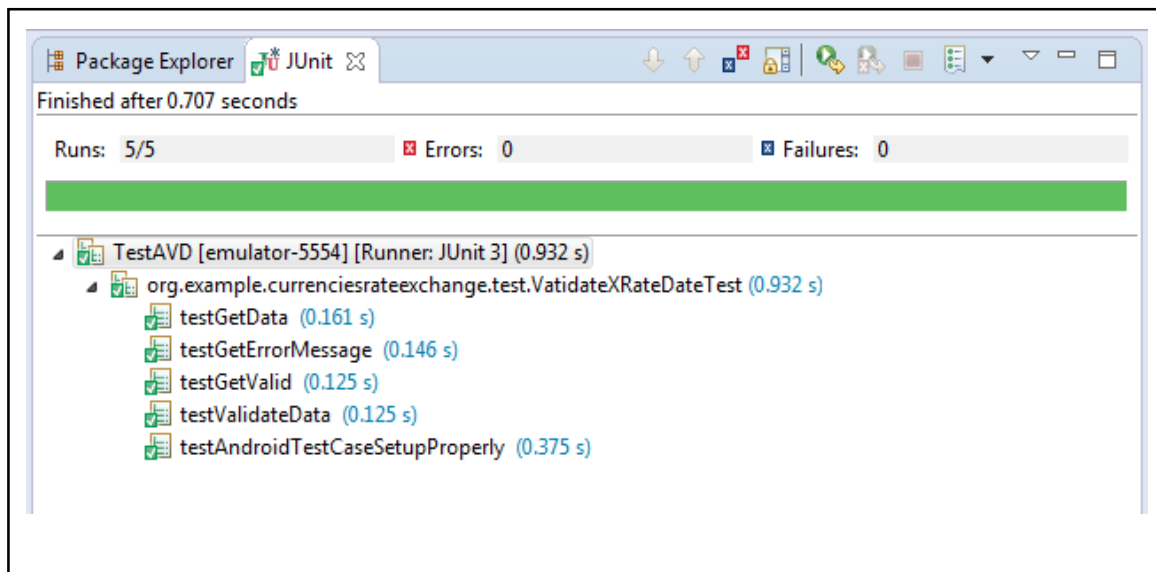


Figure A.11: The running unit test result of ValidateXRateDataTest class.

## References

### Books and PDF file used while I works on this project include:

- R.1 2014 *Beginning Android Development* Great Britain: Amazon.co.uk, Ltd, Marston Gate
- R.2 Sin George (September 2010) *The Android Bat-Phone* Birkbeck College, University of London: Department of Computer Science and Information Systems
- R.3 Gargenta Marko (2013) *Hello, Android Introducing Google's Mobile development Platform* United States of America: Susannah Davidson Pfalzer
- R.4 Smyth Neil 2014 *Android Studio development Essentials* Great Britain: Amazon.co.uk, Ltd, Marston Gate
- R.5 Simon Bennett, Steve McRobb, Ray Farmer (2010) *Object-Oriented System Analysis and Design Using UML* United Kingdom: McGraw-Hill Education

### Web links reference used while I works on this project include:

- R.6 Developer (2015) *Android Design | Market Developers* Available from < <http://developer.android.com/design/material/index.html> > [Accessed on 12/03/2015]
- R.7 EX (2015) *XE Currency Apps* Available from < <http://www.xe.com/apps/> > [Accessed on 10/02/2015]
- R.8 go-Digital Blog on Digital Marketing (2014) *chinese manufacturers >> go-Digital Blog on Digital Marketing* Available from < <http://go-digital.net/blog/tag/chinese-manufacturers/> > [Accessed on 10/02/2015]
- R.9 HMKCode (2013) *Android Internet Connection Using HTTP GET (HttpClient) | HMKCode* Available from <<http://hmrcode.com/android-internet-connection-using-http-get-httpclient/>>[Accessed on 10/12/2014]
- R.10 IDC Analyze the Future (2014) IDC: Smartphone OS Market Share 2014, 2013, 2012, and 2011 Available from < <http://www.idc.com/prodserv/smartphone-os-market-share.jsp> > [Accessed on 10/02/2015]
- R.11 Oracle (2015) *Reading Directly from a URL (The JavaMT Tutorials > Custom Networking > Working with URLs)* Available from < <http://docs.oracle.com/javase/tutorial/networking/urls/readingURL.html> > [Accessed on 10/12/014]
- R.12 Skill Guru (2014) *Free practice test , mock test, driving test, interview questionsBuilding an android application >> Free practice test , mock test, driving test, interview questions* Available from < <http://www.skill-guru.com/blog/2011/01/10/building-an-android-application/>> [Accessed on 20/11/2014]



- R.13 StackOverflow (2013) *android - What to use instead of "addPreferencesFromResource" in a PreferenceActivity?* - Stack Overflow Available from < <http://stackoverflow.com/questions/6822319/what-to-use-instead-of-addpreferencesfromresource-in-a-preferenceactivity> > [Accessed on 20/12/2014]
- R.14 Techopedia (2015) *What is SQLite? - Definition from Techopedia* Available from < <http://www.techopedia.com/definition/24610/sqlite> > [Accessed on 12/03/2015]
- R.15 Vogella (2014) *Android SQLite database and content provider – Tutorial* Available from < <http://www.vogella.com/tutorials/AndroidSQLite/article.html> > [Accessed on 20/12/2014]
- R.16 Vogella (2014) *Unit Testing with JUnit – Tutorial* Available from < <http://www.vogella.com/tutorials/JUnit/article.html> > [Accessed on 03/03/2015]
- R.17 Wikipedia (2015) *Android (operating system) - Wikipedia, the free encyclopedia* Available from < [http://en.wikipedia.org/wiki/Android\\_%28operating\\_system%29](http://en.wikipedia.org/wiki/Android_%28operating_system%29) > [Accessed on 20/03/2015]
- R.18 Wikipedia (2015) *JAR (file format) - Wikipedia, the free encyclopedia* Available from < [http://en.wikipedia.org/wiki/JAR\\_%28file\\_format%29](http://en.wikipedia.org/wiki/JAR_%28file_format%29) > [Accessed on 12/03/2015]