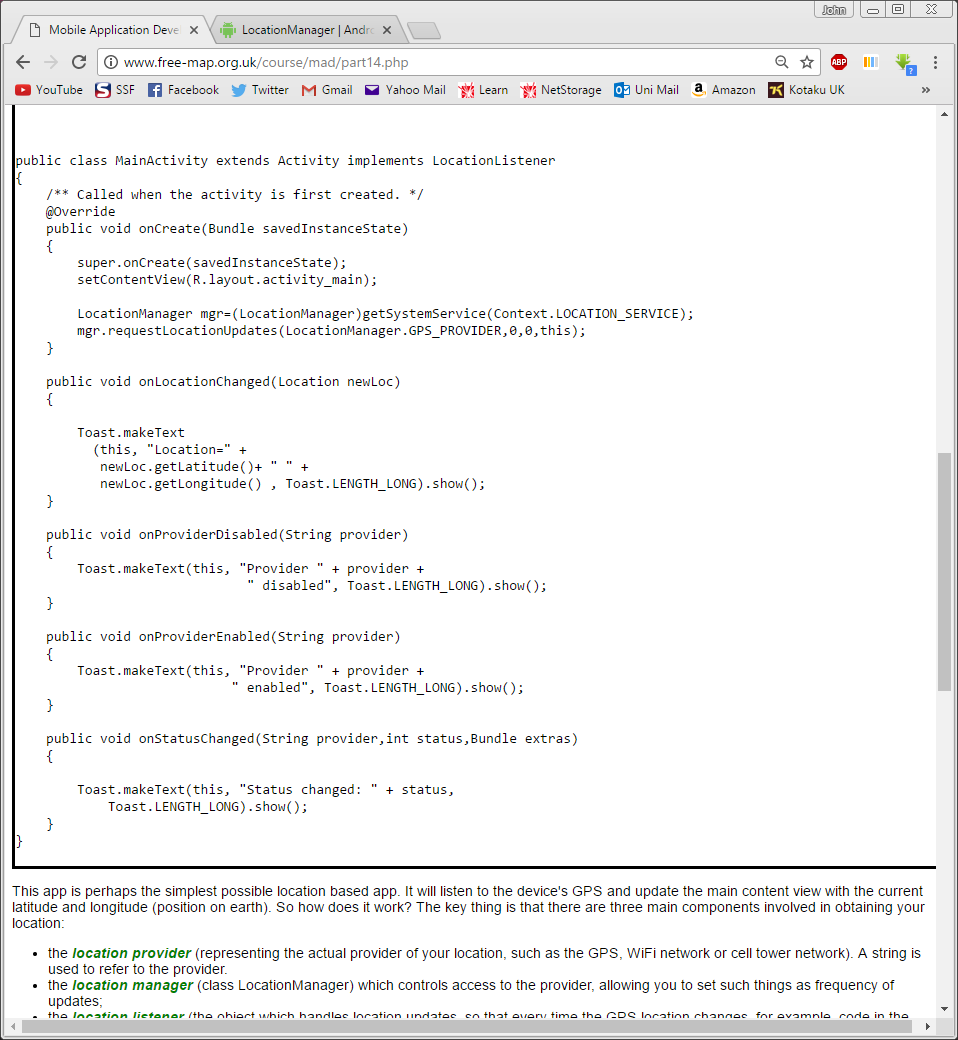
**MAD Report**

|  |
| --- |
| **Requirement 1**  A user should be able to view a map of their current area on their smartphone or tablet. This map should always be positioned at the user's current position on the earth; when the user changes their physical location, the map should reflect this. |

**Analysis:**

For this requirement, the first step will be to implement the ACCESS\_COARSE\_LOCATION and ACCESS\_FINE\_LOCATION permissions in the manifest file. These permissions will allow the app to access the specific location of the user. The LocationManager class will need to be used to periodically obtain updates of the user’s geographical location. My final app will follow a similar code structure shown below, making use of all the same methods, as these are required to react to changes in location and connectivity. The onLocationChanged method is a particularly important method for my app as this will handle the changing of the map relative to the user changing location.



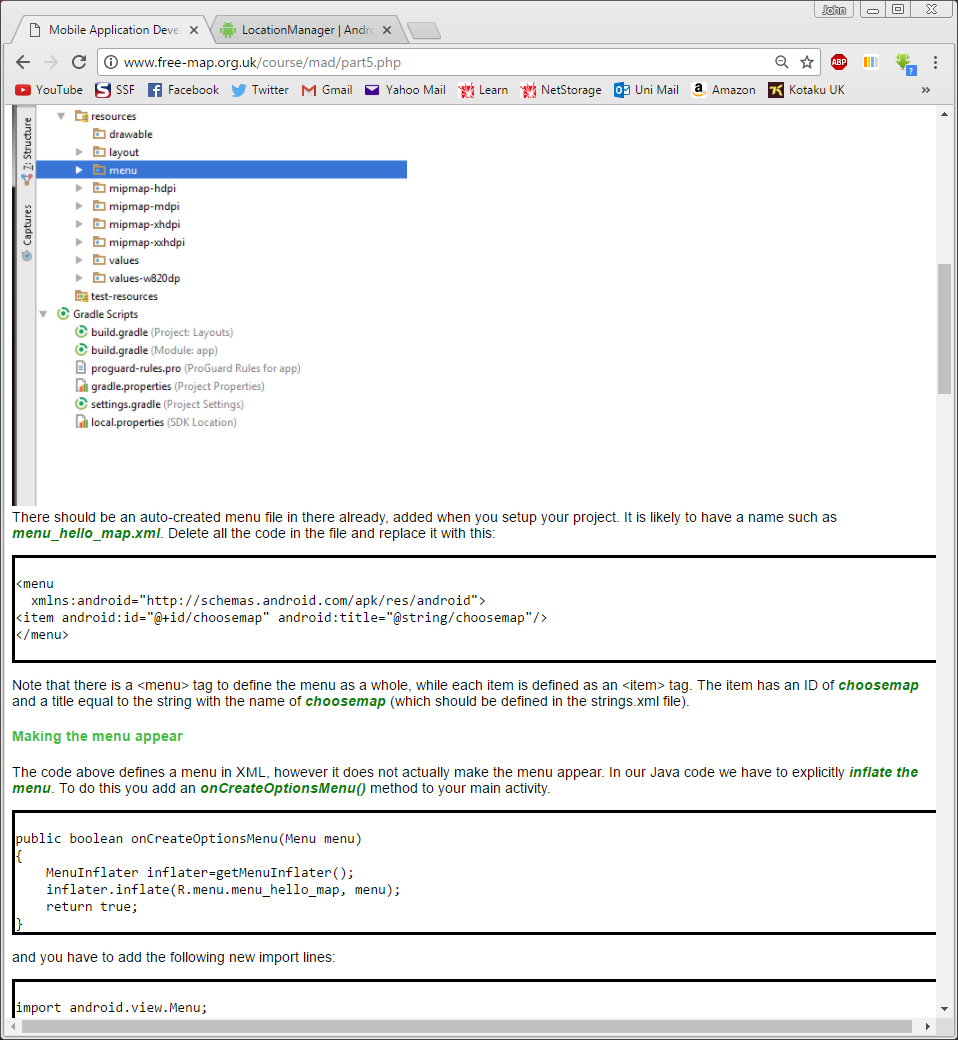
**Problems Encountered:**

Whilst developing the application I encountered a problem where I was unable to test that the functionality for this requirement as using an Android Virtual Device doesn’t automatically retrieve the location. I however overcame this problem by sending virtual GPS signals to the device which contained custom coordinates, which allowed me to view the changes of the map.

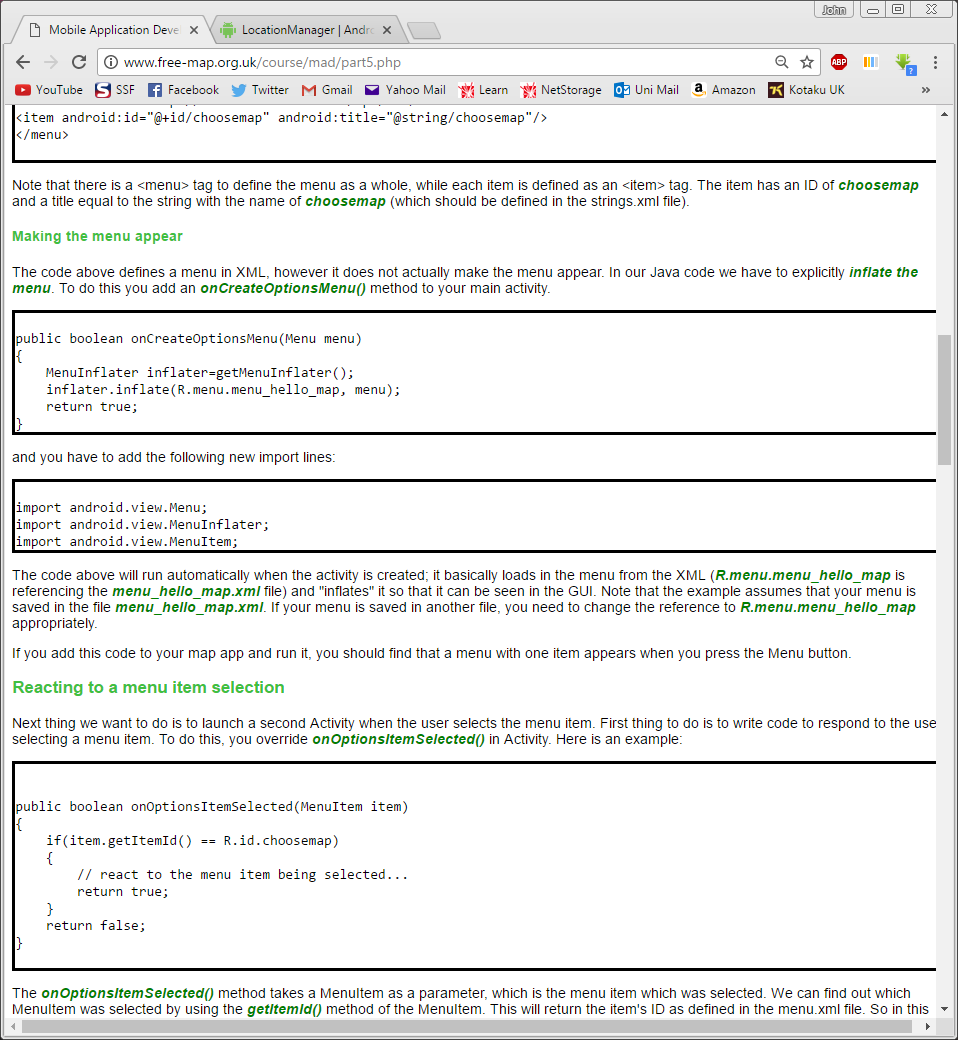
|  |
| --- |
| **Requirement 2**  A user should be able to add a new point of interest (POI) at their current location. The input should be done using a separate activity. The point of interest should appear on the map as a marker as soon as it has been added. |

**Analysis:**

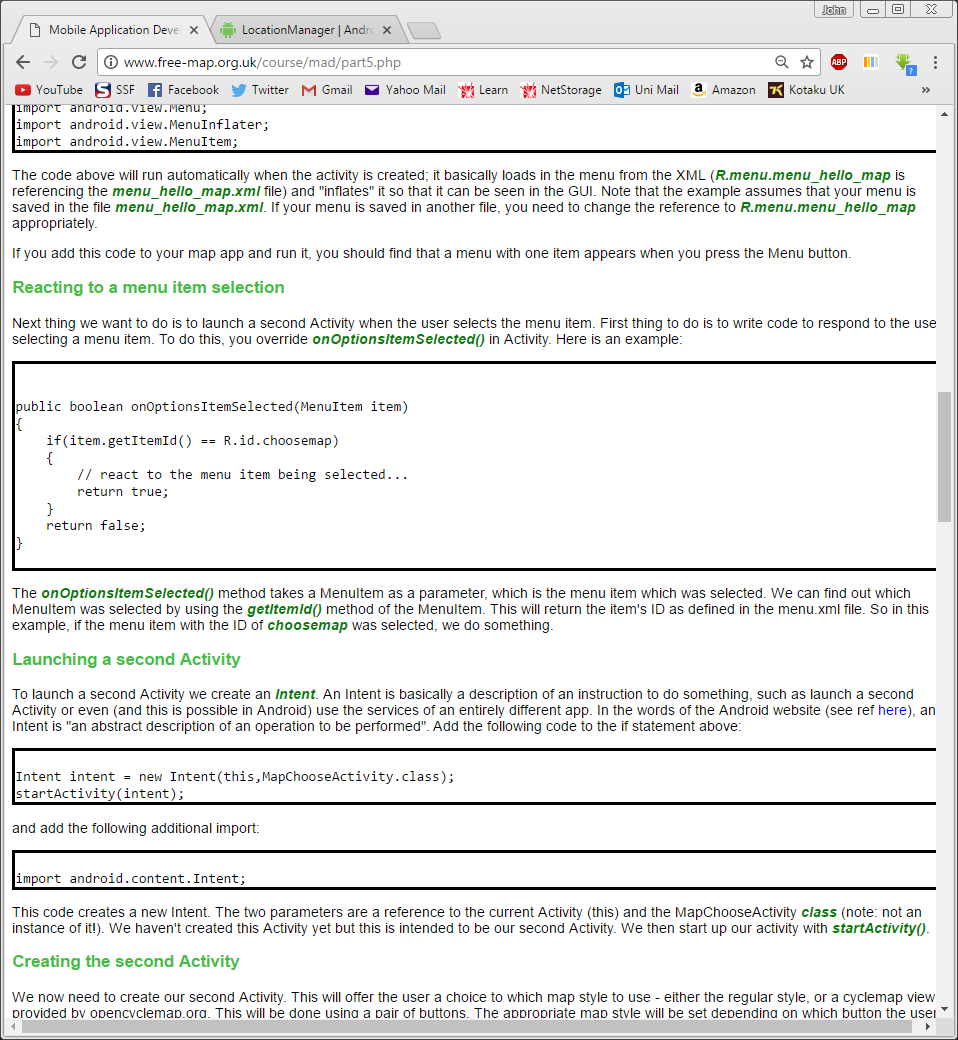
To first access a separate activity from the main Mapping activity I would need a menu to select the option add a new POI. To do this I will need to first create an xml file defining the menu with the use of the <menu> tag and the items to be included in the menu using <item> tags. This should look like this snippet from the lecture notes shown below.



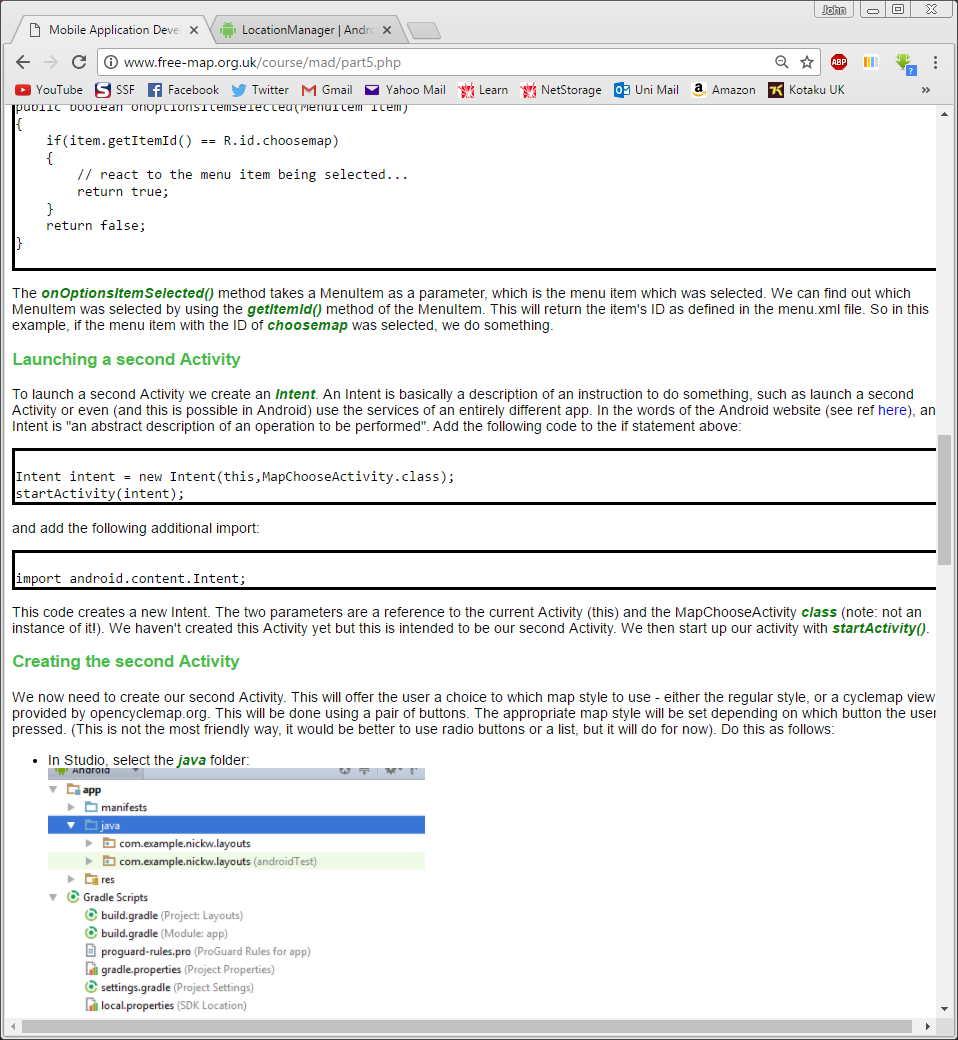
The xml alone will not do anything as I need to inflate the menu using the onCreateOptionsMenu method which will look exactly like this extract from the lecture notes shown below.



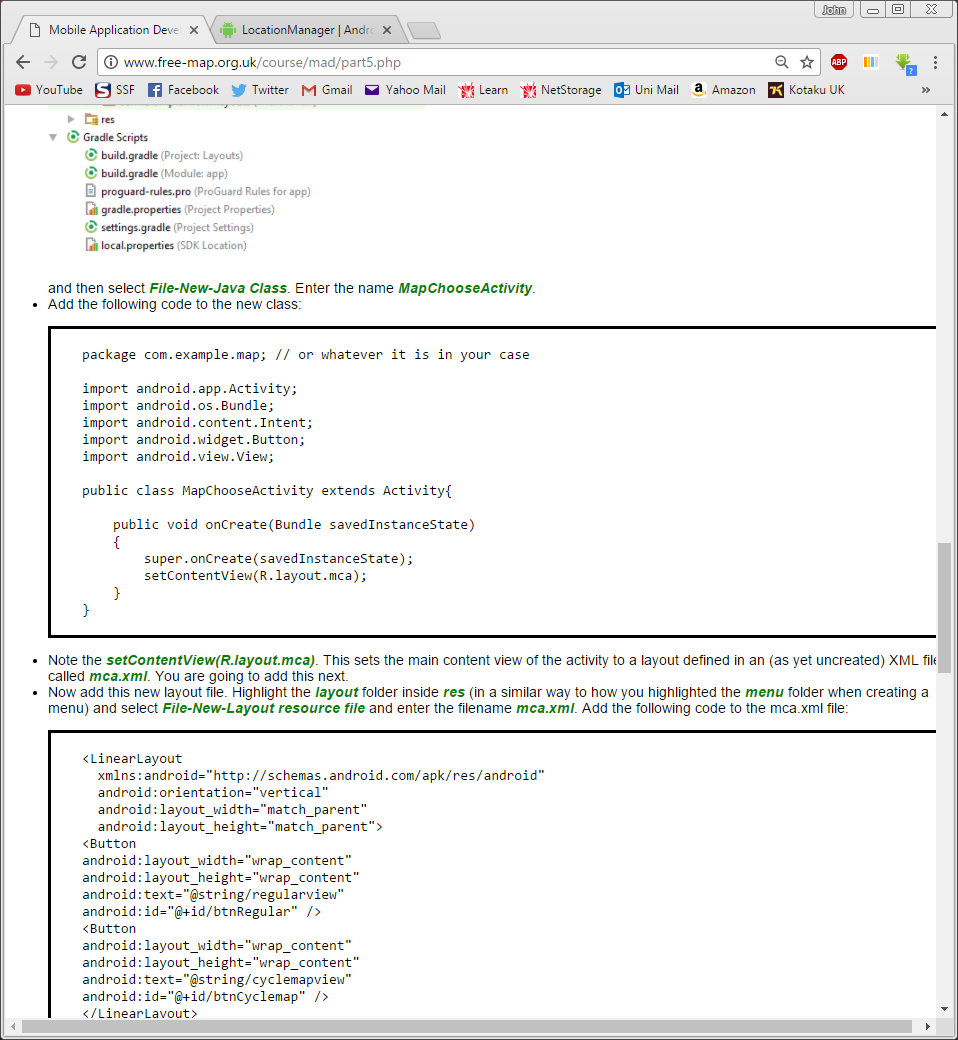
For the app to respond to the respective menu option clicks I will need to use the onOptionsItemSelected method. In my application the code will follow the same structure shown below however will have more if statements for the other requirements that need to be implemented in the app.



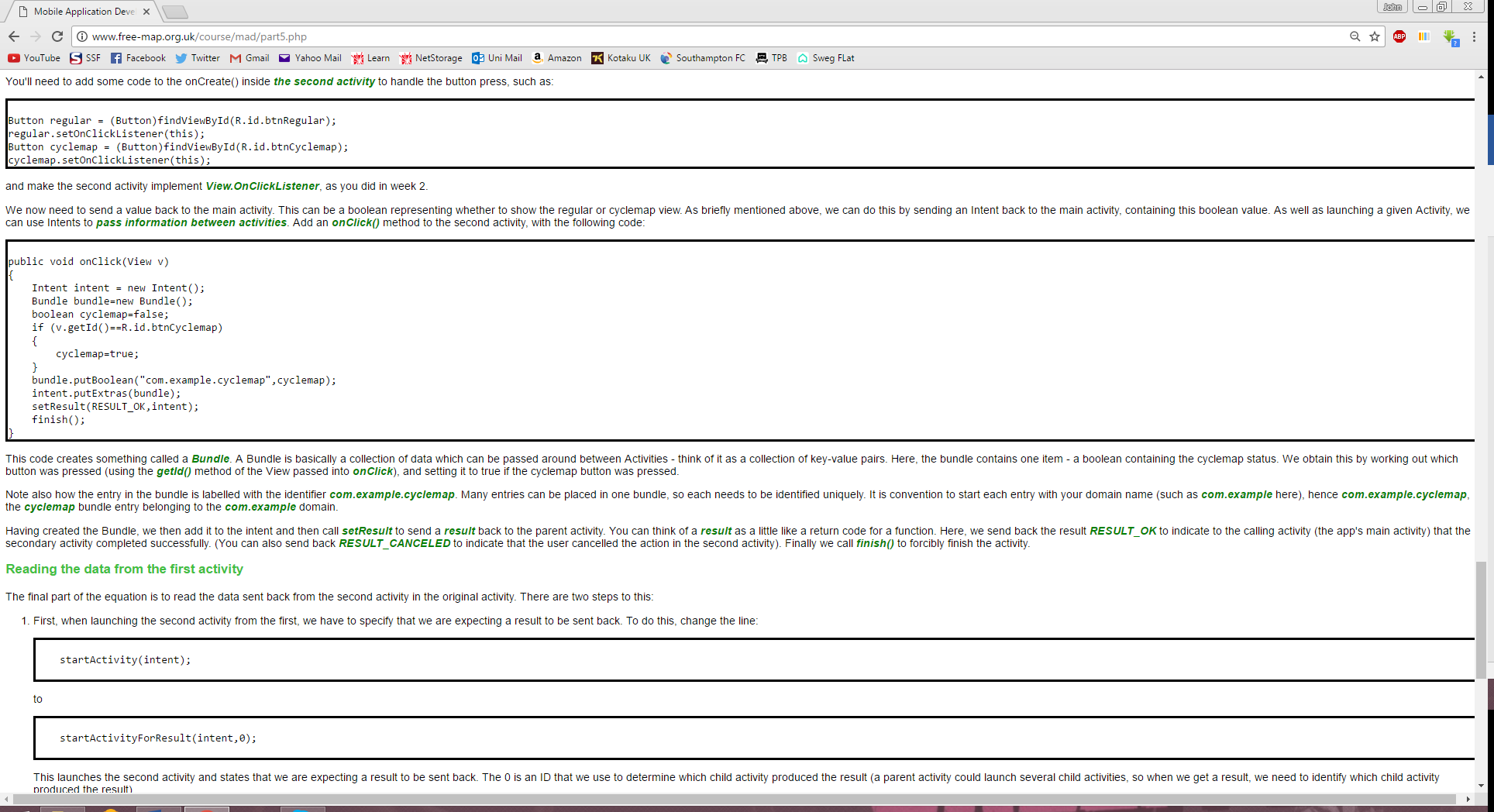
To open the activity containing the add a POI option I will need to create an Intent. To do this I will need to add the code shown below in the if statements mentioned previously. I will obviously also need to change the class which is being accessed.



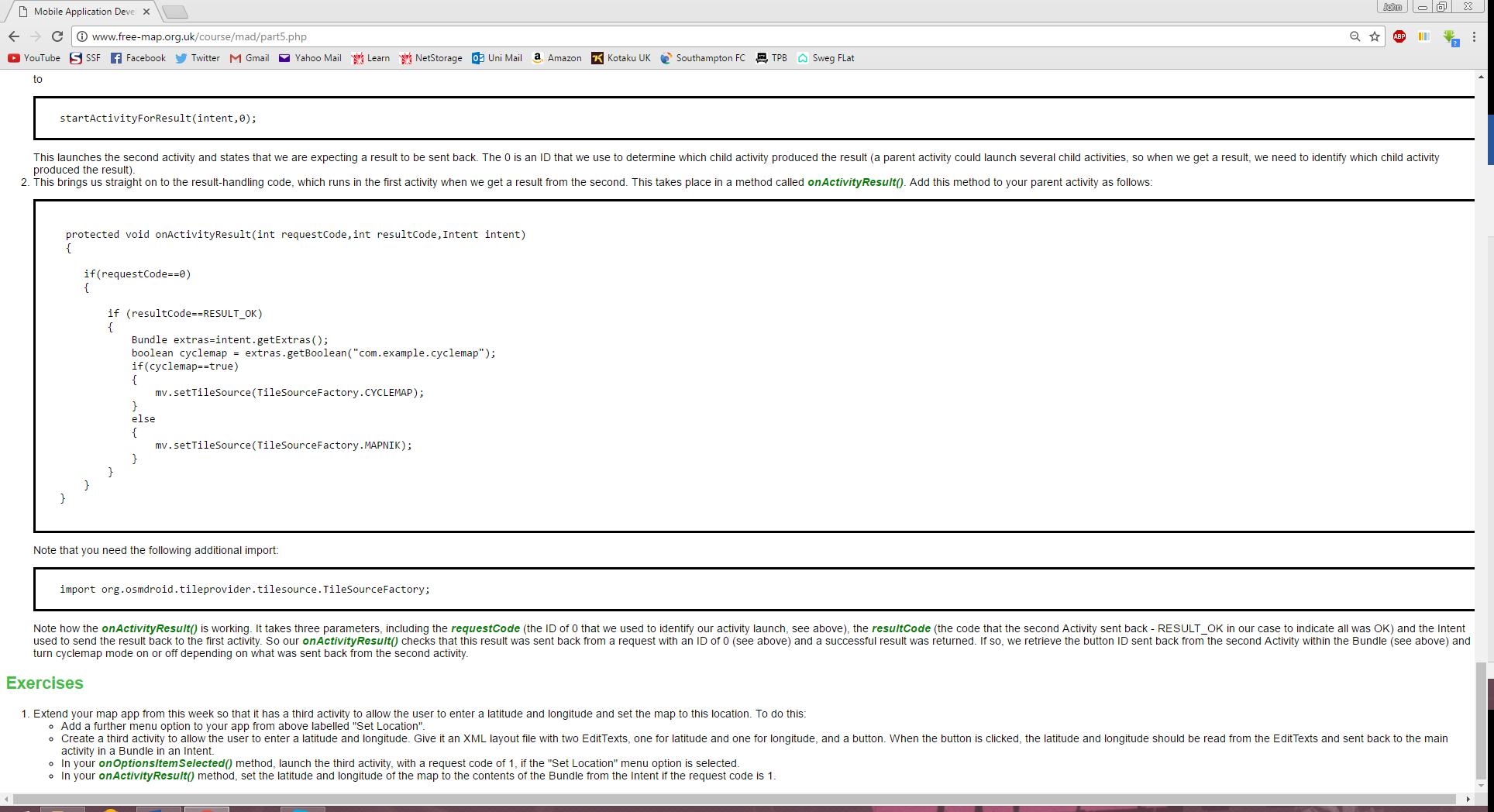
I will then need to create the layout xml file for the adding POI activity. An appropriate interface for the user to input the POI name, type and a description will need to be implemented. For the name and description, I will use text boxes but the type will be selected from a spinner. The activity will also need its own class containing all the appropriate methods to display the interface and to react to submission of a POI. To display the interface, I would need to use the setContentView method shown below in the class onCreate method.



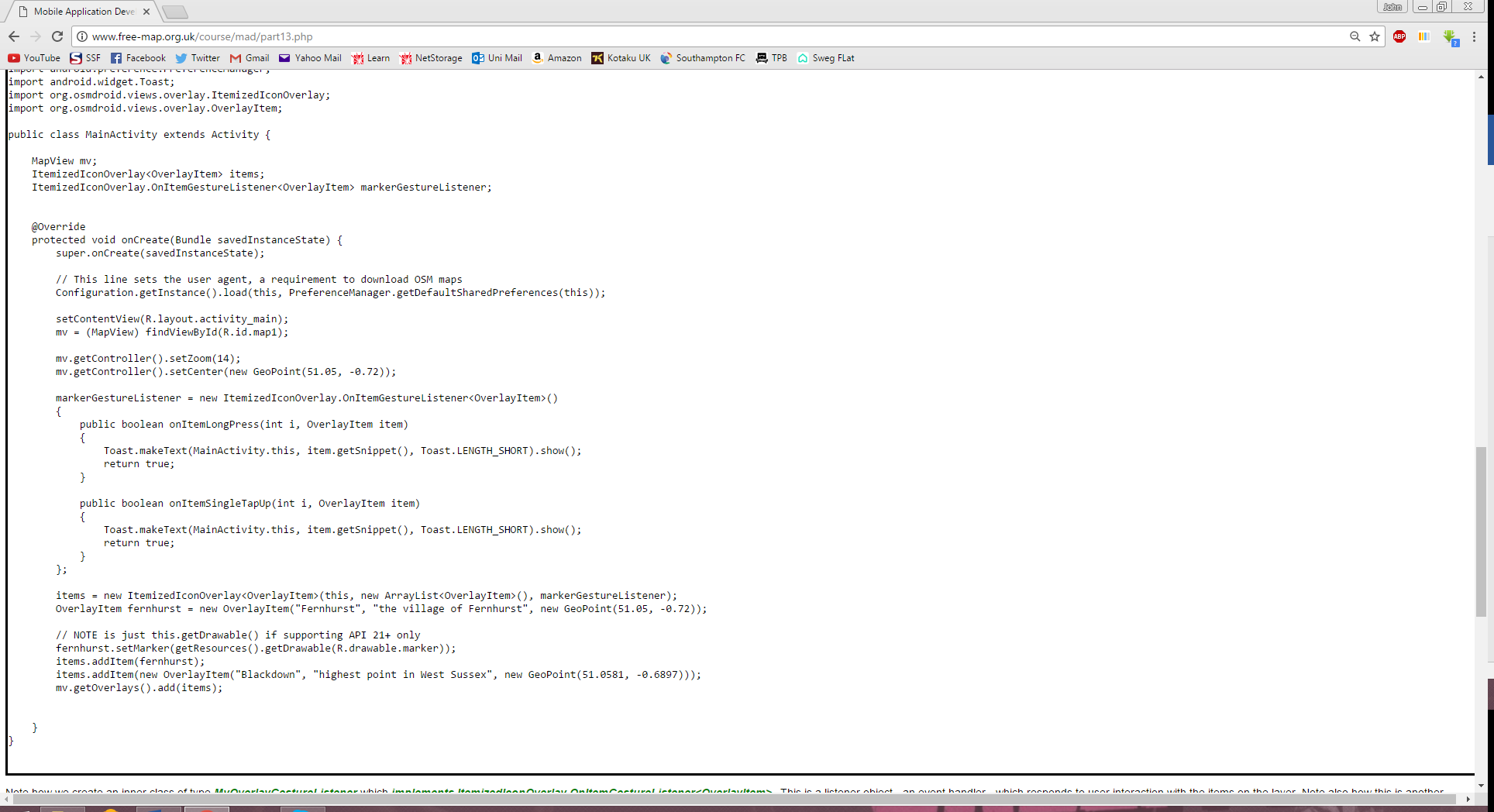
To display the users POI on the map as a marker I will first need to create an onClick method so that the submit button functions. I will also need to use a bundle and a new intent to store the details of the POI and pass it to the map activity where the marker will be displayed. Once completed it should look like this extract below.



I will then need to access information such as latitude and longitude in the bundle from the first activity which will be done using onActivityResult method like the extract shown below. The extract also shows how to retrieve the extras put into the bundle.



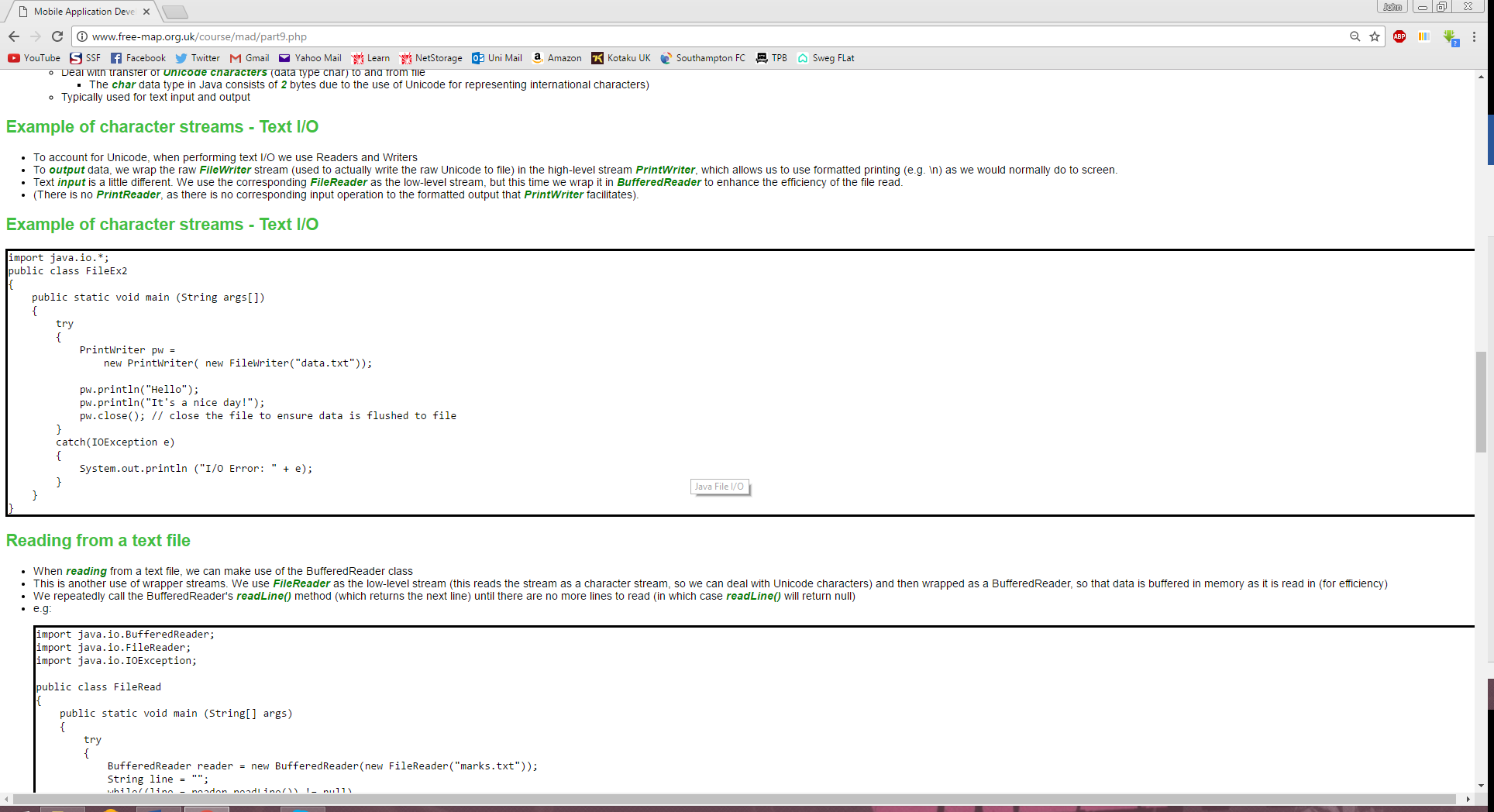
I can then use the information from the bundle to create markers with marker events associated with them. This is done using code like what is shown below.



|  |
| --- |
| **Requirement 3**  A user should be able to save all points of interest added to file. Also, when the user closes down the app, or the app is automatically closed down by the system, all points of interest should be saved to file automatically. |

**Analysis:**

To fulfil this requirement, I will need to use a combination of the FileWriter and PrintWriter character streams to write the POI details to a file saved on the user’s device. This will look like the extract shown below however the file will be written in CSV or JSON format so that the data can be used later. To ensure the file is saved when the application is closed I will use the onDestroy method which is part of the Activity lifecycle.

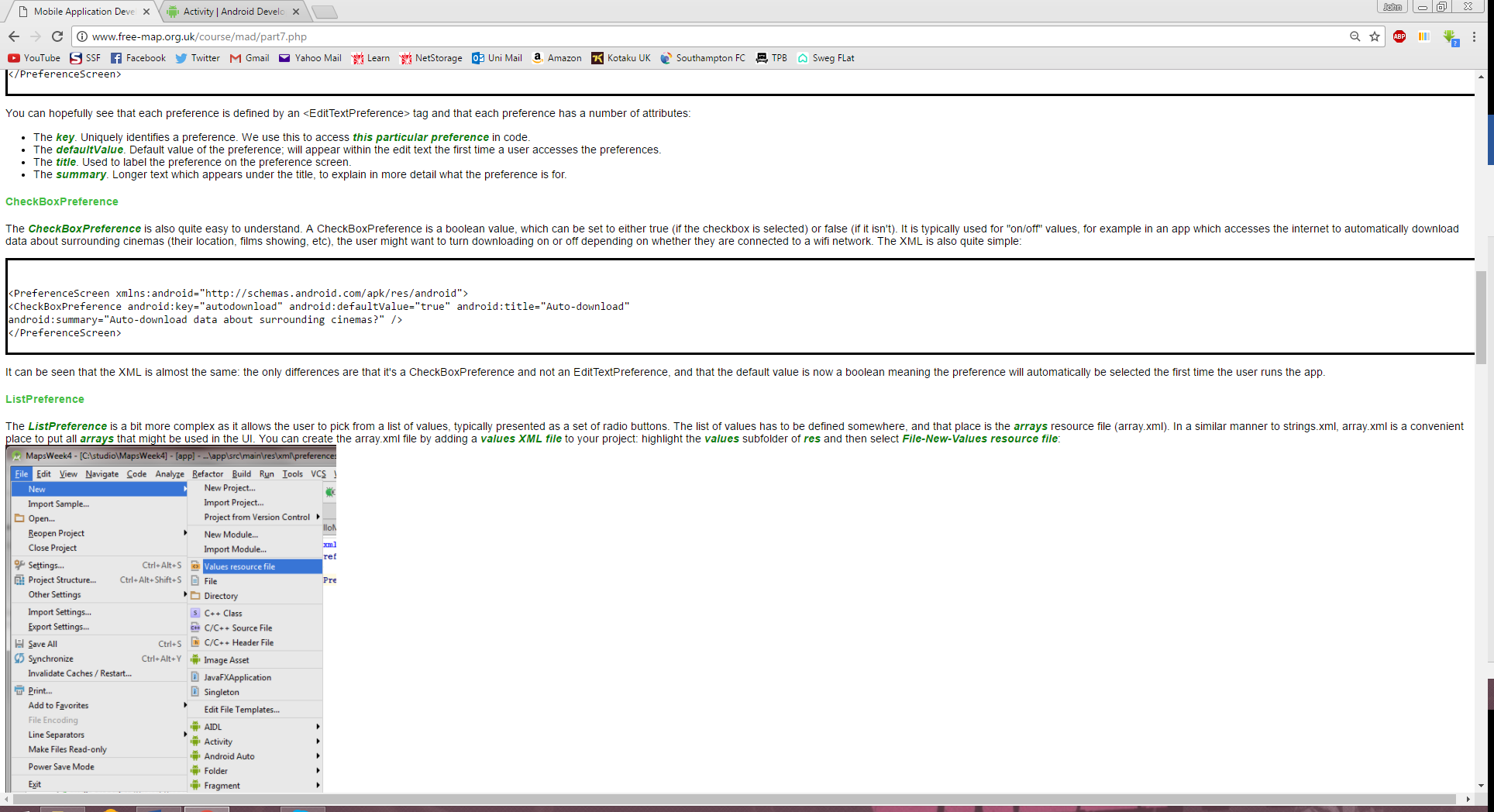


**Problems Encountered:**

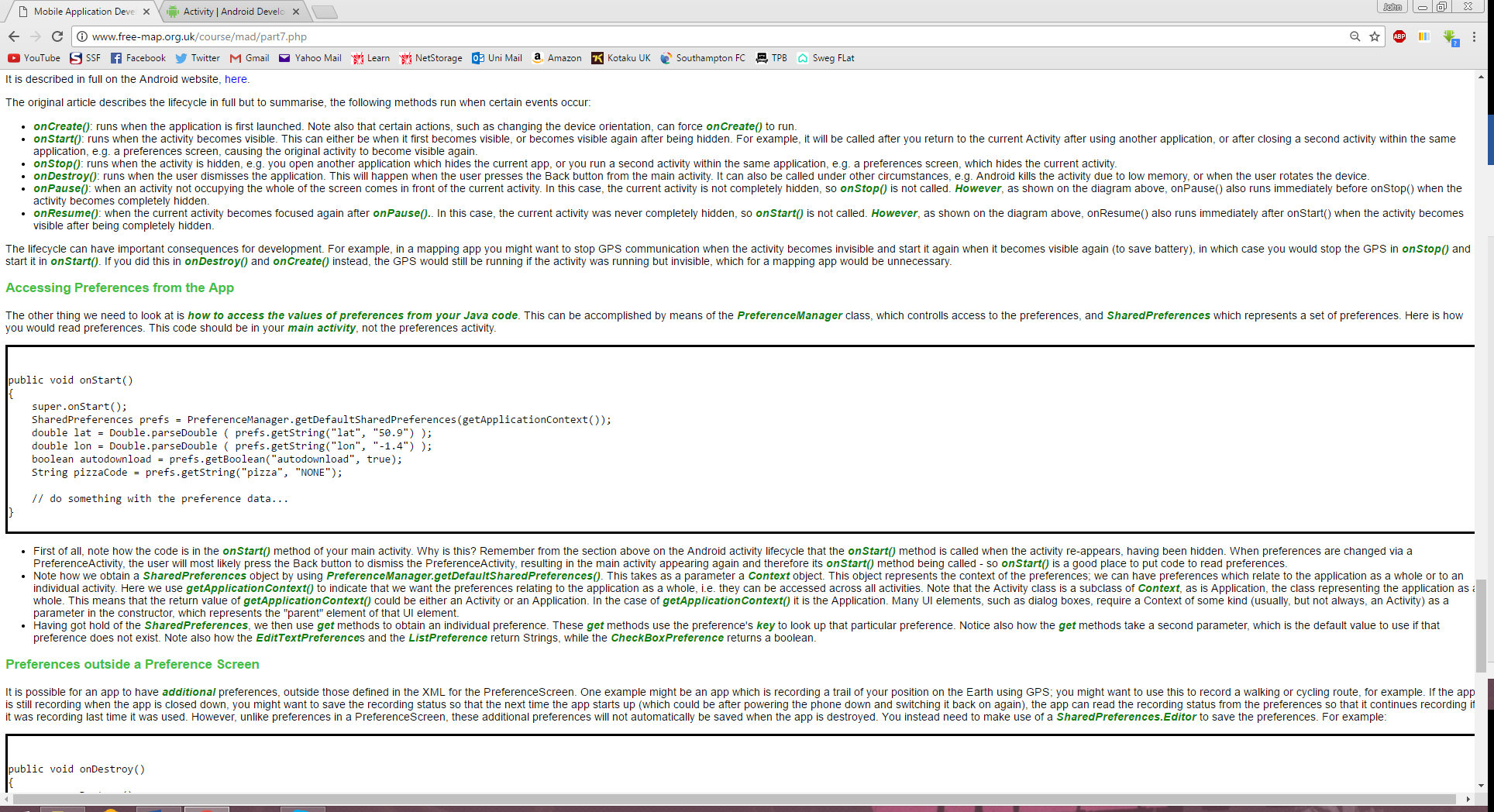
I initially tried saving the file in the onDestroy method to meet the requirement of saving the file when the app is closed however it didn’t work. To overcome this problem, I tried putting the code to save the file in the onStop method and it worked. This is because the save was being executed too late in the activity lifecycle and therefore the file wasn’t being saved.

|  |
| --- |
| **Requirement 4**  The user should be able to set a preference which determines whether points of interest are uploaded to the web when added. You just need to implement the preference, and add the appropriate “if” statement in your Java to test whether the preference was set. |

As preferences uses data which is preserved across multiple executions of an application a normal activity will not be a viable for this requirement. Instead a PreferenceScreen xml file is needed which is controlled by a PreferenceActivity. There are different preference screens which can be used depending on the preference which is going to be chosen by the user. As this case is a simple Boolean value it will be a checkbox preference screen in my application. The structure of the xml file will be like what is shown below.

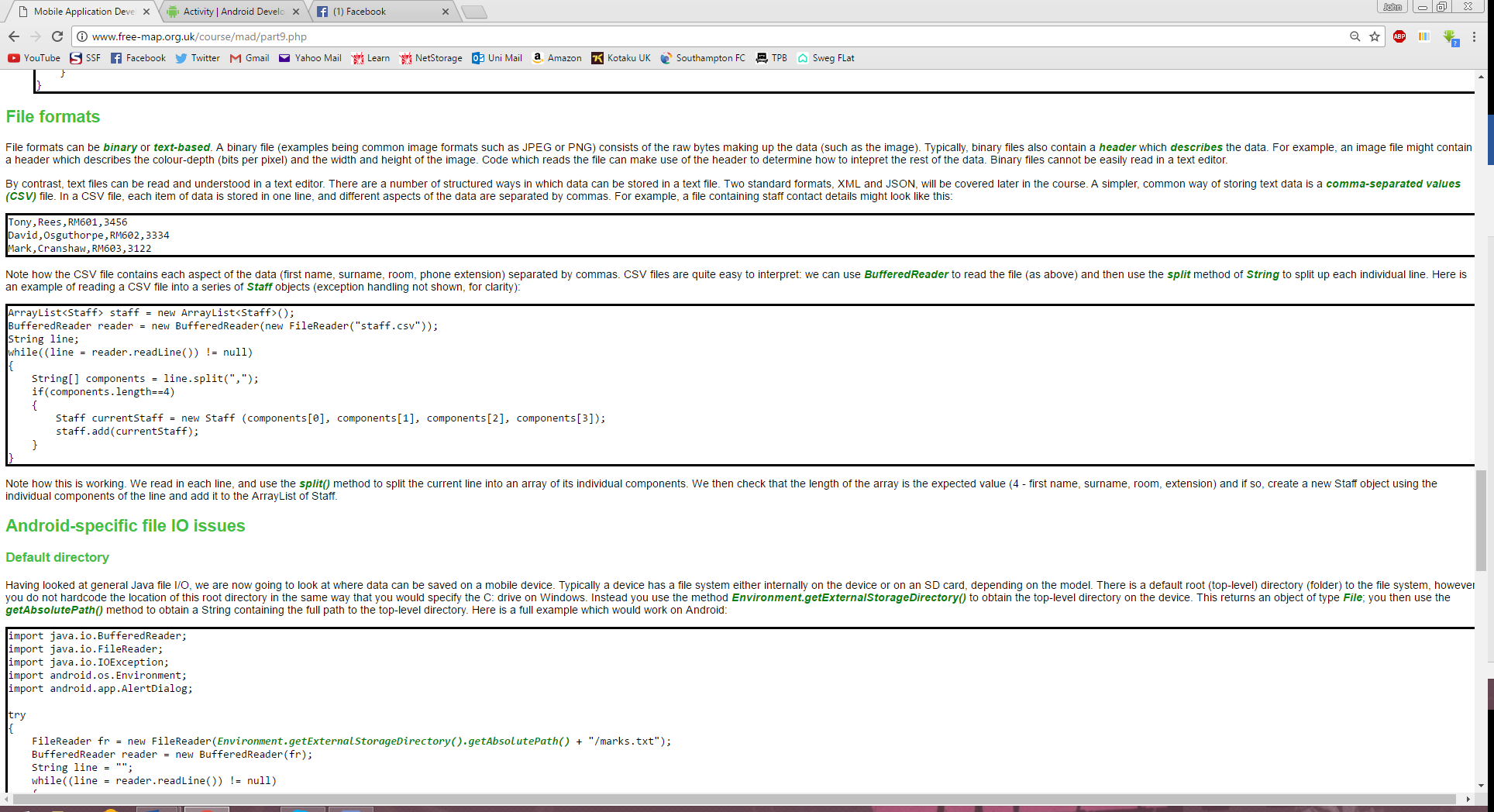


To react to the preference being selected, the value of the preference needs to be accessed in the Java code. This is done using the PreferenceManager and SharedPreferences classes. The code will look like this extract from the lecture notes.



|  |
| --- |
| **Requirement 5**  A user should be able to load all existing points of interest from file. These should be displayed on the map as markers. |

This requirement is very like requirement 3 as they both use File I/O however in this case the BufferedReader class is used with a FileReader stream. By using the readLine Method of the BufferedReader I can read in a document line by line. Depending on the format of the file the attributes listed in the file can be interpreted as strings and used in the Java code. If the file being read in is a CSV file the code structure will look like the code extract shown below.

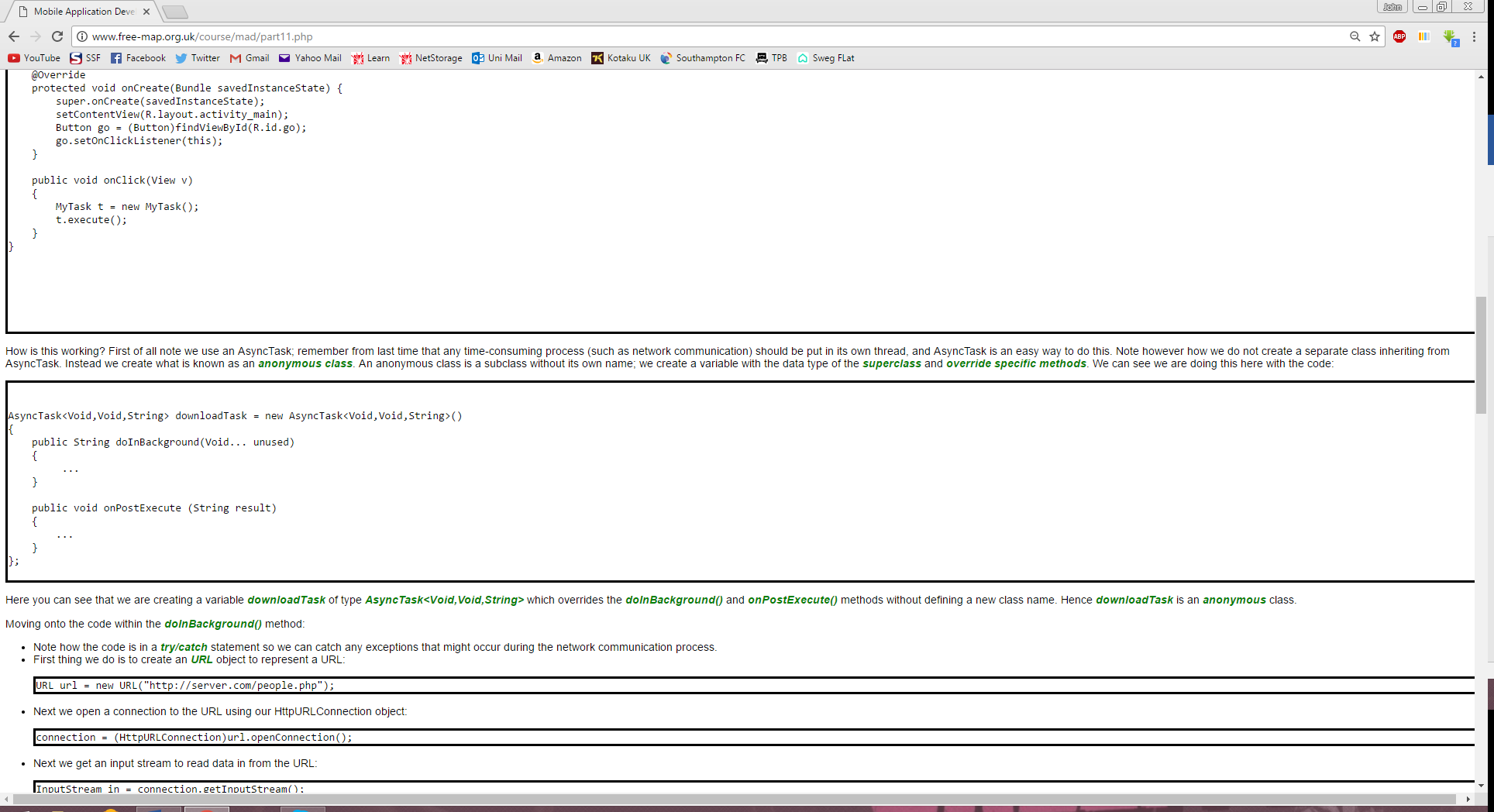


By reading in the POI information from the files, the data can be used to create map markers. This will need to be done using a loop so that as the file is being read the markers are being created.

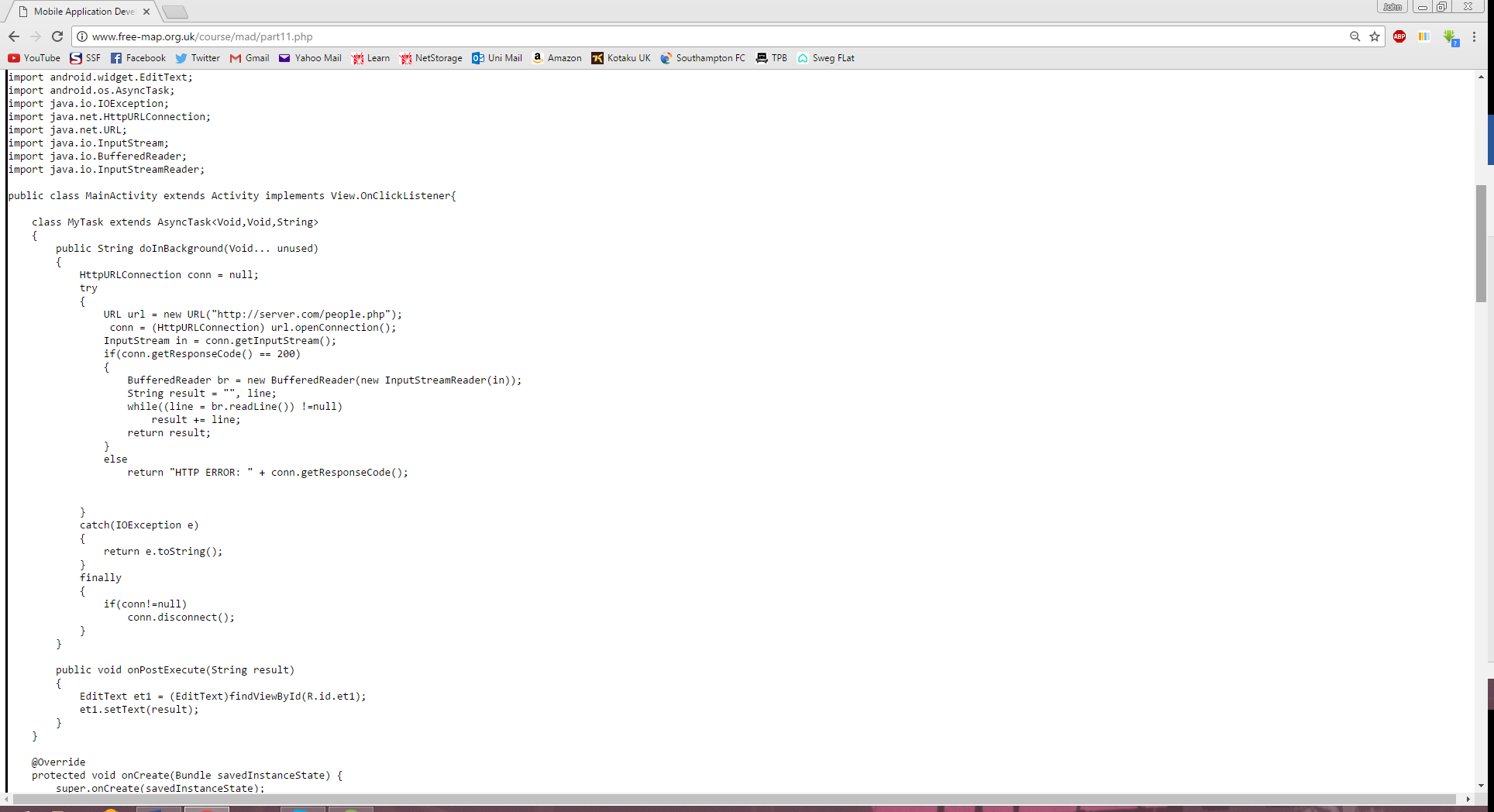
|  |
| --- |
| **Requirement 6**  The user should be able to load points of interest from the web. |

**Analysis:**

The first step in solving this requirement will be ensuring that the network communication required in this task is put in its own thread using an AsyncTask as it could be a time-consuming process. I will create an anonymous class to connect to the web service hosting the points of interests. In this class, there will be two methods doInBackground and onPostExecute as shown below.



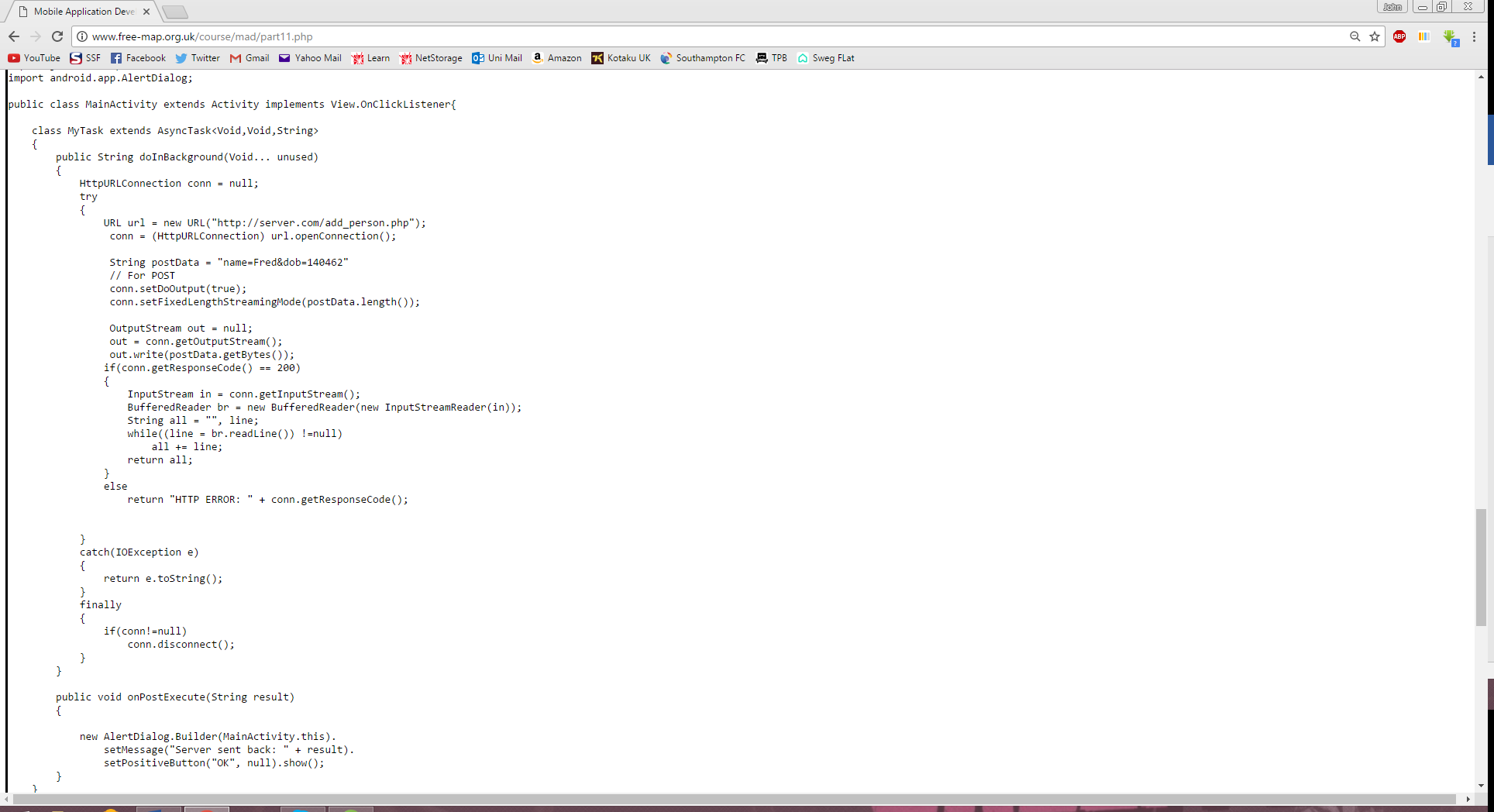
The doInBackground method will connect to the web service, load and handle the incoming data and the onPostExecute method will be used to load the POI onto the map as Overlay Items as done in previous requirements.



|  |
| --- |
| **Requirement 7**  If the appropriate preference is set (see requirement 4), a point of interest should be saved to the web after it has been added to the map. |

**Analysis:**

Following on from Requirement 4, using the SharedPreferences and PreferenceManager classes I can then upload a POI once it has been added. This will be done using an if statement to check if the option has been selected and if it has then I can save then save the POI to the web in a similar fashion to Requirement 6 but rather than using an InputStream I will use an OutputStream as seen below.

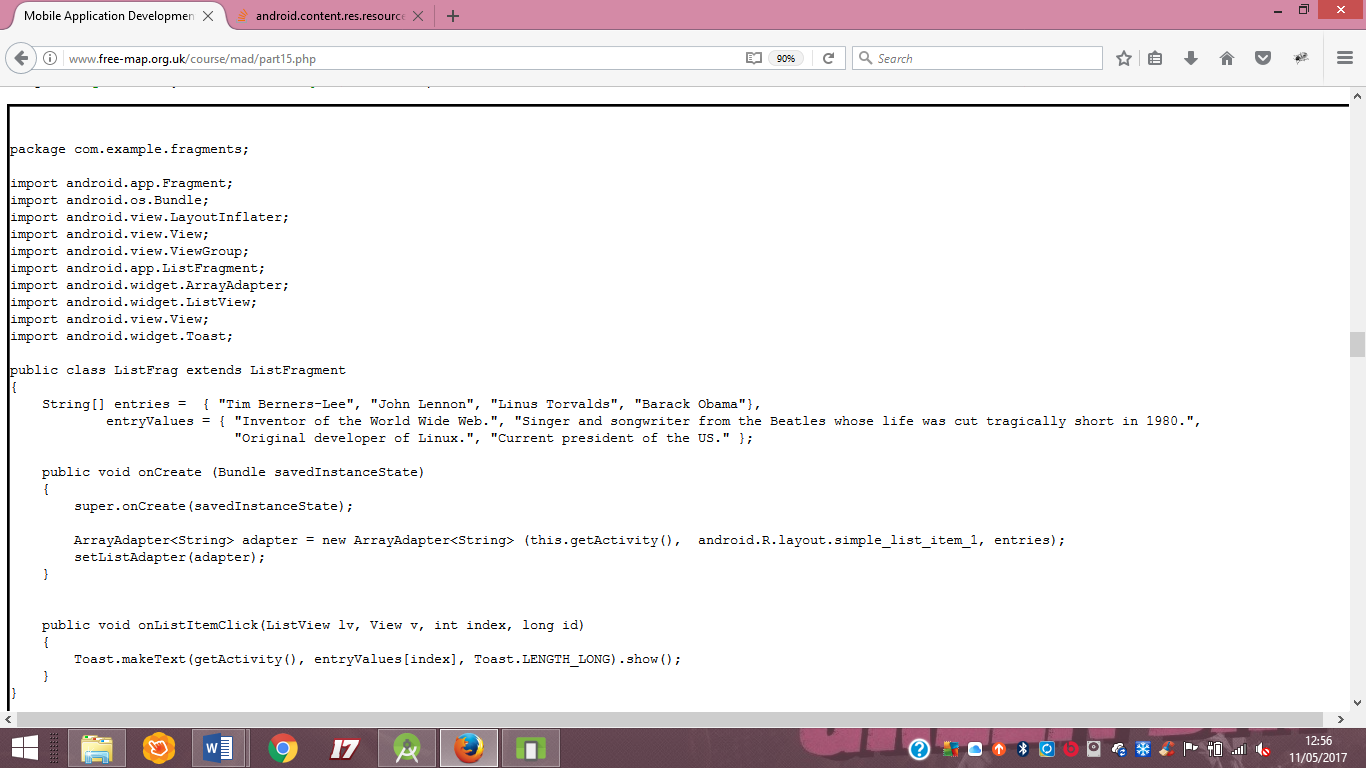


**Problems Encountered:**

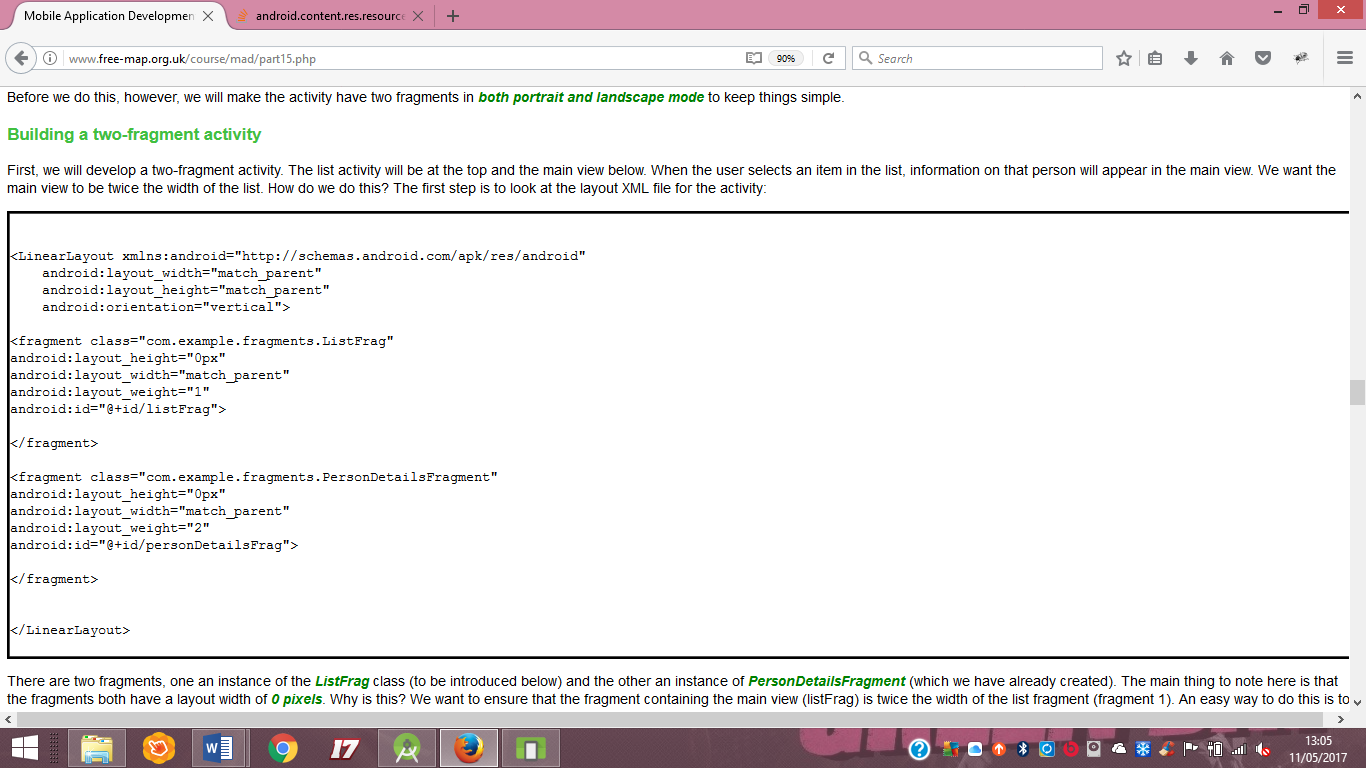
Whilst developing the application I encountered a problem where the web service was not accepting types with a space in the string from my spinner. So, for example ‘Park’ was accepted but ‘Amusement Park’ was not. I was unable to tell if this was a problem with the web service or my implementation, therefore I simply changed my apps spinner to contain simpler, one worded terms.

|  |
| --- |
| **Requirement 8**  Allow the user to see a list of all points of interest in memory so far. When the user selects an entry in this list, the map should be centred on that point of interest. This should be done using fragments. |

As this needs to be done using fragments and I need a list I will need to use a ListFragment and a fragment containing the map. In the ListFragment class I will need to create an array of place names and another of the place types. This will be done by adding to an overall array list of places currently loaded on the application which will be added to when a file is downloaded or loaded from storage. To make them clickable they will need to have an onListItemClick method which will change the view of the map in the other fragment. The ListFragment class will look like the extract below.



To make both the ListFragment and the MapFragment visible when landscape I will need to make a separate xml layout file to be used when the orientation is landscape. The xml file will contain both the map fragment and the list fragment and will look similar to this extract shown below.



**Problems Encountered:**

There was a problem where my static places class was not being added to as expected however I could not solve this problem in time for the deadline and therefore the app was submitted not working fully. This problem caused my list to not be populated with any options.