Group Report: Coursework 4: London Property Marketplace

Shakeeb Jumaan K21087021

Israfeel Ashraf K21008936

Michael Seiranian K20127931

William Atta K21097986

- **A description of your GUI, including the functionalities provided by the GUI.**

Our GUI allows users to pick a price range of the properties they would like to see. Once they have done this, they are then able to move on to the second panel using the buttons at the bottom. These buttons will allow the user to navigate through the four different panels in our GUI. The second panel consists of a range of buttons which represent the different boroughs in London. These buttons will be colour coded with different shades of purple which represents the number of available properties in that borough. Once the user has chosen the borough they would like to view, a new window will open with a table view of all the different properties in the borough of the user's choice. The table view has different headers such as: host name; price; minimum number of nights and number of reviews. In addition to that, the table view can be sorted by number of reviews, price, or alphabetically by host name. When the user has picked the property they desire, a description of that property will be displayed. The third panel shows a range of statistics for the properties in the dataset such as: average number of reviews; total number of properties; number of houses and apartments only; most expensive borough and more. Finally, the last panel has a map of London with markers placed on each borough. When the marker is clicked, it shows the underground stations in that borough as we felt it is important for our users to know the availability of underground transport in the borough they would like to live in.

- **A description of your additional statistics**

1) The first additional statistic shows the borough with the most reviews per listings in the borough. By looping through the AirBnb listings, it counts the total number of reviews of each borough, and stores the borough name-reviews pair in a HashMap. The number of reviews of each borough is divided by the number of listings in each borough and the borough corresponding to the highest result is chosen to be displayed as the borough with the most reviews.
2) The second additional statistic counts the number of listings that have a garden. It does this by going through the description of every listing and looking if the description contains the word "garden" and incrementing the count upon each find.

3) The third additional statistic finds the most luxurious borough. The program loops through every listing and looks for the words “luxurious” or “luxury” in the descriptions. The number of luxurious listings found in each borough is kept record of in a HashMap, again with the borough name-number pair arrangement. The number of luxurious listings in each borough is divided by the number of total listings in the borough, which is pre-calculated and stored in the “numListingsPerBorough” HashMap upon creating the third panel. The results for each borough are then stored in another HashMap in the borough-value arrangement and the borough with the maximum result is set to be the most luxurious borough.
4) The last additional statistic makes use of the underground-stations.csv dataset, which was adapted from the Wikipedia “List of London Underground stations”[1]. The statistic shows the borough with the most underground stations. It does this by counting the number of stations in each borough and storing this information in a HashMap, in a borough name-count arrangement. The borough with the highest count is the borough with the most underground stations.

· A description of the functionality provided by your fourth panel.

The fourth panel shows a map of London with clickable pins on each borough that open a table showing all of the underground stations and data about the stations in the borough. The data on the underground stations is loaded up from the underground-stations.csv file, and organised into an ArrayList with the help of the UndergroundDataLoader class. The GUI for the fourth panel was mainly designed in SceneBuilder, the FXML file of which was then loaded in the UndergroundPanel class. A map of London was placed on an Anchor Pane and pins were placed above each borough. The pins change colour upon the mouse hovering over them, and the mouse icon turns to an open hand, to inform the user that the pins can be clicked. This effect was achieved through linking the “On mouse entered” and “On mouse exited” events in SceneBuilder to methods in the code, which increase or decrease the brightness of the pins with the help of the JavaFX ColorAdjust package. The “On mouse clicked” event of each pin in SceneBuilder was linked to a method in the code which opens up a table containing data about each underground station in the borough. The program identifies which pin was clicked, and the ID of the pin is then compared to the borough of each underground station, allowing the creation of the table. A “Back to map” button also appears on the top-right hand corner, allowing the user to go back to the map and select a different borough if they wish.

· A description of your unit tests.

The unit tests undertaken in this project are for the Colours class. This class deals with the second panel and assigning colours to each borough based on how many properties are in that borough. The colours class takes the minimum number of properties and maximum number of properties, takes the difference between them and divides by 10 to create 10 categories, each with a corresponding colour. The result of this is a gradient effect on the map, showing boroughs with a smaller number of properties according to the user’s price range as lighter in colour, and boroughs with a darker colour as boroughs with more properties in.

This class therefore provides core functionality to this application and has been chosen to be tested, as there is a lot of room for error when deciding the colours, arising from boundary conditions or problems within the dataset.

The unit tests can be split into two parts:

1 - Setting the maximum and minimum number of properties within the dataset.

2 - Check the boundary conditions of each condition.

An example of this would be if the minimum number of properties is 0, and the maximum is 100. The increments per category is 10.

The unit tests then test boundary conditions such as if a borough has 80 properties, or if a borough doesn't have any properties in the user's selected price range.

It must be noted that some methods within the Colours class could not be used since they required JavaFX buttons to be passed in as parameters. The test class was unable to initialise the buttons, so a slightly tweaked version of the method was placed in the test class, in order to test what colour should be retrieved from the colours array list.

- **Referencing**

https://en.wikipedia.org/wiki/List_of_London_Underground_stations[1]