PPA CW 4 Group Report

Participants
Mateusz Adamski – 20063061
Alexandru Matei - 20054925
Ejaz Karim - 20059213
Godfrey Gomes - 20082149

Table of Contents

Group Organisation and Workflow:

During our first meeting, all members of the team decided on what part of the project they primarily wanted to work on. Mateusz Adamski worked on the Map panel and Unit testing. Alexandru Matei worked on the Statistics panel. Ejaz Karim worked on the GUI Introduction and Welcome panel with Mateusz Adamski. Alexandru Matei participated in the merging of the different components, and implemented the Favorite functionality.
The group met shortly after the group nomination that is 12 march. We created the new Git project and we've done the Git part from the task sheet. On the next meeting on 17th we split the work with general, common agreement. We were working separately until 30th of march giving support and feedback to one another via whatsapp and discord. On 30th we made a first meeting to check the progress together then we had meetings like this on 1st, 2nd, 4th, 5th, 6th and 7th April.

To start the application, invoke the main method in AirbnbdataLoader class.

GUI Introduction and Welcome Panel:

The GUI class is called “AirbnbFX”, which also holds the welcome panel. Scene builder was initially used to position the “next” and “previous” buttons and the spinners at the top in the welcome panel. Later on, we ended up writing the rest of the welcome panel using java code instead of Scene builder. Spinners were used instead of Comboboxes because we agreed that spinners in this use case were better as the user is able to enter numbers specifically and scroll through numbers. You are given a limited selection of numbers using comboboxes, so it is more flexible for a user to use spinners. Originally, I wanted to use pagination for the buttons at the bottom. However, I could not find any way to make the pagination loop. So, we decided to just use regular buttons instead. An arraylist called “panels” is used and all panels are added to it, this is so it is possible to iterate through panels. The methods “nextButtonClicked” and “previousButtonClicked” are used to go to the next panel in the arraylist of panels. Then an arraylist of the .csv was used, also a temporary arraylist was needed in order to filter out the original arraylist. The temporary arraylist is first copied from the original arraylist. The method “filterArray” uses lambdas to filter out the temporary arraylist based on what is currently in the spinner’s ‘from’ and ‘to’ values, and will disable the buttons if the temporary arraylist is empty. The buttons are disabled here to prevent the user from going to the next panels, as other panels require information from the temporary arraylist, and the other panels won’t function properly if the arraylist is empty. The method “checkPriceRange” checks the current values in the spinners and if the ‘toValue’ > ‘fromValue’ then another method “incorrectPriceRangeAlert” is invoked and a popup will display warning the user that the price range is invalid, and then the buttons are disabled. The buttons are disabled in this case because the invalid price range will mess up the filtering of the arraylist, and therefore give wrong information in the rest of the panels. Otherwise, the “filterArray” method is invoked, using the values from the ‘from’ and ‘to’ spinners. The welcome panel itself contains an image, title, and instructions for the user to view. This is done by using VBoxes and HBoxes to position the texts and image.

Map Panel:

MapPanel is the main class of the Map GUI. It initialises this part of the application. Map panel is responsible for loading the background map and then drawing hexagons in appropriate colours on it. The necessary information like what is each borough row and column is stored in BoroughInformation class. The information about boroughs is computed in PropertiesAvailibilityIndicator class. There all the boroughs are assigned with the correct row and column, the colour is determined and all information is being added for each borough to its own instance of BoroughInformation class. The colour is determined in such a way that boroughs with the greatest amount of properties are the greenest and with the least are the reddest. If there are no properties then the hexagon is painted black with opacity 100%. BoroughInformation class is using Hexagon class that is responsible for mathematical implementation of hexagons. If the user will click any of the hexagons, a new window will pop up with a list of all properties available in this borough in the chosen price range.

Unit Testing:

The class being tested is the PropertiesAvailibilityIndicator it provides crucial functionality for the whole map part of the application. The test focuses on correct interactions with the update call from the main class of map implementation (MapPanel class, update method). And the BoroughInformation class by checking that every property is allocated to the corrected borough. The testing considers 5 cases of the method calls to check if every case is handled appropriately.

Errors and Bugs in Map part:

I tried to make the image with coloured hexagons resizable but the problem seemed to be too elaborate for this project for me. The MapPanel can be sized down so that some boroughs are not visible.

Statistic Panel:

All four of the additional statistics are based on the user's favorite properties, to help the user narrow down their choice. The first additional statistic shows which is the most well-connected property out of the user's favorites. To this end, we use a dataset provided by Transport for London which provides connectivity ratings for different locations throughout the whole of London, at a reasonable resolution. We go through the user's favorites, find the nearest rated location and take note of its grade, and compare it with the rest of the user's favorite properties, in order to get the property with the highest connectivity rating.
The second statistic is useful for older users or those with health problems, because it picks out the least polluted of their favorite properties. In a similar way to the connectivity statistic, a secondary dataset is used (extracted from the London Datastore). Unfortunately, calculating a good air quality index is difficult and requires data we have no access to, so we had to do our best to approximate a means of comparing properties in relation to how polluted they are, thus, for each coordinate pair offered by the dataset, we create pollutant quantity sums. Then, we calculate the geometric mean of these sums (the geometric mean is preferred in order to account for the naturally big differences in the quantities between the different pollutant species), and this mean is what we use to compare the properties. The property with the smallest geometric mean, so the one which is in the area with the overall smallest quantity of pollutants, is what we return as the result.
The third additional statistic is useful for helping the user narrow down their search to a more specific area. It indicates which borough the user has favorited the most listings in. The user can use this statistic to conclude that they might prefer, in general, the establishments in a borough, and look there more closely.
The fourth additional statistic is useful for people on a really tight budget, it simply lists the cheapest listing out of their favourites.

4th Panel:

For the 4th panel Godfrey was responsible but unfortunately he didn't give his contribution to us. We have never seen his part.