Assignment 4

Group Report

Title: Assignment 4: Group Report

Deadline: 30/03/2022, 23:59

Authors: Adam Murray (K21003575)
Augusto Favero (K21059800)
Mathew Tran (K21074020)
Tony Smith (K21064940)

GUI

Application window

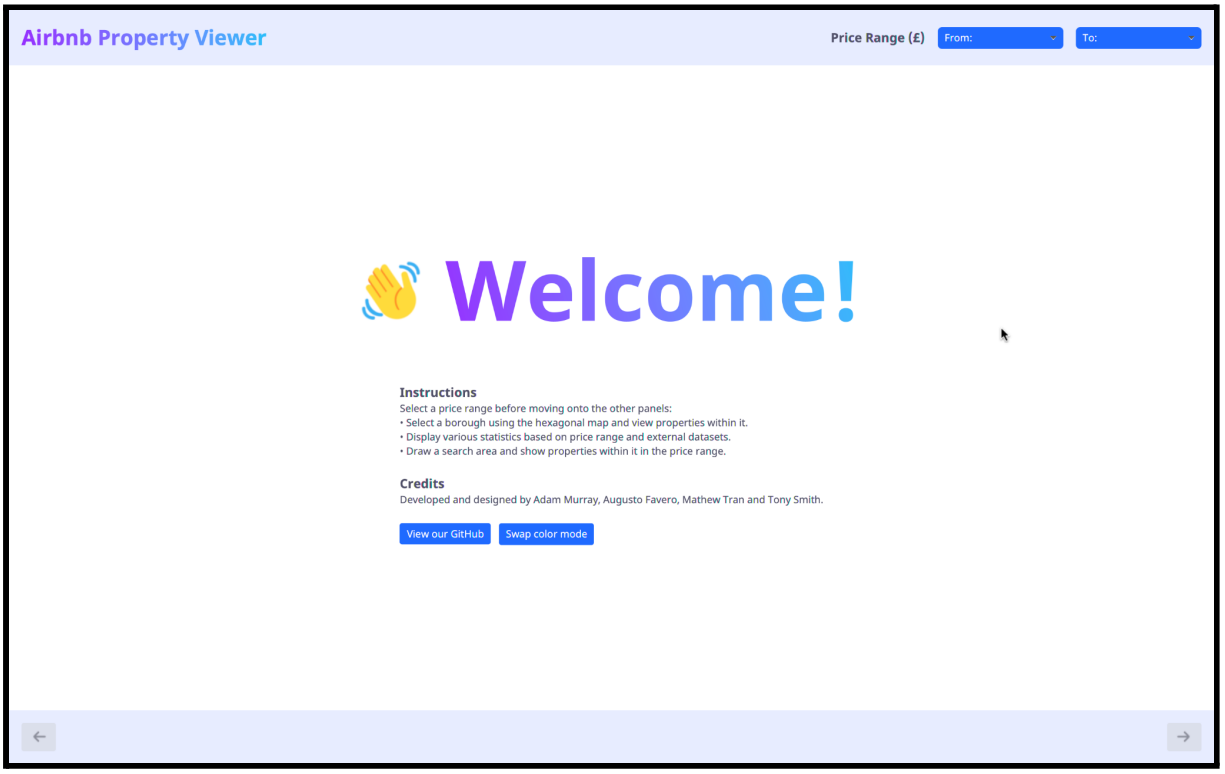
Our application window is the primary window. Technically, it is a `BorderPane` with the top, centre and bottom filled. The centre is the currently displayed panel.

The top is a bar with the title “Airbnb Property Viewer” on the left side and a pair of combo boxes for the price range on the right side. The combo boxes start with either “No min” or “No max” as we felt these were more human readable and nice for the user. The options then go up in 10’s, 100’s, and 1000’s so that the user has greater control when it matters and the combo box options are not unnecessarily long:

A demonstration of how the combo box options start with “No min” (in this case) or “No max” and then go up in 10’s, 100’s and then 1000’s.

The bottom bar contains two navigation arrow buttons. These buttons call upon a circular list, implemented as a linked circular list, in order to get the next and previous panes. I used a circular list to easily allow the panes to loop.

Welcome page



A preview of what our welcome page looks like.

On the welcome page, we thought that it would be good to include instructions for the user so that they know how to use the application. We added two buttons: one links to the GitHub repository that stores the project and the other changes the colour mode. This colour mode button simply alternates between the two available colour themes. [WAFFLE ABOUT HOW IT CHANGES OUT CSS STYLESHEETS].

OpenLayers 3 map

We have a class called `OpenLayersMap` that extends `AnchorPane` that is responsible for creating a geographical map in a `WebView`. This `WebView` refers to a local website stored at `resources/open-layers-map/map.html` which uses the OpenLayers 3 library to display an interactive geographical map. The core behaviour of the map (creating the map itself) is implemented in `resources/open-layers-map/core-behaviour.js`.

`OpenLayersMap` allows the caller to attach new behaviours such as displaying interactive property markers, the ability to draw on the map, and boundaries around the boroughs. These behaviours are just additional JS scripts attached to the core one.

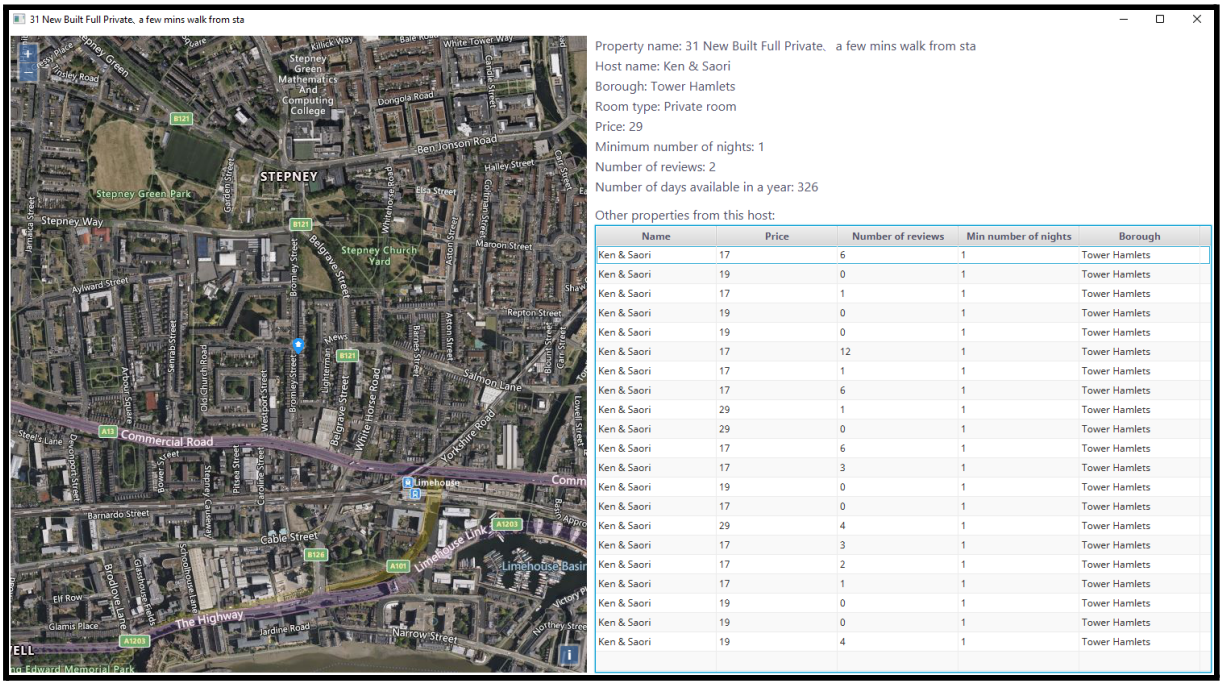
Property details window

The property window is a window displayed to the user that provides details on a given property. It shows its location on a map using the OpenLayers 3 map, more details provided

above. It shows the location using a marker and then centres the map around the property. This window also shows a list of the following property details:

- Property name
- Host name
- Borough
- Room type
- Price
- Minimum number of nights
- Number of reviews
- Number of days available in a year

Finally this window shows a table populated with all the other listings by the same host.

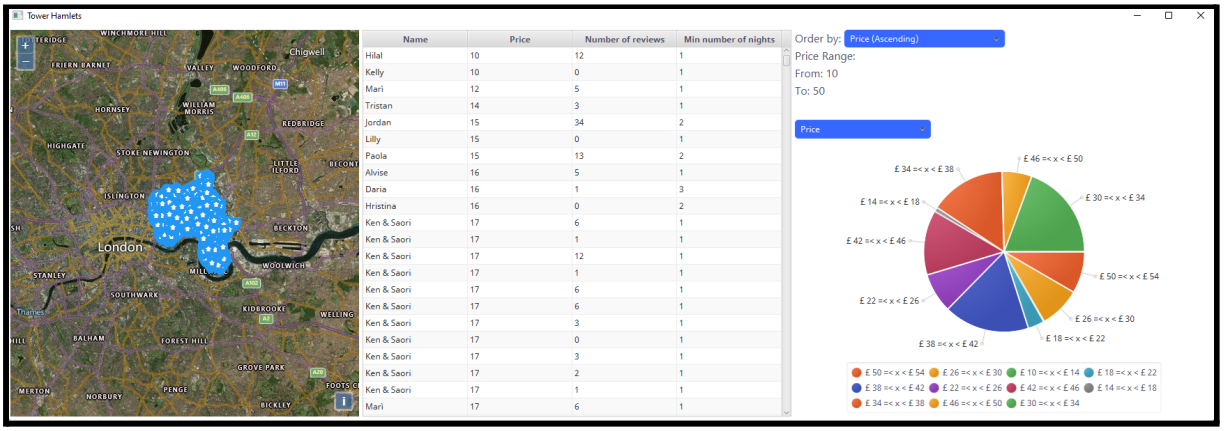


Borough details window

The borough window displays the properties contained in the borough in the price range, and displays statistics on those properties using a pie chart.
On the left of the screen is an OpenLayers 3 map, more details provided above, which displays a marker at the location of each property in the borough in the price range. These markers are also clickable and will open the property window for its property when clicked. The map is initially centred at the average position of the markers.
In the middle of the screen is a table that provides 4 details about the properties:

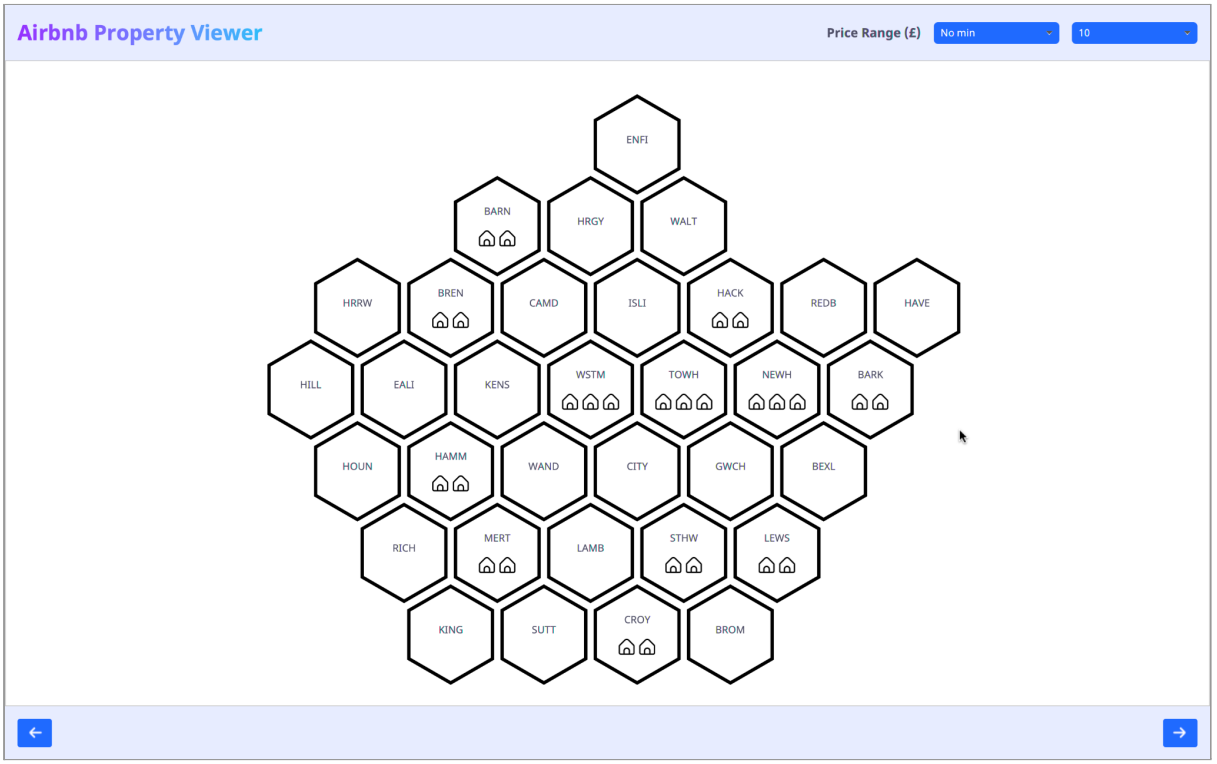
- The Host's name
- The price per night
- The number of reviews
- The min nights you can stay there

You can then click on a row in this table to open a property window for the property in the row clicked. This table can then be sorted using the combo box in the top right of the screen. The table can be sorted by:
Host name (ascending and descending)
Price (ascending and descending)
Number of reviews (ascending and descending)
The right side of the screen is then the order combo box, price range for the window and the pie chart as well as the combo box to select the drop down.



Hexagonal borough map

We recreated the hexagonal borough map in the task sheet:



Clicking on a hexagon button opens the borough details window for that borough. We created a `QuantityVisualiser` that uses house icons to represent the number of

properties in a borough on the hexagon buttons. No icons means there are no properties, and from there on the number of icons indicates relatively more properties compared to other boroughs.

Statistics

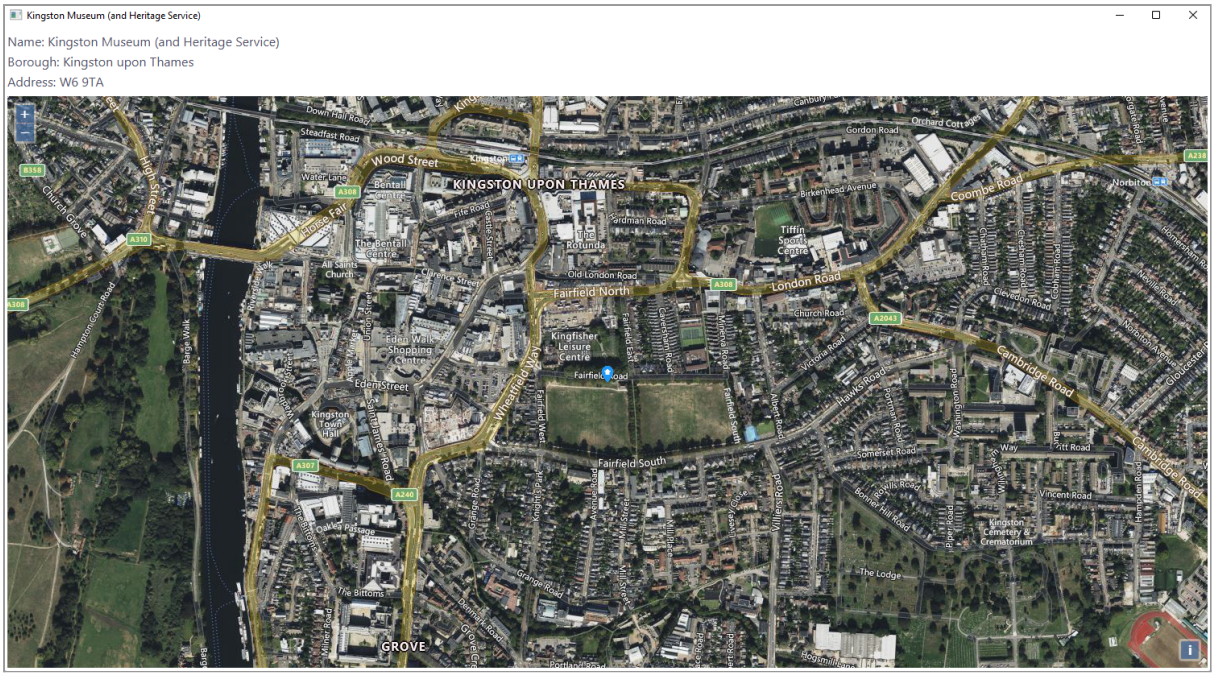
The statistics panel shows a series of statistics derived from the dataset given. When the user has selected a valid price range, they are then able to navigate to the statistics panel. Initially, the four base statistics will be shown, with each statistic separated into its own section. Each section contains the title that represents the statistic, the value calculated by performing a series of operations, and two buttons that allow the user to navigate to the other statistics as shown below:

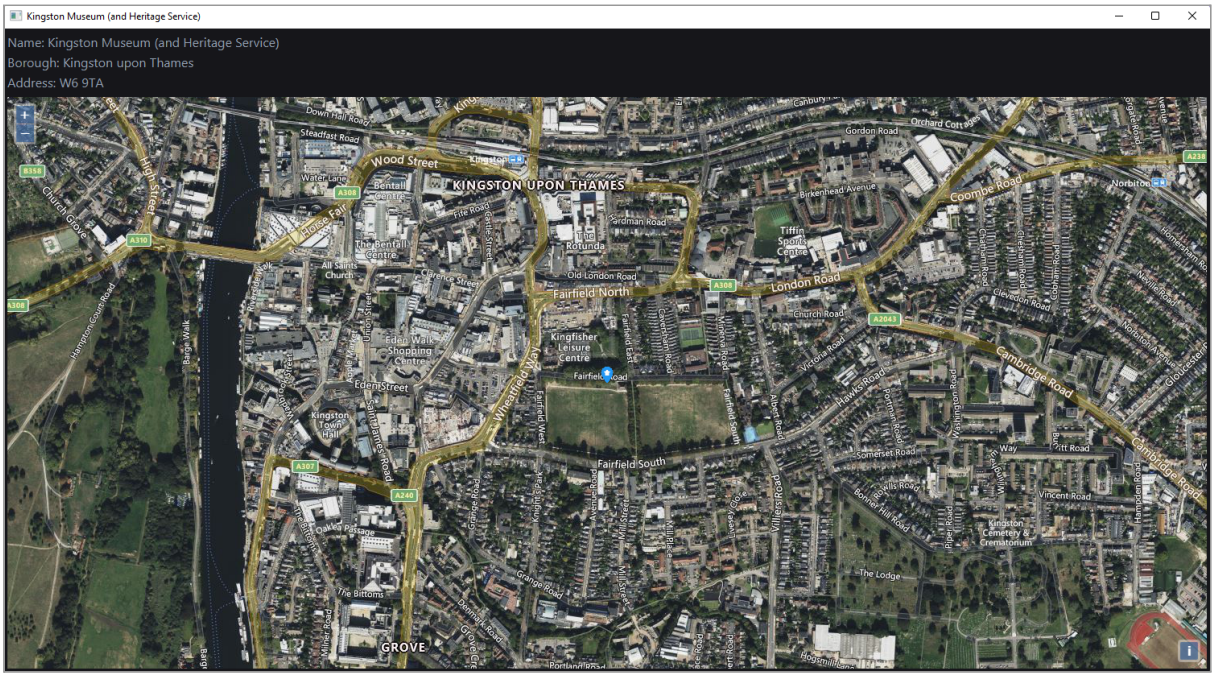
To ensure that under no circumstances should the same statistic be shown twice, as well as the ordering of the statistics be uniform, a double ended queue is implemented. The statistics that are currently displayed within the panel are not in the queue and when the user clicks one of the buttons, the item at one end of the queue is taken out and displayed, whereas the statistic that is being replaced is put into the other end of the queue. The side of the queue that is being added to and the side that is being deleted from is dependent on the button pressed, in order to reflect the sequence that they should appear in. For example, if the user clicks the right-sided button and then the left-sided button, the original statistic should be displayed.

Destination details window

The destination window displays a destination, aka a pub or an attraction. This window displays a map with a marker on the location of the destination. The map is then centred on the marker. Above the map are the following details:

- Name
- Borough
- Address





Additional statistics

The additional statistics are as follows:

- Boroughs with highest social score
- Boroughs with lowest crime
- Closest five pubs to a property
- Closest five tourist attractions to a property

These statistics are separated into separate categories as they use different csv files taken from various sources to present their data.

The first two statistics listed use data taken from the government website, where an extensive study was taken across all London boroughs, which was updated until 2018. Part of the data taken included scores on the social atmosphere within each borough. These included:

- Average Public Transport Accessibility Score
- Life Satisfaction Score
- Worthwhileness Score
- Happiness Score
- Anxiety Score

These five statistics make up a total social score and the three boroughs with the highest score are presented to the user as long as there is a property within that borough in the price range selected.

The second statistic is also derived from this dataset, where each borough has a number of crimes committed per thousand people. The three boroughs with the lowest crime rate are then displayed to the user as long as there exists a property in that borough within the price range.

The last two statistics (“Closest five pubs to a property” and “Closest five tourist attractions to a property”) utilise data from two csv files taken from the government website. The csv files have very similar structure therefore they are categorised into a single listing object (DestinationListing) which are differentiated by the use of enumerated types. Each destination has a **name**, an **address**, a **longitude** and **latitude**, the **borough** within which they reside and a **price**. The statistics have been made interactive so that the user can select from a set of three combo boxes a borough name and a property that exists within that borough (filtered by the price range selected from the main pane combo boxes) and finally a price (for pubs a metric for affordability **“£”**, **“££”, “£££”** and for tourist attractions ticket prices are used **“free”**, **“£2.50 - £5.00”**).

Once these options have been selected then the five closest, either pubs or attractions are shown to the user relative to the selected property with the destination name, address and distance in kilometres displayed.

When no destinations are found given the set of inputs nothing is shown, Given that there are 4099 rows where each row represents a single pub entity in the pubs csv file, in comparison to the 164 rows where each row represents a famous tourist attraction entity, the likelihood of finding pub destinations is higher.

Fourth panel (drawable search area map)

The fourth panel is a drawable search area map based on Rightmove's "search by area" feature. It uses the OpenLayers 3 map class we created to display an interactive map.

You can click "Toggle borough boundaries" to toggle whether or not the borough boundaries are shown in blue.

There are two modes for interacting with the map: drawing mode (the default) and marker mode. Two buttons have been provided for switching between these two.

The drawing mode allows a user to draw a green polygon by placing down points on the map. Double clicking or linking to the beginning point will close and complete this polygon. Once the polygon is completed, drawing a new one will remove the old one so that there is always only one drawn area on the map.

The marker mode allows the user to then see the properties within their search area and in the price area, represented as blue markers. These markers are clickable and bring up the property detail window for that property. If the price range changes when still in marker mode, the user must click the Refresh button to update the markers.

Unit tests

We chose the `ListingProcessor` class to test.

```
List<AirbnbListing> filterByBorough
(List<AirbnbListing> listings, String boroughName)
```

- Check that valid arguments are accepted with no issue and return the expected result (the given listings filtered by borough name).
- Check `IllegalArgumentException` is thrown with the message “The provided listings argument is invalid.” if the listings argument is null.
- Check `IllegalArgumentException` is thrown with the message “The provided listings argument is invalid.” if the listings argument is empty.
- Check `IllegalArgumentException` is thrown with the message “The provided borough name argument is invalid.” if the borough name argument is null.
- Check `IllegalArgumentException` is thrown with the message “The provided borough name argument is invalid.” if the borough name argument holds an invalid borough name not in the list of actual boroughs.

```
List<AirbnbListing> filterByPriceRange
(List<AirbnbListing> listings, int fromPrice, int toPrice)
```

- Check that valid arguments are accepted with no issue and return the expected result (the given listings filtered by price range).
- Check `IllegalArgumentException` is thrown with the message “The provided listings argument is invalid.” if the listings argument is null.
- Check `IllegalArgumentException` is thrown with the message “The provided listings argument is invalid.” if the listings argument is empty.
- Check `IllegalArgumentException` is thrown with the message “The from-price argument cannot be greater than the to-price argument.” if the price range arguments are invalid (the from-price is greater than the to-price).

```
int getNoOfPropertiesInBoroughWithMost
(List<AirbnbListing> listings)
```

- Check that valid arguments are accepted with no issue and return the expected result (the number of properties in the borough with the most properties).
- Check `IllegalArgumentException` is thrown with the message “The provided listings argument is invalid.” if the listings argument is null.
- Check `IllegalArgumentException` is thrown with the message “The provided listings argument is invalid.” if the listings argument is empty.

```
int getMaxPropertyPrice
(List<AirbnbListing> listings)
```

- Check that valid arguments are accepted with no issue and return the expected result (the highest property price out of all given property listings).
- Check `IllegalArgumentException` is thrown with the message “The provided listings argument is invalid.” if the listings argument is null.
- Check `IllegalArgumentException` is thrown with the message “The provided listings argument is invalid.” if the listings argument is empty.

```
AirbnbListing getListingWithId
(List<AirbnbListing> listings, String id)
```

- Check that valid arguments are accepted with no issue and return the expected result (the highest property price out of all given property listings).
- Check `IllegalArgumentException` is thrown with the message “The provided listings argument is invalid.” if the listings argument is null.
- Check `IllegalArgumentException` is thrown with the message “The provided listings argument is invalid.” if the listings argument is empty.
- Check `IllegalArgumentException` is thrown with the message “The id argument cannot be null.” if the id argument is null.

```
List<AirbnbListing> getOtherListingsWithHostId
(AirbnbListing hostListing)
```

- Tests that when a valid input is given that expects a list with a value in it, a list with a value in it is returned.
- Tests that when a valid input is given that expects an empty list, an empty list is returned.
- Tests that when a null value is given the method throws an Illegal argument exception.
- Tests that when a listing with a null id is given it throws an Illegal argument exception.
- Tests that when a listing with a null host id is given it returns an empty list.

```
List<String> getBoroughs
(List<AirbnbListing> listing)
```

- There is a test that checks if a given custom list of AirbnbListing are provided the correct boroughs name are returned as a list of strings
- There is a test that checks if an null list is passed to the method then the returned list is of size 0 (empty)
- There is a test that checks if an AirbnbListing object in the list has a null borough name it is filtered out
- There is a test that checks if an AirbnbListing object in the list passed has an invalid borough name it is filtered out

```
List<String> getPropertiesNameInBorough
(List<AirbnbListing> listings, String boroughName)
```

- There is test that checks if a valid list of AirbnbListing and borough name are entered then all the property names are correctly returned in a list of strings
- There is test that checks if an invalid borough name is passed then an empty list is returned since no properties exist in that invalid borough name
- There is a test that checks that if null is entered as a borough name then the returned list is empty

```
List<DestinationListing> filterDestinations
(List<DestinationListing> destinations, String borough, String
price)
```

- There is a test that checks that a correct list of values are returned given that valid inputs are given
- There is are two tests that check against the price format for the pub destinations and attractions destinations, if an invalid price format is entered then an `IllegalArgumentException` is thrown
- There is a test against entering a borough name or price that are non existent in the list of destinations passed as a parameter

```
AirbnbListing getPropertyListingByNames
(List<AirbnbListing> listings, String listingName, String
boroughName)
```

- Test that checks that if valid data is passed then the correct AirbnbListing object is returned
- Test that checks if an a null is passed for either the listing name or the borough name then `IllegalArgumentException` is thrown

- Test that checks if a non existent borough name or list name is passed that is not present in the listings list

```
int[] getListingPrices
(List<AirbnbListing> listings)
```

- Test that the list of AirbnbListing's is mapped to an array of int containing the price of each AirbnbListing when a valid list is input.
- Test that when the list of AirbnbListing's is empty, an empty array is returned.
- Tests that when a null list is input the method throws an IllegalArgumentException.

```
int[] getListingReviews
(List<AirbnbListing> listings)
```

- Test that the list of AirbnbListing's is mapped to an array of int containing the number of reviews for each AirbnbListing when a valid list is input.
- Test that when the list of AirbnbListing's is empty, an empty array is returned.
- Tests that when a null list is input the method throws an IllegalArgumentException.

```
int[] getListingMinNights
(List<AirbnbListing> listings)
```

- Test that the list of AirbnbListing's is mapped to an array of int containing the min number of nights for each AirbnbListing when a valid list is input.
- Test that when the list of AirbnbListing's is empty, an empty array is returned.
- Tests that when a null list is input the method throws an IllegalArgumentException.

```
int getMin
(int[] values)
```

- Test with valid array of numbers that returns the correct minimum value
- Test with an empty array that throws an `IllegalArgumentException`

```
int getMax
(int[] values)
```

- Test with valid array of numbers that returns the correct maximum value
- Test with an empty array that throws an `IllegalArgumentException`

```
long retrieveSpecifiedAmount
(int[] values , int from, int to)
```

- Test with a valid list of values and from and to integers, returning an expected long which represents how many integers in the array are between the from and to value
- Test with an empty array throws an `IllegalArgumentException`

```
Position getAveragePosition
(List<AirbnbListing> listings)
```

- Test that a Position containing the average longitude and latitude is returned from the valid list of AirbnbListing's given.
- Tests that when a null list is input the method throws an IllegalArgumentException.
- Tests that when an empty list is input, the method throws an IllegalArgumentException.

```
int getNumberOfReviews
(List<AirbnbListing> listings, int fromPrice, int toPrice)
```

- Test that checks if a valid list and price range is passed then the correct value is returned
- Tests that when a null list if input the method throws an IllegalArgumentException
- Tests that when an empty list is input, the method throws an IllegalArgumentException.

```
long getNumberOfListings
(List<AirbnbListing> listings, int fromPrice, int toPrice)
```

- Test that checks if a valid list and price range is passed then the correct value is returned
- Tests that when a null list if input the method throws an IllegalArgumentException
- Tests that when an empty list is input, the method throws an IllegalArgumentException.

```
long getTotalAvailableProperties
(List<AirbnbListing> listings, int fromPrice, int toPrice)
```

- Test that checks if a valid list and price range is passed then the correct value is returned
- Tests that when a null list if input the method throws an IllegalArgumentException
- Tests that when an empty list is input, the method throws an IllegalArgumentException.

```
long getNonPrivate
(List<AirbnbListing> listings, String roomNeeded, int fromPrice,
int toPrice)
```

- Test that checks if a valid list and price range is passed then the correct value is returned
- Tests that when a null list if input the method throws an IllegalArgumentException
- Tests that when an empty list is input, the method throws an IllegalArgumentException.
- Tests that when an invalid string is input, the method throws an IllegalArgument

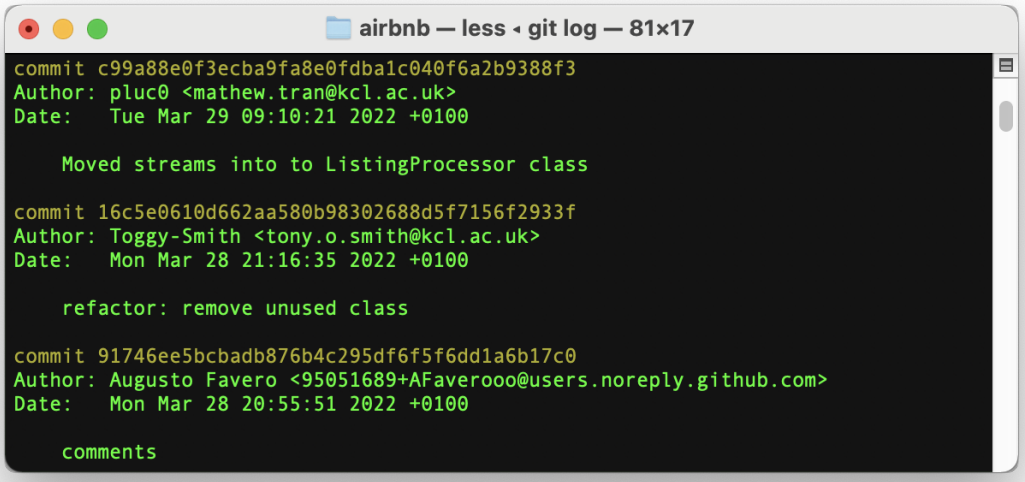
References

- Our drawable search area map is based on Rightmove's "draw a search" feature (https://www.rightmove.co.uk/student-accommodation/map.html).
- Our hexagon drawing code is based on this Stackoverflow answer (https://stackoverflow.com/a/54173321).
- We used icons from this free icon pack online (https://www.flaticon.com/uicons).
- The hand-waving emoji used in the welcome page is a Twitter emoji (https://twemoji.twitter.com/).
- The csv file used as part of our additional statistics taken from the government website: (https://data.gov.uk/dataset/248f5f04-23cf-4470-9216-0d0be9b877a8/london-borough-profiles-and-atlas).
- We used this algorithm to calculate the distance between two geographical points for our statistics panel. (https://stackoverflow.com/questions/3694380/calculating-distance-between-two-points-using-latitude-longitude).

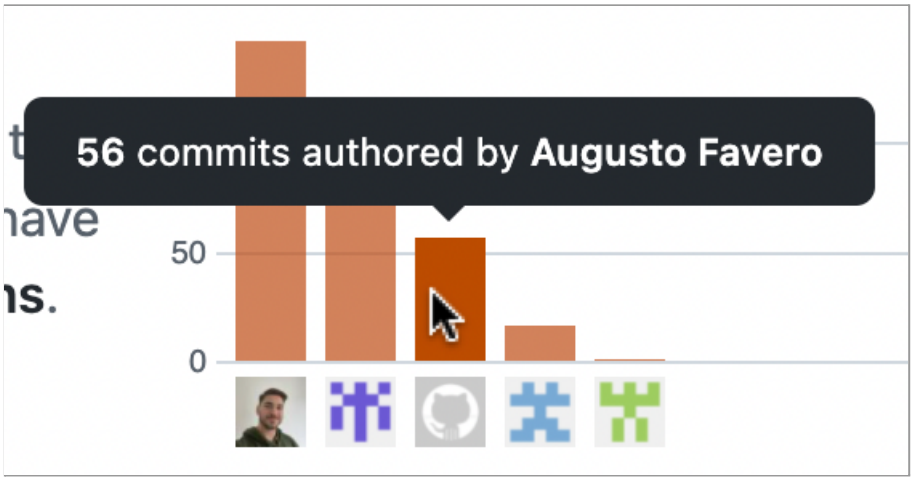
Issues

Incorrect commit email for Augusto Favero

Because Augusto Favero (K21059800) had the incorrect email address associated with his commits, those commits show under the author **Augusto Favero <95051689+AFaverooo@users.noreply.github.com>** with the default GitHub email while our assignment partners have their King's email addresses:

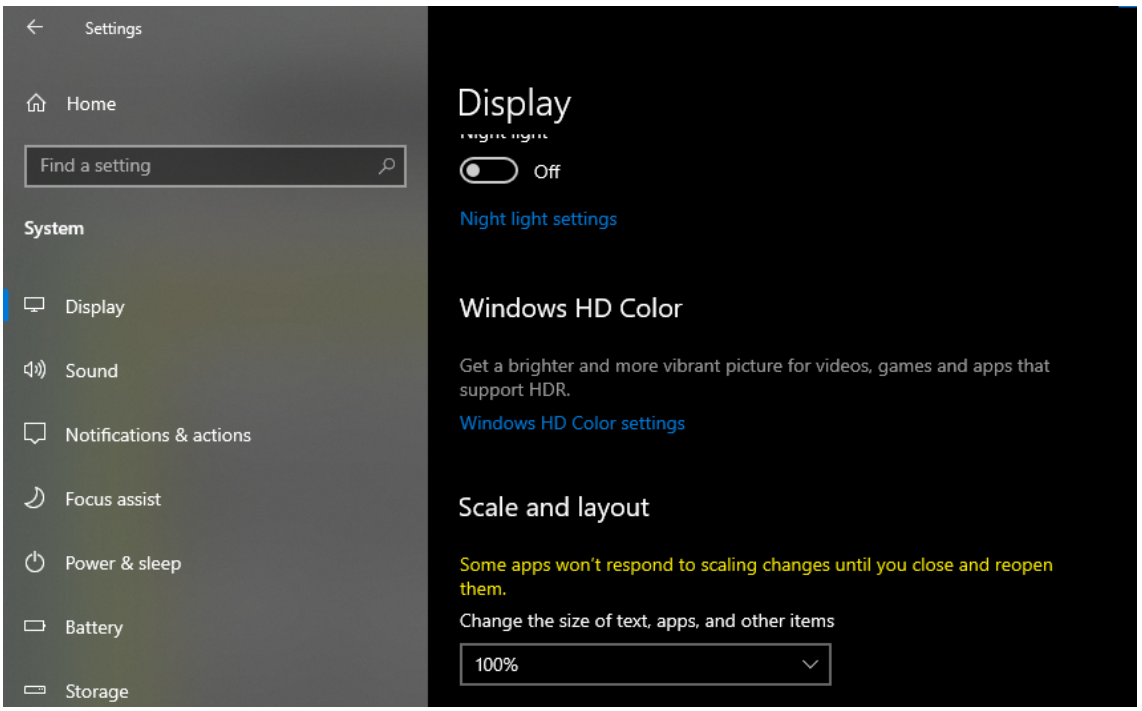


These are his commits:



Map scaling bug

We encountered a bug with our OpenLayers 3 map where the map would not fill all of the available space. This issue seemed to be exclusive to Windows but only some devices. We discovered that it can be resolved by changing the Windows DPI setting to 100%:



Changing this setting to 100% should fix DPI scaling issues.

We found that on the devices that suffered from the bug, this setting was at a non-100% value such as 125% in our case. Upon further research, we discovered that this is a known JavaFX bug (https://bugs.java.com/bugdatabase/view_bug.do?bug_id=JDK-8248126) where JavaFX informs Windows it adjusts for the given DPI setting but in fact does not.