Covid-19 London Statistics Project

Mohammed Ahmed (K22026228)

Shahriar Miah (K22023070)

Christopher Herre (K22001776)

Talal Alohali (K21130307)

GUI:

Welcome Panel (Panel 1):

This panel is the welcome panel for our project. We have used this panel to instruct the users on how to navigate and use each panel as well as inform them of any potential issues they may face. The instructions were implanted through a series of labels. This panel was also used for setting the range of dates that the user would like to display the statistics for. This date range was then used in all other panels to control how many statistics were shown form the records. We have implemented it through a calendar system. This is where the user can click two buttons at the top which reveals a set of dates in the form of a calendar. They can then choose a from date and a to date using the dates that pop up.

We have two buttons at the bottom left and bottom right of the windows which allows you to change between panels. These buttons are made disabled from the beginning of the program until the user has placed in a valid input for the dates. The invalid input for dates includes placing dates that are outside of the date range of the data. This was between 2020/02/15 and 2022/10/15 and if any dates were outside this range, then an error was given that stated the user enter a date from the date range. The other case in which an error is thrown is if the from date is after the to date in which case the same error message is provided to the user. The error label is displayed using the methods returnFromDate() and returnToDate(). These methods check whether the date doesn't have value which occurs when an invalid input is provided and changes the error label. The getValidity() is used to check the valid and invalid inputs. This is done by changing the dates to integers and then checking how large each is and comparing them. If the number for one date is larger than the other, it means the date is later. In both cases, the next button is kept disabled so the user cannot navigate between the panels until they've set a valid date. Once the user has placed a valid date, the buttons are then enabled, and they can then start to navigate between panels. The method used to disable and enable the buttons is processDates(). The previous button in this panel will remain disabled throughout the program as there is nowhere for that button to navigate through.

Boroughs Panel (Panel 2):

In the implementation of this panel, we have used a series of circle objects to replicate the map of all the boroughs. This attempts to show the general position of each borough relative to where they are on the map. We have used labels within each shape to display to the user what borough each circle references. The buttons used to switch between panels is controlled by the methods switchToWelcome() and switchToStatistics() which change the root of the FXML file to a different panel.

Clicking on each circle will produce a pop-up window which contains a table filled with all the data for that specific borough between the date period that the user has selected. The pop-up window is created using the TableView function. We have created the table ourselves through the class TableLoaderLayout in which all the values of the table are all instantiated in the class. In order to input the data, we have created instants of every column within the TableLoaderLayout class. We have then created an observable list in which every row of records is then placed into the list. We then filter out the data into whichever borough the user has clicked in order to only display that specific data in the table. This list is then set into the TableView to then display all the data. You can then click on each column to sort the order of the data by that specific column in ascending or descending order. This is done automatically through the TableView itself.

We have then implemented a colour scheme for the different boroughs dependent upon the number of deaths within each borough. This has been created using a hash map which maps a string containing the name of the borough to the number of deaths within that borough. This is completed through the getTotalDeathsHashMap() method in which we loop through all of the data and find the final date for each borough and record the final deaths. These values are then placed into the hash map. We alter the colour of the shapes using the method chanegCircleColour() in which we go through the hash map we have and check the number of deaths associated with the borough. You then call the method colourGrade() in which we have used a range of 0 to 60000, 60000 to 100000 and anything above 100000 to determine the colour of the borough we have set. We then fill the shape with that colour.

For the final part of this panel, we have a button in the top left to let the user show the colours. They can then click the button which will change all the colours accordingly to show how bad each borough was affected by COVID.

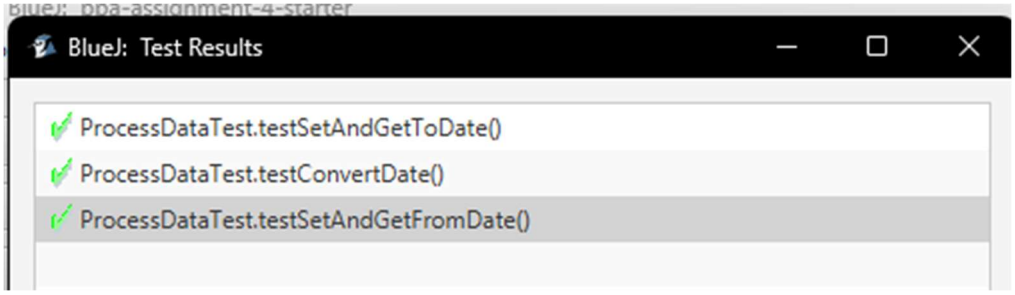
Statistics Panel (Panel 3):

For the design of this panel, we have decided to create two large buttons on the left and the right of the window. These both display arrows on them to show the user which way they will be navigating. You then also have two labels in which one initially has a prompt to click so that the statistics can be displayed and the other has the actual statistic number. We calculated the averages by just taking the end period and the start period and subtracting them to find the data between that month. We then divide that by the number of days or by 33 dependents on if it's the average for that borough or for the period. For example, to calculate total deaths, we incremented the total deaths for the to and from date and then subtracted from them and returned that value.

Most Affected Boroughs Panel (Panel 4 – Challenge Task):

Unit Testing:

For the unit testing, we created a test class for ProcessData() in order to check the functionality of all testable methods. Some methods we weren't able to test as they relied on mouse events which we cannot manually input without causing errors.



This was the output of the given test class, as you can see the methods are tested and given a pass or failure. In this case, all methods tested have passed and is working correctly when ran using the main class.

Contributions:

Panel 1: 10% Talal, 30% Hasan, 30% Shahriar, 30% Christopher

Panel 2: 60% Hasan, 10% Christopher, 20% Shahriar, 10% Talal

Panel 3: 35% Christopher, 35% Shahriar, 15% Hasan, 15% Talal

Panel 4: 40% Christopher, 40% Talal, 10% Hasan, 10% Shahriar