4CCS1PPA Programming Practice and Applications

Coursework 4: London COVID-19 Statistics

Michael Kölling and Jeffery Raphael
Questions? programming@kcl.ac.uk

Your task is to develop a GUI for exploring COVID data collected in London. You will be working with real-world data from the Greater London Authority. The dataset is a combination of information from the UK government and Google's Mobility Report. The dataset is provided to you through two Java classes, CovidData & CovidDataLoader, which are available on KEATS. You should take some time to familiarise yourself with the methods that you have access to through these classes.

Register Group

Coursework 4 is a group assignment. You must work in a group of four (or three with permission from a module leader). Everyone in the group will receive the same mark. If a group member becomes unresponsive, the other group members should keep records of contact and share those records with J. Raphael (at jeffery.raphael@kcl.ac.uk). However, a reasonable attempt should be made by the remaining group members to complete the task.

Each group member must complete the *Assignment 4: Group Selection* quiz. Your responses will be used to track group formation and identity any students that have not been able to find a group. If at anytime your group members change, please update your responses. The quiz will be available for the duration of the assignment. If you are unable to join a group, speak to your TA. The deadline for creating a group is **Sun., Mar.** 12^{th}, after which you will be assigned to a random group.

GitHub Repository

One member of each group is responsible for the following.

- Selecting a group name (humorous names are welcome)
- Creating a central repository to be shared by the group on the KCL GitHub Enterprise system here: `https://github.kcl.ac.uk/`.
- Creating a text file in the repository with the group name as the title, and the group name as the first line of the file.

You're not required to create multiple branches, the group can use a single branch.

Base Tasks

Application Window

Your application should be designed to display a dataset in a digestible form for a user. To do this, we will create a multi-panel and multi-window application using JavaFX; you are free to use SceneBuilder if you like. The first and main window of this application should look similar to the following example.

The window is designed to hold a series of different panels. Key behaviour that should be offered by this display is as follows.

- There should be the ability to move left and right through the panels contained in the centre of the display using *back* and *forward* buttons in appropriate positions. In the example, these are shown in the bottom left and right corner of the window. If there are no additional panels, then the button should cause the display to loop round to the first (or last) panel.
- The top right of the frame should feature two drop-down boxes, appropriately labelled, allowing a user to select a date range for the period they want to analyse.
- The user should be alerted if they have selected a date range that is invalid, e.g., the *from* date is after the *to* date or there's no available data for the selected date range.
- The *back* and *forward* buttons should be initially disabled until the user has selected a date range. This is because the other frames available are going to process and display the data loaded when a user selects a date range, and are thus initially empty. The first panel, however, is unconnected to the data, and should thus be shown to the user when the frame first loads.

Panel 1: Welcome

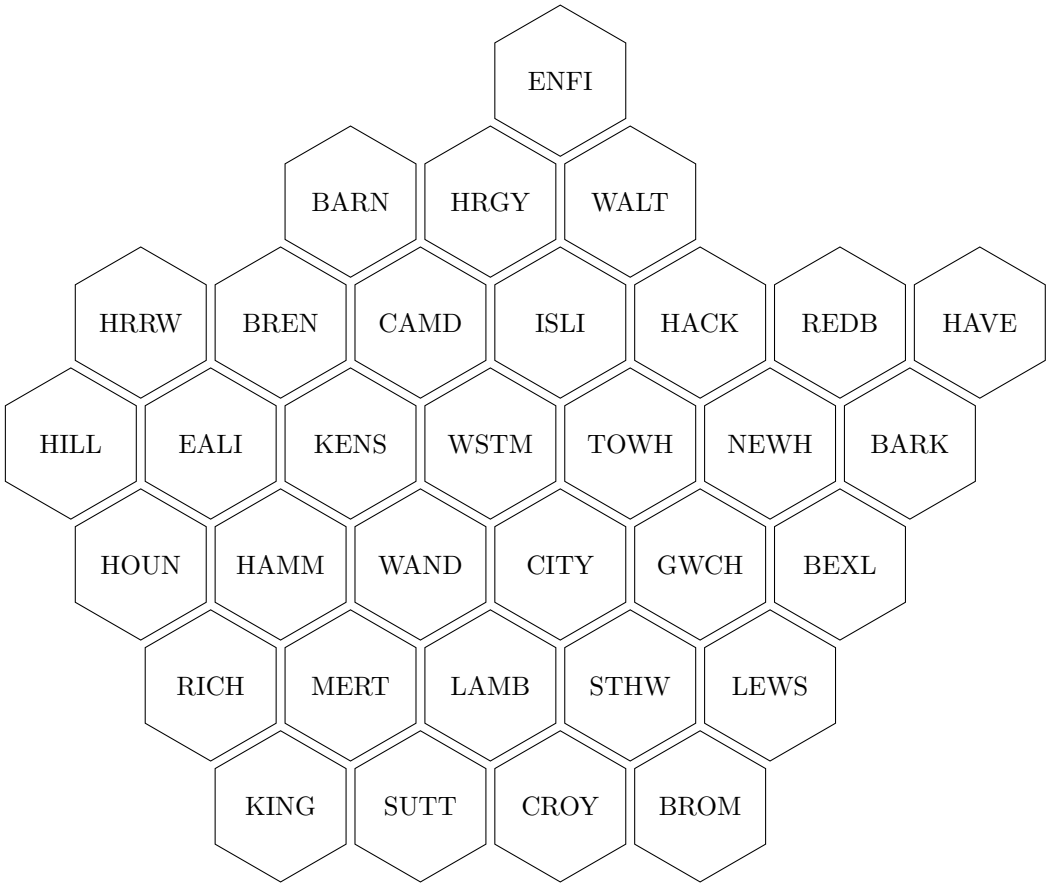
Having a user land on a blank screen like the one shown in the sample window isn't a good design. Instead, the first panel loaded into the main window when a user loads the application should welcome the user, and give them instructions on its basic use. In addition, once a date range is selected by the user, the first window should show the current date range.

Panel 2: The Map

After a date range is selected, the second panel should become available. The second panel displays a visual representation of COVID death rates. This should be done by showing the user a map of London's boroughs on the panel. Each borough on the map should be clearly labelled with its name.

Further, for each borough, there should be some visual indication on the map of how many deaths were recorded in that borough. For example, the visual indication could be done by adding markers to each borough where a large marker indicates there are more deaths in the borough. Or you could colour each borough depending on the number of deaths, e.g., a light green could denote a low death rate while a darker green could denote a higher death rate.

Feel free to make use of the graphic below if you want to (a PNG version is included in the project folder from KEATS). You may use some other visual representation or map if you want to, but your visualisation should be somewhat geographically accurate.



Borough Data

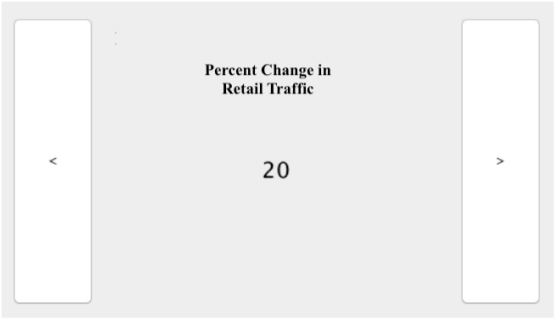
Upon a user selecting a date range, and after the appropriate visualisation of the data is displayed, it should be possible for a user to click on any of the boroughs in order to see more COVID information for that specific borough. This information should be presented in a new window, which opens when a borough is clicked. The title of the new window should include the full name of the borough that the user has selected. The window should also display a simple list of available records with the following details.

- Date
- Google mobility data
- New COVID cases
- Total COVID cases
- New COVID deaths

The window should have an additional drop-down menu that allows the user to select whether to sort the results by any one of the columns, e.g., new COVID cases. Making a selection from the drop-down menu should reorder the list automatically.

Panel 3: Statistics

This panel will present a series of statistics for the period that was selected. You will derive **five** statistics over the available data. The panel should be separated into three distinct sections, as shown in the example below.



Using the buttons, the user should be able to click between the different available statistics (described in the next section), in a similar manner to the way in which they are able to click between different panels of the main window.

Statistics

You must implement five statistics for the selected period

- Average over any **two** Google Mobility measures
- Total number of (*total*) deaths
- Average of *total cases*
- The date with the highest *total death*

Unit Testing

You should provide suitable unit tests for **one** of the classes in your project. Your testing should be thorough and appropriate. You should not select one of the classes that were provided by us. You should pick a class of your own that is complex enough to warrant testing.

Challenge Tasks

Once you have completed the base tasks, you can implement a fourth panel. The fourth and final panel is your blank canvas, designed for you to make your program do something creative and interesting. Be as creative as you like, but make sure your panel provides new functionality.

Reports

You must also write a report (*no more* than four pages) describing your application. The report should include the following.

- The names and student numbers of all students who worked on the submission.
- A description of the GUI and functionality of all four panels (and any windows)
- A description of your unit tests.
- Identify which student completed which task(s), i.e., describe everyone's contributions and the code they were responsible for implementing.

Submission and Deadline

- You must submit your assignment on Gradescope via KEATS by **Wed., Mar.** 29^{th} **16:00 (4pm)**. You'll submit a zip file containing the following
  1. A Jar file of your BlueJ project. —You can create a Jar from within BlueJ by going to Project, and then "Create Jar File...". You do not need to change any of the default options, and so you should just click the "Continue" button.
  2. A report (pdf file *only*)
  3. All of your Java files
- **The Jar file must contain your source code, i.e., the *.java files, and it must run on BlueJ.**
- Click the *Assignment 4: Submission Link* to submit your work. Follow all instructions in the 'Student Submission Guide'. If you have any trouble submitting your work, email Jeffery Raphael as soon as possible. Do not wait until the last hour to attempt your first submission.
- Marking details can be found in the 'Marking Rubric'.

Nominate Who Will Submit the Assignment

- Only **one** group member has to submit the group's assignment —you must discuss and decide which group member will be responsible for submitting the assignment.
- **The person submitting the assignment.** This person will take responsibility for uploading the correct files, submitting everything before the deadline, and for adding all other group members to the submission record.
- **Other group members.** If you are not submitting the assignment on behalf of the group, keep an eye out for a notification email that will let you know when your partner has submitted the assignment. The group member submitting must add you to the submission record. Only after they have done this will you become part of the assignment group on Gradescope.

Late Submission Policy.

All coursework must be submitted on time. If you submit coursework late and have not applied for an extension or have not had a mitigating circumstances claim upheld, you will have an automatic penalty applied. If you submit late, but within 24 hours of the stated deadline, the work will be marked, and 10 raw marks will be deducted. If this deduction brings your mark for the assessment below the usual pass mark (40%), your assessment mark will be capped at the pass mark. **All work submitted more than 24 hours late will receive a mark of zero.**