KING'S COLLEGE LONDON

4CCS1PPA PROGRAMMING PRACTICE AND APPLICATIONS

Fourth "Air Pollution" Coursework (Mar 2025)

Project Name: England is My Polluted City

Student Name: Mehmet Kutay Bozkurt
Student ID: 23162628

Student Name: Anas Ahmed
Student ID: 23171444

Student Name: Matthias Loong
Student ID: 23078800

Student Name: Chelsea Feliciano
Student ID: 22042916

1 Introduction

This project implements a fully functional JavaFX Application to display historical Air Pollution Data of the whole United Kingdom on an interactive map. Users are able to explore the historic pollution levels between 2018 and 2023 on a colour coded map, as well as view detailed location and pollution data at specific data points. There are many quality of life features implemented, such as a robust GUI to filter through different years and pollutants, an extensive statistics framework, as well as a "colourblind mode" to make the pollution viewer more accessible.

This project fulfils all of the base requirements and extends functionality greatly by implementing features such as Live Air Quality API integration and a highly detailed statistical analysis tool. The challenge extensions are listed in greater detail in the report (see section 6).

Additionally, this project takes great care and consideration for high cohesion and low coupling in the codebase, making all classes modular and following the Model View Controller design pattern to fully separate UI functionalities from the logic of the program.

2 Directions for Use

The main application class `App` can be found inside the app folder, and running it as a JavaFX application starts the programme. Additional libraries are used in the programme to increase the usability of the application. These libraries are Gluon Maps (and its dependencies), OSGB, GeographicLib, and Gson. All of these libraries are already inside the given BlueJ project.

3 Roles and Responsibilities Breakdown

Overall, the codebase was equally worked upon by the entire team. Contributions are demonstrated in the commit history best, but the following outlines the main tasks each member worked on.

3.1 Mehmet Kutay Bozkurt

- Created the statistics framework, created and added all of the statistical calculators and UI panels. Created the statistics controller to go through different panels.
- Created pop-up on the map to view detailed information about location.
- Created the API that accesses the Postcodes.io end-point to get detailed information about the location, such as region, borough, and constituency.
- Created the singleton class `DataManager` to be the only entry point to the pollution data.
- Started the colouring framework for the map pollution values.

3.2 Anas Ahmed

- Pollution layer: Rendering of pollution polygons onto the map, handling scaling, sizing, mapping squares onto spherical world-map and optimisation with LODs (Level Of Details).
- Pollution threshold system.
- Improvements and small refactors in colour system and legend.
- Major refactors throughout the integration of UI and backend elements (model-view system).

3.3 Matthias Loong

- Implemented the `DataPicker` class to dynamically select a data set according to the pollutant and year with the GUI side panel and `ComboBox` selectors.
- Implemented Real-Time Air Quality Index API.
- Updated Information Pop-Up to display more detailed information at a given data point.
- Created About and Welcome Page views.

3.4 Chelsea Feliciano

- Created collapsible legend pane.

- Added colourblind colour scheme.
- Created zoom Control Buttons — Zoom In/Out and Fullscreen.
- Created the cat button to toggle side panel visibility.
- Created and modified the design and layout of UI components (apart from statistics related stuff).

4 Base Tasks

4.1 Welcome Panel

A welcome panel was implemented as an additional window that greets users when launching the app. Users can scroll through screenshots and visual instructions on how to use the app effectively. If needed, the welcome screen is easily accessible again by clicking on "Help" and then "Tutorial" on the navigation bar.

Error handling was implemented for image file paths in the case they get deleted or are entered incorrectly in the codebase, when the screen is being updated.

Additionally, an "About" panel was implemented that gives detailed acknowledgements to the authors of the app along with credits for the external libraries and APIs used.

4.2 Data Visualisation Panel (Map View)

The map was implemented using Gluon Maps, an external maps library created by the maintainers of JavaFX. The map displays all pollution data provided from the CSV files and colour codes them on the map. While the map starts with a view of London, the user is able to drag around the map to take a closer look at specific areas of London by zooming in, or view the entirety of the UK by zooming out. This can be done with the scroll wheel or with the zoom buttons on the upper-right hand corner of the screen.

A new `PollutionLayer` class was created to process colours and visualise the pollution data on the map accurately. For smoother performance, Level of Detail optimisation was implemented. More information about the optimisations are detailed in the "Optimisations" section of this report.

Users are able to view a specified pollutant and year, in addition to being able to change the colours of the map to be colorblind friendly.

Useful buttons on the right side of the map allow users to toggle a legend to view the values of pollution in relation to the colours layered on the map, as well as allowing users to switch between full-screen and windowed mode.

A colour scheme for the map was implemented where the colour of a grid area is determined by its pollution level. The colour scheme is implemented as an interface `ColourScheme` with a single method `getColour`. This allows the application to have different colour schemes for the map, and a way to switch between them quite easily, improving maintainability and responsibility-driven design.

4.3 Pollution Statistics Panel

Various statistical information can be accessed by clicking the "View Pollutant Statistics" on the side panel. Each type of information is divided into different panels, which can be viewed through using the "Next" and "Previous" buttons on the bottom. Different panels include

- "Pollution Hotspots Trends" for recording the highest, median, and the lowest pollution levels with their locations. This panel also integrates with the Postcode API to showcase where the area is located.
- "Average Pollution Trends" for showing how the pollution values changed over time with the mean, median, and standard deviation being shown for each year. Additional information is also displayed for the overall averages and percent changes.
- "All Pollutions" for seeing how much the average value for each pollutant has changed over the years.

The statistics module follows a three-tier pattern: Calculators (backend logic for statistical computations), result objects (intermediate representation for just the data), and panels (visualisation and user inter-

face). Each one of these follows extensibility in mind. Specifically, an abstract class is used for the general `StatisticsPanel` class, which each different type of statistics views `extend`. Each result object and calculator also implement a similar interface called `StatisticsResult` and `StatisticsCalculator`, respectively. With the `StatisticsPanelFactory` and `StatisticsManager` classes, adding new statistics panels is easy and requires no changes to other parts of the code, showing the high level of cohesion and low coupling.

The actual graph charts were also divided into components for any panel to use. `LineChartPanel` expects a variable argumant for the data, allowing additional lines with different colours to be added easily. These components were created for high modularity.

Finally, the UI panels and the backend calculators are not coupled at all. The main view is managed by the `StatisticsController` which calls the `StatisticsManager` to generate the results using the calculators; then, the controller supplies the results to the `StatisticsPanelFactory`, which in return creates the actual panel views.

4.4 Detailed Grid Data

When right-clicking anywhere on the map, users are able to view a pop-up with all relevant information for a data point, such as: Latitude and Longitude Coordinates, Unique Grid Code, Pollution Level for the selected Pollutant and Year, Borough or County, and nearest UK Postcode.

The grid data also implements an API (explained in-depth in section 6.3) to retrieve real-time air quality updates and displays it accordingly.

Users can click anywhere on the map to close the pop-up. Right clicking another location on the map will close the current pop-up and open a new one with the correct information for the selected data point.

5 Unit Testing

Unit testing was implemented with JUnit for various classes and public methods in those classes to test the *interface* of those classes. These can be found under the `test` folder in the source code. The tests ensure that the methods are working as expected and use different forms of input ranging from valid to invalid. For example, for querying location data, both valid coordinates within London and invalid coordinates were tested to ensure successful querying and correct error handling. Tests were also created on all of the statistics calculators to ensure that they give out correct information. This was done by calculating statistical values differently to how they are in the calculators. Overall, these tests were vital in adding functionality to the codebase, as they ensured that new functionality did not break any existing functionality.

6 Challenge Tasks

6.1 Interactive Map

The GluonMaps external library was used to implement an interactive map. It features tile-based maps, allowing for easy management of zooming in and out and dragging of the map within the view. Furthermore, the library features robust information, allowing us to convert locations on the screen to coordinates for easy mapping of the data provided.

6.2 Adding the Entire UK in the Map

Given that the project was already utilising an external mapping library and the data provided was for the whole United Kingdom, we decided to expand the coverage of the project to account for all data points.

6.3 API Integration

The project makes use of two external APIs to display information related to the data provided:

- The Postcodes API allows the usage of provided coordinates to return details about the selected data points: UK Postal Code, County/Borough, and Country.
- The AQICN API was used to get real-time air-quality updates from the World Air Quality Index and display the values to the user along with

the time the Air Quality Index was last updated.

6.4 More Extensive Graph Based Trends

Two additional panels were added for more statistical information on the pollution data. These are:

- "Histogram" for seeing the distribution of pollution value ranges—displayed on a bar chart with a logarithmic scale. Additionally, specific values can be seen by hovering the bars to open a tooltip.
- "Distribution Analysis" to showcase the percentiles over time and some more statistical values of the data (such as skewness and kurtosis). Hovering over the table shows what these specific values mean.

These panels were added using the framework mentioned above, where two additional components were added (`DataTablePanel` and `HistogramChartPanel`). Tooltips for each cell in the table from `DataTablePanel` was added to explain the meaning of the values. For `HistogramChartPanel`, an additional `LogarithmicAxis` class was added to show a logarithmic scale for better viewing.

Overall, these modular components enhanced how the statistics information is shown.

6.5 Pollution Thresholds

Users are able to use a slider that will display areas with a certain pollution percentage and other areas with a gray colour.

7 Code Optimisations

Various optimisations were made to the codebase to vastly improve performance. These include:

- LOD (Level of Detail) was implemented to decrease the number of grid areas rendered on the map, and the number of data points rendered in the graph as the user zooms out.
- Polygon view-culling was implemented to only render grid areas that are visible on the screen.
- Multithreadding was added to fetch the LOD values, data from the CSV files, and generate the statistics for multiple years. This allowed all of the values to be loaded and generated in parallel, increasing the efficiency.

8 Bugs and Issues

Most issues were identified and fixed thanks to rigorous testing, as mentioned above, but some issues were identified and could not be fixed due to time constraints:

- Loading Pollution Statistics for the first time before caching will take a few seconds, causing some lag. All subsequent loading of the page will be instant.
- When zooming in and out of the map view quickly, due to the loading of the LODs, the map stops for a couple moments. This is due to the loading of the LODs and the rendering of the polygons, which was deemed unavoidable.
- Opening the About or Welcome page when in fullscreen might result in unexpected behaviour, which seems to be a JavaFX bug only happening on MacOS.
- Most issues were identified and fixed thanks to rigorous testing, as mentioned above.

9 Libraries Used

Various APIs and external libraries were used in this project for extended functionality:

- Gluon Maps: Maps library that implements OpenStreetMaps.
- OSGB by DST: Library to convert Easting and Northing to Latitude and Longitude.
- GeographicLib: Used for geodesic distance calculation.
- Gson: A library by Google to serialise JSON to Java objects.
- Postcodes.io: API used to get location and address data.
- World Air Quality Index API: API used to get real time air quality index updates.