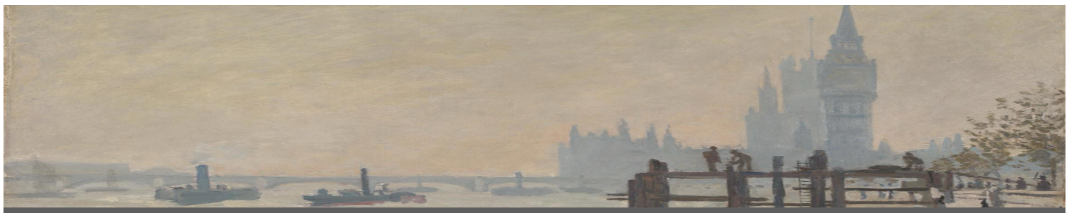

4CCS1PPA

Assignment 4: London Air Pollution

Overview

London has been battling with air pollution for some time. DEFRA (the UK Department for Environment Food and Rural Affairs) monitors and publishes air pollution data in an effort to support the reduction of pollution[1].

For this assignment, you are given data files with pollution data for three different pollutants (Nitrogen dioxide (NO_2) and Particulate Matter (PM10 and PM2.5)) for six years (2018 to 2023). Your task is to develop a Java program with a GUI to display some of the data.

This coursework is a group assignment. **You must work in a group of four**. You will not be allowed to submit if you are not in an approved group.

The Air Pollution Data

You will see a BlueJ project named *Assignment-4-starter* in KEATS. Download this to get started.

The project includes the data files in a subfolder called *UKAirPollutionData*. The data files are csv files with a specific header.

The starter project also includes four classes to help you get started:

- **FileLoadDemo**: A short class to show you how to load a dataset from file. This is for illustration.
- **DataLoader**: A helper class that can load a dataset csv file and return is as a **DataSet** object. You will use this class to load the files. You do not need to modify this class or understand the details of its implementation.
- **DataSet**: An object of this class holds all the data for one dataset (one type of pollutant for one year). The dataset includes some information about the type of data and a list of data points (type **DataPoint**). You will need to work with objects of this class.
- **DataPoint:** This object holds one data point. The data point consists of a geographical location and the pollution value for that location (the yearly mean for that location).

The **DataPoint** is a record class (discussed briefly in the LGT; if you missed it, read up on it). It contains four bits of data:

- A unique code (ukgridcode) for each 1x1km cell in the UK. See https://www.ordnancesurvey.co.uk/documents/resources/guide-to-nationalgrid.pdf

[1] See https://uk-air.defra.gov.uk/data/pcm-data for the data files

- The x and y coordinates for the centroid of each grid cell. These are referred to as "Eastings" and "Northings" in the Ordinance Survey map data (see https://getoutside.ordnancesurvey.co.uk/guides/beginners-guide-to-grid-references)
- The value for the metric itself.

For the geographic location, you can use either the gridcode or the x/y (Easting/Northing) pair (you don't need to use both). Using the x/y coordinates is probably easier.

If you are curious, a tool to translate between the different coordinate systems is here: https://gridreferencefinder.com

---

The Map

At the heart of your program will be the display of the data on a map of London. A map image is provided with the starter project (named *"London.png"*).

The map image is shown below, together with the coordinates of the four corners of the map. When you process the pollution data, you can check whether a data point is on the map by checking whether its x/y (Easting/Northing) coordinates are within the map range. (And you can calculate the exact location on the map.)

Top left: X (Easting) , Y (Northing) 510394 , 193305 Latitude , Longitude (decimal) 51.627741 , -0.40653443	Top right: X (Easting) , Y (Northing) 553297, 193305 Latitude , Longitude (decimal) 51.627741, 0.20205370
Bottom left: X (Easting) , Y (Northing) 510394, 168504 Latitude , Longitude (decimal) 51.395246, -0.40653443	Bottom right: X (Easting) , Y (Northing) 553297 , 168504 Latitude , Longitude (decimal) 51.395246 , 0.20205370

Base Tasks (60 points)

Your task is to write a Java GUI program, using **JavaFX**, to visually display some of the data. The application window should have multiple panels/windows/tabs:

1. **A Welcome Panel**

   - A welcome screen with **instructions** on how to use the application.

2. **Data Visualisation Panel (Map)**

   - A **map-based representation** of air pollution data for London.
   - Pollution data should be represented at its correct location on the map.
   - The user should be able to **hover over** or **click on** data markers to see detailed pollution metrics.
   - Colour-coded visualization: **different colours** should indicate different pollution levels.
   - Users should be able to select pollution results for the three different pollutants (NO_2, PM10, PM2.5) and different years.

3. **Pollution Statistics Panel**

   - Provides summary statistics:
     - **Average pollution level** over the selected period or area.
     - **Highest pollution levels** recorded and their locations.
     - **Trends over time** with a simple graph.
   - The user should be able to **switch between different statistics**.

4. **Detailed Grid Data**

   - Allows users to select an individual data point and view:
     - **Unique Grid Code**
     - **X and Y coordinates**
     - **Pollution level** for the selected period.

Unit Testing (10 points)

You must write unit tests for at least one of your classes. The chosen class should be complex enough to require thorough testing. **JUnit** should be used.

Challenge Tasks (20 points)

Once you complete the base tasks, consider implementing **additional features**, such as:

- **More extensive graph-based trends** of pollution over time.
- **Custom filtering options**, allowing users to view only highly polluted areas.

- **Comparisons between different years** of pollution data.
- **Integration with external APIs** for real-time air quality updates.
- **Adding additional map regions.**
- Adding an **interactive map view** (using a JavaFX component to display an *OpenStreetMap* map view, such as mapjfx or LeafletMap).

…or other interesting features you may think of. Surprise us!

As always, if you are considering your own challenge tasks, check with your class supervisor first.

Reports (10 points)

A report (max **4 pages**) must be submitted, detailing:

- Group member contributions (who implemented what).
- Description of the GUI and functionality of all panels.
- Explanation of unit testing performed.

Group work

This assignment must be done in a group of four. Groups must be formed by the **group formation deadline**, which will be posted separately.

You can form your own groups, or you can select to be assigned to a group. Information about this process will be provided separately. Each group member must complete the *Assignment 4: Group Selection* quiz to track group formation.

If you are not in a group of four by the group formation deadline, you will not be able to submit the assignment, and **you will receive a mark of zero**.

It is your responsibility to form or join a group by the deadline.

If a group member becomes unresponsive, keep a record of communication attempts and notify the module leader immediately.

Github Repository

You must use a GIT repository for your project. 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 filename, the group name as the first line of the file, and the group members in the text of the file.

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

Submission and Assessment

You must submit a zip file via the link “Assignment 4: Submission Link” in Keats 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. The Jar file must contain your source code, i.e., the *.java files, and it must run in BlueJ.
2. A report (PDF)
3. All of your Java files (*.java)

Your assignment will be penalised if you are missing any files. We strongly recommend that you review your submitted files after submission.

Nominate Who Will Submit the Assignment

- **Only one** group member has to submit the group’s assessment – 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 the other group members to the submissions record during the submission.
- **Other group members:** if you are not submitting the assessment 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. After they have done this, you will receive an email and you will be properly added to the submission.

Your code will be assessed for

- Correctness
- Functionality
- Appropriate use of language constructs
- Style (commenting, indentation, etc.)
- Design (code quality, class structure, appropriate use of inheritance)
- Difficulty (marks for extension tasks)

Deadline

This assignment (code and report) is due on

Monday, 24 March 2025, 4pm

All coursework must be submitted on time. Do not wait until the last hour to attempt your first submission as there can sometimes be technical issues. 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.