1. Instructions

Students are asked to work in groups for this assignment and are encouraged to seek help from classmates and teaching staff. Each group needs to create a new GitHub repo and collaborate on that central system. All source code and resources must present on GitHub too. Moreover, all commit messages must be meaningful and consistent.

2. Assessment details

In this assignment, you are asked to implement a text-based program using Java language and OOP techniques. The details of the assignment will be released around week 4. There are 3 main parts:

- OOP design and implementation: you need to design and implement a class hierarchy to make your program flexible and easy to maintain
- Problem-solving tasks: you need to apply control statements, algorithms, data structures, etc. to solve particular tasks
- Video demonstration: you need to create a short video (less than 10 minutes) consisting of 2 parts. Part 1 explains how you analyze, design, and implement the system. Part 2 shows a working demo of your system

3. Problem details

The COVID-19 pandemic is becoming worse over time. Although not having a firm background in medicine and health, you want to contribute to the community in fighting this battle. By reading science articles, you know that health decision-makers need to process a lot of data to make informed and insightful decisions. Possessing several RMIT programming–related courses' HD under your belt (even though your lecturer deliberately limits the number of HDs to just around 10%), you know that you can create an intensive data processing and analytics tool that may be beneficial to policymakers.

Below is the technical and functional description of this project.

The most important weapon you need for this project is data. Fortunately, pandemic data is made available by many organizations. The data file used in this project is provided by WHO. It's a CSV file. However, it has been deleted some unused columns. The column names are straightforward as well as the data.

You can get the data here. (Links to an external site.)

Note: I have not checked the data thoroughly. From what I learn so far, the data is sorted by country, then by date. The dates are consecutive without any gaps. If there are any missing dates (i.e., there are gaps), you can treat the missing dates as they contain all zero)

You will use the OOP paradigm to design and develop this project. There are 3 main object hierarchies in this project.

1. Data: each data object has a geographic area, which can be either a country (e.g., Vietnam) or a continent (e.g., Asia), and a time range. The time range can be specified as one of the following:

- A pair of start date and end date (inclusive) (e.g., 1/1/2021 and 8/1/2021)
- A number of days or weeks from a particular date (e.g., 2 days from 1/20/2021 means there are 3 days 1/20/2021, 1/21/2021, and 1/22/2021)
- A number of days or weeks to a particular date (e.g., 1 week to 1/8/2021 means there are 8 days from 1/1/2021 to 1/8/2021)

2. Summary: this is the data after processed and ready to display. To create summary data, original data are grouped (2.1), a metric is chosen (2.2), and a result is calculated (2.3). The possible ways of specifying groupings are (explanation of 2.1):

- No grouping: each day is a separate group.
- Number of groups: a number is specified and you must divide your data into that number of groups. You need to divide your data as equally as possible. For example, if your data consist of 10 days (day 1 to day 10) and 3 groups are needed, then you can divide your data into either

[day 1 to day 3], [day 4 to day 6], [day 7 to day 10]

[day 1 to day 3], [day 4 to day 7], [day 8 to day 10]

[day 1 to day 4], [day 5 to day 7], [day 8 to day 10]

Note that you can only put consecutive days into a group. In other words, putting day 1, day 2, and day 4 into a group is invalid.

- Number of days: a number is specified and you divide your data into groups so that each group contains that number of days. For this grouping, if it is not possible to divide groups equally, raise an error telling the caller about that. For example, if your data consist of 10 days (day 1 to day 10) and 5 days per group are needed, you can divide your data into 2 groups. “Number of days” like 3 or 4 are invalid in this case.

After specifying a grouping method, a metric is chosen. There are 3 possible metrics (explanation of 2.2): positive cases, deaths, and people vaccinated.

Finally, users of your program can choose one of the following result types (explanation of 2.3)

- New Total: total new cases/new deaths/new vaccinated people in a group.

Note: the vaccinated column contains the accumulated values up to a date, not new data for each date as new cases and new deaths columns. So, you can calculate vaccinated2 – vaccinated1 to get the new vaccinated after day 1 up to day 2

- Up To: total cases/deaths/vaccinated from the beginning up to the last date of a group

3. Display: summary data is displayed to viewers. There are 2 ways to display data

Tabular display: display summary data in a table. There are 2 columns: the first column named "Range" and the second column named "Value". In the table, display a row for each group. For each group, the "Range" column shows "date1 – date2" where date1 and date2 are the first date and last date of a group respectively. If a group contains just 1 date, shows that date only. The "Value" column of a group shows the calculated value (New Total or Up To) described above.

Chart display: display summary data in a textual chart. The chart area consists of 24 rows x 80 cols. The x-coordinate direction is left to right, and the y-coordinate direction is from bottom to top. The x-coordinate represents the groups and the y-coordinate represents the calculated summary results. You should position the groups as equally as possible on the x-coordinate. And you should use the minimum and maximum result values to position a result on the y-coordinate linearly. The left-most column should display all | (pipe) characters while the bottom-most row should display all _ (underscore) characters. (That means you have 23 rows and 79 columns left to display data points). Each summary data point is represented as an asterisk *.

User interface

Your program should show a menu that lets users choose data (area and range), summary (grouping condition, metric, and way of calculation), and display (tabular or chart). Users can continue as many times as they want. They can choose to end the program when they finish.