IEU SE 115 Project Description FALL 24-25

Please read the entirety of instructions very carefully before starting the project!!!

This project will be conducted individually. There will be no groups.

Version Control System: Git

We want you to learn the common "version control system" software that is used everywhere: Git. A version control system helps us to keep track of our changes in time. It is also used as a collaboration tool, which means it helps you to develop software with other developers. Git is one of many version control systems, but currently the most popular one. Therefore, it is vital for you to learn what Git is, what it does, how it works, and how to use it properly. This project is a great starting point to learn Git.

Git software is available at https://git-scm.com/ and it contains documentation, along with videos, at https://git-scm.com/doc or at https://skills.github.com/. However, the documentation also talks about some more advanced things you can do with it. Please start using Git in your project from the very first day.

While Git can be used on your own local system, it is mostly used with a remote (online) server. The most common one is the GitHub, at https://github.com/. Registration is free, so make sure you create a profile if you haven't done so already. You can create a repository for your project, and keep your source code online. Make sure you keep your project private, so that only you can access it.

Report

A short but formal report should accompany your source code, free of spelling or grammar errors or colloquialisms. In this report, indicate if you have failed to correctly implement any functional requirements, discuss your design and implementation choices, the challenges you have faced and how you have solved them. **It should also include the execution output of your program with the data provided.** This can only be done if you keep taking notes during the development. The report must be in PDF format. If you submit in another format, such as "docx" or "pages" or anything else, we will not accept it, and you will not get any points for the entire project.

Submitting Your Project

- A ZIP file that contains your source code.
- A JAR file that runs your program.
- A PDF file that contains your project report.
- A PDF file that contains screenshots of your Git commits.

The due date is Dec 22, 23:59. **Late submissions will NOT be accepted.** There will be an oral exam in the last two weeks of the semester, where you will demo and answer questions about the design and implementation of the project.

Grading

Any software project has a list of requirements. Some of them are "functional" - which are about the features the software should have, and others are "non-functional" - which are about the conditions that the software will be running.

- **Failing any "non-functional requirement" results in a grade of zero.**
- Report is 20 points. Evaluation criteria includes the narration quality, and the software design choices.
- Functional requirements explained in detail below, 80 points.

Non-functional Requirements

Non-functional Requirement 1: The program must be implemented in the Java programming language. Source code should be delivered as a .zip file, no other format is permitted.

Rationale: Since we are learning programming using Java, this is expected.

Non-functional Requirement 2: The development must be done on a private Git repository.

Rationale: Version control is important. While you are free to use any Git server, we strongly suggest that you create a GitHub account. Every time you improve your project, commit your changes to the repository, so that we can see your progress at the end of the semester. Install the "GitHub Desktop" program and you will be ready to go. The repository must be private. Students who do not have at least weekly commits will fail this req and receive a 0. Each commit must include the description of the work included in the commit. The deliverables must include your git log screenshots as a pdf file.

Non-functional Requirement 3: The program must be delivered as a JAR file.

Rationale: Instead of several class files, submit the program packed into a JAR file.

Non-functional Requirement 4: The program must be accompanied by a project report in pdf format, as detailed in this document.

Rationale: The design and implementation decisions must be justified.

Non-functional Requirement 5: Every student must be able to compile and demo the program execution, and answer questions about their entire design and implementation.

Rationale: This will be a big factor in grading. Being absent during the presentation session is not acceptable. If the code does not compile or run, you get zero. If it crashes along the way, you lose substantial points. If you cannot sufficiently answer the questions about any aspect of the project (code, design, report, etc.) during the oral exam, then you will lose substantial points overall. Absent students from the presentation get zero from the entire project.

Functional Requirements

Project Description

The goal of this project is to design a Java program called **“SE 115 Maps”** that finds the fastest route using algorithms and object-oriented programming approaches. The program will read a map's initial state from a text file (cities, routes, and time), find the fastest way between two specified cities, and output the solution to another text file.

The map’s input is described as follows:

- Each city is represented by a unique label which can be the first letter of its name (e.g., **A, B, C**).
- Each route is represented by a connection between two cities and a time value that represents the time to go between them.

Input Text File Format

Your program will read from a text file we provide with the following structure:

1. **First line**: Number of cities (e.g., **5**)
2. **Next line**: City labels (e.g., **A, B, C, D, E**), a letter here represents one city.
3. **Next lines**: Number of Routes, where each route is defined in next lines as: **<City1> <City2> <Time>**
4. **Last line**: Starting city and ending city for the fastest route computation.
5. In each line, if there are more than one character, they are separated with 1 whitespace.

Example Input File (map1.txt):

```
5
A B C D E
6
A B 30
A C 10
B D 20
C D 50
C E 40
D E 10
A E
```

This represents a map with 5 cities (**A, B, C, D, E**), 6 routes, and the task is to find the fastest way from city **A** to **E**.

Evaluation Criteria

1. **Input Reading (15 Points)**
    - Read the map from a given text file (user will enter the file name).
    - Parse the number of cities, city labels, routes, and times.
    - Validate that the input file is correctly formatted, with valid cities, routes and times.
    - If the input file is not properly formatted, the output should contain a line for each error of the format "Error Line:" (10 points)
      Ex: <File line number> <Error description>,

      If it is properly formatted, “File read is successful!” must be seen on the screen (5 points).

2. **Object-Oriented (25 Points)**
    - Create a **CountryMap** class to hold the map's structure and related methods, if you use any (10 Points).
    - Design a **City** class to represent map cities (5 Points).
    - Create a **WayFinder** class to implement your fastest way algorithm (10 Points).
    - In this part, you need to choose your own data members, their types and methods for your classes. They need to be meaningful and logical.
3. **Algorithmic Solution (20 Points)**
    - Implement your algorithm to find the fastest way between the start and end cities (15 Points).
    - Ensure your algorithm handles cases where no way exists (5 Points).
4. **Output Handling (10 Points)**
    - Write the solution to an output text file in the following format:

      ```
      Fastest Way: A -> C -> E
      Total Time: 50 min
      ```

    - Ensure accurate and readable formatting like in the example and handle file writing errors (10 Points).
5. **Testing with Instructor Test Files (10 Points)**
    - This step will be done by your instructors, after you finish your project.
    - We will test your program using multiple different input files that demonstrate important cases to check your algorithm’s performance and such.

Please note that evaluation criteria also includes adherence to basic style guidelines, proper comments, and implementation choices.