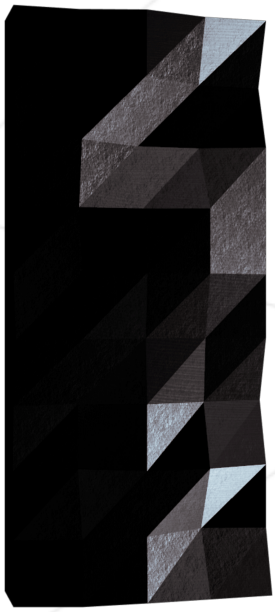

OBJECT ORIENTED PISCINE

Module 05 - Train yourself

Summary: In this module, you will train your object oriented development skill.

Version: 1.1

Contents

Chapter I

Preamble

Railway networks are critical transportation infrastructures that require efficient management and coordination to ensure safe and reliable operations.
Developing simulations of such networks can help us study and optimize their performance, identify potential issues and test various scenarios without incurring real-world costs and risks.

In this project, you will create a simulation of a railway network, including the creation of rails and trains, implementing systems to coordinate their movements and avoid collisions, and introducing random events to simulate real-world conditions.

This project will challenge your programming skills and provide you with hands-on experience in designing and implementing complex simulations.

Chapter II

Mandatory part

In this exercise, you must create a program that simulates a complex railway network with multiple trains operating simultaneously.

As you're in a Object Oriented Piscine, you're expected to use knowledge you learned in previous modules.
As such, we expect you to provide :

- Multiple UML diagrams
- Encapsulated classes/structures
- Design patterns implementations
- Smart use of attributes property.

During evaluation, you will be asked questions about your code and your composition, so be ready to awnser such questions.

II.1 Introduction

The main goal is to estimate the average travel time between two points in the network while managing train overtaking and implementing a pathfinding system to determine optimal routes between stations.

II.2 Requirements

The program should fulfill the following requirements:

- Allow defining a railway network composed of multiple cities and rail segments.
- Model trains with a name, a unique identifier (ID), and performance characteristics (maximum acceleration force and maximum brake force).
- Include a maximum allowed speed for rail segments.
- Implement a random event generator that can occur in cities or on rail segments.
- Simulate the movement of multiple trains simultaneously on the railway network, taking into account the characteristics of trains and rails.
- Design an overtaking system between trains to avoid collisions and optimize traffic on the railway network.
- Implement a pathfinding algorithm to determine optimal routes between stations in the railway network.
- Finally, provide an estimate of the average travel time between two points in the network.

You're expected to work on your architecture. You will be evaluated on this topic, on the following points :

- You must create design pattern (At least 3 to gain the maximum notation about design pattern)
- You must have provided some UML diagram, showing what you plan to do (At least 2 diagrams to gain the maximum notation about UML. One class diagram and one sequence diagram for example)
- You must encapsulate your structures and classes in a clever way.
- You must follow SOLID recommendation, and even the KISS or the DRY one

II.3 Inputs

To describe the simulation to execute, your program will be provided with 2 files as inputs. The first file will describe the rails network you will work on, and the second will describe the trains you need to simulate.

II.3.1 Rail network

This file will contain every rails of the simulation.
It can contain 3 types of datas :

- Node, representing a point in your network graph, and it's name.
- Rail, representing a link between two nodes of your network graph, with it's lenght and the speed limitation in this section.

It will be syntaxed as followed :

```
Node CityA
Node CityB
Node CityC
Node RailNodeA
Node RailNodeB
Node RailNodeC
Node RailNodeD
Node RailNodeE
Node RailNodeF
Node RailNodeG
Rail CityA RailNodeA 15.0 250.0
Rail RailNodeA RailNodeB 20.0 250.0
Rail RailNodeB RailNodeC 13.0 175.0
Rail RailNodeC CityB 5.0 150.0
Rail CityB RailNodeD 7.0 150.0
Rail RailNodeD RailNodeE 12.0 200.0
Rail RailNodeE CityC 6.0 150.0
Rail CityA RailNodeF 6.0 150.0
Rail RailNodeF RailNodeG 17.0 250.0
Rail RailNodeG CityC 17.0 250.0
Rail RailNodeA RailNodeD 23.02 250.0
Rail RailNodeA RailNodeE 19.0 250.0
Rail RailNodeA RailNodeG 6.7 150.0
Rail RailNodeB RailNodeD 20.24 225.0
```

You must verify the datas contained inside this file, and output a clear error message if you find a mistake inside it and close your application.

II.3.2 Train composition

This file will contain the description of every train that your program must simulate. Each train will be describe by, in order :

1. a name
2. the weight of the train, in metric ton
3. the coefficient of friction
4. the maximum acceleration force, in kilonewtons
5. the maximum brake force, in kilonewtons
6. the departure train station
7. the arrival train station
8. the hour of departure
9. the duration of the stop at each station encountered

It will be syntaxed as followed :

```
TrainAB 80 0.05 356.0 30.0 CityA CityB 14h10 00h10
TrainAC 60 0.05 412.0 40.0 CityA CityC 14h20 00h10
TrainBA 40 0.05 356.0 30.0 CityA CityB 14h24 00h10
```

You must verify the datas contained inside this file, and output a clear error message if you find a mistake inside it and close your application.

If your application does not receive the correct amount of argument, you must output a usage, describing how to use your executable. You must also add a –help, that will output how to create input files

II.4 Output

Your program must provide a file for each travel, containing informations about it. Those file must be named "TrainName_TrainDepartureTime.result".

The output file must start follow this format :

```
Train : TrainAB1
Final travel time : XXhXXm
```

You must provide an estimated time for the full travel, based on the acceleration force of the train and the distance to run.

Once you have given those informations, the output file must provide information about what the train have been doing during its travel, following this format :

```
[00h00] - [  CityA][RailNodeA] - [53.00km] - [Speed up] - [x][ ][ ][ ... ][ ][ ][ ]
[00h05] - [  CityA][RailNodeA] - [52.50km] - [Maintain] - [x][ ][ ][ ... ][ ][ ][ ]
[00h10] - [  CityA][RailNodeA] - [51.00km] - [Maintain] - [x][ ][ ][ ... ][ ][ ][ ]
...
...
[XXhXX] - [  CityA][RailNodeA] - [38.01km] - [Maintain] - [ ][ ][ ][ ... ][ ][ ][x]
[XXhXX] - [RailNodeA][RailNodeB] - [30.00km] - [Maintain] - [x][ ][ ][ ... ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
...
[XXhXX] - [RailNodeC][ CityB] - [01.00km] - [ Braking] - [ ][ ][ ][x][ ]
[XXhXX] - [RailNodeC][ CityB] - [00.90km] - [ Braking] - [ ][ ][ ][x][ ]
...
[XXhXX] - [RailNodeC][ CityB] - [00.00km] - [ Stopped] - [ ][ ][ ][ ][x]
```

Each line of this output file must contain the following informations:

1. The time since start
2. The node where the train started
3. The node where the train will arrive
4. The distance left to the final destination of the travel
5. An indication of what the train is doing (Speeding up, maintaining speed, bracking, stopped, or whatever you may need)
6. A simple graph, representing the rail state, from the starting node to the destination node This graph must represent the percent of completion of the distance between those node, and must be represented by one cell per km. If there is another train blocking the way, it must be represented by a 'O'

Chapter III

Bonuses part

You're free to do any bonuses you may think is cool !

We expecte bonuses like :

- Graphical interface with real time rendering of the simulation
- Possibility to add nodes/rails/events during simulation execution via run-time reloading of input file
- A cool output, showing the progression of trains on their travel
- Run the simulation several times, to obtain an average travel time/event list

Chapter IV

Submission and peer-evaluations

For this evaluation, you must provide any documents you may think is necessary to complete both the mandatory and bonuses part : diagram, source files, header, documentation, anything
You won't be evaluated on anything except what's inside the repository.

Turn in your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your folders and files to ensure they are correct.