Frankfurt University of Applied Sciences

– Faculty of Computer Science and Engineering –

Real Time Traffic Simulation with Java Milestone 1

Project Report for the course

Object-Oriented Programming with Java

submitted on November 27, 2025 by

1651339 Son Dang Huu Trung
1647833 Hung Dao Gia
1651313 Uy Pham Khac
1655931 Hoang Bui Huy
1387463 Raees Ashraf Shah

Professor : Prof. Ghadi Mahmoudi

Declaration

I hereby declare that the submitted project is my own unaided work or the unaided work of our team. All direct or indirect sources used are acknowledged as references.

I am aware that the project in digital form can be examined for the use of unauthorized aid and in order to determine whether the project as a whole or parts incorporated in it may be deemed as plagiarism. For the comparison of my work with existing sources I agree that it shall be entered in a database where it shall also remain after examination, to enable comparison with future projects submitted. Further rights of reproduction and usage, however, are not granted here.

This work was not previously presented to another examination board and has not been published.

Frankfurt am Main, November 27, 2025

1651339 Son Dang Huu Trung
1647833 Hung Dao Gia
1651313 Uy Pham Khac
1655931 Hoang Bui Huy
1387463 Raees Ashraf Shah

Contents

Abstract **1**

1 Project Overview **2**

1.1 Introduction . . . 2
1.2 Background . . . 2
1.3 Objective . . . 2
1.4 Technology Stack . . . 3

2 Project Workplan **4**

2.1 Timeline . . . 4
2.1.1 Timeline for Next Milestone . . . 4
2.1.2 Timeline for Final Application . . . 4
2.2 Task Distribution . . . 4

3 Application Features **5**

3.1 Graphical User Interface (GUI) . . . 5
3.2 Class Diagram Overview . . . 6
3.3 Traffic Network Preparation . . . 7
3.4 SUMO Network and Initial Simulation Test . . . 7
3.5 TraCI Connection Test . . . 8
3.6 Use Case Diagram . . . 9

Conclusion **10**

Abstract

This project focuses on developing a real-time traffic simulation application using Java, JavaFX, and the SUMO traffic simulator. During Milestone 1, the primary objective was to design the graphical user interface that will serve as the foundation for future simulation control and visualization.

The interface was created using JavaFX and Scene Builder, following a structured layout to support key features such as simulation configuration, vehicle and traffic light management, map visualization, and performance metrics. Although backend SUMO integration is not yet implemented, the completed UI provides a clear, organized, and fully resizable framework that prepares the application for the technical functionality planned in later milestones.

Chapter 1

Project Overview

1.1 Introduction

This project aims to build a real-time traffic simulation application using Java as the main programming language and its various libraries, such as JavaFX, TraCI as a Service (TraaS), etc. It also utilizes functionalities of the Simulation of Urban Mobility (SUMO) software package to visualize traffic network and traffic flows. The application is designed to provide some basic mobility control functionalities and allows for user's interactive control, enabling its use for teaching, learning, and researching urban mobility.

The concentration of the project is a wrapper runner application that calls functions and accesses objects from external libraries to simulate real-time traffic, resembling a bird's eye view of a complex apparatus with independent components. This is the essence of object-oriented programming (OOP) design.

Working development and documentation of the project can be found in our GitHub repository:

`https://github.com/hunggiadao/realtime-traffic-simulation-java-oop`

1.2 Background

In more detail, the surface of the application is a GUI that is rendered using JavaFX graphical library, which is a Java library that allows for the creation of interface elements like buttons, dropdown menus, and alert boxes. Such elements send command signals to a SUMO instance to initialize and control a running SUMO traffic network. The communication between the GUI frontend and the SUMO simulator is facilitated by SUMO's TraaS library, which provides functions for retrieving traffic data like vehicle statistics, traffic light states, road data, etc., and functions for commanding the network like cycling traffic lights, spawning vehicles, etc.

When a SUMO instance finishes running, its simulation results are exported in the comma separated values (CSV) format for further inspection and improvement of the application. Our documentation and project report include information about the application's features, usability, architecture diagrams, class design diagrams, and a summary of our development process.

1.3 Objective

The primary objective of this project is to develop a comprehensive real-time traffic simulation application that demonstrates practical implementation of object-oriented programming principles in Java. Specifically, we aim to create an

interactive platform that integrates multiple technologies—JavaFX for the graphical user interface, SUMO for traffic simulation, and TraaS for inter-process communication—into a cohesive system that can be used for educational and research purposes in urban mobility studies.

Beyond the technical implementation, we seek to gain hands-on experience with software engineering practices, including collaborative development using version control systems, systematic documentation of design decisions and development processes, and the creation of modular, maintainable code that follows industry best practices. The project also serves as an opportunity to explore real-world challenges in traffic simulation and control systems.

1.4 Technology Stack

To familiarize ourselves with the necessary programs, we watched various tutorial videos and read documentation on SUMO, Netedit, and JavaFX. We then shared these resources within our team so that everyone could make use of them.

At first, we focused on learning Netedit. We watched road simulation videos and studied documentation to gradually understand how the program works. We created different simulations, added traffic lights, stop signs, pedestrian paths, and bicycle lanes, and tested their behavior. In the beginning, it was difficult to understand Netedit, but over time we became more confident in using it.

After we became familiar with Netedit and SUMO, we started assigning roles within the team and defining who would be responsible for which tasks.

Chapter 2

Project Workplan

2.1 Timeline

2.1.1 Timeline for Next Milestone

- **Working Application:** Complete core functionality by 10th December
- **Code Documentation:** Comprehensive Javadoc and inline comments by 12th December
- **User Guide Draft:** Initial version by 13th December
- **Test Scenario:** At least one stress test by 7th December
- **Progress Summary:** Status report and challenges documentation by 10th December

2.1.2 Timeline for Final Application

- **Full GUI Implementation:** Complete map visualization, controls, and statistics dashboard by 5th January
- **Vehicle Grouping/Filtering:** Implement filtering and categorization features by 31st December
- **Traffic Light Adaptation:** Manual and rule-based control system by 31st December
- **Exportable Reports:** CSV and PDF export functionality by 31st December
- **Final Documentation:** Updated user guide and comprehensive code documentation by 10th January
- **Project Summary:** Enhancements overview and design decisions report by 10th January

2.2 Task Distribution

- **Traffic Network, Communication between Java and Sumo:** Raees Ashraf Shah, Gia Hung Dao
- **GUI Mockups, User Interface Design:** Huu Trung Son Dang
- **Wrapper:** Huy Hoang Bui
- **Architecture and UML Diagrams:** Khac Uy Pham

Chapter 3

Application Features

3.1 Graphical User Interface (GUI)

For Milestone 1, we designed the graphical user interface of the application using JavaFX and Scene Builder. The goal at this stage was to establish the complete visual structure of the program, without yet integrating SUMO or implementing any simulation logic.

The interface is built around a `BorderPane` layout, dividing the window into clear functional regions. At the top of the application, a responsive toolbar provides the essential simulation controls, including opening a SUMO configuration file, connecting to the simulation backend, starting or pausing the simulation, executing single simulation steps, and adjusting the simulation speed. This layout remains clean and stable when the window is resized.

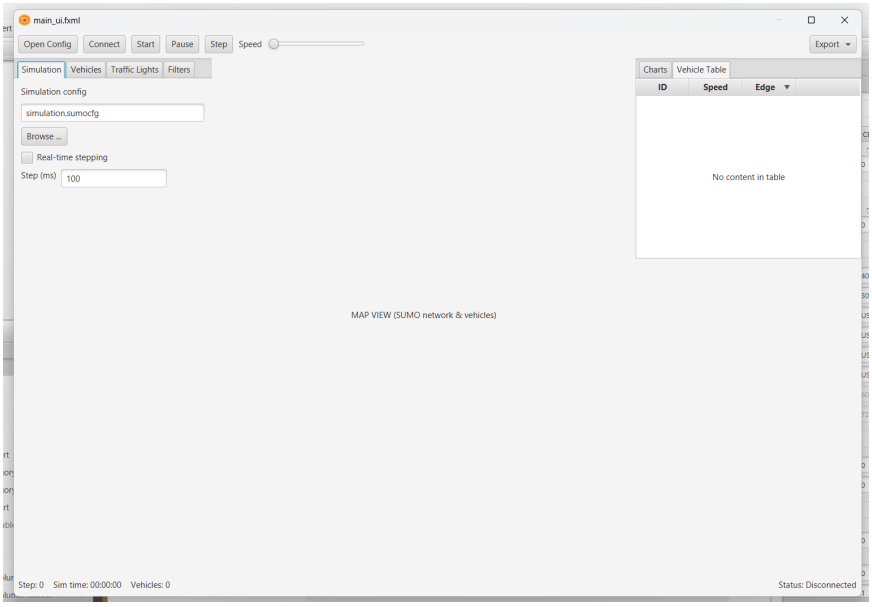

Figure 3.1: Graphical User Interface designed for Milestone 1

On the left side, a `TabPane` organizes input settings into four tabs: *Simulation*, *Vehicles*, *Traffic Lights*, and *Filters*. As shown in the interface preview, the Simulation tab includes fields for selecting a configuration file, enabling real-time stepping, and adjusting the step interval. This structure keeps related parameters grouped and easy to navigate.

The central region is reserved for the map view. In this milestone, it contains a placeholder label indicating where the SUMO network visualization will later appear. The map area is fully resizable and positioned as the main visual focus of the interface.

On the right side, a second `TabPane` is prepared for simulation metrics. The *Vehicle Table* tab includes a JavaFX `TableView` with predefined columns (ID, Speed, Edge), ready to display live data once SUMO integration is implemented.

Finally, a status bar at the bottom of the interface displays key runtime information, such as the current simulation step, simulation time, vehicle count, and connection status. This ensures that essential system feedback remains visible at all times.

Overall, the GUI developed in Milestone 1 provides a clear, organized, and fully resizable framework for the simulation application. It establishes the necessary structure for future milestones in which SUMO communication, map rendering, and real-time traffic data will be integrated.

3.2 Class Diagram Overview

Figure 3.2 shows the simplified class structure implemented for Milestone 1. The architecture is intentionally minimal, as the purpose of this stage is to establish clear responsibilities before full SUMO integration is added.

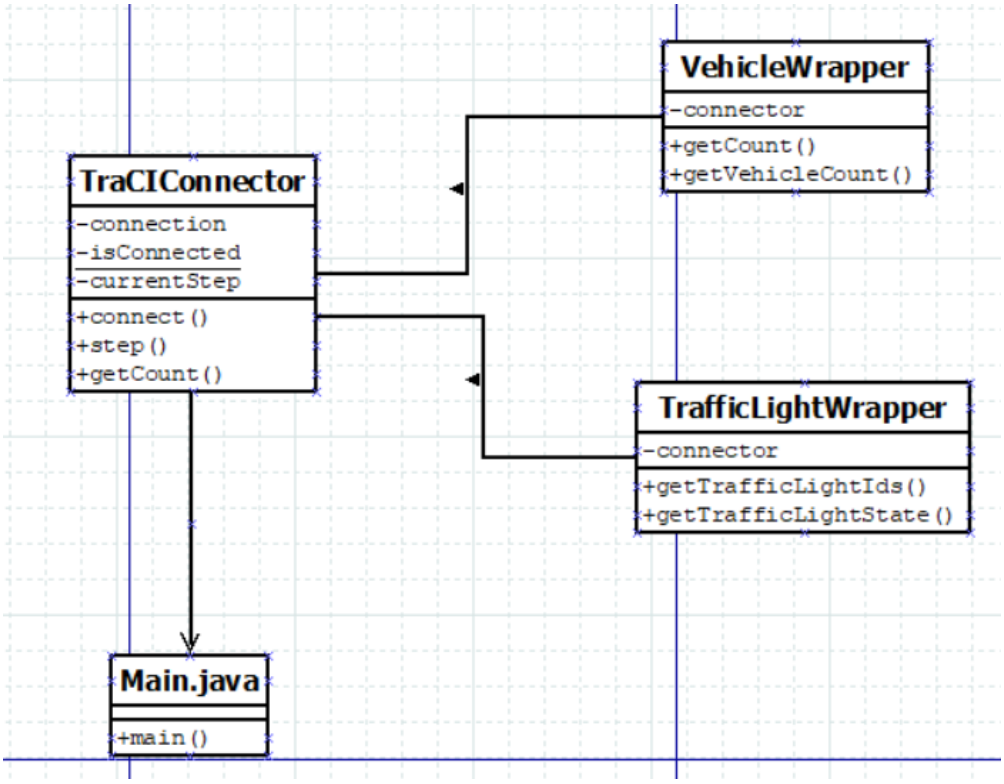

Figure 3.2: Class diagram used in Milestone 1.

The `TraCIConnector` class forms the core of the system, responsible for establishing a connection to SUMO, tracking the simulation step, and providing basic operations such as `connect()`, `step()`, and obtaining simple vehicle counts. The `VehicleWrapper` and `TrafficLightWrapper` classes each hold a reference to this connector and serve as early

abstractions for accessing SUMO vehicle data and traffic light information. The `Main` class simply launches the application. This structure creates a clean separation of concerns and prepares the system for more advanced TraCI functions in future milestones.

3.3 Traffic Network Preparation

To test the SUMO setup during Milestone 1, we constructed a small traffic network based on a real street layout in Frankfurt. The area was selected for its moderate size and clear road geometry, making it ideal for early simulation testing. Figure 3.3 shows the reference map used during the process.

Figure 3.3: Reference street layout used for building the SUMO network.

Using SUMO's NETEDIT tool, the roads, intersections, and lane connections were recreated manually. This allowed the team to produce a valid `.net.xml` file that could be loaded into SUMO for simulation.

3.4 SUMO Network and Initial Simulation Test

Once the road geometry was created in NETEDIT, the network was paired with a basic route file and loaded into SUMO for testing. Figure 3.4 shows the SUMO GUI rendering of the test network.

This simulation confirmed that lane connections, turning paths, and traffic flow behaved as expected. The successful run ensured that the SUMO environment, map setup, and route files were correctly prepared for Java integration in later phases.

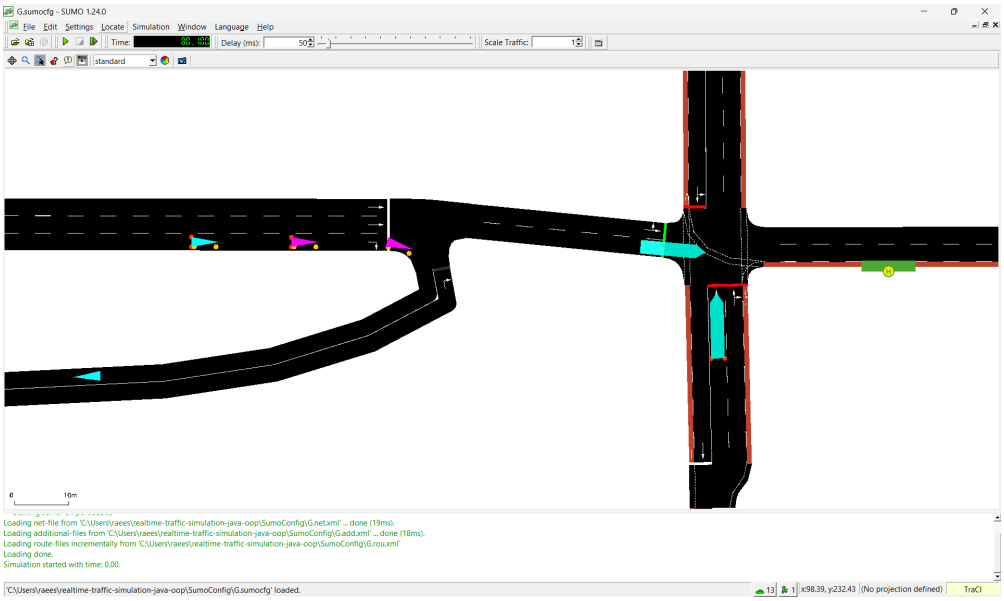

Figure 3.4: SUMO network used for the initial simulation test.

3.5 TraCI Connection Test

During Milestone 1, we performed a small integration test to verify that a Java application could successfully communicate with SUMO via TraCI.

The connector established a SUMO process, stepped the simulation once, and retrieved a simple vehicle count, confirming a working communication pipeline. Figure 3.5 illustrates the demo used during testing.

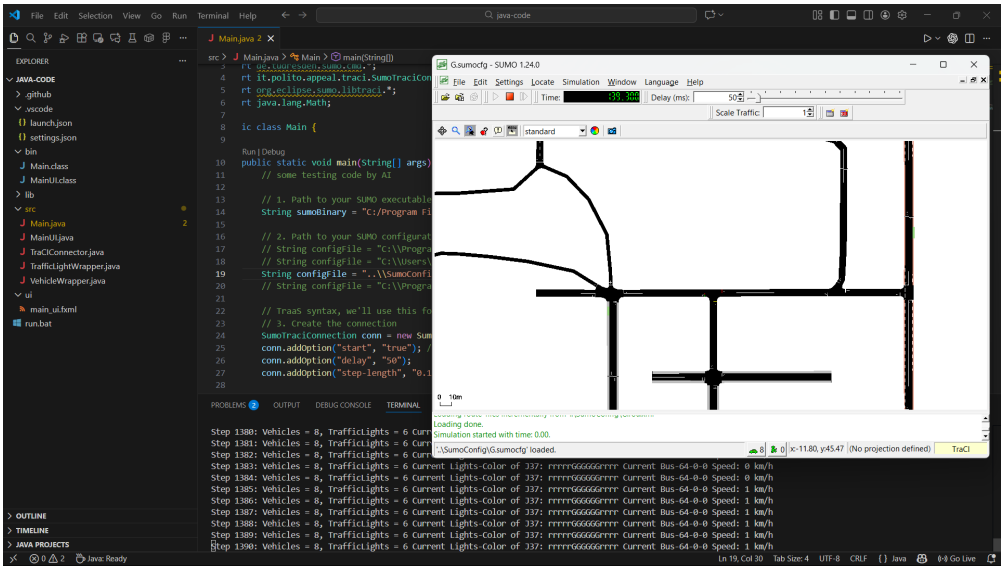

Figure 3.5: Demo of the Java–SUMO TraCI connection during Milestone 1.

This confirms that the communication layer works and can be expanded with more detailed functions in the next milestones.

3.6 Use Case Diagram

Figure 3.6 presents the simplified use case diagram created for Milestone 1.

Since the full system logic is not yet implemented, the current use cases focus on core interactions: running the SUMO simulation, stepping the simulation manually, obtaining simple vehicle counts, and displaying results either in the GUI or console.

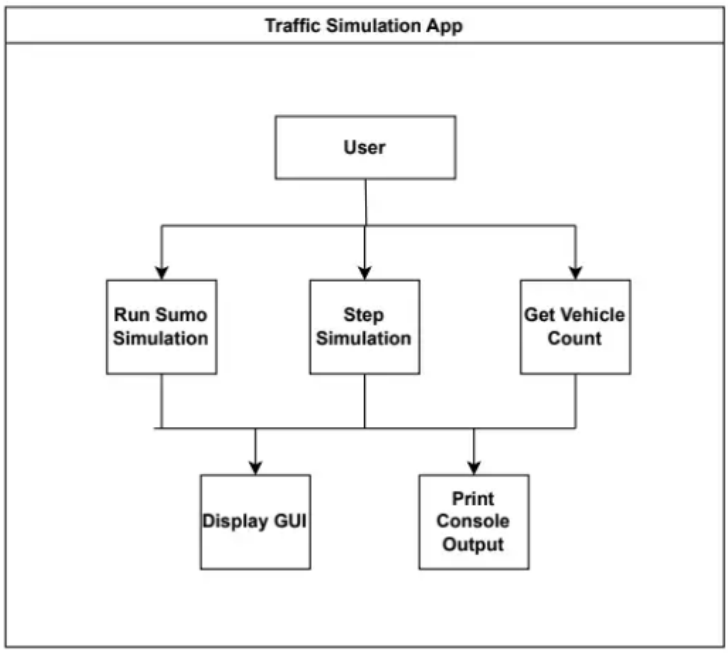

Figure 3.6: Use case diagram for the early-stage traffic simulation application.

These use cases provide a clear overview of what functionality is available at this stage and what interactions will be expanded in Milestone 2.

Conclusion

In this milestone, we successfully designed the complete graphical user interface of the application using JavaFX and Scene Builder. The GUI now provides all required structural components, including the simulation toolbar, configuration tabs, map display area, metrics panel, and status bar.

Although SUMO backend integration is not yet implemented, the interface is fully organized, responsive, and ready for the functional features that will be added in future milestones. The work completed here forms the foundation for real-time simulation control and visualization in the next development stages.