Object-Oriented Programming in Java

Project Description

Prof. Dr.-Eng. Ghadi Mahmoudi

Winter 2025-2026

1. Introduction

The examination of this module is in form of a group project. The resulting program must be fully functional and well-documented. Each team should come together and work autonomously and is responsible for distributing tasks fair and equal among all team members. The task must be solved by the team. Help from external sources or any fraudulent activities, such as copying source code from various sources without proper reference, will result in, at a minimum, disqualification or legal consequences. The amount of original work must significantly exceed the amount of code copied from external sources, even if properly referenced. This does not apply for the use of external libraries for extending own program features in a reasonable ratio of own code vs. external code. Any suspicious activity of that kind will be reported to the faculty examination board for further investigation. All further conditions of the examination regulation apply.

The following constraints apply to the module examination:

Start Date (Topic announcement)
21.10.2025

Milestone Deadlines
27.11.2025 and 14.12.2025 (Submission to the eLearning system)

Presentation of milestones
Directly after the submission deadline, to the time of your exercise group.

Submission Deadline
18.01.2026, 17:59 via the eLearning system

Final presentation
19.01.2026 – 21.01.2026, to the time of your exercise group, or to the given appointment.

Team size
4 to 5 team members

Topic
The presented project from Section 3

Submission Details
The following things are required for a successful submission

1. A working program including all necessary libraries and the source code and the (fully functional and configured) zipped project (e.g. eclipse, IntelliJ) including all necessary files e.g. project files for importing. The source code must be commented as mentioned in the clean code lecture. Ensure that your program runs on JDK 17 or higher.

   Note that a dysfunctional program, i.e., one that does not start or execute properly under the given requirements, will receive 0 points.

2. Documentation as separate document (PDF file) including technical details of the program, user documentation and declaration of the individual team member performance. A milestone document has to be submitted on the given deadline via the eLearning system. Preferably, the milestone document is one developing document over the whole project run time which is finally submitted as project documentation. An example is given in the eLearning system.

3. The declaration of authorship (German: “Ehrenwörtliche Erklärung”) must be signed by each team member and added to the submitted documentation. You find an example text below.

4. Submit the software and required documents as archive in *.zip/*.tar.gz format via the moodle platform in the specific section (will be announced). The archive must contain the full source code, a runnable compiled counterpart, the documentation, the milestone documents, and the signed/scanned declaration of authorship.

5. Mark the implemented parts of the requirements in the documentation. Only explicitly declared features can be considered for grading.

Presentation
After the submission each individual (person) has to present the team solution and the participation to it as part of a group presentation. The general conditions for the presentations will be announced during the semester.

Milestone presentations
All groups have to give a short presentation (around 10 minutes) about each individual milestone during the exercises. Additional details will be announced during the semester.

Use of external tools
You must be prepared to explain every detail of your code during the presentation if asked. In case you use external tools, e.g., AI-based tools, you have to (1) declare where and what you used it for, (2) must be able to to explain in detail what happens during execution of code snippets and (3) give the exact prompt that you used in case you use AI generated content.

Points may be deducted if the balance between original work and external assistance is deemed inadequate.

Additional Software and Frameworks
You are free to use any available IDE and GUI framework you like, as long as it fulfills the requirements. In the lecture, however, MS Windows-based Eclipse is used as the IDE, and Java Swing is used as the GUI framework. Thus, if you decide to deviate from the discussed software, there will be no support during the project regarding this software.

Hints
It is also possible to implement only parts of the given requirements. In case you do that, please make sure that your program is running even in that case. If parts of the specified program are missing, points will be deducted.

It is necessary to successfully submit every required item by the closing date. In addition, successful participation in the milestone presentations and in the final presentation is necessary.

Declaration of Authorship

Example for a declaration of authorship ("Ehrenwörtliche Erklärung"):

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.

---	----------------------------
First and last name	City, date and signature

2. Grading Criteria

The grading of the projects is based on the given lectures and the individual autonomous learning of the topic. The module is passed with a total of 50 points (50%). See the following tables for details of the grading structure (Table 1) and the necessary requirements (Table 2).

For the submission, please provide exemplary details on where you fulfilled individual requirement including class and line number, e.g., “Threads in class HelloWorld.java:123”.

Table 1: Points vs. Grades

Grade	5,0	4,0	3,7	3,3	3,0	2,7	2,3	2,0	1,7	1,3	1,0
Points	<50	50	55	60	65	70	75	80	85	90	95^{+}

Table 2: Requirements

Feature	**Points**	**Description**
Runnable Program	30	The running program is covering the described basic functionalities as described in Section 3.
Milestones	20	Milestones are submitted and presented as required. The requirements of the milestones are specified in Section 3.
Object-Orientation	10	The program was developed using adequate object orientation (class design, inheritance, abstract methods/classes, interfaces, methods overriding, etc.).
Collections	5	Information will be stored in suitable Java Collections or appropriate alternative data structures. Not using appropriate collections will require justification.
Error Handling	5	The program contains adequate error handling. At least one own exception (extending any type of Java Exception/Throwable) must be present.
Streams and Files	5	The program works with streams and files. At least one reading and/or writing file access has to be made, apart from the basic requirement.
Threads	5	The program contains at least two threads (of which one is “main”).
Clean Code	10	The program is developed using the clean code standard(s) as presented in the lecture.
Documentation	10	The documentation inside and outside the code is present as described in following sections. Appropriate logging is provided using java.util.Logging or comparable.
Total	**100**	

3. Project – Real-Time Traffic Simulation

3.1. Real-Time Traffic Simulation with Java - Description

This task involves the creation of a real-time traffic simulation platform using the SUMO mobility engine, controlled via the TraaS Java API. It features a GUI that visualizes the road network, vehicles, and traffic lights, while enabling user interaction and adaptive control.

The system connects to a running SUMO instance and steps through the simulation in real time. Vehicles are rendered on a 2D map, with optional 3D views using JavaFX's 3D capabilities. Traffic lights are displayed with current phase indicators and can be manually adjusted or tuned via the GUI.

Users can inject vehicles on specific edges, apply stress tests to simulate congestion, and monitor key metrics such as average speed, edge density, and travel time. The platform supports grouping and vehicles filtering (e.g. by color, speed, or location) and includes a wrapper around TraaS to provide object-oriented access to simulation entities.

Simulation data can be exported as CSV or PDF reports for analysis. The system is designed to support experimentation with adaptive traffic control strategies, making it suitable for research, teaching, and urban mobility prototyping.

3.2. Real-Time Traffic Simulation with Java - Challenge

Build a Java-based application that connects to the SUMO traffic simulator in real time, visualizes traffic flow, enables user interaction, and supports adaptive traffic control. This project combines simulation, visualization, and data analysis to create a dynamic environment for experimenting with urban mobility. The application features can be classified to "core features" and "additional recommended features".

3.2.1. Core Features

Your solution must support the following <u>"core features"</u>:

1. **Live SUMO Integration**

- Use the TraaS API to connect to a running SUMO simulation.
- Step through the simulation in real time.
- Inject vehicles, read telemetry, and control traffic lights programmatically.

2. **Interactive Map Visualization**

- Render the road network.
- Display moving vehicles with color-coded icons.
- Show traffic lights with current phase indicators.

- Support zooming, panning, and camera rotation.
- Show subsets of vehicles according to their properties (filtered by e.g. color, speed, or location)

3. **Vehicle Injection & Control**

- Allow users to create vehicles on specific edges via GUI.
- Support batch injection for stress testing.
- Enable control over vehicle parameters (speed, color, route).

4. **Traffic Light Management**

- Display traffic light states and phase durations.
- Enable manual phase switching via GUI.
- Allow users to adjust phase durations and observe effects on traffic flow.

5. **Statistics & Analytics**

- Track metrics such as:
    - Average speed
    - Vehicle density per edge
    - Congestion hotspots
    - Travel time distribution
- Display charts and summaries in real time.

9. **Exportable Reports**

- Save simulation statistics to **CSV** for external analysis.
- Generate **PDF summaries** with charts, metrics, and timestamps.
- Include filters (e.g. only red cars, only congested edges) in exports.

3.2.2. Additional Recommended Features

The solution should also support <u>additional recommended features</u>:

1. **Stress Testing Tools**

- Simulate heavy traffic on selected edges.
- Observe system behavior under load.
- Compare static vs adaptive traffic light strategies.

2. **Adaptive Traffic Control**
    - Allow users to experiment with traffic light timing to improve flow.
    - Provide feedback on performance metrics after adjustments.
    - Optionally integrate simple rule-based adaptation.
3. **3D Rendering (Optional)**
    - Use JavaFX 3D to render intersections or vehicles from different angles.
    - Animate camera movement or simulate drone views.

3.3. Object-Oriented Design Challenge

TraaS uses static methods and procedural patterns. Your task is to build a clean, reusable, and extensible **object-oriented wrapper** around it. This wrapper should:

- Encapsulate vehicle, edge, and traffic light logic
- Support querying and control via instance methods
- Enable grouping, filtering, and event handling

Important Note

You have a high degree of freedom in how you choose to represent the data. While creativity is encouraged, adherence to the specified requirements is most important. While considering the optional JavaFX's 3D-rendering in Java, you are free to choose whatever GUI framework you use.

4. Documentation and Non-Functional Features

The requirements for the documentation are

- JavaDoc must be applied to each public method within a class, except for getter and setter methods, unless these methods perform additional functions requiring documentation.
- Other source code documentation/comments are applied as necessary.
- An appropriate logging of information is provided using the java.util.Logging or a comparable library. Logging must at least be applied to each exceptional program flow, e.g., when throwing exceptions or handling error code but it is recommended to also log debug information, e.g., when handling data or reading in files etc. Logging must replace calls to the systems output stream completely, if used for debugging purposes.
- Documentation outside the code is submitted as PDF file containing user- and technical documentation having adequate descriptions, graphics, diagrams etc. The following elements must be included:
    - User handbook (how to use the program), e.g., create screenhots and showcase your program flow to a possible (non-technical) user
    - Technical description, e.g., create a description that helps technical people to understand your project.
    - UML diagram(s), at least a class diagram is helpful to understand the interconnection of your classes
    - Work distribution among team members, preferably as table
    - Milestone documents

  All documents may be aggregated into one PDF file, e.g., organized as individual chapters, and may also be maintained throughout the semester and submitted as milestone document, i.e., living documentation.
- Milestone documents containing the current state of the project and the participation of the team members. It must be submitted separately on the announced deadlines via the eLearning system as one document in PDF format. Submitting the milestones is mandatory. Submission is closed after the deadline, and no further submission is possible. Milestone presentations take place directly after the submission deadline, to the time of your exercise group.