AP-M2025: Assignment 3

Fleet Highway Simulator using Multithreading and GUI with Race Condition Handling

Due Date: Nov 30, 2025 (23:59hrs)

Total points: 75

1. Aim and Objectives

The objective of this assignment is to extend your previous work in Assignments 1 and 2 into a unified graphical and concurrent simulation system. Students are required to design and implement a Java-based application that:

- Demonstrates multithreaded execution through a simulated highway scenario involving multiple vehicles in motion.
- Provides a graphical user interface (GUI) for controlling and observing the simulation, devel oped using AWT and/or Swing.
- Illustrates a race condition arising from unsynchronised access to a shared resource and sub sequently resolves it using appropriate synchronisation techniques.

This assignment integrates core concepts of object-oriented programming, multithreading, basic GUI development and correctness in concurrent programming.

2. Background

In many real-world applications, multiple entities operate concurrently while sharing common re sources. Improper coordination among such concurrent components may lead to inconsistent states, known as race conditions. This assignment simulates a highway system in which multiple vehicles operate simultaneously while contributing to a shared highway usage metric.

Students must first implement the system without proper synchronisation to intentionally reveal erroneous behaviour caused by concurrent updates to a shared resource. Subsequently, they must enhance the design to ensure correctness using suitable synchronisation constructs.

3. Problem Description

You are required to design a Fleet Highway Simulator in Java where multiple vehicles travel concurrently on a simulated highway. Each vehicle operates in its own thread and periodically updates (i) its own mileage and fuel level, and (ii) a shared counter representing the total kilometres travelled by all vehicles collectively (referred to as the *Highway Distance Counter*).

The system must be fully operated using a graphical interface, which displays relevant information to the user and allows them to control the simulation.

The application must satisfy the following combined requirements:

R1. Multithreaded Simulation: Each vehicle shall run in its own thread, updating its state at regular intervals (approximately once per second).

R2. Shared Highway Counter: All vehicle threads must update a common highway dis tance counter. This update must initially be implemented without synchronisation, thereby causing a race condition.

R3. Race Condition Demonstration and Fix:

- Students must demonstrate the incorrect behaviour resulting from the race condi tion (e.g., inconsistent totals).
- The race condition must be resolved using either synchronized or a ReentrantLock.

R4. Graphical User Interface (GUI): The system must be controlled and visualised through a GUI built using AWT and/or Swing. The GUI must:

- Provide controls to start, pause, resume and stop the simulation.
- Display the state of the vehicles and the shared counter in a clear and user-friendly manner.
- Present observable behaviour of the race condition and its corrected outcome.

R5. Integration of Previous Work (Optional): Students may reuse their class design from previous assignments or implement a simplified vehicle model if preferred.

4. System Requirements

4.1 Vehicle Model

A suitable class design must be implemented for representing vehicles and their attributes. Students are encouraged to reuse or adapt designs from Assignments 1 and 2.

Each vehicle is expected to maintain, at minimum:

- A unique identifier
- Mileage travelled
- Fuel remaining
- Operational status (e.g., Running, Paused, Out-of-Fuel)

4.2 Simulation Behaviour

Each vehicle thread shall:

- Increment its mileage by approximately 1 km per second.
- Decrease its fuel level appropriately.
- Increment the shared *Highway Distance Counter*.

• Pause if the fuel level reaches zero, until re-fuelled via the GUI.

Control of the simulation must be facilitated solely through the GUI, without requiring console based I/O.

4.3 Race Condition Guidance

The following structured guidance is provided to support the implementation and correction of the race condition.

Step 1: Implement Unsynchronised Access

A shared integer variable, such as:

```
public static int highwayDistance = 0;
```

shall be incremented by each vehicle thread without any synchronisation. Students should execute the simulation with multiple vehicles to observe inconsistent behaviour (e.g., total mileage does not match the sum of individual increments).

Step 2: Record Incorrect Behaviour

Students must capture evidence of the incorrect state through GUI screenshots or logs. Step 3: Apply Synchronisation

Modify the shared-resource update using either:

• synchronized (method or block), or

• ReentrantLock

Step 4: Validate Correct Behaviour

Execute the simulation again and verify, via the GUI, that the shared counter remains consistent. Note: Kindly make a note of the following important observations while attempting the assignment:

• You must simulate atleast 3 vehicles which must be visible in realtime as labels. Animation is not essential.

• Simulate vehicles being refuelled, after which they should automatically resume.

5. System Architecture Diagram

GUI (AWT and/or Swing)

User Controls and Display

Vehicle Thread 1 Vehicle Thread 2 Vehicle Thread 3

Shared Highway
Counter

Figure 1: High-level architecture showing the interaction between the GUI, vehicle threads and the shared counter.

6. Submission Guidelines

Each submission must include the following:

S1. Complete source code of the implemented system.

S2. A ReadMe document containing:

- Steps to compile, run and test the program.
- Overview of the design and GUI layout.
- Explanation of how the simulation threads were controlled via the GUI.
- Description of the race condition and synchronisation fix, accompanied by required screen shots/logs (before and after correction).
- A brief note on GUI thread-safety considerations, such as the importance of the Event Dispatch Thread (if Swing is used).

S3. Screenshots of the GUI during execution, demonstrating both incorrect and correct behaviour.

7. Evaluation Scheme (75)

Evaluation Component	Marks
Multithreaded simulation and GUI-based control	30
Race condition demonstration and fix with synchronisa tion	25
Integration with Assignment 1/2 model or equivalent simplified model	10
Documentation/ReadMe	10
Total	75