ReadMe Document

1. Steps to Compile, Run, and Test the Program

Using Maven:

```
mvn compile
mvn exec:java
```

Ensure that `pom.xml` includes the correct main class:

```
<exec.mainClass>fleet.SimulationGUI</exec.mainClass>
```

Manual Compilation (Without Maven):

```
javac -d out src/**/*.java
java -cp out fleet.SimulationGUI
```

2. Overview of the Design and GUI Layout

Design Summary:

- Vehicles (`Car`, `Bus`, `Truck`) implement a common `Vehicle` interface.
- Each vehicle runs in its own thread.
- All threads contribute to a shared highway distance counter.
- GUI is built using Java Swing and interacts with the simulation in real-time.

GUI Layout:

- **Top Panel:**
  - Buttons to `Start (unsynchronised)`, `Start (synchronised)`, `Pause`, `Resume`, `Stop`, `Reset`
- **Main Table View:**

    - Displays each vehicle's `ID`, `Mileage`, `Fuel`, and `Status`
- **Bottom Panel:**
    - Options to refuel individual vehicles, pause/resume them individually

3. Simulation Thread Control via GUI

- **Start Buttons:** Spawn and manage threads for each vehicle.
- **Pause / Resume:** Global control of all threads using shared flags and locks.
- **Pause / Resume Vehicle:** Per-vehicle control via thread-safe mechanisms.
- Threads use a `pauseLock.wait()/notifyAll()` pattern to handle pausing and resuming without race conditions.

4. Race Condition and Synchronisation Fix

Problem (Unsynchronised Access):

Initially, the shared variable `Simulation.highwayDistance` was updated by all threads without synchronization:

```
highwayDistance += delta;
```

This caused lost updates and mismatch between the sum of all vehicle mileages and the global highway distance.

Fix (Synchronized Access):

Updated to ensure synchronized access to the shared resource:

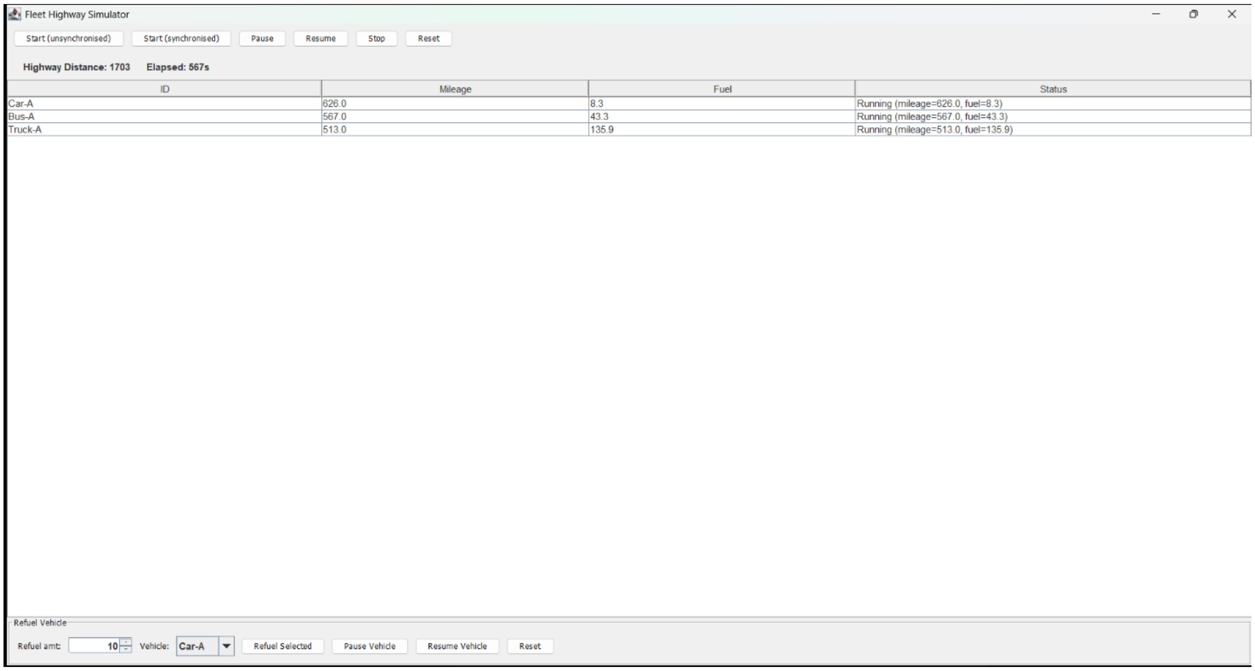
```
synchronized (highwayLock) {
    highwayDistance += delta;
}
```

This ensures only one thread modifies the shared counter at a time.

📸*Screenshots:*

Screenshot 1: Before Synchronization (Unsynchronized Mode)

Total mileage does NOT match the highway distance, confirming race condition.



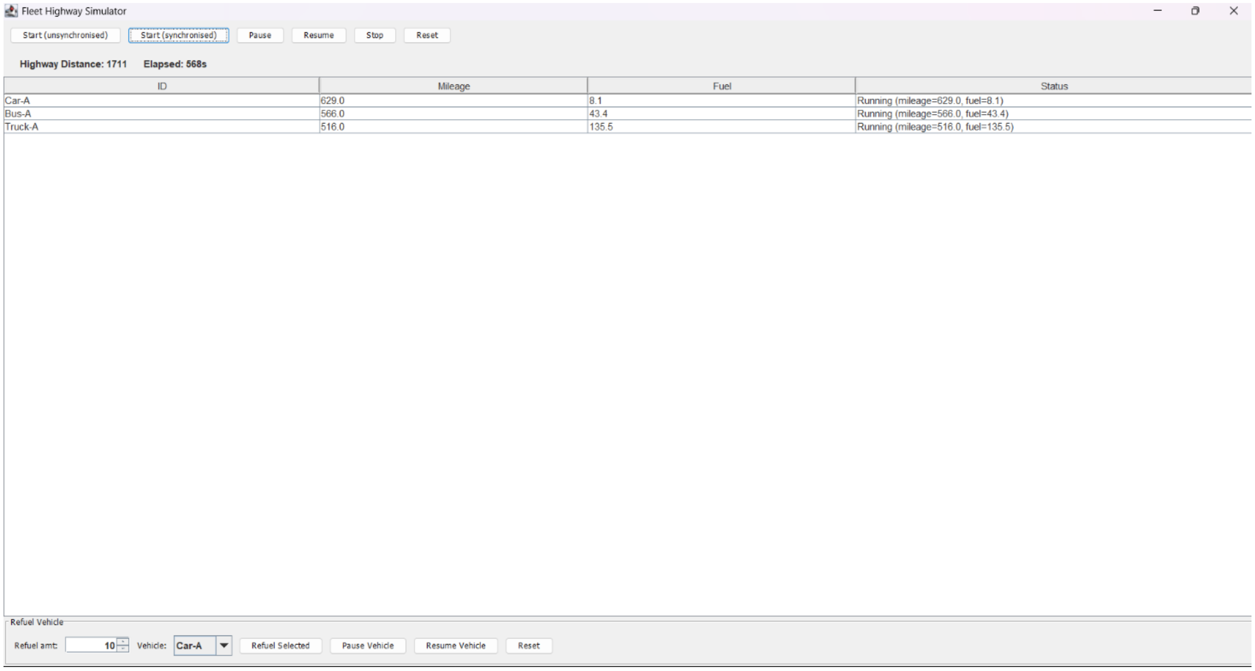
Highway Distance: 1703 Elapsed: 567s

ID	Mileage	Fuel	Status
Car-A	626.0	8.3	Running (mileage=626.0, fuel=8.3)
Bus-A	567.0	43.3	Running (mileage=567.0, fuel=43.3)
Truck-A	513.0	135.9	Running (mileage=513.0, fuel=135.9)

Refuel amt: 10 Vehicle: Car-A

Screenshot 2: After Synchronization (Synchronized Mode)

Sum of vehicle mileages exactly equals the highway distance counter.



Highway Distance: 1711 Elapsed: 568s

ID	Mileage	Fuel	Status
Car-A	629.0	8.1	Running (mileage=629.0, fuel=8.1)
Bus-A	566.0	43.4	Running (mileage=566.0, fuel=43.4)
Truck-A	516.0	135.5	Running (mileage=516.0, fuel=135.5)

Refuel amt: 10 Vehicle: Car-A

5. GUI Thread-Safety Considerations

- All Swing component updates are done via the **Event Dispatch Thread (EDT)** using `SwingUtilities.invokeLater()`.
- A `javax.swing.Timer` periodically pulls state from the simulation and updates the GUI.
- Avoids unsafe direct cross-thread GUI updates, preventing inconsistencies and potential deadlocks.

```
SwingUtilities.invokeLater(() -> {
    // update table and labels safely
});
```