FerrySimulation Program Report

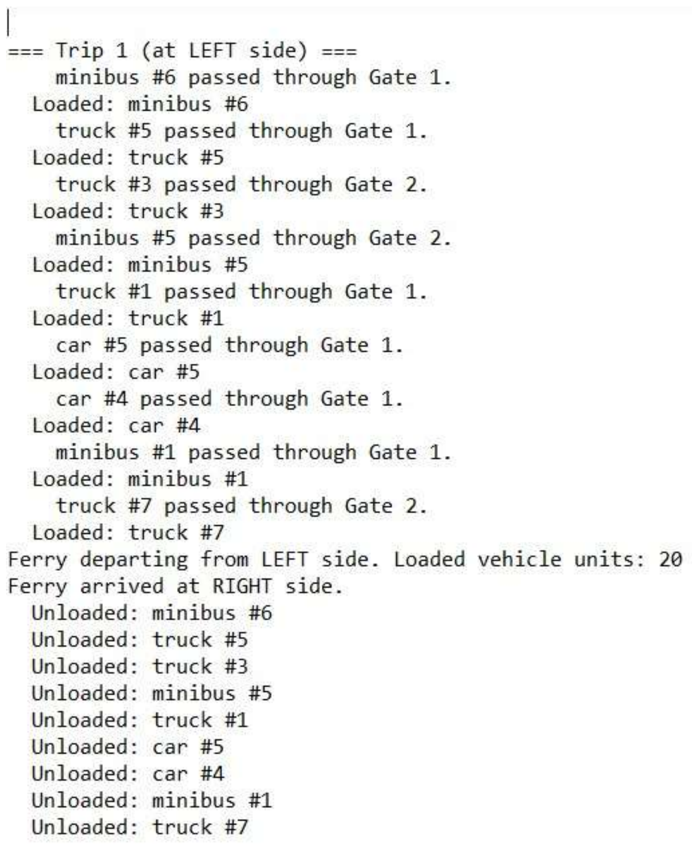
Scenario Overview

This program simulates vehicles (cars, minibuses, trucks) being transported by a ferry between two shores (LEFT and RIGHT) using multithreading in Java. Vehicles start randomly from either shore and wait in queues to board the ferry. The ferry has a limited capacity, measured in units according to vehicle size (cars = 1, minibuses = 2, trucks = 3). The ferry loads vehicles from the shore it is currently at, crosses to the opposite shore, unloads vehicles, and the vehicles continue traveling until they return to their original shore. The simulation runs until all vehicles have returned home.

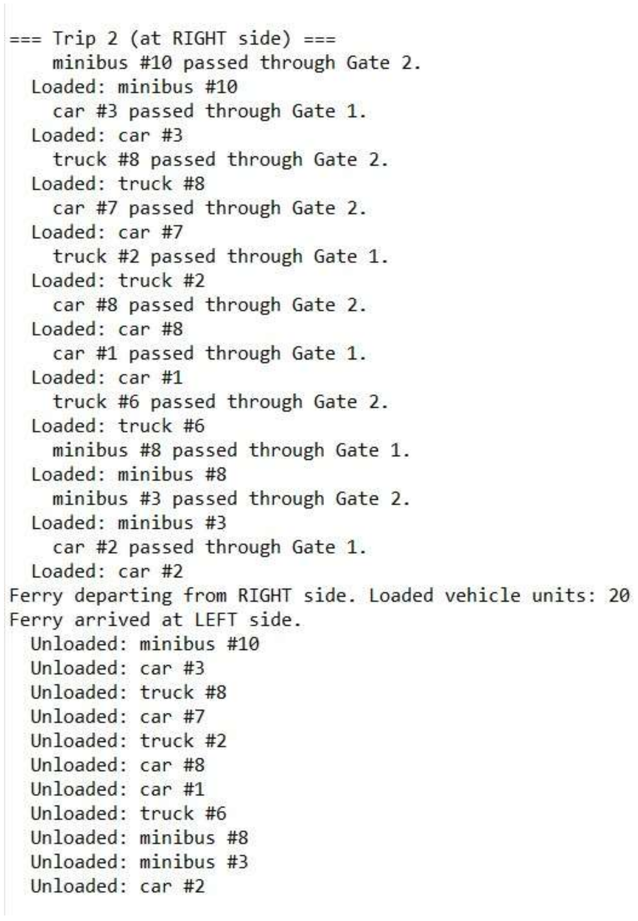
Key Concepts and Structures Used

- **enum Side**: Represents the two shores — LEFT and RIGHT.
- **enum VehicleType**: Defines vehicle types and their size units on the ferry.
- **Vehicle class (implements Runnable)**: Each vehicle runs on its own thread, simulating passing through toll booths and waiting in queues.
- **BlockingQueue**: Thread-safe queues to hold vehicles waiting on each shore.
- **TollBooth class**: Simulates a toll booth that vehicles must pass before boarding.
- **Ferry class**:
  - Has a capacity of 20 units.
  - Manages loading and unloading vehicles based on capacity.
  - Keeps track of its current shore position.
- **ExecutorService (Thread Pool)**: Manages concurrent execution of vehicle threads.
- **Multithreading**: Vehicles operate concurrently, passing through toll booths and joining queues.
- **Randomization**: Vehicle starting shore and choice of toll booth are randomized.

Sample Output



```
=== Trip 1 (at LEFT side) ===
    minibus #6 passed through Gate 1.
  Loaded: minibus #6
    truck #5 passed through Gate 1.
  Loaded: truck #5
    truck #3 passed through Gate 2.
  Loaded: truck #3
    minibus #5 passed through Gate 2.
  Loaded: minibus #5
    truck #1 passed through Gate 1.
  Loaded: truck #1
    car #5 passed through Gate 1.
  Loaded: car #5
    car #4 passed through Gate 1.
  Loaded: car #4
    minibus #1 passed through Gate 1.
  Loaded: minibus #1
    truck #7 passed through Gate 2.
  Loaded: truck #7
Ferry departing from LEFT side. Loaded vehicle units: 20
Ferry arrived at RIGHT side.
  Unloaded: minibus #6
  Unloaded: truck #5
  Unloaded: truck #3
  Unloaded: minibus #5
  Unloaded: truck #1
  Unloaded: car #5
  Unloaded: car #4
  Unloaded: minibus #1
  Unloaded: truck #7
```



```
=== Trip 2 (at RIGHT side) ===
    minibus #10 passed through Gate 2.
  Loaded: minibus #10
    car #3 passed through Gate 1.
  Loaded: car #3
    truck #8 passed through Gate 2.
  Loaded: truck #8
    car #7 passed through Gate 2.
  Loaded: car #7
    truck #2 passed through Gate 1.
  Loaded: truck #2
    car #8 passed through Gate 2.
  Loaded: car #8
    car #1 passed through Gate 1.
  Loaded: car #1
    truck #6 passed through Gate 2.
  Loaded: truck #6
    minibus #8 passed through Gate 1.
  Loaded: minibus #8
    minibus #3 passed through Gate 2.
  Loaded: minibus #3
    car #2 passed through Gate 1.
  Loaded: car #2
Ferry departing from RIGHT side. Loaded vehicle units: 20
Ferry arrived at LEFT side.
  Unloaded: minibus #10
  Unloaded: car #3
  Unloaded: truck #8
  Unloaded: car #7
  Unloaded: truck #2
  Unloaded: car #8
  Unloaded: car #1
  Unloaded: truck #6
  Unloaded: minibus #8
  Unloaded: minibus #3
  Unloaded: car #2
```

Summary

The program successfully simulates a ferry transport system using **multithreading** in Java. It ensures safe concurrent access to vehicle queues using BlockingQueue and manages thread execution with ExecutorService. Each vehicle (car, truck, minibus) is represented by its own thread, simulating independent arrivals at ferry gates. The ferry itself operates in a separate thread, enforcing capacity constraints based on vehicle size and loading vehicles from the appropriate side. Synchronization mechanisms ensure that shared resources (like queues and ferry states) are accessed safely, avoiding race conditions. Vehicles continuously move between shores and return home, accurately modeling a realistic and concurrent transport scenario.