Mars Rover Simulation

1. Project Overview:

The Mars Rover Simulation is a Java-based project that simulates a rover navigating a grid-based terrain on Mars. The rover can move forward, turn left, or turn right while avoiding obstacles and staying within grid boundaries. The project supports both CLI (Command Line Interface) and HTML-based frontend to run simulations.

Key Features:

* Grid-based navigation
* Obstacle detection and avoidance
* Command execution using behavioral patterns
* Logging and status reporting
* CLI and Web (Spring Boot + HTML) interface

1. Architecture & Design Principles

2.1 Object-Oriented Programming

* Encapsulation: Each class manages its own data and exposes controlled access. (e.g., Rover class encapsulates position and direction).
* Inheritance: Command classes (MoveCommand, LeftCommand, RightCommand) inherit from a common Command interface.
* Polymorphism: Commands are executed polymorphically on the Rover using the Command interface.

2.2 Design Patterns

1. Behavioral Pattern – Command Pattern
   * Commands (M, L, R) are encapsulated as objects.
   * Provides flexibility in executing, extending, or scheduling commands.

Command cmd = CommandFactory.getCommand("M");

cmd.execute(rover);

1. Structural Pattern – Composite Pattern
   * Grid and obstacles represented using a composite structure (GridComposite and Obstacle).
   * This allows scalable and maintainable representation of obstacles.

GridComposite grid = new GridComposite(10, 10);

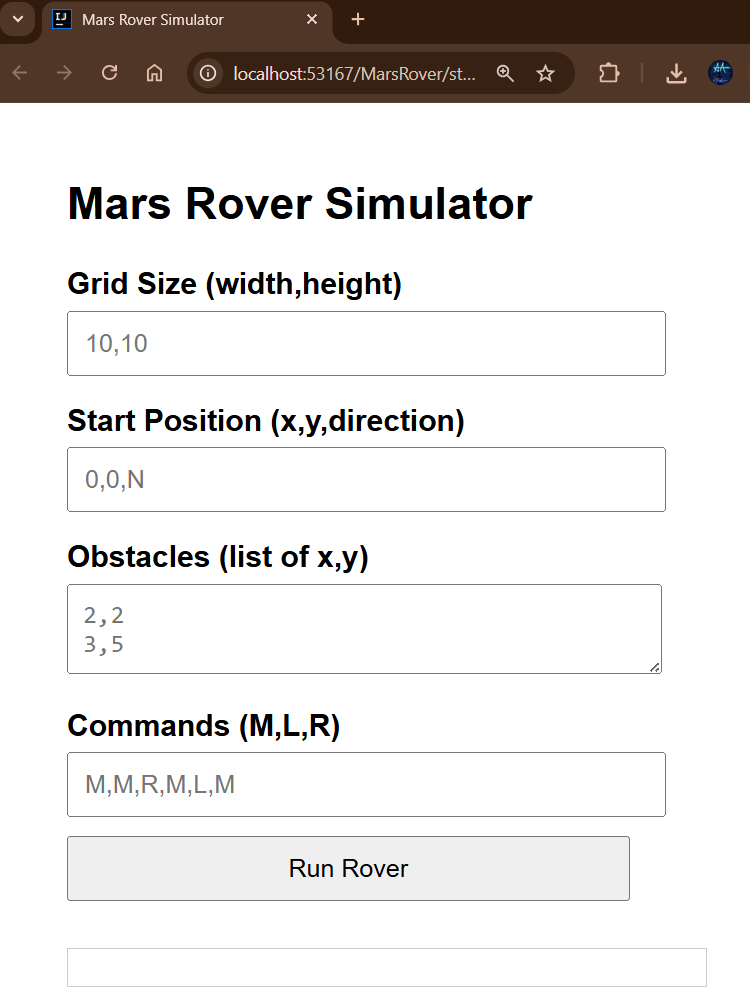
grid.add(new Obstacle(new Position(1,2)));

1. SOLID Principles
   * Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion.
2. Key Components

|  |  |
| --- | --- |
| Component | Responsibility |
| Rover | Maintains position and direction; executes moves/turns; detects obstacles. |
| GridComposite | Represents the grid and manages obstacles; checks bounds. |
| Obstacle | Leaf component representing an obstacle. |
| Command Interface | Abstract command pattern for rover actions. |
| MoveCommand, LeftCommand, RightCommand | Concrete commands for moving and turning the rover. |
| CommandFactory | Factory to generate command objects from input. |
| RoverController | Spring Boot REST controller for HTML frontend. |
| MarsRoverCLI | CLI interface for command-line simulation. |

1. User Interfaces

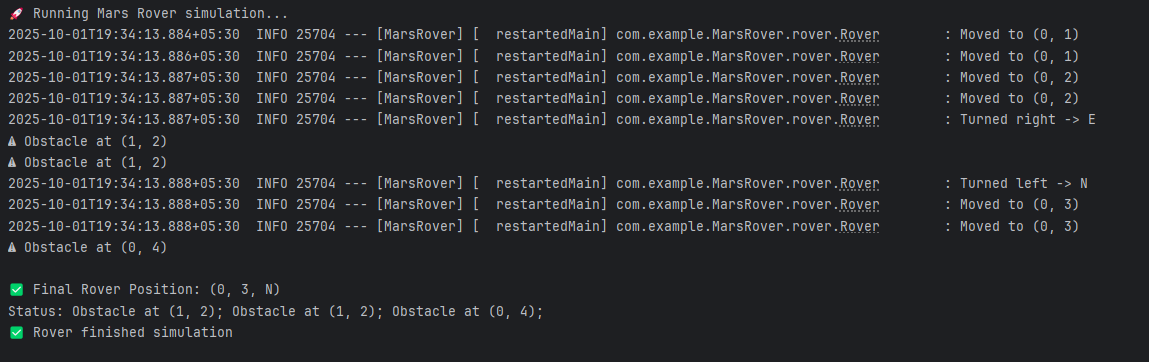
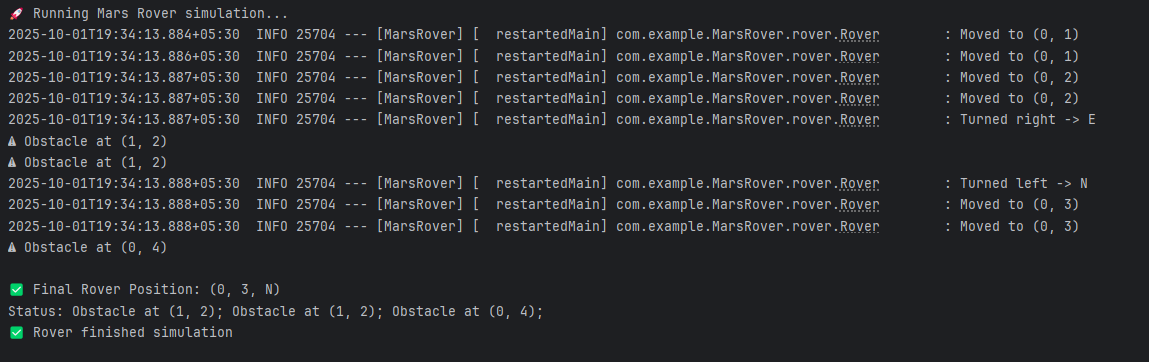
4.1 HTML Frontend



* Input fields: Grid Size, Start Position, Obstacles, Commands.
* Submit button calls /commands endpoint.
* Displays final position and status.

4.2 CLI

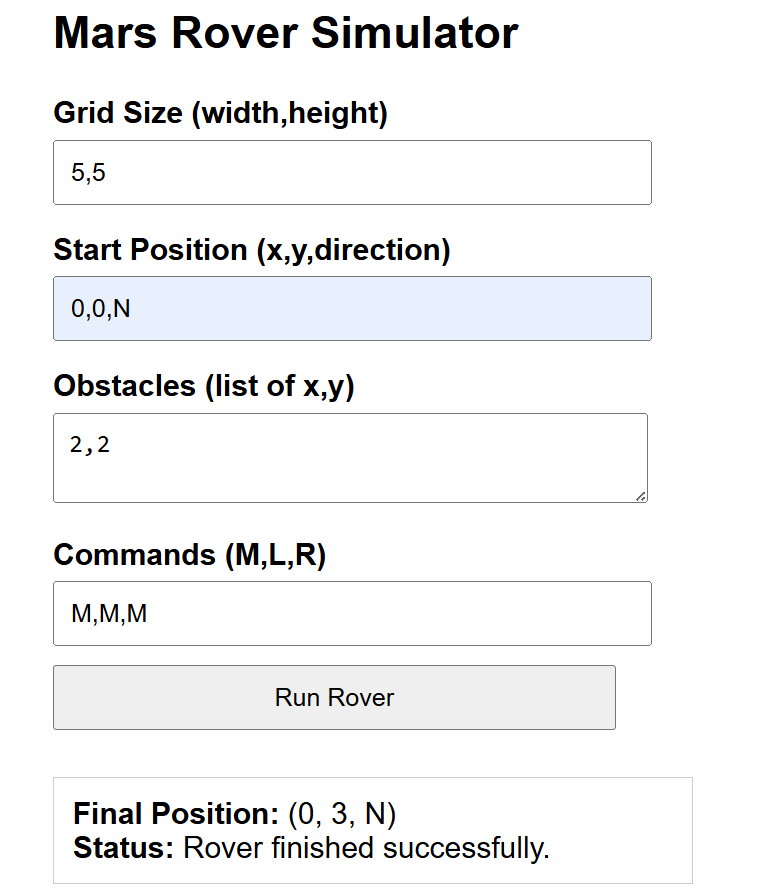
* Default values for grid, obstacles, start, and commands.
* Step-by-step moves and obstacles shown in console.

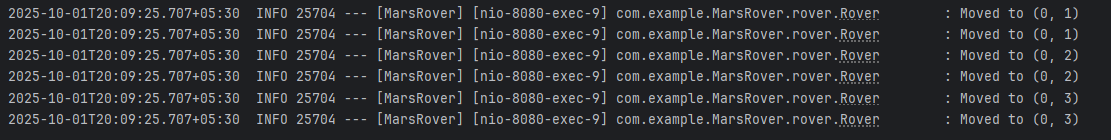


1. How It Works – Scenarios

Scenario 1: Simple Movement

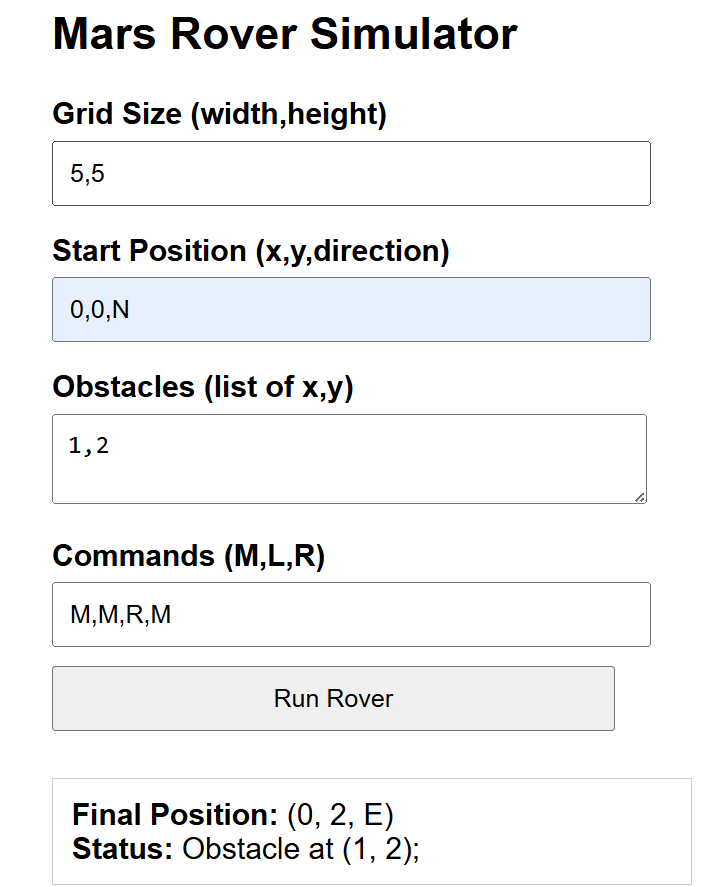
* Input: Grid 5x5, start (0,0,N), commands M,M,M, obstacles (2,2)
* Output: Final (0,3,N), Status: Success.

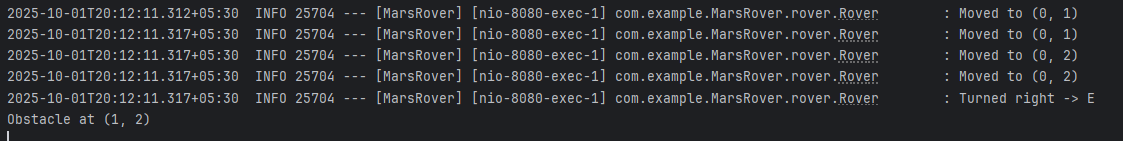




Scenario 2: Obstacle Encounter

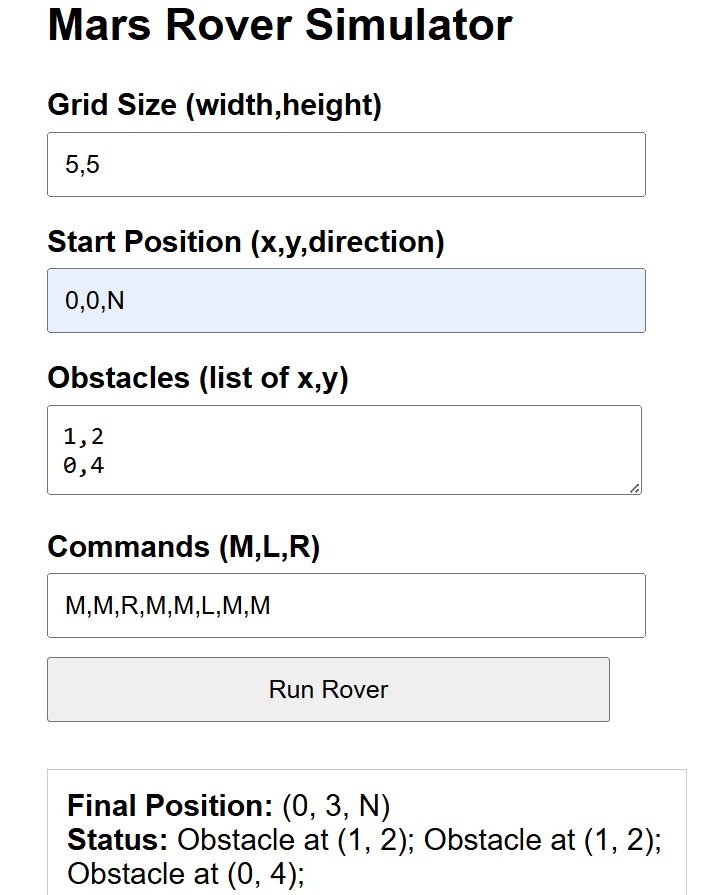
* Input: Grid 5x5, start (0,0,N), obstacles (1,2), commands M,M,R,M.
* Output: Final (0,2,E), Status: Obstacle at (1,2).

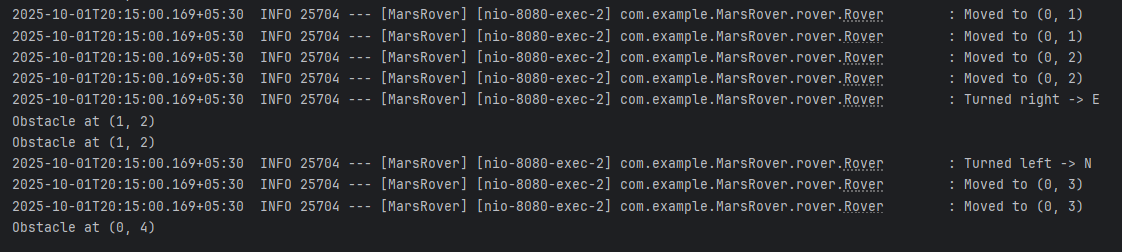




Scenario 3: Multiple Obstacles and Command Sequence

* Input: Grid 5x5, start (0,0,N), obstacles (1,2), (0,4), commands M,M,R,M,M,L,M,M.
* Output: Final Position: (0,3,N), Status: Obstacle at (1,2); Obstacle at (0,4).





1. Logging and Exception Handling

• Uses SLF4J Logger in Rover class to track movements, turns, and obstacles.

• Exceptions:

ObstacleDetectedException: logs obstacle hit, skips move.

OutOfBoundsException: logs boundary violation, skips move.

• Status messages are returned to the frontend or CLI.

1. Java Backend

 **Spring Boot** application.

 REST controller (RoverController) handles command requests from HTML frontend.

 Rover and Grid logic are **decoupled** from frontend.

 Fully extendable for future commands or UI features.

1. Summary

* Behavioral Pattern: Commands encapsulated.
* Structural Pattern: Composite grid and obstacles.
* OOP: Encapsulation, inheritance, polymorphism.
* SOLID & Best Practices: SRP, Open/Closed, Liskov, Dependency Inversion.
* Functionality: Works on CLI and HTML frontend.
* Logging & Exceptions: Steps logged; obstacles and out-of-bounds handled gracefully.