Mars Rover Programming Exercise

import java.util.Arrays;

public class RoverSimulation {

private static final int GRID\_SIZE = 10;

// Directions

enum Direction { N, E, S, W }

// Rover class

static class Rover {

int x;

int y;

Direction direction;

Rover(int x, int y, Direction direction) {

this.x = x;

this.y = y;

this.direction = direction;

}

}

// Grid class

static class Grid {

boolean[][] obstacles = new boolean[GRID\_SIZE][GRID\_SIZE];

Grid() {

for (boolean[] row : obstacles) {

Arrays.fill(row, false);

}

}

}

// Function prototypes

static void initializeRover(Rover rover, int x, int y, Direction direction) {

rover.x = x;

rover.y = y;

rover.direction = direction;

}

static void initializeGrid(Grid grid) {

for (boolean[] row : grid.obstacles) {

Arrays.fill(row, false);

}

}

static void moveRover(Rover rover, Grid grid) {

// Implement movement logic based on rover's direction and check for obstacles

}

static void turnLeft(Rover rover) {

// Implement turning left logic

}

static void turnRight(Rover rover) {

// Implement turning right logic

}

static void reportStatus(Rover rover, Grid grid) {

// Implement status reporting logic

}

static boolean isObstacle(Grid grid, int x, int y) {

return grid.obstacles[x][y];

}

// Command functions

interface Command {

void execute(Rover rover, Grid grid);

}

static class MoveCommand implements Command {

public void execute(Rover rover, Grid grid) {

moveRover(rover, grid);

}

}

static class LeftCommand implements Command {

public void execute(Rover rover, Grid grid) {

turnLeft(rover);

}

}

static class RightCommand implements Command {

public void execute(Rover rover, Grid grid) {

turnRight(rover);

}

}

}