Mars Rover Programming Exercise

import java.util.Arrays;

enum Direction {

N, E, S, W

}

class Rover {

int x;

int y;

Direction direction;

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

this.x = x;

this.y = y;

this.direction = direction;

}

}

class Grid {

static final int GRID\_SIZE = 10;

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

Grid() {

for (boolean[] row : obstacles) {

Arrays.fill(row, false);

}

// Example obstacles

obstacles[2][2] = true;

obstacles[3][5] = true;

}

boolean isObstacle(int x, int y) {

return obstacles[x][y];

}

}

interface Command {

void execute(Rover rover, Grid grid);

}

class MoveCommand implements Command {

public void execute(Rover rover, Grid grid) {

moveRover(rover, grid);

}

private void moveRover(Rover rover, Grid grid) {

int newX = rover.x;

int newY = rover.y;

switch (rover.direction) {

case N -> newY++;

case E -> newX++;

case S -> newY--;

case W -> newX--;

}

if (newX >= 0 && newX < Grid.GRID\_SIZE && newY >= 0 && newY < Grid.GRID\_SIZE && !grid.isObstacle(newX, newY)) {

rover.x = newX;

rover.y = newY;

} else {

System.out.println("Move blocked by obstacle or boundary.");

}

}

}

class LeftCommand implements Command {

public void execute(Rover rover, Grid grid) {

turnLeft(rover);

}

private void turnLeft(Rover rover) {

rover.direction = Direction.values()[(rover.direction.ordinal() + 3) % 4];

}

}

class RightCommand implements Command {

public void execute(Rover rover, Grid grid) {

turnRight(rover);

}

private void turnRight(Rover rover) {

rover.direction = Direction.values()[(rover.direction.ordinal() + 1) % 4];

}

}

public class Main {

public static void main(String[] args) {

Rover rover = new Rover(0, 0, Direction.N);

Grid grid = new Grid();

Command[] commands = {new MoveCommand(), new RightCommand(), new MoveCommand(), new LeftCommand(), new MoveCommand()};

for (Command command : commands) {

command.execute(rover, grid);

}

reportStatus(rover, grid);

}

private static void reportStatus(Rover rover, Grid grid) {

String[] directions = {"North", "East", "South", "West"};

System.out.printf("Rover is at (%d, %d) facing %s.%n", rover.x, rover.y, directions[rover.direction.ordinal()]);

if (grid.isObstacle(rover.x, rover.y)) {

System.out.println("Obstacle detected at current position.");

} else {

System.out.println("No obstacles detected.");

}

}

}