· **Obstacles**: As the VacBot moves, it updates the map with new obstacles it encounters (recordObstacle in Routing.java).

· **Target Calculation**: When the VacBot needs to move to a new target, Routing.java calculates the shortest path using the gradient map generated by GradientMap.java.

· **Gradient Descent**: The VacBot follows the path with the lowest distance values (calculated in GradientMap.java).

· **Path Following**: The VacBot follows the route by repeatedly choosing the neighboring cell with the lowest value from the gradient map.

· **GradientMap.java** calculates the grid distances based on the flood-fill algorithm and stores obstacle locations.

· **Routing.java** provides the interface for the VacBot to interact with the map, detect obstacles, and calculate routes to its target destination based on the gradient values.

· Together, these files allow the VacBot to plan a route from its current position to a target, avoiding obstacles, and update its knowledge of the environment as it moves.

Task3

**Calculate and Follow Routes**: The VacBot should be able to calculate a route to a specified target location and follow that route, considering any obstacles.