**ROBOT PATH PLANNING**

Implementation:

Compiler: Java

Input: Obstacle map, the start and goal position

Two obstacle maps are used to test the path planning (obstacleMapA, obstacleMapB).



The GUI graph layout is basically a four quadrant graph with a resolution of 1500\*900

The x-axis varies from -15 to +15 and y-axis varies from -9 to +9. The start and goal point locations can only be at the points which are not inside the obstacles.

Sample start and stop positions:

a)

Start point = -3,-2

Goal point = 7,1

b)

Start point = -3.1,0.55

Goal point = 10.5,0

c)

Start point = 0,-2

Goal point = 3,4

Formulation:

A\* algorithm is used as the state space search algorithm and the heuristics function is the straight line distance between the point and the goal state.

State Space: State space includes all the corners of the obstacles and also start and goal position

Initial State: The given starting location

Actions: The visibility graph is generated from the current state in which set of reachable states or nodes are derived checking all the obstacles

Successor function: From the set of visible nodes the successor state is formulated as the next node or vertex of an obstacle having shortest distance from the current state

Goal test: Checking whether the generated successor is a goal state or not

Path cost: Distance between current state to successor state

Design:

The input file of obstacle map consists of number of obstacles, number of edges for each obstacle and the vertices comprising edges.

All the polygonal obstacles with the given edges is maintained and their corresponding resized polygon obstacles are generated. The altering of all the obstacles corners and corresponding edges is done relative to the size of robot object. This is done for the future enhancement. The Point2d and Line2D classes in java are mainly used to maintain the obstacles.

The checking of the convexity of obstacles is performed based on grahams scan algorithm.

A separate check list is created in order to find if any corners of obstacles are overlapped on other obstacles.

The visibility graph is generated by checking whether all the lines joining the corners intercepting the obstacles or not. If the line doesn’t intersect any obstacle it is include in the visibility graph as an edge.

A\* search algorithm is used as state search algorithm in which the distance between the current state vertex and the goal is used as the heuristic function.

Java is used to create GUI with the help of input AWT libraries.

Program structure:

The program has total five main classes.

MainClass.java

This is class takes input obstacle map, starting and goal position as input and calls different methods to perform various operations on obstacles to find the shortest path between the start and goal state.

PolygonalObstacle.java

This will maintain obstacle by performing various operations on obstacles like altering the obstacles size according to robot size, finding the obstacle hull of resized polygons

PathPlanning.java

This class will generate the visibility lines for all the obstacles and also performs A\* algorithm search on all the nodes of obstacles.

GraphnodeWrapper.java

This class is a wrapper class for all the nodes of visibility graph. It maintains various attributes of the node like level cost, heuristic cost and total evaluation cost.

GUILayoutClass.java

This is for UI maintaining the graph layout which is a four quadrant graph, vertices and edges.

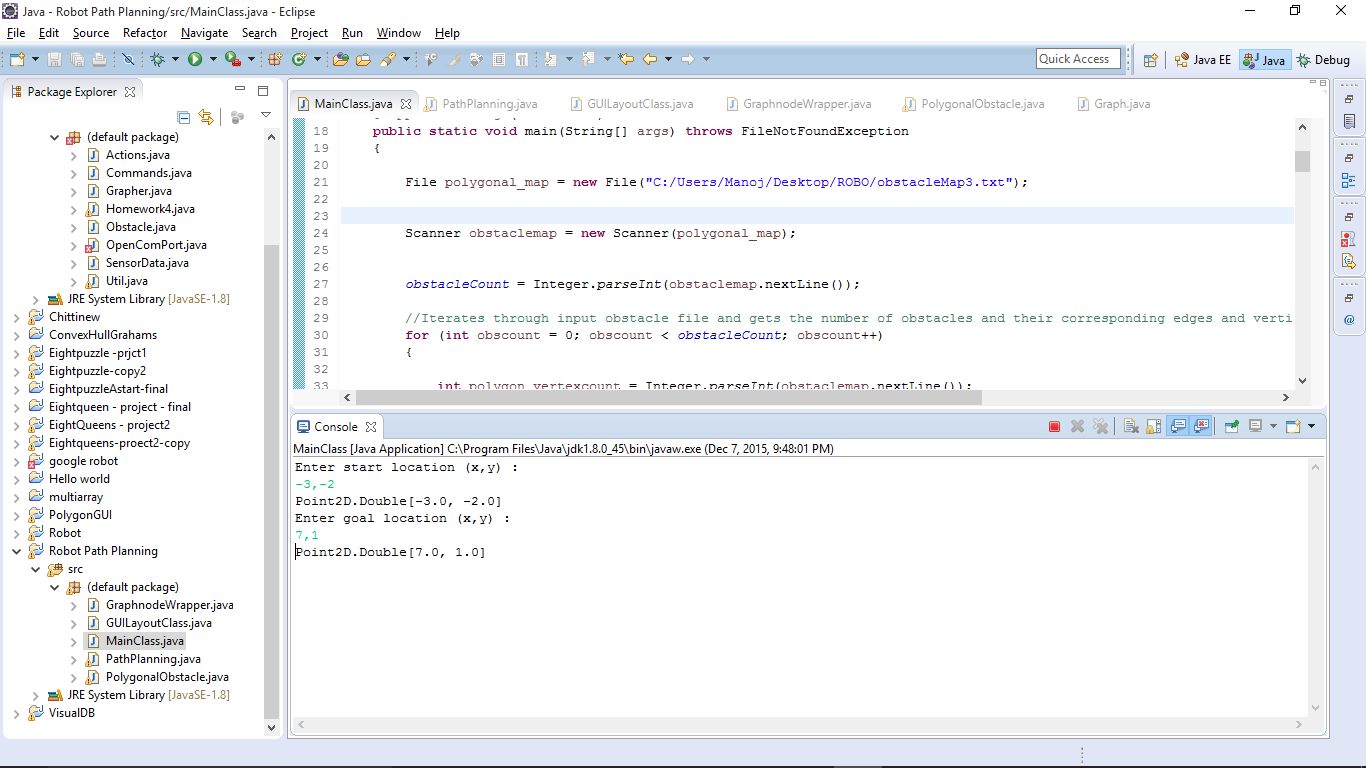
**Results**:

ObstacleMapA:

Input1:

Start point = -3,-2

Goal point = 7,1

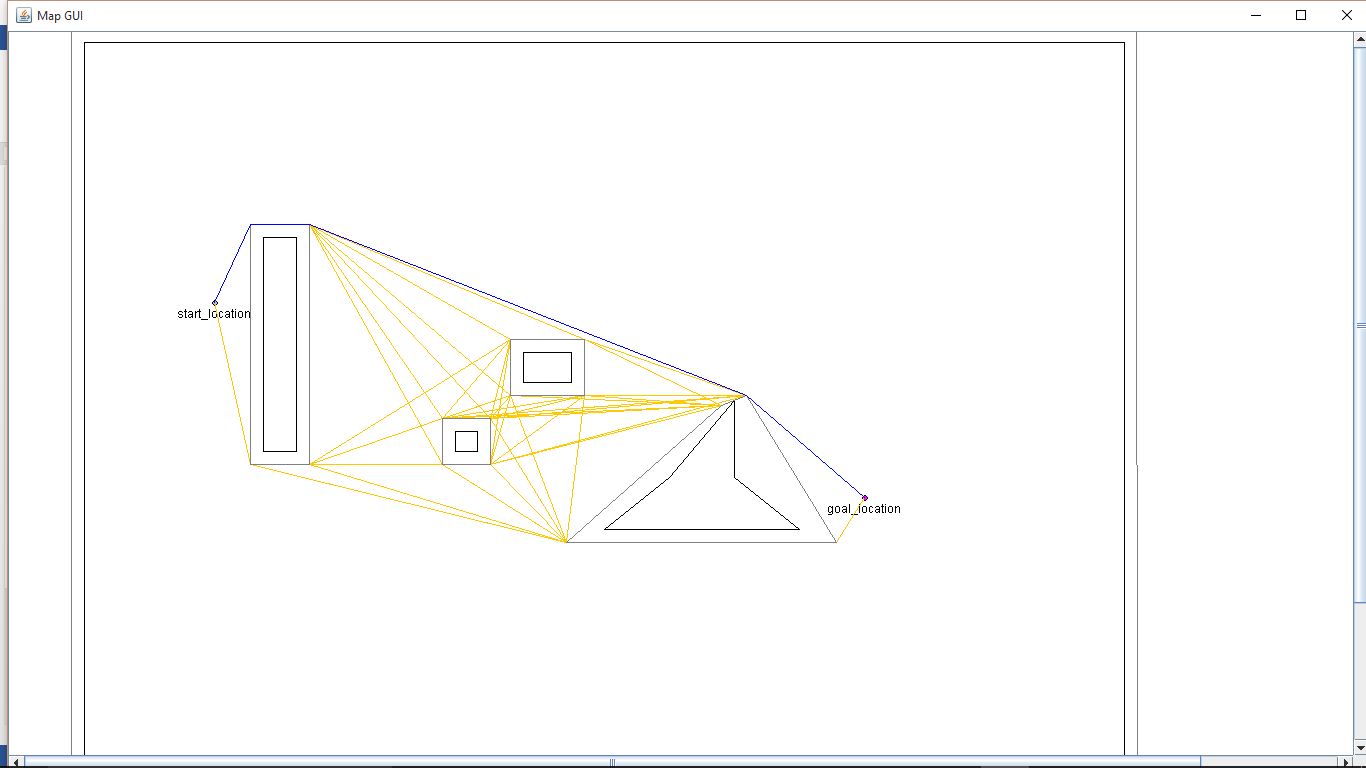


Robot path:

The blue line from start location to goal location represents the robot path.

Yellow lines represents the visibility lines between reachable nodes.

Black obstacles are the original obstacles and the pink polygons are the resized polygons of original polygons according to the radius of the robot. (This is implemented as part of future enhancement of project)

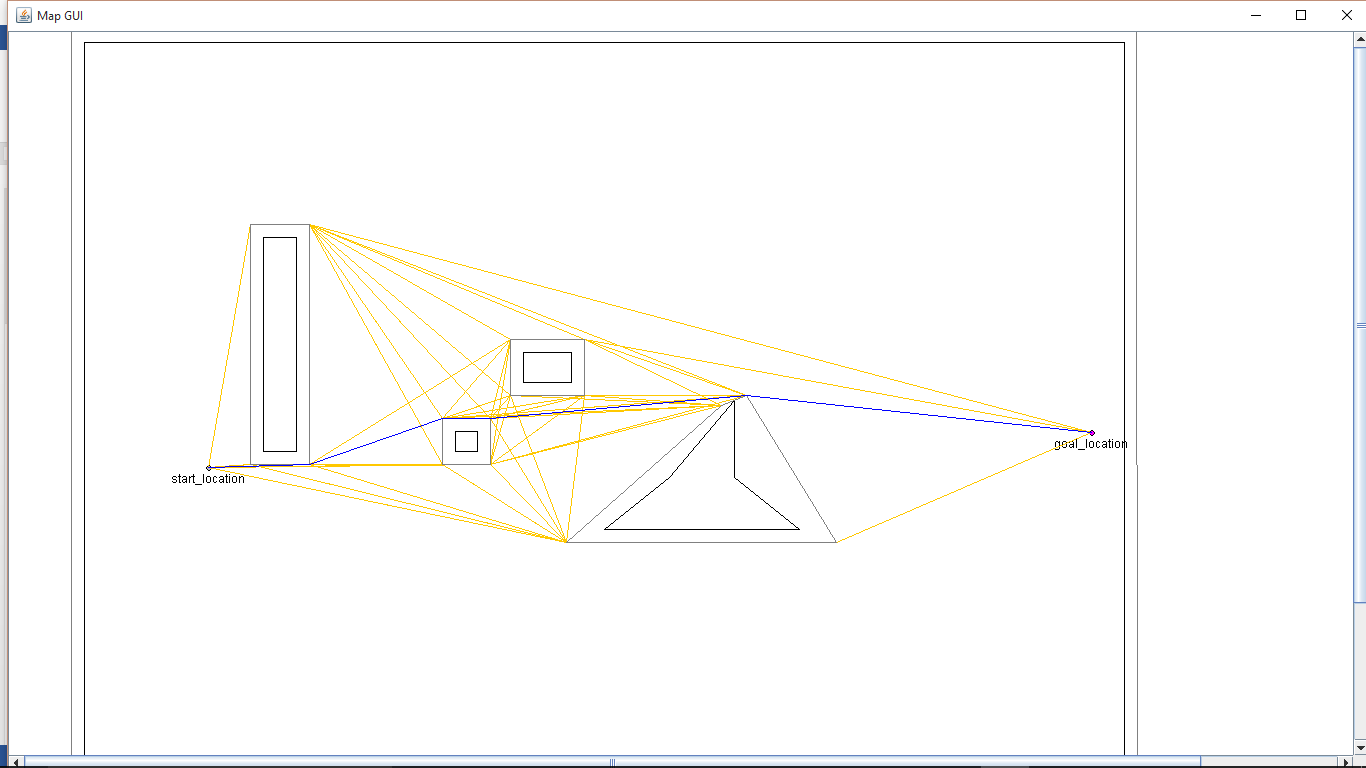


Input2:

Start point = -3.1,0.55

Goal point = 10.5,0

Robot path:

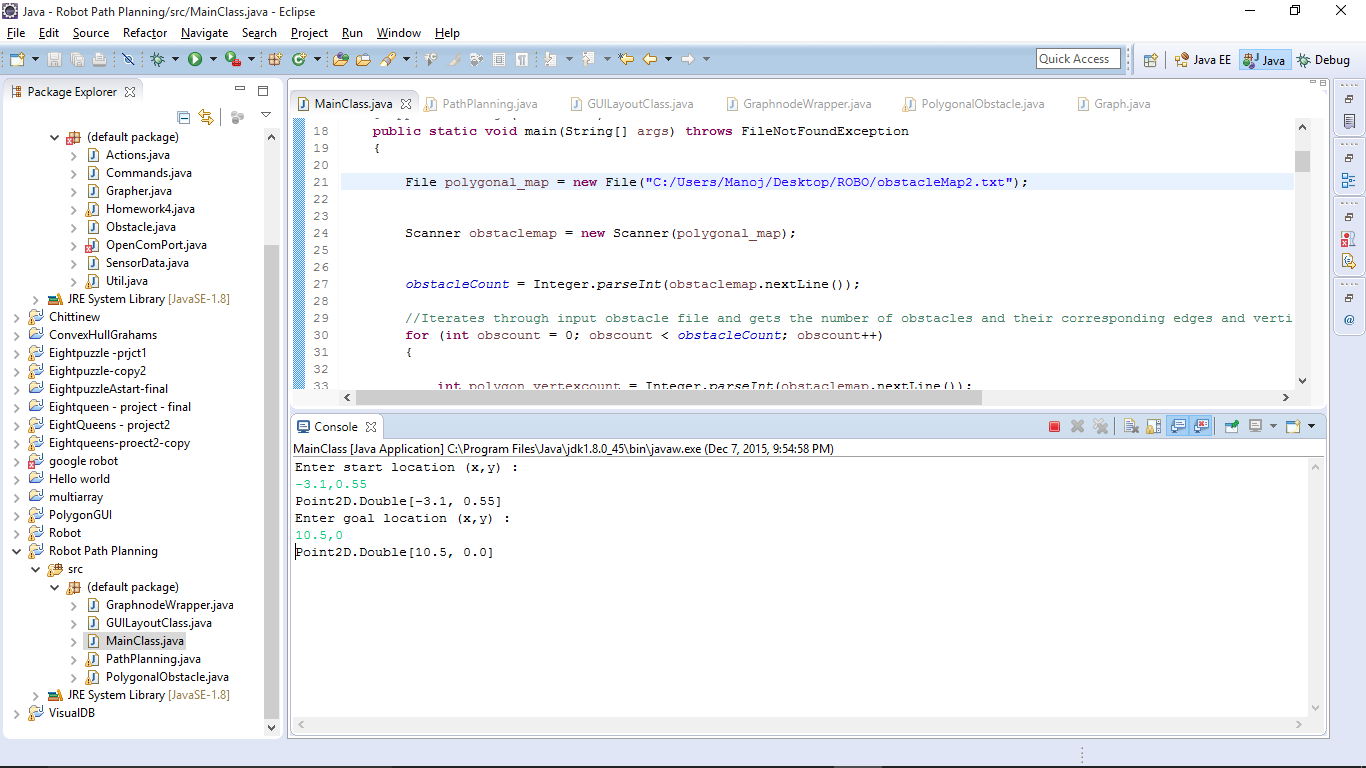


ObstacleMapB:

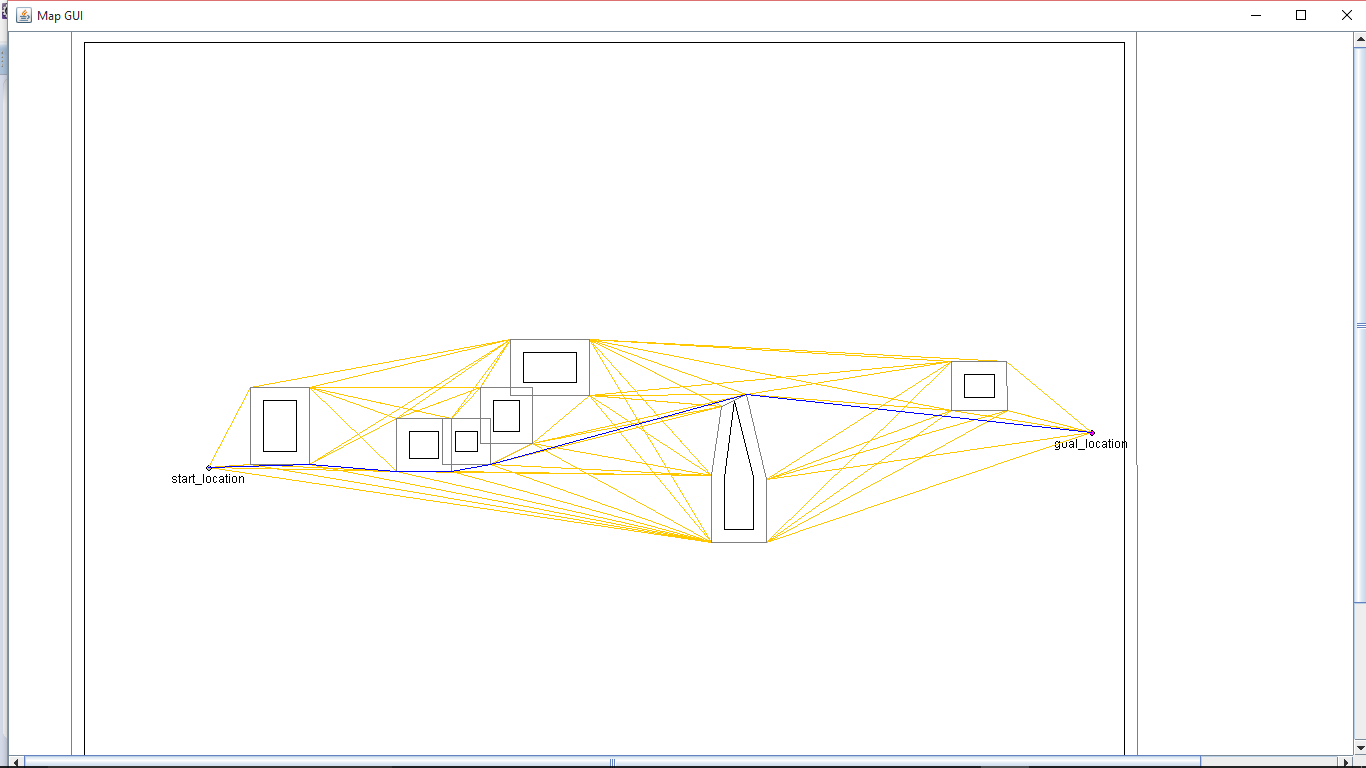
Input1:

Start point = -3.1,0.55

Goal point = 10.5,0



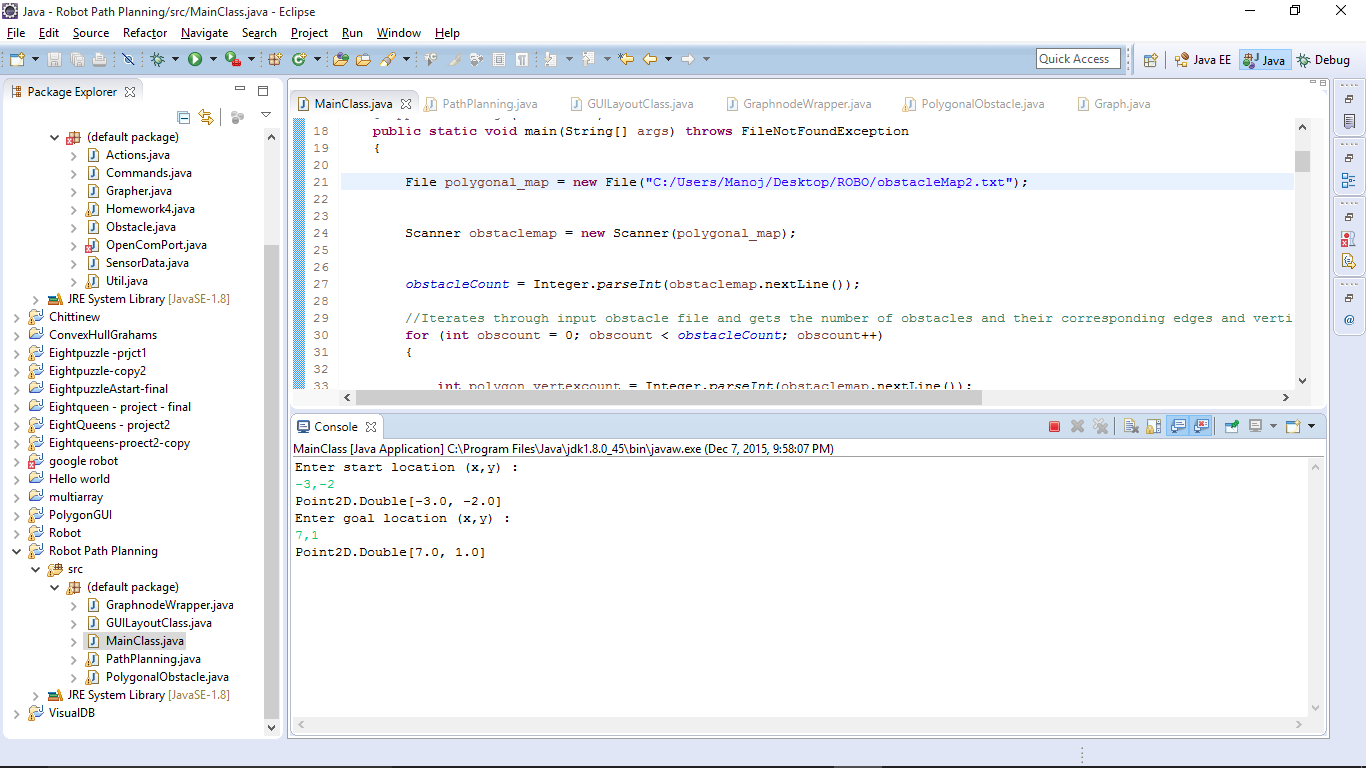
Robot path:



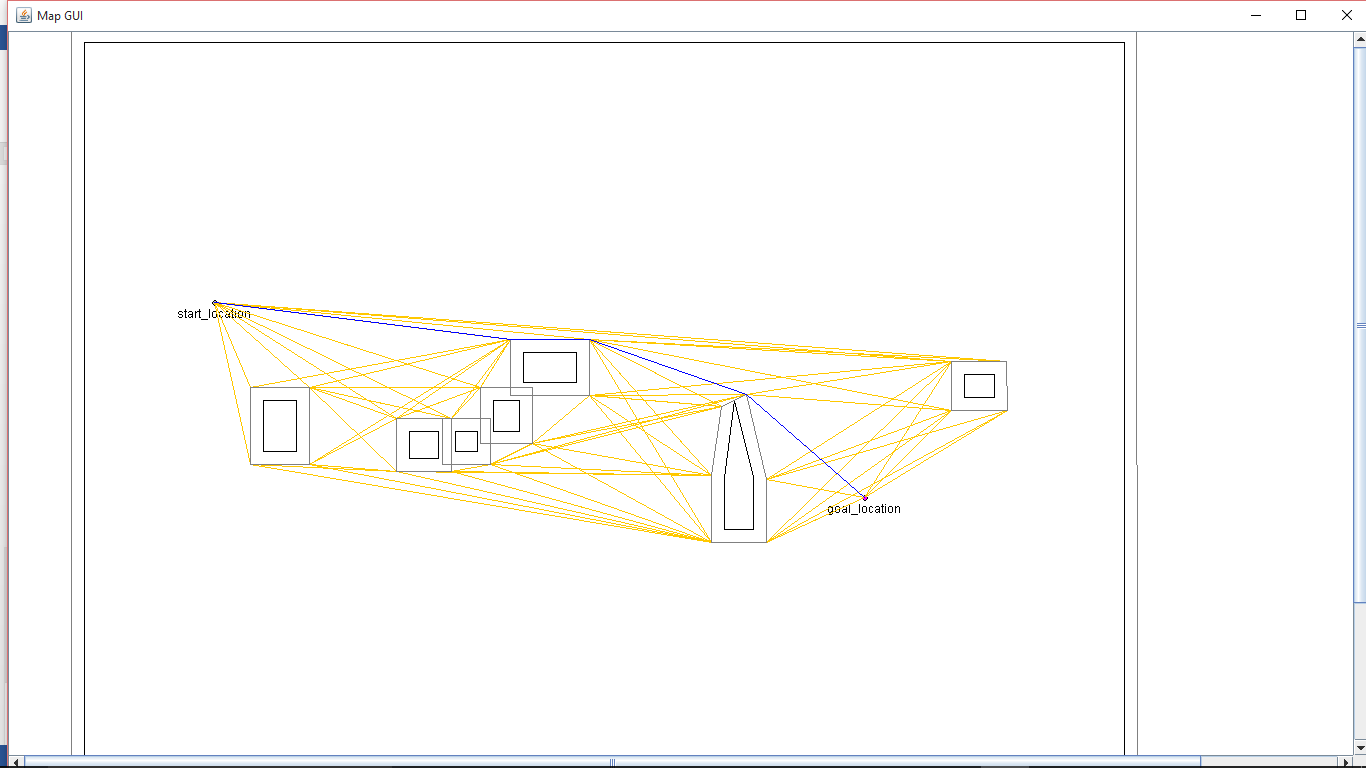
Input2:

Start point = -3,-2

Goal point = 7,1



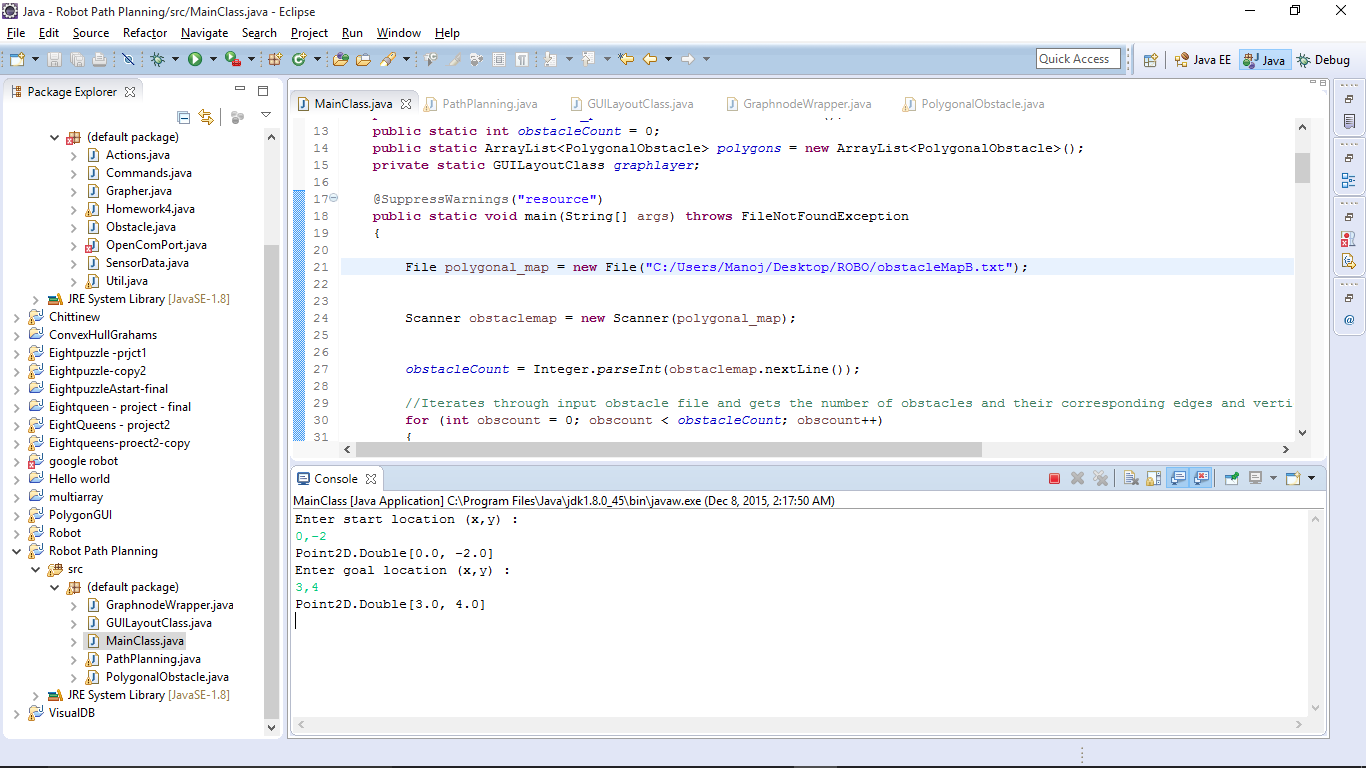
Robot path:



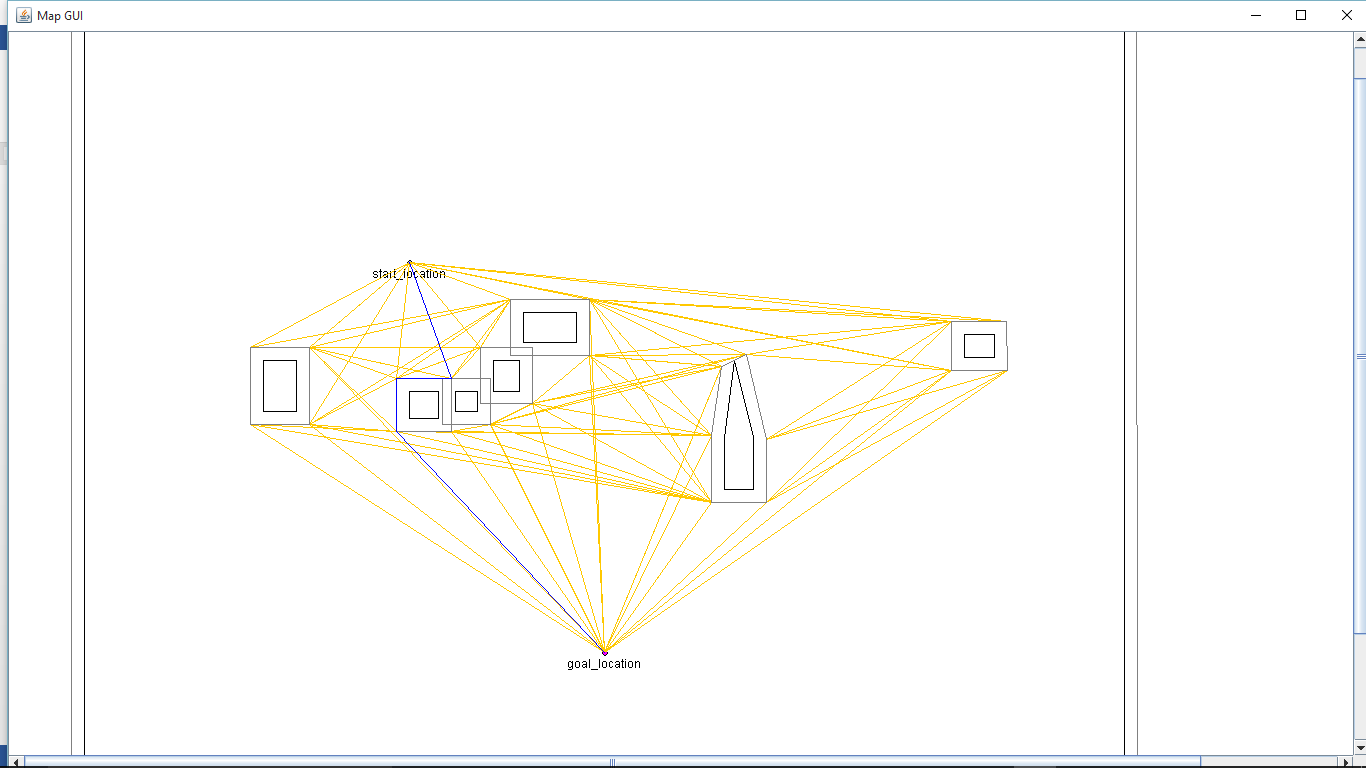
Input3:

Start point = -3,-2

Goal point = 7,1



Path:



Input Files:

ObstacleMapA:

5

4

11,7

-5,7

-5,-6

11,-6

4

-2.25,0.30

-2.25,-3

-1.75,-3

-1.75,0.30

5

3,1.5

4,0.7

5,-0.5

5,0.7

6,1.5

4

1.76,-0.77

1.76,-1.24

2.5,-1.24

2.5,-0.77

4

0.72,0.3

0.72,-0.03

1.05,-0.03

1.05,0.3

ObstacleManpB:

8

4

11,7

-5,7

-5,-6

11,-6

4

-2.25,0.30

-2.25,-0.5

-1.75,-0.5

-1.75,0.30

5

4.85,1.5

4.85,0.7

5,-0.5

5.3,0.7

5.3,1.5

4

1.76,-0.77

1.76,-1.24

2.57,-1.24

2.57,-0.77

4

1.30,-0.03

1.30,-0.5

1.7,-0.5

1.7,-0.03

4

-0.002,0.4

-0.002,-0.03

0.46,-0.03

0.46,0.4

4

0.72,0.3

0.72,-0.03

1.05,-0.03

1.05,0.3

4

8.55,-0.55

8.55,-0.90

9,-0.90

9,-0.55