```
1   package Assign_4;
2
3
4   import BasicIO.*;                // for IO classes
5
6
7   /** This class ...
8     *
9     * @author Dhairya Jaiswal
10    * @version 1.0 (2018/03/19)
    */
11
12
13  public class MazeB {
14
15    private ASCIIDisplayer display;
16    private ASCIIDisplayer display1;
17    private ASCIIDataFile  file;
18    private char[][] maze;
19
20    /** This constructor ...                                                */
21
22    public MazeB ( ) {
23
24      file = new ASCIIDataFile();
25
26      int x = file.readInt();
27      int y = file.readInt();
28
29      display = new ASCIIDisplayer(x+1,y+1);
30
31      maze = new char[x+1][y+1];
32
33      loadMazeArray(x,y+1);
34      loadMazeDisplay(x,y+1);
35
36      int xStart = file.readInt();
37      int yStart = file.readInt();
38
39      boolean ans = findPath(xStart,yStart);
40      if (ans) System.out.println("TRUE");
41
42      display1 = new ASCIIDisplayer(x+1,y+1);
43      loadMazeDisplay2(x,y+1);
44
45    }; // constructor
46
47
48    private void loadMazeArray ( int x, int y ) {
49      for (int i=0; i<x; i++) {
50        for (int j=0; j<y; j++) {
51          maze[i][j] = file.readC();
52        }
53      }
54    }
55
56    private void loadMazeDisplay ( int x, int y ) {
57      for (int i=0; i<x; i++) {
58        for (int j=0; j<y; j++) {
59          display.writeC(maze[i][j]);
60        }
61      }
62    }
63
64    private void loadMazeDisplay2 ( int x, int y ) {
65      for (int i=0; i<x; i++) {
66        for (int j=0; j<y; j++) {
67          display1.writeC(maze[i][j]);
68        }
69      }
```

```
70    }
71
72    private boolean findPath ( int x, int y ) {
73      if (maze[x][y] == ' ') {
74        maze[x][y] = '*';
75        return findPath(x,y);
76      } else if (maze[x+1][y] == ' ') {  //Checking south side
77
78        return findPath(x+1,y);
79      } else if (maze[x][y+1] == ' ') {  //Checking East Side
80
81        return findPath(x,y+1);
82
83      } else if (maze[x-1][y] == ' ') {  //Checking North Side
84
85        return findPath(x-1,y);
86
87      } else if (maze[x][y-1] == ' ') {  //Checking west side
88
89        return findPath(x,y-1);
90      } else if (maze[x+1][y] == 'E' || maze[x][y+1] == 'E' || maze[x-1][y] == 'E' ||
    maze[x][y-1] == 'E') {
91        return true;  //Once you reach the end
92      }
93
94      //BackTracking
95      if (maze[x][y] == '*') {
96        maze[x][y] = '.';
97        return findPath(x,y);
98
99      } else if (maze[x-1][y] == '*') {  //Checking North Side
100       return findPath(x-1,y);
101     } else if (maze[x][y+1] == '*') {  //Checking East Side
102       return findPath(x,y+1);
103     } else if (maze[x][y-1] == '*') {  //Checking west side
104       return findPath(x,y-1);
105     } else if (maze[x+1][y] == '*') {  //Checking south side
106       return findPath(x+1,y);
107     }
108     return false;
109   }
110
111
112
113   public static void main ( String[] args ) { MazeB c = new MazeB(); };
114
115
116 } // <className>
```