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

```
70        for (int j=0; j<y; j++) {
71          display1.writeC(maze[i][j]);
72        }
73      }
74    }
75
76    private boolean findPath ( int x, int y, char c ) {
77
78      LnkStack<Position> s = new LnkStack<Position>();
79
80      Position p;
81      p = new Position (x,y,c,0);
82
83      int xpos, ypos, ch, st;
84      st = 0;
85
86      s.push(p);
87
88      while ( s.top() != null ) {
89        ch = s.top().getC();
90        st = s.top().getS();
91        xpos = s.top().getX();
92        ypos = s.top().getY();
93        if ( s.top() == p ) {
94          st += 1;
95        }
96        if ( ch == 'E' ) break;
97        if ( st != 4 ) {
98          if ( maze[xpos+1][ypos] == ' ' || maze[xpos+1][ypos] == 'E' ) {
99            Position newP = new Position (xpos+1,ypos,maze[xpos+1][ypos],0);
100           maze[xpos][ypos] = '*';
101           s.push(newP);
102         }
103         else if ( maze[xpos-1][ypos] == ' ' || maze[xpos-1][ypos] == 'E' ) {
104           Position newP = new Position (xpos-1,ypos,maze[xpos-1][ypos],0);
105           maze[xpos][ypos] = '*';
106           s.push(newP);
107         }
108         else if ( maze[xpos][ypos+1] == ' ' || maze[xpos][ypos+1] == 'E' ) {
109           Position newP = new Position (xpos,ypos+1,maze[xpos][ypos+1],0);
110           maze[xpos][ypos] = '*';
111           s.push(newP);
112         }
113         else if ( maze[xpos][ypos-1] == ' ' || maze[xpos][ypos-1] == 'E' ) {
114           Position newP = new Position (xpos,ypos-1,maze[xpos][ypos-1],0);
115           maze[xpos][ypos] = '*';
116           s.push(newP);
117         }
118         else {
119           maze[xpos][ypos] = '.';
120           if ( s.top() != p ) {
121             s.pop();
122           }
123           else {
124             break;
125           }
126         }
127       }
128     }
129
130     if (s.top() == p) {
131       return false;
132     }
133     else {
134       return true;
135     }
136
137   }
138
139
```

```
140
141   public static void main ( String[] args ) { Maze c = new Maze(); };
142
143
144 } // <className>
```