```java
/** This class MazeWalk, solve a maze  using recursive calls
 * It asks the user for a starting location in the maze and solves it
 * from that location and prints the file.
 *
 *  *Name: Long Nguyen: Student # 5427059
 *
 * @version  1.0 (Mar. 2014)                                                */

import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;

import BasicIO.*;

public class MazeWalk {

    int row = 0; //Row size of the Maze
    int column = 0; //Column size of the Maze

 ASCIIDataFile in=new ASCIIDataFile("mz1.txt");//reading in the maze text file
  char[][] mazeArray;//size of the array
  boolean exit = false;

  boolean[][] visited; //tracking location if it's been visited

 public MazeWalk(){

  //fields for input when getting values
  int field1 = 0;
  int field2 =0;

  //creating a basic form
  BasicForm form = new BasicForm();

  //Creating the Maze.txt file to write too
  PrintWriter writer = null;
  try {
   writer = new PrintWriter("Maze.txt", "UTF-8");
  } catch (FileNotFoundException e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
  } catch (UnsupportedEncodingException e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
  }

  //Fields of the forms
   form.addTextField("field1", "X");
  form.addTextField("field2", "Y");
  form.writeString("field1", "1");
  form.writeString("field2", "1");
  form.accept();

  field1 = form.readInt("field1"); //reading from field1
     field2 = form.readInt("field2"); //reading from field2

     form.hide();//hiding the form
     ReadFile();//calling the reading file method

      findPath(field1, field2);//

     //printing out the Maze after it's been solve
      for (int i = 0; i < mazeArray.length; i++) { //row
       for (int j = 0; j < mazeArray[i].length; j++) {//column
        if(mazeArray[i][j] == ' '){//if there's a white space
         //outPut.writeChar(' '); //writing out the output to file
         writer.print(" ");
          System.out.print(" "); //print line
          }
       else{ //print the value
```

```
71            //writing out the output to file
72           writer.print(mazeArray[i][j]);
73         System.out.print( mazeArray[i][j]);
74          }
75          }
76         //writing a new line to the file
77          writer.println();
78            System.out.println();
79    }//end for loop
80
81        //closing the file
82     writer.close();
83     //closing the form
84     form.close();
85   }
86
87   /*recursive method that calls itself until it finds 'E'/exit*/
88
89   public void findPath(int x, int y){
90    //System.out.println("OutSide Value of x " + x + "Value of y " + y);
91    //System.out.println("x = " + x + " y = " + y);
92    if(mazeArray[x][y] == 'E') {
93     System.out.println("Found");
94    exit = true; return;}
95     if(mazeArray[x][y] == '#' || visited[x][y] == true ){
96     return;
97    } //else{
98
99      visited[x][y] = true;//been here
100    //mazeArray[x][y] = '.';
101
102
103 while(mazeArray[x][y] != 'E'){
104
105  /*Going Forward*/
106   if(exit == false) {
107      mazeArray[x][y] ='>';
108   findPath(x,y+1);
109   }
110   /*Going backward*/
111           if(exit == false) {
112            mazeArray[x][y] ='<';
113            findPath(x,y-1);
114            }
115     /*going up*/
116      if(exit == false) {
117       mazeArray[x][y] ='^';
118       findPath(x-1,y);
119       }
120      /*going down*/
121           if(exit == false) {
122            mazeArray[x][y] ='V';
123            findPath(x+1,y);
124            }
125             /*If checked all the position*/
126           if(mazeArray[x][y] == 'V') {
127            mazeArray[x][y] ='.';
128            }
129    return;
130 }
131
132
133  } //end of findPath method
134
135  private void ReadFile() {
136
137    row = in.readInt(); //getting the row size of the Maze
138    column = in.readInt(); //getting the column size of the Maze
139    mazeArray = new char [row][column]; //creating the Maze base on the values when
    reading in
```

```
140    visited = new boolean [row][column];  //creating the Maze base on the values
    when reading in, for the visted part
141
142    //reading a line and creating a charArray base on each character
143       for (int i = 0; i < mazeArray.length; i++) {
144     mazeArray[i] = in.readLine().toCharArray();
145    }
146       //System.out.println(mazeArray[3][10]);
147   in.close(); //close the file
148  }
149  public static void main(String[] args) {
150   new MazeWalk();
151
152  }
153
154 }
```