1

```
package p3 submit;
* Project 3 - Maze
* @author Youchen Ren
import java.io.*;
public class Project3 {
      public static int [][] maze = new int[10][10];
      public static int [][] temp = new int [10][10];
      public static boolean found = false;
      public static int nr; //new row
      public static int nc; //new column
      public static int count = 0;
      public static void main(String[] args) {
            if(args.length == 0)System.out.println("No file specified.");
            else{
                  FileReader theFile;
                  BufferedReader inFile;
                  String oneLine;
                  try{//FileNotFoundException must be caught
                        theFile = new FileReader(args[0]);
                        inFile = new BufferedReader(theFile);
                        /**
                         * now read the text file line by line.
                        while ((oneLine = inFile.readLine()) != null){
                              String numbers[] = oneLine.split(" ");
                              for(int i = 0; i <= 9; i++) {</pre>
                                    for(int j = 0; j <= 9; j++) {
                                          maze[i][j] = Integer.parseInt(numbers[j]);
                                          temp[i][j] = Integer.parseInt(numbers[j]);
                                    if (i <= 8) {
                                          oneLine = inFile.readLine();
                                          numbers = oneLine.split(" ");
                              }//for
                              Findway(0, 0, 9, 9);
                              if (found == true) {
                                    for(int i = 0; i <= 9; i++) {
                                          for(int j = 0; j <= 9; j++) {</pre>
                                                 System.out.print(maze[i][j]+" ");
                                          System.out.println();
                              }//if
                              else {
                                    System.out.println("\nSorry! No path was founded for
                                                                            this Maze."):
                              inFile.readLine();//This line is Empty new line
                              System.out.println("\n\n\n\n\n");
                              found = false;
                              count = 0;
```

2

```
}//while
                  }//try
                  catch (Exception e){System.out.println(e);}
            }//else
      }//main
      public static void Findway(int sr, int sc, int dr, int dc){
            if(sr == dr && sc == dc) found = true;
            else{
                  temp[sr][sc] = 1;//mark the position
                  while(!found && PossibleToMove(sr, sc)) {
                        count = 0;
                        Findway(nr, nc, dr, dc);
                        if(!found) {
                              System.out.println("Backtracking from [" + nr + ", " + nc
                                                       +" | to ["+ sr + ", " + sc + "]");
                              nr = sr; nc = sc;
                        }
                  }//while
                  if(!PossibleToMove(sr, sc) && count == 0) {
                        System.out.println("\nStart backtracking from [" + sr +", " + sc
                        count++;
                  }
            if(found) maze[sr][sc] = 2; //mark the path
      }//Findway
      /**
       *
                 | ? |
            sr | ? | x | ? |
                 | ? |
                 SC
       *PossibleToMove -> Returns false if there is no place to go from(sr, sc);
                  otherwise, returns true and stes(nr, nc) to a new position.
       * The order of Directions of searching for obstacles is
                  East -> South -> West -> North
       * @param sr start row for testing if it's possible to move to either four
      directions.
       * @param sc start column for testing if it's possible to move to either four
       * @return if possible to move, return true, else false.
      public static boolean PossibleToMove(int sr, int sc) {
            if(sc+1 \le 9 \&\& temp[sr][sc+1] == 0) \{nr = sr; nc = sc+1; return true;\}
            else if(sr+1 <= 9 && temp[sr+1][sc] == 0) {nr = sr+1; nc = sc; return true;}</pre>
            else if(sc-1 >= 0 && temp[sr][sc-1] == 0) {nr = sr; nc = sc-1; return true;}
            else if(sr-1 >= 0 && temp[sr-1][sc] == 0) {nr = sr-1; nc = sc; return true;}
            else return false;
      }
}//Class
```

3

```
Start backtracking from [3, 9]
Backtracking from [3, 9] to [3, 8]
Backtracking from [3, 8] to [4, 8]
Backtracking from [4, 8] to [5, 8]
Backtracking from [5, 8] to [5, 9]
Backtracking from [5, 9] to [6, 9]
Backtracking from [6, 9] to [7, 9]
Backtracking from [7, 9] to [7, 8]
Start backtracking from [8, 8]
2100000001
2 2 2 1 0 1 0 1 0 1
0 1 2 1 0 1 1 1 1 1
1 2 2 1 0 0 0 1 0 0
1 2 1 1 0 1 0 1 0 1
1 2 2 2 1 1 0 1 0 0
0012222110
0 1 1 0 1 1 2 2 2 0
1 1 0 1 1 1 0 1 2 1
1000000122
```

```
Start backtracking from [3, 6]
Backtracking from [3, 6] to [3, 5]
Backtracking from [3, 5] to [3, 4]
Backtracking from [3, 4] to [2, 4]
Start backtracking from [0, 5]
Backtracking from [0, 5] to [0, 4]
Start backtracking from [0, 3]
Backtracking from [0, 3] to [0, 4]
Backtracking from [0, 4] to [1, 4]
Backtracking from [1, 4] to [2, 4]
Backtracking from [2, 4] to [2, 3]
Backtracking from [2, 3] to [2, 2]
Backtracking from [2, 2] to [1, 2]
Backtracking from [1, 2] to [1, 1]
Backtracking from [1, 1] to [0, 1]
Backtracking from [0, 1] to [0, 0]
```

Sorry! No path was founded for this Maze.

```
Start backtracking from [0, 8]
Backtracking from [0, 8] to [0, 9]
Backtracking from [0, 9] to [1, 9]
Backtracking from [1, 9] to [2, 9]
```

```
Backtracking from [2, 9] to [2, 8]
Start backtracking from [4, 9]
Backtracking from [4, 9] to [4, 8]
Start backtracking from [6, 8]
Backtracking from [6, 8] to [5, 8]
Start backtracking from [6, 6]
Backtracking from [6, 6] to [5, 6]
Start backtracking from [8, 8]
2 2 2 2 1 1 0 1 0 0
0 1 1 2 2 2 2 1 1 0
0110112220
1 1 0 1 1 1 0 1 2 1
1000222120
0 1 0 1 2 1 2 2 2 1
0001210101
0 1 0 1 2 1 1 1 1 1
1 0 1 1 2 1 2 2 2 1
1001222122
```

```
Start backtracking from [0, 8]
Backtracking from [0, 8] to [0, 9]
Backtracking from [0, 9] to [1, 9]
Backtracking from [1, 9] to [2, 9]
Backtracking from [2, 9] to [2, 8]
Start backtracking from [4, 9]
Backtracking from [4, 9] to [4, 8]
Start backtracking from [6, 8]
Backtracking from [6, 8] to [5, 8]
Start backtracking from [6, 6]
Backtracking from [6, 6] to [5, 6]
Start backtracking from [4, 5]
Backtracking from [4, 5] to [4, 6]
Start backtracking from [3, 6]
Backtracking from [3, 6] to [4, 6]
Backtracking from [4, 6] to [5, 6]
Backtracking from [5, 6] to [5, 7]
Backtracking from [5, 7] to [5, 8]
Backtracking from [5, 8] to [4, 8]
Backtracking from [4, 8] to [3, 8]
Backtracking from [3, 8] to [2, 8]
Backtracking from [2, 8] to [2, 7]
Backtracking from [2, 7] to [2, 6]
Backtracking from [2, 6] to [1, 6]
```

Start backtracking from [0, 6]

```
Backtracking from [0, 6] to [1, 6]
Backtracking from [1, 6] to [1, 5]
Backtracking from [1, 5] to [1, 4]
Backtracking from [1, 4] to [1, 3]

Start backtracking from [2, 3]
Backtracking from [2, 3] to [1, 3]
Backtracking from [1, 3] to [0, 3]
Backtracking from [0, 3] to [0, 2]
Backtracking from [0, 2] to [0, 1]
Backtracking from [0, 1] to [0, 0]

Start backtracking from [2, 0]
Backtracking from [2, 0] to [1, 0]
Backtracking from [1, 0] to [0, 0]
```

Sorry! No path was founded for this Maze.

Page | 5