**import** java.applet.Applet;

**import** java.awt.\*;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.io.\*;

**import** java.util.Scanner;

/\*\*

**@author** Govind Yatnalkar & Kanimozhi Kalaichelvan

\* **@Marshall** ID: 901-87-5614, 901881363

\* **@date**: 04/01/2018

\* **@CS580** Advanced OOPS Programming Assignment - JAVA Project - Maze Solver

\*/

**public** **class** MazeSolverTest **extends** Applet{

**static** **char**[][] *maze*;

**int** n, startX, startY, goalX, goalY;

String filename="C:\\Users\\Ezhil Malliga\\eclipse-workspace\\AssignmentLab\_MazeProject\\src\\maze5.txt";

String filename1="C:\\Users\\Ezhil Malliga\\eclipse-workspace\\AssignmentLab\_MazeProject\\src\\sample.txt";

// public void paint(Graphics g) {

// MazeSolver myMaze = new MazeSolver("C:\\Users\\Ezhil Malliga\\eclipse-workspace\\AssignmentLab\_MazeProject\\src\\maze5g.txt");

// myMaze.readMaze();

// myMaze.paint(g);

//// myMaze.solveMaze();

// if(myMaze.solveMaze(g))

// System.out.println("Maze Solved Successfully!!");

// else

// System.out.println("Maze Not Solvable...");

// }

Button nextButton = **new** Button("Next");

**int** count =0;

**public** **void** init() {

**this**.add(nextButton);

**this**.setSize(500,500); //you can choose the size

ActionListener listener = **new** NextListener();

nextButton.addActionListener(listener);

}

**class** NextListener **implements** ActionListener{

**public** **void** actionPerformed(ActionEvent event)

{

**if**(solveMaze())

System.***out***.println("Maze Solved Successfully!!");

**else**

System.***out***.println("Maze Not Solvable...");

WriteMaze();

// solveMaze();

repaint();

}

}

**public** **void** readMaze() {

**try**

{

FileReader fr = **new** FileReader(filename);

BufferedReader br = **new** BufferedReader(fr);

n = Integer.*parseInt*(br.readLine());

*maze* = **new** **char**[n][n];

**for**(**int** i = 0; i < n ;i++)

{

String s = br.readLine();

**for**(**int** j = 0; j< n; j++)

{

*maze*[i][j] = s.charAt(j);

**if**(*maze*[i][j] == 'S')

{

//Code to get start coordinate

startX=i;

startY=j;

}

**if**(*maze*[i][j] == 'G')

{

//Code to get the end coordinate

goalX=i;

goalY=j;

}

}

}

}

**catch**(FileNotFoundException e)

{

e.getMessage();

e.printStackTrace();

System.***out***.println("File Not Found");

}

**catch**(IOException e) {

e.getMessage();

e.printStackTrace();

System.***out***.println("Invalid Entry");

}

}

**public** **void** paint(Graphics g)

{

readMaze();

System.***out***.println();

**int** k= **this**.getHeight()/n;

**int** l= **this**.getWidth()/n;

**for**(**int** i = 0; i < n ; i++)

{

**for**(**int** j = 0; j < n ; j++)

{

**if**((*maze*[i][j]=='S')||(*maze*[i][j]=='G'))

{

g.setColor(Color.***RED***);

g.fillRect(i\*l,j\*k,l,k);

}

**if**(*maze*[i][j]=='#')

{

g.setColor(Color.***BLACK***);

g.fillRect(i\*l,j\*k,l,k);

}

**if**(*maze*[i][j]=='.')

{

g.setColor(Color.***WHITE***);

g.fillRect(i\*l,j\*k,l,k);

}

System.***out***.print(*maze*[i][j]);

**if**(*maze*[i][j]=='P')

{

g.setColor(Color.***CYAN***);

g.fillRect(i\*l,j\*k,l,k);

}

// System.out.print(maze[i][j]);

}

}

}

**public** **boolean** solveMaze()

{

**boolean** status = solveMaze(startX, startY);

**return** status;

}

**private** **boolean** solveMaze(**int** x, **int** y) {

// Scanner mys = new Scanner(System.in);

System.***out***.println("Press Enter for next step \n "

+ "Currently Checking row = "+x+ " and column ="+y+" goalX = "+goalX+ " goalY = "+goalY );

// String e = mys.nextLine();

**if**(x<0 || x>=n || y<0 || y>=n)

**return** **false**;

**if**(x == goalX && y == goalY)

**return** **true**;

**if** (*maze*[x][y] == '#') {

**return** **false**;

}

**if** (*maze*[x][y] == 'G') {

**return** **true**;

}

**if** (*maze*[x][y] == 'P') {

**return** **false**;

}

**if**(*maze*[x][y]=='.')

{

*maze*[x][y] = 'P';

displayMaze();

}

**if**(solveMaze(x+1,y) == **true**)

**return** **true**;

**if**(solveMaze(x, y+1) == **true**)

**return** **true**;

**if**(solveMaze(x-1,y) == **true**)

**return** **true**;

**if**(solveMaze(x,y-1) == **true**)

**return** **true**;

// ActionListener listener = new NextListener();

// nextButton.addActionListener(listener);

**return** **false**;

}

**public** **void** displayMaze()

{

System.***out***.println();

**for**(**int** i = 0; i < n ; i++)

{

**for**(**int** j = 0; j < n ; j++)

System.***out***.print(*maze*[i][j]);

System.***out***.println();

}

}

**public** **void** WriteMaze() {

**try**

{

// BufferedWriter writer = new BufferedWriter(new FileWriter(filename1, true));

// n = Integer.parseInt(br.readLine());

// maze = new char[n][n];

// String str1= "Hello";

// char str = (char) n;

// writer.append(str1);

// writer.newLine();

// for(int i = 0; i < n ;i++)

// {

// for(int j = 0; j< n; j++)

// {

// str= maze[i][j];

// br.append(str);

// }

// br.newLine();

// }

FileWriter fwrite = **new** FileWriter(filename);

PrintWriter pw = **new** PrintWriter(fwrite);

String len = Integer.*toString*(n);

fwrite.write(len);

pw.println();

// fwrite.write("This is second line");

**for**(**int** i = 0; i < n ;i++)

{

**for**(**int** j = 0; j< n; j++)

{

String str= Character.*toString*(*maze*[i][j]);

fwrite.append(str);

}

pw.println();

}

fwrite.close();

// String bytes = Integer.toString(n);

// byte[] buffer = bytes.getBytes();

// FileOutputStream outputStream = new FileOutputStream(filename1);

// BufferedWriter writer = new BufferedWriter(new FileWriter(filename1, true));

// outputStream.write(buffer);

// outputStream.println();

// outputStream.write(buffer);

}

**catch**(FileNotFoundException e)

{

e.getMessage();

e.printStackTrace();

System.***out***.println("File Not Found");

}

**catch**(IOException e) {

e.getMessage();

e.printStackTrace();

System.***out***.println("Invalid Entry");

}

}

}