ASSIGNMENT 7 – MAZE SOLVER

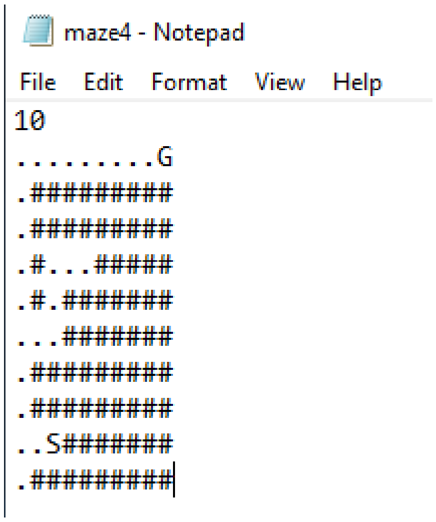
SUBMISSION

Submit your code to the Blackboard Assignment by the due date. Plan to demonstrate your working program to the instructor on or before the due date in class.

DETAILS

In this assignment, you will use a Depth First Search to solve a maze. In class, we discussed how to represent a Maze as a text file.

Here is a maze created in Notepad. The first line contains the integer value 10 which indicates that this maze has 10 rows and 10 columns.



```
10
.........G
.#########
.#########
.#...#####
.#.#######
...#######
.#########
.#########
..S#######
.#########
```

My suggestion is that you write two Java classes like those described below

- `MazeSolver`
- `MazeTest`

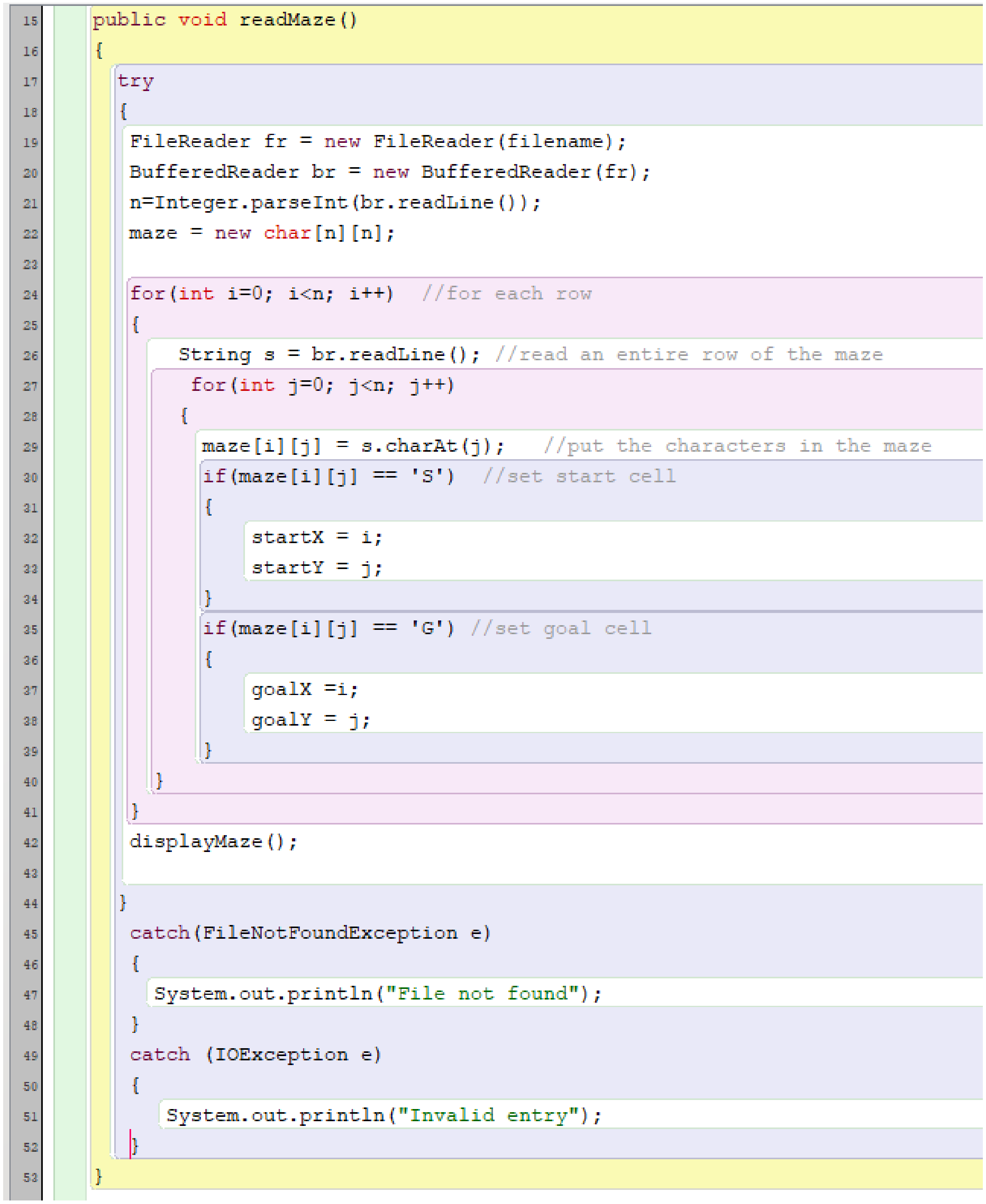
`MazeSolver` had the following properties

- Member variable `n` indicating the size of the n x n member variable called `maze.` `maze` stores the values of the maze and the current path
- Member variables `startX, startY, goalX`, and `goalY` indicating the starting and goal positions in the maze
- Method `readMaze` to read a maze
- Method `displayMaze` to display a maze in text
- Method `solveMaze` that finds the path between the start and goal positions. You were given detailed pseudocode for solveMaze on slide 70

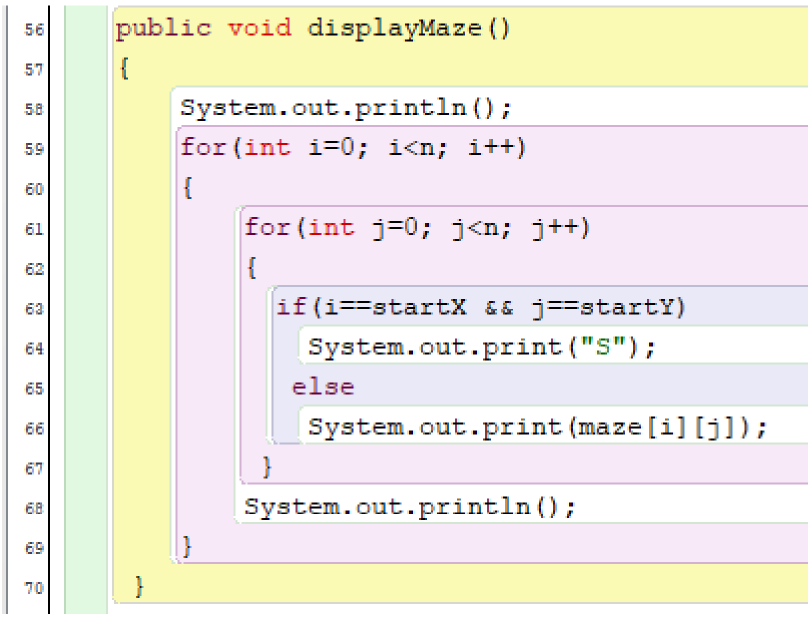
You were shown a sample program that solves a maze using the algorithm described above.

ASSIGNMENT 6 – MAZE SOLVER

Below are the methods the instructor used: `readMaze` and `displayMaze`.

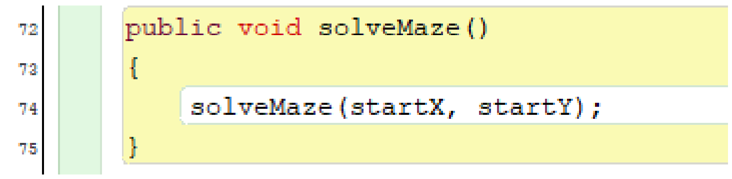


```java
public void readMaze()
{
    try
    {
        FileReader fr = new FileReader(filename);
        BufferedReader br = new BufferedReader(fr);
        n=Integer.parseInt(br.readLine());
        maze = new char[n][n];

        for(int i=0; i<n; i++)  //for each row
        {
            String s = br.readLine(); //read an entire row of the maze
            for(int j=0; j<n; j++)
            {
                maze[i][j] = s.charAt(j);   //put the characters in the maze
                if(maze[i][j] == 'S')  //set start cell
                {
                    startX = i;
                    startY = j;
                }
                if(maze[i][j] == 'G') //set goal cell
                {
                    goalX =i;
                    goalY = j;
                }
            }
        }
        displayMaze();

    }
    catch(FileNotFoundException e)
    {
        System.out.println("File not found");
    }
    catch (IOException e)
    {
        System.out.println("Invalid entry");
    }
}
```



```
public void displayMaze()
{
    System.out.println();
    for(int i=0; i<n; i++)
    {
        for(int j=0; j<n; j++)
        {
            if(i==startX && j==startY)
                System.out.print("S");
            else
                System.out.print(maze[i][j]);
        }
        System.out.println();
    }
}
```

I suggest that the MazeSolver have a public method that looks like:



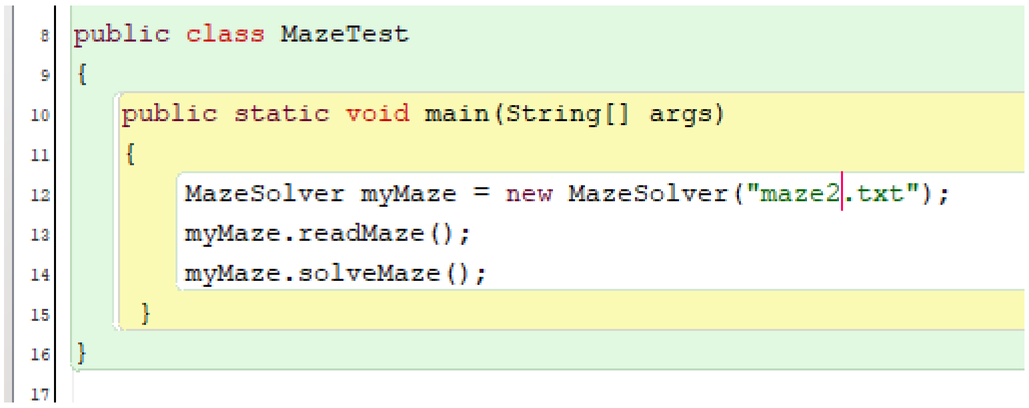
```
public void solveMaze()
{
    solveMaze(startX, startY);
}
```

Then you should write a private method called solveMaze with the following signature:

public boolean solveMaze(int x, int y)

This method should implement the algorithm called findpath that we covered in the lecture.

The MazeTest program is very simple:



```
public class MazeTest
{
    public static void main(String[] args)
    {
        MazeSolver myMaze = new MazeSolver("maze2.txt");
        myMaze.readMaze();
        myMaze.solveMaze();
    }
}
```

If you reproduce the text version demonstrated on class, you can earn up to 32 points (80%). If you use Graphics, you may earn up to 100%.