Homework Turnin

Email:	rgalanos@fcps.edu
Section:	6G
Course:	TJHSST APCS 2016-17
Assignment:	02-07
Receipt ID:	f485d11482452f9f52a5db84a87aba86

Warning: Your turnin is 3 days late. Assignment 02-07 was due Friday, October 21, 2016, 11:30 PM.

Replacing prior submission from Mon 2016/10/24 07:09pm.

Turnin Successful!

The following file(s) were received:

MazeMaster.java (7484 bytes)

```
//name:    date:
import java.util.*;
import java.io.*;

public class MazeMaster
{
   public static void main(String[] args)throws FileNotFoundException
   {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter the maze's filename: ");
      char[][] retArr = buildCharArr(sc.next());
      Maze m = new Maze(retArr);
      m.display();
      System.out.println("Options: ");
      System.out.println("1: Mark all paths.");
      System.out.println("2: Mark all paths, and display the count of all steps.");
      System.out.println("3: Show only the correct path.");
      System.out.println("4: Show only the correct path. If no path exists, display 'No path exists'.");
      System.out.println("5: Show only the correct path, and display the count of steps.");
      System.out.print("Please make a selection: ");
      m.solve(sc.nextInt());
      m.display();
   }
   //take in a filename, and return a char[][]
   public static char[][] buildCharArr(String fileName)throws FileNotFoundException
   {
      Scanner infile = new Scanner(new File(fileName));
      int rows = infile.nextInt();
      int columns = infile.nextInt();
      char[][] maze = new char[rows][columns];
      infile.nextLine();


      for(int i=0;i<rows;i++)
      {
         String line = infile.nextLine();
         for(int a=0;a<columns;a++)
         {
            maze[i][a] = line.charAt(a);
         }
```

```
      }
      return maze;
   }
}


class Maze
{
   //Constants
   private final char WALL = 'W';
   private final char PATH = '.';
   private final char START = 'S';
   private final char EXIT = 'E';
   private final char STEP = '*';
   //fields
   private char[][] maze;
   private int startRow, startCol;
   private boolean S_Exists=false, E_Exists=false;
   int counter = 0;
   //constructor initializes all the fields
   public Maze(char[][] inCharArr)
   {
      maze = inCharArr;
      startRow = 0;
      startCol = 0;

      for(int i=0;i<maze.length;i++)
      {
         for(int a=0;a<maze[0].length;a++)
         {
            if( maze[i][a] == START)
            {
               startRow = i;
               startCol = a;
               S_Exists = true;
            }
            else if(maze[i][a] == EXIT)
            {
               E_Exists = true;
            }
         }
      }
      if(!(S_Exists&&E_Exists))
      {
         System.out.println("This maze cannot be solved");
         System.exit(0);
      }
   }

   public void display()
   {
      if(maze==null)
         return;
      for(int a = 0; a<maze.length; a++)
      {
         for(int b = 0; b<maze[0].length; b++)
         {
            System.out.print(maze[a][b]);
         }
         System.out.println("");
      }
      System.out.println("");
   }
   public void solve(int n)
   {
      if(n==1)
      {
         markAllPaths(startRow, startCol);
      }
      else if(n==2)
      {
         int count = markAllPathsAndCountStars(startRow, startCol);
         System.out.println("Number of steps = " + count);
      }
      else if(n==3)
      {
         displayTheCorrectPath(startRow, startCol);
      }
      else if(n==4)   //use maze3 here
      {
         if( !displayTheCorrectPath(startRow, startCol) )
```

```
            System.out.println("No path exists");
        }
        else if(n==5)
        {
            displayCorrectPathAndCountStars(startRow, startCol, 0);
        }
        else System.out.println("invalid submission");

    }
    private void markAllPaths(int r, int c)
    {
        if(maze[r][c]==EXIT);

        else
        {
            maze[r][c] = STEP;
            if(r<maze.length-1)
            {
                if(maze[r+1][c]!=WALL&&maze[r+1][c]!=STEP)
                {
                    markAllPaths(r+1,c);
                }
            }
            if(r>0)
            {
                if(maze[r-1][c]!=WALL&&maze[r-1][c]!=STEP)
                {

                    markAllPaths(r-1, c);
                }
            }
            if(c<maze[0].length-1)
            {
                if(maze[r][c+1]!=WALL&&maze[r][c+1]!=STEP)
                {

                    markAllPaths(r, c+1);
                }
            }
            if(c>0)
            {
                if(maze[r][c-1]!=WALL&&maze[r][c-1]!=STEP)
                {

                    markAllPaths(r, c-1);
                }
            }
        }

    }

    private int markAllPathsAndCountStars(int r, int c)
    {
        int counter = 0;

        if(maze[r][c]==EXIT)
        {
            counter += 0;
        }

        else
        {
            maze[r][c] = STEP;
            if(r<maze.length-1)
            {
                if(maze[r+1][c]!=WALL&&maze[r+1][c]!=STEP)
                {
                    counter += 1+markAllPathsAndCountStars(r+1,c);
                }
            }
            if(r>0)
            {
                if(maze[r-1][c]!=WALL&&maze[r-1][c]!=STEP)
                {

                    counter += 1+markAllPathsAndCountStars(r-1, c);
                }
            }
            if(c<maze[0].length-1)
            {
                if(maze[r][c+1]!=WALL&&maze[r][c+1]!=STEP)
```

```
            {
               counter += 1+markAllPathsAndCountStars(r, c+1);
            }
         }
         if(c>0)
         {
            if(maze[r][c-1]!=WALL&&maze[r][c-1]!=STEP)
            {
               counter += 1+markAllPathsAndCountStars(r, c-1);
            }
         }
      }
      return counter;
   }

   private boolean displayTheCorrectPath(int r, int c)
   {
      boolean bool = false;

      if(r<0||r>maze.length-1||c<0||c>maze[0].length-1)
         bool = false;
      else
      {
         if(maze[r][c] == WALL)
            bool = false;
         else  if(maze[r][c] == STEP)
            bool = false;
         else if(maze[r][c]==EXIT)
            return true;
         else
         {
            maze[r][c]=STEP;
            if(displayTheCorrectPath(r-1, c)||displayTheCorrectPath(r+1, c)||displayTheCorrectPath(r, c-1)||displayThe(
               bool = true;
            maze[r][c]=PATH;
         }
      }

      if(bool)
         maze[r][c] = STEP;

      return bool;
   }

   private boolean displayCorrectPathAndCountStars(int r, int c, int count)
   {
      boolean bool = false;

      if(r<0||r>maze.length-1||c<0||c>maze[0].length-1)
         bool = false;
      else
      {
         if(maze[r][c] == WALL)
            bool = false;
         else  if(maze[r][c] == STEP)
            bool = false;
         else if(maze[r][c]==EXIT)
         {
            System.out.println("Took " + count + " steps.");
            return true;
         }
         else
         {
            maze[r][c]=STEP;
            if(displayCorrectPathAndCountStars(r-1, c,count+1)||displayCorrectPathAndCountStars(r+1, c,count+1)||displa
               bool = true;
            maze[r][c]=PATH;
         }
      }

      if(bool)
         maze[r][c] = STEP;

      return bool;
   }

   //This is for testing purposes. Do not change or remove this method.
   public char[][] getMaze()
   {
```

```
        return maze;
    }
}
```