**CS 210: Algorithms and Data Structures   
Maze Project (100 points) – Part 3 of 3  
Due: Thursday, March 27th, 2014**

**Project Description:**

All classes for this project are in package maze. You are welcome to use your previous codes as well as incorporating Java library and of course writing several new ones. Note that this project does not explicitly list all of the classes you'll have to write--part of that will be design decisions left up to you.

**The Solver and GUI**

To handle all the fiddly graphical stuff, write a MazeApp class. When the program is started, it should present the following widgets: a text field for messages, a button labeled "Load", a button labeled "Start” and an animation speed control.

When the user loads a maze file through the "Load" button, the maze in that file should be read in and displayed on the applet as follows: open squares are white, walls are black. The start button should initiate the maze solution process. Once initiated, the solver proceed as an animation that uses a Timer (see below) to set the solution running at an adjustable rate of a few steps a second. Make sure that your display is big enough to fit a 25 x 25 x 10 maze.

What we should see as the solution process goes on is that the squares that have been explored---yes, we can reach them, but no, we haven't found the finish yet---change from white to medium-grey. We should be able to see the maze slowly shaded in until the solver reaches the finish (or determines that it's unreachable), make sure that there is a sufficient delay in the animation to see the path that it takes while solving the maze.

Somewhere conspicuous, the GUI should display a status line on the text field containing the current status of the solution process: “No maze”, “Maze loaded”, “Solution in progress“, “Solution complete: finish reachable at X,Y,Z in N moves”, “Solution complete: finish not reachable”. Make sure that your program is capable of loading, displaying and solving multiple mazes without having to restart/rerun the program.

**IF** you choose to deviate and/or append from the specifications make sure that it is clearly explained and identifiable during the demo. You can add other GUI components to add features the program but make sure it is not over complicating the program.

**Notes on Timer**

You might need to use the two Timer classes in the java libraries, they have similar functionality. You would probably want the one in javax.swing, because the way it triggers its action is exactly the same way a Button does---via an ActionListener. Once the Timer is created, you will call its start(), stop(), and isRunning() methods to control it. The one you don't want to use is in java.util. If you are freely importing everything from that package, make sure you're not accidentally using the wrong Timer!

**Deliverables:**

* **There will be three (3) assignments in MUOnline.**
* Each team must submit the complete maze package using MUOnline.
* Each team must submit a UML diagram that shows Class properties and relationships.
* Everyonesubmit one **valid** maze text file to MUOnline (no extra characters or unrecognized characters, etc) with a maximum size of 25 x 25 x 10. Make sure to also enter the total count for the shortest path as a note in your assignment. Filename must be formatted: lastname-maze.txt.
* Everyone will submit an individual teammate assessment that discusses division of work, teammate performance, teammate contribution, etc. (🡪 this is by email)

**Grades:**

You are to show that your program have completed the three main components. Although artistic value does not warrant a higher grade, intuitive and ease of use does have an impact for part 3.

Grades are calculated as follows:

* Completion of Part 1 = 15 pts.
* Completion of Part 2 = 50 pts.
* Completion of Part 3 = 35 pts.
* Individual assessments will be used to *adjust* the individual grade for the project.

**BONUS:**

* Team(s) that develops a shortest path algorithm will receive a 15 points bonus. **HOWEVER**, shortest path algorithms are often slow and inefficient. To receive credit the team must have a conventional solver algorithm and a shortest path algorithm that can solve a problem in a ***reasonable*** time. It is highly suggested that you have a regular and a shortest path algorithm implemented and make it interchangeable.
* Every student that comes up with a valid maze that cannot be properly solved by ALL maze solver in the class (excluding their own) will receive a 10 points individual bonus.

**Have fun with the project! You can be as creative as you want BUT make sure that the program satisfies all the requirements**!