**Maze Solver (alternate version)**

*RFrank.net, copyright © 2010 by Asylum Computer Services LLC*

From the earliest days when mazes were on cereal boxes, I have always liked to solve mazes. You will write a program to solve a maze using a computer using the Stack data structure. The algorithm for solving a maze is easy: go everywhere until you find the exit. The "go everywhere" part is straightforward if you follow a few simple rules: always go left if you can, otherwise go straight, otherwise go right. If you can't go anywhere, back up until you can go somewhere. The "back up" part implies a stack of previous locations. When you find the exit, the stack has the path.

It turns out the challenging part of this lab is dynamically creating a maze to solve. Every cell must be connected, yet there can only be one path through the maze. The code to generate a maze is provided for you to study and potentially modify as an extension to the project.

**Assignment**

This project follows the Model-View-Controller (MVC) model. You are provided with the Controller, [Maze.java](file:///C:\Users\bunnb\Dropbox\Work\00%20Courses\04%20CS%20III\Lab___-MazeSolver%20(alternate%20version)\Maze.java) and the View, [MazeView.java](file:///C:\Users\bunnb\Dropbox\Work\00%20Courses\04%20CS%20III\Lab___-MazeSolver%20(alternate%20version)\MazeView.java). You are also provided with a skeleton Model component, [MazeModel.java](file:///C:\Users\bunnb\Dropbox\Work\00%20Courses\04%20CS%20III\Lab___-MazeSolver%20(alternate%20version)\MazeModel.java). Of the three, you will make modifications in the model component only. To support your work, the [ListStack.java](file:///C:\Users\bunnb\Dropbox\Work\00%20Courses\04%20CS%20III\Lab___-MazeSolver%20(alternate%20version)\ListStack.java) file and the [Stack.java](file:///C:\Users\bunnb\Dropbox\Work\00%20Courses\04%20CS%20III\Lab___-MazeSolver%20(alternate%20version)\Stack.java) file are provided, along with a suggested [MazeCell.java](file:///C:\Users\bunnb\Dropbox\Work\00%20Courses\04%20CS%20III\Lab___-MazeSolver%20(alternate%20version)\MazeCell.java) class that is consistent with the view component. Downloading all these mentioned files to a subdirectory should allow you to generate a maze. You will have to write the code to solve it.

In the model component, a constructor is provided that generates the maze. It also calls the updateView method of the view component to show the unsolved maze. You will have to add fields to the MazeModel class, along with initialization code in the constructor that sets it up to be solved by a series of calls by the controller to stepMaze.

The work of solving the maze is written in the stepMaze method. Several times a second, the controller will call this method to advance the solution. At the end of the stepMaze method, be sure to call updateView to show your progress.

The final result is to start with a blank maze and solve it, as shown below. Yellow represents all the paths that were traversed. Blue, of course, represents the solution.

