# Oh, no! Minotaur!

Daniel Yee
CIS 2168

## Purpose

In this assignment, you will apply your knowledge about stacks to find the path through a labyrinth. It will also introduce you to the concept of recursion, which we'll study next, and some more simple Java graphics.

## Background

In Greek mythology, the Labyrinth was an elaborate and confusing maze created by Daedalus for King Minos of Crete. The Labyrinth was built to hold the Minotaur so it wouldn't escape.

Every nine years, King Minos demanded, as war tribute, seven Athenian boys and seven Athenian girls to be sent to Crete to be devoured by the Minotaur. Theseus, takes the place of one of the boys to stop the Minotaur. To keep from getting lost, he uses a ball of string to keep track of his way through the maze. Eventually, he finds the Minotaur, kills it, and escapes the Labyrinth.

## The Problem

Several files have been provided to generate the maze. Your assignment is to create a basic labyrinth solving algorithm to find your way from the top-left corner of the maze to the bottom-right corner. You need to finish implementing `solveMaze()` to perform a depth-first search using a stack.

A depth-first search algorithm is straight-forward. If you are able, make a choice in which direction to travel. Follow that path and continue making direction choices as needed. If you reach a dead end, backup until you find a new route to explore.

The algorithm can be summarized as follows

```
Push start position onto stack
While not finished exploring maze and stack isn't empty
        Determine our current position
        If we can go north and haven't visited there yet
                Push the location onto the stack
                Mark the current location as visited
        else if we can go south and haven't visited there yet
```

….
else if we can go east and haven't visited there yet
…
else if we can go west and haven't visited there yet
…
else
we're at a dead-end
mark current as a dead-end
pop off the stack

There are numerous videos online that explore depth-first search for graph traversals (which we'll talk about in Chapter 10). However, each labyrinth cell is really a graph node without links so the overall concepts still apply.

# The Code

## `Maze` Class

This class contains the `main` method. It instantiates a `MazeGridPanel` with parameters that determine the size of the labyrinth – anything above 100 x 100 tends to have slow performance.

## `MazeGridPanel` Class

This is the actual labyrinth, stored in a 2D array of `Cell` objects called `maze`.

Implement your labyrinth solving algorithm in the `solveMaze()` method. `genenerateMazeByDFS()` is extra credit. The `visited()` will check if the `Cell` has been visited. `generateMaze()` is the method that actually creates the labyrinth.

## `Cell` Class

The labyrinth is composed of individual grid units, each represented by a `Cell`. Each cell has a `boolean` field for each of the four possible walls it can have as well as a `row` and `col` field to identify its location in the labyrinth.

The color of the `Cell` indicates whether we've previously visited a `Cell` or not. White and red cells are unvisited (red indicates the labyrinth exit). Any other color indicates that it has been visited. The color of the `Cell` can be set using the `setBackground` method while the color of a `Cell` can be retrieved by `getBackground()`.

# Extra Credit

Complete `genenerateMazeByDFS()` which will build a labyrinth using depth-first search. The algorithm can be found on Wikipedia.

Implement a better labyrinth solving algorithm than the one presented above.

# Rubric

**90 points**

Your implementation correctly solves the labyrinth.

**45 points**

Your implementation only partially solves the labyrinth.

**10 points**

Code is neat and properly indented.

**5 points extra credit**

Implement the `genenerateMazeByDFS()` method to generate the labyrinth using depth-first search.

**3 points extra credit**

Implement a better labyrinth solving algorithm.

# Presentation

You will only receive credit for the assignment if you present your project.