CS151 - Prelab №3

James Capuder, Oberlin College February 23, 2015

2. Write a toString method that returns the appropriate character (in this case, write actual code rather than pseudocode). This is a perfect time to use a switch() statement [Weiss 1.5.8], although nested if-then-elses would also work.

```
public String toString(int squareCharacter){
    if (squareCharacter == 0){
    return "_"
    } else if (squareCharacter == 1){
    return "#"
    } else if (squareCharacter == 2){
    return "S"
    } else{
    return "E"
    }
}
```

4. Draw the contents of stack and queue data structures.

See attached

5, 6. Run the algorithm

See attached

7.What are the benefits of having the mazeSolver as an abstract class? Why not just make two different versions? Are we really saving anything by using the single superclass with two subclasses?

The abstract class makes sense in this situation because both MazeSolverStack and MazeSolverQueue share all their functionalities, and only differ by the form of data structure they use. Making two more simple classes will give you less trouble than one complex one, and the abstract classes can be applied in a more general way and keeps code more organized.

8. Suppose the mazeSolverStack class' stack is called stack. What will the void add(square sq) method look like?

This method would just add the square to the stack.

10. Write a pseudocode for the retracePath method

Check most recent square in stack.
Print Square
Keep printing squares until you reach the start square, unless a square you encounter leads to a different path, in that case check surrounding squares for a marked one.