**CS 210: Algorithms and Data Structures   
Maze Project (100 points)**

**Module 1 Due: *Feb. 20, 2014***

Consider your basic multi level maze. It has walls and pathways, and it has one starting point and one (or more) finish point(s). Furthermore, one wall is just like another, and any open spaces are also identical. Your program will read a maze stored in a text file. Each maze component has a one-character to represent it:

|  |  |
| --- | --- |
| **Walls** | **# (hash mark)** |
| **Open spaces** | **. (period)** |
| **Start** | **@ (at sign)** |
| **Elevator** | **+ (plus sign)** |
| **Finish** | **\* (asterisk)** |

The format will be that the maze will always be in a square form, where subsequent lines each contain one row of the maze with each character representing one square of the maze. Maze levels will be separated by a horizontal dash and connected by stairs. An elevator will have corresponding elevators in the same location at various levels, bear in mind that an elevator may connect several levels but not every level.

**Program requirements**:

1. Create a class called **MazeReader** that can store the logical layout of a maze.

* **MazeReader** should have a constructor that can read from a Scanner.
* Your program must be modular, do not put everything into one method.
* Utilize enumerated data type that allows for a more flexible checking.
* Write a ***toString()*** method that makes a String representation of the entire Maze in the same format as the input.
* Be sure to include a ***check(x,y,z)*** method that returns a one-character-long string containing only the character corresponding to a location.
* Properly comment your methods, variables and other components.
* Perform error checking for both methods.

1. Create a test class **MazeReaderTest** with the following menu options:
2. Load a maze (filename)
3. Return element at specific location (x,y,z)
4. Print entire maze
5. Exit Program

* Perform error checking for the menu option and handle it accordingly.
* Remember that some of the error in the menu might already be handled by the **MazeReader** class.

1. Draw a UML diagram for your Maze project.

**Program requirements**:

1. What you have to submit:
   1. Your UML diagram.
   2. Your Java programs.
2. How to submit your work
   1. Copy the items mentioned in 2) into your folder under \\cistecs\cs210\

Here are some simple examples of the maze with one, two and three levels:

|  |  |  |
| --- | --- | --- |
| ##########  #...#....#  #\*#....#@#  #.#.#..#.#  ########## | #######  #...#@#  #+#...#  #######  ------- #######  #...###  #+#.\*##  ####### | #######  #+...@#  #######  -------  #######  #+...+#  #######  ------- #######  #\*...+#  ####### |

Slightly more complex examples:

|  |  |  |
| --- | --- | --- |
| ############  #.#........#  #.#.######.#  #.#....#...#  #.###.+#.#.#  #...####.#.#  #.#.#..#.#.#  #.#.#.##.#.#  #@#......#.#  ############  ------------  ############  #.#........#  #.#.######.#  #.#....#...#  #.###.+#.#.#  #...####.#.#  #.#.#..#.#.#  #.#.#.##.#.#  #@#......#\*#  ############ | ############  #.#........#  #.#.######.#  #.#....#...#  #.###.+#.#.#  #..\*####.#.#  ------------  ############  #.#.#..#.#.#  #.#.#.##.#.#  #@#......#.#  #.#......#.#  ############  ------------  ############  #.#........#  #.#.######.#  #.#....#...#  #.#....#...#  ############  ------------  ############  #.#....\*...#  #.#.######.#  #.#....#...#  #.#...+#...#  ############ | ############  ##@........#  ###..+####+#  ############  ------------  ############  ##....#+...#  ###+.+####+#  ############  ------------  ############  ##+...#+...#  ###+..####\*#  ############  ------------############  ##+.+.#....#  ###...####.#  ############  ------------############  ##..+.######  ###...######  ############ |