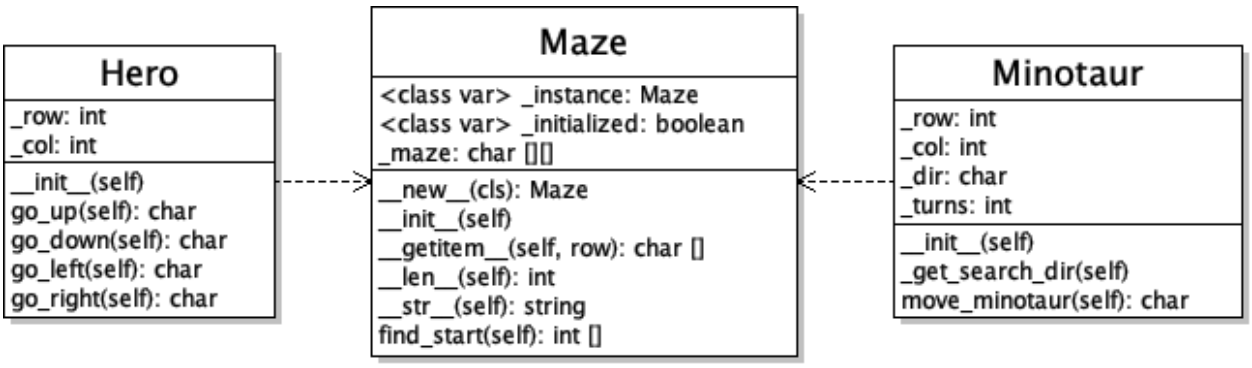
CECS 277 – Lab 10 – Singleton

Minotaur's Maze

Create a program that allows the user to wander through a maze that is guarded by a minotaur. The user wins if they reach the maze's exit without being captured by the minotaur. Use the following UML diagram and the class descriptions below to create your program:



Classes:

1. Maze – singleton – the minotaur's maze.
   a. __new__(cls) – if the maze hasn't been constructed, then construct it and store it in the instance class variable and return it. If it has, then just return the instance.
   b. __init__(self) – create and fill the 2D list from the file contents.
   c. __getitem__(self, row) – overloaded [] operator – returns the specified row from the maze. (Note: this operator can be used to access a row (ex. m[row]) or can be used to access a character at a row and column (ex. m[row][col])).
   d. __len__(self) – returns the number of rows in the maze. (ex. to get the number of rows, use: num_rows = len(m), for columns, use: num_cols = len(m[row])).
   e. __str__(self) – returns the maze as a string in a grid format
   f. find_start(self) – returns the location of the start ('s') in the maze as a two-item 1D list (ex. [row, col]).
2. Hero – the player's character
   a. __init__(self) – sets the hero's starting location by setting the row and col attributes to the result of Maze's find_start. Place an 'H' in the maze at that spot.
   b. go_up/go_down/go_left/go_right(self) – update the hero's location by adding or subtracting 1 to the row or col attribute, but only move there if it is not a wall ('*'). Overwrite the old 'H' with a space and place a new 'H' at the updated position. Return the character that was at that location so that main knows if the user ran into the minotaur, the finish, or a wall.
3. Minotaur – the monster that guards the maze.
   a. __init__(self) – randomizes the starting location of the minotaur to any blank space in the maze. Place an 'M' at that spot. Set the turns to 0.
   b. _get_search_dir(self) – chooses the direction that the minotaur will move in. Set the direction attribute using the chosen value (ie. 'U', 'D', 'L', 'R' (or similar)).
   c. move_minotaur(self) – every third turn the minotaur will detect the hero's location in the maze and update its direction by calling _get_search_dir. Using the determined direction,

update the minotaur's location (row and col attributes) if the space is not a wall ('*') or the finish ('f'). Overwrite the old 'M' with a space and place a new 'M' at the updated position. Return the character that was at that location so that main knows whether the minotaur captured the hero.

Main – construct the Maze, Hero, and Minotaur objects. Display the maze and prompt the user to enter a direction to move the Hero in. Move the Hero, if the Hero reaches the finish, then the user wins, if the hero runs into the Minotaur, then they lose, if the Hero hits a wall or moves to an empty space, then the Minotaur moves. If the Minotaur captures the Hero, then the user loses the game.

Example Output:

```
* * * * * * * * * * * * * * * * * * * * *
* H                       *     M         *
*   * * *   *   * * *   *   *   * * *   *
*   *       *       *       *           *
*   *   * * *   *   * * *   *   *       *
*       *       *       *   *   *       *
* * *   *   * * *   *   *       *   *   *
*           *       *   *   *       *   *
*   *               *       *   *   *   *
*   * * * * *   *   * * *   *   *   *   *
*       *       *                   *   *
*   * * * * * *   *   * * * * *   * * *   *
*               *       *               f *
* * * * * * * * * * * * * * * * * * * * *
Choose a Direction (WASD):x
Choose a Direction (WASD):d
```

```
* * * * * * * * * * * * * * * * * * * * *
*   H                       *   M         *
*   * * *   *   * * *   *   *   * * *   *
*   *       *       *       *           *
*   *   * * *   *   * * *   *   *       *
*       *       *       *   *   *       *
* * *   *   * * *   *   *       *   *   *
*           *       *   *   *       *   *
*   *               *       *   *   *   *
*   * * * * *   *   * * *   *   *   *   *
*       *       *                   *   *
*   * * * * * *   *   * * * * *   * * *   *
*               *       *               f *
* * * * * * * * * * * * * * * * * * * * *
Choose a Direction (WASD):w
You ran into a wall!
```

```
* * * * * * * * * * * * * * * * * * * * *
*   H                     * M             *
*   * * *   *   * * *   *   *   * * *   *
*   *       *       *       *           *
*   *   * * *   *   * * *   *   *       *
*       *       *       *   *   *       *
* * *   *   * * *   *   *       *   *   *
*           *       *   *   *       *   *
*   *               *       *   *   *   *
*   * * * * *   *   * * *   *   *   *   *
*       *       *                   *   *
*   * * * * * *   *   * * * * *   * * *   *
*               *       *               f *
* * * * * * * * * * * * * * * * * * * * *
Choose a Direction (WASD):d
```

```
* * * * * * * * * * * * * * * * * * * * *
*     H                       *           *
*   * * *   *   * * *   * M *   * * *   *
*   *       *       *       *           *
*   *   * * *   *   * * *   *   *       *
*       *       *       *   *   *       *
* * *   *   * * *   *   *       *   *   *
*           *       *   *   *       *   *
*   *               *       *   *   *   *
*   * * * * *   *   * * *   *   *   *   *
*       *       *                   *   *
*   * * * * * *   *   * * * * *   * * *   *
*               *       *               f *
* * * * * * * * * * * * * * * * * * * * *
Choose a Direction (WASD):d
...
```

```
* * * * * * * * * * * * * * * * * * * * *
*                       *               *
*   * * *   *   * * *   *   *   * * *   *
*   *       *       *       *           *
*   *   * * *   *   * * *   *   *       *
*       *       *       *   *   *       *
* * *   *   * * *   *   *       *   *   *
*           *       *   *   *       *   *
*   *               *       *   *   *   *
*   * * * * *   *   * * *   *   *   *   *
*       *       *                   *   *
*   * * * * * *   *   * * * * *   * * * H *
*               *       *           M f *
* * * * * * * * * * * * * * * * * * * * *
Choose a Direction (WASD):s
```

```
* * * * * * * * * * * * * * * * * * * * *
*                       *               *
*   * * *   *   * * *   *   *   * * *   *
*   *       *       *       *           *
*   *   * * *   *   * * *   *   *       *
*       *       *       *   *   *       *
* * *   *   * * *   *   *       *   *   *
*           *       *   *   *       *   *
*   *               *       *   *   *   *
*   * * * * *   *   * * *   *   *   *   *
*       *       *                   *   *
*   * * * * * *   *   * * * * *   * * *   *
*               *       *           M H *
* * * * * * * * * * * * * * * * * * * * *
You found the exit!
```

Notes:

1. You should have 4 different files: main.py, minotaur.py, hero.py, maze.py.
2. Use docstrings to document each of the classes, their attributes, and their methods.
3. Place your names, date, and a brief description of the program in a comment block at the top of your main file. Place brief comments throughout your code.
4. Do not create any extra methods, parameters, attributes, or modify the class structure.
5. Please do not create any global variables (besides the singleton maze) or use attributes globally (ie. do not access any of the attributes using the underscores).
6. The Maze is a singleton, that means that whenever and wherever you need to use one of its methods, just construct an instance (which will always be the same instance).
7. The Minotaur's _get_search_dir method should determine the new direction in any way you'd like. You might want to start off by choosing a random direction (up, down, left, right), and then later, refining your method to track the user by searching for the 'H' in the maze and then heading toward that location by using a few if statements. Hint: try not to let it get stuck on a wall, if its direction is into a wall, choose a random direction.
8. Thoroughly test your program before submitting:
    a. Make sure that the Maze class is a singleton so that only the one instance is ever used throughout the program.
    b. Make sure that the maze is read in correctly from the file and stored in the 2D list.
    c. Make sure that when you're reading in the maze it can work for any size maze (not just the one provided).
    d. Make sure that the Hero begins the game at the starting location ('s').
    e. Make sure that the Minotaur and the Hero cannot move out of bounds of the maze or walk through walls. The Minotaur also cannot move onto the finish ('f').
    f. Make sure that the Hero can move in any direction within the maze.
    g. Make sure that the Minotaur starts at a random space in the maze (' ').
    h. Make sure that when the Hero or Minotaur move, their old mark is removed and replaced with a space and a new mark is placed at their updated location.
    i. Make sure that the Minotaur resets its direction on every third turn.
    j. Make sure that the game ends when the finish is reached or when the Hero is captured.