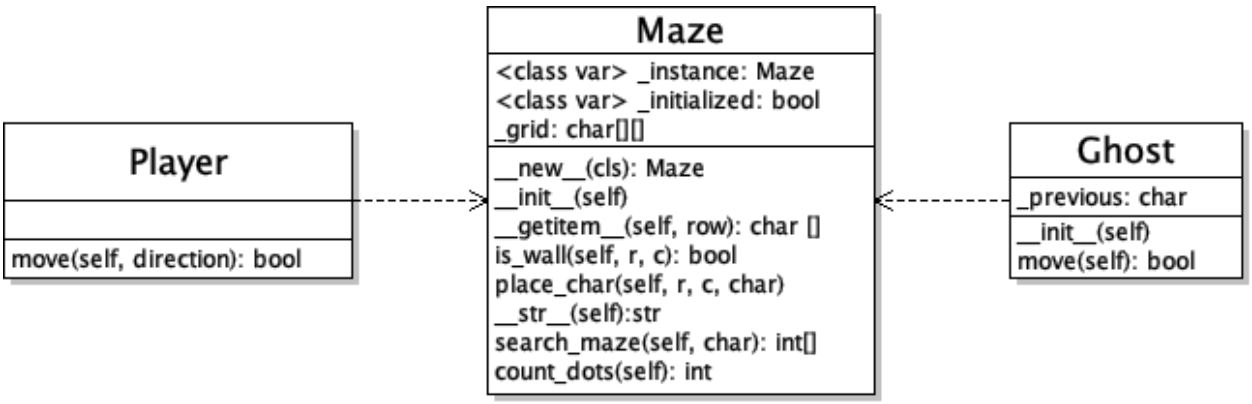
CECS 277 – Lab 9 – Singleton

Pac-Man

Create a program that allows the user to explore a maze that is guarded by a ghost. The user wins if they can eat all of the dots in the maze without being captured by the ghost. Use the following UML diagram and the class descriptions below to create your program:



Classes:

1. Maze – a singleton – 2D grid of characters.
   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 grid 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. is_wall(self, r, c) – if the item at row r and col c is a '*', return True, otherwise False.
   e. place_char(self, r, c, char) – overwrite the character at row r and col c with the new char.
   f. __str__(self) – returns the contents of the maze as a string in a grid format.
   g. search_maze(self, char) – returns the location of the first occurrence of the character in the maze as a two-item 1D list (ex. [row, col], or [-1,-1] if not found).
   h. count_dots(self) – counts and returns the number of dots '.' in the grid.
2. Player – the player's character.
   a. move(self, direction) – pass in the player's chosen direction (w-up, a-left, s-down, d-right) and access the Maze. Update the player's location by finding the 'P' in the maze, overwrite it with a space, then add or subtract 1 to either the row or col for the appropriate direction. Check to make sure the new location is not a wall '*', if it isn't, overwrite the new location with an 'P', if it is, then put the player back where they were. Return true if the player ran into the ghost, false otherwise.
3. Ghost – the guardian of the maze that chases the player.
   a. __init__(self) – initializes the previous attribute with a '.'.
   b. move(self) – access the Maze. The ghost will detect the player's location in the maze and then move one space toward that direction. Check that the space is not a wall '*', if it is, then repeatedly choose a random direction until it isn't. Overwrite the old 'G' with a

character stored in the previous attribute and place a new ‘G’ at the updated position. Return true if the ghost ran into the player, false otherwise.

Main – construct the Maze, Player, and Ghost objects. Display the maze and prompt the user to enter a direction to move in (wasd). Move the player, if there are no more dots in the maze, then the player wins, if the player runs into the ghost, then they lose. After the player moves, then it is the ghost’s turn to move. If the ghost runs into the player, then the player loses. If the player runs into a wall, the ghost still gets a turn to move.

Example Output:

```
**********
*........*
*........*
*..*..*..*
*..*.G*..*
*..*..*..*
*........*
*..****..*
*...P....*
**********

Move (WASD): d
**********
*........*
*........*
*..*..*..*
*..*..*..*
*..*.G*..*
*........*
*..****..*
*... P...*
**********

Move (WASD): d
**********
*........*
*........*
*..*..*..*
*..*..*..*
*..*..*..*
*....G...*
*..****..*
*...  P..*
**********

Move (WASD): d
**********
*........*
*........*
*..*..*..*
*..*..*..*
*..*..*..*
*.....G..*
*..****..*
*...   P.*
**********

Move (WASD): d
```

```
**********
*........*
*........*
*..*..*..*
*..*..*..*
*..*..*..*
*....G...*
*..****..*
*...    P*
**********

Move (WASD): w
**********
*........*
*........*
*..*..*..*
*..*..*..*
*..*..*..*
*...G....*
*..****.P*
*...     *
**********

Move (WASD): w
**********
*........*
*........*
*..*..*..*
*..*..*..*
*..*..*..*
*....G..P*
*..****. *
*...     *
**********

Move (WASD): a
**********
*........*
*........*
*..*..*..*
*..*..*..*
*..*..*..*
*.....GP *
*..****. *
*...     *
**********

Move (WASD): a
```

```
**********
*........*
*........*
*..*..*..*
*..*..*..*
*..*..*..*
*.....P  *
*..****. *
*...     *
**********

You ran into the ghost!  Game over...
```

-or-

```
**********
*........*
*........*
*..*..*..*
*..*..*..*
*..*..*..*
*......GP*
*..****. *
*...     *
**********

Move (WASD): d
**********
*........*
*........*
*..*..*..*
*..*..*..*
*..*..*..*
*.......G*
*..****. *
*...     *
**********

The ghost caught you!  Game over...
```

-or-

```
**********
*.       *
*P       *
* G*  *  *
*  *  *  *
*  *  *  *
*        *
*  ****  *
*        *
**********

Move (WASD): w
**********
*P       *
* G      *
*  *  *  *
*  *  *  *
*  *  *  *
*        *
*  ****  *
*        *
**********

All dots eaten.  You win!
```

Notes:

1. You should have 5 different files: main.py, ghost.py, player.py, maze.py, pacman.txt.
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 a new instance (which will always be the same instance).
7. The Ghost's move method can determine the movement in any way you'd like as long as it moves toward the player. Search for the 'P' in the maze and then calculate and move in the direction of that location by using a few if statements. Note: don't let it move into a wall, if its chosen direction is into a wall, choose a random direction instead.
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 Player and the Ghost begin the game at their starting locations defined in the file.
e. Make sure that the Player and the Ghost can move in any direction within the bounds of the maze and cannot walk through walls.
f. Make sure that when the Player or Ghost move, their old mark is removed and replaced at their updated location.
g. A Player's old location is always replaced with a space. A ghost's old location is replaced with whatever was previously there (either a space or a dot, which is stored in the previous attribute). The Ghost does not eat any dots, only the Player eats dots.
h. Make sure that the game ends when all of the dots have been eaten, or when the ghost catches the player (either by the player walking into the ghost, or the ghost walking into the player).
i. Make sure that the Ghost moves every valid turn (ie. if the player walks into a wall, then the ghost still moves, but if the user enters an invalid input, then it does not move).

Pac-Man Rubric – Time estimate: 4 hours

Pac-Man **10 points**	Correct. 2 points	A minor mistake. 1.5 points	A few mistakes. 1 point	Several mistakes. 0.5 points	No attempt. 0 points
Maze class (in a separate file)**:** 1. Singleton: has 2 class variables. 2. new checks if instantiated and always returns the same instance. 3. init method checks if initialized and reads in maze from file if not. 4. Overrides getitem, str. 5. Has is_wall, place_char, search_maze, and count_dots methods.					
Ghost class (separate file)**:** 1. init initializes previous to '.'. 2. move method moves ghost one space toward the player and returns true if player is caught. 3. Removes 'G' from old position and re-adds it at new position & replaces old position with previous character. 4. Does not move ghost into walls. 5. Accesses singleton correctly.					
Player class (separate file)**:** 1. move method moves the player one space in the specified direction and returns true if it ran into the ghost. 2. Removes 'P' from old position and re-adds it at the new position. 3. Accesses singleton correctly.					
Main file (in separate file)**:** 1. Creates Maze, Player, and Ghost. 2. Repeatedly prompts player to move. 3. Correctly moves player and ghost and displays the updated maze. 4. Error checks user input. 5. Ends game when player eats all dots or is captured by the ghost.					
Code Formatting: 1. Correct spacing. 2. Meaningful variable names. 3. No exceptions thrown. 4. No global variables (other than the singleton) or uses attributes directly. 5. Correct documentation.					