**Computer Science 220/220L**

**Laboratory 12 / Homework 12**

**Hangman**

**Learning objectives:**

Develop a Python program that:

* Reads from a file
* Uses classes and methods
* Uses decision and repetition structures to solve a problem

See the second page for a complete description of the classes you need to write. There is a starter set in HW12.py.

**Part I: Assignment:**

Write a program to support the children's spelling game hangman. Call the program **hangman.py**. Since there are many versions of hangman, be sure that you implement exactly the following.

The player is to guess the letters in a secret word. Use underscores to display the number of letters in the word. When the player guesses a correct letter, display the word with that letter showing. Correct guesses don't count against the player. Incorrect guesses count, and the player loses on the seventh incorrect guess. Each time the player is asked to enter a letter, the program should display how many guesses they have left and a list of all of the incorrect letters they have guessed. If the player accidentally chooses a letter that has already been guessed, this should not count as a guess. See sample games below.

Your program must contain at least the following classes. The methods are suggested for you to use. You may create your own if you like. Just make sure that the function is contained in one of the classes. (If you want you may use other functions as well.)

* Write a class, Words(file\_name), that reads a list of words read from a file and contains a method, get\_word(), to supply and remove a random word from the list. Call the file **wordlist.txt.** Use your own small file with known words to test your program.
* Write a class, Guess(word), that has a method \_\_str\_\_ method that returns a string for the word at any given point. (E.g. If the word is “slant” and the user has guessed the letters ‘t’ and ‘a’, this method should return “\_ \_ a \_ t” string with spaces between the letters or \_’s for readability). There should also be a method, gameover(), to return a Boolean, True for end of game and False for continue the game, to indicate whether the game is over. There should also be a method for guess\_letter(letter) to enter guesses when the Enter button is clicked. This method should also count the number of bad guesses. When the letter is in the word, it replaces \_ with the letter in each position of the letter. Hint: changing your word string into a list makes editing the letters easier. This method will also return a 0 if the letter has already been guessed, a 1 if the letter is not in the word and a 2 if the letter is in the word.
* Use the button class to create all button objects for Yes, No and Enter
* Create a function in main that plays the game once. The main should handle playing the game again. You can close the graphics window and start fresh with the graphics but the list of words should not be refreshed.
* Write a Noose(win) class to display the losses graphically. Write a method, wrong() to cause the next section to be drawn.

**Part II: User Interface**

Once you get your functions working with text, you can add a graphical user interface of your design. The GUI should have the following elements:

* A text area where the word with underscores representing the letters of the word is presented.
* A text area where the letters used are displayed. Optionally you can display the letters not chosen yet. In either case you must identify which role the letters displayed serve.
* A text area indicating the number of guesses the user has remaining.
* An area where the user enters the letter s/he wishes to play and an “enter” button.
* When a game completes, a message should appear in the GUI window asking the player whether they want to play another game. The GUI should also provide **Yes** and **No** buttons for the player to make a choice.

**Once your basic program is working:**

The traditional way to play hangman is to draw a figure on a gallows. Another part of the hanged man is added for each incorrect guess. When the figure is complete, the player loses. How you draw the figure is up to you, but it should have exactly seven parts. This drawing should be done by another class, Noose, that contains a method, wrong(), that draws another part of the seven each time the method is called. When it is called the 7th time, the hanging is complete.

List of classes and methods needed for your lab/homework:

Homework12.py contains Guess, Button and Words classes.

class Guess

method \_\_init\_\_ (word) - creates the object

method missed () - returns the word string that could not be guessed

method guess\_letter (letter) - uncovers letters that match the guess, counts the bad guesses and keeps track of the letters guessed. It returns a number, 0, if the letter has already been guessed, 1 if the letter is NOT in the word and 2 if the letter is in the word

method gameover () - returns Boolean, T if word is guessed or the number of guesses has exceeded the limit and F otherwise.

method num\_of\_guesses() – returns the number of guesses used so far

method \_\_str\_\_ () - returns a string with the letters in the word and \_ for each unguessed letter separated by spaces

method letters\_guessed () - returns a string, in alphabetical order, of all the letters that have been guessed

class Button

method \_\_init\_\_ ( win, center, width, height, label, color) - Creates a button of that weight, height, label and color

method clicked (p) – did the button get clicked True, otherwise, False.

method activate () – draws the button and allows the button to be clicked

method deactivate () – undraws the button and prevents the button from being clicked

class Words:

method \_\_init\_\_ (filename) - creates the list of words object

method get\_word () - returns a randomly selected word

class Noose:

method \_\_init\_\_ ( win) – creates a Noose object with 7 sections that can be drawn one at a tim

method wrong (sect\_num) - draws the sect\_num section of the platform and/or the hanged figure of the 7 sections

function play\_one\_game () - plays one game of hangman

**Submission:**

This lab will count as two homework grades for CSCI220. You should complete the assignment and submit your finished product on or before the date specified in OAKS.