HANGMAN

Documentation by Enrico Miguel Parallon

This doc will describe my hangman game and how it works. This project of mine started at October 1, 2023 with many assignments while the Intramurals was going. I hope to explain my project as simple as I can, but do note that this is my first time doing this.

**Rules of Hangman**

The objective of the game is to correctly answer the hidden word through guessing by inputting a letter that you think the word contains. Ex. Typing “e” while the word is “TIME”, will reveal the letter “E”(but keeps the other letter hidden until you correctly guess them).

**The Structure of the program**

The program is divided into 5 separate classes that each do a specific task. Each class are named by the main tasked that they are assigned at. Generally, they purposed as such

* **MainHM**
  + This is the main “Brain” of the program, this dictates the flow and structure of the game.
  + Player input is gathered here.
* **wordlist**
  + Acts as the “word bank”, randomly picks a word from the wordBank array, then returns that chosen word.
* **outputHM**
  + This is an algorithm that simply takes the chosen word(PICKword) and prints it out but with a certain criteria.
* **processHM**
  + The algorithm that does two important tasks.
    - First, “proc()” is responsible for individually checking each of the letter of PICKword with the player’s answer. Every time a positive match is found, the proc will switch the letter to a lowercase state, signaling a correct guess.
    - Second, “proc2()” will follow the same individual checking, but will instead return a true Boolean if a positive match is found. However, if a false match is found then it will call the HangMan function from the gameLogic class.
* **gameLogic**
  + It is responsible for managing the programs game condition. It will check for certain conditions to decide whether the player has won or made a wrong guess. If the player lose all 6 tries(as seen by the Hang man stick) then the game ends.

**CASE states.**

The program keeps track of the player’s progress by reading whether the individual letters, the PICKword variable, are uppercase or lowercase. When a letter is in uppercase, then it is considered as “not solve” but if it switches into a lowercase state then the game reads it as solved – this is observed by whether the game prints out a letter(solved state) or an underscore(un-solved state).

**Author’s note**

You may use study this program, change this program, or do whatever you like. You may delete it if you deem it necessary, but I hope nga dili n animo buhaton kay gusto nako nga maka gamit pa ang next students. If you decide to change the program, create a copy and change it in there, try to keep the original file unchanged as this can be used by other students or teachers by their own discretion.

Most importantly, have fun learning programming. And if makaila ka nako. Yes I know, gwapo kaayu ko.