	Carátula para entrega de prácticas

Facultad de Ingeniería	Ingeniería en Computación

Profesor(a):	
Asignatura:	Programación Orientada a Objetos
Grupo:	1
No de Práctica(s):	5
Integrante(s):	11700202-9
No. de Equipo de cómputo empleado:	
Semestre:	2024-1
Fecha de entrega:	6 de octubre de 2023
Observaciones:	

CALIFICACIÓN: _______

Index.

Practice 5 - Abstraction and Encapsulation.

The hangman game.

1.- Introduction.

This report has the objective of explaining the fifth practice of the Object oriented programming subject, in which we must explore the abstraction and encapsulation principles.

We must build an algorithm for playing the Hangman game with 10 possible words, using the concepts mentioned before and with the restriction of not using StringBuilder class.

1.1.- Hypothesis

We have to create two classes (Figure and Word) and make their object instances interact in the main class. To avoid the restriction we will use a character array just like the C strings. As we can't repeat letters, it's a wise choice to use a set data structures for checking the used words.

2.- Development

The steps to follow were:

- Create the Figure class.
- Create the Word class.
- Define the attributes and methods for Figure..
- Define the attributes and methods for Player.
- Define the interactions between the instances and the user.

Hash set.

HashSet is a collection that allows you to store unique elements efficiently without any specific order. It's commonly used when you need to ensure uniqueness in a collection of items, and you don't care about their order.

The HashSet hierarchy seems like this:

```
java.lang.Object
    java.util.AbstractCollection<E>
        java.util.AbstractSet<E>
            java.util.HashSet<E>
```

2.1.- Figure class abstraction.

Figure should be a 3 x 3 character matrix in which.

- “ ” (SPACE) is a blank space.
- “/ \” Are the legs and arms.
- “|” Is the torso.
- “O” Is the head.

Each character has a single wrong answer value of 6.

So we need only two attributes and using encapsulation, both are private:

- Errors (Integer).
- hangMan (3 x 3 character matrix).

There are no setters and no getters because of the class behavior (check methods for more information).

Builder: Initializes the matrix in blank spaces and the errors in 0.

Public methods:

wrongAnswer (void): Before the user types a letter, if the checkForWord method of the Word class is false, it will increment the errors by one and check in which error we are for drawing the corresponding part of the hangman in the matrix.

checkForLose (boolean): Check if the errors are less than 6. If it’s false the game continues, and if the errors are 6, returns true and the game is over.

printHangman (void): Displays the hangman current state on the screen.

Class Figure
- Errors (Integer, private) - hangMan (Character matrix, private)
- wrongAnswer (void, public) - checkForLose (boolean, public) - printHangman (void, public)

2.2.- Word class abstraction.

Word needs of six attributes:

- Words (Array of strings): Contains the 10 possible words for the game.
- Word (Array of characters): Contains the target word.
- Display (Array of characters): Contains the word to display on screen.
- Word Size (Integer): Contains the chosen word size.
- Missing Letters (Integer): Contains the missing letters number on the display array.
- Used (HashSet of wrapper Characters): Contains the letters already used by the player.

Builder: Generates a random number in the [0,9] range, for choosing a string of the Words array. Then we iterate over the selected string for setting the characters on the Word and Display array.

We need to encapsulate those attributes so we've made them private.

There are no setters and no getters because of the class behavior (check methods for more information).

Methods:

checkForWord (boolean): Checks if the input letter is already on the set, then, linear searches in word for the character, if the letter is in the word, adds the letter to the set and returns true, otherwise will add the letter to the set and returns false.

Pseudocode.

```
Set a boolean flag to false.
If the 'a' character is already in the 'used' collection:
    Print "La letra ya fue usada, por jugarle al vivo ahi va otro error"
    Return false (indicating that the operation failed)
For each index 'i' from 0 to 'wordSize - 1':
    If the character at 'word[i]' is equal to 'a':
        Set the character at 'display[i]' to 'a'
        Decrement 'missingLetters' by 1
        Set the flag to true
Add the character 'a' to the 'used' collection.
Return the value of the flag (true if 'a' was found and updated, false otherwise).
```

checkForWin (boolean): Checks if there are no missing letters at the display word, if it's true, the game is over.

printWord (void): Displays the word on the screen

Class Word
- Word (Array of characters, private) - Display (Array of characters, private) - Word Size (Integer, private)

- Missing Letters (Integer, private) - Used (HashSet of wrapper Characters, private)
- checkForWords (boolean, public) - checkForWin (boolean, public) - printWord (void, public)

2.3.- Main class.

Pseudocode of the main class:

```
Create a new game instance called 'game'.

Create a new hangman figure instance called 'donChepe'.

Create an empty set called 'alphabet'.


For each character 'letter' from 'A' to 'Z':

    Add 'letter' to the 'alphabet' set.


For each character 'letter' from 'a' to 'z':

    Add 'letter' to the 'alphabet' set.


Print "Bienvenido al juego del Ahorcado!"

Print "Tienes 6 errores posibles!"

Print "Inicio!"

Print the current state of the word in the game.
```

```
Repeat the following loop while the game is not won and 'donChepe' has not lost:
    Prompt the user to enter a letter and store it in the variable 'move'.

    Repeat the following loop while 'move' is not in the 'alphabet':
        Print "Ingresa una letra: "
        Read user input into 'move'.

    If the game's 'checkForWord' method returns true for 'move':
        Print "Has encontrado la letra 'move'!"
    Else:
        Print "La palabra no contiene 'move'!"
        Update 'donChepe' to indicate a wrong answer.

    Print the current state of the hangman figure.
    Print the current state of the word in the game.
```

Conclusion: The hypothesis was correct, the set was very useful for storing the already used values and the correct letters and checking them in constant time. Also the classes handling was very efficient and intuitive.

This is the second practice in which we use encapsulation and abstraction, and I'm keeping my idea that these principles are so intuitive and programmer friendly.

References:

- [1] Oracle. (2023, July). Java API, HashSet [Online]. Available: https://docs.oracle.com/javase/8/docs/api/java/util/HashSet.html