Custom Project: Anagrams

Sydney Amaya

Project Purpose

Name: Anagrams

The purpose of my project is to recreate a version of the game anagrams. The user chooses their difficulty, the letters are presented to the user, and then the user must rearrange the letters into words. Each level ends when the user types done. Also, each correct guess is a point which will be tallied at the end of the game.

Overview of Software and Structure

- **Scanner**: I used scanners a lot to read files and store user input.
- **Files**: I used files to store each level. I used a scanner to read each file. This was more efficient than using a lot of strings.
- **Array**: I stored the lines from each file in an array to access them easily in my code and use them in loops.
- **Loops**: I used a lot of loops to compare the user's input to each element in the arrays containing the lines from the files.
- **If statements**: I used if statements to determine if user input matched any possible correct words.

Overview of Software and Structure Continued

I used one class

Overview of methods:

- readInstructions: reads from a file and Sop the instructions for the user
- whichDifficulty: determines which difficulty the user wants
- readLinesEasy & readLinesMedium: reads each file, calls userInput to start each level
- userInput: prompts user for input and calls over methods to check the input
- checkInput: uses a loop to check input with the array of possible answers and returns a boolean
- tellUserIfcorrect: takes the boolean from checkInput, either sop correct and add one to count or sop false

Key Part

This method shows how I used arrays and loops in my project.

It takes two parameters: String word which stores the user's input and String [] correct which stores all the lines from the file - each line is a possible word*

Using a for loop it compares word to each element. If word equals to i, then myBoolean = true.

If the loop ends and word is not equal to any element, then myBooleam = false.

Returns myBoolean

```
1 usage
public static boolean checkInput(String word, String[] correct) {
    boolean myBoolean = false;
    for (int i = 1; i < correct.length; i++) {
        if (word.equals(correct[i])) {
            myBoolean = true;
        }
    }
    return myBoolean;
}
```

File Example

*Element 0 is the first line which is not a word.

Anagrams.java | LevelOne.txt

```
1  t,o,n,u
2  not
3  nut
4  ton
5  out
6  unto
```

Changes

- The user can't choose the number of levels
- Only one round
- The user can only choose the difficulty once
- The levels were going to be presented to the user in a random order each time the game is play

I didn't include these features because I felt like it made the game too complicated to code.

Problems

1.Originally, the user could only guess 5-6 times depending on the length of the array containing the correct answer because I used a for loop.

Solution: I used a while loop to allow the user unlimited attempts.

2.Count would reset to zero after each level.

Solution: I originally had the variable count in the userInput method which kept setting it to zero because it is called three times, so I moved it to readLinesEasy and readLinesMedium methods which are only called once.

Testing

I had my non-cs friends test my game while I took notes. Problems I saw:

- It wasn't clear to them when a new level started
- They were not given enough attempts

Solutions:

- Used a while loop to give the user more attempts
- Added a statement stating when a new level started

What did I learn?

I learned how useful files are in programs. Instead of using numerous String or print ln statements, I can just access a file. I also learned how useful loops are, especially when they are used to traverse arrays as well as prompt the user for input. Finally, I learned how intricate programs are for games since all my methods relied on other methods.

Conclusion

While it's far from perfect, my game is functional. The methods easily work with each other, and the user interface is comprehensible. It is not completely free of bugs and redundancy, but it runs how I expected it to.