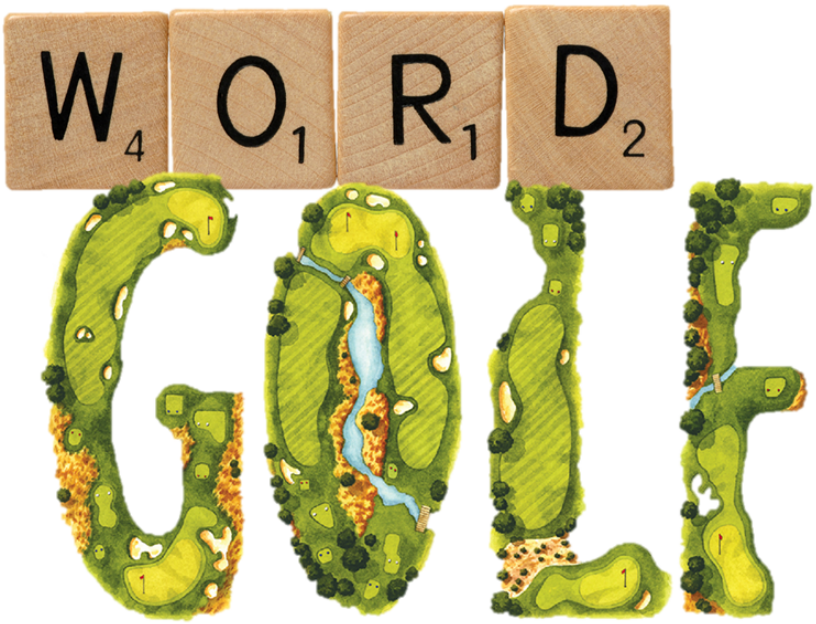
APCS Word Golf Project

Name_____



Introduction

The objective of this project is to create a game called **`WordGolf`** using the Java programming language. The user will input words into the console, and the program should calculate the value of the words. **The goal of the game is to reach exactly 100 points in the least number of sentences/strokes.**

Summary

- Your goal is to score 100 points with the least number of sentences/strokes.
- Each sentence is made up of the sum of its words.
- Each word is scored from left to right as follows:
  - Each word begins with 1 point.
  - Each consonant will add 1 point.
  - Each vowel will double the value of the word up to that point.
- Each time you type a sentence that gets scored, it counts as a stroke.
- If you overshoot your goal of 100, the following sentence will count as negative.

Specifications

Input words will be typed into the console, separated by spaces. Each sentence of words is worth a certain amount of points, based on the following rules:

Words: Starting at the left side of the word and moving to the right, each character in the word will add to the value.

- Every word begins with a value of 1 point.
- Consonants are worth 1 additional point.
- Vowels (a,e,i,o,u,y) double the value of the word up to that point.

- Example: The word “Java” would be worth 10 points. The value starts at 1, ‘J’ makes it 2, ‘a’ doubles it to 4, ‘v’ makes it 5, then the final ‘a’ doubles the value again to 10.

word start	**“J"**	**“A”**	**“V”**	**“A”**
((((1	+ 1)	* 2)	+ 1)	* 2)
1	2	4	5	**10**

- Example: The word “Google” would be worth 20 points. The value starts at 1, ‘G’ makes it 2, ‘o’ doubles it to 4, the second ‘o’ makes it 8, ‘g’ makes it 9, ‘l’ makes it 10, and ‘e’ doubles it to 20.

word start	**“G”**	**“o”**	**“o”**	**“g”**	**“l”**	**“e”**
((((1	+ 1)	* 2)	* 2)	+ 1)	+ 1)	* 2)
1	2	4	8	9	10	**20**

Sentences: Words are separated by spaces. Multiple words in an input line form a sentence.

- Example: The sentence “The owls are not what they seem” would be worth 50 points.

“The”	**“owls”**	**“are”**	**“not”**	**“what”**	**“they”**	**“seem”**
6	+ 5	+ 6	+ 5	+ 7	+ 12	+ 9
	11	17	22	29	41	**50**

Overall Game Logic:

If a player scores a total value higher than 100, they have overshot the goal. The game continues, but the next sentence must be counted as negative.

The game should continue accepting sentences, counting the number of strokes along the way, until the player reaches exactly 100 points (or enters the “quit” command).

Printed Reports:

After each sentence/stroke, the game should output:

- the Stroke number,

- the sentence just entered,
- the value of the sentence's points,
- the total points accumulated so far, and
- the distance away from the goal that the user still is.

Input	Output	Notes
`Hello world`	`Stroke 1: "Hello world" = 19 points` `Your total is 19 points, 81 from the goal!`	*"Hello" = 12 pts* *"world" = 7 pts*
`I love Computer Science`	`Stroke 2: "I love Computer Science" = 67 points` `Your total is 86 points, 14 from the goal!`	*"I" = 2 pts* *"love" = 10 pts* *"Computer" = 27 pts* *"Science" = 28 pts*
`but not Apple`	`Stroke 3: "but not Apple" = 20 points` `Your total is 106 points. You overshot your goal by 6 pts!`	*"but" = 5 pts* *"not" = 5 pts* *"Apple" = 10 pts*
`help`	`Stroke 4: "help" = 6 points` `Congratulations, you won with only 4 strokes!`	*"help" = 6 pts*

Checkpoints

There will be 3 checkpoints to ensure your game is on track for completion. You must submit them on time to receive full credit.

Checkpoint 1: Parsing One Word [due Wed 12/14]

- Your source file starts with a comment that includes your name!

- Your `main()` method
  - creates a Scanner,
  - reads one word, and
  - passes that word as a parameter to a function called `parseWord()`.
- `parseWord()` takes in the word, calculates the value of that word and returns it.
  - Remember: The value of a word always starts at 1!
  - Hint: You should look at each letter one at at a time to evaluate:
    - Multiply by 2 for each vowel
    - Add 1 for each consonant
- `main()` prints the result returned by `parseWord()`.

Checkpoint 2: Determining the Value of a Sentence [due Fri 12/16]

- Your `main()` method now
  - Accepts in a sentence instead of a single word, and
  - passes that sentence as a parameter into a function called `parseSentence()`.
- `parseSentence()`
  - takes in the sentence, calculates the value of the entire sentence, and returns that value.
  - Each word must be parsed and evaluated:
    - *How can you identify what a word is in a sentence?*
    - For each word in the sentence
      - Evaluate that word with the `parseWord()` method and add it to the sentence total

- Demonstrate that the words are being parsed correctly by returning a string that describes each word with its value and correctly computes the total value of the sentence.

  Examples:
  - "Hello": 12 points, "world": 7 points, sentence value: 19 points
  - "I": 2 points, "love": 10 points, "Computer": 27 points, "Science": 28 points, sentence value: 67 points

Checkpoint 3: Multiple Sentences and Punctuation [due Tuesday 12/20]

- Keeps track of the number of sentences/strokes.
- Keeps track of the total value of all of the sentences entered.
- Continues reading the user's sentences until they score exactly 100 points.
    - Ensure that each a new sentence is added **or subtracted** as appropriate.
- Provide useful feedback to the user after each sentence/stroke:
    - What is the current sentence count? ("stroke number")
    - Points of the current sentence
    - Total points scored from all sentences
    - (How many points away from the goal?)
- Provides separate reports each sentence depending on if the user is under 100 or overshot 100.

Final Project: All requirements complete [due Wed 12/21]

- Your final output should look like the examples from the table above.
- Write a congratulations message for when a user wins!

Above: Strengthen Game Options [Extra Credit]

- Ensures that uppercase letters and lowercase are scored equally.
- All kinds of punctuation must be accepted from user input. Punctuation should not affect the value of any of the words in any way.
- Gives the user the option to "quit" in the middle of the game.
- Tells the user that they are a quitter if they do so.
- Allows the user to play another game without restarting the program.

Beyond: Convert Word Golf into a 2 player game [Extra Credit]

- Ask for names for 2 players
- Each player takes turns typing a sentence
- The player with the lowest number of strokes wins!

Comments

- **IDE:** You can use Eclipse or repl.it for this project. Ask for an invite code if you want to use repl.it
- **TOPICS:** You should be able to be successful on this project with ONLY the topics we have learned so far in class. Remember to review your notes and textbook
    - Scanner, indexOf(), substring(), etc.
- **COMMENTS:** You should be heavily commenting your code so it is very clear what you are trying to do all along the way.

Grading Breakdown

	REQUIREMENT	POINTS
A	Word Parsing	25
B	Sentence Parsing	25
C	Stroke Reports	15
D	Game Outcomes	20
E	Handles overshots	15
F	Code Comments	10
TOTAL		**110 points**
G	*Above: Improved Game Ops*	10
H	*Beyond: Multi-Player*	10

Remember to Practice Academic Integrity:

- All submitted code should be your own work.
- Do NOT "help" a classmate by typing code for them.
- Do NOT "help" a classmate by sending them your code.
- **DO** ask questions and get feedback about **your own code** from teachers and classmates!
  See MIT's Academic Integrity policy