UpCurve PE 2024

day 8 assignment

Assignment #1

Word Reversal

Write a Java method `reverseWords` that takes a string as input and returns a new string where the order of words is reversed. Words in the input string are separated by one or more spaces. You should preserve the spacing between words in the output string.

For example:

- Input: "Hello World" Output: "World Hello"
- Input: "This is a test" Output: "test a is This"

Write JUnit 5 test cases to validate the functionality of the `reverseWords` method. Ensure to cover various test scenarios, including:

1. Testing with a single word input.
2. Testing with multiple words input.
3. Testing with leading and trailing spaces.
4. Testing with extra spaces between words.
5. Testing with an empty string input.
6. Testing with special characters and punctuation marks.

Code structure:

```
public class WordReversal {

    public static String reverseWords(String input) {
        // TODO: implement your logic here
        return null;
    }
}
```

Ensure to cover edge cases and boundary conditions in your test cases to validate the correctness of the solution.

Assignment #2

Anagram Groups

Write a Java method `groupAnagrams` that takes an array of strings as input and groups the strings into lists of anagrams. Anagrams are words that have the same characters but in a

different order.

For example:

- Input: ["eat", "tea", "tan", "ate", "nat", "bat"] Output: [["eat", "tea", "ate"], ["tan", "nat"], ["bat"]]

Write JUnit 5 test cases to validate the functionality of the `groupAnagrams` method. Ensure to cover various test scenarios, including:

1. Testing with an array containing anagrams.
2. Testing with an array containing no anagrams.
3. Testing with an empty array input.
4. Testing with arrays containing words of different lengths.
5. Testing with arrays containing words with special characters.
6. Testing with arrays containing a mix of uppercase and lowercase letters.

Code structure:

Here's an implementation of the `groupAnagrams` method to be done:

```
public class AnagramGroups {

    public static List<List<String>> groupAnagrams(String[] strs) {
        // TODO: implement your logic here
        return null;
    }
}
```

Ensure to cover edge cases and boundary conditions in your test cases to validate the correctness of the solution.

Assignment #3

Prime Factorization

Write a Java method `primeFactors` that takes an integer as input and returns a list of its prime factors. A prime factor is a prime number that divides the given integer without leaving a remainder.

For example:

- Input: 30 Output: [2, 3, 5]
- Input: 56 Output: [2, 7]

Write JUnit 5 test cases to validate the functionality of the `primeFactors` method. Ensure to cover various test scenarios, including:

1. Testing with prime numbers.
2. Testing with composite numbers.
3. Testing with a number that has only one prime factor.
4. Testing with negative numbers (if your implementation supports them).
5. Testing with zero and one.

Code structure:

```
import java.util.ArrayList;
import java.util.List;

public class PrimeFactorization {

    public static List<Integer> primeFactors(int n) {
        // TODO: implement your logic here
        return null;
    }
}
```

Ensure to cover edge cases and boundary conditions in your test cases to validate the correctness of the solution.
