CE Iterator | Recursion

0/10 Points

| 9/26/2022

Attempt 1

REVIEW FEEDBACK

Offline Score: **0/10**

Add Comment

Unlimited Attempts Allowed

Details

Fundamentals

CE: Iterator | Recursion

Learning Objectives

- Implement an iterator.
- Identify common pitfalls of recursion.
- Implement a recursive method.

Overview

This CE consists of three parts.

In Part 1 you will add an iterator to class WordList.
In Part 2 you will trace code statements and identify common pitfalls of recursion.
In Part 3 you will modify and implement recursive methods.

Previous (https://slcc.instructure.com/courses/817632/modules/items/18752963)

Submit Assignment

Next (https://slcc.instructure.com/courses/817632/modules/items/18752966)

Instruction

Part 1

Part 1 builds on a class we started in CE ArrayList | LinkedList

- Open class **WordList** from the package **ceLinked.**
- Add a foreach loop at the end of the main method to print all the list elements side by side, separated by single spaces. Include a brief description so the output shows that these are the list elements printed with the foreach loop.
  Notice the squiggly line. It won't compile because foreach loops can only iterate over arrays or instances of java.lang.Iterable.
- Implement the interface Iterable<String> for class WordList.
  When you are done, the foreach loop should compile.
- Run it and verify that all elements are listed in the expected order.
  If you followed the instructions posted in CE ArrayList | LinkedList the combined output should look like ***this***

Part 2

Recursions allow us to write clear and concise code. You will encounter it throughout this course, especially in connection with trees and graphs.

However, there are **four common pitfalls** that need to be avoided:

A. Missing base case
B. Excessive memory usage
C. No guaranteed convergence
   That's when there are situations when the base case might not be reached.
D. Excessive recomputation
   When the same calculations are performed over and over again.

Below you find three methods that attempt to calculate the nth **Harmonic number (https://en.wikipedia.org/wiki/Harmonic_number)**, however, each of them has an implementation issue.

Previous (https://slcc.instructure.com/courses/817632/modules/items/18752963)

Submit Assignment

Next (https://slcc.instructure.com/courses/817632/modules/items/18752966)

```
        return 1;
    return harmonicNumber1(n) + 1.0 / n;
}
```

Trace the code to determine the value is returned for n = 1. Repeat for n = 3.
Which value(s) did the method return? Which pitfall did you encounter?
When you are done, ***compare***

- ```
  private static double harmonicNumber2(int n) {
      if(n == 0)
          return 0.0;
      return harmonicNumber2(n - 1) + 1.0 / n;
  }
  ```
  Trace the code to determine the value is returned for n = 1. Repeat for n = 3.
  Which value(s) did the method return? Which pitfall did you encounter?
  When you are done, ***compare***

- ```
  private static double harmonicNumber3(int n) {
      return harmonicNumber3(n - 1) + 1.0 / n;
  }
  ```
  Trace the code to determine the value is returned for n = 1. Repeat for n = 3.
  Which value(s) did the method return? Which pitfall did you encounter?
  When you are done, ***compare***

This exercise is based on an example from Prof. Sedgewick's book 'Computer Science - An Introductory Approach'.

Part 3

In this last part, you will write a recursive method.
It can take a while to feel comfortable with recursive methods. If you would like additional practice opportunities, check out **Coding Bat Recursion-1 (https://codingbat.com/java/Recursion-1)** .

- Create a package **ceRecursion** and add a class **Recursion**. It includes the main method

Previous (https://slcc.instructure.com/courses/817632/modules/items/18752963)

Submit Assignment

Next (https://slcc.instructure.com/courses/817632/modules/items/18752966)

needs to be positive.

How to calculate Hailstorm numbers:

1. The first number is the seed provided by the client.
2. Calculate the next number the following way:
   If the number is even, the next number is half the number
   otherwise, the next number is three times the number + 1
3. Repeat 2 until you reach 1

   Examples:
   3 10 5 16 8 4 2 1 // seed 5
   12 6 3 10 5 16 8 4 2 1 // seed 12

- In the main method, call hailstone four times and pass the values 3, 16, 17, and 24 as arguments. Include a title as shown in the 'Expected Output' section.

Expected Output (Part 3)

```
Hailstone numbers
-----------------
3 10 5 16 8 4 2 1
16 8 4 2 1
17 52 26 13 40 20 10 5 16 8 4 2 1
24 12 6 3 10 5 16 8 4 2 1
```

Submission

Previous (https://slcc.instructure.com/courses/817632/modules/items/18752963)

Submit Assignment

Next (https://slcc.instructure.com/courses/817632/modules/items/18752966)

Post the video.

(https://slcc.instructure.com/courses/817632/modules/items/18752955)

(https://slcc.instructure.com/courses/817632/modules/items/18752956)

(https://slcc.instructure.com/courses/817632/modules/items/18752957)

(https://slcc.instructure.com/courses/817632/modules/items/18752958)

(https://slcc.instructure.com/courses/817632/modules/items/18752959)

(https://slcc.instructure.com/courses/817632/modules/items/18752960)

(https://slcc.instructure.com/api/v1/courses/817632/module_item_redirect/18752962)

(https://slcc.instructure.com/api/v1/courses/817632/module_item_redirect/18752963)

(https://slcc.instructure.com/courses/817632/modules/items/18752965)

(https://slcc.instructure.com/courses/817632/modules/items/18752966)

(https://slcc.instructure.com/courses/817632/modules/items/18752968)

Choose a submission type

Text | Upload | Office 365 | More

Previous (https://slcc.instructure.com/courses/817632/modules/items/18752963)

Submit Assignment

Next (https://slcc.instructure.com/courses/817632/modules/items/18752966)