Hogeschool van Amsterdam

Datastructures

Assignment 1B Recursion

Tijmen Stor, 500752989, IS205

Hogeschool van Amsterdam

Index

Introduction

In this assignment, I will be answering the questions which are part of assignment 1B. These questions will revolve around recursion in programming. In this assignment, I will be writing a recursive method to determine whether a row of numbers is increasing or not. If anywhere in the list the numbers do not increase, it will return false. I will also be writing a recursive program that will take a single argument, which determines the amount of syllables in the words. The program will then write down all the possible words while taking in account the amount of syllables asked.

Assignment 1B

Sub-Assignment 1

The first question attached to assignment 1B is:

Write a recursive method that checks if a row of numbers increases. The method gets a row of numbers as argument and returns a Boolean. This Boolean will show if the row is increasing or not.

To create this method, we start off by creating a new Java class and implementing the interface already given to us. Then we start to write out the method given to us from the interface called isStijgend (isIncreasing). We want the method to be recursive, so whenever the method is not done checking yet, it should call on itself again. However, when the last 2 numbers in the row are checked and proof to be increasing, the method itself should return true. The method will look like this:

```
@Override
public <T extends Comparable<T>> boolean isStijgend(List<T> rijtje) {

    if (rijtje.size() <= 1 && a == 0) {
        return false;
    }
    a++;
    if (rijtje.get(0).compareTo(rijtje.get(1)) == -1) {

        rijtje.remove(0);
        if (rijtje.size() == 1) {
            a = 0;
            return true;
        }
        isStijgend(rijtje);
    }
    return a == 0;
}
```

Let us go over the entire method. The method cannot check a row if it only contains one value. So whenever the row is only one value, it will return false by default. Part of the if-statement is a == 0. This will check whether the method has been called already or not. Because it is possible that there is one value left in the row, but the method has been used before. The method will go on to add value to the variable a. This will prevent the if-statement from before to being triggered next time.

Next up is the if-statement that will compare the current number against the upcoming number. If the current number is bigger than the upcoming number or the same, it will make the method return false. However, if the value is smaller than the upcoming number, it will continue the statement. It will remove the current number from the row so that next time the method is called, the next 2 numbers will be used in the comparison. Between the removal of the current number and the calling of the method, another if-statement is issued. Within this if-statement, the amount of numbers within the row is checked. If there is only one number left in the row, the method will return true. This is, because every number before that is checked and showed to be increasing in value.

Sub-Assignment 2

The second assignment attached to assignment 1B is:

Write a recursive method String[] yololian(int n) that accepts one argument: an integer n. The method returns an array containing all the words containing integer n amount of syllables.

At the start of this assignment, a new Java class will be created too. The class will be implementing the interface given to us and the method yololian will be written. The language consists of 2 syllables, so we need to create 2 variables containing those Strings. After that, we need to create an if-statement for when the given argument is only 1. In this statement, the method will return the syllables yo and lo. If that if-statement is ignored, a list will be created. After the creation of the list, a for-statement is called. In this statement, the list is filled with yo's and lo's. For every argument n, a new yo and lo will be added to the previously stored yo's and lo's, creating the necessary syllables in the method. However, key in this for-loop is the n-1 argument at the start, because whenever 2 syllables are needed, the loop will not stop after 1, but at 0. Because of this, the n-1 is necessary. The code will look like this:

```
@Override
public String[] yololian(int n) {
    String yo = "yo";
    String lo = "lo";

    if (n == 1) {
        return new String[]{yo, lo};
    }

    List<String> list = new ArrayList<>();

    for (String s : yololian(n - 1)) {
        list.add(s + yo);
        list.add(s + lo);
    }

    return list.toArray(new String[0]);
}
```

When the method is called with argument yololian(2), the method will return yoyo, yolo, loyo, lolo. When yololian(3) is called, yoyoyo, yoyolo, yoloyo, yololo, loyoyo, loyolo, loloyo, lololo are returned by the method.

Conclusion

After this assignment, I have learned to successfully make a recursive method. I do not feel like my second assignment was recursive enough, but I was unable to find another solution to the problem. In the first assignment however, I was successful in implementing a recursive method to determine if the rows were increasing or decreasing. The implementation of recursion into your own methods takes some more mathematical skill, because you need to calculate what the method will do whenever it calls itself again and what possible outcomes are possible.