TO RECOURSE
OR
NOT TO RECOURSE ?

THIS IS THE **QUESTION** …

Massimo Caccia

DIVIDE ET IMPERA

Una delle citazioni più famose di Cesare è Divide et impera (Divide and Conquer). Letteralmente, questa espressione si traduce come "dividi per regnare". Questa frase, che è diventata popolare, significa che se le persone sono divise tra loro, diventa molto più facile da gestire. Dopotutto, la forza consiste nell'unità e una ad una diventa molto più difficile resistere. Le parole di Giulio Cesare "Dividi e governa" sono ancora oggi utilizzate da molti leader come credo principale. Ma spesso il sovrano non ha nemmeno bisogno di dividere il popolo - le persone stesse si riuniscono in "gruppi di interesse" in cui esiste una sola verità e qualsiasi dissenso è considerato nemico di questo gruppo.

THE **QUESTION** IS HOW…

L'approccio TDD e la relativa giusta separazione del **COSA** (il comportamento specificato, validato e assicurato dai test) dal **COME** (l'implementazione) permettono di sperimentare anche altre modalità di suddividere il problema in uno o più problemi più piccoli come il ricorrere alla tecnica della ricorsione (e di tornare indietro se questa si rileva poco adatta)

Giovedì 10 febbraio

Massimo Caccia 17:03
avessimo implementato il metodo add recursivo adesso il test "1,\n2" sarebbe fallito 😀😀
👏 1

Alessandro 17:04
prova a implementarlo con la ricorsione

17:04 e' un buon esercizio

```java
public class StringCalculator {


    public int add(String numbers) {
        if (numbers.isEmpty()) {
            return 0;
        }

        String[] separators = {",", "\n"};

        for (int i = 0; i < separators.length; i++) {

            int lastSeparatorIndex = numbers.lastIndexOf(separators[i]);

            if (lastSeparatorIndex == 0 || lastSeparatorIndex == numbers.length() - 1) {
                throw new RuntimeException("Double separator");
            } else if (lastSeparatorIndex != -1) {
                return add(numbers.substring(0, lastSeparatorIndex)) + add(numbers.substring(lastSeparatorIndex + 1, numbers.length()));
            }
        }

        return Integer.parseInt(numbers);
    }
}
```

QUESTION: Why this subject ?

QUESTION: What is Recursion?

Recursion is when a function calls itself again and again until it reaches a specified stopping condition.

Each recursive function has two parts:
Base Case: The base case is where the call to the function stops i.e., it does not make any subsequent recursive calls
Recursive Case: The recursive case is where the function calls itself again and again until it reaches the base case.

How do you solve a problem using recursion?
To solve a problem using recursion, break the problem into one or more smaller problems, and add one or more base conditions that stop the recursion. i.e. make a **recurrence relation** for that problem.

```java
public class Fibonacci {


    public int nthFibonacci(int position) {

        if (position < 2)
            return position;
        else
            return (nthFibonacci( position: position - 1) + nthFibonacci( position: position - 2));
    }
}
```

Iteration

- Iteration uses **repetition structure.**
- An infinite loop occurs with iteration if the loop condition test never becomes false and Infinite looping uses CPU cycles repeatedly.
- An iteration **terminates** when the **loop condition fails.**
- An iteration does not use the **stack** so it's **faster than recursion.**
- Iteration consumes **less memory.**
- Iteration makes the **code longer.**

Recursion

- Recursion uses **selection structure.**
- **Infinite recursion** occurs if the recursion step does not reduce the problem in a manner that converges on some condition (**base case**) and Infinite recursion can crash the system.
- Recursion terminates when a **base case** is recognized.
- Recursion is usually **slower than iteration** due to the overhead of maintaining the stack.
- Recursion uses **more memory than iteration.**
- Recursion makes the **code smaller.**

THE **QUESTION** IS: ARE THEM STRENGTHS, WEAKNESSES, OPPORTUNITIES OR THREATS?

Disadvantages of Recursion (Recursion takes up a lot more space and time!)

Recursion, broadly speaking, has the following disadvantages:
A recursive program has **greater space requirements** than an iterative program as each function call will remain in the stack until the base case is reached.
It also has **greater time requirements** because each time the function is called, the stack grows and the final answer is returned when the stack is popped completely.

Advantages of Recursion (Recursion makes code smaller)

On the other hand, recursion has the following advantages:
For a recursive function, you only need to define the base case and recursive case, so the code is simpler and shorter than an iterative code.
Some problems are inherently recursive, such as Graph and Tree Traversal.

THE **QUESTION** IS: ARE THEM STRENGTHS, WEAKNESSES, OPPORTUNITIES OR THREATS?

TO CONCLUDE, BACK TO THE **QUESTION** …

TO RECOURSE OR NOT TO RECOURSE ?

IT'S A MATTER OF WHEN IT IS TIME TO ASK THE **QUESTION** …

WHEN TRIANGULATION BY TRIANGULATION IT APPEARS THE PROBLEM BROKEN INTO MORE SMALLER PROBLEMS

Transformation Priority Premise - What is "Obvious implementation" ?

#	TRANSFORMATION	STARTING CODE	FINAL CODE
1	{} => **nil**		`return nil`
2	nil => **constant**	`return nil`	`return "1"`
3	constant => **constant+**	`return "1"`	`return "1" + "2"`
4	constant => **scalar**	`return "1" + "2"`	`return argument`
5	statement => **statements**	`return argument`	`return arguments`
6	unconditional => **conditional**	`return arguments`	`if(condition)return arguments`
7	scalar => **array**	`dog`	`[dog, cat]`
8	array => **container**	`[dog, cat]`	`{dog = "DOG", cat = "CAT"}`
9	statement => **recursion**	`a + b`	`a + recursion`
10	conditional => **loop**	`if(condition)`	`while(condition)`
11	recursion => **tail recursion**	`a + recursion`	`recursion`
12	expression => **function**	`today - birthday`	`CalculateAge()`
13	variable => **mutation**	`day`	`var day = 10; day = 11;`
14	**switch case**		

GRAZIE PER IL VOSTRO CONTRIBUTO (**PARLIAMONE!**) E DELLA VOSTRA ATTENZIONE

ANY **QUESTION** ?