

Estructuras de Datos

Gonzalo Gabriel Méndez, Ph.D.

Definición

Recursión

Un proceso recursivo se llama a sí mismo
continuamente

```
public class RecursionExample {  
  
    static void recursiveMethod() {  
        System.out.println("Hello");  
        recursiveMethod();  
    }  
  
    public static void main(String[] args) {  
        recursiveMethod();  
    }  
}
```

Resultado

¿Cuál es la salida del código anterior?

```
Hello  
Hello  
Hello  
Hello  
Hello  
Hello  
Hello  
Hello  
Hello  
Hello  
Hello  
Hello  
Hello  
Hello  
Hello  
Hello  
Hello
```

...

```
java.lang.StackOverflowEr  
ror
```

Solución

Se necesita una condición de salida

```
public class RecursionExample {  
    static int count = 0;  
    static void recursiveMethod() {  
        count++;  
        if(count<=5){  
            System.out.println("Hello " +  
count);  
            recursiveMethod();  
        }  
    }  
    public static void main(String[] args) {  
        recursiveMethod();  
    }  
}
```

Resultado

¿Cuál es la salida del código anterior?

Hello

1

Hello

2

Hello

3

Hello

4

Hello

5

OK, pero ¿por qué?

Hay problemas que, por naturaleza, son recursivos.

Por ejemplo, el **factorial** de un número:

$$\text{factorial}(n) = \begin{cases} 1, & \text{si } n = 0 \\ n * \text{factorial}(n-1), & \text{si } n > 0 \end{cases}$$

$$\text{factorial}(3) = 3 * \text{factorial}(2)$$

$$\text{factorial}(3) = 3 * (2 * \text{factorial}(1))$$

$$\text{factorial}(3) = 3 * (2 * (1 * \text{factorial}(0)))$$

$$\text{factorial}(3) = 3 * (2 * (1 * 1))$$

$$\text{factorial}(3) = 6$$

Método factorial en Java

```
public class RecursionExample {  
  
    static int factorial(int n) {  
        if (n == 0) {  
            return 1; // condición de salida o caso base  
        } else {  
            return (n * factorial(n-1)); // llamada recursiva  
        }  
    }  
  
    public static void main(String[] args) {  
        System.out.println ("Factorial of 5 is: " +  
factorial(5));  
    }  
}
```


Ejercicio: La sucesión de Fibonacci

La serie comienza con los números 0 y 1

Cada nuevo término de la sucesión es la suma de los dos anteriores

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610,
987...

La serie comienza con los números 0 y 1

Cada nuevo término de la sucesión es la suma de los dos anteriores

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14
fibonacci (n)	0	1	1	2	3	5	8	13	21	34	55	89	144	233

Escriba un método recursivo que calcule el n-ésimo término de la
sucesión de Fibonacci

Ejemplo: fibonacci (9) = 21

Una posible solución

```
public class Main {  
  
    public static int fibonacci (int n) {  
        if (n == 1) {  
            return 0;  
        } else if (n == 2) {  
            return 1;  
        } else {  
            return fibonacci(n-1) + fibonacci (n-2);  
        }  
    }  
  
    public static void main(String[] args) {  
        for (int i = 1; i <= 14; i++) {  
            System.out.println(i + ": " + fibonacci(i));  
        }  
    }  
}
```

Una posible solución

```
public class Main {  
    public static int fibonacci (int n) {  
        switch (n) {  
            case 1:  
                return 0;  
            case 2:  
                return 1;  
            default:  
                return fibonacci(n-1) + fibonacci (n-2);  
        }  
    }  
  
    public static void main(String[] args) {  
        for (int i = 1; i <= 14; i++) {  
            System.out.println(i + ": " + fibonacci(i));  
        }  
    }  
}
```

```
1: 0  
2: 1  
3: 1  
4: 2  
5: 3  
6: 5  
7: 8  
8: 13  
9: 21  
10: 34  
11: 55  
12: 89  
13: 144  
14: 233
```

Comentarios finales

Para recordar:

Un proceso recursivo se llama a sí mismo continuamente

La implementación debe contener al menos un caso base o condición de salida

De otro modo, el proceso se ejecuta un número infinito de veces

A lo largo del curso abordaremos más problemas recursivos

Recursión será un tema **fundamental** en el segundo parcial

Ejercicios en Clase

Escribir los siguientes métodos estáticos aplicando una estrategia recursiva:

esPalindrom Recibe un objeto de tipo String y retorna si ésta o no es un palíndromo (que se lee igual de izquierda a derecha y viceversa)

revertir Recibe un arreglo y lo modifica invirtiendo el orden de los elementos que éste contiene

Nota: Considere primero los casos bases y piense en qué parámetros sus métodos necesitan recibir