

## RO1 Recursion: Factorial

$$n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n$$

long fact = 1;  $\rightarrow$  se guarda el resultado y comienza en 1  
para no alterar el valor ya que  $n \times 1 = n$

esto ya simula  $1 \times 2 \times 3 \times \dots \times n$   
al llegar al caso base

if ( $n \leq 1$ ) return 1;

Por lo que si:  $n = 1 \rightarrow$  devuelve 1  
 $n = 0 \rightarrow$  devuelve 1

en el caso recursivo:

return  $n * \text{factorialRecursivo}(n-1)$

lo que se ve matemáticamente como:  $n! = n \times (n-1)!$

## RO2 Recursion: Suma

Sumar b veces +1 a a cuando  $b > 0$  y restar 1 a a cuando  $b < 0$ , si  $b = 0$ , si  $b = 0$  entonces la suma solo es a

int resultado = a  $\rightarrow$  guarda el valor inicial

Para  $b > 0$ :  $\rightarrow$  se repite "b" veces.

for (int i = 0; i < b; i++) {

resultado ++

}

$\rightarrow$  cada vuelta suma 1

para  $b < 0$   $\rightarrow$  Repite  $|b|$  veces

```
for (int i = 0; i > b; i--) {
```

```
    resultado --;
```

```
}
```

$\rightarrow$  cada vuelta resta 1

El proceso es el mismo para while y do-while pero cambiando el orden en la estructura.

RO3 Recursion: Multiplicación

$a + a + a + \dots$  ( $b$  veces)

```
private int multiplicacionFor (int a, int b)
```

```
int resultado = 0  $\rightarrow$  la multiplicación inicia en 0 porque se irá sumando a repetidamente
```

```
for (int i = 0; i < b; i++) {
```

```
    resultado += a
```

$\rightarrow$  repite el ciclo  $b$  veces

```
}
```

$\rightarrow$  cada vuelta suma  $a$

```
return resultado;
```

```
}
```

El proceso es el mismo para while y do-while pero cambiando el orden en la estructura

RO4 Recursion: Potencia

$$a^b = \underbrace{a \cdot a \cdot a \cdot \dots \cdot a}_{b \text{ veces}}$$

private long potenciaFor(int a, int b)

long resultado = 1;  $\rightarrow$  comienza con 1

for (int i = 0; i < b; i++)  $\rightarrow$  repite el ciclo b veces

resultado \*= a

}

$\rightarrow$  cada vuelta multiplica a consigo mismo

return resultado; hasta las b veces que se repitió el ciclo.

}

El proceso sería lo mismo para el while

private long potenciaDoWhile(int a, int b) {

if (b == 0) return 1;  $\rightarrow$  Como do-while se ejecuta al menos una

long resultado = 1;

vez igualando b con cero para evitar que multiplique una vez más.

do {

$\rightarrow$  hace entrar el ciclo en 1 en el primer ciclo

resultado \*= a;

i++;

$\rightarrow$  cada vuelta multiplica a consigo mismo

} while (i < b)

return resultado;

}

ROS: Cuentas Progresivo

Imprimir: 1, 2, 3, ..., n

$a_k = k$  para  $k = 1, 2, 3, \dots, n$

sol. incrementa 1 en cada paso.

```
private static void counterFor (int n) {  
    for (int i = 1; i <= n; i++) {  
        System.out.print(i + " ");  
    }  
}
```

Inicio del control →   
condición para seguir →   
acción en cada ciclo →   
acción del ciclo

El primer sería el mismo cambiando únicamente la estructura para while y do-while considerando que do-while se ejecuta al menos una vez.

### RO6. Control regresivo

Imprimir  $n, n-1, n-2, \dots, 1, 0$   
 $a_k = n - k$  para  $k = 0, 1, 2, \dots, n$   
esto una en cada paso

```
private static void counterFor (int n) {  
    for (int i = n; i >= 0; i--) {  
        System.out.print(i + " ");  
    }  
}
```

Inicio →   
decremento →   
condición

Para el while es lo mismo solo que declarando i a parte, la condición  $i >= n$  está arriba y el decremento dentro del bloque, y para do-while lo mismo considerando que al menos se ejecutará una vez.