Programación en Java 101 (2)

ÍNDICE:

Programación 101 (2)

1.) Estructuras de selección:

1.1) Estructuras if: Evalúa una condición mediante una expresión booleana. Si se cumple, devolverá lo que se encuentre en su interior.

Existen 3 tipos: Simple, doble y anidada.

1.1) Estructura simple:

```
If (<booleano>) {
        <sentencias>
}
```

```
if (a > b) {
c++;
}
```

1.2) Estructura doble:

```
If (<booleano>) {
        <sentencias1>
} else {
        <sentencias2>
}
```

```
if (a > b) {
        c++;
} else {
        c- -;
}
```

**AÑADE UNA CONDICIÓN DE “SI NO…” a la estructura.

1.3) Estructura anidada:

```
If (<booleano1>) {
        <sentencias1>
} else if (<booleano2>) {
        <sentencias2>
} else if (<booleano3> {
        <sentencias 3>
} else {
        <sentencias4>
}
```

```
if (a > b) {
        mayor = 1;
} else if (a < b) {
        mayor = -1;
} else {
        mayor = 0;
}
```

1.4) Estructura Switch

Evalúa una variable y ejecuta bloques en función del valor que tome.
ES IMPORTANTE PONER EL BREAK para que el código no se ejecute en sucesión.
EL DEFAULT ES TODO AQUELLO QUE OCURRE CUANDO NO SE CUMPLEN LAS CONDICIONES.

```
Switch (<expresion>) {
        case <valor1>:
                <sentencias1>;
                break;
        case <valor2>:
                <sentencias2>;
                break;
        case <valor3>;
                break;
        [default: <sentencia default>;]
}
```

```
int posicion = 4;
switch (posicion) {
        case 1: System.out.println (“ORO”); break;
        case 2: System.out.println (“PLATA”); break;
        case 3: System.out.println(“BRONCE”); break;
        default: System.out.println (“SIN PREMIO”);
        }
```

*La expresión puede ser tipo entero, carácter, boolean, enum, String o clases envoltorio pero NO condiciones, rangos ni listas. Y debe ser del mismo tipo que <expresion>.
DEFAULT es opcional.

2.) Estructuras de Repetición

2.1) While

Mientras se cumpla la condición, se ejecuta el código del bloque.

```
While (<exp_booleana>) {
        <sentencias>;
}
```

```
while (numero <= 10) {
        System.out.println(numero);
        numero++;
}
```

2.2) Do… While

Ejecuta el código, mientras se cumpla la condición dada.

```
Do {
        <sentencias>;
} while (<exp_booleana>)
```

```
do {
        System.out.println(numero);
        numero++;
} while (numero <= 10);
```

2.3) For

El bucle básico que se ejecuta con una inicialización, una expresión de parada y un incremento.

```
For (<inicialización>); <exp_booleana>; <incremento>) {
        <sentencias>;
}
```

Cualquiera de las partes del bucle puede estar vacía. Se ejecutará una vez la inicialización y se repetirá el bucle hasta que la expresión booleana de parada dé como resultado un “false”. El incremento se evalúa antes de evaluar la condición de fin. Suele ser un incremento de la variable índice.

```
For (numero = 1; numero <= 10; numero++) {
        System.out.println(numero);
}
```

3.) Saltos Incondicionales

SE RECOMIENDA NO USARLAS. Provocan una mala estructuración del código e incrementa la dificultad al mantenerlo. Hay que evitar saltar a regiones remotas de código.

3.1) Break y Continue

Break: en estructuras Switch, While, Do-While y For.
        Sale del bloque en el que se encuentra la sentencia.
Continue: en estructuras While, Do-While y For.
        Finaliza la iteración y continúa en la siguiente iteración.

Es decir, Break detiene la ejecución en el lugar donde se encuentra. Continue reinicia la iteración.

3.2) Break y Continue con Etiquetas

En lugar de salir del bucle o saltar a la siguiente iteración, se continuará por la sección del código identificada por la etiqueta.
Únicamente se pueden hacer saltos a bucles dentro de los que nos encontremos.
Las etiquetas se declaran como **nombreEtiqueta:** y deberán formarse en bloque (Con llaves encerrando las sentencias { })