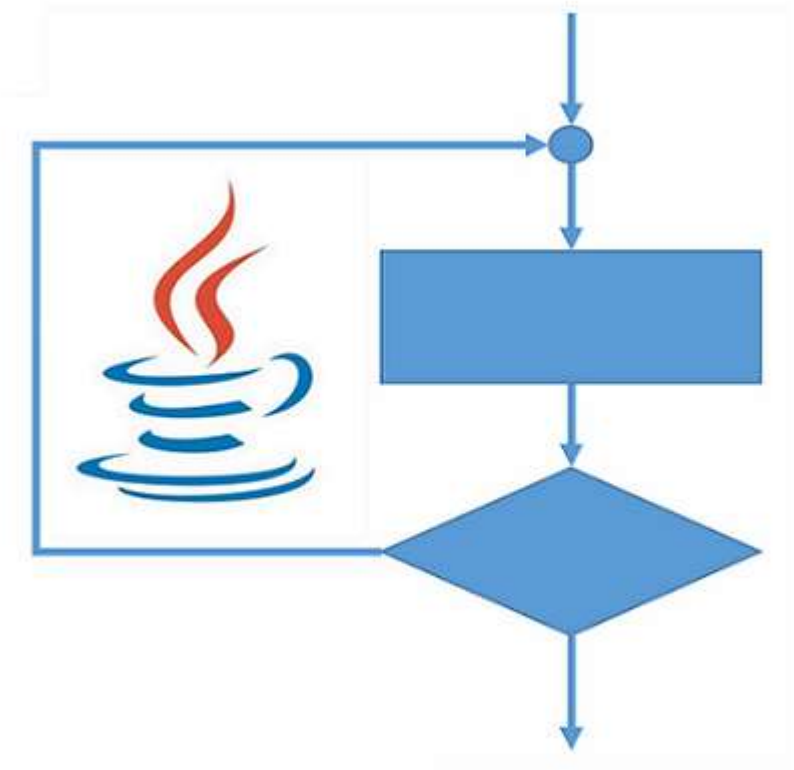
CEPSUNI
www.ceps.uni.edu.pe

FUNDAMENTOS DE PROGRAMACIÓN CON JAVA



UNIDAD 05
ESTRUCTURAS CONDICIONALES

Eric Gustavo Coronel Castillo
youtube.com/DesarrollaSoftware
gcoronel@uni.edu.pe

INDICE

ESTRUCTURA IF SIMPLE **3**
SINTAXIS 3
EJERCICIO 1 3
ESTRUCTURA IF DOBLE **4**
SINTAXIS 4
EJERCICIO 2 4
ANIDAMIENTO DE INSTRUCCIONES IF **5**
EJERCICIO 3 6
EJERCICIO 4 6
ESTRUCTURA SWITCH **7**
SINTAXIS 7
EJERCICIO 5 8
SWITCH EXPRESSIONS **9**
SINTAXIS 9
EJERCICIO 6 9
CURSOS RELACIONADOS **10**

ESTRUCTURA IF SIMPLE

La estructura if en su forma básica es la más simple de todas.

Sintaxis

```
if (condición)
  instrucción;
```

Si **condición** se evalúa como **verdadera**, se ejecuta **instrucción** y después se sigue ejecutando el resto del programa. Si se evalúa como falsa, no se ejecuta **instrucción** y continúa con el resto del programa.

Si lo que se quiere es ejecutar varias instrucciones, estas deben ser consideradas como un bloque, tal como se ilustra a continuación:

```
if (condición)
{
  instrucción_1;
  instrucción_2;
  . . .
}
```

Ejercicio 1

Desarrollar un programa que permita calcular el importe de una venta de un determinado artículo, los datos son: precio, descuento y cantidad.

El descuento solo se aplica si la cantidad es mayor o igual a 6 unidades.

ESTRUCTURA IF DOBLE

A una estructura if se le puede añadir instrucciones que se ejecuten cuando la condición es falsa mediante la cláusula else.

Sintaxis

```
if (condición)
  instrucción1;
else
  instrucción2;
```

Si **condición** se evalúa como verdadera se ejecuta **instrucción1**, en caso contrario se ejecuta **instrucción2**.

Es importante recordar que si lo que se quiere es ejecutar varias instrucciones, estas deben ser consideradas dentro de un bloque, tal como se ilustra a continuación:

```
if (condición)
{
  instruccion1a;
  instruccion1b;
  . . .
} else {
  instruccion2a;
  instruccion2b;
  . . .
}
```

Ejercicio 2

Desarrollar un programa que permita determinar si un número es par o impar.

ANIDAMIENTO DE INSTRUCCIONES IF

Personalmente, recomiendo evitar las estructuras anidadas, dificultan entender el flujo de un código, se debe buscar tener código de fácil lectura.

Las estructuras if se pueden anidar. Es recomendable demarcar claramente los bloques que comprende cada sección de la estructura if para una mayor claridad de la lógica que se está programando.

Por ejemplo, se tienen los siguientes scripts:

Script 1

```
if (condición1)
  if (condición2)
    instrucción1;
  else
    instrucción2;
else
  instrucción3;
```

Script 2

```
if (condición1)
{
  if (condición2)
    instrucción1;
  else
    instrucción2;
} else {
  instrucción3;
}
```

Ambos scripts realizan los mismo:

- instrucción1 se ejecuta cuando condición1 y condición2 son verdaderas.
- instrucción2 se ejecuta cuando condición1 es verdadera y condición2 es falsa.
- instrucción3 se ejecuta cuando condición1 es falsa, no se toma en cuenta el valor de condición2.

Pero en el segundo script se tiene mayor claridad en la lógica del programa.

Ejercicio 3

Desarrollar un programa que permita determinar el mayor de dos números o si son iguales.

Ejercicio 4

Desarrollar un programa que permita calificar la nota de un estudiante según el siguiente cuadro:

NOTA	CALIFICACIÓN
[0,5]	PESIMA
<5,11]	BAJA
<11,15]	REGULAR
<15,18]	BUENA
<18,20]	EXCELENTE

ESTRUCTURA SWITCH

La instrucción switch evalúa una expresión, y en función a su resultado selecciona las instrucciones que serán ejecutadas.

Sintaxis

```
switch (expresión) {
   case <lista de valores>:
     sentencia1;
     [break;]
   case <lista de valores>:
     sentencia2;
     [break;]
   case <lista de valores>:
     sentencia3;
     [break;]
   . . .
   . . .
   [ default:
     sentancia_por_defecto;
     [break;]  ]
}
```

La expresión debe ser de tipo byte, char, short, int o String.

El resultado de evaluar la expresión se compara con cada uno de los casos: valor1, valor2, etc., si coincide con alguno de ellos ejecuta la instrucción asociada a ese caso, finaliza cuando encuentra una sentencia break o llega al final de la instrucción.

La opción default es opcional y solo se ejecuta cuando no se encuentra ninguna coincidencia con algún valor.

A partir de la versión se ha incluido la sintaxis lambda en la estructura switch para mejorar su semántica, revisemos los siguientes ejemplos equivalentes:

Utilizando sintaxis clásica:

```
int x = 1;
switch (x) {
  case 1,2, 3:
    System.out.println("1er trimestre");
    break;
  case 4,5,6:
    System.out.println("2do trimestre");
    break;
  default:
    System.out.println("2do sementre");
    Break;
}
```

Utilizando sintaxis lambda:

```
int x = 1;
switch (x) {
  case 1,2,3 -> System.out.println("1er trimestre");
  case 4,5,6 -> System.out.println("2do trimestre");
  default-> System.out.println("2do sementre");
}
```

Puedes observar que la sintaxis lambda es mucho más sencilla y más fácil de leer.

Ejercicio 5

Desarrollar un programa que simule una calculadora básica. La solución se debe desarrollar utilizando la estructura switch.

SWITCH EXPRESSIONS

Desde el JDK 12 puedes utilizar la sintaxis lambda para obtener un valor especifico según el valor de un selector, pero como expresión.

Sintaxis

```
switch (expresión) {
  case <lista de valores> -> <expresión 1>;
  case <lista de valores> -> <expresión 2>;
  case <lista de valores> -> <expresión 3>;
  default -> <expresión por defecto>;
}
```

Revisemos el siguiente ejemplo:

```
int mes = 11;
String dato = switch (mes) {
  case 1,2,3 -> "1er trimestre";
  case 4,5,6 -> "2do trimestre";
  default-> "2do sementre";
};
System.out.println("Dato: " + dato);
```

Según el valor que tome la variable **mes**, en la variable **dato** se tendrá el valor respectivo.

Ejercicio 6

Desarrollar un programa que permita identificar el trimestre al que pertenece un mes.

CURSOS RELACIONADOS

https://www.ceps.uni.edu.pe/


SQL SERVER
INTELIGENCIA DE
NEGOCIOS
NIVEL IV


Fundamentos
de Programación


PROGRAMACIÓN
ORIENTADA A OBJETOS
CON JAVA


CLIENTE - SERVIDOR


DESARROLLO WEB


FACEBOOK
PARA NEGOCIO