códigofacilito

Java Profesional

Bloque 3: Control de flujo

Tatiana Borda
https://tatianaborda.com/
https://www.linkedin.com/in/tatiana-borda/
https://www.youtube.com/@AlienExplorer

códigofacilito

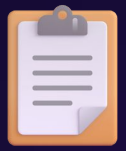

Tabla de contenido

códigofacilito

/01 Condicionales

En programación existen algo llamado las sentencias `if`, `else` y `else if`, estas nos permiten crear estructuras de control para la ejecución condicional del código, es decir, dependiendo del cumplimiento o no de una condición específica que nosotros mismos expresemos. Su objetivo es hacer que los programas sean más flexibles y capaces de responder a diferentes situaciones.
Veamos cada una con un ejemplo!

códigofacilito

Condicionales

El bloque más básico es if

```
if (edad >= 18) {
    System.out.println("Sos mayor de edad");
}
```

También podemos agregar una opción alternativa con else

```
if (edad >= 18) {
    System.out.println("Mayor de edad");
} else {
    System.out.println("Menor de edad");
}
```

códigofacilito

Condicionales

Y para manejar varios casos usamos else-if

```
if (nota >= 9) {
    System.out.println("Excelente");
} else if (nota >= 6) {
    System.out.println("Aprobado");
} else {
    System.out.println("Reprobado");
}
```

códigofacilito

/02 Operador ternario

El operador ternario es una forma concisa de escribir una declaración if-else en una sola línea. Ya lo hemos estudiado en el bloque anterior pero veamoslo en profundidad, se compone de tres partes:

1.La condición a evaluar (condición booleana)
2.El valor si la condición es verdadera (valor verdadero)
3.El valor si la condición es falsa (valor falso)

```
condición ? valorSiVerdadero : valorSiFalso;
```

códigofacilito

Operador ternario ? :

Ejemplo:

```
int edad = 20;
String estado = (edad >= 18) ? "Adulto" : "Menor";
System.out.println(estado); // "Adulto"
```

Es útil para expresiones simples y breves donde se requiere una asignación basada en una condición. Por otro lado, las declaraciones if-else son más adecuadas para situaciones complejas que involucran múltiples condiciones o instrucciones más extensas.

códigofacilito

/03

Switch-case

En programación, un "switch case" o "switch statement" es una estructura de control de flujo que permite elegir entre diferentes bloques de código a ejecutar, dependiendo del valor de una variable o expresión. Es una alternativa a los if/else anidados y hace el código más legible y fácil de mantener.

códigofacilito

Cómo funciona?

1. Evaluación de la expresión: El switch recibe una expresión que se evalúa.

2. Comparación con 'case': el valor de la expresión se compara con los valores de los diferentes 'case' dentro del switch.

3. Ejecución del código: Si se encuentra una coincidencia (el valor de la expresión coincide con el valor de un 'case'), se ejecuta el código dentro de ese 'case'.

4. break (opcional pero necesario): Si se utiliza el break después del código de un 'case', se termina la ejecución del switch. Si no se usa break, el switch continuará ejecutando el código de los 'case' siguientes, incluso si no coinciden.

5. default (opcional): Si no se encuentra ninguna coincidencia con un 'case', se puede definir un bloque 'default' que se ejecutará.

Pero veamos un ejemplo para clarificar su funcionamiento!

Ejemplo:

```java
int dia = 3;
switch (dia) {
    case 1:
        System.out.println("Lunes");
        break;
    case 2:
        System.out.println("Martes");
        break;
    case 3:
        System.out.println("Miércoles");
        break;
    default:
        System.out.println("Día inválido");
}
```

códigofacilito

/04 Ciclos:

En programación, un ciclo (también conocido como bucle) es una estructura que permite ejecutar repetidamente un bloque de código hasta que se cumpla una condición específica. Los ciclos son fundamentales para automatizar tareas repetitivas y mejorar la eficiencia del código. Nos evitan la necesidad de escribir el mismo código repetidamente, nos permite realizar tareas complejas sin necesidad de intervención manual y facilitan la comprensión del código.

códigofacilito

Ciclo for

Uso: Cuando sabes de antemano cuántas veces se repetirá el código
Sintaxis:

```
for (inicialización; condición; actualización) {
    // Código a repetir
}
```

Ejemplo:

```
for (int i = 1; i <= 5; i++) {
    System.out.println("Iteración " + i);
}
```

Salida:
Iteración 1
Iteración 2
Iteración 3
Iteración 4
Iteración 5

códigofacilito

Ciclo while

Uso: Cuando no conoces cuántas veces se repetirá el código, pero sabes la condición de parada.

Sintaxis:

```
while (condición) {
    // Código a repetir
}
```

Ejemplo:

```
int contador = 1;
while (contador <= 5) {
    System.out.println("Contador: " + contador);
    contador++;
}
```

Salida:
Contador: 1
Contador: 2
Contador: 3
Contador: 4
Contador: 5

códigofacilito

Ciclo do-while

Uso: Similar al while, pero garantiza al menos una ejecución (la condición se evalúa al final)

Sintaxis:

```
do {
    // Código a repetir
} while (condición);
```

Ejemplo: En este caso el programa pide un número hasta que el usuario ingrese uno positivo.

```
int numero;
do {
    System.out.print("Ingresa un número positivo: ");
    numero = new Scanner(System.in).nextInt();
} while (numero <= 0);
```

códigofacilito

/05 Control de Ciclos: break y continue

En Java, `break` y `continue` son palabras clave utilizadas para controlar el flujo de ejecución dentro de los bucles (loops). break termina el bucle actual, mientras que continue salta la iteración actual y continúa con la siguiente.

códigofacilito

break

Esta palabra le indica al programa que debe salir del bucle en el que se encuentra
Es comúnmente utilizado para terminar la ejecución de un bucle cuando una condición específica se cumple.

Salida:
0
1
2
3
4

```java
for (int i = 0; i < 10; i++) {
    if (i == 5) break;
    System.out.println(i);
}
```

códigofacilito

continue

Esta palabra le indica al programa que debe saltar la iteración actual del bucle y continuar con la siguiente iteración.
No termina el bucle, sino que se salta la ejecución de las instrucciones que están después de continue en la iteración actual.
Es útil para omitir ciertas partes de un bucle bajo ciertas condiciones

```java
for (int i = 0; i < 5; i++) {
    if (i == 2) continue;
    System.out.println(i); // se saltea el 2
}
```

Salida:
0
1
3
4

códigofacilito

/06 HANDS ON

A codear, vamos al editor de código!

REPOSITORIO CON EL CÓDIGO:
https://github.com/tatianaborda/java-course

códigofacilito

Objetivo:

Repetir el menú mientras el usuario no elija salir

Usar condicionales y ciclos para manejar opciones del menú

Temas aplicados del bloque:

if, else, else if	Validar prioridad
switch-case	Elegir acción según opción del menú
do-while	Repetir el menú hasta que se elija salir