CONDICIONALES

CONDICIONALES

Los condicionales son instrucciones de Java que en función de la evaluación de una condición booleana ejecutará una acción (o serie de acciones) u otra alternativa.

Siempre tendremos una estructura, no idéntica, pero sí muy parecida a:

```
if (condicional_booleana) {
    bloque_true
} else {
   bloque_false
}
```

En función de si la condicional_booleana se evalua como true o como false, nuestro programa realizará las acciones del bloque_true u las acciones del bloque_false.

De forma general se diferencian: condicionales simples, condicionales dobles y condicionales múltiples.

CONDICIONALES ÚNICOS Y DOBLES

Los condicionales simples nos van a permitir realizar una u otra acción en base a una única condición. Son de dos tipos:

- **Condicionales únicos (if).** En estos condicionales sólo vamos a tener una sentencia de tipo if sin que vaya acompañada de un tipo else. La construcción sintáctica de este tipo de condicionales es la siguiente:

  ```
  if (condicion) {
        instrucción/es;
  }
  ```

  Esta condición siempre va a ser una condición de comparación, es decir, siempre va a haber que usar los operaciones de comparación (<, <=, >, >=, != o ==) y puede ser simple o compuesta.

  - Operación simple:

    ```
    if (intNum > 10) {instrucciones;}
    ```

    En este caso si el número es mayor de 10 se realizarán X instrucciones.

- Operación compuesta: En este caso tendremos 2 o más operaciones de comparación unidas por operadores lógicos (&& y ||):

  ```
  if (intNum > 10 && intNum < 20) {instrucciones;}
  ```

  En este caso el if se activará si el número es mayor de 10 y, además, es menor de 20.

Por ejemplo:

```java
public class CondicionalSimple {
    public static void main(String[] args) {
        Scanner scEntrada = new Scanner(System.in);
        int intNumUsuario;
        System.out.print("Introduzca un numero: ");
        intNumUsuario = scEntrada.nextInt();
        scEntrada.close();
        if(intNumUsuario % 2 != 0){
            intNumUsuario += 1;
            /*************************************************************
             * OTRAS FORMAS DE EXPRESAR ESTA INSTRUCCION:                *
             *  a) ++intNumUsuario (se suma uno antes de su siguiente uso)  *
             *  b) intNumUsuario = intNumUsuario + 1                     *
             *************************************************************/
        }
        System.out.println(intNumUsuario);
    }
}
```

Tenemos un programa que va a solicitar un numero a un usuario y si este es par lo imprimirá directamente por consola y si es impar (`intNumUsuario % 2 != 0` -el módulo (resto) de la división del número del usuario entre 2 da un resultado diferente a 0-) le sumará 1 para convertirlo en par y, a continuación, lo imprimirá por consola.

Es decir, si introducimos un par este será el resultado por consola:

```
Introduzca un numero: 4
4
```

Y si introducimos un número impar este será el resultado por consola:

```
D:\JDK\bin\java.exe "-jav
Introduzca un numero: 3
4
```

- **Condicionales dobles (if-else).** En estos condicionales vamos a tener un elemento if que va a ir acompañado de un else. A diferencia del anterior aquí vamos a tener instrucciones en ambos elementos. En este caso la construcción sintáctica será la siguiente:

```
if (condicion) {
        instrucción/es;
}else {
        Instrucción/es;
}
```

Por ejemplo:

```java
public class CondicionalSimple {
    public static void main(String[] args) {
        Scanner scEntrada = new Scanner(System.in);
        int intNumUsuario;
        System.out.print("Introduzca un numero: ");
        intNumUsuario = scEntrada.nextInt();
        scEntrada.close();
        if(intNumUsuario % 2 == 0){
            System.out.println("Has introducido un numero par.");
        }else{
            ++intNumUsuario;
            System.out.println("Has introducido un numero impar, el siguiente par es el: " + intNumUsuario + ".");
        }
    }
}
```

El programa es muy similar al anterior, volvemos a pedir un número al usuario y valoramos si ese número es par (intNumUsuario % 2 == 0), en cuyo caso se seguirá la instrucción incluida en el elemento if (informar de que ha introducido un par) en caso contrario seguirá las instrucciones incluidas en el elemento else (aumentar en 1 el número introducido e informar al usuario de que ha introducido un impar y de cual sería el siguiente par).

El resultado por consola si introducimos un par es este:

```
D:\JDK\bin\java.exe "-javaagent:D:\Ir
Introduzca un numero: 4
Has introducido un numero par.
```

En cambio, si introducimos un impar tendremos este resultado:

```
D:\JDK\bin\java.exe "-javaagent:D:\IntelliJ\IntelliJ IDEA Commu
Introduzca un numero: 3
Has introducido un numero impar, el siguiente par es el: 4.
```

CONDICIONALES DOBLES ANIDADOS

En los condicionales dobles vamos a tener varios bloques if/else de forma continua, para evitar la creación de varios bloques anidados que aumentaría la complejidad de lectura se utiliza el elemento else if. Podemos tener tantos else if como necesitemos para completar todas las condiciones posibles:

Su sintaxis es la siguiente:

```
if (condicion) {
        instrucción;
} else if (condicion) {
        instrucción;
} else if (condicion) {
        instrucción;
} else {
        instrucción;
}
```

Esto sería igual que escribir este código:

```
if (condicion) {
instrucción;
} else {
        if (condicion) {
                instrucción;
        } else
                if (condicion) {
                        instrucción;
                } else {
                        instrucción;
                }
```

Como se puede ver la complejidad de código y de entendimiento del código es mayor (además de ocupar más líneas de código).

Por ejemplo:

```java
public class CondicionalesDobles {
    public static void main(String[] args) {
        int intNum = 100;
            if (intNum >= 0 && intNum < 10){
                System.out.println("El numero tiene 1 cifra.");
            } else if (intNum >= 10 && intNum < 100) {
                System.out.println("El numero tiene 2 cifras.");
            } else if (intNum >= 100 && intNum < 1000) {
                System.out.println("El numero tiene 3 cifras.");
            } else {
                System.out.println("El numero tiene mas de 3 cifras");
            }
    }
}
```

En este caso tenemos un programa para calcular tiene un numero dado (en este ejemplo está fijado en la declaración de la variable), la variable intNum entra en el primer if y se comprueba si es mayor o igual a 0 y, además, menor a 10, en este caso no lo es por lo que sigue con el siguiente if y se comprueba si es mayor o igual a 10 y, además, menor a 100, vuelve a dar un false como respuesta por lo que pasa al tercer if, en este comprobará si el número es mayor o igual a 100 y menor a 100, lo es así que se imprimirá el mensaje asociado a este condicional.

SWITCH

En algunas ocasiones el uso de condicionales dobles anidados enlentecen el código o lo hacen demasiado largo, para evitar esto tenemos los Switch o condicionales múltiples. Este tipo de condicionales se organizan en forma de minicondicionales llamados case que funcionan a modo de opción, si el valor de la condicion coincide con el valor del case, se activan sus instrucciones asociadas.

Además pueden incluir aunque de forma opcional una opción default que se activa si ninguno de los valores de los case coincide con el valor de la condición.

En este caso tenemos varias opciones de sintaxis en función del número de instrucciones que haya que realizar.

- Opción 1: Se puede usar siempre, haya una instrucción o varias por cada case.

```
switch (opcion){
        case 1:
```

```
            instrucciones;
            break;
        case 2:
            instrucciones;
            break;
        case 3:
            instrucciones;
            break;
        ...
        Case n:
            instrucciones;
            break;
        default:
            instrucciones;
}
```

La sentencia "break" es necesaria ya que si no se escribiera el código se ejecutaría desde el case de entrada (el que coincida con la opción) hasta el final.

Además, si 2 o más cases van a producir la misma instrucción se pueden incluir en el mismo case separando cada valor con coma por ejemplo:

```
case 1,2,3,4:
        instrucciones;
        break;
```

Un ejemplo de este tipo de Switch sería el siguiente:

```java
public class Switch {
    public static void main(String[] args) {
        Scanner scEntrada = new Scanner(System.in);
        int intNota;

        System.out.print("Introduzca nota: ");
        intNota = scEntrada.nextInt();

        switch (intNota){
            case 0,1,2,3,4:
                System.out.println("SUSPENSO");
                break;
            case 5:
                System.out.println("SUFICIENTE");
                break;
            case 6:
                System.out.println("BIEN");
                break;
            case 7,8:
                System.out.println("NOTABLE");
                break;
            case 9,10:
                System.out.println("SOBRESALIENTE");
                break;
            default:
                System.out.println("NOTA NO VALIDA");
        }
    }
}
```

En este programa el usuario va a introducir una nota y el programa va a determinar si esa nota es suspenso (del 0 al 4), si es suficiente (5), si es bien (6) si es notable (7 u 8) o si es sobresaliente (9 o 10). Si el usuario mete una nota no válida se ejecutaría la instrucción del default para informar que la nota no es válida.

- Opción 2: Sólo la vamos a poder usar cuando tengamos un switch cuyos cases sólo tengan una instrucción en su interior. En este caso los break no se deben escribir, con esta sintaxis el programa entiende que sólo debe ejecutar una instrucción y, por tanto, ejecuta la instrucción asignada al case pero ninguna otra.

Al igual que en el anterior se pueden agrupar cases que produzcan el mismo resultado en una única sentencia y el default es opcional.

En este caso la sintaxis será la siguiente:

```
Switch (opcion) {
        case 1 -> instruccion;
        case 2 -> instruccion;
        case 3 -> instruccion;
        ...
        Case n -> instruccion;
        default -> instruccion;
}
```

Utilizando el mismo ejemplo que arriba tendremos el siguiente código:

```java
public class Switch {
    public static void main(String[] args) {
        Scanner scEntrada = new Scanner(System.in);
        int intNota;

        System.out.print("Introduzca nota: ");
        intNota = scEntrada.nextInt();

        switch (intNota){
            case 0,1,2,3,4 -> System.out.println("SUSPENSO");
            case 5 -> System.out.println("SUFICIENTE");
            case 6 -> System.out.println("BIEN");
            case 7,8 -> System.out.println("NOTABLE");
            case 9,10 -> System.out.println("SOBRESALIENTE");
            default -> System.out.println("NOTA NO VALIDA");
        }
    }
}
```

OPERADOR TERNARIO

Por último, hay que destacar la existencia del llamado operador ternario que es un condicional doble, pero con una sintaxis diferente.

Su funcionamiento es idéntico al de un condicional doble y se utiliza principalmente cuando tenemos una única instrucción en el lado del if y una única instrucción en el lado del else.

Tiene la siguiente sintaxis:

```
condicion ? instruccionTrue : instruccionFalse;
```

Por ejemplo, vamos a crear un programa que decida si un programa es par o no y que informe al usuario con un true (par) o false (impar):

```java
public class OperadorTernario {
    public static void main(String[] args) {
        Scanner scEntrada = new Scanner(System.in);
        boolean blnPar;
        int intUsuario;

        System.out.print("Introduzca un numero: ");
        intUsuario = scEntrada.nextInt();

        blnPar = (intUsuario % 2 == 0) ? true : false;

        System.out.println("El numero " + intUsuario + " es par? " + blnPar);
    }
}
```

El operador ternario en este caso lo guardamos en una variable de tipo boolean la cual dará ese true/false.

Si introducimos un número par obtendremos este resultado:

```
D:\JDK\bin\java.exe "-javaagent:D
Introduzca un numero: 4
El numero 4 es par? true
```

Si introducimos un número impar obtendremos este otro resultado:

```
D:\JDK\bin\java.exe "-javaa
Introduzca un numero: 5
El numero 5 es par? false
```

En estos casos el operador ternario lo que hace es activarse y dar una respuesta true (en el caso del 4) o false (en el caso del 5) que se guarda en la variable booleana `blnPar` que luego se imprime por consola.