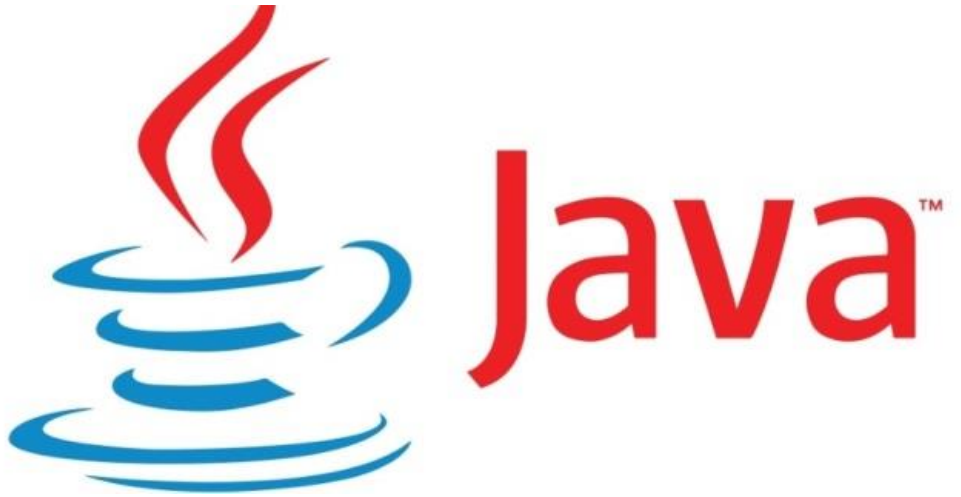
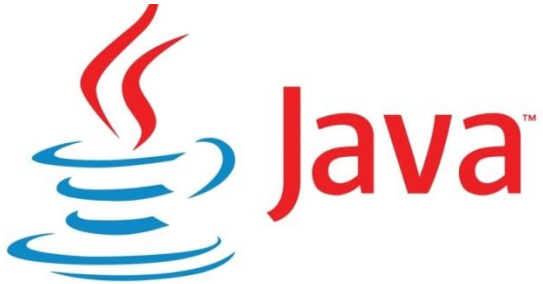


Java 8 SE Fundamentals

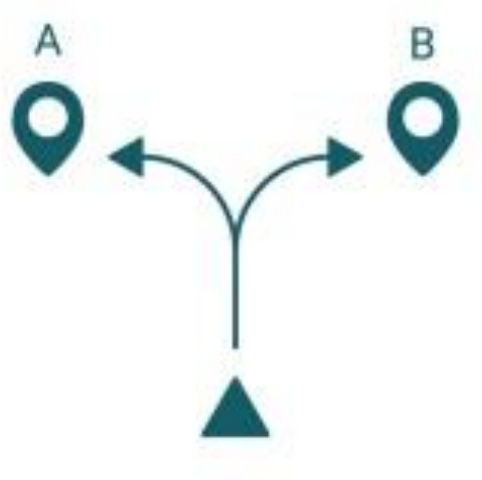


Manipulating multiple items

- ✓ Working with Conditions
- ✓ Working with a List of Items
- ✓ Processing a list of items

¿Qué es un condicional en Java?

Son estructuras de control por selección, nos permiten generar diferentes flujos en la ejecución de un programa.

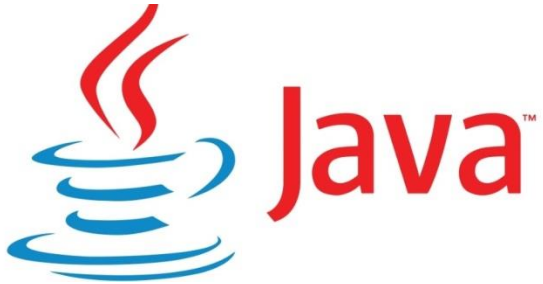


Condicional if

Utilizamos la sentencia **if** para especificar un bloque de código Java que se ejecutará si una condición es verdadera.

```
8       if(condicion){
9           //Instrucciones a ejecutar
10          System.out.println("Hola");
11      }
```

Hagamos algunos ejemplos.



Ejercicios.

1. Pedir al usuario una variable de tipo numérica (int o double) y verificar si su valor está en los número positivos e imprimirlo en consola.

2. Crear dos variables y compararlas para ver cuál es mayor.

Sentencia else (Complemento de if).

Utilice la sentencia **else** para especificar un bloque de código que se ejecutará si la condición es falsa.

```
8       if(condicion){
9           System.out.println("Hola");
10      }else{
11          System.out.println("Adiós");
12      }
```

Ejercicios.

1. Crear un programa en Java que pida un número al usuario y diga si es múltiplo de 7 o si no lo es.

2. Crear un programa en Java que simule un volado de una moneda.

Estructura de control: if- else if - else.

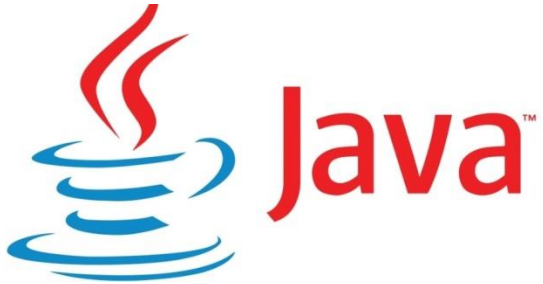
Utilice la sentencia else if para especificar una nueva condición si la condición anterior es falsa.

Hagamos más ejemplos.

Ejercicios.

1. Crear una variable que guarde un valor y verificar si el valor es positivo, negativo o exactamente cero.

2. Hacer un programa que pida una letra y a continuación diga si ésta es vocal o no lo es.



Operadores lógicos.

Operador	Nombre
&&	AND
||	OR
!	NOT

Hagamos sus tablas de verdad.

Ejercicios.

1. Crear un programa en Java que pida un número entero e imprima: “dentro del rango” si el número es 2,3,4 o 5. En caso contrario que imprima fuera de rango.

2. Crear un programa similar que pida un número entero y verifique si está en el rango 5,6,7 o 10,11,12.

Estructura de control: switch.

Utilice la instrucción switch para seleccionar uno de los muchos bloques de código que se ejecutarán.

Veamos los elementos de la estructura switch.

Switch es útil cuando la selección se basa en el valor de una variable o una expresión simple llamada selector.

Selector sólo puede ser de tipo **int** o **char**.

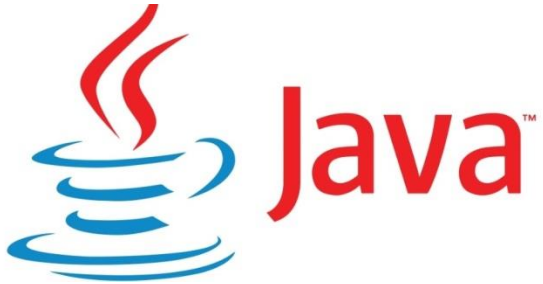
Sintaxis:

```
switch(selector)
{
   case etiqueta1: sentencia1; break;
   case etiqueta2: sentencia2; break;
   .
   .
   .
   case etiqueta_n: sentencia_n; break;
   default: sentencias_d; /*Opcional*/
}
```

etiqueta es un valor único, constante y todas las etiquetas deben ser diferentes.

Break termina la ejecución del switch.

default ejecuta una sentencia en caso de que se introduzca un valor de *selector* no incluido.



Ejercicios.

1. Hacer programa en Java con un menú que nos permita elegir entre 4 opciones.

2. Hacer un programa en Java que pida una letra y a continuación diga si ésta es vocal o no lo es.

3. Crear un programa en Java que simule el comportamiento de un dado.

Estructura de control: for.

Recorre un bloque de código varias veces

Inicializa la variable de control del bucle.

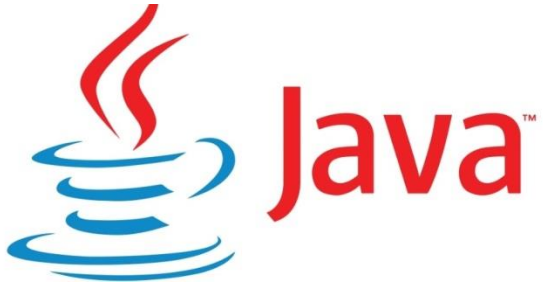
Expresión lógica que determina si han de ejecutar mientras sea verdadera.

```
for(int i=0; i<10; i++){
    System.out.println("Hola");
}
```

Sentencias a ejecutar en cada iteración del bucle.

Incrementa o decrementa la variable de control del bucle.

Hagamos algunos ejemplos.



Ejercicios.

1. Crear un bucle for en Java que cuente en forma descendente del 10 al 0.

2. Crear un bucle for en Java que imprima las potencias de 2 del exponente 0 al 10.

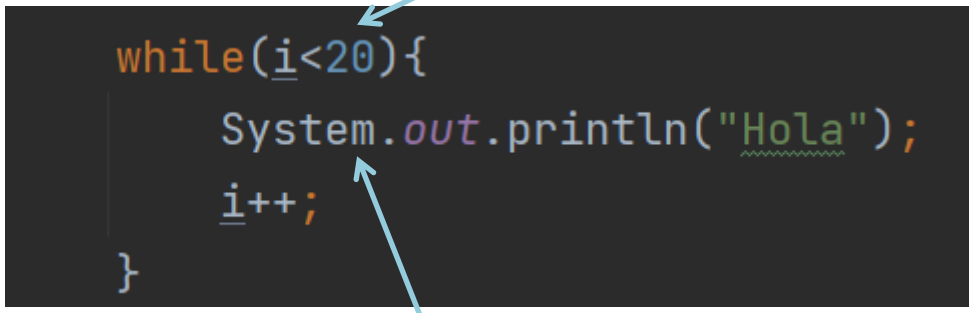
3. Crear un programa en Java que pida un número entero y calcule su factorial con un bucle for.

4. Crear un programa en Java que pida un número entero N e imprima una matriz de NxN de números enteros aleatorios entre 10 y 50 en la consola.

Estructura de control: while.

La sentencia while ejecuta un bloque de código siempre que se cumpla una condición específica.

Condición especifica que provocará que termine el bucle.



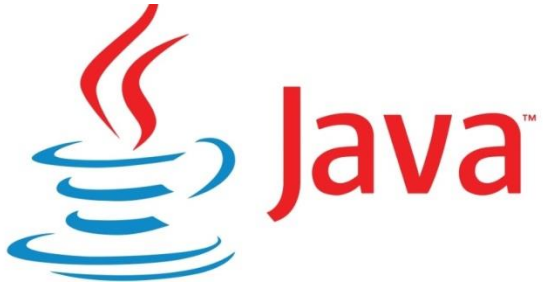
Sentencias a ejecutar en cada iteración del bucle.

Hagamos un ejemplo.

Ejercicios.

1. Crear un ciclo while con Java que haga la suma de los números del 1 al 100.

2. Hacer un programa en Java que pida un número entero e imprima un triangulo usando while anidados.



Estructura de control: do-while.

El bucle do while es una variante del bucle while.

Este ciclo ejecutará el bloque de código una vez, antes de verificar si la condición es verdadera, luego repetirá el ciclo mientras la condición sea verdadera.

Tip: podemos usarlo para filtrar numéricos dado un rango determinado.

Estructura de control: do-while.

Sentencias a ejecutar al menos una vez antes.

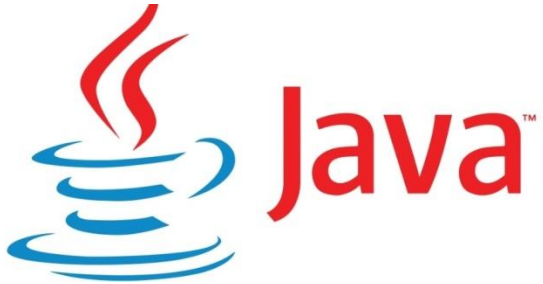
```
do{
    //Intrucciones
    System.out.println("Hola");
}while(x<0);
```

Condición especifica que provocará que termine el bucle, será evaluada después de ejecutar una vez el bloque.

Ejercicios.

1. Crear un filtro en Java para números enteros entre [1,5]

2. Crear un filtro en Java para números enteros que sólo acepte los siguientes valores: 1,2,3,7,8,9.



Interrumpir, continuar o abandonar un bucle.

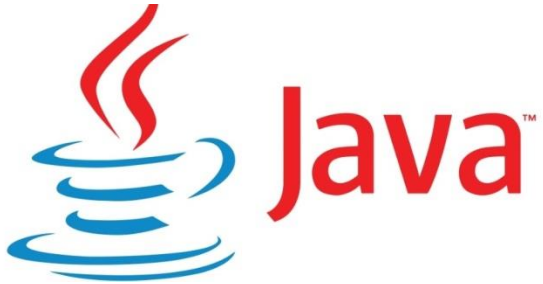
Break es la instrucción que salta de un bucle.

Continue es la instrucción que salta una iteración en el bucle.

Hagamos un ejemplo de cada uno.

Ejercicio.

1. Hacer un bucle while que imprima los números del 1 al 10 excepto el 4 y el 9.



Trabajando con una lista de elementos.

La List nos ofrece una amplia gama de implementaciones. La más común es ArrayList.

El usuario de esta interfaz tiene un control preciso sobre en qué parte de la lista se inserta cada elemento.

El usuario puede acceder a los elementos por su índice entero (posición en la lista) y buscar elementos en la lista.

Trabajando y procesando un ArrayList.

Consultemos la API de Java para conocer ArrayList:

https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

Trabajando y procesando un ArrayList.

Estructura de control: foreach.

Esta es la estructura ideal para ArrayList se itera de forma automática.

Se declara una variable temporal para asignarle con cada iteración el valor de un elemento de la lista.

Nombre del ArrayList del cuál se toman los elementos.

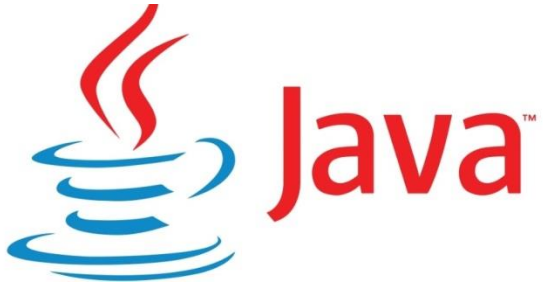
```
for(String nombre : nombres){
    System.out.println(nombre);
}
```

Sentencias a ejecutar en cada iteración del bucle.

Hagamos un ejemplos.

Ejercicio.

1. Crear un ArrayList en Java que guarde elementos de tipo Double, leer los valores desde el teclado, tanto el número de elementos como los valores. Buscar el elemento menor de la lista e imprimir su valor y su índice.



Ejercicio.

2. Crear un ArrayList en Java que guarde elementos de tipo Double, leer los valores desde el teclado, tanto el número de elementos como los valores. Ordenar los elementos de la lista con el siguiente algoritmo:

ALGORITMO:

1. Crear una lista vacía `ArrayList<Double>` llamada la lista final
2. Mientras la lista original tenga elementos
3. Buscar el elemento menor de la lista original (índice y valor)
4. Agregar el elemento menor en la lista final
5. Eliminar el elemento menor de la lista original
6. Imprimir la lista final que es la lista ya ordenada