FUNDACIÓN ACCENTURE accenture

JAVA I

1. CONCEPTOS AVANZADOS DE JAVA

Adéntrate en los conceptos más avanzados de Java, como los tipos de operadores y las diferentes estructuras de control.

TABLA DE CONTENIDOS

JAVA I

1.1. TIPOS DE OPERADORES

¿QUÉ ES UN OPERADOR?

Los operadores permiten realizar operaciones entre los tipos de datos básicos: suma, resta, producto, cociente, etc., de dos números. También permiten concatenar palabras, es decir sumar caracteres (Ej. CASA y GRANDE = CASA GRANDE).

TIPOS DE OPERADORES

Existen diferentes tipos de operadores

- Operador de asignación
- Operadores aritméticos
- Operadores combinados
- Operadores unarios
- Operadores relacionales
- Operadores lógicos
- Operador lógico ternario

1. OPERADOR DE ASIGNACIÓN

Sirve para asignar un valor a una variable, usa el signo igual (=).

Operador	Significado	Ejemplos

=	igual	Int edad = 3;
		String saludo = "Hola Mundo";
		double nota = 4.5;
		boolean aprobado = true;

→ Recuerda que usaremos int para declarar las variables de tipo número entero edad, y double para las de tipo decimal. Además usaremos String para las variables de tipo texto y boolean para true/false.

2. OPERADORES ARITMÉTICOS

Sirve para realizar operaciones matemáticas: suma resta, multiplicación y división.

Operador	Significado	Ejemplo	Resultado
+	suma	int resultado = 3 + 4;	7
-	resta	int resultado = 7 - 2;	5
*	multiplicación	int resultado = 3 * 2;	6
/	división	int resultado = 4 / 2;	2
%	resto*	int resultado = 20 % 7;	6

→ Ten en cuenta que usaremos int para resultados de cálculos con números enteros y double para resultados de cálculos con números decimales.

Además, el operador suma (+) puede usarse también para cadenas de texto: String saludo = a + b , donde a=”Buenos ” y b=”días.” Entonces saludo es igual a "Buenos días."

*El operador % devuelve el resto de la división entera.

3. OPERADORES ARITMÉTICOS COMBINADOS

Permiten combinar un operador aritmético con el operador de asignación.

Operador	Significado	Expresión	Significado expresión	Ejemplo	Resultado Si x=5; y=10
+=	suma combinada	x+=y;	x=x+y;	x=5+10;	15
-=	resta combinada	x-=y;	x=x-y;	X= 5-10;	-5
*=	multiplicación combinada	x*=y;	x=x*y;	X=5*10;	50
/=	división combinada	x/=y;	x=x/y;	X=5/10;	0,5
%=	resto combinado	x%=y;	x=x%y;	X= 5%10;	5

4. OPERADORES UNARIOS

Actúan sobre un único operando y sirven para incrementar y decrementar en uno el valor.

Operador	Significado	Expresión	Ejemplo	Resultado Si y=0; x=5;
++	Preincremento postincremento	Variable = ++ contador Variable = contador++	int y = ++x; int y = x++;	Y=6 Y=5
--	Predecremento postdecremento	Variable = -- contador Variable = contador--	int y = --x; int y = x--;	Y=4 Y=5

5. OPERADORES RELACIONALES

Sirven para realizar comparaciones entre tipos primitivos con resultado true o false.

Operador	Significado	Ejemplo	Resultado Si x=3; y=2; z=3
==	Igual que	x==y	false
!=	Distinto que	x!=y	true
>	Mayor que	x>y	true
<	Menor que	x<y	false
>=	Mayor o igual que	x>=y	true
<=	Menor o igual que	x<=z	true

→ Recuerda que
true = verdadero
false = falso

6. OPERADORES LÓGICOS

Se usan para comparar variables o expresiones y el resultado será true o false.

Operador	Significado	Expresión	Ejemplo	Resultado
||	OR: si al menos uno de los dos tiene valor true el resultado de la evaluación será true. Si encuentra un valor true, ya no sigue evaluando, el resultado será true	true || true true || false false || true false || false	(5==5)||(5==5) (5==5)||(5<4) (5<4)||(5=5) (5<4) || (5<4)	true true true false
&&	AND: si todos los valores son true, el resultado será true. Si encuentra un valor false, ya no sigue evaluando, el resultado será false	true && true true && false false && true false && false	(5==5)&&(5==5) (5==5)&&(5<4) (5<4)&&(5=5) (5<4)&&(5<4)	true false false false
!	Negación: devuelve el valor invertido	!true !false	! (5==5) ! (5<4)	false true
^	XOR: el OR exclusivo es verdadero cuando los dos tiene un valor diferente, uno es true y el otro es false	true ^ true true ^ false false ^ true false ^ false	(5==5)^(5==5) (5==5)^(5<4) (5<4)^(5==5) (5<4)^(5>4)	false true true false
|	OR: si al menos uno de los dos tiene valor true el resultado de la evaluación será true. Siempre evalúa todos los operadores	true | false	(5==5) | (5< 4)	true
&	AND: si todos los valores son true, el resultado será true. Siempre evalúa todos los operadores	true & false	(5==5)&(5<4)	false

¿Sabías que los operadores **&&** y **||** conocidos también como operadores de circuito corto, mejoran el rendimiento del programa? Esto se debe a que dependiendo del resultado de la primera expresión podemos evitarnos evaluar la segunda.

7. OPERADOR LÓGICO TERNARIO

Utiliza la lógica de la condicional, nos sirve para asignar un valor a una variable dependiendo del resultado de una comparación. En función del resultado de la expresión lógica, devolverá un valor u otro.

Operador	Significado	Expresión	Ejemplo	Resultado Si x= 50
?	Si la expresión es Verdadera devuélveme el valor1, y si es Falsa el valor2	(condición)?valor1:valor2;	(x<100)?10:20	10

1.2. ESTRUCTURAS DE CONTROL

SENTENCIAS DE CONTROL DE FLUJO

Un programa en Java se ejecuta en orden desde la primera sentencia hasta la última, aunque existen las **sentencias de control de flujo**, las cuales permiten alterar el flujo de ejecución para tomar decisiones o repetir sentencias.

TIPOS DE ESTRUCTURAS O SENTENCIAS DE CONTROL

- Sentencias condicionales
- Sentencias repetitivas o bucle
- Sentencias anidadas

1. SENTENCIAS CONDICIONALES

→ SENTENCIA TIPO IF

Ejecutarán las instrucciones en función de la **condición** definida.

Un ejemplo... Todos sabemos que si tienes 18 años eres mayor de edad y tendrás algún tipo de privilegio. La condición es tener 18 años, si no eres menor de edad.

¿Cómo se declara en Java?

Tenemos un programa de tratamiento de personas. Queremos indicarle que cuando la persona tiene 18 años le trate como “mayor de edad”.

Para ello, en primer lugar, aparece el condicional **if (si...),** y seguido de esto indicamos la condición que queramos poner para que se ejecute la sentencia que va a continuación. Opcionalmente se puede añadir el bloque **else** (sentencias que sólo se realizan cuando no se cumple la condición if).

```
if (edad>=18) {
     System.out.println(“Eres mayor de edad”);

} else {

     System.out.println(“Eres menor de edad");
}
```

SENTENCIA TIPO SWITCH..CASE

Es una **instrucción de múltiples vías** (casos). Proporciona una forma de enviar a ejecución diferentes partes del código en función del valor de la expresión.

El valor de la expresión debe ser de tipo **char, byte, short** o **int** y los valores de los diferentes casos deben ser únicos. Después de cada caso se usa la palabra **break**. Opcionalmente, se puede añadir el bloque **default** (sólo se ejecutará si no se cumple ninguno de los casos anteriores).

Un ejemplo... Todos sabemos que hay 7 días a la semana y cada día tiene asignado un nombre (lunes, martes...).

¿Cómo se declara en Java?

Tenemos un programa de tratamiento de días de la semana. Queremos que nos indique el **nombre del día de la semana** en el que nos encontramos cuando introduzcamos el **número de día** de la semana.

En primer lugar, hemos introducido el día en el que estamos **(int day = 5)**. A continuación, aparece la instrucción **switch** y seguido de esto los diferentes casos para cada uno de los 7 días (finalizando cada uno con un **break**). Opcionalmente, hemos añadido el bloque **default** por si una persona mete un número que no esté comprendido entre 1 y 7. En nuestro ejercicio, irá comprobando uno a uno hasta encontrar el caso 5 y devolvernos "Viernes".

```
public class DiasSemana {
  public static void main(String[] args){

    int day = 5;
    String dayString;

    switch (day) {
     case 1: dayString = "Lunes";
         break;
     case 2: dayString = "Martes";
         break;
     case 3: dayString = "Miércoles";
         break;
     case 4: dayString = "Jueves";
         break;
     case 5: dayString = "Viernes";
         break;
     case 6: dayString = "Sabado";
         break;
     case 7: dayString = "Domingo";
         break;
     default: dayString = "Día inválido";
         break;
    }
    System.out.println(dayString);
  }
}
```

2. SENTENCIAS REPETITIVAS O DE BUCLE

→ Sentencia tipo FOR

Se ejecutarán las **sentencias repetidamente** un número de veces, basándose en una **condición**. La condición se evaluará antes de cada iteración.

Un ejemplo... Todos sabemos los números y sabemos contar en orden del 1 al 10 por ejemplo.

¿Cómo se declara en Java?

Tenemos un programa que cuenta una misma sentencia 10 veces hasta devolvernos los números del 1 al 10 en orden.

En primer lugar, hemos introducido la función **FOR** y entre paréntesis hemos puesto (valor inicial; condición; expresión). En nuestro caso el **valor inicial es 1**, la condición es que sea **menos que 11**, y la expresión será el operador que sume 1 unidad más cada vez. Finalmente decimos que, mientras se cumpla la condición, **se imprima en pantalla el valor de la variable i**. En consola se mostrará: Número: 1, Número: 2, Número: 3... Número: 10.

```
class Contar {
  public static void main(String[] args) {

    for (int i=1; i<11; i++) {
      System.out.println("Número: " + i);
    }
  }
}
```

Ten en cuenta que...
Puedes inicializar la variable desde cualquier número entero (-10, 0, 23...). La variable puede aumentar o disminuir de valor y puede hacerlo de uno en uno, dos en dos, tres en tres...

→ Sentencia tipo WHILE

Se ejecutarán y **repetirán las sentencias mientras la condición sea verdadera**. Dentro del bucle, se han de actualizar los valores que se usan para la condición, si no se daría el caso de un bucle infinito. Si el resultado es false las instrucciones no se ejecutan y el programa seguirá ejecutándose por la siguiente instrucción a continuación del **while**.

Un ejemplo... Todos sabemos cuáles son los números positivos y negativos.

¿Cómo se declara en Java?

Tenemos un programa que nos pide introducir un número y espera que el valor sea positivo. En el caso contrario nos hará repetir la introducción del dato.

En primer lugar, hemos aplicado un método de escáner de teclado para que **el usuario introduzca el número**. A continuación, la función **while** que evalúe si el número cumple la condición de ser menor que cero, y en ese **caso imprimir la frase "introduce un número positivo"**. El mensaje se repetirá todas las veces que el usuario meta un número negativo (por ejemplo: -1)

```
public int leerNumero() {
   Scanner sc = new Scanner(System.in);
   int numero = sc.nextInt();

   while (numero <= 0) {
      System.out.println("Introduce un numero positivo: ");
      numero = sc.nextInt();
   }
}
```

→ Sentencia tipo DO WHILE

Igual que el anterior (WHILE), la diferencia es que aquí primero se ejecuta la condición y luego se evalúan las sentencias.

Un ejemplo... Todos sabemos cuáles son los números pares... 0,2,4,6,8...

¿Cómo se declara en Java?

Tenemos un programa que nos ayuda a obtener un número par aleatorio menor que el que nosotros le digamos.

En primer lugar, introducimos un **número aleatorio (n)**. A continuación, la función **do** (lo que queramos que haga) en nuestro caso **imprimir el número aleatorio**. Y seguido de esto, la condición que debe cumplir, que sea **par** contando con el 0.

```
public void numeroPar(int numero) {
    int n;
    do {
        n = (int) (Math.random() * numero);
        System.out.println(n);
    } while (n % 2 == 0);
    System.out.println("Y el número impar que ha detenido el bucle es: " + n);
}
```

3. SENTENCIAS ANIDADAS

Son aquellas que **incluyen instrucciones if, for, while o do-while unas dentro de otras**, teniendo en cuenta que cada estructura que se anide debe estar cerrada dentro de la otra.

Un ejemplo... Todos sabemos cuáles son los números pares que hay hasta el número 20.

¿Cómo se declara en Java?

Tenemos un programa que nos cuenta los números pares con un límite (hasta 20).

Para esto usaremos el **for** y el **if**. El for lo usaremos para indicar el valor de inicio (cero), la condición (limite 20) y el valor incremento para cada cálculo. Hasta aquí estaría dándonos los números del 0 al 20. Pero como sólo queremos los pares, pondremos la condición if para que sólo tenga en cuenta esos valores, los que sean pares.

```
public static void main (String [] args) {
  int limite = 20;
  System.out.println("Mostrar pares hasta el 20");

  for (int contador=0; contador<=limite;contador++) {
    if(contador%2 ==0) {
       System.out.println(contador + " ");
    }
  }
}
```

1.3. PREGUNTAS FRECUENTES

- **¿Qué diferencias hay entre While y Do-While en Java?**

  El bucle do while es prácticamente igual al while, pero con la diferencia de que el código del bucle se ejecutara al menos una vez ya que la comprobación se hace después de cada iteración y no antes como en el caso del while.

- **¿Podemos usar varios operadores a la vez?**

  Sí, por ejemplo, con el operador aritmético combinado (por ejemplo: x=5+10 estamos usando el operador aritmético (+) y el de asignación (=).

FUNDACIÓN ACCENTURE

accenture