CURSO
BACKEND 1

Estructuras de control

Material de lectura

OBJETIVOS DE LA GUÍA

En esta guía aprenderemos a:

- Definir y utilizar estructuras de control
- Definir y utilizar estructuras repetitivas
- Entender las sentencias de salto

egg | Argentina programa 4.0

Estructuras de control

Ya conocemos las estructuras de control de nuestro paso por PseInt pero ahora vamos a verlas en detalle en Java y como es su funcionamiento.

Instrucciones de bifurcación

Mediante estas instrucciones el desarrollo lineal de un programa se interrumpe. Las bifurcaciones o al flujo de un programa puede ser según el punto del programa en el que se ejecuta la instrucción hacia adelante o hacia atrás. De esto se encargan las estructuras de control.

Para esto también vamos a usar los operadores lógicos o condicionales, estos son los mismos que en PseInt pero se escriben de distintas formas:

Operadores Condicionales	
&&	AND
||	OR
!	Operador Lógico de Negación.

Estructuras de Control

Las estructuras de control son construcciones hechas a partir de palabras reservadas del lenguaje que permiten modificar el flujo de ejecución de un programa. De este modo, pueden crearse construcciones de alternativas mediante sentencias condicionales y bucles de repetición de bloques de instrucciones. Hay que señalar que un bloque de instrucciones se encontrará encerrado mediante llaves {.....} si existe más de una instrucción.

Estructuras condicionales

Los condicionales son estructuras de control que cambian el flujo de ejecución de un programa de acuerdo con si se cumple o no una condición. Cuando el flujo de control del programa llega al condicional, el programa evalúa la condición y determina el camino a seguir. Existen dos tipos de estructuras condicionales, las estructuras *if / else* y la estructura *switch*.

La teoría y el funcionamiento es IGUAL que en PseInt. ¿Qué aprenderemos ahora? A *"escribir"* estas estructuras en Lenguaje Java.

If/Else

La estructura *if* es la más básica de las estructuras de control de flujo. Esta estructura le indica al programa que ejecute cierta parte del código sólo si la condición evaluada es verdadera («true»). La forma más simple de esta estructura es la siguiente:

```
if(<condición>){
  <sentencias>
}
```

En donde, <*condición*> es una expresión condicional cuyo resultado luego de la evaluación es un dato booleano(lógico) verdadero o falso. El bloque de instrucciones <sentencias> se ejecuta si, y sólo si, la expresión (que debe ser lógica) se evalúa a true, es decir, se cumple la condición.

Luego, en caso de que la condición no se cumpla y se quiera ejecutar otro bloque de código, otra forma de expresar esta estructura es la siguiente:

```
if(<condición>){
  <sentencias A>
} else {
   <sentencias B>
}
```

El flujo de control del programa funciona de la misma manera, cuando se ejecuta la estructura if, se evalúa la expresión condicional, si el resultado de la condición es verdadero se ejecutan las sentencias que se encuentran contenidas dentro del bloque de código if (<sentencias A>). Contrariamente, se ejecutan las sentencias contenidas dentro del bloque else (<sentencias B>).

En muchas ocasiones, se anidan estructuras alternativas **if-else**, de forma que se pregunte por una condición si anteriormente no se ha cumplido otra y así sucesivamente.

```
if (<condicion1>) {

<sentencias A>

} else if(<condicion2>){

<sentencias B>

} else {

<sentencias C>

}
```


```
public static void main(String[] args) {
int num1 = 5;
int num2 = 7;
if (num1 < num2) {
System.out.println("La variable num1 aloja un número menor a la variable num2");
}else {
System.out.println("La variable num1 aloja un número mayor a la variable num2");
}
```

¡MANOS A LA OBRA!

Ejercicio 6

Implementar un programa que le pida dos números enteros al usuario y determine si ambos o alguno de ellos es mayor a 25.

Switch

El bloque *switch* evalúa qué valor tiene la variable, y de acuerdo con el valor que posee ejecuta las sentencias del bloque case correspondiente, es decir, del bloque case que cumpla con el valor de la variable que se está evaluando dentro del switch.

```
switch(<variable>) {

case <valor1>:

<sentencias1>

break;

case <valor2>:

<sentencias2>

break;

default:

<sentencias3>

}
```

El uso de la sentencia break que va detrás de cada case termina la sentencia switch que la envuelve, es decir que el control de flujo del programa continúa con la primera sentencia que se encuentra a continuación del cierre del bloque switch. Si el programa comprueba que se cumple el primer valor (valor1) se ejecutará el bloque de instrucciones <sentencias1>, pero si no se encuentra inmediatamente la sentencia break también se ejecutaría las instrucciones <sentencias2>, y así sucesivamente hasta encontrarse con la palabra reservada break o llegar al final de la estructura.

Las instrucciones dentro del bloque default se ejecutan cuando la variable que se está evaluando no coincide con ninguno de los valores case. Esta sentencia equivale a else de la estructura if-else.

¿NECESITAS UN **EJEMPLO**?

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

      Scanner leer = new Scanner(System.in);

      int opcion;

      System.out.println("Ingrese una opción");

      opcion = leer.nextInt();

      switch (opcion) {
            case 1:
            System.out.println("Esta línea de código se ejecuta si
opcion = 1");
            break;

            case 2:
            System.out.println("Esta línea de código se ejecuta si
opcion = 2");
             break;

            default:
            System.out.println("El valor ingresado en la variable
opcion es diferente" + "a todos los casos analizados por el
switch");
      }
}
```

¡MANOS A LA OBRA!

Ejercicio 7

Considera que estás desarrollando una web para una empresa que fabrica motores (suponemos que se trata del tipo de motor de una bomba para mover fluidos). Definir una variable tipoMotor y permitir que el usuario ingrese un valor entre 1 y 4. El programa debe mostrar lo siguiente:

- Si el tipo de motor es 1, mostrar un mensaje indicando “La bomba es una bomba de agua”.
- Si el tipo de motor es 2, mostrar un mensaje indicando “La bomba es una bomba de gasolina”.
- Si el tipo de motor es 3, mostrar un mensaje indicando “La bomba es una bomba de hormigón”.
- Si el tipo de motor es 4, mostrar un mensaje indicando “La bomba es una bomba de pasta alimenticia”.
- Si no se cumple ninguno de los valores anteriores mostrar el mensaje “No existe un valor válido para tipo de bomba”

Pueden encontrar un ejemplo de estructuras condicionales en Java en tu Aula Virtual.

Estructuras repetitivas

Durante el proceso de creación de programas, es muy común encontrarse con que una operación o conjunto de operaciones deben repetirse muchas veces. Para ello es importante conocer las estructuras de algoritmos que permiten repetir una o varias acciones, un número determinado de veces.

Las estructuras que repiten una secuencia de instrucciones un número determinado de veces se denominan *bucles*, y se denomina *iteración* al hecho de repetir la ejecución de una secuencia de acciones.

Todo bucle tiene que llevar asociada una condición, que es la que va a determinar cuándo se repite el bucle y cuando deja de repetirse.

Sentencias de Salto

En Java existen dos formas de realizar un salto incondicional en el flujo "normal" de un programa. A saber, las instrucciones, break y continue.

Break

La instrucción break sirve para abandonar una estructura de control, tanto de las condicionales (if-else y switch) como de las repetitivas (for, do-while y while). En el momento que se ejecuta la instrucción break, el control del programa sale de la estructura en la que se encuentra contenida y continua con el programa.

Continue

La sentencia *continue* corta la iteración en donde se encuentra el continue, pero en lugar de salir del bucle, continúa con la siguiente iteración. La instrucción continue transfiere el control del programa desde la instrucción continue directamente a la cabecera del bucle (for, do-while o while) donde se encuentra.

Pueden encontrar un ejemplo de sentencias de salto en Java en tu Aula Virtual.

While

La estructura *while* ejecuta un bloque de instrucciones mientras se cumple una condición. La condición se comprueba antes de empezar a ejecutar por primera vez el bucle, por lo tanto, si la condición se evalúa a «false» en la primera iteración, entonces el bloque de instrucciones no se ejecutará ninguna vez.

```
while (<condición>) {

<sentencias>

}
```

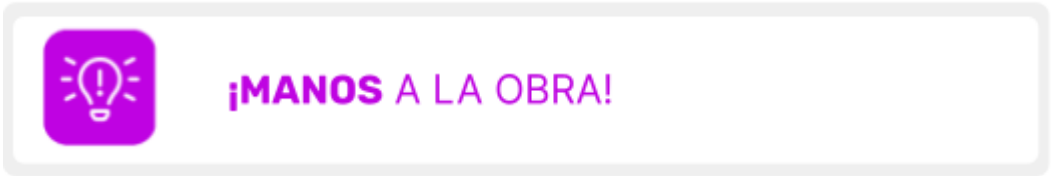

```java
public static void main(String[] args) {

    Scanner leer = new Scanner(System.in);

    String respuesta = "S";

    while (respuesta.equalsIgnoreCase("S")) {

    System.out.println("Desea continuar?");
    respuesta = leer.nextLine();
       }
  }
```



Ejercicio 8

Escriba un programa que valide si una nota está entre 0 y 10, sino está entre 0 y 10 la nota se pedirá de nuevo hasta que la nota sea correcta.

Do / While

En este tipo de bucle, el bloque de instrucciones se ejecuta siempre al menos una vez. El bloque de instrucciones se ejecutará mientras la condición se evalúe a «true». Por lo tanto, entre las instrucciones que se repiten deberá existir alguna que, en algún momento, haga que la condición se evalúe a «false», de lo contrario el bucle será infinito.

```
do {

<sentencias>

} while (<condición>);
```

¿NECESITAS UN **EJEMPLO**?

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

    Scanner leer = new Scanner(System.in);

    String respuesta;

    do {

    System.out.println("Desea continuar?");
    respuesta = leer.nextLine();

    } while (respuesta.equalsIgnoreCase("S"));
}
```

Compara este ejemplo con el dado en la estructura WHILE.
¿Qué diferencias encuentras además de la estructura?

¡MANOS A LA OBRA!

Ejercicio 9

Escriba un programa que lea 20 números. Si el número leído es igual a cero se debe salir del bucle y mostrar el mensaje "Se capturó el numero cero". El programa deberá calcular y mostrar el resultado de la suma de los números leídos, pero si el número es negativo no debe sumarse. **Nota: recordar el uso de la sentencia break.**

La diferencia entre do-while y while es que do-while evalúa su condición al final del bloque en lugar de hacerlo al inicio. Por lo tanto, el bloque de sentencia después del "do" siempre se ejecutan al menos una vez.

Revisemos lo aprendido hasta aquí

- Utilizar una estructura DO / WHILE
- Distinguir cómo se ejecutarán los comandos y cuándo se verifica la condición
- Comprender cuándo debo usar una estructura DO /WHILE
- Diferenciar una estructura WHILE de una DO / WHILE

For

La estructura *for* proporciona una forma compacta de recorrer un rango de valores cuando la cantidad de veces que se debe iterar un bloque de código es conocida. La forma general de la estructura for se puede expresar del siguiente modo:

```
for (<inicialización>; <terminación>; <incremento>) {

<sentencias>

}
```

La expresión <inicialización> inicializa el bucle y se ejecuta una sola vez al iniciar el bucle. El bucle termina cuando al evaluar la expresión <terminación> el resultado que se obtiene es false. La expresión <incremento> se invoca después de cada iteración que realiza el bucle; esta expresión incrementa o decrementa un valor hasta que se cumpla la condición de <terminación> del bucle.

Como regla general puede decirse que se utilizará el bucle for cuando se conozca de antemano el número exacto de veces que ha de repetirse un determinado bloque de instrucciones. Se utilizará el bucle do-while cuando no se conoce exactamente el número de veces que se ejecutará el bucle, pero se sabe que por lo menos se ha de ejecutar una. Se utilizará el bucle while cuando es posible que no deba ejecutarse ninguna vez.


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

      for (int i = 0; i < 10; i++) {

      System.out.println("Imprimo el valor de i: " + i);

      }

      System.out.println("====================================");

      System.out.println("For decreciendo");

      for (int i = 10; i > 10; i--) {

      System.out.println("Imprimo el valor de i: " + i);

      }
}
```

Resultado:

```
Imprimo el valor de i: 0
Imprimo el valor de i: 1
Imprimo el valor de i: 2
Imprimo el valor de i: 3
Imprimo el valor de i: 4
Imprimo el valor de i: 5
Imprimo el valor de i: 6
Imprimo el valor de i: 7
Imprimo el valor de i: 8
Imprimo el valor de i: 9
====================
For decreciendo
Imprimo el valor de i: 10
Imprimo el valor de i: 9
Imprimo el valor de i: 8
Imprimo el valor de i: 7
Imprimo el valor de i: 6
Imprimo el valor de i: 5
Imprimo el valor de i: 4
Imprimo el valor de i: 3
Imprimo el valor de i: 2
Imprimo el valor de i: 1
```

Recuerdas el atajo del sout, vamos a probar lo mismo con el bucle for. Al escribir for y tabular nos va a generar automáticamente un bucle for.

Además, pueden encontrar un ejemplo de estructuras repetitivas en tu Aula Virtual.

Ejercicio 10

Realizar un programa que lea 4 números (comprendidos entre 1 y 20) e imprima el número ingresado seguido de tantos asteriscos como indique su valor. Por ejemplo:

5 *****

3 ***

11 **********

2 **

Revisemos lo aprendido hasta aquí

- Utilizar una estructura for
- Distinguir el inicio, la terminación y el incremento.
- Comprender cómo itera el valor de i
- Implementar for anidados