**UT3: ESTRUCTURAS DE CONTROL**

**1. Estructuras de selección.**

**1.1 Estructura If.**

**1.2 Switch.**

**1.3 Operador Condicional.**

**2. Estructuras de repetición.**

**2.1 Bucle For.**

**2.2 Bucle While.**

**2.3 Bucle Do-While**

**1.- Estructuras de selección.**

Las estructuras de selección son aquellas que se utilizan cuando se quiere ejecutar una instrucción o bloque de instrucciones si se cumple una determinada condición, de ahí que también se conozcan con el nombre de condicionales.

**1.1 Estructura If.**

Permite ejecutar una instrucción (o secuencia de instrucciones) si se da una condición. La sentencia **if** es la sentencia básica de selección y existen tres variantes: simple, doble y selección múltiple, pero las 3 se basan en la misma idea.

a) **Selección Simple**. Solo tiene la parte positiva de la selección. Su sintaxis es:

|  |
| --- |
| **if (*condición*){**  ***sentencias;***  **}** |

Donde **condición** es una expresión booleana y **sentencias** representa una sentencia o bloque de sentencias, las cuales se ejecutarán si el valor de la condición es True (cierta).

Si es una **sentencia única**, se pueden quitar las llaves

**Ej:**

if (numero%2 != 0){

System.out.println(“El numero es impar “);

cont++;

}

b) **Selección Doble.** Se añade una parte else a la sentencia if la cual se ejecutará si la condición es falsa. Su sintaxis es:

|  |
| --- |
| ***if (condición){***  ***sentencias1;***  ***}***  ***else{***  ***sentencias2;***  ***}*** |

**Ej:**

if (a>b)

System.out.println (“El número mayor es “ + a);

else

System.out.println(“El número mayor es “ + b+ “ o son iguales”);

c) **Selección Múltiple**. Es habitual cuando hay más de una condición. Su sintaxis es:

|  |
| --- |
| ***if (condición1){***  ***sentencias1;***  ***}***  ***else if (condición2){***  ***sentencias2;***  ***}***  ***else if (condición3){***  ***sentencias3;***  ***else{***  ***sentencias;***  ***}*** |

**Ej:**

if (mes == 12 || mes == 1 || mes == 2)

System.out.println ( “Invierno”);

else if (mes == 3 || mes == 4 || mes == 5)

System.out.println( “Primavera”);

else if (mes == 6 || mes == 7 || mes == 8)

System.out.println (“Verano”);

else

System.out.println (“Otoño”);

**1.2 Switch.**

Es una estructura de selección múltiple más cómoda de leer y utilizar que el else-if. Selecciona un bloque de sentencias dependiendo del valor de una expresión.

Su sintaxis es:

|  |
| --- |
| ***switch (expresion) {***  ***case valor1:sentencias;***  ***break;***  ***case valor2:sentencias;***  ***break;***  ***case valor3:sentencias;***  ***break;***  ***...***  ***default: sentencias;***  ***break;***  ***}*** |

Donde ***expresión*** tiene que tomar un valor entero o un carácter. A partir de la versión 7 de Java también se permite que la expresión sea un String (cadena), ***break*** indica que ha acabado la ejecución de ese caso y seguiría ejecutando la sentencia siguiente al switch y ***default*** es opcional y se ejecutará cuando la expresión tome un valor que no esté recogido en los distintos casos especificados.

**Ej:**

switch (mes) {

case 4:

case 6:

case 9:

case 11: dias = 30;

break;

case 2: dias = 28;

break;

default: dias = 31;

break;

}

Es posible juntar distintos casos, dejándolos en blanco y especificando las instrucciones en el último de los casos de grupo.

En el ejemplo, si la variable mes toma los valores 4, 6, 9 u 11 se hace en los 4 casos lo mismo, se le asigna a la variable día el valor 30.

**1.3 Operador Condicional.**

El operador condicional es un operador ternario, es decir, consta de tres operandos y su función es asignar un valor entre dos posibles a una variable si se cumple o no una condición.

Su sintaxis es la siguiente:

|  |
| --- |
| ***tipo variable = (condicion) ? v\_cond\_true: v\_cond\_false;*** |

Realmente es un if…else simple que podemos utilizar si solo queremos asignar un valor a una variable si se cumple o no una condición.

**Ejemplo**:

**public** **class** EjemploOperadorCondicional

{

**public** **static** **void** main([String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I'm%20Feeling%20Lucky)[] args)

   {

**int** edad = 18;

*//Utilizando el operador condicional*

[String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I'm%20Feeling%20Lucky) resultado = (edad >=18) ? "Mayor de edad." : "Menor de edad.";

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I'm Feeling Lucky).out.println(resultado);

*//Utilizando if...else*

**if**(edad >= 18)

       {

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I'm Feeling Lucky).out.println("Mayor de edad.");

       }

**else**

       {

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I'm Feeling Lucky).out.println("Menor de edad.");

       }

   }

}

**2.- Estructuras de repetición.**

Estas estructuras se utilizan para repetir un bloque de sentencias un número de veces. Son también llamadas sentencias de iteración o bucles.

Los tipos de bucles que hay son: ***for***, cuando se sabe el número de veces que se va realizar y ***while o do-while***, cuando no se conoce de antemano el número de veces que se repetirá el bloque de sentencias.

**2.1 Bucle For**

La sintaxis de la sentencia es:

|  |
| --- |
| ***for(inicialización;condición;incremento/decremento){***  ***sentencias;***  ***}*** |

Dónde:

* **Inicialización** se realiza solo una vez, **antes de la primera iteración**.
* **Condición** se comprueba **cada vez, antes de entrar** al bucle y si es cierta entra, si no lo es se termina la ejecución de la sentencia y pasa a ejecutarse la siguiente sentencia al for.
* I**ncremento/decremento** se realiza siempre **después de terminar de ejecutar las sentencias del cuerpo de la iteración** y antes de volver a comprobar la condición de nuevo.

**Ej**:

int i;

for (i=0; i<5; i++)

System.out.println (i);

El funcionamiento de esta instrucción seria el siguiente:

**1º.** Se inicializa la i.

En el ejemplo i toma el valor 0.

**2º.** Se comprueba si cumple la condición, si se cumple entra en el bucle pero si no la cumple no entraría.

En el ejemplo sí la cumple pues 0<5.

**3º.** Se ejecuta el bloque de sentencias que pertenezcan al for.

En el ejemplo escribirá en pantalla un 0

**4º.** Vuelve al principio del bucle, incrementa la i.

En el ejemplo se incrementa en 1, con lo cual pasa a tomar valor 1.

**5º.** Se repite el proceso desde el paso 2, hasta que deje de cumplirse la condición.

En el ejemplo se repetirá hasta que i tome valor que 5 pues no se cumple que 5<5 Los siguientes dos ejemplos son un bucle infinito y un bucle que nunca se llega a ejecutar.

**Bucle infinito**: for (i=1;i<5;i--) pues nunca se dejará de cumplir que i<5.

**Bucle que no ejecuta nunca sus sentencias**: for (i=5; i<3;i++) pues desde el principio no se cumple la condición pues 5 no es menor que 3.

**2.2 Bucle While**

La sintaxis de la sentencia es:

|  |
| --- |
| ***while (condición){***  ***sentencias;***  ***}*** |

Donde **condición** es una expresión booleana que se evalúa al principio del bucle y antes de cada iteración de las sentencias.

Si la *condición es verdadera*, se ejecuta el bloque de sentencias, y se vuelve al principio del bucle.

Si la *condición es falsa*, no se ejecuta el bloque de sentencias, y se continúa con las siguientes sentencias del programa.

Si la condición es falsa desde un principio, entonces el bucle nunca se ejecuta.

Si la condición nunca llega a ser falsa, se tiene un bucle infinito.

**Ej:**

int i=0;

while (i<5){

System.out.println (i);

i++;

}

El resultado de ejecutar este bloque de instrucciones es el mismo que para el for, escribirá en pantalla del 0 al 4 en diferentes líneas.

Los dos siguientes ejemplos son también un bucle infinito y un bucle que no se ejecutará nunca.

**Bucle infinito:**

int i=0;

while (i<5){

System.out.println(i);

}

Sera infinito pues al no cambiar el valor de la i nunca, la condición siempre se está cumpliendo.

**Bucle que nunca se ejecutará:**

int i=6;

while (i<5){

System.out.println(i);

i++;

}

No se ejecuta nunca el bucle pues desde el principio no se cumple la condición que permita entrar en el while, pues 6 no es menor que 5.

**2.3 Bucle Do-While**

La sintaxis de la sentencia es:

|  |
| --- |
| *do{*  *sentencias;*  *}while (condición);* |

Es muy parecida a while. El bloque de sentencias se repite mientras se cumpla la condición pero en este caso, la condición se comprueba después de ejecutar el bloque de sentencias por lo que el bloque **se ejecuta siempre al menos una vez**.

Este tipo de bucle será muy útil cuando se quiere obligar a que una determinada variable solo pueda tomar unos ciertos valores, o cuando se quiere comprobar una contraseña antes de seguir ejecutando el bucle, etc…

**Ej**:

int i;

do {

System.out.println (“Introducir un numero distinto de 0”);

i = teclado.nextInt();

} while(i == 0);

En este ejemplo se pide introducir un número que no sea 0, con lo cual no se saldrá del bucle mientras se introduzca un 0. (Probarlo en clase).