CONDICIONALES

Condicional simple if .. else

Se emplea cuando se quiere ejecutar un bloque de código siempre y cuando se cumplan una o varias condiciones. Estas condiciones devuelven un valor booleano, es decir true o false.

Sintaxis:

```
.
.
if(condición que devuelve un valor booleano){
        Bloque de código 1
}

//Opcional
else{
        Bloque de código alternativo
}
.
.
```

Ejemplo:

```
1  public class EjemploCondicional1{
2      public static void main(String args[]){
3          int edadLuis=24;
4          if(edadLuis>=18){
5              System.out.println("Luis es mayor de edad y puede votar");
6              System.out.println("Luis tiene "+edadLuis+" a"+(char)164+"os");
7          }
8          else{
9              System.out.println("Luis no es mayor de edad y no puede votar");
10             System.out.println("Luis tiene "+edadLuis+" a"+(char)164+"os");
11         }
12         System.out.println("FIN DE PROGRAMA");
13     }
14 }
```

Por consola:

Luis es mayor de edad y puede votar.

Luis tiene 24 años.

NOTA 1: si el bloque de código del if tienen una sola línea de código no es necesario utilizar llave, aunque si se escribe, no da error. Lo mismo ocurre con el bloque de código alternativo

NOTA 2: si no se escribe el else opcional y no se cumple la condición del if, el programa continúa ejecutándose pasando a las siguientes líneas de código

NOTA 3: si se cambia el valor de la variable int edadLuis a 16, entonces se ejecutará el bloque de código del

else. Es decir, por consola se mostrará:

Luis no es mayor de edad y no puede votar

Luis tiene 16 años

Condicional compuesto if .. else if .. else if .. etc .. else

Admite tantas condiciones como necesite el programador.

Sintaxis:

```
.
.
if(condición 1){

    Bloque de código 1

}
else if(condición 2){

    Bloque de código 2

}
.
.
//Opcional
else{

    Bloque de código alternativo

}
.
.
```

NOTA: si el bloque de código de los if tienen una sola línea de código no es necesario utilizar llaves, aunque si se escriben, no da error. Lo mismo ocurre con el bloque de código alternativo

Ejemplo:

```java
public class EjemploCondicionalCompuesto{
    public static void main(String args[]){
        int edadPedro=25;
        int edadLuis=22;
        System.out.println("BIENVENIDO AL PROGRAMA");

        if (edadPedro>=18 && edadLuis>=18){
            System.out.println("Pedro y Luis son mayores de edad");
            System.out.println("Pedro y Luis pueden votar");
        }
        else if(edadPedro>=18 && edadLuis<18){
            System.out.println("Pedro es mayor de edad,Luis no");
            System.out.println("Pedro puede votar, Luis no");
        }
        else if(edadPedro<18 && edadLuis>=18){
            System.out.println("Luis es mayor de edad, Pedro no");
            System.out.println("Luis puede votar, Pedro no");
        }
        else{
            System.out.println("Pedro y Luis no son mayores de edad");
            System.out.println("Ni Pedro ni Luis pueden votar");
        }
        System.out.println("FIN DE PROGRAMA");
    }
}
```

Por consola:

Pedro y Luis son mayores de edad

Pedro y Luis pueden votar

Operador ternario

Se emplea para asignar valores a una variable en función del cumplimiento o no de una condición. Su sintaxis se basa en tres términos.

Sintaxis: los paréntesis son opcionales

algunaVariable=(condición que devuelve un valor booleano) ? (valor si devuelve true) : (valor si devuelve false)

NOTA: los valores que se devuelven deben ser compatibles con los que admite la variable que se inicializa

Ejemplo: en este ejemplo se muestra el funcionamiento del operador ternario y se introduce la variable referenciada String, que se utiliza para almacenar cadenas de texto. Se estudiará en profundidad más adelante.

```java
public class OperadorTernario{
    public static void main(String args[]){
        int edadPedro=25;

        //Declaración e inicialización de dos Strings como si fueran
        //variables primitivas. Las cadenas de texto, entre comillas
        String opcion1="Pedro es mayor de edad y puede votar";
        String opcion2="Pedro es menor de edad y no puede votar";

        //El valor que devuelve el operador ternario se recoge en una String
        String respuesta=edadPedro>=18?opcion1:opcion2;
        System.out.println(respuesta);
    }
}
```

Por consola:

Pedro es mayor de edad y puede votar

Condicional switch

Es parecido o similar a la estructura if()...else if()...else if()...else, pero con ciertas limitaciones:

- **Limitación 1:** sólo admite una condición. No son validos los operadores && y ||.
- **Limitación 2:** en esa condición se presupone que el operador de relación es la igualdad (==).
- **Limitación 3:** la variable asociada a la condición sólo puede ser de tipo byte, short, int o char, ninguna otra es válida.

Ejemplo 1:

```
1  public class EjemploSwitch1{
2      public static void main(String args[]){
3          char operador='+';
4          switch(operador){
5              case '-':
6                  System.out.println("El operador es -");
7              case '+':
8                  System.out.println("El operador es +");
9              case '*':
10                 System.out.println("El operador es *");
11             case '/':
12                 System.out.println("El operador es /");
13
14             default:
15                 System.out.println("Operador desconocido");
16         }
17         System.out.println("FIN DE PROGRAMA");
18     }
19 }
```

Por consola:

El operador es +

El operador es *

El operador es /

Operador desconocido

Explicación:

Se ejecutan todas las instrucciones a partir del segundo case (es el que coincide con la variable operador). Si no hay ningún case coincidente, se ejecuta el código asociado a default.

NOTA 1: no se puede repetir ningún case con el mismo valor. Se produce error de compilación

NOTA 2: si se quiere que el programa, cuando encuentre el primer case coincidente con el valor de la variable evaluada, salga del switch, debe agregarse a cada case la instrucción break del siguiente modo:

Ejemplo 2:

```
1  public class EjemploSwitch2{
2      public static void main(String args[]){
3          char operador='+';
4          switch(operador){
5              case '-':
6                  System.out.println("El operador es -");
7                  break;
8              case '+':
9                  System.out.println("El operador es +");
10                 break;
11             case '*':
12                 System.out.println("El operador es *");
13                 break;
14             case '/':
15                 System.out.println("El operador es /");
16                 break;
17             default:
18                 System.out.println("Operador desconocido");
19         }
20         System.out.println("FIN DE PROGRAMA");
21     }
22 }
```

Por consola:

El operador es +

BUCLES

Bucle for

Estructura utilizada para ejecutar una o varias líneas de código un determinado número de veces. Este número de veces depende del cumplimiento de una condición fijada por el programador.

Sintaxis:

```
.
.
for(variable de control;condición;paso de la variable de control){

    Bloque de código que va a repetirse
    hasta que se incumpla la condición
    del segundo argumento

}
.
.
```

Ejemplo 1: se quiere desarrollar un programa que muestre por consola los números del 0 al 4. En este momento sólo se podría hacer de la siguiente manera:

```
1  public class Numeros{
2      public static void main(String args[]){
3          int x1=0;
4          int x2=1;
5          int x3=2;
6          int x4=3;
7          int x5=4;
8          System.out.println(x1);
9          System.out.println(x2);
10         System.out.println(x3);
11         System.out.println(x4);
12         System.out.println(x5);
13         System.out.println("FIN DE PROGRAMA");
14     }
15 }
```

Este ejemplo se hace mucho mejor y con menos trabajo para el programador mediante un bucle for:

```
1 public class BucleFor{
2     public static void main(String args[]){
3         for(int i=0;i<5;i++)
4             System.out.println(i);
5         System.out.println("FIN DE PROGRAMA");
6     }
7 }
```

Si se modifica la línea System.out.println(i) por System.out.print(i) no se insertará una línea vacía después de cada número, aparecerán todos en la misma línea.

```
1 public class BucleForBis{
2     public static void main(String args[]){
3         for(int i=0;i<5;i++)
4             System.out.print(i);
5         System.out.println("FIN DE PROGRAMA");
6     }
7 }
```

Por consola:

01234FIN DE PROGRAMA

Bucle while

Estructura utilizada para ejecutar una o varias líneas de código un indeterminado número de veces, mínimo 0 veces. Este número de veces depende del cumplimiento de una condición fijada por el programador.

Sintaxis:

```
while (condicion_permanencia){
  instrucciones;
}
```

Ejemplo 1: Se quiere desarrollar un programa que muestre por consola los números del 0 al 4.

```
public class BucleWhile{
```

```
  public static void main(String args[]){
     int i=0;
     while (i<=4)
     {
        System.out.print(i);
        i++;
     }
     System.out.println("FIN DEL PROGRAMA");
    }
}
```

Se podrá realizar un bucle infinito mediante la sentencia `while` del siguiente modo:

```
while (true){
    // ponga aquí su código
}
```

Ejemplo 2: Introduce una serie de números hasta el 0, ve mostrándolos e indica cúantos se han introducido.

```
import java.util.*;
public class BucleWhile_2{
  public static void main(String args[]){
     Scanner teclado = new Scanner(System.in);
     int x,i=0;

     System.out.print("Dime un número :");
     x=teclado.nextInt();
     while (x!=0)
     {
        System.out.print(x);
        i++;

        System.out.print("Dime un número :");
        x=teclado.nextInt();
     }
     System.out.println("FIN DEL PROGRAMA");
    }
}
```

Bucle do-while

Estructura utilizada para ejecutar una o varias líneas de código un indeterminado número de veces, mínimo 1 vez. Este número de veces depende del cumplimiento de una condición fijada por el programador.

Sintaxis:

```
do{
  instrucciones;
}while(condicion_permanencia);
```

Ejemplo 1: Se quiere desarrollar un programa que muestre por consola los números del 0 al 4.

```
public class BucleDoWhile{
  public static void main(String args[]){
     int i=0;
     do{
        System.out.print(i);
        i++;
     } while (i<=4);
     System.out.println("Se han introducido: "+i+" números distintos de 0");
    }
}
```

Sentencias break, return y continue

Break

Aplicable dentro de bucles y estructuras condicionales de tipo switch. Su ejecución provoca que se salga del bucle o etiqueta que se esté ejecutando sin finalizar el resto de las sentencias asociadas al mismo.

Return

Se utiliza para truncar la ejecución de un método o para almacenar el dato que devuelven. Puede emplearse para truncar bucles cuando no haya más código después del bucle

Ejemplo 1:

```
1  public class EjemploBreak{
2      public static void main(String args[]){
3          for(int i=0;i<=10;i++){
4              if(i==3)
5
6                  //Se sale del bucle en el momento en que i llega a 3
7                  break;
8              System.out.print(i+" ");
9          }
10         System.out.println("\nFuera del bucle");
11     }
12 }
```

Por consola:

0 1 2

Fuera del bucle

NOTA: se sale del bucle en el momento en que i llega a 3.

Si se sustituye break por return, se trunca la ejecución del método main

Por consola:

0 1 2

Continue:

Aplicable sólo a bucles. Finaliza la iteración "i" que en ese momento se está ejecutando (significa que no ejecuta el resto de sentencias hasta el final del bucle) y vuelve a analizar la condición del bucle, pero con la siguiente iteración "i+1".

```
1  public class EjemploContinue{
2      public static void main(String args[]){
3          for(int i=0;i<=10;i++){
4              if(i==3)
5                  continue;
6              System.out.print(i+" ");
7          }
8          System.out.println("\nFuera del bucle");
9      }
10 }
```

Por consola:

0 1 2 4 5 6 7 8 9 10

Fuera del bucle

Ejemplo 3: Introduce una serie de números hasta el 0, ve mostrándolos e indica cúantos se han introducido, hacedlo con do-while.

```
import java.io.*;
public class BucleWhile_2{
  public static void main(String args[]){
     Scanner teclado = new Scanner(System.in);
     int x,i=0;

     do{
       System.out.print("Dime un número :");
       x=teclado.nextInt();
        if (x==0) break;
       System.out.print(x);
       i++;
     } while (x!=0);

    System.out.println("Se han introducido: "+i+" números distintos de 0");
    }
}
```

NOTA: Si el primer número es 0, lo muestra y lo cuenta, por lo cual, se provoca la ruptura del bucle si la variable es 0.