Control de flujo

Java como otros muchos lenguajes de programación admite tanto sentencias condicionales como bucles para determinar el control de flujo.

Ámbito de bloque

Es muy importante a la hora de trabajar con estructuras de control de flujo tener claro el concepto de "**ámbito de bloque**". Un bloque es un grupo de sentencias encerradas entre llaves. Los bloques definen el ámbito de las variables que se encuentren en nuestro código.

Existe la posibilidad de crear bloques dentro de otros bloques, lo que se denomina bloques anidados. **Ejemplo:**

```
public static void main(String[] args) {
    // TODO Auto-generated method stub

    int n;

    {   //  En este punto comienza al área de actuación de la variable k

        int k;

    }  //  Hasta este punto llega la definición de la variable "k"

}
```

No se pueden declarar dos variables con el mismo nombre en dos bloques anidados. Esto ocasionaría un error en tiempo de compilación.

Operadores relacionales y booleanos

Las estructuras de control de flujo (condicionales y bucles) hacen uso de operadores relaciones (comparación) y booleanos (lógicos) para construir las condiciones a cumplir. Por esto, es fundamental conocer bien estos operadores. En las siguientes tablas pueden verse ambos grupos de operadores:

OPERADORES RELACIONALES (COMPARACIÓN)	
Operador	**Significado**
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
==	Igual que
!=	Distinto que

OPERADORES BOOLEANOS	
Operador	**Significado**
&&	Y Lógico
||	O lógico
!	Negación
&	Y en cortocircuito
|	O en cortocircuito

Sentencias condicionales

La sentencia condicional en Java tiene la siguiente estructura:

If (condición) sentencia

La condición debe ir entre paréntesis.

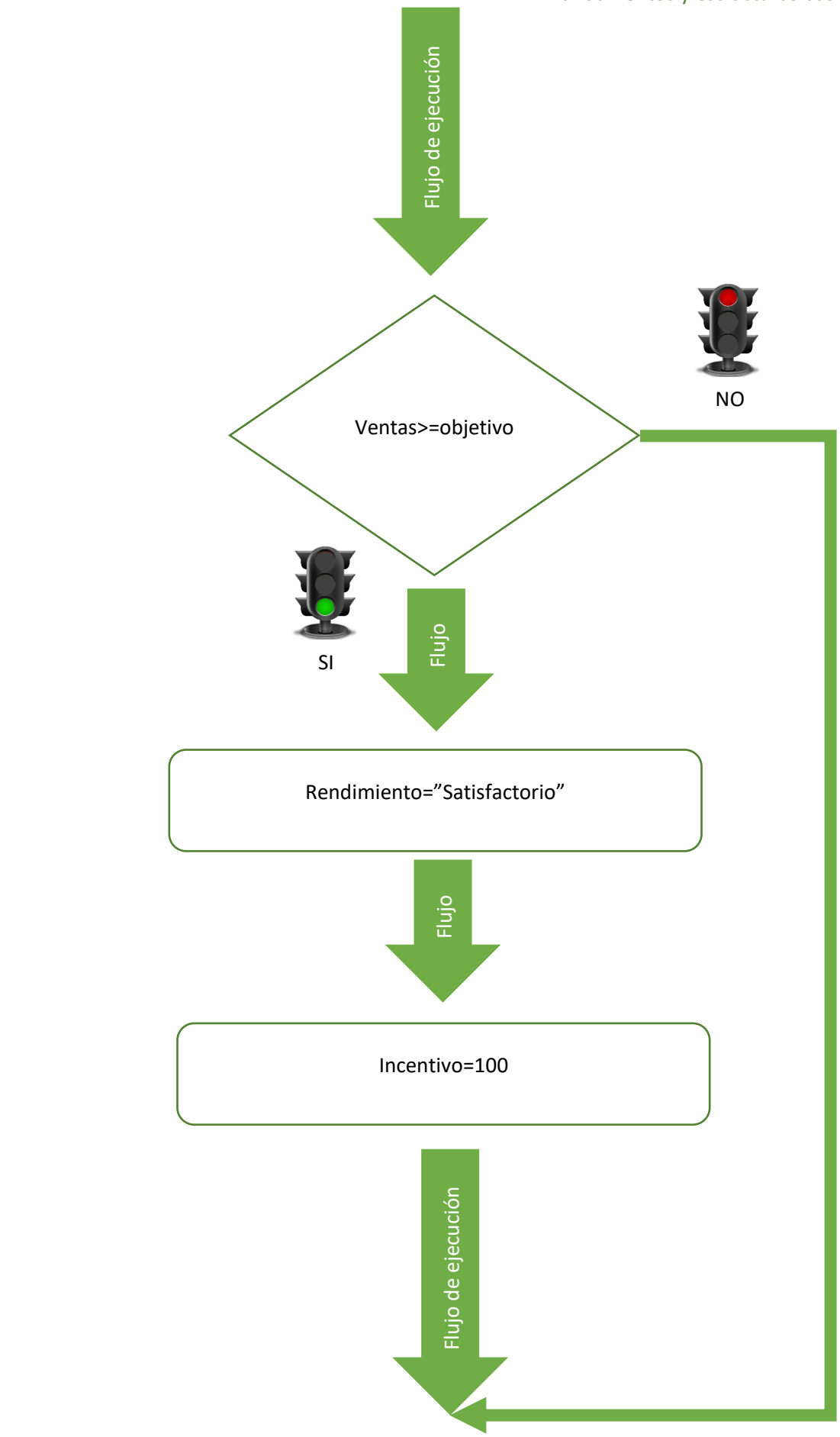
Suele ser necesario en muchas ocasiones ejecutar varias sentencias cuando la condición toma el valor trae. En estos casos se hace uso de una sentencia de bloque, que tendrá la siguiente estructura:

Ejemplo:

```
public static void main(String[] args) {
    // TODO Auto-generated method stub

    double ventas, objetivo;

    ventas=1800.50;

    objetivo=1950.25;

    int Incentivo;

    String Rendimiento;

    if ( ventas >= objetivo)

    { //  Solo ejecutará este código si la condición es verdadera
        Rendimiento = "Satisfactorio";

        Incentivo = 100;
    }

}
```

En el ejemplo anterior todas las sentencias encerradas entre llaves se ejecutarán cuando la condición se evalúe como verdadera, esto es cuando sea **true**.

El siguiente esquema muestra el diagrama de flujo del condicional IF del ejemplo anterior:


Flujo de ejecución
Ventas>=objetivo
NO
SI
Flujo
Rendimiento=”Satisfactorio”
Flujo
Incentivo=100
Flujo de ejecución

Otro ejemplo, en este caso con else:

```
public static void main(String[] args) {
    // TODO Auto-generated method stub

    double ventas, objetivo;

    ventas=1800.50;

    objetivo=1950.25;

    int Incentivo;

    String Rendimiento;

    if ( ventas >= objetivo)

    { //  Solo ejecutará este código si la condición es verdadera
        Rendimiento = "Satisfactorio";

        Incentivo = (int) (100+0.01*(ventas-objetivo));
    } else{  //  Si la condición no es verdadera, ejecutara este código

        Rendimiento="No satisfactorio";

        Incentivo=0;
    }

}
```

La parte **else** es opcional. Cada **else** está asociado con el **if** más próximo. Es frecuente encontrar sentencias anidadas, de la forma **if-elseif**.

```
    if ( ventas >=2* objetivo)

    {
        Rendimiento = "Excelente";

        Incentivo = 1000;

    } else if(ventas>=1.5 * objetivo){

        Rendimiento="Muy bueno";

        Incentivo=500;
    }else if(ventas>=objetivo){

        Rendimiento="Satisfactorio";

        Incentivo=100;
    }else{
        System.out.println("Despedido");
    }
```