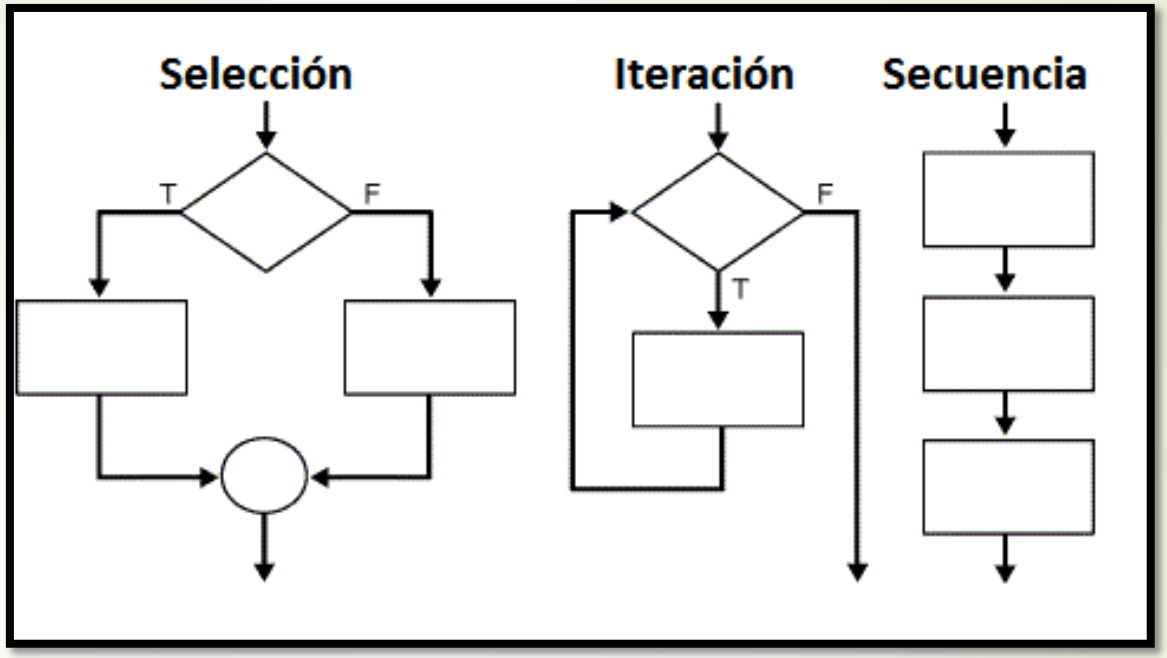
Desarrollo de Aplicaciones Web y Multiplataforma: Programación

DOCENTE: Daniel López Lozano

escuelaartegranada



Tema 2.
Estructuras de Control

Índice de contenidos

- Los operadores relacionales sirven para realizar comparaciones de igualdad, desigualdad y relación de mayor o menor.

- El resultado de estos operadores es siempre un valor booleano (true o false)

Operador	Utilización	El resultado es true
>	op1 > op2	si op1 es mayor que op2
>=	op1 >= op2	si op1 es mayor o igual que op2
<	op1 < op2	si op1 es menor que op2
<=	op1 <= op2	si op1 es menor o igual que op2
==	op1 == op2	si op1 y op2 son iguales
!=	op1 != op2	si op1 y op2 son diferentes

¿Cual es el valor de expresión por pantalla al ejecutar el código?

```
int A,B;
A=5;
B=3;
System.out.println(A>B);
System.out.println(A<B);
System.out.println(A>=B);
System.out.println(A<=B);
System.out.println(A==B);
System.out.println(A!=B);
```

Operadores lógicos o booleanos

- Los **operadores lógicos** se utilizan para construir expresiones **lógicas o booleanas**.
- Combinando valores lógicos o resultados de los operadores relacionales.

Operador	Nombre	Utilización	Resultado
`&&`	`AND`	`op1 && op2`	true si op1 y op2 son true. Si op1 es false ya no se evalúa op2
`||`	`OR`	`op1 || op2`	true si op1 u op2 son true. Si op1 es true ya no se evalúa op2
`!`	`negación`	`! op`	true si op es false y false si op es true

Tablas de verdad

OPERADOR AND

OP1	OP2	Salida
V	V	V
V	F	F
F	V	F
F	F	F

OPERADOR OR

OP1	OP2	Salida
V	V	V
V	F	V
F	V	V
F	F	F

OPERADOR NOT

OP1	Salida
V	F
F	V

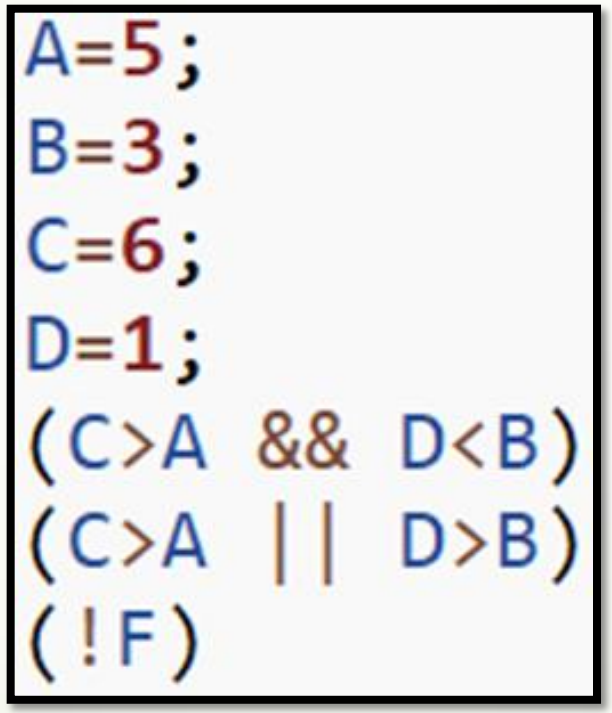
Operadores lógicos o booleanos

AND: sirve para UNIR

OR: sirve para ELEGIR

NOT: sirve para EXCLUIR

¿Cual es el valor de cada variable al ejecutar el código?



Prioridad de los operadores booleanos

Mas a menos prioridad	Operador	Nombre
NOT	!	Negación lógica
AND	&&	Conjunción
OR	||	Disyunción

Si A , B , C y D son variables del tipo boolean con valores **true , false , false y true** respectivamente, cuál es la evaluación de las siguientes expresiones:

- **!A || B**
- **A && !B**
- **! (A && B)**
- **! (A || B)**
- **A && B || ! A && D**
- **! C && (B || D)**
- **(A || D) && (C || D)**
- **!((A || B) && (C || D))**

Si a , b , c y d son variables del tipo **int** con valores 10, 12, 13, 10 respectivamente, cuál es la evaluación de las siguientes expresiones:

- **a < b && a < c || a ==c && a >= d**
- **(a != b || a > c) && (a != c || a >= b)**
- **(a >= b || a < d) && a >= d && c > d**
- **!(a == c) && c > b**

Si NUM tiene el valor 58, VALOR tiene el valor 8 y CAR tiene el valor ‘Z’, cuál es la evaluación de las siguientes expresiones:

- **(NUM>=0 && NUM<=31)**
- **(VALOR<0 || VALOR>10)**
- **!(VALOR>=0 && VALOR<=10)**
- **(NUM=<0 && NUM>=31)**
- **(NUM<0 || NUM>31)**
- **(LETRA==‘Z’ || LETRA==‘z’)**
- **(LETRA==‘Z’ && LETRA==‘z’)**
- **(LETRA>=‘A’ && LETRA<=‘Z’)**

- Hasta el momento, todos los programas y aplicaciones que hemos realizado y abordado, **no alteraban el curso de ejecución** de las sentencias.

- Es decir, las sentencias se ejecutaban ordenadamente **desde la primera a la ultima** dentro de nuestro código fuente.

- Vamos a abordar las principales estructuras de control que nos van a permitir alterar el orden de ejecución:
  - ✓ Selección
  - ✓ Repetición

Estructuras de selección

- Las estructuras de selección nos permiten decidir si queremos ejecutar o no un bloque de código entre una o más opciones, mediante la evaluación de una condición.
  - ✓ **Sentencias if-else**: Nos permiten decidir si queremos ejecutar o no un fragmento de código dependiendo de si se cumple o no una condición
  - ✓ **Sentencias switch**: Permite ejecutar un bloque diferente de código para cada valor posible que pueda tomar una expresión

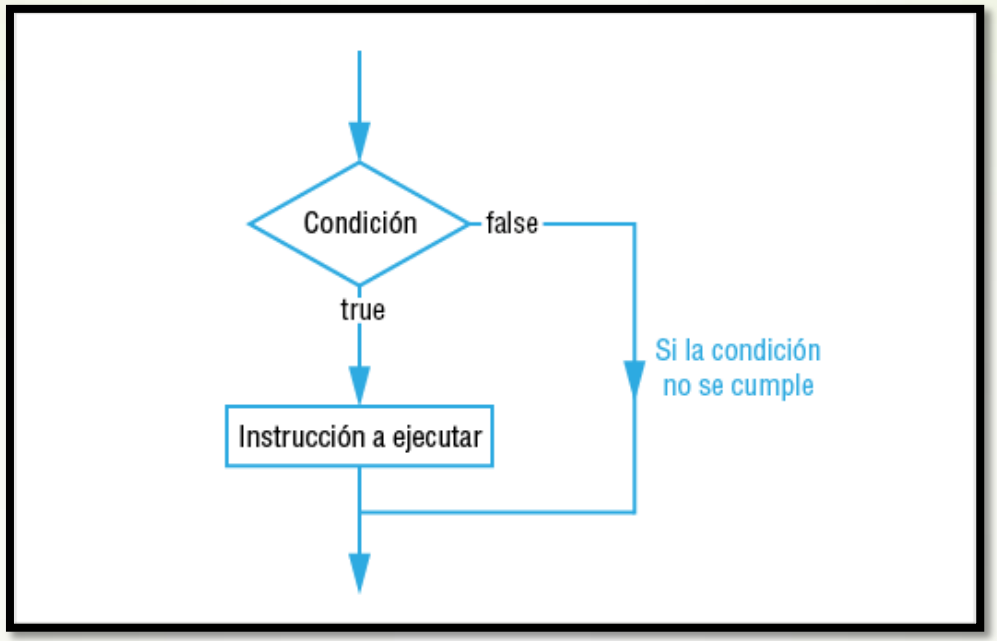
- La sentencia if permite **decidir si ejecutar o no** un fragmento de código **en función** del resultado de una **condición**.

- Para evaluar estas condiciones se utilizan los operadores de comparación y los lógicos: == , != , < , > , <= , >= , && , || y !

- En ocasiones es necesario tomar una decisión **dependiendo de varios criterios**.

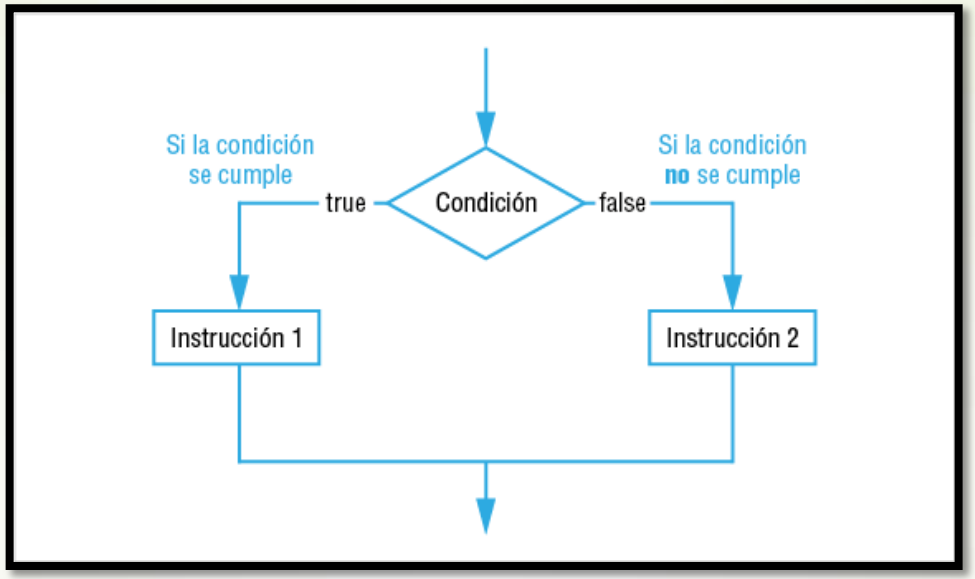
- A continuación se muestra una serie de sentencias **if - else**

Esquemas if-else



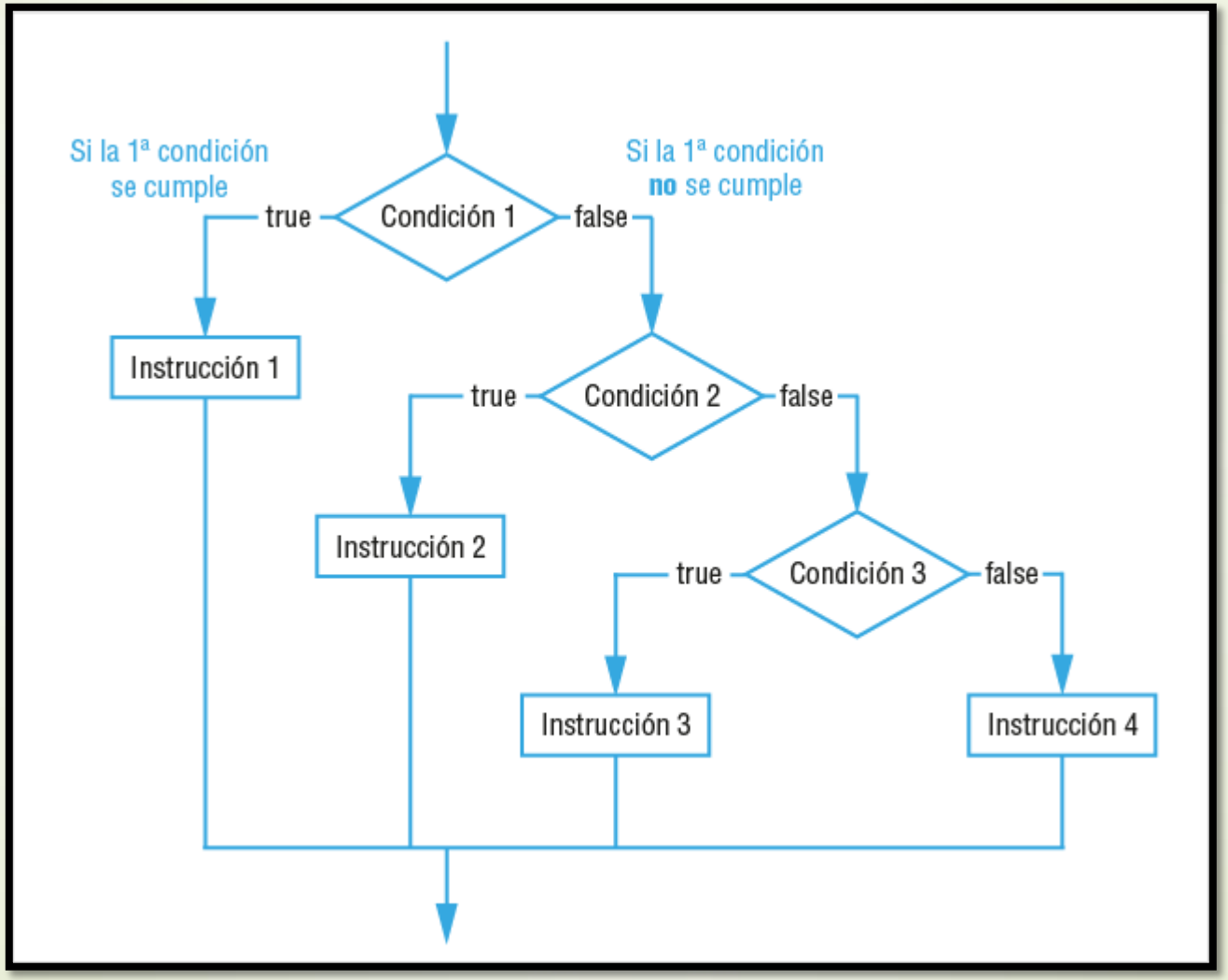

```
int edad;
System.out.println("introduce tu edad: ");
edad=sc.nextInt();
if(edad > 30)
{
    System.out.println("Ya eres una persona adulta");
}
```

Esquemas if-else




```
if(edad > 30)
{
    System.out.println("Ya eres una persona adulta");
}else{
    System.out.println("Todavia te queda para ser adulto");
}
```

- Los bloques if-else se pueden anidar cuantas veces se desee para comprobar más condiciones.



```
if(nota>=5 && nota<=10){
    System.out.println("Enhorabueba, estas aprobado");
}else{
    System.out.println("Lo siento, estas suspenso");
}
```

- Con ello podríamos hacer comprobaciones más completas.

```
if(nota>=5 && nota<=10){
    if(nota>=5 && nota<6.5){
       System.out.println("Tienes un aprobado simple");
    }else{
        if(nota>=6.5 && nota<8.5){
            System.out.println("Tienes un notable");
        }else
            ...
    }
}else{
    System.out.println("Lo siento, estas suspenso");
}
```

- En algunas ocasiones podemos escribir de forma equivalente sin usar anidaciones mediante la forma if-else if-else.

```
if(nota>=0 && nota<5){
    System.out.println("Suspenso");
}else if(nota>=5 && nota<6.5)
    System.out.println("Aprobado");
}else if(nota>=6.5 && nota<8.5){
    System.out.println("Notable");
}else if(nota>=8.5 && nota<=10){
    System.out.println("Sobresaliente");
}else{
    System.out.println("Nota invalida");
}
```

- Aunque situaciones donde es necesario anidar condiciones.

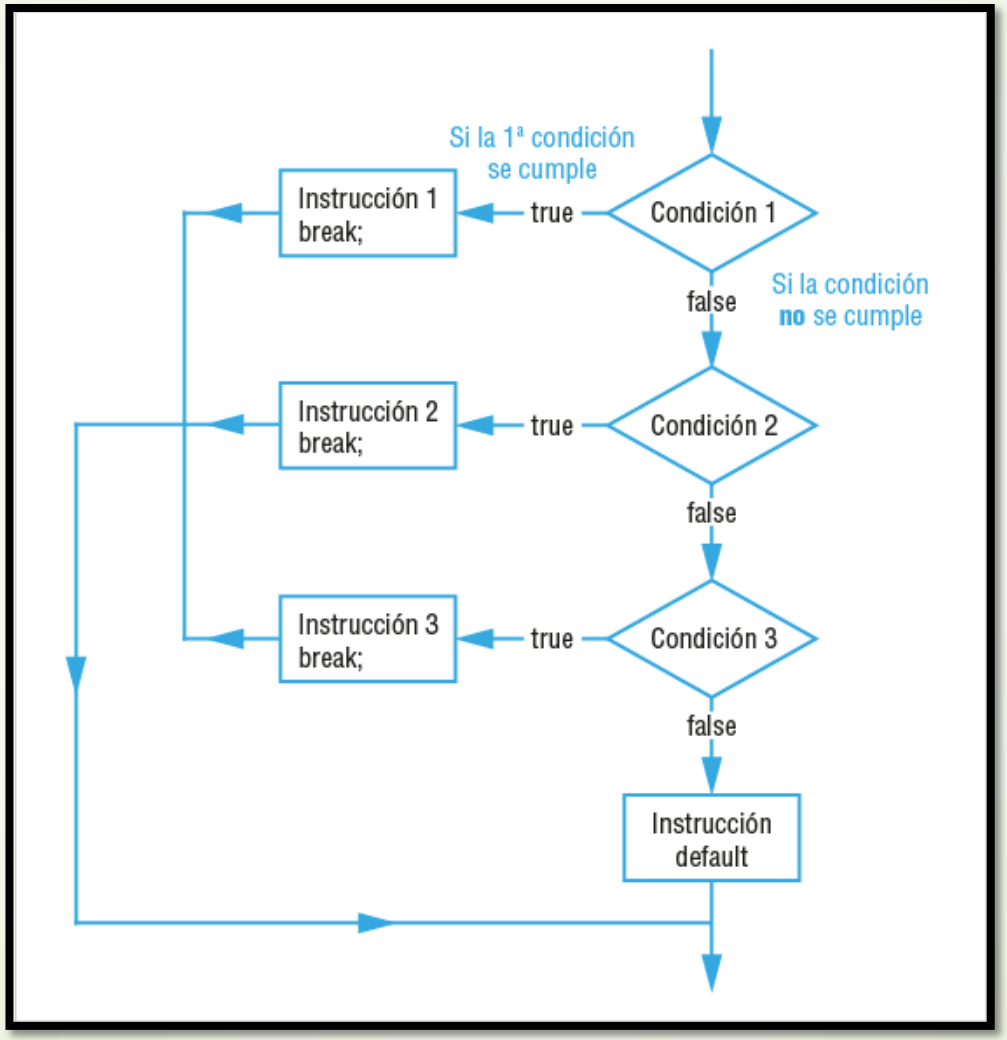
```
System.out.println("1. Sacar dinero");
System.out.println("2. Meter dinero");
System.out.println("3. Hacer una transferencia");
opcion1=teclado.nextInt();
if(opcion1==1){
    System.out.println("1. A debito");
    System.out.println("2. A credito");
    opcion2=teclado.nextInt();
    if(opcion2==1){
        ...
    }else if(opcion1==2){
        ...
    }else if(opcion1==3){
        ...
    }
}else if(opcion1==2){
    System.out.println("1. En efectivo");
    System.out.println("2. Cheque bancario");
    opcion2=teclado.nextInt();
    if(opcion2==1){
        ...
    }else if(opcion2==2){
        ...
```

- La **sentencia switch** nos permite elegir entre varios bloques de código cual deseamos ejecutar, dependiendo del valor de una expresión multivalor

- La principal diferencia es que la estructura if comprueba si **una condición de cualquier tipo** es true o false.

- Mientras que la estructura switch solo permite definir **condiciones de igualdad** con varios valores concretos.

Esquema switch



Esquema switch

```
int dia;
System.out.println("Introduce el dia de la semana");
dia=sc.nextInt();
switch(dia)
{
  case 1:
     System.out.println("Lunes");
  break;
  case 2:
     System.out.println("Martes");
  break;
  ...
  default:
     System.out.println("No es un dia de la semana");
}
```

Esquema switch

```
char c;
c = sc.next().charAt(0);
switch(c) {
    case 'a':
        System.out.println("Es la vocal a");
    break;
    case 'e':
        System.out.println("Es la vocal e");
    break;
    case 'i':
        System.out.println("Es la vocal i");
    break;
    case 'o':
        System.out.println("Es la vocal o");
    break;
    case 'u':
        System.out.println("Es la vocal u");
    break;
    default:
        System.out.println("No es una vocal");
    break;
}
```

Comparación de String

```
String deporte;
System.out.println("Dime que deporte te gusta");
deporte=teclado.nextLine();
if(deporte=="Futbol"){
    ...
}else if(deporte=="Baloncesto"){
    ...
}else if...

}
```

Comparación de String

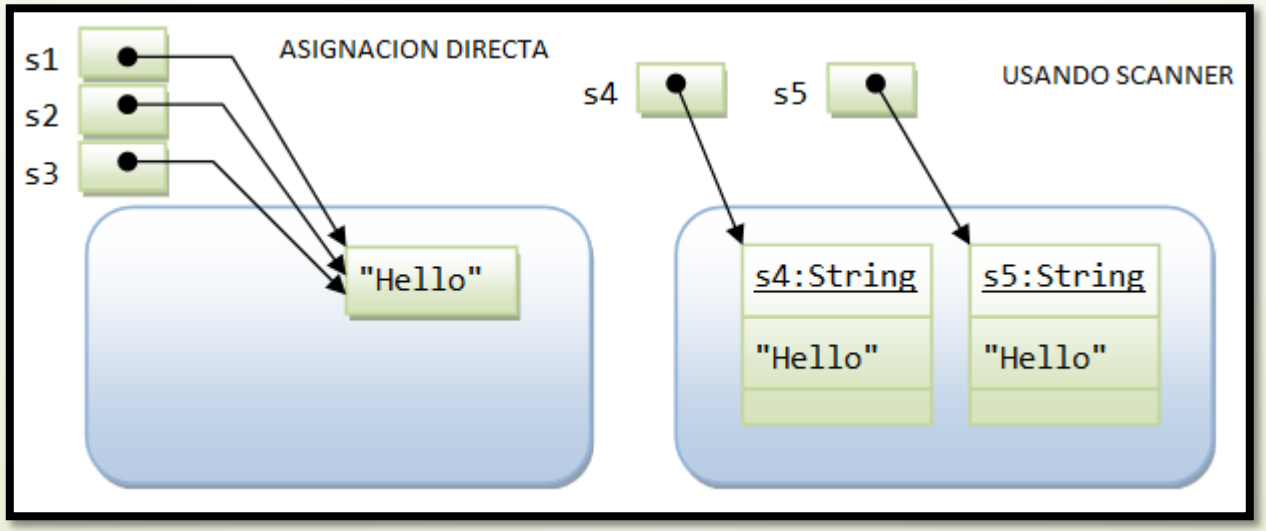
```
//Lo anterior no funciona hay que usar equals
if(deporte.equals("Futbol")){
    ...
}else if(deporte.equals("Baloncesto")){
    ...
}else if...

}

//Sin diferencia entre mayusculas y minusculas
if(deporte.equalsIgnoreCase("Futbol")){
    ...
}else if(deporte.equalsIgnoreCase("Baloncesto")){


//swicth case si sabe comparar String
```

- Como se puede observar en el caso anterior en la parte del **if** se utilizar una función llamada **equals** y en el **switch-case no**.
- Es necesario usar **equals** porque el tipo **String** es una clase y para comparar si dos objetos son iguales no se puede usar el operador == **ó** !=
- El **switch** si es capaz de comparar String.



Comparación if y switch

```java
String materia;
materia=sc.next();
if(materia.equals("matematicas"))
{
   System.out.println("Va de numeros");
}else if(materia.equals("lengua")){
   System.out.println("Va de letras");
}else if(materia.equals("ingles")){
   System.out.println("Va de hablar");
}else if(materia.equals("ciencias")){
   System.out.println("Va de investigar");
else{
   System.out.println("No sé de que va");
}
```

```java
String materia;
materia=sc.next();
switch(materia)
{
   case "matematicas":
     System.out.println("Va de numeros");
     break;
   case "lengua":
     System.out.println("Va de letras");
     break;
   case "ingles":
     System.out.println("Va de hablar");
     break;
   case "ciencias":
     System.out.println("Va de investigar");
     break;
   default:
     System.out.println("No sé de que va");
}
```

Bibliografía

- García de Jalón, j.: “Aprende Java como si estuvieras en primero”. Editorial TECNUN. 2000
- Holzner, S.: “La biblia de JAVA 2”. Editorial Anaya Multimedia 2000.
- Moreno Pérez, J.C.: “C.F.G.S Entornos de desarrollo” Editorial RA-MA. 2012
- Wikipedia, la enciclopedia libre. http://es.wikipedia.org/ Última visita: Octubre 2018.
- López, J.C.: “Curso de JAVA http://www.cursodejava.com.mx Última visita: Octubre 2018.
- Documentación oficial Java JSE 8 http://docs.oracle.com/javase/8/ Última visita: Octubre 2015.
- Programación en castellano: Java. http://www.programacion.net/java Última visita: Octubre 2015.