Palabras reservadas en Java

No pueden usarse como identificadores

`abstract`	`continue`	`for`	`new`	`switch`
`assert`	`default`	`goto`	`package`	`synchronized`
`boolean`	`do`	`if`	`private`	`this`
`break`	`double`	`implements`	`protected`	`throw`
`byte`	`else`	`import`	`public`	`throws`
`case`	`enum`	`instanceof`	`return`	`transient`
`catch`	`extends`	`int`	`short`	`try`
`char`	`final`	`interface`	`static`	`void`
`class`	`finally`	`long`	`strictfp`	`volatile`
`const`	`float`	`native`	`super`	`while`

Tipos primitivos de Java y su rango

Tipo	Uso	Tamaño	Rango
`byte`	entero corto	8 bits	de –128 a 127
`short`	entero	16 bits	de –32 768 a 32 767
`int`	entero	32 bits	de –2 147 483 648 a 2 147 483 647
`long`	entero largo	64 bits	± 9 223 372 036 854 775 808
`float`	real precisión sencilla	32 bits	de -10^{32} a 10^{32}
`double`	real precisión doble	64 bits	de -10^{300} a 10^{300}
`boolean`	lógico	1 bit	*true* o *false*
`char`	texto	16 bits	cualquier carácter

- **Widening Casting** (automático) – convierte de un tipo "pequeño" a un tipo más "grande"
`byte` -> `short` -> `char` -> `int` -> `long` -> `float` -> `double`

- **Narrowing Casting** (manual) - convierte de un tipo "grande" a un tipo más "pequeño"
`double` -> `float` -> `long` -> `int` -> `char` -> `short` -> `byte`

```
double b = 3;
```

```
int a = (int) 2.6;
```

Operadores

Aritméticos

Símbolo	Descripción
+	Suma
+	Más unario: positivo
−	Resta
−	Menos unario: negativo
*	Multiplicación
/	División
%	Resto módulo
++	Incremento: +1
−−	Decremento: –1

Relacionales

Símbolo	Descripción
==	Igual que
! =	Distinto que
<	Menor que
<=	Menor o igual que
>	Mayor que
>=	Mayor o igual que

Símbolo	Descripción
? :	Operador ternario

Lógicos

Símbolo	Descripción
&&	Operador and: Y
||	Operador or: O
!	Operador not: Negación

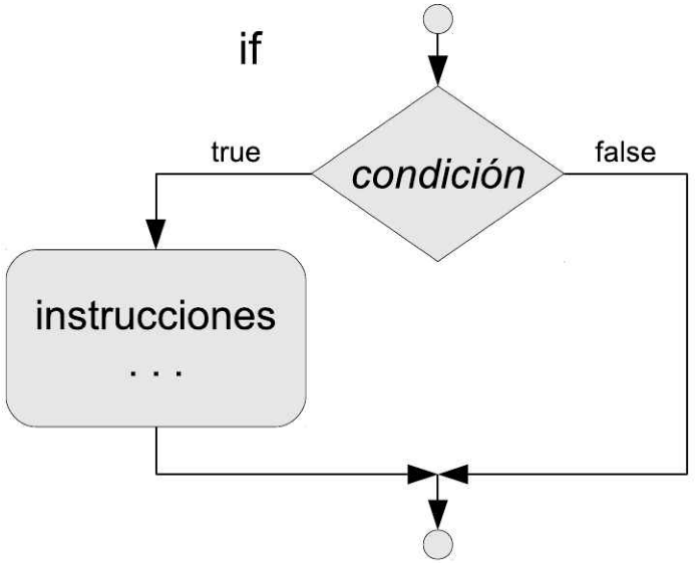
Opera y asigna

Símbolo	Descripción
+ =	Suma y asigna
− =	Resta y asigna
* =	Multiplica y asigna
/ =	Divide y asigna
% =	Módulo y asigna

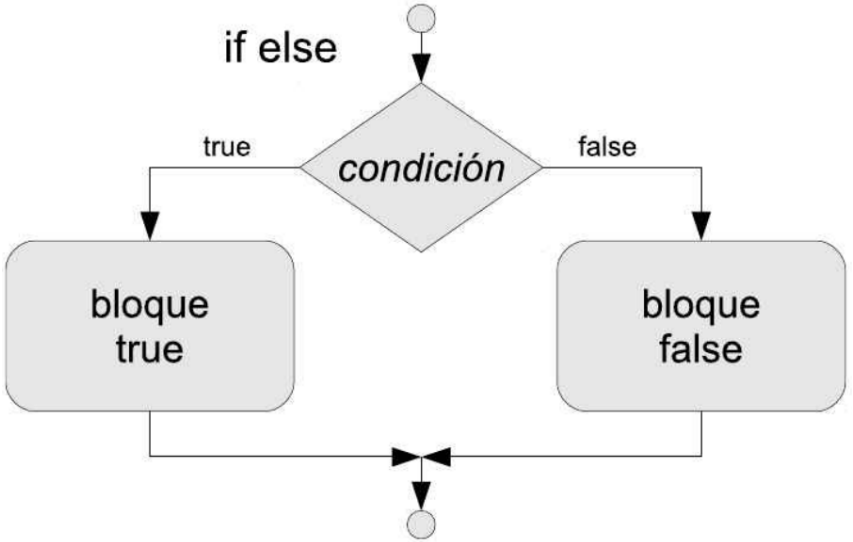
Precedencia de operadores

Descripción	Operador
Postfijos	expr++ expr−−
Unarios prefijos	++expr −−expr +expr −expr !expr
Aritméticos	* / %
Aritméticos	+ −
Relacionales	< <= > >=
Comparación	== !=
AND lógico	&&
OR lógico	||
Ternario	? :
Asignación	= += −= *= /= %= &= ^=

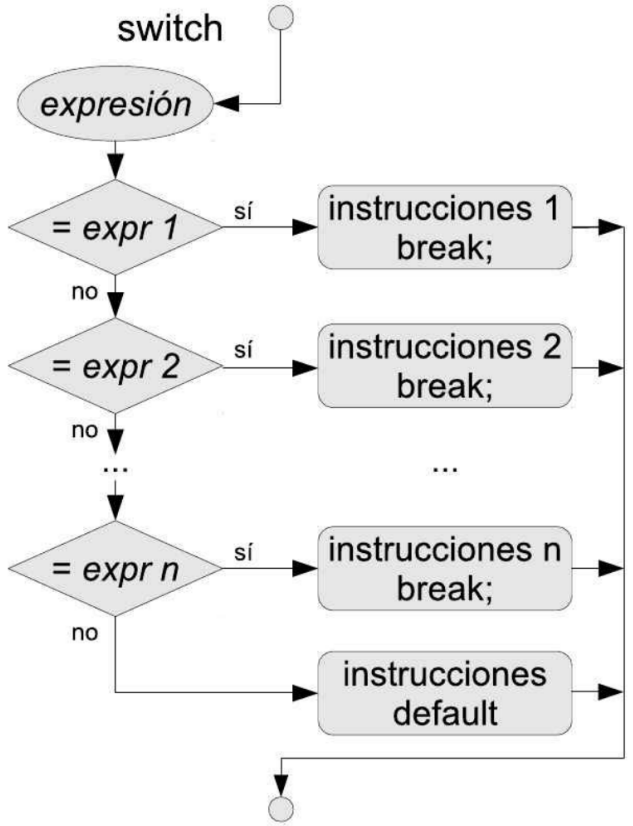
Estructuras condicionales




```
int x = 10;

if (x > 5) {
    System.out.println( "x es mayor que 5" );
}
```

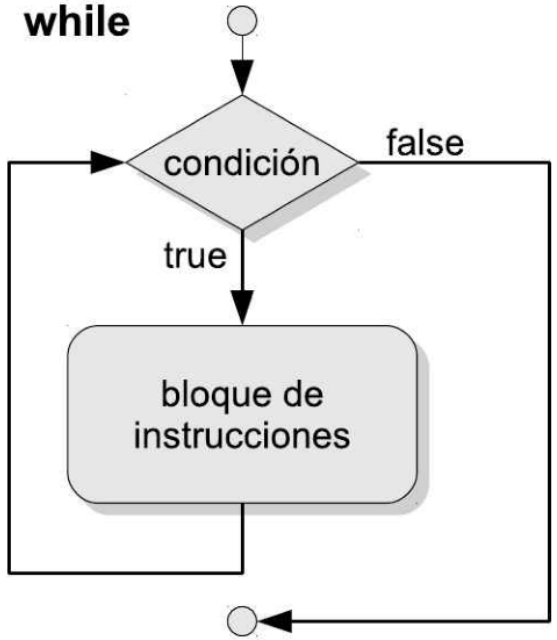



```
int x = 10;

if (x > 5) {
    System.out.println( "x es mayor que 5" );
}
else {
    System.out.println( "x es menor o igual que 5" );
}
```

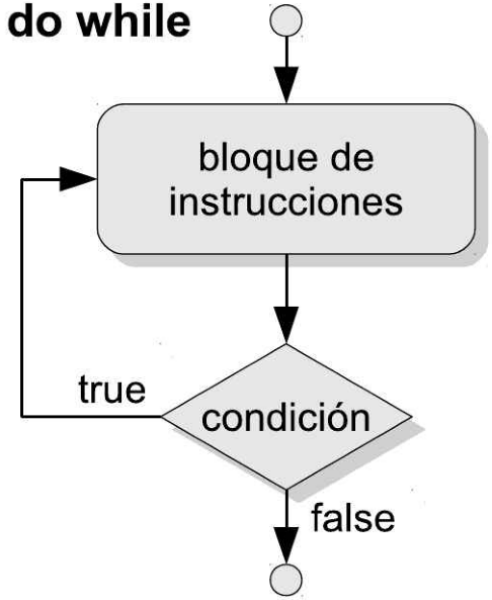



```
int mes = 2;

switch (mes) {
  case 1:
    System.out.println("El mes es Enero");
    break;
  case 2:
    System.out.println("El mes es Febrero");
    break;
  case 3:
    System.out.println("El mes es Marzo");
    break;
}
```

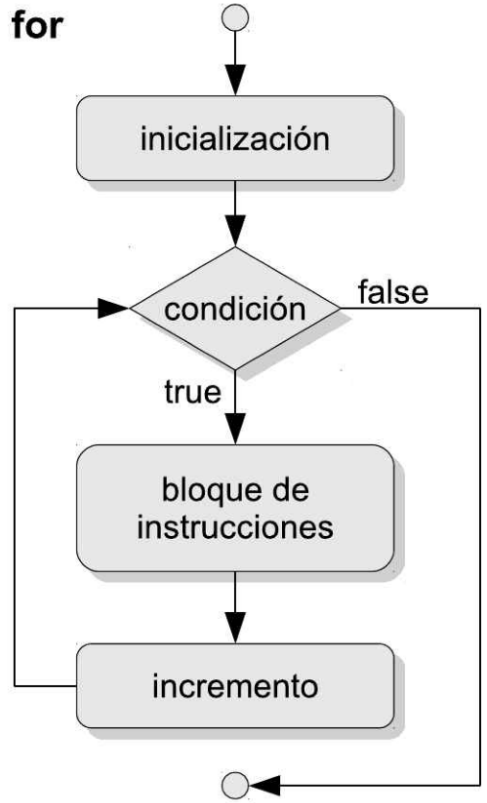
Estructuras repetitivas




```
int x = 1;

while (x <= 5) {

System.out.println(x);
  x++;
}
```




```
int x = 1;

do {
  System.out.println(x);
  x++;
} while (x <= 5);
```




```
int i;

for ( i=1 ; i<=10 ; i++ ) {
    System.out.print( "Hola " );
}
```

Paquetes en Java y API de Java

Un paquete en Java se usa para agrupar clases relacionadas. Piensa como si fuera **una carpeta dentro de una estructura de directorios**. Los paquetes se usan para evitar conflictos de nombres y para escribir código mantenible. Los paquetes se dividen en dos categorías:

- Built-in Packages (Paquetes de la API de Java)
- User-defined Packages (los paquetes definidos por el usuario)

```
import package.name.Class;   // Importa una clase nada más
import package.name.*;   // Importa el paquete entero
```

```
└── root
    └── alixar
        └── Clase.java
```

```
package alixar;

class Clase {
  public static void main(String[] args) {
    System.out.println("Paquete personalizado!");
  }
}
```

Salva el fichero como **Clase.java**, y compílalo:

```
C:\Users\usuario>javac Clase.java
```

Entonces compila el paquete:

```
C:\Users\usuario>javac -d . Clase.java
```

Esto fuerza al compilador a crear el paquete "alixar".
La opción `-d` especifica el destino donde salvar el fichero class. Puedes usar cualquier nombre de directorio, como c:/user (windows), o, si quieres mantener el paquete dentro del mismo directorio, puedes usar el punto "`.`", como en el ejemplo de arriba.
Note: El nombre del paquete debería ser escrito en minúsculas para evitar conflictos nombres de clases.