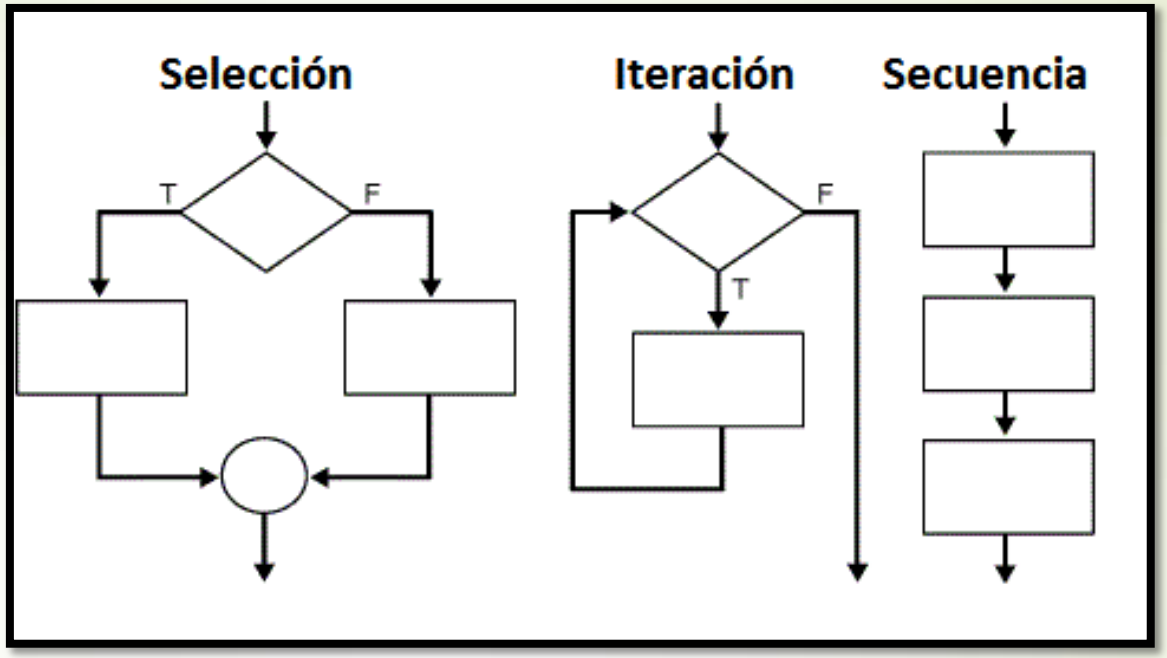
Desarrollo de Aplicaciones Web y Multiplataforma: Programación

DOCENTE: Daniel López Lozano

escuelaartegranada



Tema 2.
Estructuras de Control

Índice de contenidos

- Estructuras de repetición
  - ✓ Bucle for
  - ✓ Bucle while
  - ✓ Bucle do-while

Estructuras de repetición

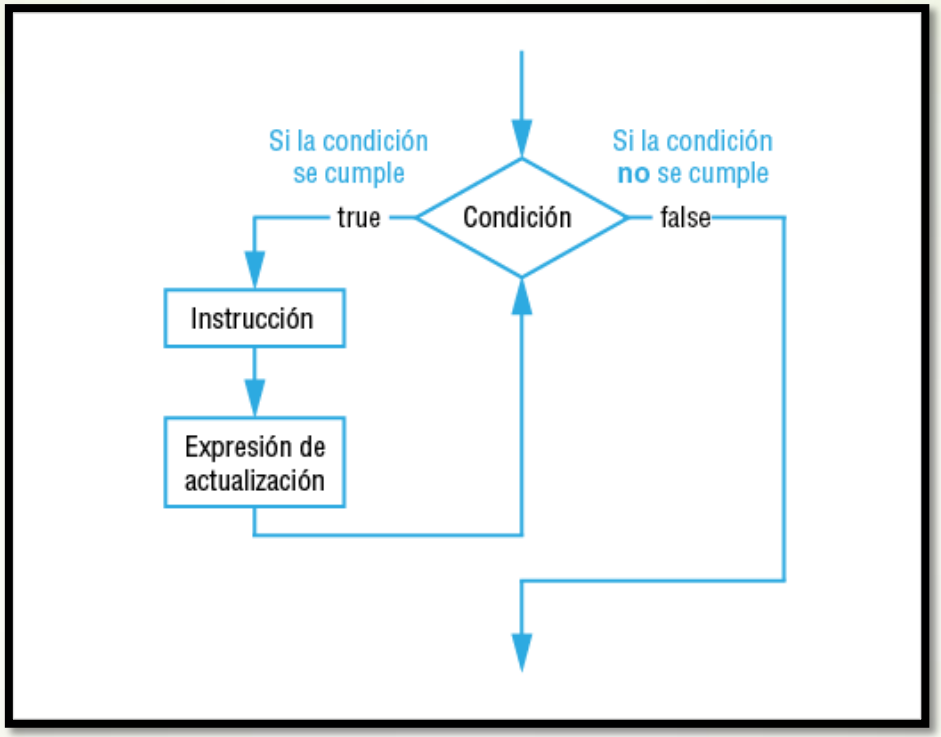
- Las estructuras de repetición o bucles nos van a permitir ejecutar un **bloque de código cero, una o más veces**. Existen tres tipos de bucles:
  - ✓ **Bucle for:** Permite ejecutar un bloque de código un número fijo y conocido de veces.
  - ✓ **Bucle while:** Permite ejecutar un bloque de código 0 o más veces.
  - ✓ **Bucle do-while:** Permite ejecutar un bloque de código 1 o más veces.

- El Bucle for permite ejecutar un bloque de código un **número fijo y conocido de veces**.

- Ese número fijo de veces se establece mediante un condición de <= ó =>.

```
for (expresión inicial; condición; incremento)
{
    // Instrucciones a ejecutar dentro del bucle.
}
```

Esquemas for




```
(contador = 0; contador < 10; contador ++)
```

expresión de inicialización | expresión de evaluación | expresión de actualización

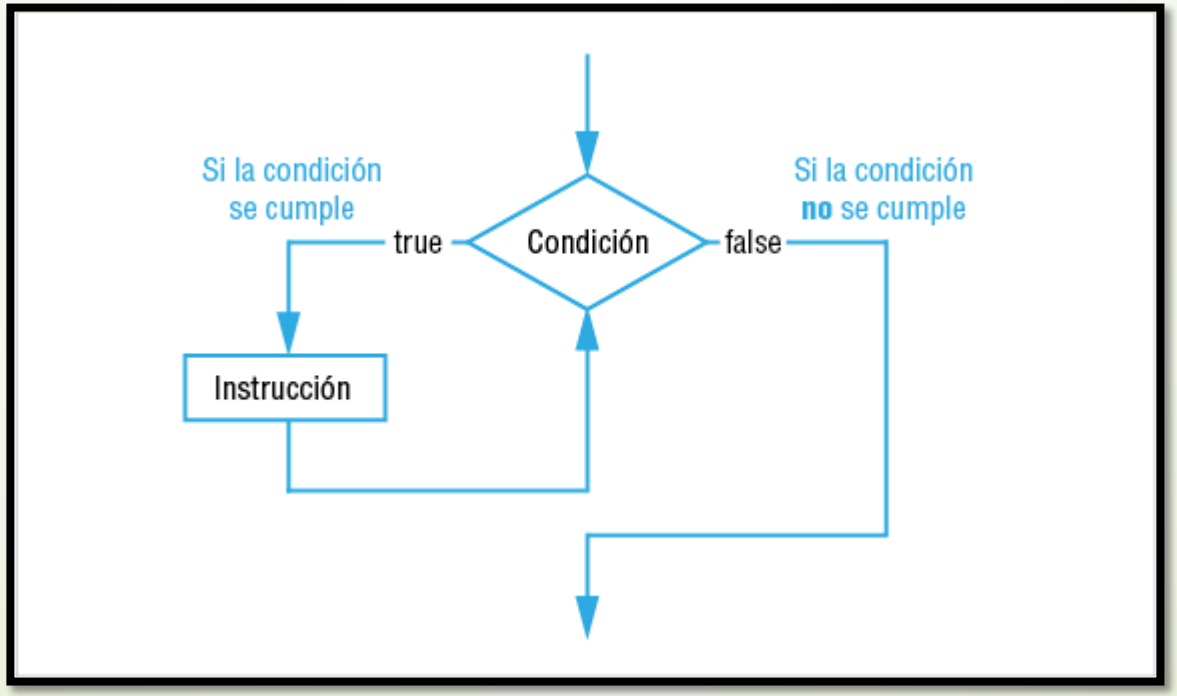
Ejemplos for

```
for (int i=1;i<=6;i++)
{
    System.out.println("Contador: "+i);
}
```

```
int num_alum;
double nota;
String nombre;
System.out.println("Introduce cuantos alumnos hay en clase");
num_alum=sc.nextInt();
for (int i=1;i<=num_alum;i++)
{
    System.out.println("Introduce el nombre y la nota del alumno:");
    nombre=sc.nextLine();
    nota=sc.nextDouble();
    System.out.println("La nota del alumno "+nombre+" es:"+nota);
}
```

- Con el bucle for, podemos repetir una tarea un número fijo de veces.

- El bucle while permite crear bucles que se ejecutan cero o más veces de manera indefinida.



Ejemplos while

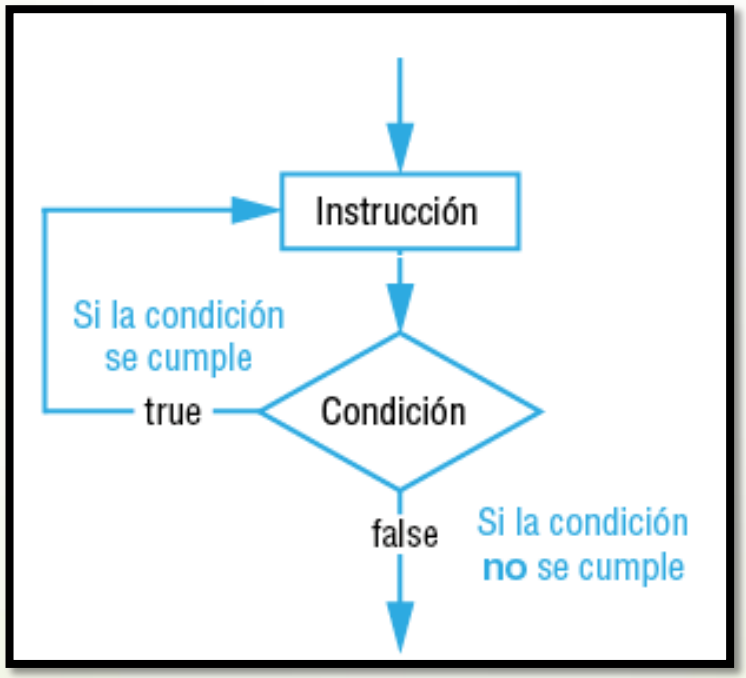
```java
int i=1;
while(i<=10)
{
   System.out.println("Contador" + i);
   i++;
}
```

```java
System.out.println("Introduce tu edad");
edad=sc.nextInt();
while(edad<0)
{
    System.out.println("Introduce tu edad");
    edad=sc.nextInt();
}
```

```java
int secreto=12;
int intento=-1;
while(intento!=secreto)
{
    System.out.println("¿Qué numero positivo es?");
    intento=sc.nextInt();
}
```

- El bucle do-while es una variante del bucle while **que se ejecuta siempre al menos una vez**.

- Permite escribir de manera más clara en ciertas situaciones



Ejemplos do-while

```java
int i=1;
do
{
   System.out.println("Contador" + i);
   i++;
}while(i<=10);
```

```java
do{
   System.out.println("Introduce tu edad");
   edad=sc.nextInt();
}while(edad<0);
```

```java
char respuesta;
int i=1;
do
{
   System.out.println("¿Desea salir?(S/N) Intento" + i);
   respuesta=sc.next().charAt();
   i++;
}while(respuesta!='s' && respuesta!='S');
```

- Realmente **los 3 bucles son equivalentes** y existe una teoría unificada que afirma que **cualquier algoritmo** se puede programar **usando solo el bucle while**.

- Existen situaciones donde usar un bucle **es más natural que usar otro**, quedando en general un código más legible.

- Algo parecido a los que nos sucede con **if y switch**.

- La clase **Random** proporciona un generador de números aleatorios para programas que usen **el azar** o necesiten de un conjunto de datos de prueba.

- Para ello existe su constructor y una serie de métodos parecidos a la clase Scanner para obtener números aleatorios.

```
Random generador=new Random();//Semilla aletoria por defecto
Random otro_gen=new Random(3816);//Semilla aletoria 3816

generador.nextInt(); //Un numero entre 0 y el MAX int
generador.nextInt(90); //Un numero entre 0 y 89
```

Para generar números aleatorios en distintos rangos

```
Random generador=new Random();

//Entre 1 y 100
generador.nextInt(100)+1;
//Entre 1 y X
generador.nextInt(X)+1;
//Entre 25 y 46
generador.nextInt(22)+25;
//Entre X e Y
generador.nextInt(Y-X+1)+X;
```

- Las **aplicaciones** de los números aleatorios en general pueden ser **el número resultante de tirar** un dado, número premiado en un **sorteo de lotería**, ruleta de un casino, máquina tragaperras y cualquier cuestión relacionada con el azar. También es interesante a la hora de hacer pruebas.

- Cada vez que creamos un objeto de la clase Random con la misma semilla obtendremos **la misma secuencia de números aleatorios**. Esto no es útil en el caso de **loterías**, pero puede ser útil en el caso de **juegos y/o simulaciones** que se repitan de la misma forma una y otra vez, etc.

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.