
Bucles y estructuras de decisión

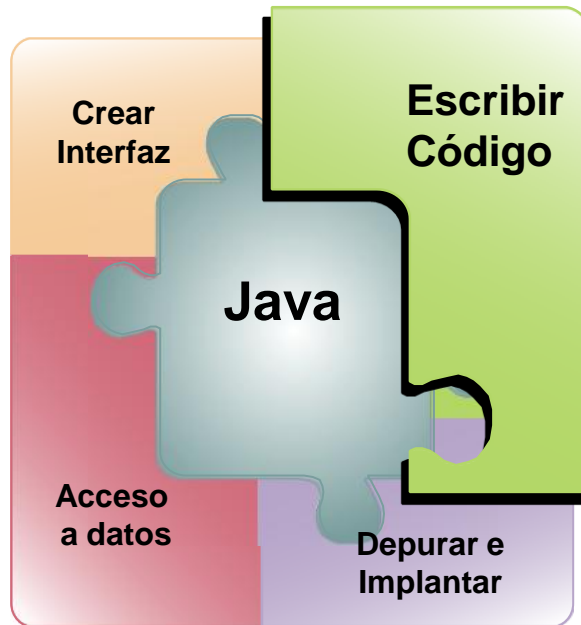
- Unidad 3

- Apuntes referenciados:

- ✓ A1.- Capítulo 2

- ✓ A2.- Tema II. 4

Descripción

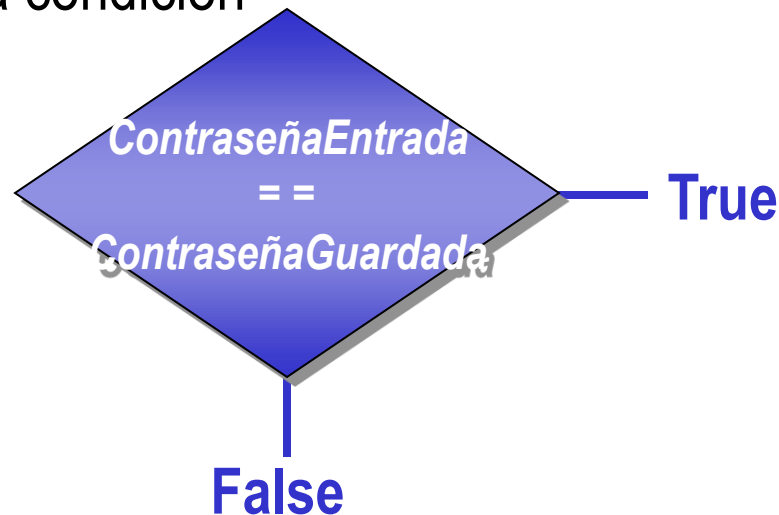


- **Expresiones condicionales**
- **Uso de estructuras de decisión**
- **Uso de estructuras de bucle condicionales**
- **Para ampliar...**

¿Que son las expresiones condicionales?

■ Expresiones condicionales:

- Incluyen una condición que debe evaluarse si es **true** o **false**
- Incluyen un operador para especificar cual es el resultado de la condición



Si la contraseña es la correcta, la condición es true

Cómo utilizar instrucciones if

Cómo utilizar instrucciones if...else

Cómo utilizar instrucciones if...else if...

Cómo utilizar instrucciones switch...case

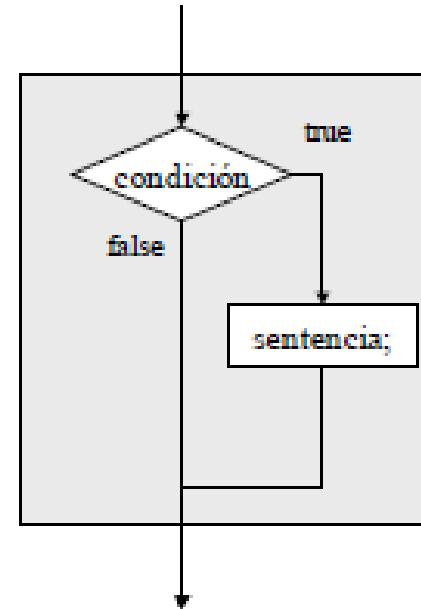
Operador ? :

USO DE ESTRUCTURAS DE DECISIÓN

Cómo utilizar instrucciones if...

```
if (condición)
    sentencia;
    ... o bien .....
```

```
if (condición) {
    sentencia 1;
    sentencia 2;
    ...
    sentencia n;
}
```



Cómo utilizar instrucciones if...

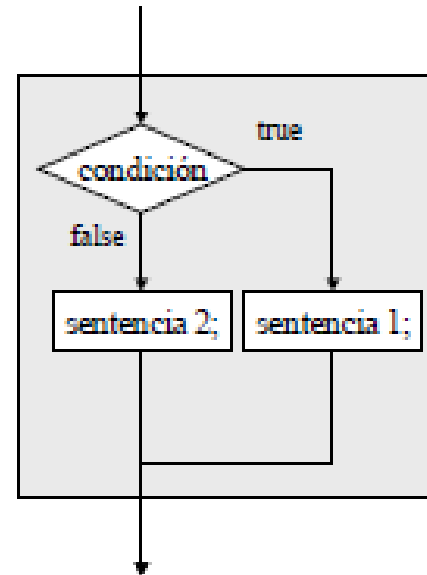
- Se utilizan para una decisión true o false
- Si la condición es true, se ejecutan las instrucciones que siguen a la instrucción if
- Si la condición es false, las instrucciones que siguen a la instrucción if no se ejecutan

```
if (carácter == 'a') {  
    // código para realizar si la condición se cumple  
    System.out.println( "Se ha leído el carácter a" );  
}
```

Cómo utilizar instrucciones if...else

```
if (condición)
    sentencia 1;
else
    sentencia 2;
    ..... o bien .....
if (condición) {
```

```
    sentencia 11;
    ...
    sentencia 1n;
} else {
    sentencia 21;
    ...
    sentencia 2n;
}
```



Cómo utilizar instrucciones if...else

- Se utilizan para una decisión con dos opciones como mínimo
- Si la condición es true, se ejecutarán las instrucciones que siguen a la instrucción if
- Si la condición es false, se ejecutarán las instrucciones que siguen a la instrucción else

```
if (edad < 18) {  
    // código para realizar si la condición se cumple  
    System.out.println( "no puedes salir a la hora del patio" );  
}  
else {  
    // código para realizar si la condición no se cumple  
    System.out.println( "puedes salir " );  
}
```


Cómo utilizar instrucciones if...else

```
if ( condición )
```

```
    Bloque de código a ejecutar si la condición es  
    cierta
```

```
else
```

```
    Bloque de código a ejecutar si la condición es  
    falsa
```

- La parte del e/se es opcional
- Se requieren llaves {... } si el bloque contiene más de una instrucción. La llaves son opcionales, pero recomendables si el bloque contiene una sola instrucción
- Un bloque de código puede ser simplemente la sentencia vacía
 - Requiere el punto-y-coma: ;
 - representa que en ese caso no se ha de ejecutar nada

Cómo utilizar instrucciones If...else if...

- Se utilizan para anidar instrucciones de decisión
- Dos estilos:

```
if (mes == 1)
    System.out.print("enero");
else
    if (mes == 2)
        System.out.print("febrero");
    else
        if (mes == 3)
            System.out.print("marzo");
        ...
    else
        print("no se");
```

```
if (mes == 1)
    System.out.print("enero");
else if (mes == 2)
    System.out.print("febrero");
else if (mes == 3)
    System.out.print("marzo");
...
else
    System.out.print("no se");
```

Cómo utilizar instrucciones switch... case

- **Construcción sintáctica muy compacta para seleccionar un bloque de código a ejecutar dependiendo de un valor**
- **Se utilizan como alternativa a instrucciones if...else anidadas**
- **Sólo funcionan sobre enteros, booleanos o caracteres**
 - Chequean que no hay duplicados
 - las condiciones tienen que ser excluyentes

```
switch (mes) {  
    case 1:  
        System.out.print("enero");  
        break;  
    case 2:  
        System.out.print("febrero")  
        ;  
        break;  
    ...  
    default:  
        System.out.print("no se");  
}
```

Cómo utilizar instrucciones switch... case

- Las sentencias “break” provocan la terminación de la sentencia condicional
 - Si no aparece “break” el código siguiente se sigue ejecutando
- El “default” es opcional
 - si no aparece no se ejecuta nada

```
switch (mes) {  
    case 1: case 3: case 5: case 7:  
    case 8: case 10: case 12:  
        dias = 31;  
        break;  
    case 4: case 6: case 9: case 11:  
        dias = 30;  
        break;  
    case 2:  
        if (bisiesto)  
            dias = 29;  
        else  
            dias = 28;  
        break;  
    default:  
        dias = 0;  
}
```

Directrices para elegir una estructura de decisión

- Las instrucciones **if...** se utilizan para controlar la ejecución de un único bloque de código
- Las instrucciones **if...else** se utilizan para controlar la ejecución de dos secciones de código mutuamente excluyentes
- Las instrucciones **case...** se utilizan cuando se dispone de una lista de valores posibles

Operador ? :

- **Es una forma compacta de decidir entre dos valores**
condición ? valor_1 : valor_2
 - si es cierta la condición se toma el primer valor
 - Si es falsa la condición se toma el segundo valor
- **Ambos valores deben ser del mismo tipo o tipos compatibles (vía *casting*)**

forma compacta

```
variable = condicion ? v1 : v2 ;
```

forma clásica

```
if (condicion)  
    variable = v1;  
else  
    variable = v2;
```

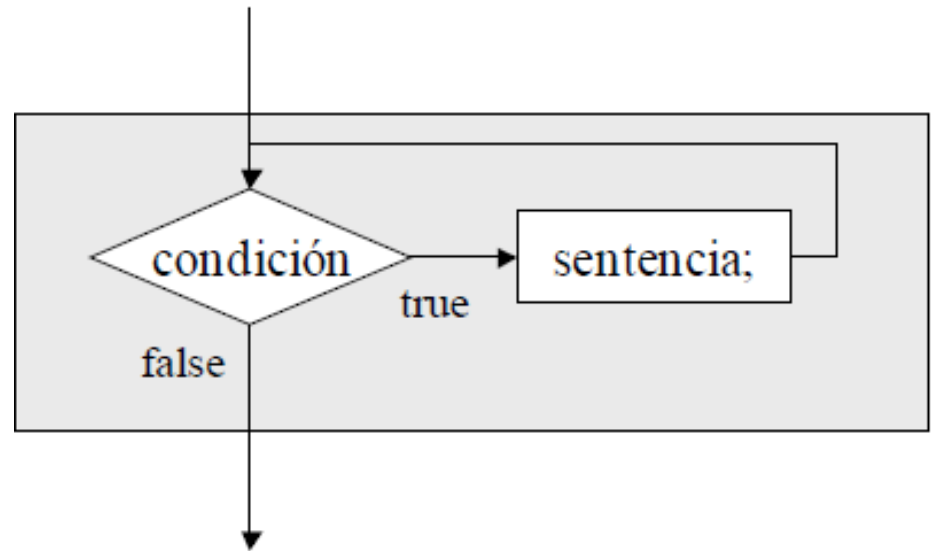
Uso de estructuras de bucle condicionales

- **Cómo utilizar instrucciones while**
- **Cómo utilizar instrucciones do...while**
- **Cómo utilizar instrucciones for...**
- **Cómo utilizar instrucciones break y continue**

Cómo utilizar las instrucciones while...

```
while (condición)  
    sentencia;  
..... o bien .....
```

```
while (condición) {  
    sentencia 1;  
    sentencia 2;  
    ...  
    sentencia ...;  
}
```



Cómo utilizar las instrucciones while...

- Ejecuta el código en el bucle sólo si la condición se evalúa como true y repite hasta que la expresión sea false
 - Se ejecutan cero o más veces
 - la condición de terminación se chequea al principio

```
//mostrar del 1 al 100

int num = 1;

while (num <100) {
    System.out.print(num);
    num++;
}
```

```
//calculo del factorial de n

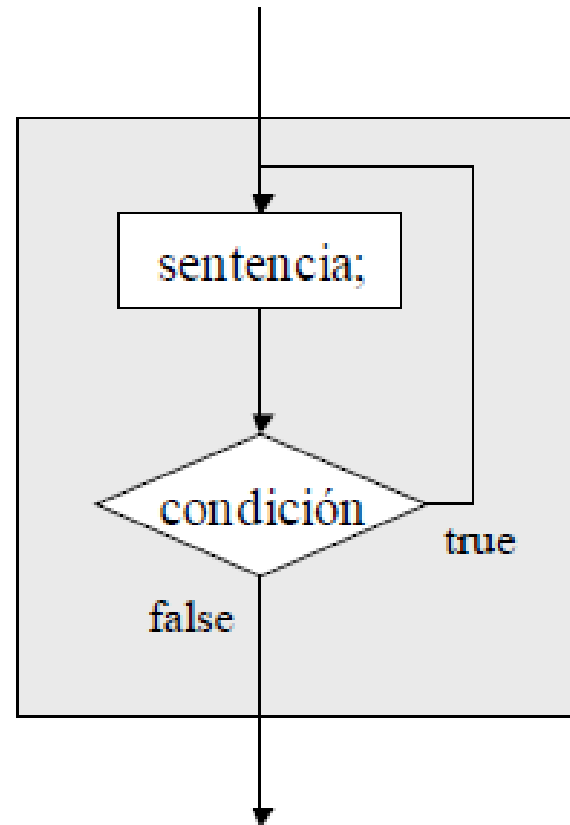
int fact = 1;

while (n > 0) {
    fact*= n;
    n--;
}
```

Cómo utilizar instrucciones do...while

```
do {  
    sentencia;  
} while (condición);  
..... o bien .....
```

```
do {  
    sentencia 1;  
    sentencia 2;  
    ...  
    sentencia ...;  
} while (condición);
```



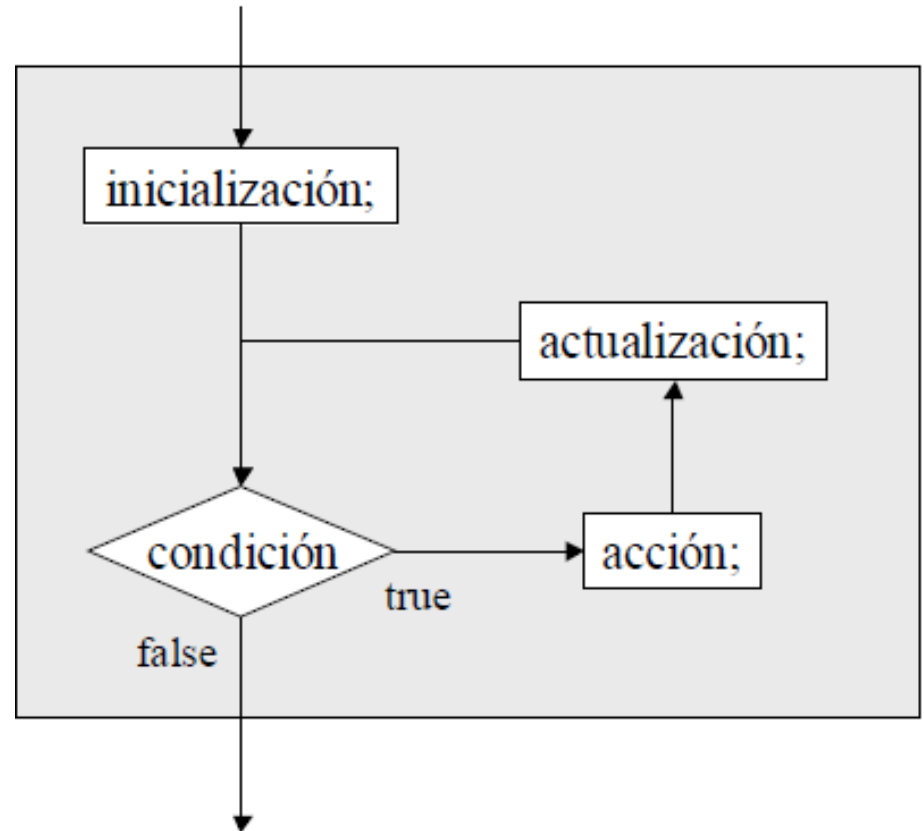
Cómo utilizar las instrucciones do...while

- **Ejecuta el código en el bucle y evalúa la condición. Repite hasta que la condición sea false**
 - Se ejecutan una o más veces
 - la condición de finalización se chequea al final

```
//Pedir una nota hasta que sea mayor que 5  
  
double nota;  
  
do {  
    nota = lector.nextDouble();  
} while (nota < 5.0);
```

Cómo utilizar instrucciones for...

```
for    (inicialización;  
        condición;  
        actualización)  
    acción;  
..... o bien .....  
for    (inicialización;  
        condición;  
        actualización) {  
    acción 1;  
    acción 2;  
    ...  
    acción ...;  
}
```



Cómo utilizar instrucciones for...

- Se utilizan cuando conocemos el número de veces que deseamos que se repita la ejecución de un código

```
for (i = 0; i < 10; i++)  
    System.out.print(i);
```

```
int fact = 1;  
for (n = m; n > 0; n--)  
    fact *= n;
```

Cómo utilizar instrucciones for...

- Los bucles “for” tradicionalmente emplean un contador que es una variable (local al bucle)
 - *Local al bucle quiere decir que su ámbito es el propio bucle, no pudiendo utilizarse dicha variable fuera del bucle*

```
for (int i = 0; i < 10; i++)  
    System.out.print(i);
```

- Los componentes sintácticos de un bucle son opcionales

Ejemplo

- Las tres construcciones de bucle (for, do-while y while) pueden utilizarse indistintamente realizando unas pequeñas variaciones en el programa
- La serie de Fibonacci es una serie de números enteros:
 - comienza con 1, 1,...
 - a partir del tercer término, el siguiente término de la serie se calcula como la suma de los dos anteriores.
 - 1, 1, 2, 3, 5, 8, 13, 21, 34, 55,
- El siguiente programa muestra en pantalla la serie de Fibonacci hasta el término n

Fibonacci. for...

```
import java.util.Scanner;

public class FibonnaciFor {

    public static void main(String args[]) {
        int numTerm, v1 = 1, v2 = 1, aux, cont;
        Scanner lector = new Scanner(System.in);

        System.out.println(" Términos de la serie Fibonacci?");
        numTerm = lector.nextInt();
        System.out.print("1,1");
        for (cont = 2; cont < numTerm; cont++) {
            aux = v2;
            v2 += v1;
            v1 = aux;
            System.out.print(", " + v2);
        }
        System.out.println();
    }
}
```


Fibonacci. while...

```
import java.util.Scanner;

public class FibonnaciWhile {

    public static void main(String args[]) {
        int numTerm, v1 = 1, v2 = 1, aux, cont = 2;
        Scanner lector = new Scanner(System.in);

        System.out.println(" Términos de la serie Fibonacci?");
        numTerm = lector.nextInt();
        System.out.print("1,1");
        while (cont++ < numTerm) {
            aux = v2;
            v2 += v1;
            v1 = aux;
            System.out.print(", " + v2);
        }
        System.out.println();
    }
}
```

Fibonacci. do..while

```
import java.util.Scanner;

public class FibonnaciDoWhile {

    public static void main(String args[]) {
        int numTerm, v1 = 0, v2 = 1, aux, cont = 1;
        Scanner lector = new Scanner(System.in);

        System.out.println(" Términos de la serie Fibonacci?");
        numTerm = lector.nextInt();
        System.out.print("1");
        do {
            aux = v2;
            v2 += v1;
            v1 = aux;
            System.out.print(", " + v2);
        } while (++cont < numTerm);
        System.out.println();
    }
}
```

break i continue

- Cuando un bucle está lanzado, java proporciona dos formas de forzarlo desde dentro
- **"break":**
 - provoca la terminación del bucle: lo aborta
- **"continue":**
 - provoca el comienzo de una nueva repetición: aborta la pasada actual
- **Se recomienda NO usar sentencias "break" i "continue" salvo que sea evidente su necesidad**