Estructuras de control de flujo en Java

Programación I

Grado en Ingeniería Informática
MDR, JCRdP y JDGD

Estructuras de control de flujo

- Bloques

- Estructuras de selección
  - De dos alternativas
  - De múltiples alternativas

- Estructuras de repetición
  - Preevaluada
  - Postevaluada
  - Controlada por un contador

Estructura de selección de dos alternativas

- Tiene la forma:

```
if (condición) instrucción1;
else instrucción2;
```

  - Los paréntesis que delimitan la condición **NO** son opcionales
  - La segunda línea (instrucción **`else`**) es opcional
- Si la condición es verdadera se ejecuta instrucción1
- Si es falsa se ejecuta instrucción2, si existe
- Cada instrucción puede ser un conjunto de sentencias encerradas entre llaves

Estructuras de selección múltiple

Construida con if/else:

```
if (condición1) instrucción1;
else if (condición2) instrucción2;
else if (condición3) instrucción3;
else instrucción4;
```

Estructuras de selección múltiple

Construida con switch/case/break/default

```
switch (expresión) {
      case Valor1:  instrucción1;
            break;
      case Valor2:  instrucción2;
            break;
      case Valor3:  instrucción3;
            break;
      default:      instrucción4;
}
```

Ejemplo de estructura switch

```
public class DiasMes {
   public static void main(String[] args) {
        int mes=2;          //podría ser un valor pasado por parámetro
        switch (mes) {
        case 2: System.out.println("28 días y si es año bisiesto 29");
                break;
        case 4:
        case 6:
        case 9:
        case 11: System.out.println("30 días");
                break;
        default: System.out.println("31 días");
        }
   }
}
```

Estructura de repetición preevaluada

- Tiene la forma:

```
while (condición) instrucción;
```

- Si la condición es verdadera se ejecuta instrucción y se repite el proceso
- Si el cuerpo del bucle consta de varias instrucciones se encierran entre llaves
- Los paréntesis que delimitan la condición no son opcionales

Estructura de repetición postevaluada

- Tiene la forma:

```
do
    instrucción;
while (condición);
```

- Se ejecuta instrucción y si la condición es verdadera se repite el proceso
- La instrucción se ejecuta al menos una vez
- Si el cuerpo del bucle consta de varias instrucciones se encierran entre llaves
- Los paréntesis que delimitan la condición no son opcionales

Ejemplo de estructura do-while

```
public class DoWhile {
  public static void main(String[] args) {
       int factorial=1, aux;
       int numero=8;   //podría ser un valor pasado por parámetro
       if (numero==0) System.out.println("Factorial de 0 es 1");
       else if (numero > 0) {
              aux=numero;
              do {
                     factorial*=aux;
                     aux--;
              } while (aux > 0);
              System.out.println("El factorial de "+numero+" es "+factorial);
       }
       else
              System.out.println("El factorial de "+numero+" no existe");
  }
}
```

Estructura de repetición controlada por un contador

- Tiene la forma:

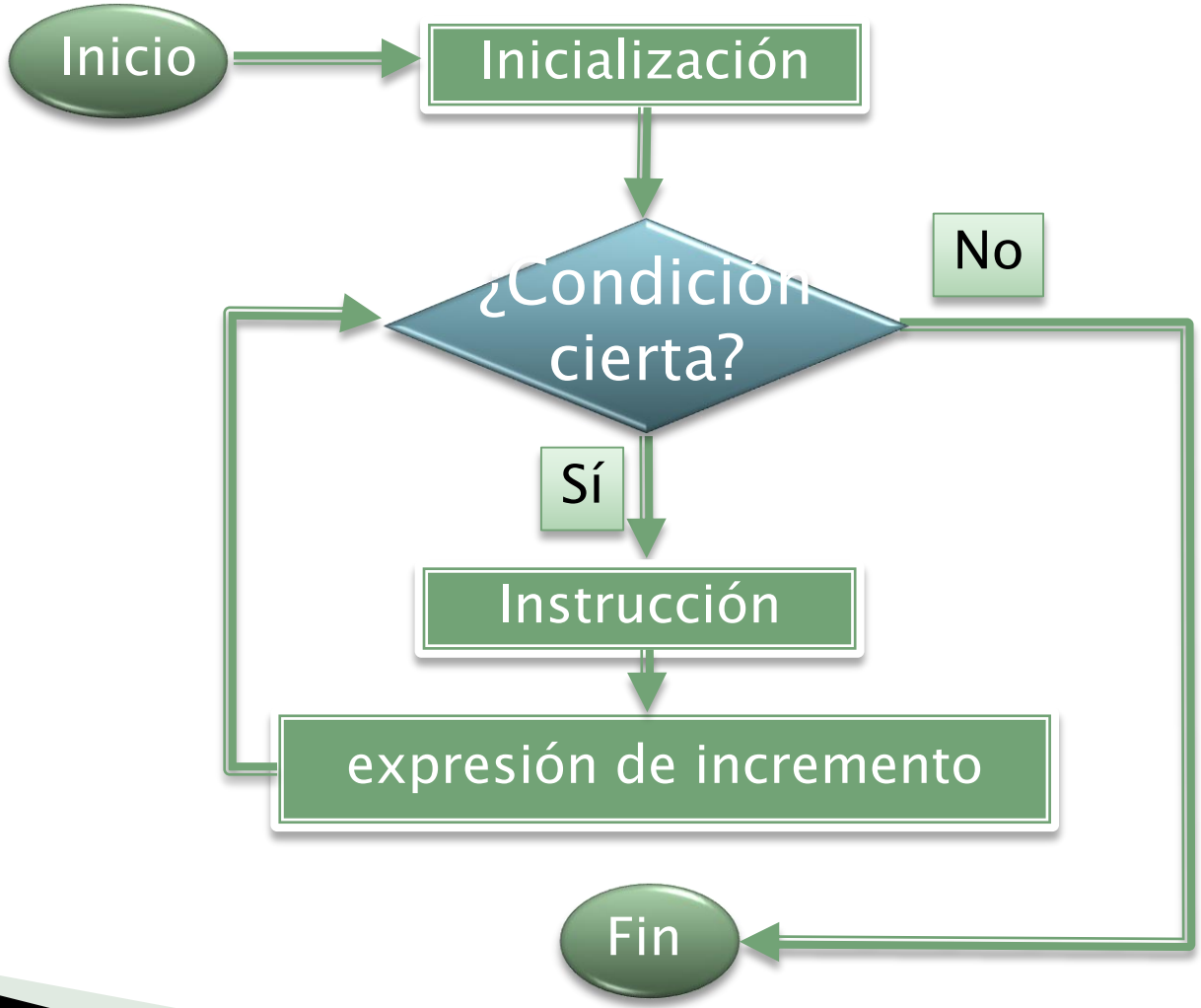
```
for (inicialización; condición; expresión) instrucción;
```

- Si el cuerpo del bucle consta de varias instrucciones se encierran entre llaves
- El operador "," se puede utilizar en la inicialización y en la expresión para evaluación secuencial

- Equivale a una estructura de repetición preevaluada de la forma

```
inicialización;
while (condición) {
        instrucción;
        expresión;
}
```

Diagrama de una estructura de repetición controlada por contador



Sentencia break

- Se puede situar dentro de un bucle o bucles anidados
- Cuando se ejecuta se abandona el bucle más interno
- Actúa como un salto a la instrucción siguiente al bucle donde está situada
- Se puede usar con etiqueta

Sentencia continue

- Interrumpe la iteración actual e inicia la siguiente
- En las estructuras de repetición preevaluada y postevaluada el flujo de ejecución salta a la condición
- En la estructura de repetición controlada por un contador el flujo de ejecución salta a la expresión de incremento y comprueba la condición
- Se puede usar con etiqueta

Ejemplo de break y continue

```
public class BreakYContinue {
   public static void main(String args[]) {
         int i, j, inicio=110, fin=244;   //podrían ser valores pasados por parámetro
         for (j=inicio;j<fin;j++) {
                  i=2;
                  if (j%i==0) continue;
                  while (i < j) {
                           i++;
                           if (i%2==0) continue;
                           if (j%i==0) break;
                  }
                  if (i==j)  {
                           System.out.println(j+" es el primer nº primo del rango");
                           break;
                  }
         }
         if (j==fin) System.out.println("No hay ningún nº primo en ese rango");
   }
}
```

Arrays

- Es una secuencia de objetos o de datos primitivos todos del mismo tipo
- Son objetos que se manejan con referencias
- Su tamaño no se puede modificar y no se pueden asignar.
- Se definen y utilizan con el operador de indexación [ ]

```
int[] v1;
int v2[];
```

- Se pueden inicializar la vez que se define la referencia

```
int v2[]={1, 2, 3, 4, 5};
```

- Después de su definición

```
v2=new int[5];
```

- Tienen asociado un atributo **length** que indica su tamaño

Ejemplo de Arrays

```
public class Array {
    public static void main(String[] args) {
        int[] a1={1, 2, 3, 4, 5};
        int[] a2;
        a2=a1;
        for (int i=0; i <a2.length; i++)
                a2[i]++;
        for (int i=0; i <a1.length; i++)
                System.out.println("a1["+i+"]= "+a1[i]);
    }
}
```

Arrays y for

- La sentencia **for** añade características específicas para recorrer arrays
- Formato: `for(T v: a) …`
- Donde T es el tipo de datos que almacena el array
- ***v*** es la variable que almacena cada valor del array
- ***a*** es el array a recorrer
- Ventajas:
  - Notación simple y compacta
  - Menor posibilidad de acceso a elementos inexistentes
- Desventaja:
  - Inevitable perdida de control sobre el recorrido

Ejemplo de Arrays y for

```
public class ArrayFor {
  public static void main(String[] args) {
      int[] a1={1, 2, 3, 4, 5};
      int s=0;
      for (int v : a1)
            s +=v;
      System.out.println("La suma del vector:");
      for (int v : a1)
            System.out.println(v);
      System.out.println("Da: "+s);
  }
}
```