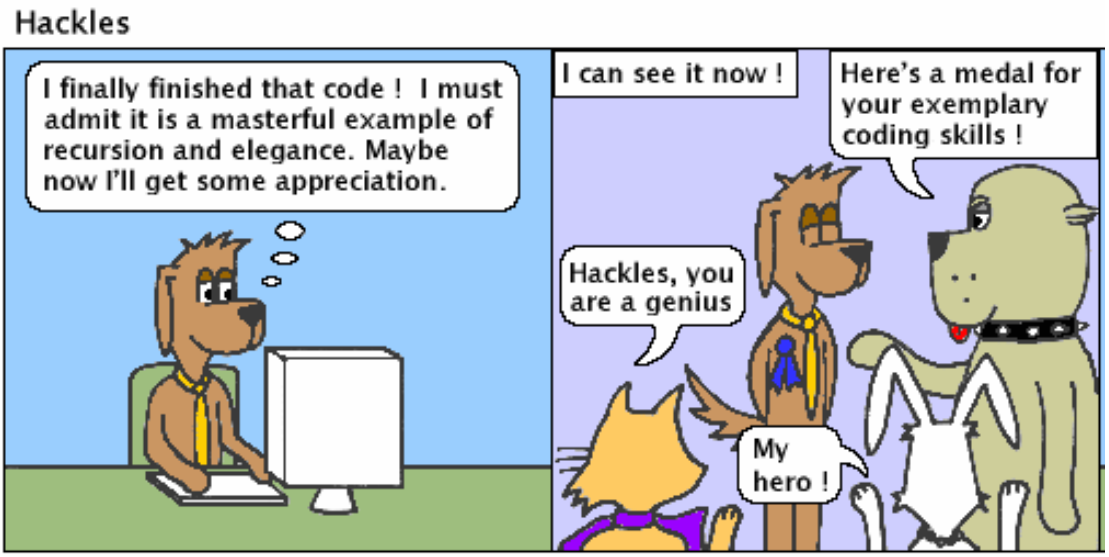
CONTROL DE FLUJO EN JAVA: SECUENCIAS, BIFURCACIONES Y CICLOS

LUIS ERNESTO
RUBIO TORRES

NOVIEMBRE 2018

El control del flujo de procesamiento en Java



Las estructuras de control determinan la secuencia de ejecución de las sentencias de un programa.

Las estructuras de control se dividen en tres categorías:

Secuencial
Condicional o Selectiva
Iterativa o Repetitiva.

ESTRUCTURA SECUENCIAL

El orden en que se ejecutan por defecto las sentencias de un programa es secuencial. Esto significa que las sentencias se ejecutan en secuencia, una después de otra, en el orden en que aparecen escritas dentro del programa.

La estructura secuencial está formada por una sucesión de instrucciones que se ejecutan en orden una a continuación de la otra.

Cada una de las instrucciones están separadas por el carácter punto y coma (;).
Las instrucciones se suelen agrupar en bloques.

El bloque de sentencias se define por el carácter llave de apertura ({) para marcar el inicio del mismo, y el carácter llave de cierre (}) para marcar el final.

Ejemplo:

```
{
instrucción 1;
instrucción 2;
instrucción 3;
}
```

En Java si el bloque de sentencias está constituido por una única sentencia no es obligatorio el uso de las llaves de apertura y cierre ({ }), aunque sí recomendable.

Ejemplo de programa Java con estructura secuencial: Programa que lee dos números por teclado y los muestra por pantalla.

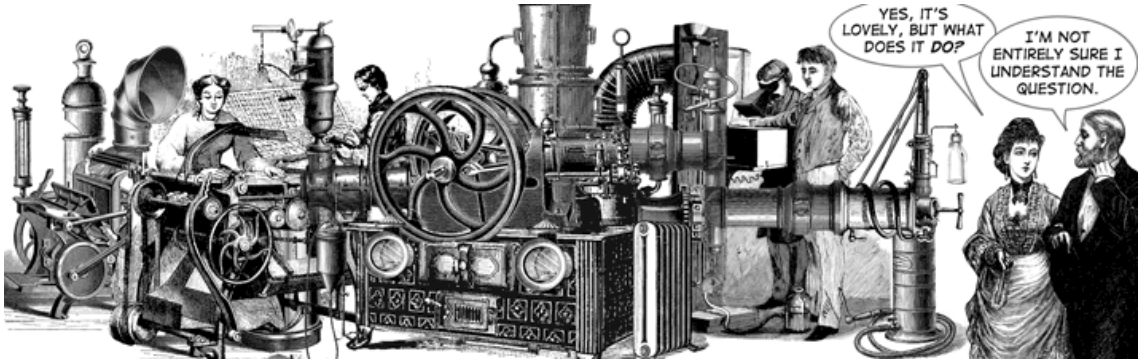
```
/* Programa que lea dos números por teclado y los muestre por pantalla.
*/
import java.util.*;
public class Main {
   public static void main(String[] args){
      //declaración de variables
      int n1, n2;
      Scanner sc = new Scanner(System.in);
      //leer el primer número
      System.out.println("Introduce un número entero: ");
      n1 = sc.nextInt();     //lee un entero por teclado
      //leer el segundo número
      System.out.println("Introduce otro número entero: ");
      n2 = sc.nextInt();     //lee un entero por teclado

      //mostrar resultado
      System.out.println("Ha introducido los números: " + n1 + " y " + n2);
   }
}
```

Ejemplo de programa Java con estructura secuencial: Programa que lee dos números de tipo double por teclado y calcula y muestra por pantalla su suma, resta y multiplicación.

```
/*
 * Programa que lee dos números de tipo double por teclado
 * y muestra su suma, resta y multiplicación.
 */
import java.util.*;
public class Main {
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      double numero1, numero2;
      System.out.println("Introduce el primer número:");
      numero1 = sc.nextDouble();
      System.out.println("Introduce el segundo número:");
      numero2 = sc.nextDouble();
      System.out.println("Números introducido: " + numero1 + "  " + numero2);
      System.out.println
        (numero1 + " + " + numero2 + " = " + (numero1+numero2));
      System.out.println
        (numero1 + " - " + numero2 + " = " + (numero1-numero2));
      System.out.println
        (numero1 + " * " + numero2 + " = " + numero1*numero2);
   }
}
```

Para modificar el orden de ejecución de las instrucciones de un programa Java se utilizan las estructuras condicionales y repetitivas.

ESTRUCTURA CONDICIONAL, ALTERNATIVA O SELECTIVA



La estructura condicional determina si se ejecutan unas instrucciones u otras según se cumpla o no una determinada condición.
En java la estructura condicional se implementa mediante:

- Instrucción if.
- Instrucción switch.
- Operador condicional ? :

INSTRUCCION if

Puede ser del tipo:

- Condicional simple: if
- Condicional doble: if ... else ...
- Condicional múltiple: if .. else if ..

La condición debe ser una **expresion booleana** es decir debe dar como resultado un valor booleano (**true ó false**).

Condicional simple: se evalúa la condición y si ésta se cumple se ejecuta una determinada acción o grupo de acciones. En caso contrario se saltan dicho grupo de acciones.

```
if(expresión_booleana){
   instrucción 1
   instrucción 2
   .......
}
```

Si el bloque de instrucciones tiene **una sola instrucción** no es necesario escribir las llaves { } aunque para evitar confusiones se recomienda escribir las llaves siempre.

Ejemplo de programa Java con estructura condicional: Programa que pide por teclado la nota obtenida por un alumno y muestra un mensaje si el alumno ha aprobado.

```
/*
 * Programa que pide una nota por teclado y muestra un mensaje si la nota es
 * mayor o igual que 5
 */
import java.util.*;
public class Ejemplo0If {
   public static void main( String[] args ){
      Scanner sc = new Scanner( System.in );
      System.out.print("Nota: ");
      int nota = sc.nextInt();
      if (nota >= 5 ){
         System.out.println("Enorabuena!!");
         System.out.println("Has aprobado");
      }
   }
}
```

Condicional doble: Se evalúa la condición y si ésta se cumple se ejecuta una determinada instrucción o grupo de instrucciones. Si no se cumple se ejecuta otra instrucción o grupo de instrucciones.

```
if(expresión booleana){
   instrucciones 1
}
else{
      instrucciones 2
                                                                    }
```

Ejemplo de programa Java que contiene una estructura condicional doble: Programa que lee la nota de un alumno y muestra si el alumno ha aprobado o no.

```
/*
 * Programa que pide una nota por teclado y muestra si se ha aprobado o no
 */
import java.util.*;
public class Ejemplo0If {
   public static void main( String[] args ){
      Scanner sc = new Scanner( System.in );
      System.out.print("Nota: ");
      int nota = sc.nextInt();
      if (nota >= 5 ){
         System.out.println("Enorabuena!!");
         System.out.println("Has aprobado");
      }
      else
         System.out.println("Lo Siento, has suspendido");
   }
}
```

Otro ejemplo de programa Java que contiene una estructura condicional doble: Calcular si un número es par. El programa lee un número por teclado y muestra un mensaje indicando si es par o impar.

```
/*
 * programa que pide un número por teclado y calcula si es par o impar
 */
import java.util.*;
public class EjemploIf {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int num;
      System.out.println("Introduzca numero: ");
      num = sc.nextInt();
      if ((num%2)==0)
        System.out.println("PAR");
      else
         System.out.println("IMPAR");
   }
}
```

Condicional múltiple: Se obtiene anidando sentencias if ... else. Permite construir estructuras de selección más complejas.

```
if (expresion_booleana1)
instruccion1;
else if (expresion_booleana2)
instruccion2;
      else
instruccion3;
```

Cada else se corresponde con el if más próximo que no haya sido emparejado.
Una vez que se ejecuta un bloque de instrucciones, la ejecución continúa en la siguiente instrucción que aparezca después de las sentencias if .. else anidadas.

Ejemplo de programa Java que contiene una estructura condicional múltiple: Programa que lee una hora (número entero) y muestra un mensaje según la hora introducida.

```
/*
 * Programa que muestra un saludo distinto según la hora introducida
 */
import java.util.*;
public class Ejemplo2If {
   public static void main(String[] args) {
     Scanner sc = new Scanner(System.in);
     int hora;
     System.out.println("Introduzca una hora (un valor entero): ");
     hora = sc.nextInt();
     if (hora >= 0 && hora < 12)
        System.out.println("Buenos días");
     else if (hora >= 12 && hora < 21)
         System.out.println("Buenas tardes");
     else if (hora >= 21 && hora < 24)
          System.out.println("Buenas noches");
     else
          System.out.println("Hora no válida");
   }
}
```

Ejemplo de programa Java que contiene una estructura condicional múltiple: Programa que lee una nota (número entero entre 0 y 10) y muestra la calificación equivalente en forma de texto.

```
/*
 * programa que lee una nota y escribe la calificación correspondiente
 */
import java.util.*;
public class Ejemplo3If {
   public static void main(String[] args) {
     Scanner sc = new Scanner(System.in);
     double nota;
     System.out.println("Introduzca una nota entre 0 y 10: ");
     nota = sc.nextDouble();
     System.out.println("La calificación del alumno es ");
     if(nota < 0 || nota > 10)
       System.out.println("Nota no válida");
     else if(nota==10)
         System.out.println("Matrícula de Honor");
     else if (nota >= 9)
         System.out.println("Sobresaliente");
     else if (nota >= 7)
         System.out.println("Notable");
     else if (nota >= 6)
         System.out.println("Bien");
     else if (nota >= 5)
         System.out.println("Suficiente");
     else
         System.out.println("Suspenso");
   }
}
```

Comparar String en Java

Para comparar el contenido de dos Strings en Java se usa el método **equals**:
if ((cadena1.equals(cadena2))
En caso de que una cadena coincida exactamente con una constante se puede usar ==
String nombre = "Lucas";
if (nombre == "Lucas")
Para comparar Strings en el orden alfabético se usa el método **compareTo**
if (cadena1.compareTo(cadena2) < 0) // cadena1 antes que cadena2
if (cadena1.compareTo(cadena2) > 0) // cadena1 después que cadena2
if (cadena1.compareTo(cadena2) == 0) // cadena1 igual que cadena2

INSTRUCCION switch

Se utiliza para seleccionar una de entre múltiples alternativas.
La forma general de la instrucción switch en Java es la siguiente:

```
switch (expresión){
case valor 1:
instrucciones;
break;
case valor 2:
instrucciones;
break;
. . .
default:
instrucciones;
}
```

La instrucción switch se puede usar con datos de tipo byte, short, char e int. También con tipos enumerados y con las clases envolventes Character, Byte, Short e Integer. A partir de Java 7 también pueden usarse datos de tipo String en un switch.

Funcionamiento de la instrucción switch:

- Primero se evalúa la expresión y salta al case cuya constante coincida con el valor de la expresión.

– Se ejecutan las instrucciones que siguen al case seleccionado hasta que se encuentra un break o hasta el final del switch. El break produce un salto a la siguiente instrucción a continuación del switch.

– Si ninguno de estos casos se cumple se ejecuta el bloque default (si existe). No es obligatorio que exista un bloque default y no tiene porqué ponerse siempre al final, aunque es lo habitual.

Ejemplo de programa Java que contiene una instrucción switch: Programa que lee por teclado un mes (número entero) y muestra el nombre del mes.

```
/*
 * Programa que pide un número de mes y muestra el nombre correspondiente
 */
import java.util.*;
public class Ejemplo0Switch {
   public static void main(String[] args) {
      int mes;
      Scanner sc = new Scanner(System.in);
      System.out.print("Introduzca un numero de mes: ");
      mes = sc.nextInt();
      switch (mes)
      {
            case 1: System.out.println("ENERO");
                  break;
            case 2: System.out.println("FEBRERO");
                  break;
            case 3: System.out.println("MARZO");
                  break;
            case 4: System.out.println("ABRIL");
                  break;
            case 5: System.out.println("MAYO");
                  break;
            case 6: System.out.println("JUNIO");
                  break;
            case 7: System.out.println("JULIO");
                  break;
            case 8: System.out.println("AGOSTO");
                  break;
            case 9: System.out.println("SEPTIEMBRE");
                  break;
            case 10: System.out.println("OCTUBRE");
                  break;
            case 11: System.out.println("NOVIEMBRE");
                  break;
            case 12: System.out.println("DICIEMBRE");
                  break;
            default : System.out.println("Mes no válido");
      }
   }
}
```

Ejemplo de programa Java que contiene una una instrucción switch: Programa que lee dos números enteros por teclado y un operador (de tipo carácter) y muestra el resultado de la operación.

```
/*
 * Programa que pide dos números y un operador y muestra el resultado
 */
import java.util.*;
import java.io.*;
public class Ejemplo1Switch {
   public static void main(String[] args) throws IOException{
      int A,B, Resultado = 0 ;
      char operador;
      boolean calculado = true;
      Scanner sc = new Scanner(System.in);
      System.out.print("Introduzca un numero entero:");
      A = sc.nextInt();
      System.out.print("Introduzca otro numero entero:");
      B = sc.nextInt();
      System.out.print("Introduzca un operador (+,-,*,/):");
      operador = (char)System.in.read();
      switch (operador) {
            case '-' : Resultado = A - B;
                    break;
            case '+' : Resultado = A + B;
                    break;
            case '*' : Resultado = A * B;
                    break;
            case '/' : if(B!=0)
                        Resultado = A / B;
                      else{
                        System.out.println("\nNo se puede dividir por cero");
                        calculado = false;
                      }
                      break;
            default : System.out.println("\nOperador no valido");
                     calculado = false;

      }
      if(calculado){
         System.out.println("\nEl resultado es: " + Resultado);
      }
   }
}
```

OPERADOR CONDICIONAL ? :

Se puede utilizar en sustitución de la sentencia de control if-else.
Los forman los caracteres **?** y **:**

Se utiliza de la forma siguiente:

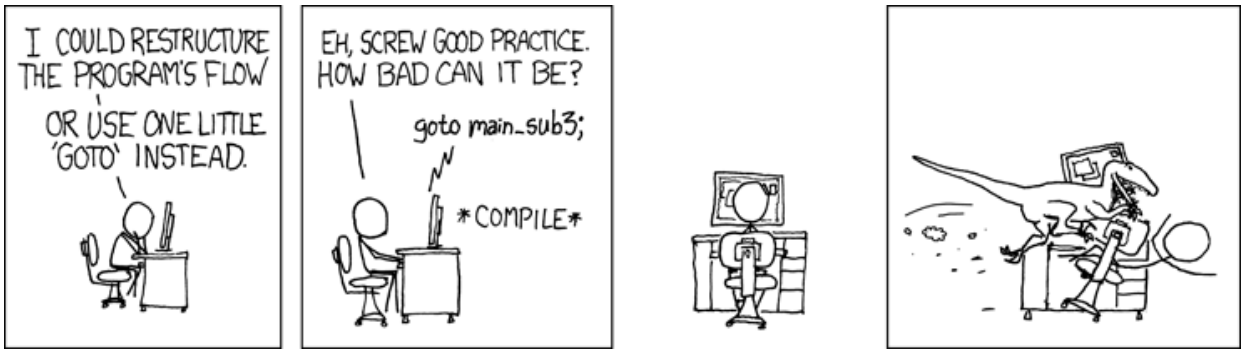
expresión1 **?** expresión2 **:** expresión3

Si expresión1 es cierta entonces se evalúa expresión2 y éste será el valor de la expresión condicional. Si expresión1 es falsa, se evalúa expresión3 y éste será el valor de la expresión condicional.

Ejemplo de programa Java que contiene un operador condicional: Programa que calcula y muestra si un número que se lee por teclado es par o impar.

```
 /*
  * programa que pide un número por teclado y calcula si es par o impar
  */
import java.util.*;
public class Ejemplo1OperadorCondicional {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int num;
      System.out.println("Introduzca numero: ");
      num = sc.nextInt();
      System.out.println((num%2)==0 ? "PAR" : "IMPAR");
   }
}
```

ESTRUCTURA ITERATIVA O REPETITIVA



Permiten ejecutar de forma repetida un bloque específico de instrucciones.
Las instrucciones se repiten mientras o hasta que se cumpla una determinada condición. Esta condición se conoce como **condición de salida**.
Tipos de estructuras repetitivas:

- ciclo while
- ciclo do – while
- ciclo for

CICLO WHILE

Las instrucciones se repiten mientras la condición sea cierta. La condición **se comprueba al principio** del bucle por lo que las acciones se pueden ejecutar **0 ó más veces**.

La ejecución de un bucle while sigue los siguientes pasos:

1. Se evalúa la condición.
2. Si el resultado es false las instrucciones no se ejecutan y el programa sigue ejecutándose por la siguiente instrucción a continuación del while.
3. Si el resultado es true se ejecutan las instrucciones y se vuelve al paso 1

Ejemplo de programa Java que contiene una instrucción while:

Programa que lee números por teclado. La lectura acaba cuando el número introducido sea negativo. El programa calcula y muestra la suma de los números leídos.

```
/*
 * Programa que lee números hasta que se lee un negativo y muestra la
 * suma de los números leídos
 */
import java.util.*;
public class Ejemplo1While {
   public static void main(String[] args) {
      int suma = 0, num;
      Scanner sc = new Scanner(System.in);
      System.out.print("Introduzca un número: ");
      num = sc.nextInt();
      while (num >= 0){
           suma = suma + num;
           System.out.print("Introduzca un número: ");
           num = sc.nextInt();
      }
      System.out.println("La suma es: " + suma );
   }
}
```

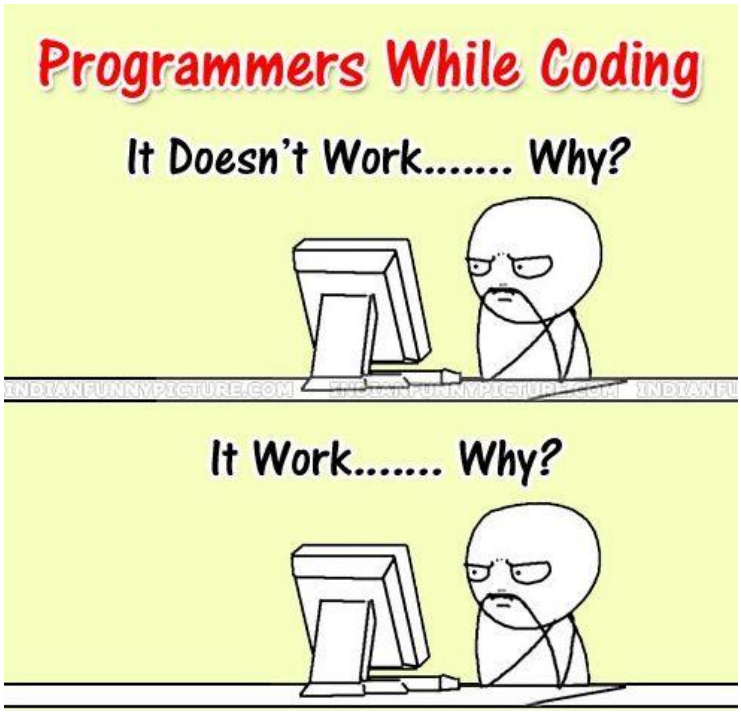
Ejemplo de programa Java que contiene una instrucción while:
Programa que lee un número entero N y muestra N asteriscos.

```
/*
 * programa que lee un número n y muestra n asteriscos
 */
import java.util.*;
public class Ejemplo2While {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n, contador = 0;
      System.out.print("Introduce un número: ");
      n = sc.nextInt();
      while (contador < n){
           System.out.println(" * ");
           contador++;
      }
   }
}
```

Ejemplo de programa Java con una instrucción while:

```
/*
 * programa que muestra una tabla de equivalencias entre
 * grados Fahrenheit y grados celsius
 */
public class Ejemplo3While {
   public static void main(String[] args) {
      final int VALOR_INICIAL = 10;  // limite inf. tabla
      final int VALOR_FINAL = 100;  // limite sup. tabla
      final int PASO = 10 ; // incremento
      int fahrenheit;
      double celsius;
      fahrenheit = VALOR_INICIAL;
      System.out.printf("Fahrenheit \t Celsius \n");
      while (fahrenheit <= VALOR_FINAL ){
           celsius = 5*(fahrenheit - 32)/9.0;
           System.out.printf("%7d \t %8.3f \n", fahrenheit, celsius);
           fahrenheit += PASO;
      }
   }
}
```

CICLO DO – WHILE



Las instrucciones se ejecutan mientras la condición sea cierta.

La condición **se comprueba al final** del bucle por lo que el bloque de instrucciones se ejecutarán **al menos una vez**. Esta es la diferencia fundamental con la instrucción while. Las instrucciones de un bucle while es posible que no se ejecuten si la condición inicialmente es falsa.

La ejecución de un bucle do - while sigue los siguientes pasos:

1. Se ejecutan las instrucciones a partir de do{
2. Se evalúa la condición.
3. Si el resultado es false el programa sigue ejecutándose por la siguiente instrucción a continuación del while.
4. Si el resultado es true se vuelve al paso 1

Ejemplo de programa Java que contiene una instrucción do while:
Programa que lee un número entero N. El número debe ser menor que 100.

```
/*
 * Programa que obliga al usuario a introducir un número menor que 100
 */
import java.util.*;
public class Ejemplo1DoWhile {
   public static void main(String[] args) {
      int valor;
      Scanner in = new Scanner( System.in );
      do {
         System.out.print("Escribe un entero < 100: ");
         valor = in.nextInt();
      }while (valor >= 100);
      System.out.println("Ha introducido: " + valor);
   }
}
```

Ejemplo de programa Java con una instrucción do while:

```
/*
 * Programa que lee un número entre 1 y 10 ambos incluidos
 */
import java.util.*;
public class Ejemplo2DoWhile {
   public static void main(String[] args) {
      int n;
      Scanner sc = new Scanner( System.in );
      do {
         System.out.print("Escribe un número entre 1 y 10: ");
         n = sc.nextInt();
      }while (n<1 || n >10);
      System.out.println("Ha introducido: " + n);
   }
}
```

CICLO FOR

Hace que una instrucción o bloque de instrucciones se repitan un **número determinado de veces mientras se cumpla la condición**.

La estructura general de una instrucción for en Java es la siguiente:

```
for(inicialización; condición; incremento/decremento){
instrucción 1;
.....
instrucción N;
}
```

A continuación de la palabra for y entre paréntesis debe haber siempre **tres zonas separadas por punto y coma**:

- zona de inicialización.
- zona de condición
- zona de incremento ó decremento.

Si en alguna ocasión no es necesario escribir alguna de ellas se pueden dejar en blanco, pero los dos punto y coma deben aparecer.

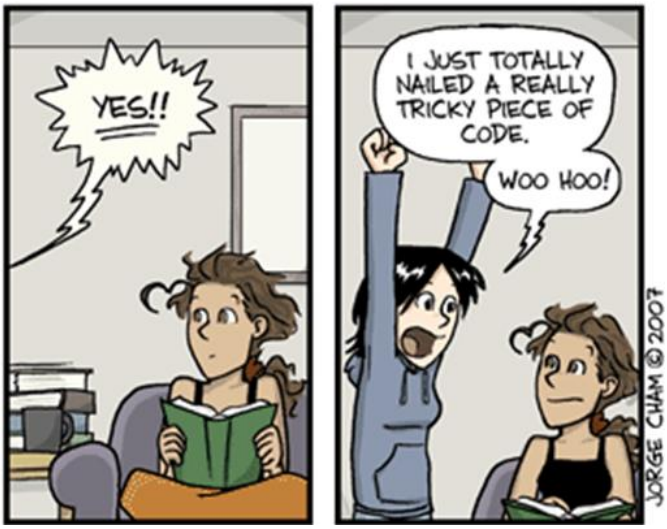
Inicialización es la parte en la que la variable o variables de control del bucle toman su valor inicial. Puede haber una o más instrucciones en la inicialización, separadas por comas. La inicialización se realiza solo una vez.

Condición es una expresión booleana que hace que se ejecute la sentencia o bloque de sentencias mientras que dicha expresión sea cierta. Generalmente en la condición se compara la variable de control con un valor límite.

Incremento/decremento es una expresión que decrementa o incrementa la variable de control del bucle.

La ejecución de un bucle for sigue los siguientes pasos:

1. Se inicializa la variable o variables de control (inicialización)
2. Se evalúa la condición.
3. Si la condición es cierta se ejecutan las instrucciones. Si es falsa, finaliza la ejecución del bucle y continúa el programa en la siguiente instrucción después del for.
4. Se actualiza la variable o variables de control (incremento/decremento)
5. Se vuelve al punto 2.



Ejemplo de programa Java que contiene una instrucción for:

```
/*
 * programa que muestra los números del 1 al 10
 */
public class Ejemplo0For {
   public static void main(String[] args) {
      int i;
      for(i=1; i<=10;i++)
         System.out.println(i + " ");
   }
}
```

La instrucción for del ejemplo anterior la podemos interpretar así:

Asigna a i el valor inicial 1, mientras que i sea menor o igual a 10 muestra i + " " , a continuación incrementa el valor de i y comprueba de nuevo la condición.

Ejemplo de programa Java con una instrucción for:

```
/*
 * programa que muestra los números del 10 al 1
 */
public class Ejemplo2For {
   public static void main(String[] args) {
      int i;
      for(i=10; i>0;i--)
         System.out.println(i + " ");
   }
}
```

Ejemplo de programa Java con una instrucción for:

```
/*
 * programa que muestra una tabla de equivalencias entre
 * grados Fahrenheit y grados celsius
 */
public class Ejemplo1For {
   public static void main(String[] args) {
      final int VALOR_INICIAL = 10; // limite inf. tabla
      final int VALOR_FINAL = 100; // limite sup. tabla
      final int PASO = 10 ; // incremento
      int fahrenheit;
      double celsius;
      fahrenheit = VALOR_INICIAL;
      System.out.printf("Fahrenheit \t Celsius \n");
      for (fahrenheit = VALOR_INICIAL; fahrenheit <= VALOR_FINAL;
           fahrenheit+= PASO) {
          celsius = 5*(fahrenheit - 32)/9.0;
          System.out.printf("%7d \t %8.3f \n", fahrenheit, celsius);
      }
   }
}
```

Ejemplo: En las zonas de inicialización e incremento/decremento puede aparecer más de una variable. En ese caso deben ir separadas por comas.

```
/*
 * programa que muestra el valor de a, b y su suma mientras que la suma de
 * ambas es menor de 10. En cada iteración el valor de a se incrementa en
 * 1 unidad y el de b en 2
 */
public class Ejemplo3For {
   public static void main(String[] args) {
      int a, b;
      for(a = 1, b = 1; a + b < 10; a++, b+=2){
         System.out.println("a = " + a + "  b = " + b + "  a + b = " + (a+b));
      }
   }
}
```

La salida de este programa es:

```
a = 1  b = 1  a + b = 2
a = 2  b = 3  a + b = 5
a = 3  b = 5  a + b = 8
```

Aunque la instrucción repetitiva for, al igual que las instrucciones while y do- while, se puede utilizar para realizar repeticiones cuando no se sabe a priori el número de pasadas por el bucle, esta instrucción es especialmente indicada para bucles donde se conozca el número de pasadas.
Como regla práctica podríamos decir que las instrucciones while y do-while se utilizan generalmente cuando no se conoce a priori el número de pasadas, y la instrucción for se utiliza generalmente cuando sí se conoce el número de pasadas.

Se ha de tener cuidado con escribir el punto y coma (;) después del paréntesis final del bucle for. Un bucle for generalmente no lleva punto y coma final.

Por ejemplo el bucle:

```
int i;
for (i = 1; i <= 10; i++);
{
    System.out.println("Elementos de Programación");
}
```

no visualiza la frase "Elementos de Programación" 10 veces, ni produce un mensaje de error por parte del compilador.

En realidad lo que sucede es que se visualiza una vez la frase "Elementos de Programación", ya que aquí la sentencia for es una sentencia vacía al terminar con un punto y coma (;).

La sentencia for en este caso hace que i empiece en 1 y acabe en 11 y tras esas iteraciones, se ejecuta la sentencia

System.out.println("Elementos de Programación");

BUCLES INFINITOS EN JAVA

Java permite la posibilidad de construir bucles infinitos, los cuales se ejecutarán indefinidamente, a no ser que provoquemos su interrupción. Tres ejemplos:

```
for(;;){
   instrucciones
}
for(;true;){
    instrucciones
}
while(true){
      instrucciones
}
```

BUCLES ANIDADOS

Bucles anidados son aquellos que incluyen instrucciones for, while o do-while unas dentro de otras.

Debemos tener en cuenta que las variables de control que utilicemos deben ser distintas.

Los anidamientos de estructuras tienen que ser correctos, es decir, que una estructura anidada dentro de otra lo debe estar totalmente.

Ejemplo de programa Java con bucles anidados:

```
/*
 * Programa que dibuja un rectángulo sólido de asteriscos.
 * El número de filas y columnas se pide por teclado
 */

import java.util.*;
public class Ejemplo1BuclesAnidados {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int filas, columnas;
      //leer número de filas hasta que sea un número > 0
      do{
        System.out.print("Introduce número de filas: ");
        filas = sc.nextInt();
      }while(filas<1);
      //leer número de columnas hasta que sea un número > 0
      do{
        System.out.print("Introduce número de columnas: ");
        columnas = sc.nextInt();
      }while(columnas<1);
      for(int i = 1; i<=filas; i++){    //filas
         for(int j = 1; j<=columnas; j++){   //columnas
             System.out.print(" * ");
         }
         System.out.println();
      }

   }
}
```

La salida de este programa para filas = 6 y columnas = 10 es:

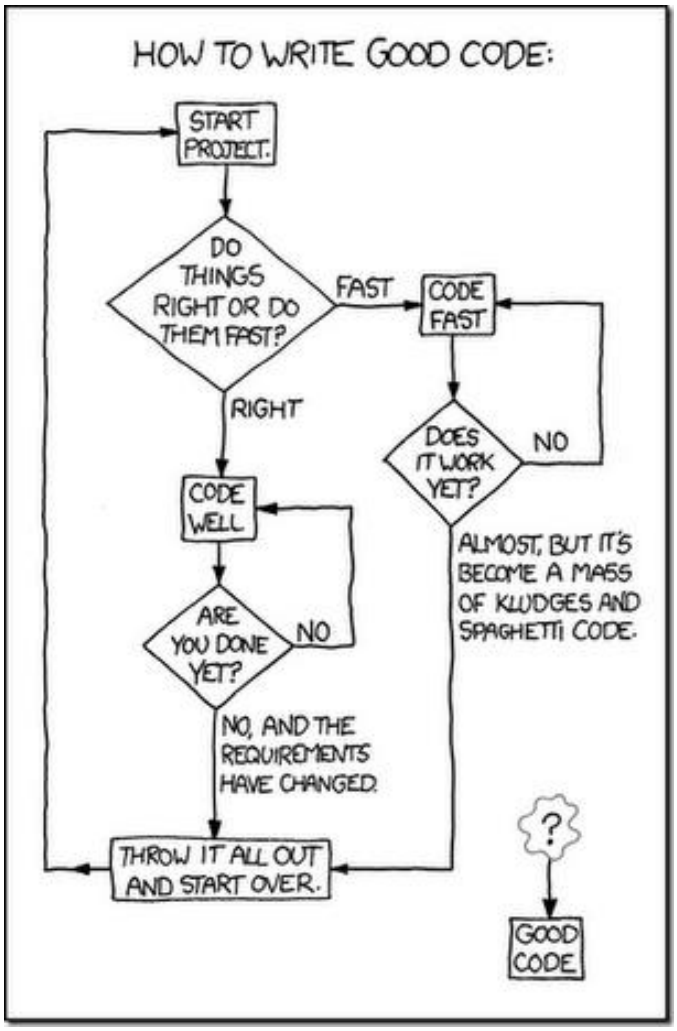
```
 *  *  *  *  *  *  *  *  *  *
 *  *  *  *  *  *  *  *  *  *
 *  *  *  *  *  *  *  *  *  *
 *  *  *  *  *  *  *  *  *  *
 *  *  *  *  *  *  *  *  *  *
 *  *  *  *  *  *  *  *  *  *
```

Ejemplo de programa Java con bucles anidados:

```
/*

 * Programa que muestra una tabla con las potencias de x (x x2 x3 x4)
 * para valores de x desde 1 hasta XMAX
 */
public class JavaApplication22 {
   public static void main(String[] args) {
      final int XMAX = 10;
      int x, n;
      //mostrar la cabecera de la tabla
      System.out.printf("%10s%10s%10s%10s%n", "x","x^2","x^3","x^4");
      for (x = 1; x <= XMAX; x++){   //filas
          for (n = 1; n <= 4; n++){   //columnas
              System.out.printf("%10.0f", Math.pow(x,n));
          }
          System.out.println();
      }
   }
}
```

El programa muestra por pantalla la siguiente tabla de potencias:

x	x^2	x^3	x^4
1	1	1	1
2	4	8	16
3	9	27	81
4	16	64	256
5	25	125	625
6	36	216	1296
7	49	343	2401
8	64	512	4096
9	81	729	6561
10	100	1000	10000



Fuente:
http://puntocomnoesunlenguaje.blogspot.mx/2012/04/estructuras-de-control.html