Ra-Ma®
Juan Carlos Moreno

Ciclo Formativo de Grado Superior:

Desarrollo de Aplicaciones Multimedia

Módulo: Programación

Autor: Juan Carlos Moreno

Editorial: RA-MA®

Ra-Ma®

Juan Carlos Moreno

Capítulo 3

Estructuras básicas de control

Índice

Objetivos del capítulo:

- Conocer y saber aplicar las estructuras básicas del lenguaje de programación Java.
- Controlar las excepciones básicas que pueda generar un programa.
- Reconocer la importancia de la documentación en el desarrollo de aplicaciones.
- Conocer el proceso de prueba y depuración de programas.

Las sentencias

Una **expresión** es una serie de variables/constantes/datos unidos por operadores (por ejemplo 2*PI*radio).
Una **sentencia** es una expresión que acaba en ;
(por ejemplo area = 2*PI*radio;).

Ra-Ma®
Juan Carlos Moreno

3.1 ESTRUCTURAS DE SELECCIÓN

3.1.1 ESTRUCTURAS IF

1

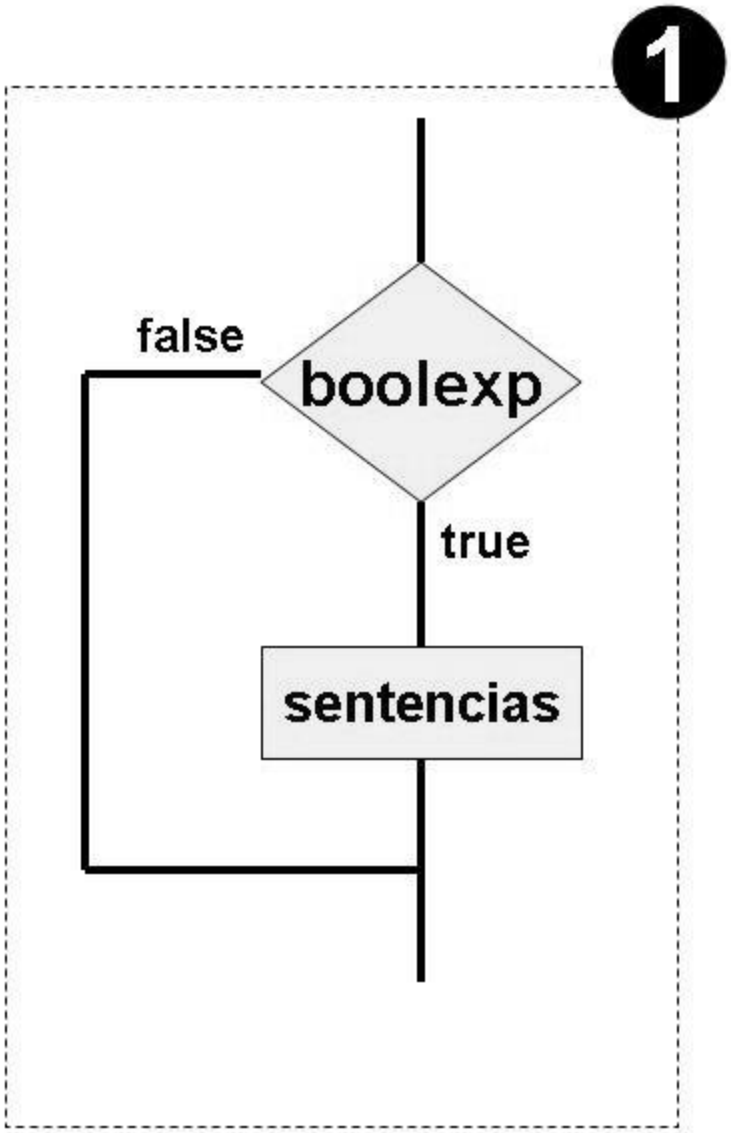
```
if (boolexp){
    <sentencias>
}
```

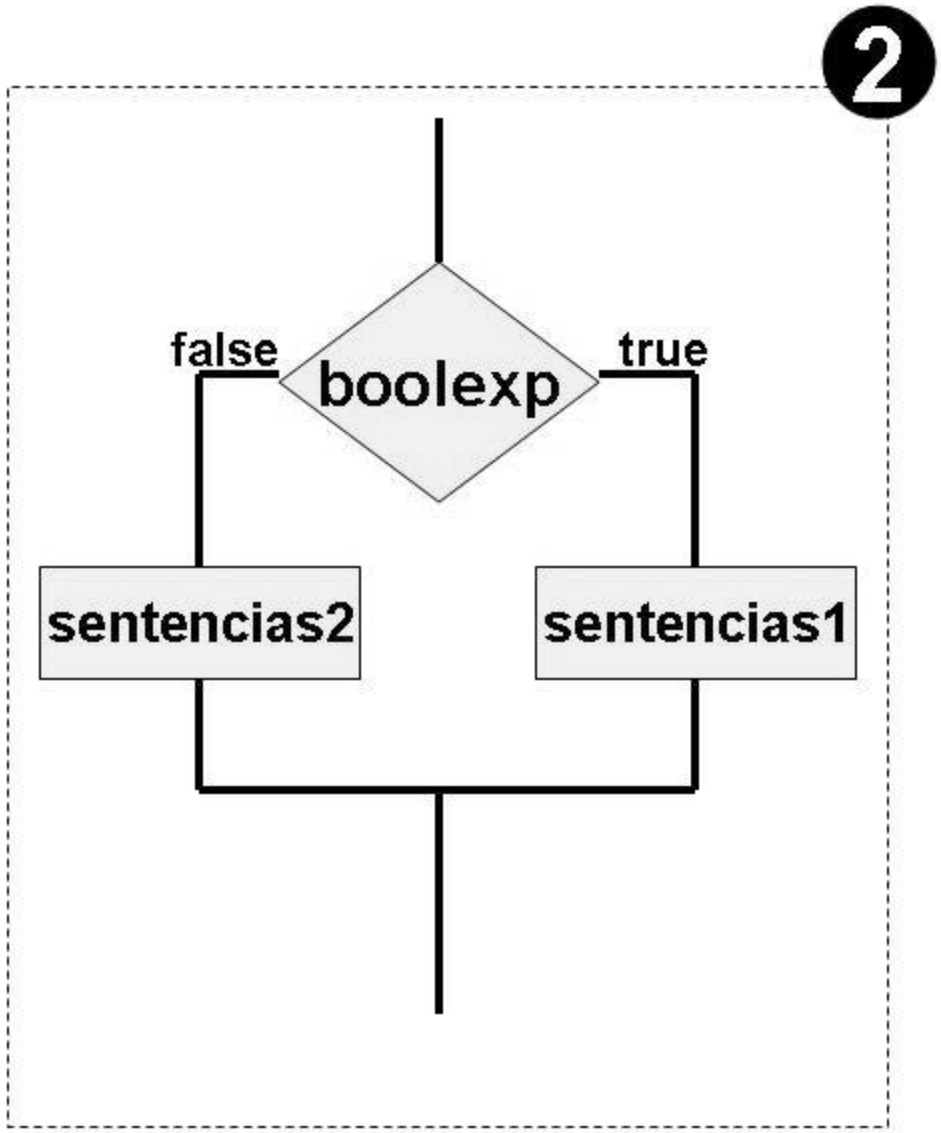
2

```
if (boolexp){
    <sentencias1>
}else{
    <sentencias2>
}
```

3

```
if (boolexp){
    <sentencias1>
}else if (boolexp2){
    <sentencias2>
}else if (boolexp3) {
    <sentencias3>
}else if (boolexp4) {
    <sentencias4>
}else{
    <sentencias5>
}
```


1
false
boolexp
true
sentencias


2
false
boolexp
true
sentencias2
sentencias1

Ejemplo de la sentencia IF

```
int matematicas = 4, lengua = 2;
if (matematicas >= 5){
   if (lengua >= 5){
         System.out.println("Enhorabuena");
   }else{
         System.out.println("No has aprobado todas las
   asignaturas");
   }
}else{
   System.out.println("No has aprobado todas las
   asignaturas");
}
```

3.1.2 **SWITCH**

```
switch (expresión){
        case  valor1:  sentencias1;  break;
        case  valor2:  sentencias2;  break;
        case  valor3:  sentencias3;  break;
        .......
        case  valorn:  sentenciasn;  break;
        [default: sentenciasdef;]
}
```

Recuerda...

Si no se escribe la sentencia **break**, el programa seguirá ejecutando las siguientes sentencias hasta encontrarse con un *break* o el fin del *switch* .

Ejemplo de la sentencia SWITCH

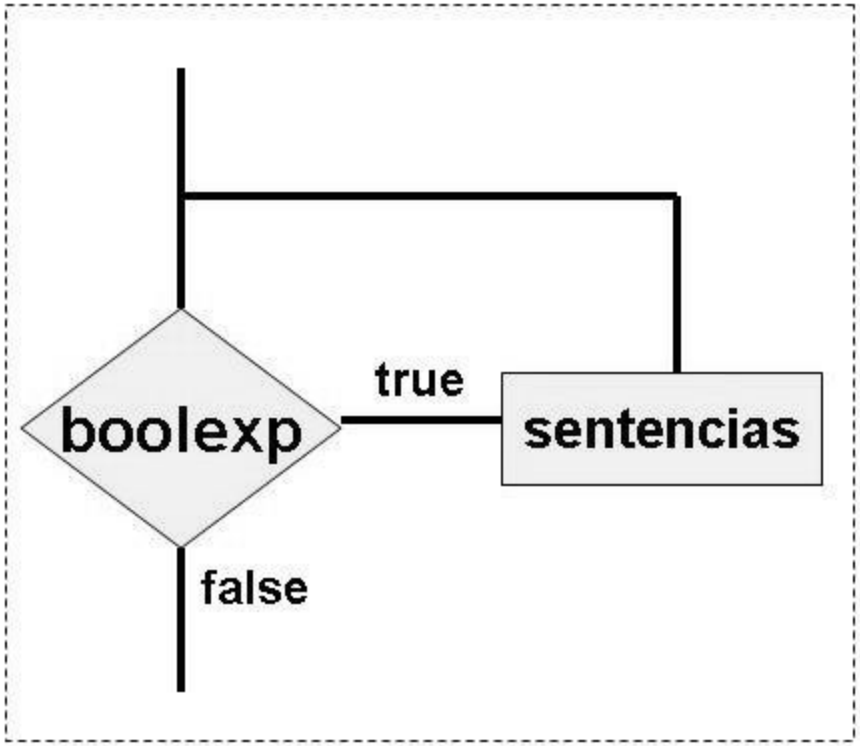
```
switch(posicion){
    case 1: System.out.println("ORO");break;
    case 2: System.out.println("PLATA");break;
    case 3: System.out.println("BRONCE");break;
    case 4: System.out.println("DIPLOMA");break;
    case 5: System.out.println("DIPLOMA");break;
    default: System.out.println("SIN PREMIO");break;
}
```

Ra-Ma®
Juan Carlos Moreno

3.2 ESTRUCTURAS DE REPETICIÓN

3.2.1 **BUCLE WHILE**

```
while (boolexp) {
        sentencias;
}
```

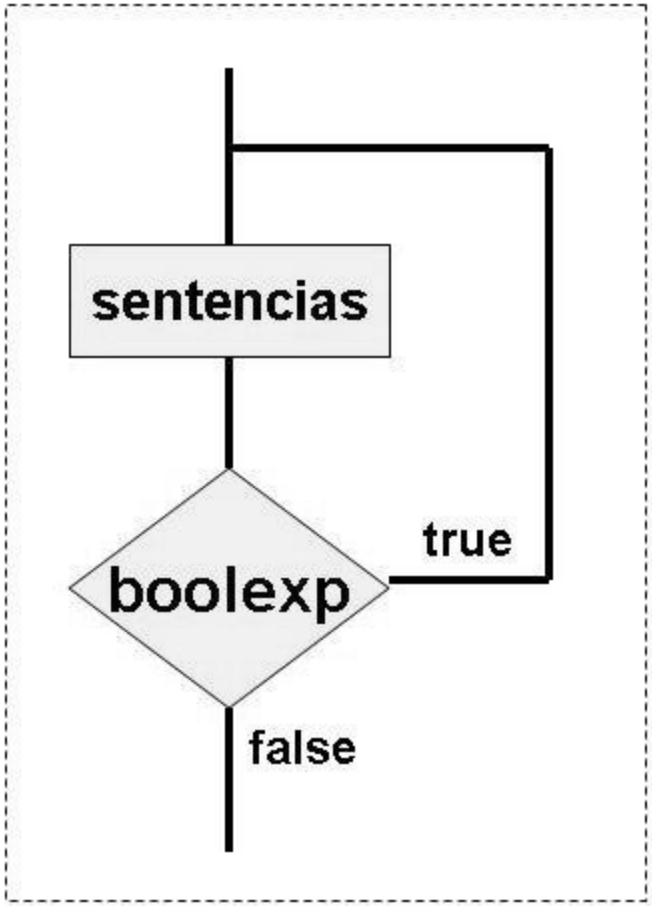


Ejemplo de la sentencia WHILE

```
int numero = 1;
while(numero<=10){ //bucle que cuenta
 hasta 10
    System.out.println(numero);
    numero++;
}
```

3.2.2 BUCLE DO WHILE

```
do{
        sentencias;
} while (expresión);
```

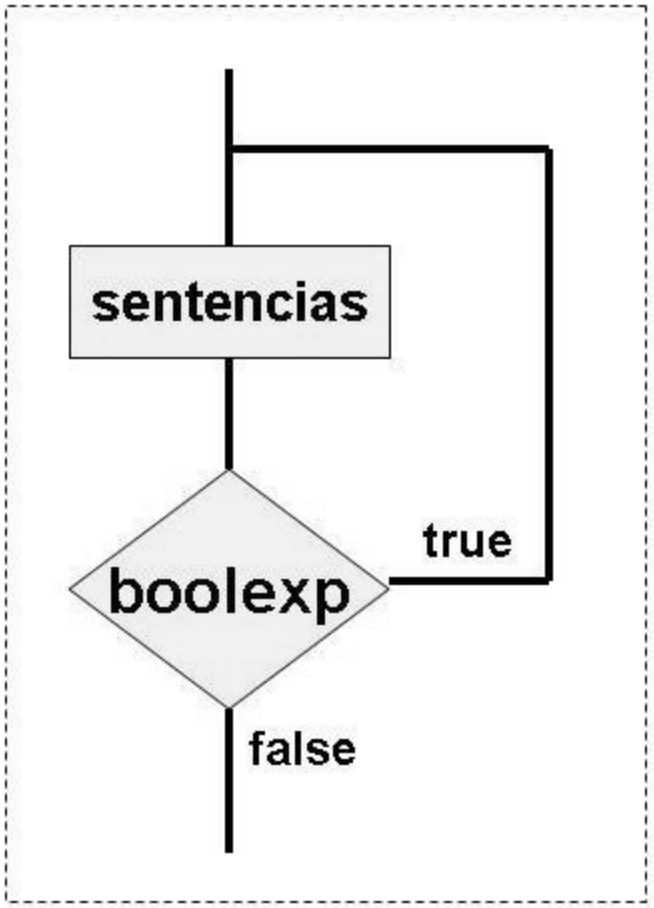


Ejemplo de la sentencia DO WHILE

```
int numero = 1;
do{ //bucle que cuenta hasta 10
   System.out.println(numero);
   numero++;
}while(numero<=10);
```

3.2.3 EL BUCLE FOR

```
do{
        sentencias;
} while (expresión);
```



Ejemplo de la sentencia FOR

```
int numero = 1;
for (numero=1;numero<=10;numero++){
  //bucle que cuenta hasta 10
   System.out.println(numero);
}
```

Ra-Ma®
Juan Carlos Moreno

3.3 ESTRUCTURAS DE SALTO

3.3.1 SENTENCIAS BREAK Y CONTINUE

- La sentencia **break** sirve tanto para las estructuras de selección como para las estructuras de repetición. El programa al encontrar dicha sentencia se saldrá del bloque que está ejecutando.
- La sentencia **continue,** por el contrario, solo se utiliza en las estructuras de repetición (bucles) y lo que hace es terminar la iteración i y continúa por la iteración i+1.

Ejemplo de la sentencia CONTINUE


```
int i=0;
while(i<10){
//este programa muestra los números del 1 al 10
i++;    // sin mostrar el 5
if (i==5){continue;}
System.out.println(i);
}
```

3.3.2 SENTENCIAS BREAK Y CONTINUE CON ETIQUETAS

- Las sentencias **break** y **continue** con etiquetas mantienen el mismo funcionamiento salvo que en esta ocasión el programador puede controlar qué bucle es el que se deja de ejecutar o en qué bucle continua.

Ejemplo de la sentencia BREAK CON ETIQUETAS


```
int i=0;
bucleext :
while(i<100){
   i++;
   for (int j=0;j<i;j++){
        System.out.print("*");
        if (i==5){break bucleext;}
   }
   System.out.println("");
}
```

3.3.3 SENTENCIA RETURN

- La sentencia **return** es otra forma de salir de una estructura de control.
- La diferencia con **break** y **continue** es que **return** sale de la función o método que está ejecutando y permite devolver un valor.

Ra-Ma®
Juan Carlos Moreno

3.4 CONTROL DE EXCEPCIONES

La estructura del control de excepciones

```
try {
    Sentencias a proteger
}catch (excepción_1){
    Control de la excepción 1
}
...
catch (excepción_n){
    Control de la excepción n
}[finally{
    Control opcional
}]
```

Ejemplo del control de excepciones


```
class Test
{
  public static void main(String [] args)   {
  int a=10, b=0, c;

  try{
       c=a/b;
  }catch(ArithmeticException e){
       System.out.println("Error: "+e.getMessage());
       return;
  }
  System.out.println("Resultado:"+c);
  }
}
```

Ra-Ma®
Juan Carlos Moreno

3.5 PRUEBA Y DEPURACIÓN DE APLICACIONES

PRUEBA Y DEPURACIÓN DE APLICACIONES

- Los proyectos de desarrollo de software han sufrido tradicionalmente problemas de calidad .
- Las **pruebas de software** o **testing** son aquel conjunto de procesos que permiten **verificar** y **validar** la calidad de un producto software identificando errores de diseño e implementación .

Plan de Prueba

- Planificación del proceso de pruebas.
- Se trata de ejecutar el software que se está desarrollando o ya está desarrollado bajo condiciones controladas.
- Se pretenden detectar **errores de programación** o **bugs** y lo que se denominan **defectos de forma**.

¿Sabías que?

Es habitual que los **errores** se originen con mayor frecuencia en las primeras fases del desarrollo. Más del 80% de los errores cometidos provienen de las primeras fases del ciclo de vida (análisis de requisitos y diseño funcional y técnico).

Además, el coste de corregir dichos errores crece exponencialmente según avanza el proyecto.

3.5.1 FALLOS DEL SOFTWARE

- Poca o falta de comunicación
- Complejidad del software
- Errores de programación
- Cambios continuos
- Presiones de tiempos
- Pobre documentación del código

¿Sabías que?

Existe un famoso enunciado cabecera de todo el que lleva a cabo pruebas de software: **El testing puede probar la presencia de errores pero no la ausencia de ellos** (*Edsger Dijkstra*).

El tester

Agradecimientos a :
iDream_in_Infrared

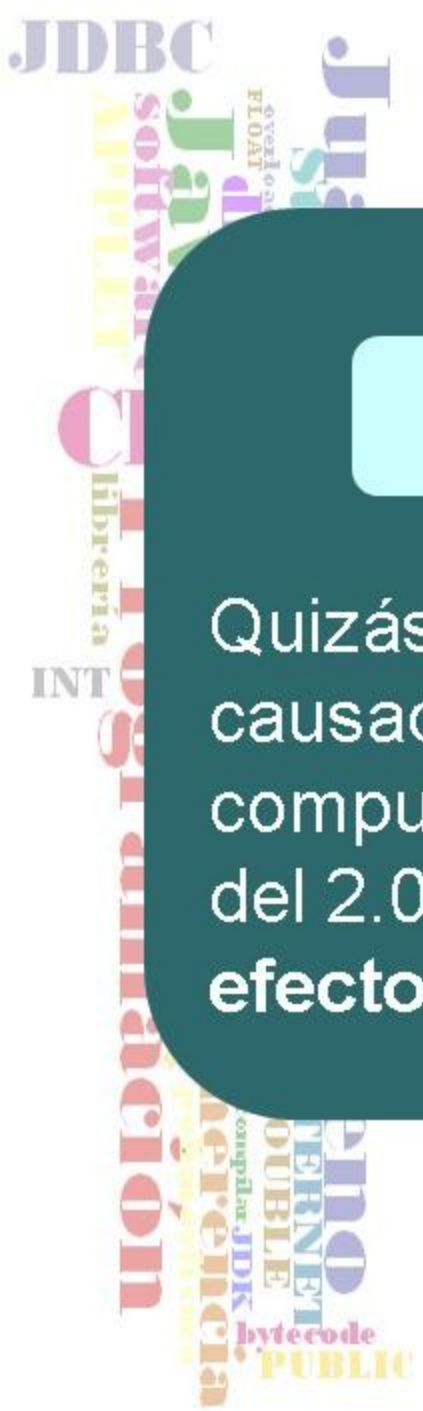
- Persona que realiza las pruebas
- Profesional de altos conocimientos en lenguajes de programación y métodos, técnicas y herramientas especializadas de pruebas.
- Somete al software a una serie de acciones de forma que éste responde con su comportamiento como reacción.
- Actitud de probar para romper
- Habilidad de conseguir el punto de vista del cliente y un buen análisis de detalle para encontrar errores que no se ven a simple vista.

3.5.2 TIPOS DE PRUEBAS

- **Verificación**. Consiste en demostrar que un programa cumple con sus especificaciones. Se centra en la comprobación de las distintas fases del desarrollo antes de pasar a la siguiente.
- **Validación**. Se encarga de comprobar que el programa da la respuesta que espera el usuario. Se centra en la comprobación de los requerimientos del *software*.

Las pruebas

- Práctica popular: Distribuir la **versión beta** del producto (a dicha fase de pruebas, **beta testing)**.
- Versión anterior en el proceso de desarrollo llamada -> **versión alpha** (el programa aun estando incompleto presenta una funcionalidad básica y puede ser ya testeado).
- Antes de salir al mercado -> **RTM testing (***release to Market*), donde se comprueba cada funcionalidad del programa completo en entornos de producción.



Ra-Ma®

¿Sabías que?

Quizás uno de los fallos más históricos causados por un bug en sistemas de computación fue el que se produjo en enero del 2.000 al sufrir las consecuencias del llamado **efecto Y2K (Y2K bug)**.

Ra-Ma®
Juan Carlos Moreno

3.6 DOCUMENTACIÓN DE PROGRAMAS

Documentos mínimos a generar

- **Manual de usuario**. Es el manual que utilizará el usuario para desenvolverse con el programa.
- **Manual técnico**. Es el manual dirigido a los técnicos. Con esta documentación, cualquier técnico que conozca el lenguaje con el que la aplicación ha sido creada debería de poder conocerla casi tan bien como el personal que la creó.
- **Manual de instalación**. En este manual se explican paso a paso los requisitos y cómo se instala y pone en funcionamiento la aplicación.