

Unidad 4: Sentencias de control

Sentencia IF

Es un tipo de sentencia que habilita la selección de varios caminos alternativos en la ejecución del programa.

Para ello se van a utilizar las sentencias de selección. Java provee varias como son:

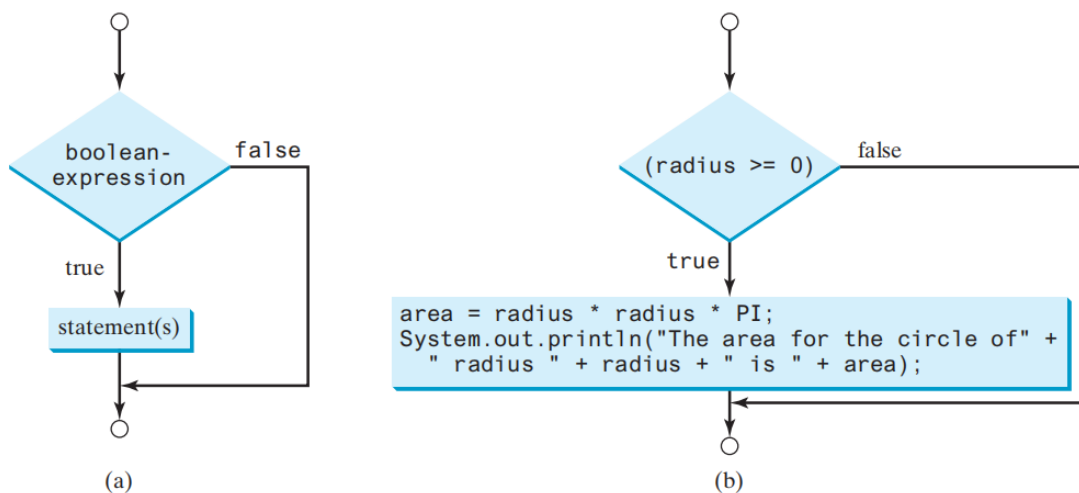
- `if`
- `if-else`
- `switch`

La sentencia `if` de una sólo vía tiene la siguiente sintaxis:

```
if (boolean-expression) {  
    statement1;  
    statement2;  
    statement3;  
    ...  
    statementn;  
}
```

- Sólo se va a ejecutar el `if` siempre y cuando `boolean-expression` sea **true**.

Se puede ver en el siguiente diagrama de flujo:



- En b se puede ver que si el `radius >= 0` pasa a realizar el bloque de código, si no, sigue con el flujo del programa.
- Obsérvese que no se puede ejecutar nunca el bloque de sentencias dentro de un **if** si la condición no es **true**.

Algunas observaciones a `if`:

```
if i > 0 {
    System.out.println("i is positive");
}
```

(a) Wrong

```
if (i > 0) {
    System.out.println("i is positive");
}
```

(b) Correct

Si sólo hay una sentencia:

```
if (i > 0) {
    System.out.println("i is positive");
}
```

(a)

Equivalent

```
if (i > 0)
    System.out.println("i is positive");
```

(b)

Escribe el siguiente código:

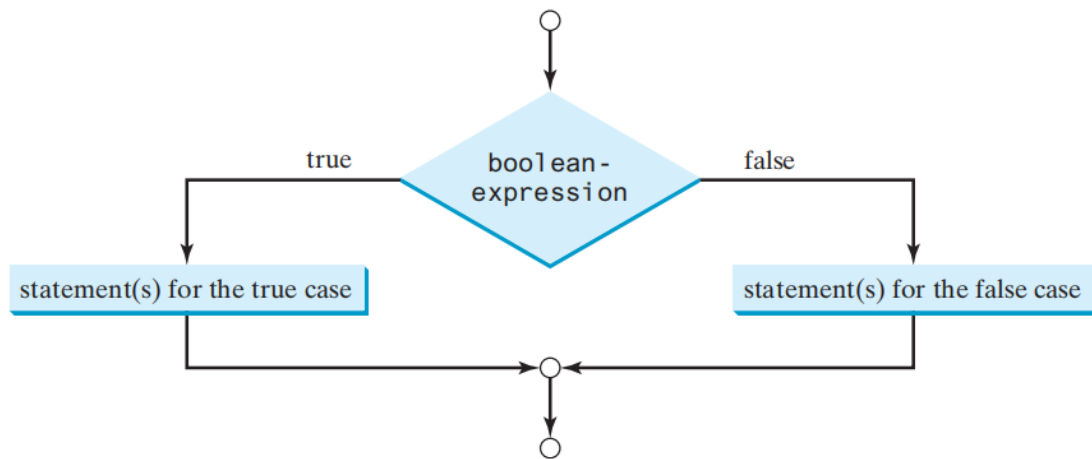
```
// SimpleIfDemo.java
1 import java.util.Scanner;
2
3 public class SimpleIfDemo {
4     public static void main(String[] args) {
5         Scanner input = new Scanner(System.in);
6         System.out.print("Enter an integer: ");
7         int number = input.nextInt();
8
9         if (number % 5 == 0)
10            System.out.println("HiFive");
11
12        if (number % 2 == 0)
13            System.out.println("HiEven");
14    }
15 }
```

Sentencias if-else

La sentencia **if-else** funciona de la siguiente forma:

```
if (boolean-expression) {
    statement(s)-for-the-true-case;
}
else {
    statement(s)-for-the-false-case;
}
```

El diagrama de flujo queda de la siguiente forma:



En este caso si `boolean-expression` es **true** pasa por la parte del `if` mientras que si es **false** pasa por la parte del `else`.

Veamos este ejemplo:

```

if (radius >= 0) {
    area = radius * radius * PI;
    System.out.println("The area for the circle of radius " +
        radius + " is " + area);
}
else {
    System.out.println("Negative input");
}
  
```

Otro ejemplo:

```

if (number % 2 == 0)
    System.out.println(number + " is even.");
else
    System.out.println(number + " is odd.");
  
```

Ejercicios:

- Escribe una sentencia if que aumenta una cantidad en 3% si es mayor que 90 y en caso contrario aumenta un 1%
- Cuál es la salida si `number` es 30 y si es 35.

```

if (number % 2 == 0)
    System.out.println(number + " is even.");
System.out.println(number + " is odd.");
  
```

(a)

```

if (number % 2 == 0)
    System.out.println(number + " is even.");
else
    System.out.println(number + " is odd.");
  
```

(b)

Sentencias if-else anidadas y alternativa múltiple

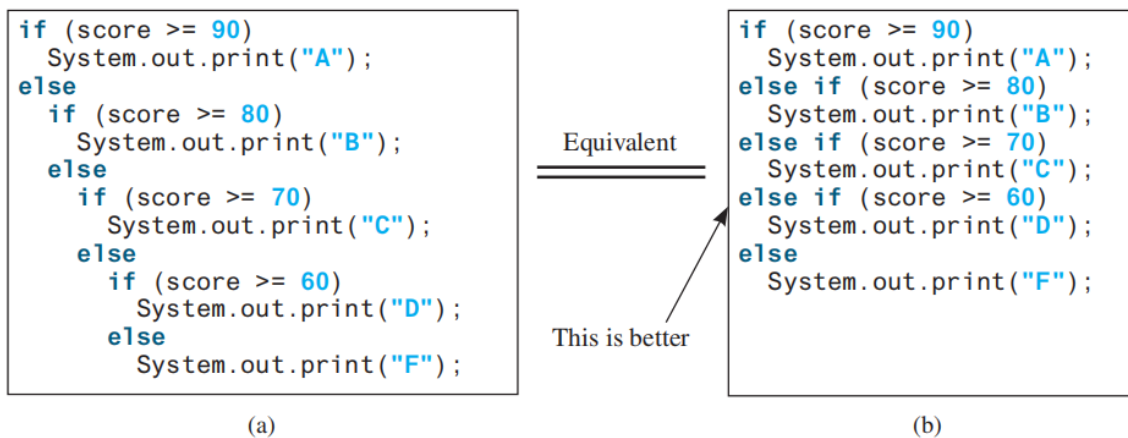
Una sentencia `if-else` puede estar dentro de otra y así sucesivamente, esto se llama anidación o que una sentencia está anidada

Veamos el ejemplo:

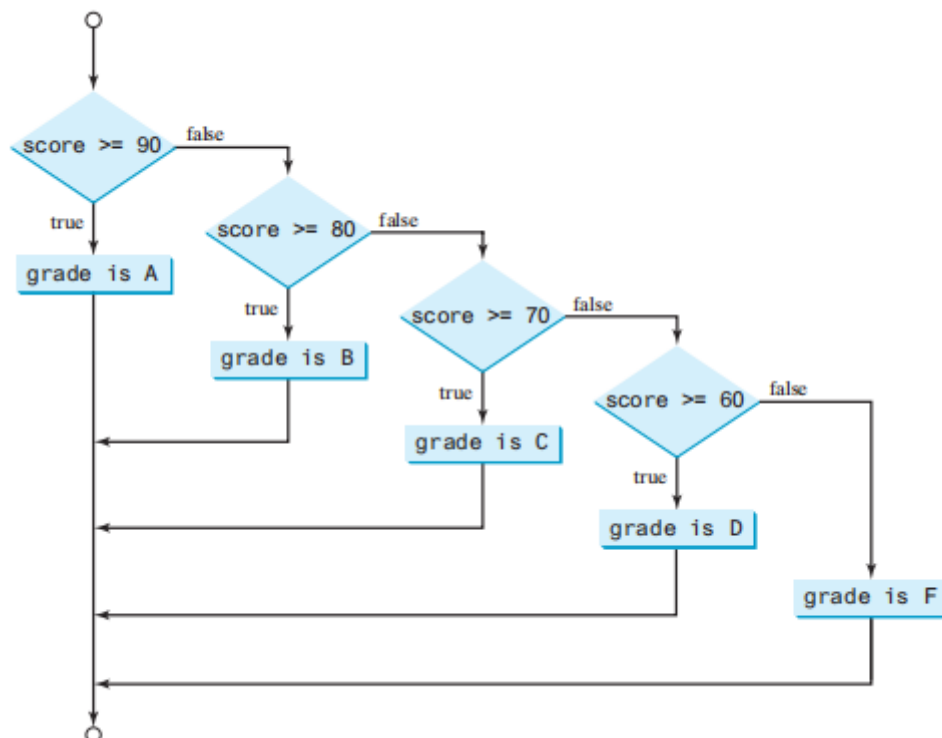
```
if (i > k) {  
    if (j > k)  
        System.out.println("i and j are greater than k");  
}  
else  
    System.out.println("i is less than or equal to k");
```

- La sentencia del primer if sólo si $i > k$ pasará a la segunda sentencia y evaluará $j > k$, si es cierto, imprimirá el mensaje de que i y j es mayor que k .
- Si $i > k$ es false no entrará en el if anidado y pasará a imprimir que **i es menor o igual a k** puesto que sólo sabemos que NO es mayor.
- Tampoco sabemos la relación entre i y j .

Para ello podemos utilizar la **alternativa múltiple**:



La forma b es mejor por ser más legible. Un diagrama que muestre la salida del programa anterior es la siguiente:



Operadores Lógicos

Los operadores lógicos o booleanos son:

- `||`: OR
- `&&`: AND
- `!`: NOT
- `XOR`: OR Exclusiva

TABLE 3.3 Boolean Operators

<i>Operator</i>	<i>Name</i>	<i>Description</i>
!	not	Logical negation
&&	and	Logical conjunction
 	or	Logical disjunction
^	exclusive or	Logical exclusion

TABLE 3.4 Truth Table for Operator `!`

p	!p	<i>Example (assume age = 24, weight = 140)</i>
true	false	<code>!(age > 18)</code> is false, because <code>(age > 18)</code> is true.
false	true	<code>!(weight == 150)</code> is true, because <code>(weight == 150)</code> is false.

TABLE 3.5 Truth Table for Operator `&&`

p₁	p₂	p₁ && p₂	<i>Example (assume age = 24, weight = 140)</i>
false	false	false	
false	true	false	<code>(age > 28) && (weight <= 140)</code> is false, because <code>(age > 28)</code> is false.
true	false	false	
true	true	true	<code>(age > 18) && (weight >= 140)</code> is true, because <code>(age > 18)</code> and <code>(weight >= 140)</code> are both true.

TABLE 3.6 Truth Table for Operator `||`

p₁	p₂	p₁ p₂	<i>Example (assume age = 24, weight = 140)</i>
false	false	false	<code>(age > 34) (weight >= 150)</code> is false, because <code>(age > 34)</code> and <code>(weight >= 150)</code> are both false.
false	true	true	
true	false	true	<code>(age > 18) (weight < 140)</code> is true, because <code>(age > 18)</code> is true.
true	true	true	

TABLE 3.7 Truth Table for Operator ^

p ₁	p ₂	p ₁ ^ p ₂	Example (assume age = 24, weight = 140)
false	false	false	(age > 34) ^ (weight > 140) is false, because (age > 34) and (weight > 140) are both false.
false	true	true	(age > 34) ^ (weight >= 140) is true, because (age > 34) is false but (weight >= 140) is true.
true	false	true	
true	true	false	