

The logo for Oracle Academy is centered on a light gray background. It features the word "ORACLE" in a bold, orange, sans-serif font. Below it, the word "Academy" is written in a smaller, dark gray, sans-serif font. The entire logo is framed by two horizontal dark gray bars, one at the top and one at the bottom.

# ORACLE

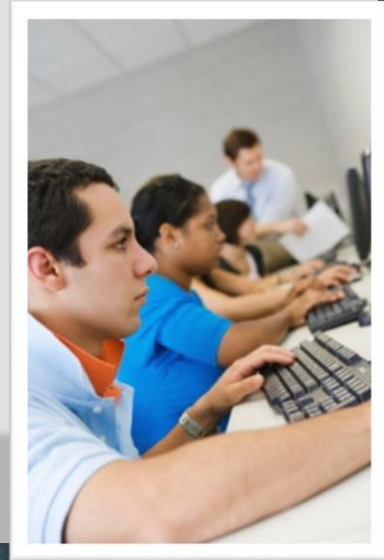
## Academy

# Java Foundations

5-2

**Descripción de la ejecución condicional**

**ORACLE**  
Academy



Copyright © 2022, Oracle y/o sus filiales. Oracle, Java y MySQL son marcas comerciales registradas de Oracle y/o sus filiales. Todos los demás nombres pueden ser marcas comerciales de sus respectivos propietarios.

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Describir la ejecución condicional
  - Describir los operadores lógicos
  - Comprender la evaluación de "cortocircuito" de operadores lógicos
  - Crear construcciones if encadenadas



## Si se aplican varias condiciones

- ¿Qué ocurre si una acción concreta solo se va a llevar a cabo si varias condiciones son true?
- Imagine un caso en el que un estudiante opta a recibir una beca si se cumplen las dos condiciones siguientes:
  - La calificación debe ser  $\geq 88$
  - El número de días ausente = 0

# Manejo de varias condiciones

- Los operadores relacionales son correctos al comprobar una sola condición
- Puede utilizar una secuencia de sentencias if para comprobar más de una condición

```
if (grade >= 88) {  
    if (numberDaysAbsent == 0) {  
        System.out.println("You qualify for the scholarship.");  
    }//endif  
}//endif
```

# Manejo de varias condiciones: Ejemplo

- Como se muestra en el ejemplo:
  - La secuencia de sentencias if es difícil escribir, más difícil de leer y aún más difícil conforme agregue más condiciones
  - Afortunadamente, Java tiene una forma sencilla de manejar varias condiciones: operadores lógicos

# Operadores lógicos de Java

- Puede utilizar tres operadores lógicos de Java para combinar varias expresiones booleanas en una expresión booleana

| Operador lógico | Significado |
|-----------------|-------------|
| &&              | AND         |
|                 | OR          |
| !               | NOT         |

# Tres operadores lógicos

| Operación                             | Operador          | Ejemplo   |
|---------------------------------------|-------------------|---|
| Si una condición AND otra condición   | <b>&amp;&amp;</b> | <code>int i = 2;<br/>int j = 8;<br/>((i &lt; 1) &amp;&amp; (j &gt; 6))</code> |
| Si una condición OR ambas condiciones | <b>  </b>         | <code>int i = 2;<br/>int j = 8;<br/>((i &lt; 1)    (j &gt; 10))</code>        |
| NOT                                   | <b>!</b>          | <code>int i = 2;<br/>(!(i &lt; 3))</code>                                     |

En la tabla de la diapositiva se muestran los operadores lógicos en el lenguaje de programación Java. Todos los ejemplos producen un resultado booleano false.



## Aplicación de operadores lógicos

- Puede escribir el ejemplo anterior mediante el operador lógico AND cuando:

```
grade >= 88 && numberDaysAbsent == 0
```

Expresión  
booleana 1

Operador  
lógico

Expresión  
booleana 2

- El operador lógico permita comprobar varias condiciones más fácilmente y el código sea más legible

En este ejemplo, se utiliza el operador lógico AND porque ambas expresiones booleanas deben ser true para que el estudiante pueda optar a la beca.

Operador lógico AND:

- La condición combinada es true únicamente si ambas expresiones booleanas son true.
- La condición combinada es false si una o ambas expresiones booleanas son false.

# Operador lógico AND: Ejemplo

```
public static void main(String[] args) {  
    int numberDaysAbsent = 0;  
    int grade = 95;  
    if (grade >= 88 && numberDaysAbsent == 0) {  
        System.out.println("You qualify for the scholarship.");  
    }  
    else {  
        System.out.println("You do not qualify for the  
                           scholarship.");  
    }  
} //endif  
} //end method main
```

Se evalúa como true si  
ambas expresiones  
booleanas son true

Este ejemplo ilustra el operador lógico AND. Para que la salida muestre "You qualify for the scholarship", las dos condiciones deben ser true. Es decir, la calificación debe ser mayor o igual que 88 y el número de días ausente debe ser igual a cero.

# Operador lógico OR

- Imagine un caso en el que un estudiante opta a formar parte de un equipo de deporte si se cumple una de las dos condiciones siguientes:
  - Calificación  $\geq 70$
  - Número de días ausente  $< 5$
- En este caso, puede utilizar el operador lógico OR para conectar varias expresiones booleanas

```
grade >=70 || numberDaysAbsent < 5
```

|                         |                    |                                |
|-------------------------|--------------------|--------------------------------|
| <u>grade &gt;=70</u>    | <u>  </u>          | <u>numberDaysAbsent &lt; 5</u> |
| Expresión<br>booleana 1 | Operador<br>lógico | Expresión<br>booleana 2        |

# Operador lógico OR: Ejemplo

```
public static void main(String[] args) {  
    int numberDaysAbsent = 3;  
    int grade = 85;  
    if (grade >= 70 || numberDaysAbsent < 5) {  
        System.out.println("You qualify for a sports team");  
    }  
    else {  
        System.out.println("You do not qualify for a sports  
team");  
    }  
}  
}
```

Se evalúa como true si una de las expresiones booleanas se evalúa como true

Este ejemplo ilustra el uso del operador lógico OR. En este ejemplo, "You qualify for a sports team" se muestra incluso si una de las condiciones es verdadera. Es decir, el nivel debe ser  $\geq 70$  o el número de días ausente debe ser inferior a cinco.

# Operador lógico NOT

- Imagine un caso en el que un estudiante opta a tutoría gratuita si se cumplen las dos condiciones siguientes:
  - Calificación < 88
  - Número de días ausente < 3
- Utilice el operador lógico !

```
!madeFreeTutor && numberDaysAbsent < 3
```

Operador  
lógico

Expresión  
booleana 1

Expresión  
booleana 2

Este ejemplo ilustra el operador lógico !. Como la calificación es igual a 65, !madeFreeTutor es true porque madeFreeTutor es false.

La expresión combinada se evalúa en true y muestra la siguiente salida: "You qualify for free tutoring help."

# Operador lógico NOT

```
public static void main(String args[]) {  
    int numberDaysAbsent = 2;  
    int grade = 65;  
    boolean madeFreeTutor = grade >= 88;  
    if (!madeFreeTutor && numberDaysAbsent < 3) {  
        System.out.println("You qualify for free tutoring  
help");  
    }  
}  
}
```

Este ejemplo ilustra el operador lógico !. Como la calificación es igual a 65, !madeFreeTutor es true porque madeFreeTutor es false.

La expresión combinada se evalúa en true y muestra la siguiente salida: "You qualify for free tutoring help."

## Ejercicio 1

- Cree un nuevo proyecto y agréguele el archivo `WatchMovie.java`
- Modifique `WatchMovie.java` para ver una película que cumpla con las dos condiciones siguientes:
- El precio de la película es mayor o igual que 12 \$
- La clasificación de la película es igual a 5
  - Muestre la salida como **"Me interesa ver la película"**
  - De lo contrario, muestre la salida como **"No me interesa ver la película"**

## Omisión de la segunda prueba AND

- Los operadores `&&` y `||` son operadores cortocircuito
- Si la primera expresión (del lado izquierdo) es false, no es necesario evaluar la segunda expresión (del lado derecho)

```
b = (x != 0) && ((y / x) > 2);
```

Expresión de la izquierda      Expresión de la derecha

La evaluación se realiza de izquierda a derecha y se detiene tan pronto como se conozca el resultado. Esto significa que la expresión de la parte derecha no se evaluará si no es necesario.



## Omisión de la segunda prueba AND

```
b = (x != 0) && ((y / x) > 2);
```

Expresión  
de la  
izquierda

Expresión de la  
derecha

- Si x es 0, (x != 0) es false
- Para el operador &&, como no importa si ((y/x) > 2) es true o false, el resultado de esta expresión es false
- Por lo tanto, Java no continúa evaluando ((y/x) > 2)

La evaluación se realiza de izquierda a derecha y se detiene tan pronto como se conozca el resultado. Esto significa que la expresión de la parte derecha no se evaluará si no es necesario.

## Omisión de la segunda prueba OR

- Si la primera expresión (del lado izquierdo) es true, no es necesario evaluar la segunda expresión (del lado derecho)
- Considere este ejemplo:

```
boolean b = (x <= 10) || (x > 20);
```

Expresión  
de la  
izquierda

Expresión de la  
derecha

- Si  $(x \leq 10)$  es true,  $(x > 20)$  no se evaluará porque no importa si  $(x > 20)$  es true o false
- El resultado de esta expresión es true

# ¿Qué es un operador condicional ternario?

| Operación   | Operador | Ejemplo   |
|---|----------|---|
| Si la condición es true, asigne<br>result = value1<br>De lo contrario, asigne result =<br>value2<br>Nota: value1 y value2 deben<br>ser del mismo tipo de dato | ?:       | result=condition ? value1 : value2<br><br>Ejemplo:<br><code>int x = 2, y = 5, z = 0;</code><br><br><code>z = (y &lt; x) ? x : y;</code> |

## Sentencias equivalentes

```
z = (y < x) ? x : y;
```

```
if(y<x){  
    z=x;  
}  
else{  
    z=y;  
} //endif
```

**ORACLE**  
Academy

JFo 5-2  
Descripción de la ejecución condicional

Copyright © 2022, Oracle y/o sus filiales. Oracle, Java y MySQL son marcas comerciales registradas de Oracle y/o sus filiales. Todos los demás nombres pueden ser marcas comerciales de sus respectivos propietarios.

19

El operador ternario es un operador condicional que requiere tres operandos. Tiene una sintaxis más compacta que una sentencia `if/else`.

Utilice el operador ternario en lugar de una sentencia `if/else` si desea hacer el código más corto.

Hay tres operandos en el ejemplo de la diapositiva:

`(y < x)` : Esta expresión booleana (condición) que se está evaluando.

`? x` : Si `(y < x)` es true, a `z` se le asignará el valor de `x`.

`: y` : Si `(y < x)` es false, a `z` se le asignará el valor de `y`.

En el ejemplo de la diapositiva, `z = 5`.

## Operador condicional ternario: Escenario

- Imagine que está jugando un partido de fútbol y está realizando el seguimiento de los goles como se indica a continuación:

```
public static void main(String args[]) {  
    int numberOfGoals = 5;  
    String s;  
    if (numberOfGoals == 1) {  
        s = "goal";  
    }  
    else {  
        s = "goals";  
    } //endif  
    System.out.println("I scored " + numberOfGoals + " " + s);  
} //end method main
```

En función del número de goles, estos ejemplos imprimirán la forma adecuada en singular o plural de "gol". La operación es compacta, ya que solo produce dos resultados basados en una expresión booleana.

## Operador condicional ternario: Ejemplo

- Un resultado similar se logra con el operador ternario al reemplazar toda la sentencia if/else por una sola línea

```
int numberOfGoals = 1;  
  
System.out.println("I scored " + numberOfGoals + " "  
                  + (numberOfGoals == 1 ? "goal" : "goals")  
                  );
```

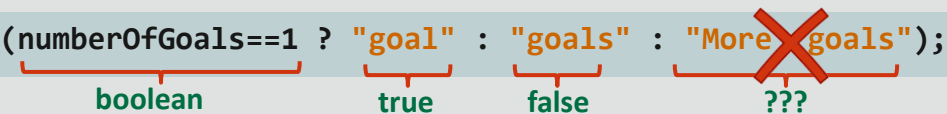
## Operador condicional ternario: Ejemplo

- Ventaja: Puede colocar la operación directamente en una expresión

```
int numberOfGoals = 1;  
  
System.out.println("I scored " + numberOfGoals + " "  
    + (numberOfGoals == 1 ? "goal" : "goals")  
);
```

- Desventaja: Solo puede tener dos posibles resultados

```
(numberOfGoals==1 ? "goal" : "goals" : "MoreXgoals");
```



boolean      true      false      ???

Como puede ver, el operador ternario puede ser útil para reducir el número de líneas de código, pero puede hacer que el código sea difícil de leer, por lo que no es lo mejor para sentencias anidadas.

## Ejercicio 2

- Agregue el archivo `TernaryOperator.java` al proyecto creado para el ejercicio 1
- Modifique `TernaryOperator.java` para duplicar la lógica proporcionada en la sentencia `if/else` mediante el operador ternario

# Manejo de condiciones complejas con un constructor if encadenado

- La sentencia if encadenada:
  - Conecta varias condiciones en un único constructor
  - Tiende a resultar confusa de leer y difícil de mantener



# Encadenamiento de construcciones if/else

- Puede encadenar construcciones if y else juntas para indicar distintos resultados para varias expresiones diferentes
- Sintaxis:

```
if (<condition1>) {  
    //code_block1  
}  
else if (<condition2>) {  
    // code_block2  
}  
else {  
    // default_code  
}
```

En el ejemplo de la diapositiva se muestra la sintaxis de una construcción if/else encadenada:

Cada una de las condiciones es una expresión booleana.

code\_block1 representa las líneas de código que se ejecutan si la condition1 es true.

code\_block2 representa las líneas de código que se ejecutan si la condition1 es false y la condition2 es true.

default\_code representa las líneas de código que se ejecutan si ambas condiciones se evalúan como false.

**Nota:** Varias sentencias else if se pueden evaluar. La sentencia else es opcional.

## Encadenamiento de construcciones if/else: Ejemplo

```
public static void main(String args[]) {  
    double income = 30000, tax;  
  
    if (income <= 15000) {  
        tax = 0;  
    }  
    else if (income <= 25000) {  
        tax = 0.05 * (income - 15000);  
    }  
    else {  
        tax = 0.05 * (income - (25000 - 15000));  
        tax += 0.10 * (income - 25000);  
    }  
} //endif  
} //end method main
```

Este ejemplo muestra el encadenamiento de construcciones `if/else` para probar varias condiciones. La sentencia `else` se ejecuta si todas las condiciones son false.

## ¿Se pueden anidar las sentencias if?

- En Java, una sentencia if puede estar presente en el cuerpo de otra sentencia if

```
if (tvType == "color") {  
    if (size == 14) {  
        discPercent = 8;  
    }  
    else {  
        discPercent = 10;  
    } //endif  
} //endif
```

- En este ejemplo, la sentencia else se empareja con la sentencia if (size==14)

En una sentencia `if` anidada:

Es muy importante asegurarse de que la construcción `else` va con cada construcción `if`. Esta sangría ayuda en gran medida a la claridad del código para un lector.

En este ejemplo, si la sentencia `if` exterior es `true`, a continuación, la sentencia `if` interior se ejecuta.

## Descripción de las sentencias if anidadas

- En este ejemplo, la sentencia else se empareja con la sentencia if externa (TVType=="color")

```
if (tvType == "color") {  
    if (size == 14) {  
        discPercent = 8;  
    }//endif  
}  
else {  
    discPercent = 10;  
}//endif
```

## Ejercicio 3

- Agregue el archivo `ComputeFare.java` al proyecto creado para el ejercicio 1
- Examine `ComputeFare.java`
- Implante lo siguiente con las construcciones `if/else`:
  - Declare un variable de entero, `age`
  - Pida al usuario que introduzca el valor para `age`
- Con una construcción `if` encadenada, calcule la tarifa en función del valor de `age` según estas condiciones:
  - Si `age` es inferior a 11, la tarifa=3 \$
  - Si `age` es superior a 11 e inferior a 65, la tarifa=5 \$
  - Else para todos los demás valores de `age`, la tarifa= 3 \$

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Describir la ejecución condicional
  - Describir los operadores lógicos
  - Comprender la evaluación de "cortocircuito" de operadores lógicos
  - Crear construcciones if encadenadas



The logo for Oracle Academy is centered on a light gray background. It features the word "ORACLE" in a bold, orange, sans-serif font. Below it, the word "Academy" is written in a smaller, dark gray, sans-serif font. The entire logo is framed by two horizontal dark gray bars, one at the top and one at the bottom.

# ORACLE

## Academy