

CAPÍTULO 4

/ /

» Operadores

Operadores de Java

Los operadores de Java producen nuevos valores a partir de uno o más operandos (para que quede claro, recuerde que los operandos son las cosas del lado derecho o izquierdo del operador).

Los resultados de la mayoría de las operaciones es un valor booleano o numérico. Esto es debido a que Java no es C++ y en Java los operadores son típicamente sobrecargados. Sin embargo, hay algunos operadores excepcionales que están sobrecargados.

• El operador + se puede utilizar para somar dos primitivas numéricas juntas, o para realizar una operación de concatenación si alguno de los operandos es una cadena.

• Los operadores &, | y ^ se pueden utilizar de dos formas diferentes, aunque en esta versión del examen, no se probarán sus capacidades de alteración de bits.

Los operadores y las asignaciones son parte de muchas preguntas en otros temas.

Operadores de Asignación

Como su nombre lo indica, se encargan de asignarle a una variable el resultado de una expresión matemática o el valor de otra variable.

(1/1)

Para resumir:

- Al asignar un valor a una primitiva, el tamaño sí importa. Asegúrate de saber cuándo se producirá la conversión implícita, cuando sea necesaria la conversión explícita y cuándo se produzca el truncamiento.
- Recuerde que una variable de referencia no es un objeto; es una forma de llegar a un objeto.
- Al asignar un valor a una variable de referencia, el tipo es importante.

Operadores de Asignación Compuesta

En realidad, hay 11 o más operadores de asignación compuesta, pero solo hay cuatro más comúnmente utilizados ($+=$, $-=$, $*=$ y $/=$). Los operadores de asignaciones compuestas permiten que los mecanógrafos perezosos afeiten algunas pulsaciones de teclas de su carga de trabajo.

Aquí hay varios ejemplos de asignaciones, primero sin usar un operador compuesto:

$$y = y - 6;$$

$$x = x + 2 * 5;$$

Ahora con operadores compuestos

$$y -= 6;$$

$$x += 2 * 5;$$

Estas dos últimas asignaciones dan el mismo resultado que las dos primeras.



Operadores Relacionales

Los operadores relacionales siempre dan como resultado un valor booleano (verdadero o falso). Este valor booleano es utilizado con mayor frecuencia en una prueba `if`, como a continuación:

```
int x=8;
if (x < 9) {
    // do something
}
```

Pero el valor resultante también se puede asignar directamente a una primitiva booleana:

```
class CompareTest {
    public static void main (String [] args) {
        boolean b = 100 > 99;
        System.out.println ("The value of b is " + b);
    }
}
```

Java tiene cuatro operadores relacionales que se pueden utilizar para comparar cualquier combinación de enteros, números de coma flotante o caracteres:

- `>` mayor que
- `>=` mayor o igual que
- `<` menor que
- `<=` menor o igual que

Los caracteres se pueden usar en operadores de comparación. Cuando comparando un carácter con un carácter, o un carácter con un número,

Java usaba el valor Unicode del carácter como valor numérico, para la comparación.

Operadores de Igualdad

Java también tiene dos operadores relacionales (a veces llamados "operadores de igualdad") que compara dos "cosas" similares y devolver un booleano representando lo que es cierto acerca de siendo las dos "cosas" iguales. Estos operadores son:

- `==` es igual a
- `!=` No es igual a

Cada comparación individual puede incluir dos números (incluido el carácter), dos valores booleanos o dos variables de referencia de objeto.

Hay cuatro tipos diferentes de cosas que se pueden probar:

- números
- personajes
- booleanas primitivas
- Variables de referencia de objeto

Igualdad para Primitivas

Si un número de punto flotante se compara con un entero y los valores son los mismos, el operador `=` devuelve verdadero como se esperaba.

Igualdad para las variables de referencia

Dos variables de referencia pueden referirse al mismo objeto, como se muestra a continuación:

```
JButton a = new JButton("Exit");  
JButton b = a;
```

Las variables de referencia se pueden probar para ver si hacen referencia al mismo objeto utilizando el operador `==`. El operador `==` es mirando los bits en la variable, por lo que para las variables de referencia esto significa que si los bits en ambas variables de referencia son idénticos, ambas se refieren al mismo objeto.

Igualdad para enumeraciones

Una vez que haya declarado una enumeración, no se puede expandir. En tiempo de ejecución, no hay tiempo y forma de hacer constantes de enumeración adicionales. Se pueden tener tantas variables como se desee, así como es al referirse a una constante de enumeración dada, es por esto que es importante poder comparar dos enumeraciones de variables de referencia para ver si son iguales, es decir, que si pertenecen a la misma enumeración constante. Se puede utilizar el operador `==` o el método `equals()` para determinar si dos variables se refieren a la misma constante de enumeración.

Instancia de Comprobación (instance of)

El operador `instanceof` se usa solo para variables de referencia de objeto, y puede ser utilizado para comprobar si un objeto es de un tipo en particular. Por tipo, nos referimos a clase o tipo de interfaz, en otras palabras, si el objeto al que hace referencia la variable de la izquierda del lado del operador pasa la prueba IS-A para la clase o tipo de interfaz a la derecha.

Instancia de error del compilador (instanceof Compile Error)

No se puede ocupar el operador `instanceof` para probar en dos jerarquías de clases diferentes.

Por ejemplo, lo siguiente no se compilará:

```
class Cat {}  
class Dog {  
    public static void main (String [] args) {  
        Dog d = new Dog ();  
        System.out.println (d instanceof Cat);  
    }  
}
```

la compilación falla ya que es imposible que `d` pueda referirse a un gato o un subtipo de gato.

En la siguiente tabla se resume el uso del operador `instanceof` dado lo siguiente:

```

interface Face {
}
class Bar implements Face {}
class Foo extends Bar {}

```

First OPERAND
 (Reference Being Tested) instanceof Operand
 (Type We're Comparing
 the Reference Against)

RESULT

Null

Cualquier clase o tipo de
 interfaz.

False

Foo instance

Foo, Bar, Face, Object

True

Bar instance

Bar, Face, Object

True

Bar instance

Foo

False

Foo []

Foo, Bar, Face

False

Foo []

Object

True

Foo [1]

Foo, Bar, Face, Object

True

Operadores Aritméticos

Operadores Aritméticos Basicos:

+ suma

* multiplicación

- resta

/ división

El Operador Restante (%)

El Operador Restante divide el operando
 (izquierdo) por el operando del lado derecho,
 y el resultado es el resto, como se muestra a
 continuación:

```

class MathTest {
    public static void main(String [] args) {
        int x=15;
    }
}

```

```

int y = x % 4;
System.out.println("The result of 15%4 is the "
+ "remainder of 15 divided by 4. The remainder
is" + y);
}
    
```

Compilándolo daria como resultado:

The result of 15%4 is the remainder of 15 divided by
4. The remainder is 3.

Las expresiones se evalúan de izquierda a derecha de forma predeterminada. Puede cambiar esta secuencia, o precedencia, agregando parentesis. También recuerde que el *, /, y los operadores %, tienen mayor precedencia que los operadores + y -.)

Operador de Concatenación de Cadenas

El signo + también se puede usar para concatenar dos cadenas juntas, como se ve a continuación:

String animal = "Grey" + "elephant";

la regla que debemos recordar es:

Si alguno de los operandos es una cadena, el operador + se convierte en una concatenación de cadenas de operador. Si ambos operandos son números, el operador + es el operador de suma.

Operadores de incremento y disminución

Java tiene dos operadores que incrementan o reducen una variable exactamente en uno.

Estos operadores se componen de dos signos más (++) ó dos signos menos (--)

++ incremento, -- decremento (prefijos y sufijos)

El operador se coloca antes (prefijo) o después (sufijo) de una variable para cambiar es válido. Si el operador viene antes o después del operando puede cambiar el resultado de una expresión.

Operador Condicional

El operador condicional es un operador ternario ya que, tiene tres operandos y se usa para evaluar expresiones booleanas, muy parecido a una sentencia if, excepto que en lugar de ejecutar un bloque de código si la prueba es verdadera, un operador condicional asignará un valor a una variable.

En otras palabras, el objetivo del operador condicional es decidir cuál de los dos valores asignar a una variable. Este operando se construye usando un signo de interrogación (?) y dos puntos (:). Los paréntesis son opcionales. Su estructura es:

```
x = (boolean expression)? value to assign : value to assign
```

Un operador condicional comienza con una operación booleana, seguida de dos posibles valores a la izquierda del valor del operador de asignación (=).

El primer valor (el que a la izquierda de los dos puntos) se asigna si la prueba condicional (booleana) es verdadera, y el segundo valor se asigna si la prueba condicional es falsa.

OPERADORES LÓGICOS

Se especifican seis operadores lógicos los cuales son `&`, `|`, `^`, `!`, `&&` y `||`.

Operadores Bit a Bit

De los seis operadores lógicos enumerados anteriormente, tres de ellos (`&`, `|` y `^`) también se pueden utilizar como operadores "bit a bit".

Los operadores bit a bit comparan dos variables bit a bit y devuelven una variable cuyos bits se han establecido en función de si las dos variables que se comparan tenían bits respectivos que estaban "en" (`&`), uno u otro "en" (`|`), o exactamente uno "encendido" (`^`).

Operadores Lógicos de Cortocircuito

- `&&` cortocircuito AND
- `||` cortocircuito OR

Se utilizan para vincular pequeñas expresiones booleanas para formar booleanas más grandes.