# Chapter 2; Operators

Friday, November 8, 2024 11:57 AM

1. Which of the following Java operators can be used with `boolean` variables? (Choose all that apply.)
   - A. `==`
   - B. `+`
   - C. `--`
   - D. `!`
   - E. `%`
   - F. `~`
   - G. Cast with `(boolean)`

//

2. What data type (or types) will allow the following code snippet to compile? (Choose all that apply.)

```
byte apples = 5;
short oranges = 10;
_____ bananas = apples + oranges;
```

   - A. `int`
   - B. `long`
   - C. `boolean`
   - D. `double`
   - E. `short`
   - F. `byte`

**Byte + short = Int**

**Tambien se puede asignar a un double**

**Reglas Numericas;**

**Numeric Promotion Rules**

1. If two values have different data types, Java will automatically promote one of the values to the larger of the two data types.
2. If one of the values is integral and the other is floating-point, Java will automatically promote the integral value to the floating-point value's data type.
3. Smaller data types, namely, `byte`, `short`, and `char`, are first promoted to `int` any time they're used with a Java binary arithmetic operator with a variable (as opposed to a value), even if neither of the operands is `int`.
4. After all promotion has occurred and the operands have the same data type, the resulting value will have the same data type as its promoted operands.

//

3. What change, when applied independently, would allow the following code snippet to compile? (Choose all that apply.)

```
3: long ear = 10;
4: int hearing = 2 * ear;
```

   - A. No change; it compiles as is.
   - B. Cast `ear` on line 4 to `int`.
   - C. Change the data type of `ear` on line 3 to `short`.
   - D. Cast `2 * ear` on line 4 to `int`.
   - E. Change the data type of `hearing` on line 4 to `short`.
   - F. Change the data type of `hearing` on line 4 to `long`.

**Error de compilacion porque no puede ser un tipo de primitivo de menores bits, tendria que ser long o castearlo.**

//

**FIGURE 2.2** The logical truth tables for &, |, and ^

AND (x & y)

| | y = true | y = false |
|---|---|---|
| x = true | true | false |
| x = false | false | false |

INCLUSIVE OR (x | y)

| | y = true | y = false |
|---|---|---|
| x = true | true | true |
| x = false | true | false |

EXCLUSIVE OR (x ^ y)

| | y = true | y = false |
|---|---|---|
| x = true | false | true |
| x = false | true | false |

**El valor canino es difetente a 20, teeth no se cambia y wolf se asigna a false porque solo usa un " = ".**
**Exclusive ^ indica que mientras exista un valor negativo, el resultado sera true.**

**4.** What is the output of the following code snippet?

```
3: boolean canine = true, wolf = true;
4: int teeth = 20;
5: canine = (teeth != 10) ^ (wolf=false);
6: System.out.println(canine+", "+teeth+", "+wolf);
```

**A.** `true, 20, true`
**B.** `true, 20, false`
**C.** `false, 10, true`
**D.** `false, 20, false`
**E.** The code will not compile because of line 5.
**F.** None of the above.

//

**5.** Which of the following operators are ranked in increasing or the same order of precedence? Assume the + operator is binary addition, not the unary form. (Choose all that apply.)

**A.** `+, *, %, --`
**B.** `++, (int), *`
**C.** `=, ==, !`
**D.** `(short), =, !, *`
**E.** `*, /, %, +, ==`
**F.** `!, ||, &`
**G.** `^, +, =, +=`

Por orden de prescedencia

//

**6.** What is the output of the following program?

```
1: public class CandyCounter {
2:    static long addCandy(double fruit, float vegetables) {
3:       return (int)fruit+vegetables;
4:    }
5:
6:    public static void main(String[] args) {
7:       System.out.print(addCandy(1.4, 2.4f) + ", ");
8:       System.out.print(addCandy(1.9, (float)4) + ", ");
9:       System.out.print(addCandy((long)(int)(short)2, (float)4)); } }
```

**A.** `4, 6, 6.0`
**B.** `3, 5, 6`
**C.** `3, 6, 6`
**D.** `4, 5, 6`
**E.** The code does not compile because of line 9.
**F.** None of the above.

F. El codigo no compila porque cuando realizamos el casteo en la linea 3, solo lo hacemos en fruit, tendriamos que castear tanto fruit como vegetable para que el codigo funcione correctamente.

//

7. What is the output of the following code snippet?

```
int ph = 7, vis = 2;
boolean clear = vis > 1 & (vis < 9 || ph < 2);
boolean safe = (vis > 2) && (ph++ > 1);
boolean tasty = 7 <= --ph;
System.out.println(clear + "-" + safe + "-" + tasty);
```

A. true-true-true
B. true-true-false
C. true-false-true
D. true-false-false
E. false-true-true
F. false-true-false
G. false-false-true
H. false-false-false

**Cuando utilizamos && ya no es necesario hacer la siguiente evaluacion**

//

8. What is the output of the following code snippet?

```
4: int pig = (short)4;
5: pig = pig++;
6: long goat = (int)2;
7: goat -= 1.0;
8: System.out.print(pig + " - " + goat);
```

A. 4 - 1
B. 4 - 2
C. 5 - 1
D. 5 - 2
E. The code does not compile due to line 7.
F. None of the above.

**Un short cabe en un int**
**Se hace una operacion compuesta en goat**

**Internamente se veria; // goat = (long) (goar - 1.0);**

//

9. What are the unique outputs of the following code snippet? (Choose all that apply.)

```
int a = 2, b = 4, c = 2;
System.out.println(a > 2 ? --c : b++);
System.out.println(b = (a!=c ? a : b++));
System.out.println(a > b ? b < c ? b : 2 : 1);
```

A. 1
B. 2
C. 3
D. 4
E. 5
F. 6
G. The code does not compile.

**Por presedencia y respetando los termarios el resultado seria (4, 5, 1)**

//

**10.** What are the unique outputs of the following code snippet? (Choose all that apply.)

```
short height = 1, weight = 3;
short zebra = (byte) weight * (byte) height;
double ox = 1 + height * 2 + weight;
long giraffe = 1 + 9 % height + 1;
System.out.println(zebra);
System.out.println(ox);
System.out.println(giraffe);
```

**A.** 1
**B.** 2
**C.** 3
**D.** 4
**E.** 5
**F.** 6
**G.** The code does not compile.

**No compila porque la multiplicacion de short debe ser un entero, no un double. Para que funcione se tendria que castear.**

//

**11.** What is the output of the following code?

```
11: int sample1 = (2 * 4) % 3;
12: int sample2 = 3 * 2 % 3;
13: int sample3 = 5 * (1 % 2);
14: System.out.println(sample1 + ", " + sample2 + ", " + sample3);
```

**A.** 0, 0, 5
**B.** 1, 2, 10
**C.** 2, 1, 5
**D.** 2, 0, 5
**E.** 3, 1, 10
**F.** 3, 2, 6
**G.** The code does not compile.

**Respetando orden de precedencia y parentesis**

//

**12.** The _____ operator increases a value and returns the original value, while the _____ operator decreases a value and returns the new value.

**A.** post-increment, post-increment
**B.** pre-decrement, post-decrement
**C.** post-increment, post-decrement
**D.** post-increment, pre-decrement
**E.** pre-increment, pre-decrement
**F.** pre-increment, post-decrement

**Post-increment; Incrementa el valor y regresa el valor original.**
**pre-decrement; decrementa el valor y regresa el nuevo valor.**

//

13. What is the output of the following code snippet?

```
boolean sunny = true, raining = false, sunday = true;
boolean goingToTheStore = sunny & raining ^ sunday;
boolean goingToTheZoo = sunday && !raining;
boolean stayingHome = !(goingToTheStore && goingToTheZoo);
System.out.println(goingToTheStore + "-" + goingToTheZoo
   + "-" +stayingHome);
```

A. `true-false-false`
B. `false-true-false`
C. `true-true-true`
D. `false-true-true`
E. `false-false-false`
F. `true-true-false`
G. None of the above

**Siguiendo la tabla de orden de precedencia el resultado seria true-true-false**

//

14. Which of the following statements are correct? (Choose all that apply.)
A. The return value of an assignment operation expression can be `void`.
B. The inequality operator (`!=`) can be used to compare objects.
C. The equality operator (==) can be used to compare a `boolean` value with a numeric value.
D. During runtime, the & and | operators may cause only the left side of the expression to be evaluated.
E. The return value of an assignment operation expression is the value of the newly assigned variable.
F. In Java, `0` and `false` may be used interchangeably.
G. The logical complement operator (`!`) cannot be used to flip numeric values.

**!= // se refiere a que apuntan a diferentes objetos, asi que es verdadero que se usa para comparar objetos**

**! // solo se puede usar con booleans**

//

15. Which operators take three operands or values? (Choose all that apply.)
A. =
B. &&
C. *=
D. `? :`
E. &
F. ++
G. /

**El ternario puede ocupar tres operandos**

//

**16.** How many lines of the following code contain compiler errors?

```
int note = 1 * 2 + (long)3;
short melody = (byte)(double)(note *= 2);
double song = melody;
float symphony = (float)((song == 1_000f) ? song * 2L : song);
```

**A.** 0
**B.** 1
**C.** 2
**D.** 3
**E.** 4

**Linea 1 no compila porque no podemos meter un long en un int.**

//

**17.** Given the following code snippet, what are the values of the variables after it is executed? (Choose all that apply.)

```
int ticketsTaken = 1;
int ticketsSold = 3;
ticketsSold += 1 + ticketsTaken++;
ticketsTaken *= 2;
ticketsSold += (long)1;
```

**A.** `ticketsSold` is 8.
**B.** `ticketsTaken` is 2.
**C.** `ticketsSold` is 6.
**D.** `ticketsTaken` is 6.
**E.** `ticketsSold` is 7.
**F.** `ticketsTaken` is 4.
**G.** The code does not compile.

**Por la suma implicita se puede realizar la operacion de int a long, asi que ticketSold es 6 y ticketTaken4.**

//

**18.** Which of the following can be used to change the order of operation in an expression? (Choose all that apply.)

**A.** `[ ]`
**B.** `< >`
**C.** `( )`
**D.** `\ /`
**E.** `{ }`
**F.** `" "`

**Los parentesis son los unicos que se usan para cambiar el orden de precedencia ( ) ;**

//

**19.** What is the result of executing the following code snippet? (Choose all that apply.)

```
3: int start = 7;
4: int end = 4;
5: end += ++start;
6: start = (byte)(Byte.MAX_VALUE + 1);
```

**A.** `start` is `0`.
**B.** `start` is `-128`.
**C.** `start` is `127`.
**D.** `end` is `8`.
**E.** `end` is `11`.
**F.** `end` is `12`.
**G.** The code does not compile.
**H.** The code compiles but throws an exception at runtime.

**Max.Value(); // regresa el valor maximo en este casi del byte que es 127**

**Underflow; en este caso se sale del rango del byte y se regrea al -128, que seria su valor principal.**

//

**20.** Which of the following statements about unary operators are true? (Choose all that apply.)

**A.** Unary operators are always executed before any surrounding numeric binary or ternary operators.
**B.** The `-` operator can be used to flip a `boolean` value.
**C.** The pre-increment operator (`++`) returns the value of the variable before the increment is applied.
**D.** The post-decrement operator (`--`) returns the value of the variable before the decrement is applied.
**E.** The `!` operator cannot be used on numeric values.
**F.** None of the above

**Las operaciones unarias siempre se ejecutan antes de los binarios o ternarios**
**Post-increment; Incrementa el valor y regresa el valor original.**
**El operador ! No se puede usar en valores numericos**

//

**21.** What is the result of executing the following code snippet?

```
int myFavoriteNumber = 8;
int bird = ~myFavoriteNumber;
int plane = -myFavoriteNumber;
var superman = bird == plane ? 5 : 10;
System.out.println(bird + "," + plane + "," + --superman);
```

**A.** `-7,-8,9`
**B.** `-7,-8,10`
**C.** `-8,-8,4`
**D.** `-8,-8,5`
**E.** `-9,-8,9`
**F.** `-9,-8,10`
**G.** None of the above

//