Chapter 2: Operators

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)`

El operador == se puede usar en booleanos para comparar que dos booleanos tengan el mismo valor. El operador ! invierte un valor booleano. Los operadores aritméticos no se pueden usar con booleanos.

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`

Debido a que los tipos de datos byte y short son los primitivos de menor capacidad, el resultado de la suma puede ser almacenado en tipos de datos, menos en booleanos.

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`.

El código no compilara ya que el valor de ear es de tipo long y este no se puede almacenar en la variable hearing de tipo int, por lo que una opción válida es hacer el cast de esa variable a int. También se puede hacer el cast a toda la operación. Otras opciones validas son cambiar el tipo de dato de ear por int para que sea igual al de hearing o por un short para que quepa en hearing.

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.

En la línea 5 primero se compara si teeth es diferente de 10 y es verdadero, después se le asigna el valor false a wolf y el resultado se compara con el operador ^ por lo que canine ahora es true.

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. `^, +, =, +=`

Algunas opciones como la B tienen un orden de precedencia correcto, pero es decreciente.

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.

Ninguna de las opciones dadas es correcta. El código realmente no compila, pero es porque el método addCandy debe regresar un valor de tipo long pero en el return se devuelve la suma de un valor de tipo double que se castea a int y uno de tipo float.

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`

El valor que se le dará a la variable clear es true, ya que el operador & hace que ambas expresiones se comparen incluso si la primera es false, el resultado de lo que está en paréntesis es true, ya que el valor de vis es menor que 9, el valor de la primera expresión también es true porque vis es mayor que 1. El valor de safe es false, ya que se está comparando con el operador &&, la primera expresión es false ya que vis es igual y no mayor a 2, la otra expresión ya no se compara. El valor de tasty es false, ya que 7 es menor que el valor de ph (a ph se le aplica el pre decremento haciendo que valga menos que 7).

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.

En la línea 5 al tener un post incremento de la misma variable a la que se le está asignando esta operación, se sobre escribe y el valor de la variable no cambia. En la línea 7 al hacer uso del operador -= se hace un cast automático permitiendo que el valor sea asignado a la variable de tipo long. Un valor short cabe en un tipo int, y un int en un long haciendo que el código compile sin problema.

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.

En la primera línea donde se hace la evaluación con el operador ternario da false ya que a es igual a 2 y no mayor, por esto el valor que se imprimirá será b++, pero al ser un post incremento primero se imprime b que es 4 y después se hace el incremento. En la segunda evaluación a es igual y no diferente a c, por lo tanto, es false dando como resultado de nuevo b++, este valor se le asigna a b, ocasionando que se sobrescriba el incremento, b queda con el mismo valor (5). En la tercera evaluación se obtiene false, ya que a es menor que b, haciendo que se imprima 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.

Si se hace cualquier tipo de operación con datos short o byte, el resultado se convertirá en int, haciendo que no se pueda almacenar en una variable de tipo short o byte.

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.

En la línea 11 debido al orden de precedencia primero se ejecuta lo que hay en paréntesis dando como resultado 8, después el resultado de 8 % 3 es 2. En la línea 12 el orden de precedencia de multiplicaciones y módulos es el mismo, por lo tanto, se ejecuta de izquierda a derecha, el resultado de 3 * 2 es 6 y su módulo con 3 es 0. En la línea 13 se ejecuta primero lo de los paréntesis, el resultado de 1 % 2 es 1 y al multiplicarlo por 5 se obtiene 5.

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

Con post incremento y post decremento primero devuelve el valor original y luego incrementa o decrementa. Con pre incremento y pre decremento hace el incremento o decremento y se devuelve ese mismo 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

El resultado de sunny & raining es false ya que raining es false, después, el resultado de ese false ^sunday es true, ya que solo sunday es true. Debido a que raining es false, al invertir su valor se obtiene true y al comparar (&&) con sunday da true. Ya que el resultado de goingToTheStore && goingToTheZoo es true, al invertir ese valor da 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.

El operador != se puede usar para comparar que dos objetos no tengan la misma referencia. El operador ! no se puede usar para negar valores numéricos, para hacer algo similar se usaría el -. Los operadores & y | comparan siempre ambos valores.

15. Which operators take three operands or values? (Choose all that apply.)

A. `=`

B. `&&`

C. `*=`

D. `? :`

E. `&`

F. `++`

G. `/`

De las opciones solo el operador ternario toma tres valores, los demás son operadores de tipo binarios.

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

Solo se tiene un error, ya que, en la primera línea, se intenta almacenar un valor long a 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.

El resultado de ticketsSold += 1 + ticketsTacken++ es 5, ya que se suma 3 y 1, a ese resultado se le suma ticketsTaken (1), el resultado se almacena en ticketsSold y por ultimo ticketsTaken se incrementa en 1. TicketsTaken que ahora vale 2 se multiplica por 2, dando como resultado 4. ticketsSold += (long)1 es 6, a pesar de que se hace el casteo a long y la variable donde se almacenara es int, al tener el operador += se hace un casteo automático, pasando el valor a int.

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 () se pueden usar para cambiar el orden de una operación debido al 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.

Debido a que el valor máximo de un Byte es 127, al querer obtener la suma de este valor y 1, y esto castearlo a byte, se obtiene el valor pero negativo, ya que esta fuera del rango de este tipo de dato.

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

El operador que se usa para invertir valores booleanos es !, el – es para valores numéricos. El operador pre incremento primero hace el incremento antes de devolver el valor.