Chapter 3 Selection Statements

1. \(<, <=, ==, !=, >, >=\)

2. \((true) \&\& (3 > 4)\)
   false

   \(! (x > 0) \&\& (x > 0)\)
   false

   \((x > 0) \|\| (x < 0)\)
   true

   \((x != 0) \|\| (x == 0)\)
   true

   \((x >= 0) \|\| (x < 0)\)
   true

   \((x != 1) == !(x == 1)\)
   true

3. \((x > 1) \&\& (x < 100)\)

4. \(((x > 1) \&\& (x < 100)) \|\| (x < 0)\)

5. \(x > y > 0\)
   incorrect

   \(x = y \&\& y\)
   incorrect

   \(x /= y\)
   correct

   \(x or y\)
   incorrect

   \(x and y\)
   incorrect

6. No. Boolean values cannot be cast to other types.

7. \(x is 2.\)

8.
x is 1.

9.

d
d
false
-4

10. Note: else matches the first if clause. No output if \(x = 3\) and \(y = 2\). Output is “z is 7” if \(if\ \ x = 3\ \ and\ \ y = 4\). Output is “x is 2” if \(if\ \ x = 2\ \ and\ \ y = 2\).

```
if x > 2
    true
    y > 2
    false
    System.out.println("x is " + x);

false
    true
    System.out.println("z is " + z);
```

11. a, c, and d are the same. (B) and (C) are correctly indented.

12. No output if \(x = 2\) and \(y = 3\). Output is “x is 3” if \(x = 2\) and \(y = 2\). Output is “z is 6”.

13. Yes.

14. 0.5, 0.0, 0.234

15. \((\text{int})(\text{Math.random()} \times 20)\)

\[10 + (\text{int})(\text{Math.random()} \times 10)\]

\[10 + (\text{int})(\text{Math.random()} \times 41)\]
16. Switch variables must be of `char`, `byte`, `short`, or `int` data types. If a `break` statement is not used, the next `case` statement is performed. You can always convert a `switch` statement to an equivalent `if` statement, but not an `if` statement to a `switch` statement. The use of the `switch` statement can improve readability of the program in some cases. The compiled code for the `switch` statement is also more efficient than its corresponding `if` statement.

17. `y` is 2.

18. `switch (a) {
    case 1: x += 5; break;
    case 2: x += 10; break;
    case 3: x += 16; break;
    case 4: x += 34;
    }

19. `System.out.println((count % 10 == 0) ? count + "\n" : count + " ");`

20. The specifiers for outputting a boolean value, a character, a decimal integer, a floating-point number, and a string are `%b, %c, %d, %f, and %s.`

21. (a) the last item 3 does not have any specifier.
   (b) There is not enough items
   (c) The data for `%f` must a floating-point value

22.
(a) amount is 32.320000 3.233000e+01
(b) amount is 32.3200 3.2330e+01
(c) *false // * denote a space
(d) **Java // * denote a space
(e) false*****Java
(f) *falseJava

23. Use the String.format method to create a formatted string.

24. The precedence order for boolean operators is &., &., &&, and ||

   true | true && false is false
   true || true && false is true

25. a. The operands are evaluated first and from left to right. So (-i + i + i++) is -1 - 1 -1. i++ return the value of i, then i is incremented by 1. So, in the next println statement i + ++i is 0 + 1.

   b. The operands are evaluated first and from left to right. So, the left i before the + operator is 0 and right i is assigned to 1 by (i = 1). Therefore the println statement prints 1.

   c. The operands are evaluated first and from left to right. So, before i in the right of the + operator is evaluated, i is assigned to 1. Therefore, (i = 1) + i evaluates to 2.

26. 

   a = (a = 3) + a; => a = 3 + 3 = 6
   a = a + (a = 3); => a = 1 + 3 = 4
   a += a + (a = 3); => the value of a in the left side of += is obtained (1), a + (a = 3) is 4, therefore, the final result a is 5.
   a = 5 + 5 * 2 % a--; => a = 5 + 10 % a = 5 + 0 = 5
   a = 4 + 1 + 4 * 5 % (++a + 1) => a = 5 + 4 * 5 % (++a + 1) = 5 + 20 % (++a + 1) = 5 + 0 % (2 + 1) = 5 + 0 % 3 = 5 + 2 = 7;
   d += 1.5 * 3 + (++d); => the value of d in the left side of += is obtained (1.0), 1.5 * 3 + (++d) = 4.5 + (++)d = 4.5 + 2.0 = 6.5, therefore, the final result a is 7.5.
   d -= 1.5 * 3 + d++; => the value of d in the left side of += is obtained (1.0), 1.5 * 3 + d++ = 4.5 + d++ = 4.5 + 1.0 = 5.5, therefore, the final result is 1.0 - 5.5 = -4.5.