Chapter 4. Boolean operations - The boolean Primitive

This chapter presents exercises which enable you to practice your knowledge of the **boolean** primitive type and its associated operators. Exercises on compound boolean expressions are also presented.

1.	Describe what the `boolean` type is.	The `boolean` is a type which represents logical *true* or logical *false*.
2.	What is the range of values that the Java type `boolean` can support?	Either of the following two distinct values: `true` *or* `false` *(case sensitive)*.
3.	Write a line of code to declare a `boolean` variable named `b01` and assign it a value of `true`.	`boolean b01 = true;`

Boolean Testing

4. I have an `int` variable `x` with a value of `5` and an `int` variable `y` with a value of `7`. Write a program which tests whether *x is equal to y*, putting the result of this test into a `boolean` variable named `result`. Print your findings to the console, stating the following: *"The result of whether 5 is equal to 7 is* `<result>`*"*

```java
public class PracticeYourJava {
  public static void main(String[] args) {

    boolean result;
    int x = 5;
    int y = 7;

    result = x == y;

    System.out.printf("The result of whether %d is equal to %d = %b\n",x,y,result);
  }
}
```

Notes

Think about the statement `result = x==y;` in the following way:

`result = (is x equal to y, true or false?)`

Also note that `result = x==y;` can also be written as `result = (x==y);` that is, with brackets, if it makes the statement easier to understand.

5. I have an `int x` with a value of `5` and an `int y` with a value of `7`. Write a program which tests whether *x is greater than y*, putting the result of this test into a `boolean` variable named `result`. Print your findings out to the console.

```java
public class PracticeYourJava {
  public static void main(String[] args) {

    boolean result;
    int x = 5;
    int y = 7;

    result = x > y;

    System.out.printf("The result of whether %d is greater than %d = %b\n",x,y,result);
  }
}
```

6. I have an `int x` with a value of `5` and an `int y` with a value of `7`. Write a program which tests whether *x is less than or equal to y*, putting the result of this test into a `boolean` variable named `result`. Print your findings out to the console.

```java
public class PracticeYourJava {
  public static void main(String[] args) {

    boolean result;
    int x = 5;
    int y = 7;

    result = x <= y;

    System.out.printf("The result of whether %d is less than or equal to %d = %b\n",x,y,result);
  }
}
```

7. I have an `int` `x` with a value of `5` and an `int` `y` with a value of `7`. Write a program which tests whether *x is greater than or equal to y*, putting the result of this test into a `boolean` variable named `result`. Print your findings out to the console.

```
public class PracticeYourJava {
  public static void main(String[] args) {

    boolean result;
    int x = 5;
    int y = 7;

    result = x >= y;

    System.out.printf("The result of whether %d is >=  %d = %b\n",x,y,result);
  }
}
```

8. I have an `int` `x` with a value of `5` and an `int` `y` with a value of `7`. Write a program which tests whether *x is less than y*, putting the boolean result of this test into a `boolean` variable named `result`. Print your findings out to the console.

```
public class PracticeYourJava {
  public static void main(String[] args) {

    boolean result;
    int x = 5;
    int y = 7;

    result = x < y;

    System.out.printf("The result of whether %d is less than %d = %b\n",x,y,result);
  }
}
```

9. I have an `int` `x` with a value of `5` and an `int` `y` with a value of `7`. Write a program which tests whether *x is not equal to y*. Put the boolean result of this test into a `boolean` variable named `result` and print your findings out to the console.

```
public class PracticeYourJava {
  public static void main(String[] args) {

    boolean result;
    int x = 5;
    int y = 7;

    result = x != y;

    System.out.printf("The result of whether %d is not equal to %d = %b\n",x,y,result);
  }
}
```

10. I have an `int` `x` with a value of `5` and an `int` `y` with a value of `7`. Write a program which tests whether *x is <u>not</u> greater than y*. Put the boolean result of this test into a `boolean` named `result` and print your findings out to the console.

```
public class PracticeYourJava {
  public static void main(String[] args) {

    boolean result;
    int x = 5;
    int y = 7;

    result = !(x > y);

    System.out.printf("The result of whether %d is less than or equal to %d = %b\n",x,y,result);
  }
}
```

Note: There is no `not greater than` operator. Therefore putting the symbol for `not` (`!`) outside the bracket where the `>` is used has the desired effect of `not greater than`.

11. I have a `boolean x` with a value of `false` and a `boolean y` with a value of `true`. Write a program which tests whether *x is equal to y*. Put the boolean result of this test into a `boolean` variable named `result` and print your findings out to the console.

```java
public class PracticeYourJava {
  public static void main(String[] args) {

    boolean x=false, y=true;

    boolean result = x == y;
    System.out.printf("The result of whether %b == %b is %b",x,y,result);
  }
}
```

12. Write a program which tests whether a given `int x` is greater than 5 and a given `int y` is greater than 7.
Test the program with the following combinations of `x` and `y` respectively and print the result out after each test: (1,1), (6,7) and (15,15).

Solution #1

```java
public class PracticeYourJava {
  public static void main(String[] args) {

    boolean result;
    int x,y;

   x = 1; y = 1;   result = ((x > 5) && (y > 7)); System.out.println(result);
   x = 6; y = 7;   result = ((x > 5) && (y > 7)); System.out.println(result);
   x = 15; y = 15; result = ((x > 5) && (y > 7)); System.out.println(result);
  }
}
```

Solution #2

```java
public class PracticeYourJava {
  public static void main(String[] args) {

    boolean result;
    int x,y;

    x = 1; y = 1;   result = ((x > 5) & (y > 7)); System.out.println(result);
    x = 6; y = 7;   result = ((x > 5) & (y > 7)); System.out.println(result);
    x = 15; y = 15; result = ((x > 5) & (y > 7)); System.out.println(result);

  }
}
```

13. What is the difference between the two solutions presented to the preceding exercise?

The first solution is written using a "short-circuiting" logical operator, whereas the second one is not.

14. What is the difference between short-circuiting logical operators and the regular logical operators?

The short-circuiting logical operators stop testing as soon as they realize that the whole test can be deemed to have failed/passed as soon as the failure/passing of a sub-test of the test under assessment can be determined as causing the failure/passing of the whole test; therefore there would be no point in continuing/completing the evaluation. Using the first test in the solution to exercise 12 above as an example, there is no point in bothering to check whether `y > 7` when you already know that `x` is less than 5 and that both parts of the test are required to be true. The non-short-circuiting logical operators however will evaluate the whole conditional statement instead of stopping at a failed/passed sub-test.
The result of this difference is that the short-circuiting logical operator can potentially perform faster than the regular logical operators.

15. Write a program which tests whether `x` is greater than 5 and `y` is less than 7. (Note: `x`, `y` and any other variables in this and subsequent exercises in this chapter are of type `int`; however you can use any mixture of numerical primitives that you prefer).
Test the program with the following combinations of `x` and `y` respectively and print the result out after each test:

(1,1), (6,5) and (4,5).

```java
public class PracticeYourJava {
  public static void main(String[] args) {
    boolean result;
    int x;
    int y;
    x = 1; y = 1;  result = ((x > 5) && (y < 7)); System.out.println(result);
    x = 6; y = 5;  result = ((x > 5) && (y < 7)); System.out.println(result);
    x = 4; y = 5;  result = ((x > 5) && (y < 7)); System.out.println(result);
  }
}
```

16. Explain the difference between logical OR and XOR.

Logical OR means that “at least one of the conditions being compared presented is true”.
Logical XOR means “absolutely only one of the conditions being compared can be true”.

17. Write a program which tests whether x is greater than 5 or y is less than 7.
Test the program with the following combinations of x and y respectively and print the result out after each test: (1,1), (6,5) and (4,15).

```java
public class PracticeYourJava {
  public static void main(String[] args) {
    boolean result;
    int x,y;
    x = 1; y = 1;  result = ((x > 5) || (y < 7)); System.out.println(result);
    x = 6; y = 5;  result = ((x > 5) || (y < 7)); System.out.println(result);
    x = 4; y = 15; result = ((x > 5) || (y < 7)); System.out.println(result);
  }
}
```

18. Write a program which tests whether x is greater than or equal to 5 and y is equal to 0.
Test the program with the following combinations of x and y respectively and print the result out after each test: (6,0), (5,0) and (4,1).

```java
public class PracticeYourJava {
  public static void main(String[] args) {
    boolean result;
    int x,y;
    x = 6; y = 0; result = ((x >= 5) && (y == 0)); System.out.println(result);
    x = 5; y = 0; result = ((x >= 5) && (y == 0)); System.out.println(result);
    x = 4; y = 1; result = ((x >= 5) && (y == 0)); System.out.println(result);
  }
}
```

19. Write a program which tests whether x is greater than or equal to 5 and y is *not* equal to 0.
Test the program with the following combinations of x and y respectively and print the result out after each test: (6,0), (5,0), (4,1) and (7,2).

```java
public class PracticeYourJava {
  public static void main(String[] args) {
    boolean result;
    int x,y;
    x = 6; y = 0; result = ((x >= 5) && (y != 0)); System.out.println(result);
    x = 5; y = 0; result = ((x >= 5) && (y != 0)); System.out.println(result);
    x = 4; y = 1; result = ((x >= 5) && (y != 0)); System.out.println(result);
    x = 7; y = 2; result = ((x >= 5) && (y != 0)); System.out.println(result);
  }
}
```

20. Write a program which tests whether *only one of* x or y is greater than 5.
Test the program with the following combinations of x and y respectively and print the result out after each test: (5,5), (6,6), (3,3), (1,10) and (10,1).

Hint: xor (^)

```java
public class PracticeYourJava {
  public static void main(String[] args) {

    boolean result;
    int x,y;

    x = 5; y = 5;  result = ((x > 5) ^ (y > 5)); System.out.println(result);
    x = 6; y = 6;  result = ((x > 5) ^ (y > 5)); System.out.println(result);
    x = 3; y = 3;  result = ((x > 5) ^ (y > 5)); System.out.println(result);
    x = 1; y = 10; result = ((x > 5) ^ (y > 5)); System.out.println(result);
    x = 10; y = 1; result = ((x > 5) ^ (y > 5)); System.out.println(result);
  }
}
```

21. Write a program which tests whether x is greater than 5 or y is greater than 15 or z is less than or equal to 25. Test the program with the following combinations of x, y and z respectively and print the result out after each test: (5,16,25), (5,16,24), (5,15,24) and (4,5,30).

```java
public class PracticeYourJava {
  public static void main(String[] args) {

    boolean result;
    int x,y,z;

    x=5; y=16; z=25; result = ((x > 5) || (y > 15) || (z <= 25)); System.out.println(result);
    x=5; y=16; z=24; result = ((x > 5) || (y > 15) || (z <= 25)); System.out.println(result);
    x=5; y=15; z=24; result = ((x > 5) || (y > 15) || (z <= 25)); System.out.println(result);
    x=4; y=5;  z=30; result = ((x > 5) || (y > 10) || (z <= 25)); System.out.println(result);
  }
}
```

22. Write a program which tests whether x is greater than or equal to 5 and either of y or z are less than 15. Test the program with the following combinations of x, y and z respectively and print the result out after each test: (5,14,20), (5,15,13), (5,10,10) and (4,5,30).

```java
public class PracticeYourJava {
  public static void main(String[] args) {

    boolean result;
    int x,y,z;

    x=5; y=14; z=20; result = ((x >= 5) && ((y < 15) || (z < 15))); System.out.println(result);
    x=5; y=15; z=13; result = ((x >= 5) && ((y < 15) || (z < 15))); System.out.println(result);
    x=5; y=10; z=10; result = ((x >= 5) && ((y < 15) || (z < 15))); System.out.println(result);
    x=4; y=5;  z=30; result = ((x >= 5) && ((y < 15) || (z < 15))); System.out.println(result);
  }
}
```

If desired, the individual sub-conditions can be broken down in the following manner:

```java
 boolean subResult01 = x >= 5;
 boolean subResult02 = ((y < 15) || (z < 15));
 boolean result = (subResult01 && subResult02);
```

23. Write a program which tests whether x is greater than or equal to 5 or either y or z are less than 15. Test the program with the following combinations of x, y and z respectively and print the result out after each test: (5,14,20), (5,15,13), (5,10,10), (4,5,30), (1,20,7) and (1,20,20).

```java
public class PracticeYourJava {
  public static void main(String[] args) {

    boolean result;
    int x,y,z;

    x=5;y=14;z=20;  result = ((x >= 5) || ((y < 15) || (z < 15))); System.out.println(result);
    x=5;y=15;z=13;  result = ((x >= 5) || ((y < 15) || (z < 15))); System.out.println(result);
    x=5;y=10;z=10;  result = ((x >= 5) || ((y < 15) || (z < 15))); System.out.println(result);
    x=4;y=5;z=30;   result = ((x >= 5) || ((y < 15) || (z < 15))); System.out.println(result);
    x=1;y=20;z=7;   result = ((x >= 5) || ((y < 15) || (z < 15))); System.out.println(result);
    x=1;y=20;z=20;  result = ((x >= 5) || ((y < 15) || (z < 15))); System.out.println(result);
  }
}
```

24. Write a program which tests whether `x` is greater than or equal to 5 or only one of `y` or `z` is less than 15.
Test the program with the following combinations of `x`, `y` and `z` respectively and print the result out after each test: (5,14,14), (5,20,13), (6,5,30) and (6,20,20).

```
public class PracticeYourJava {
  public static void main(String[] args) {
    boolean result;
    int x,y,z;
    x=5;y=14;z=14;  result = ((x >= 5) || ((y < 15) ^ (z < 15))); System.out.println(result);
    x=5;y=20;z=13;  result = ((x >= 5) || ((y < 15) ^ (z < 15))); System.out.println(result);
    x=6;y=5;z=30;   result = ((x >= 5) || ((y < 15) ^ (z < 15))); System.out.println(result);
    x=6;y=20;z=20;  result = ((x >= 5) || ((y < 15) ^ (z < 15))); System.out.println(result);
  }
}
```

25. Write a program which tests whether `x` is greater than or equal to 5 and only one of `y` or `z` is less than 15.
Test the program with the following combinations of `x`, `y` and `z` respectively and print the result out after each test: (5,14,14), (5,15,13) and (5,10,10).

```
public class PracticeYourJava {
  public static void main(String[] args) {
    boolean result;
    int x,y,z;
    x=5;y=14;z=14;  result = ((x >= 5) && ((y < 15) ^ (z < 15))); System.out.println(result);
    x=5;y=15;z=13;  result = ((x >= 5) && ((y < 15) ^ (z < 15))); System.out.println(result);
    x=5;y=10;z=10;  result = ((x >= 5) && ((y < 15) ^ (z < 15))); System.out.println(result);
  }
}
```

26. Given the following values and boolean statements, state at which point the value of the result is determined and the processing moves on to the next command.

```
1. x=7;y=14;z=14;     result = ((x >= 5) || ((y < 15) ^ (z < 15)));
2. x=5;y=20;z=13;     result = ((x >= 5) |  ((y < 15) ^ (z < 15)));
3. x=6; y=0;          result = ((x >= 5) && (y == 0));
4. x=4; y=0;          result = ((x >= 5) && (y == 0));
5. x=4; y=0;          result = ((x >= 5) &  (y == 0));
```

1. The processing can stop right after determining that `x >= 5` is true. The reason for this is because the statement to be evaluated has two parts; and given that they are bound by an OR statement, as soon as one part of it whole statement is true, then the overall statement is true. Therefore, because there is a short-circuiting OR operator between both parts, the evaluation can stop as early as possible.
2. The processing will evaluate right to the end of the statement. The reason is because we use a regular OR operator, rather than a short-circuiting one.
3. The statement must be fully evaluated, because even though it uses a short-circuiting boolean operator, the fact that it is an AND operator mandates that all parts have to be true, therefore all parts have to be evaluated until either of 1) a subpart being evaluated is determined to be false or 2) the end of the statement.
4. Evaluation is completed as soon as it is determined that x is less than 5, given that we have a short-circuiting operator.
5. The whole statement is evaluated, even after it is determined that the first part has failed (and thus in this case the whole statement has failed) because we are not using a short-circuiting operator.

In the chapter on *Conditional Processing*, we will use the return values from boolean operations to make decisions on which parts (branches) of our code to run.