Chapter 3 WS

1. Which ***relational*** symbol do I use for the following scenarios?

Scenario	**Relational Symbol**
To find out if n1 is larger than n2	**>**
To find out if n1 is the same as n2	**==**
To find out if n1 is the same as or larger than n2	**>=**
To find out if n1 is smaller than n2	**<**
To find out if n1 is smaller than or the same as n2	**<=**
To find out if n1 is not the same as n2	**!=**

2. Which ***logical*** symbol do I use for the following scenarios?

Scenario	**Logical Symbol**
You want to do something if a and b are both true	**&&**
You want to do something if either a or b is true, you don't care if both are	**||**
You want to do something if a is false	**!**
You want to do something if a or b is true, but not both	**(a || b) && !(a && b)**

3. Evaluate the following, based on:

```
int x = 2;
int y = 3;
int z = 6;
```

   a. **(x < y && y < z) evaluates to True.**
   b. **(x < y || y < z) evaluates to True.**
   c. **!(x < y) evaluates to False.**
   d. **(x + y < z) evaluates to True.**
   e. **(x - y < z) evaluates to True.**

4. Evaluate the following based on:

```
int x =4;
int y = 5;
```

   a. **!(x == y) evaluates to true.**

b. x != y evaluates to true.

c. x == y evaluates to false.

d. x >= y evaluates to false.

5. What are the results of the following?
   a. (8<6) && (2==3) **evaluates to False.**
   b. (8<6) || (2==3) **evaluates to False.**
   c. (8<6) ^ (2==3) **evaluates to True.**
6. What is x after evaluating the following:

```
int x = 10;
int y = 10;

    (y > 10) && (x-- > 10);
```

The value of x remains 10.

7. What will be evaluated first for the following Java statements?
   true || true && true: **The result is true.**
   true || !true && true: **The result is true.**
   true && true ^ true: **The result is false.**
   true == true != true: **The result is false.**

8. Will the result of 1.0 + 1.0 + 1.0 + 1.0 + 1.0 == 5.0 always evaluate to true?

   The result of 1.0 + 1.0 + 1.0 + 1.0 + 1.0 == 5.0 may not always be true. You see, when dealing with floating-point numbers, there can be some tricky precision issues. The thing is, floating-point representations have limited accuracy, so certain decimal values cannot be represented exactly. Now, most of the time, the result will probably be true because the values 1.0 and 5.0 are pretty friendly to the floating-point format. But, here's the catch: due to those precision limitations, there might be some tiny rounding errors happening during the calculations. And these little deviations can lead to the final result being slightly different from 5.0, causing the expression to evaluate as false. To tackle this, handle floating-point comparisons with a touch of flexibility. Instead of directly checking for equality using ==, you can consider using a small difference tolerance. You know, like saying "close enough is good enough." By comparing the absolute difference between the values with a small epsilon, you can account for those precision quirks. So, something like Math.abs(a - b) < epsilon could be a more reliable way to compare floating-point numbers.

9. If given int x = 3 and int y =9, what is the outcome of the following:
   a. x / y > 0 **is false.**
   b. y % x == 0 **is true.**
   c. x >= y **is false.**
   d. y >= x **is true.**
   e. y == x **is false.**
10. If given int age = 79, what is the outcome of the following:

a. age >= 65 **is true.**

b. age >= 18 **is true.**

c. age >= 21 **is true.**

d. age >= 25 **is true.**

e. age >= 16 **is true.**

11. If given double side = 5, what is the value for valid, each statement is independent:
    a. boolean valid = side >= 0; will be true.

    b. boolean valid = side > 0; will be true.

    c. boolean valid = side > -1; will be true.

12. If given int num = 7; what is the value for jackpot, each statement is independent:
    a. boolean jackpot = num == 9; will be false.

    b. boolean jackpot = num == 7; will be true.

    c. boolean jackpot = num == 1; will be false.

13. If given int score = 87, what is the value for passing, each statement is independent:
    a. boolean passing = score >= 60; will be true.

    b. boolean passing = score >= 90; will be false.

    c. boolean passing = score >= 70; will be true.

14. Declare a boolean variable doorClosed to false.
    boolean doorClosed = false;

15. Declare a boolean variable isTeenager that evaluates true if age is greater than 13 and less **than 18.**
    int age = // assign age here
    boolean isTeenager = age > 13 && age < 18;

16. Declare a boolean variable carCanTurnOn if hasKey and brakeEngaged are true.
    boolean hasKey = // assign hasKey here
    boolean brakeEngaged = // assign brakeEngaged here
    boolean carCanTurnOn = hasKey && brakeEngaged;

17. Declare a boolean variable, shortPerson, that compares Napoleon's height, 68, to 67 (the **average height during his time). If his height is less than the average, short is true, otherwise it is false.**
    int napoleonHeight = 68;
    boolean shortPerson = napoleonHeight < 67;

18. Declare a boolean variable, correctWeight, that evaluates true if either weight is greater than **50 lbs and height is greater than 60 inches.**
    int weight = // assign weight here
    int height = // assign height here
    boolean correctWeight = weight > 50 && height > 60;

19. Declare a boolean variable, h_w_eight, that evaluates true if either weight is greater than 50 **lbs or height is greater than 60 inches, but not both.**
boolean h_w_eight = (weight > 50) ^ (height > 60); // ^ is the XOR operator in Java

20. Assume x is 0. What is the output of the following statement?

```
if (x > 0)
    System.out.print("x is greater than 0");
else if (x < 0)
    System.out.print("x is less than 0");
else
    System.out.print("x equals 0");
```

The output of the statement will be: x equals 0. Since x is equal to 0, none of the conditions in the if and else if statements evaluate to true. As a result, the code inside the else block is executed

21. Correct the following statement:

```
If i > 0 {
    System.out.print(“i is positive.”);
}
```

The corrected statement:
if (i > 0) { System.out.print("i is positive."); }

22. Fix the following code:

```
double change;

if change >= 0
    System.out.print(“No change.”);
else change >= 50;
    System.out.print(“50% increase);
if change >= 80
    System.out.print(“80% increase”);
```

The fixed code:
double change; if (change >= 0) { System.out.print("No change."); } else if (change >= 50) { System.out.print("50% increase"); } else if (change >= 80) { System.out.print("80% increase"); }

23. Pick the better Java statement for the following:

```
boolean test = true;

if (test == true)
    System.out.println("I'll always print out.");


if (test)
```

```
        System.out.println("I'll always print out.");
```

The better Java statement:
boolean test = true; if (test) System.out.println("I'll always print out.");

24. Fix the following if statements so each is coherent, and easy to read.

```
if (danger == true){
    System.out.println(“Run!”);


if( danger == false){
    System.out.println(“Relax”);


if(illegalMove == true){
    System.out.println(“Go for it!”);


if (SchoolIn != false){
        System.out.println(“Drive normal speed”);
```

```
if (danger) {
    System.out.println("Run!");
}

if (!danger) {
    System.out.println("Relax");
}

if (illegalMove) {
    System.out.println("Go for it!");
}

if (SchoolIn) {
    System.out.println("Drive normal speed");
}
```

25. Given the code below, does that mean it must be dark?

```
boolean lightsOn = true;
if (lightsOn){
    System.out.println(“It must be dark”);
}
```

No, the code does not guarantee that it must be dark. It only implies that the lightsOn variable is true, indicating that the lights are turned on.

26. Something to think about:

```
boolean raining = true;
boolean cold = true;
if (raining && cold){
    System.out.println("I need an umbrella and a jacket");
else{
        System.out.println("I don't need an umbrella, nor a jacket – but wait is that true?  What if
    is raining but not cold, or cold but not raining?");
```

This code is designed to handle a specific scenario. If it is both raining and cold, it prints "I need an umbrella and a jacket." Otherwise, it prints the second message that considers different combinations of raining and cold conditions.

27. Declare a double variable, timeOfDay. If timeOfDay is less than 10, print out "Good Morning!" If the timeOfDay is greater than 18, print out "Good Evening!"

```
double timeOfDay = 14.5;

if (timeOfDay < 10) {
    System.out.println("Good Morning!");
}

if (timeOfDay > 18) {
    System.out.println("Good Evening!");
}
```

28. Write the Java statements that will test if num is even or odd.

```
int num = 7;

if (num % 2 == 0) {
    System.out.println("The number is even.");
} else {
    System.out.println("The number is odd.");
}
```

29. What is the output for the following:

```
int score = 88;
if (score >= 60)
    System.out.println("D");
else if (score >= 70)
    System.out.println("C");
else if (score >= 80)
    System.out.println("B");
else if (score >= 90)
    System.out.println("A");
else
```

System.out.println(“F”);

The output will be "B". Since score is greater than or equal to 80, but less than 90, it falls into the "B" range and "B" will be printed.

30. Rewrite the following statements using a Boolean expression:

```
boolean newLine;
if(count % 10 == 0)
    newLine = true;
else
    newLine = false;
```

boolean newLine = count % 10 == 0;

31. Rewrite the following to a Java test statement (that could be used for an if statement or Boolean expression):

1 <= numberOfDaysInAMonth <= 31

boolean testStatement = 1 <= numberOfDaysInAMonth && numberOfDaysInAMonth <= 31;

32. What is the output for the following?

```
int x = 0;
if (x > 0);
{
   System.out.println("x");
   }
```

The output will be "x". The semicolon after the if (x > 0) statement, doesn't affect the flow of execution. The block following the if statement (within curly braces) will be executed unconditionally, resulting in "x" being printed.

33. What is the output?

```
int q = 0;
if (q > 0);
{
   System.out.println("x");
}
```

The semicolon immediately after if (q > 0) acts as an empty statement, terminating the if statement. Therefore, the block within the curly braces will be executed unconditionally, resulting in "x" being printed.

34. Which of the following code displays the area of a circle if the radius is positive.

a.
```
if (radius <= 0)
        System.out.println(radius * radius * 3.14159);
```
b.
```
if (radius != 0)
        System.out.println(radius * radius * 3.14159);
```
c.
```
if (radius >= 0)
        System.out.println(radius * radius * 3.14159);
```
d.
```
if (radius > 0)
        System.out.println(radius * radius * 3.14159);
```

35. What is the output for the following code?

```
int temp = 78;
if (temperature <= 100)
   System.out.println("too hot");
else if (temperature <= 40)
   System.out.println("too cold");
else
   System.out.println("just right");
```

The output will be "too hot". Since the temperature (temp) is 78, the condition temperature <= 100 is true. Therefore, "too hot" will be printed.

36. What is the output for the following? If it is not correct, fix it!

```
int score = 88;
if (score >= 60)
     System.out.println(“D”);
else if (score >= 70)
     System.out.println(“C”);
else if (score >= 80)
     System.out.println(“B”);
else if (score >= 90)
     System.out.println(“A”);
else
     System.out.println(“F”);
```

The output will be "B". Since the score is 88, it satisfies the condition score >= 80. The code executes the corresponding println statement for the first matching condition, which in this case is "B".

37. Write an if statement that increases pay by 3% if score is greater than 90, otherwise increase pay by 1%.

```
if (score > 90) {
   pay *= 1.03; // Increase pay by 3%
} else {
   pay *= 1.01; // Increase pay by 1%
}
```

38. Generating random numbers with the Math.random() method that are between:

    a. 1 and 5, inclusive
    int randomNumber1To5 = (int) (Math.random() * 5) + 1;

    b. 1 and 10, inclusive
    int randomNumber1To10 = (int) (Math.random() * 10) + 1;

    c. 100 and 199, inclusive
    int randomNumber100To199 = (int) (Math.random() * 100) + 100;

d. 20 and 80, inclusive
int randomNumber20To80 = (int) (Math.random() * 61) + 20;

39. What is the output of the following switch statement?

```
char ch = 'a';
switch (ch) {
   case 'a':
   case 'A':
      System.out.print(ch); break;
   case 'b':
   case 'B':
      System.out.print(ch); break;
   case 'c':
   case 'C':
      System.out.print(ch); break;
   case 'd':
   case 'D':
      System.out.print(ch);       }
```

The output will be "a". Since ch is 'a', it matches the first case label 'a'. The code under that case label is executed, which prints 'a'. The subsequent case labels ('A', 'b', 'B', 'c', 'C', 'd', 'D') are not executed as there is no fall-through due to the break statement.