REVIEW QUESTIONS

Section 3.2

3.1 List six comparison operators.

Sol:

<, less than other
<=, less than or equal to other
>, greater than other
>=, greater than or equal to other
==, equal to other
!=, not equal to other

3.2 Can the following conversions involving casting be allowed? If so, find the converted result.

a.
```
boolean b = true;
i = (int)b;
```

b.
```
int i = 1;
boolean b = (boolean)i;
```

Sol:

a. boolean can not be converted other data types.
b. other data types can not be converted boolean.
Therefore both of cast will be resulted syntax error.

Sections 3.3–3.11

3.3 What is the printout of the code in (a) and (b) if number is 30 and 35, respectively?

Sol:

a. Number is 35 just "35 is odd." will be printed but when number is 30 both of print statement will be printed.

Because if just one if statement used and that statement is true, there will be not selection in the code.

All statement will be executed.

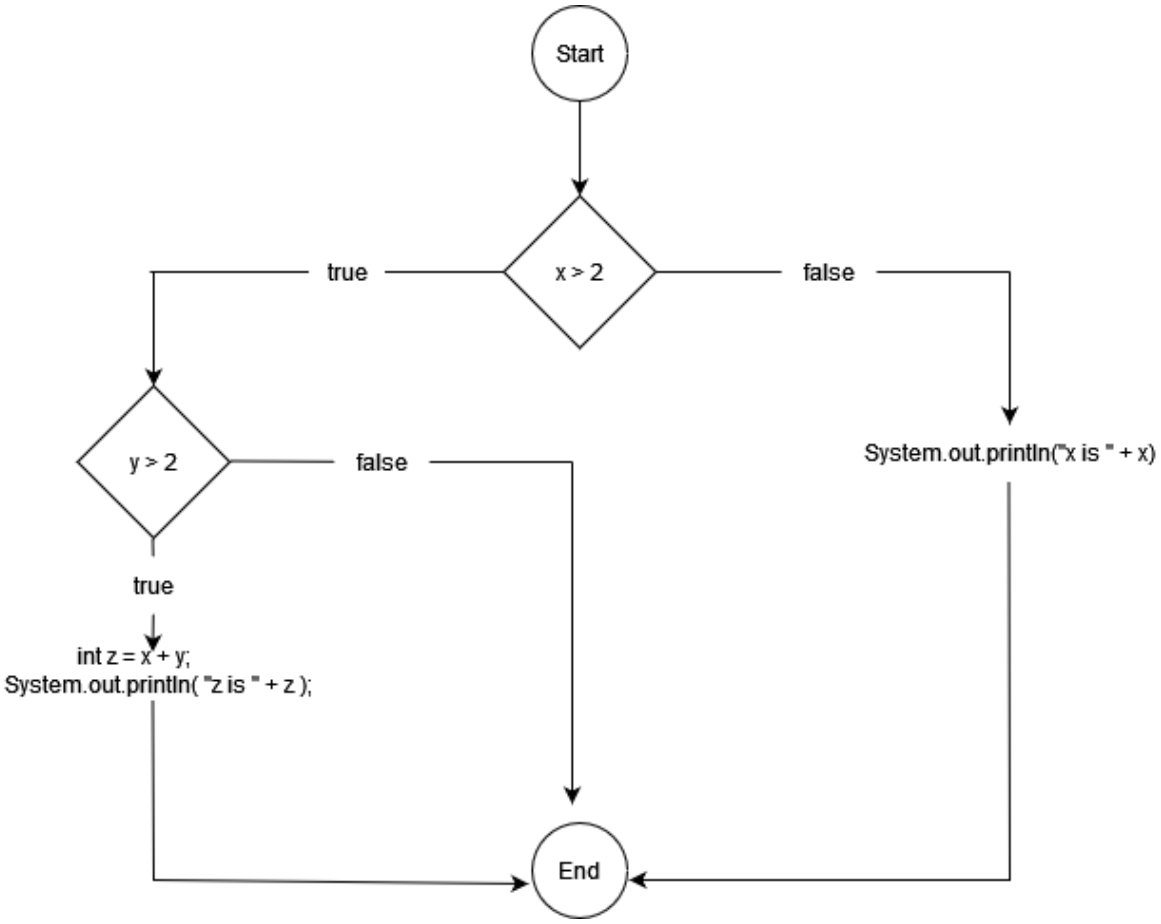
b. Number is 30 just "30 is even." will be printed and number is 35 just "35 is odd." will be printed because of the use of "else" statement.

3.4 Suppose x = 3 and y = 2; show the output, if any, of the following code. What is

the output if x = 3 and y = 4? What is the output if x = 2 and y = 2? Draw a flow chart of the code:

```
if ( x > 2 )
{
        if ( y > 2 )
        {
                int z = x + y;
                System.out.println( "z is " + z );
        }
}
else
        System.out.println("x is " + x);
```

Sol:

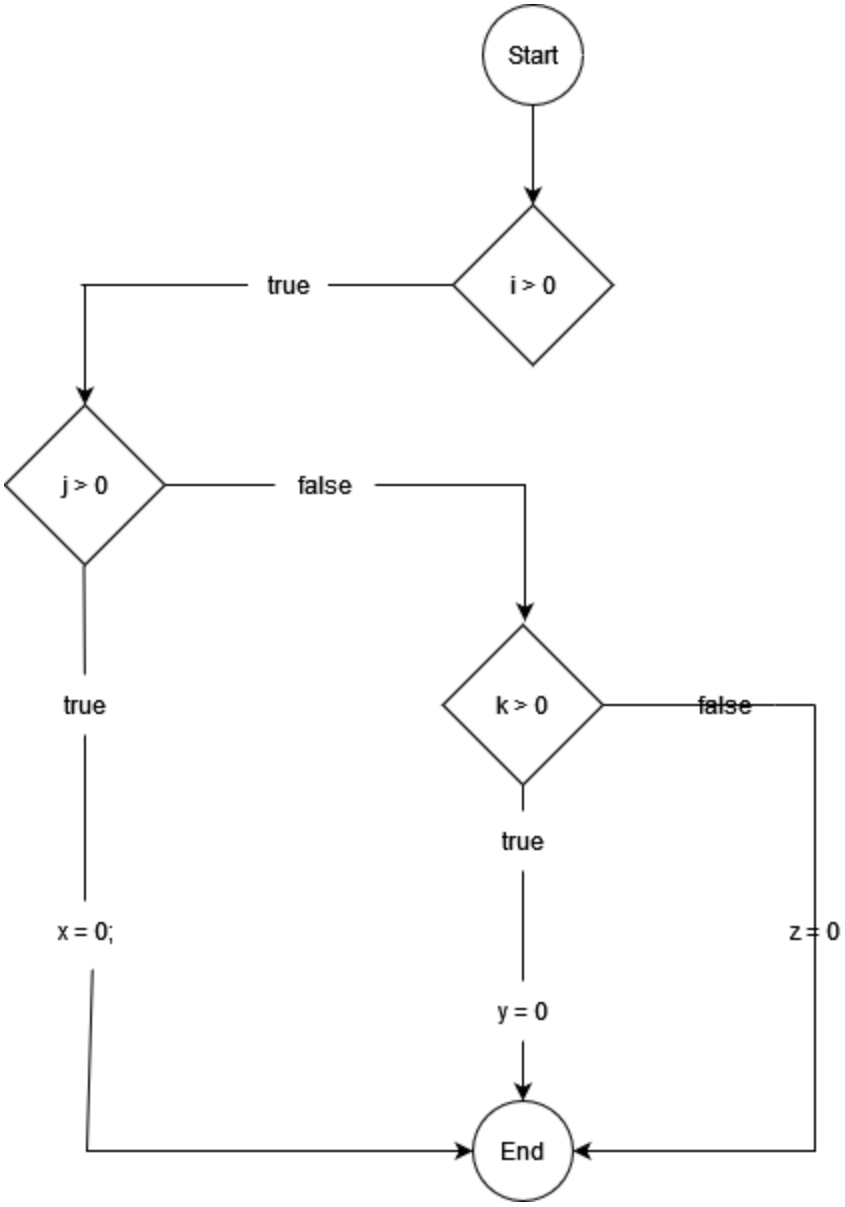


3.5 Which of the following statements are equivalent? Which ones are correctly indented?

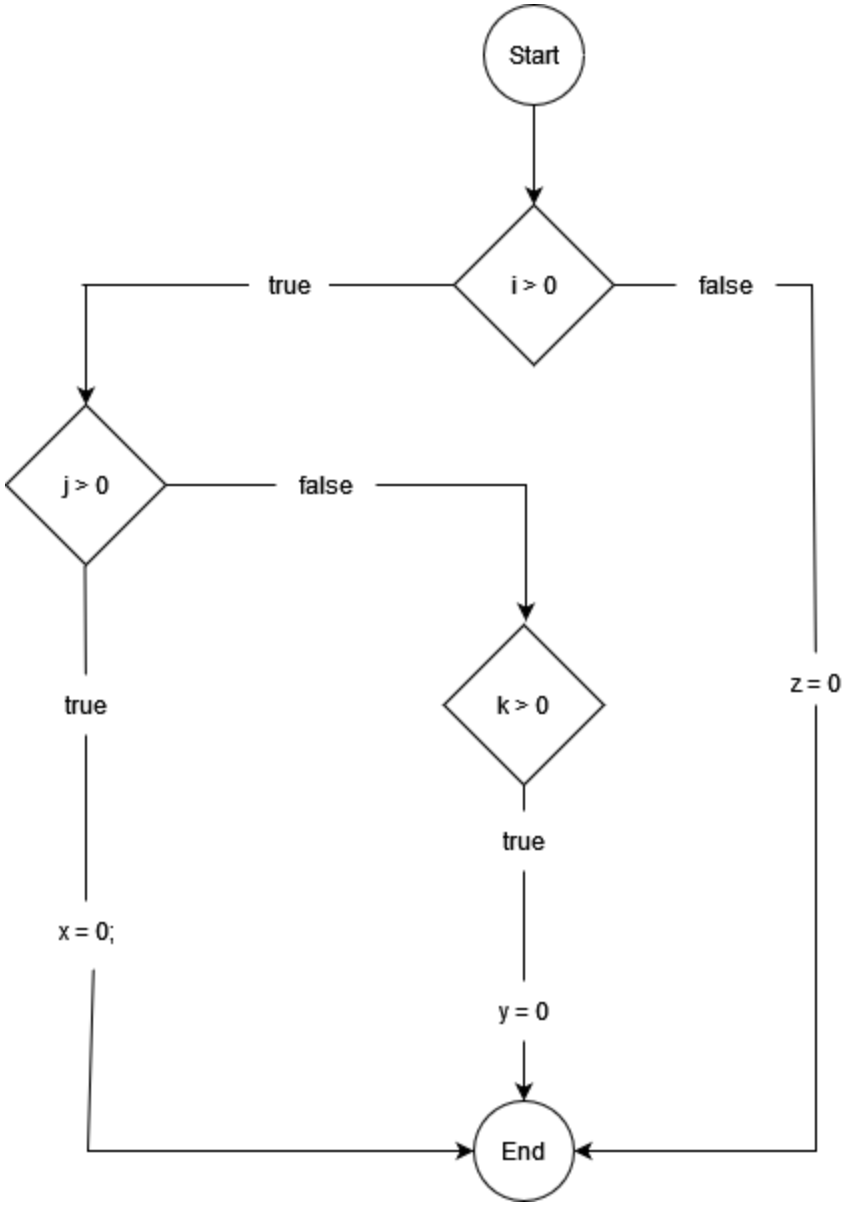
Sol:

a, c and d are equivalent. b and c are correctly indented.

a, c and d's flow chart



b's flow chart



3.6 Suppose x = 2 and y = 3. Show the output, if any, of the following code. What is the output if x = 3 and y = 2? What is the output if x = 3 and y = 3?
(Hint: Indent the statement correctly first.)

```
if ( x > 2 )
        if ( y > 2 )
        {
                int z = x + y;
                System.out.println( "z is " + z );
        }
```

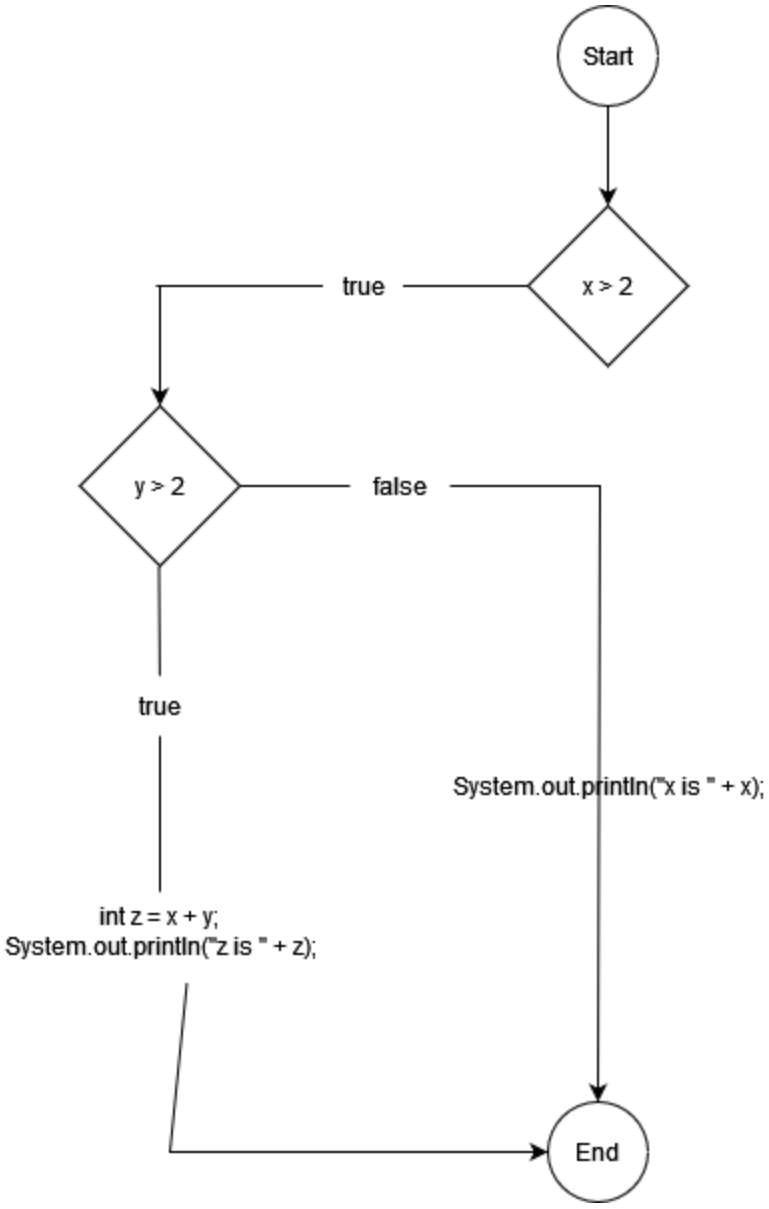
else

System.out.println("x is " + x);

Sol:

	x = 2	x = 3
y = 2		"x is 3"
y = 3	there is no output	"z is 6"

Flowchart



3.7 Are the following two statements equivalent?

Sol:

Yes, second statement's "income > 1000" logical operator usage is dummy because of other two logical operators cover this possibility.

3.8 Which of the following is a possible output from invoking Math.random()?
323.4, 0.5, 34, 1.0, 0.0, 0.234

Sol:

0.5, 0.0, 0.234. Because Math.random() returns between 0.0 (inclusive) and 1.0 (exclusive) double number.

3.9 How do you generate a random integer i such that 0 <= i < 20 ?
How do you generate a random integer i such that 10 <= i < 20 ?
How do you generate a random integer i such that 10 <= i <= 50

Sol:

```
int x = ( int )(Math.random() * 20 );
int y = 10 + ( int )(Math.random() * 10 );
int z = 10 + ( int )(Math.random() * 41 );
```

3.10 Write an if statement that assigns 1 to x if y is greater than 0.

Sol:

```
if ( y > 0 )
        x = 1;
```

3.11 (a) Write an if statement that increases pay by 3% if score is greater than 90.
(b) Write an if statement that increases pay by 3% if score is greater than 90, otherwise increases pay by 1%.

Sol:

a.
```
if ( score > 90 )
        pay *= 1.03;
```

b.
```
if ( score > 90 )
        pay *= 1.03;
else
        pay *= 1.01;
```

3.12 What is wrong in the following code?

```
if (score >= 60.0)
grade = 'D';
else if (score >= 70.0)
grade = 'C';
else if (score >= 80.0)
grade = 'B';
else if (score >= 90.0)
grade = 'A';
else
grade = 'F';
```

Sol:

There is a logical error aka bug in the code because of the wrong sorted statements. If we sort scores descending the code will be work properly.

3.13 Rewrite the following statement using a Boolean expression:

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

Sol:

```
boolean newLine = ( count % 10 == 0 );
```

3.14 Assuming that x is 1, show the result of the following Boolean expressions.

```
(true) && (3 > 4)
!(x > 0) && (x > 0)
(x > 0) || (x < 0)
(x != 0) || (x == 0)
(x >= 0) || (x < 0)
(x != 1) == !(x == 1)
```

Sol:

```
false   ->      (true) && (3 > 4)
false   ->      !(x > 0) && (x > 0)
true    ->      (x > 0) || (x < 0)
true    ->      (x != 0) || (x == 0)
true    ->      (x >= 0) || (x < 0)
true    ->      (x != 1) == !(x == 1)
```

3.15 Write a Boolean expression that evaluates to true if a number stored in variable

num is between 1 and 100.

Sol:

```
boolean result =  ( number < 1 ) && ( number > 1 );
```

3.16 Write a Boolean expression that evaluates to true if a number stored in variable num is between 1 and 100 or the number is negative.

Sol:

```
boolean result = ( ( number  < 1 ) && ( number > 1 ) )
                                                || ( number < 0 );
```

3.17 Assume that x and y are int type. Which of the following are legal Java expressions?

x > y > 0
x = y && y
x /= y
x or y
x and y
(x != 0) || (x = 0)

Sol:

```
illegal ->      x > y > 0
illegal ->      x = y && y
legal   ->      x /= y
illegal  ->     x or y
illegal  ->     x and y
illegal  ->     (x != 0) || (x = 0)
```

3.18 Suppose that x is 1. What is x after the evaluation of the following expression?

(x >= 1) && (x++ > 1)
(x > 1) && (x++ > 1)

Sol:

x will be 3.

3.19 What is the value of the expression ch >= 'A' && ch <= 'Z' if ch is 'A', 'p', 'E', or '5'?

Sol:

ch is 'A' -> true
ch is 'p' -> false
ch is 'E' -> true

ch is '5' -> false

3.20 Suppose, when you run the program, you enter input 2 3 6 from the console. What is the output?

```
public class Test
{
	public static void main(String[] args)
	{
		java.util.Scanner input = new java.util.Scanner(System.in);
		double x = input.nextDouble();
		double y = input.nextDouble();
		double z = input.nextDouble();

		System.out.println("(x < y && y < z) is " + (x < y && y < z));
		System.out.println("(x < y || y < z) is " + (x < y || y < z));
		System.out.println("!(x < y) is " + !(x < y));
		System.out.println("(x + y < z) is " + (x + y < z));
		System.out.println("(x + y < z) is " + (x + y < z));
	}
}
```

Sol:

The output will be;
(x < y && y < z) is true
(x < y || y < z) is true
!(x < y) is false
(x + y < z) is true
(x + y < z) is true

3.21 Write a Boolean expression that evaluates true if age is greater than 13 and less than 18.

Sol:

```
boolean ageLimit = ( age > 13 ) && ( age < 18 );
```

3.22 Write a Boolean expression that evaluates true if weight is greater than 50 or height is greater than 160.

Sol:

```
boolean weightHeightLimit = ( weight > 50 ) || ( height > 160 );
```

3.23 Write a Boolean expression that evaluates true if weight is greater than 50 and height is greater than 160.

Sol:

```
boolean weightHeightLimit = ( weight > 50 ) && ( height > 160 );
```

3.24 Write a Boolean expression that evaluates true if either weight is greater than 50 or height is greater than 160, but not both.

Sol:

```
boolean weightHeightLimit = ( weight > 50 ) ^ ( height > 160 );
```

Section 3.15

3.25 What data types are required for a switch variable? If the keyword break is not used after a case is processed, what is the next statement to be executed? Can you convert a switch statement to an equivalent if statement, or vice versa? What are the advantages of using a switch statement?

Sol:

Char, byte, short, int or string data types are allowed for switch variables.
If a break statement is not used, other executable statements will be executed.
You can always convert a switch statement to an equivalent if statement.
Using switch statements can improve readability sometimes. Additionally, switch statements are more efficient than exactly the same if statement. Because when you use switch the compiler just decides which statement will be completed, not which case will be satisfied.

3.26 What is y after the following switch statement is executed?

```
x = 3; y = 3;
switch (x + 3) {
case 6: y = 1;
default: y += 1;
}
```

Sol:

y will be 2.

3.27 Use a switch statement to rewrite the following if statement and draw the flow chart for the switch statement:

```
if (a == 1)
x += 5;
```

```
else if (a == 2)
x += 10;
else if (a == 3)
x += 16;
else if (a == 4)
x += 34;
```

Sol:

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

Start   ----->  check -> a is 1 -> x += 5; -> break     I
                I
        v
                v ->    check -> a is 2 -> x += 10; -> break   I
                I
        v
                v ->    check -> a is 3 -> x += 16; -> break   I
                I
        I
                v ->    check -> a is 4 -> x += 34;                             v
End             <-----------------------------------------------
```

3.28 Write a switch statement that assigns a String variable dayName with Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, if day is 0, 1, 2, 3, 4, 5, 6, accordingly.

Sol:

```
int dayNumber = 2;
String dayName = "";
```

```
switch( dayNumber )
{
        case 0:
                dayName = "Sunday";
                break;
        case 1:
                dayName = "Monday";
                break;
        case 2:
                dayName = "Tuesday";
                break;
        case 3:
                dayName = "Wednesday";
                break;
        case 4:
                dayName = "Thursday";
                break;
        case 5:
                dayName = "Friday";
                break;
        case 6:
                dayName = "Saturday";
}

System.out.println( dayName );
```

Section 3.16

3.29 Rewrite the following if statement using the conditional operator:

```
if (count % 10 == 0)
System.out.print(count + "\n");
else
System.out.print(count + " ");
```

Sol:

```
int count = 11;
String result = "";

result = ( count % 10 == 0 ) ? count + "\n" :  count + " " ;

System.out.println(result);
```

3.30 Rewrite the following statement using a conditional expression:

```
if (temperature > 90)
pay = pay * 1.5;
else
pay = pay * 1.1;
```

Sol:

```
int temperature = 91;
double pay = 10;

pay = ( temperature > 90 ) ? pay * 1.5 : pay * 1.1;

System.out.println(pay);
```

Section 3.17

3.31 What are the specifiers for outputting a Boolean value, a character, a decimal integer, a floating-point number, and a string?

Sol:

```
boolean -> %b
char    -> %c
integer -> %d
float   -> %f
string  -> %s
```

3.32 What is wrong in the following statements?

(a) System.out.printf("%5d %d", 1, 2, 3);
(b) System.out.printf("%5d %f", 1);
(c) System.out.printf("%5d %f", 1, 2);

Sol:

a. 3 doesn't have any specifier.
b. Too much specifier. There is not enough data.
c. Data for %f must a float value.

3.33 Show the output of the following statements.

(a) System.out.printf("amount is %f %e\n", 32.32, 32.32);
(b) System.out.printf("amount is %5.4f %5.4e\n", 32.32, 32.32);

(c) System.out.printf("%6b\n", (1 > 2));
(d) System.out.printf("%6s\n", "Java");
(e) System.out.printf("%-6b%s\n", (1 > 2), "Java");
(f) System.out.printf("%6b%-s\n", (1 > 2), "Java");

Sol:

a. amount is 32.320000 3.232000e+01
b. amount is 32.3200 3.2320e+01
c. false
d. Java
e. false Java
f. "java.util.MissingFormatWidthException: %-s" error

3.34 How do you create a formatted string?

Sol:

System.out.printf("I am formatted %s", "string");
We can use a desired number between "%" and "s" for limiting character count of string.

Section 3.18

3.35 List the precedence order of the Boolean operators. Evaluate the following expressions:
true || true && false
true && true || false

Sol:

Descending;

!
==,!=
^
&&
||

Both of them true.

3.36 True or false? All the binary operators except = are left associative.

Sol:

True

3.37 Evaluate the following expressions:
2 * 2 - 3 > 2 && 4 - 2 > 5
2 * 2 - 3 > 2 || 4 - 2 > 5

Sol:

False
False

3.38 a. Is (x > 0 && x < 10) the same as ((x > 0) && (x < 10))?

b. Is (x > 0 || x < 10) the same as ((x > 0) || (x < 10))?

c. Is (x > 0 || x < 10 && y < 0) the same as (x > 0 || (x < 10 && y < 0))?

Sol:

a. Yes both of them same because according to operator precedence comparison operators execute before logical operators.

b. Yes both of them same because according to operator precedence comparison operators execute before logical operators.

c. Yes both of them same because according to operator precedence comparison operators execute before logical operators.

Section 3.19

3.39 How do you display a confirmation dialog? What value is returned when invoking JOptionPane.showConfirmDialog?

Sol:

We can use this statement for the display confirmation dialog;
int option = JOptionPane.showConfirmDialog(null, "String");

Method returns an integer data type option value. The value is JOptionPane.YES_OPTION (0) for the Yes button, JOptionPane.NO_OPTION (1) for the No button, and JOptionPane.CANCEL_OPTION (2) for the Cancel button.