Lab 04
Java Operators and Expressions

Submission Details

In this lab, you are to write three Java programs to solve the given problems in the Lab Exercises section. The first one will be done with sample solution disclosed during the lab session. For the other **TWO** programs, you need to submit their source files named **ChangeMaker.java** and **QuadraticSolver.java** to the assignment page of Canvas. The deadline is **ONE week** after your lab session is conducted. Please verify your solution to ensure that everything is correct before submission.

Note: You are required to write down your name, student id, and lab section id in a comment block at the beginning of your source file.

Objectives

The objectives of this lab are to reinforce your programming concepts taught in last week, including (1) variable declaration and primitive data types; (2) standard Java I/O; and (3) various operators in Java.

Software Required

1. Java Development Kit, e.g. Java SE 8
2. An Integrated Development Environment (IDE), e.g. Eclipse or NetBeans

Please start your lab PC with Windows 10 and install your preferred IDE via WorkDesk.

Background

Java Summary of Operators
Here is a quick reference of the operators supported by the Java programming language.

Simple Assignment Operator

=	Simple assignment operator

Arithmetic Operators

+	Additive operator (also used for String concatenation)
-	Subtraction operator
*	Multiplication operator
/	Division operator
%	Modulo operator (or Remainder operator)

Unary Operators

+	Unary plus operator; indicates positive value (numbers are positive without this, however)
-	Unary minus operator; negates an expression
++	Increment operator; increments a value by 1
--	Decrement operator; decrements a value by 1

!	Logical complement operator; inverts the value of a boolean

Equality and Relational Operators

==	Equal to
!=	Not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

Conditional Operators

&&	Conditional-AND
||	Conditional-OR
? :	Ternary (shorthand for if-then-else statement)

Bitwise and Bit Shift Operators (*Out of Our Syllabus*)

~	Unary bitwise complement
<<	Signed left shift
>>	Signed right shift
>>>	Unsigned right shift
&	Bitwise AND
^	Bitwise exclusive OR
|	Bitwise inclusive OR

Java also provides shorthand assignment operators to simplify coding. For example,

- `x += 10;       // Equivalent to x = x + 10;`
- `x -= 10;       // Equivalent to x = x – 10;`
- `x *= 10 + 2;   // Equivalent to x = x * (10+2);`
- `x /= 10;       // Equivalent to x = x / 10;`
- `x %= 10;       // Equivalent to x = x % 10;`
- `x &&= true;    // Equivalent to x = x && true;`
- `x ||= true;    // Equivalent to x = x || true;`

Java has well-defined rules for specifying the order in which the operators in an expression are evaluated when the expression has several operators. For example, multiplication and division have a higher precedence than addition and subtraction. Precedence rules can be overridden by explicit parentheses.

Precedence order
When two operators share an operand the operator with the higher *precedence* goes first. For example, 1 + 2 * 3 is treated as 1 + (2 * 3), whereas 1 * 2 + 3 is treated as (1 * 2) + 3 since multiplication has a higher precedence than addition.

Associativity
When an expression has two operators with the same precedence, the expression is evaluated according to its *associativity*. For example, `x = y = z = 17` is treated as `x = (y = (z = 17))`, leaving all three variables with the value 17, since the = operator has right-to-left associativity (and an assignment statement evaluates to the value on the right-hand side). On the other hand, `72 / 2 / 3` is treated as `(72 / 2) / 3` since the / operator has left-to-right associativity. Some operators are not associative: for example, the expressions `(x <= y <= z)` and `x++--` or `(x++)--` are invalid.

Precedence and associativity of Java operators
The table below shows all Java operators from highest to lowest precedence, along with their associativity. Most programmers do not memorize them all, and even those that do still **use parentheses** for clarity.

Level	Operator	Description	Associativity

16	`[]`
	`.`
	`()`
		access array element
		access object member
		parentheses
			left to right

15	`++` `--`	unary post-increment unary post-decrement	not associative

14	`++`
	`--`
	`+`
	`-`
	`!`
	`~`
		unary pre-increment
		unary pre-decrement
		unary plus
		unary minus
		unary logical NOT
		unary bitwise NOT
			right to left

13	`()`
	`new`
		cast
		object creation
			right to left

12	`* / %`	multiplicative	left to right

11	`+ -`
	`+`
		additive
		string concatenation
			left to right

10	`<< >>`
	`>>>`
		shift	left to right

9	`< <=`
	`> >=`
	`instanceof`
		relational	not associative

8	`==` `!=`	equality	left to right
7	`&`	bitwise AND	left to right
6	`^`	bitwise XOR	left to right
5	`|`	bitwise OR	left to right
4	`&&`	logical AND	left to right
3	`||`	logical OR	left to right
2	`?:`	ternary	right to left
1	`=` `+=` `-=` `*=` `/=` `%=` `&=` `^=` `|=` `<<=` `>>=` `>>>=`	assignment	right to left

There is no explicit operator precedence table in the Java Language Specification. Different tables on the web and in textbooks disagree in some minor ways.

(Source: http://introcs.cs.princeton.edu/java/11precedence/)

Prelab Questions (No need to submit)

Attempt these questions without using any computers. Dry run please!

1. Write down the output next to each of the println() statements.

```
class PrePostDemo1 {
    public static void main(String[] args){
        int i = 3;
        i++;
        System.out.println(i);      ◄──── Output:
        ++i;
        System.out.println(i);      ◄──── Output:
        System.out.println(++i);    ◄──── Output:
        System.out.println(i++);    ◄──── Output:
        System.out.println(i);      ◄──── Output:
    }
}
```

2. Write down the output next to each of the println() statements.

```
class PrePostDemo2 {
    public static void main (String[] args){
        // increment and decrement operators
        int x = 5;
        int y = 10;
        int z = ++x * y--;
        System.out.println("z = " + z);        ← Output:

        int i = 10;
        int n = i++ % 5;
        System.out.printf("i = %d, n = %d\n", i, n);
                                                ← Output:
        i = 10;
        n = ++i % 5;                            ← Output:
        System.out.printf("i = %d, n = %d\n", i, n);
    }
}
```

remember \n is the "new line" character

format string

argument list

printf() is yet another method (in C style) that you can use to print data to the console.
The first half is called the **format string** (%d is a **format specifier** for printing a signed decimal integer; see this for more format specifiers if you like). The second half is the **argument list**. The output is much like getting each argument's value substituted into the placeholder marked with %d in sequential order. If you don't feel comfortable with printf(), the above statement can be rewritten as:
`System.out.println("i = " + i + ", n = " + n); // as usual`

3. Write down the output next to each of the println() statements.

```
class ShortcutAssignmentDemo {
    public static void main (String[] args){
        int result = 3;
        System.out.println(result);        ← Output:

        result -= 1;
        System.out.println(result);        ← Output:

        result *= 2;
        System.out.println(result);        ← Output:

        result /= 2;
        System.out.println(result);        ← Output:

        result += 8;
        result %= 7;
        System.out.println(result);        ← Output:
    }
}
```

4. Write down the output next to each of the println() statements.

```
class PlusDemo {
    public static void main(String[] args){
        System.out.println("1 + 2 = " + 1 + 2);        ← Output:
        System.out.println("1 + 2 = " + (1 + 2));      ← Output:
    }
}
```

5. Write down the output next to each of the println() statements.

```
class BooleanDemo {
    public static void main (String[] args){
        boolean b = true;
        b = !b;
        System.out.println(b);              ← Output:

        // relational operators
        b = (1 == 2);
        System.out.println(b);              ← Output:

        b = (1 != 2);
        System.out.println(b);              ← Output:

        b = (1 < 2);
        System.out.println(b);              ← Output:

        b = (1 >= 1);
        System.out.println(b);              ← Output:

        // logical operators
        b = (2 >= 1 && 4 <= 3);      // && means AND
        System.out.println(b);              ← Output:

        b = (2 >= 1 || 4 <= 3);      // || means OR
        System.out.println(b);              ← Output:

        int year = 2020;
        boolean leap = year % 4 == 0 && year % 100 != 0 || year % 400 == 0;
        System.out.println(leap);           ← Output:
    }
}
```

By the way, using the space provided below, can you rewrite the above expression for computing the leap variable by adding parentheses to make the order of evaluation more clear?

```
boolean leap =
```

Lab Exercises

In this lab, you will be asked to solve **THREE** problems using Java. For these problems, write their solution as Java programs (one .java file for each) based on the following problem specifications.

Question 1 will be done during the lab session and you need not submit answer for Question 1.

1. Write a Java application called **DigitSeparator** that inputs an integer consisting of **five** digits from the user, separates the input integer into its individual digits, and prints the digits separated from one another by three spaces each. For example, if the user types in 12345, the program should print "1   2   3   4   5". Assume that the user enters the correct number of digits.

   Sample Output:

   ```
   Enter a five-digit integer: 12345
   Digits in 12345 are 1   2   3   4   5
   ```

   (Note: The underlined text refers to the user input. It is NOT meaning that your program needs to underline it. Please also follow closely to the output format as shown above.)

   Problem-solving tips

   1. The input data consists of one integer, so you will use an `int` variable to represent it. Note that the description indicates that one five-digit number is to be input – not five separate digits.

   2. You will use a series of statements to "break down" the number into its individual digits, using integer arithmetic with remainder (%) and division (/) calculations.

   3. After the number is entered, divide it by 10000 to get the first digit. Why does this operation work? In Java, dividing an integer by an integer yields an integer result. Because the number input is five digits long, dividing it by 10000 gives the leftmost digit. For example, 42339 / 10000 evaluates to 4 because 10000 divides into 42339 four times. The remainder is truncated in integer arithmetic.

   4. Change the number to a four-digit number using the mod operator which returns the remainder after the number is divided by 10000 – in this case, the rightmost four digits. For example, 42339 % 10000 results in 2239.

   5. Repeat this pattern of division and remainder calculations. Each time, the number used in the division and remainder calculations is reduced by a factor of 10.

2. Vending machines often need to return the change in coins to the buyer and embed a small program to control their operation. Their I/O subsystem may look different from that of a usual computer, but this is not the point. In this question, we are to simulate one part of a vending machine on a computer – how it calculates the number of coins per face value for a specific amount of change in cents to return. The input and output will be performed via the keyboard and screen.

   Write a Java application called **ChangeMaker**. The user of this program enters an amount of change from 1 to 99 cents. The program responds by telling the user one combination of coins that equals that amount of change. We adopt the US currency system as follows:

Face value	Coin's name
1¢	penny
5¢	nickel
10¢	dime
25¢	quarter

(Note: In fact, there are 50¢ coins called half dollars in US, but we will neglect it.)

Sample Output:

```
Enter a whole number from 1 to 99: 87
One combination of coins that equals to 87 cents is:
3 quarters
1 dimes
0 nickels
2 pennies
```

We know the computed numbers 3, 1, 0, 2 are correct because mapping with the above table, **3** × 25¢ + **1** × 10¢ + **0** × 5¢ + **2** × 1¢ = 87¢.

(Note: The underlined text refers to the user input. It is NOT meaning that your program needs to underline it. Please also follow closely to the output format as shown above. The output above does not exactly use correct grammar. The labels should show "1 dime" instead of "1 dimes". Please ignore this presentation problem.)

3. A quadratic equation is an equation defined as a polynomial of degree two, it could be represented by: $ax^2 + bx + c = 0$, where $a \neq 0$, b, c are called coefficient of the equation and x is the unknown value that we wish to find. To solve the quadratic equation, we can apply the following formula:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Write a Java program named **QuadraticSolver** to compute the solution(s) of a user-specified quadratic equation. The program should first welcome the user by printing a welcome banner:

```
Welcome to Using Quadratic Solver
=================================
```

Starting on the next line, the program should prompt the user to input the coefficients of the quadratic equation: a, b and c. It will prompt the user by printing:

```
Please enter the value of a: 1
Please enter the value of b: 3.5
Please enter the value of c: 2
```

The user will input the corresponding value after the print line.

Assumptions:

1. Notice that the number entered by the user can be assumed to be correct (this lab does not require input validation). Therefore, you can simply assume that user won't input some invalid numbers, like "a", "abc", "-12-0.-2", etc.
2. The value b^2 - 4ac is always greater than or equal to 0, i.e. negative value doesn't appear.

Your program should print the solutions as follows.

Sample Output:

A session of inputs and outputs:

```
Welcome to Using Quadratic Solver
=================================
Please enter the value of a: 1
Please enter the value of b: 3.5
Please enter the value of c: 2
x1 = -0.7192235935955849
x2 = -2.7807764064044154
```

Another session of inputs and outputs:

```
Welcome to Using Quadratic Solver
=================================
Please enter the value of a: 1
Please enter the value of b: 2
Please enter the value of c: 1
x1 = -1.0
x2 = -1.0
```

(Note: The underlined text refers to the user input. It is NOT meaning that your program needs to underline it. Please also follow closely to the output format as shown above.)

Marking Scheme

The marking of this exercise will be based on the following criteria.

Graded items	Weighting
1. Correctness of program (i.e. whether your code is implemented in a way according to the requirements as specified.)	70%
2. Presentation of the Java codes (i.e. whether the program is properly indented, how close you follow the common conventions as mentioned in class, etc.)	15%
3. Documentation (with reasonable amount of comments embedded in the code to enhance the readability)	15%
	100%

Be aware of plagiarism! DON'T copy the program file from your friends or classmates. If any identical copy is found, 5% of the coursework marks will be deducted for each of the file owner.

Program Submission Checklist

Before submitting your work, please check the following items to see you have done a decent job.

Items to be checked	☑
1. Did I put my name, student id and lab section id at the beginning of all the source files as comment?	☐
2. Did I put a reasonable amount of comments to describe my program?	☐
3. Are they all in .java extension and named according to the specification?	☐
4. Have I checked that all the submitted code can compile and run without any errors?	☐
5. Did I zip my source files using Winzip / zip provided by Microsoft Windows? Also, did I check the zip file and see if it could be opened? ***(Only applicable if the work has to be submitted in zip format.)***	☐
6. Did I submit my lab assignment to Canvas before the deadline?	☐

(This checklist need not be submitted.)