A. **OBJECTIVES.** After the end of the session students should be able to understand Java's programming fundamentals such as:

- Java Coding Guidelines
- Java Data types
- Variable Declaration
- Java Operators such as:
  - Arithmetic Operators
  - Relational Operators
  - Logical Operators
  - Conditional Operators
- Increment and Decrement Operators
- Relational Operators
- Logical Operators
- Operator Precedence: Coding Guidelines

Coding Guidelines

1. Always start your Java program by creating a PROJECT
2. Next inside the Project you will create a CLASS
3. Reminder: Project Name and Class Name should not contain spaces
4. You should write comments in your code explaining what a certain class does, or what a certain method do.

Data Types (Primitive)

● The Java programming language defines eight primitive data types.

Type		Description
1	**boolean** (for logical)	A Boolean data type represents two states: true and false.
2	**char** (for textual)	• A character data type (char), represents a single Unicode character • It must have its literal enclosed in single quotes (' ').
3	**String**	• A data type that contains multiple characters. • It is not a primitive data type, it is a class • It has its literal enclosed in double quotes (" ").
4	**int**	• Numeric Type • Does not contain decimal place
5	**double**	• Numeric Type • Contains decimal place
6	**float (floating point)**	Floating point types has double as default data type.
7	**long (integral)**	• Integral data types in Java uses three forms – decimal, octal, or hex • Integral types has **int** as default data type

8	**byte(integral)**	
9	**short(integral)**	

Variables

- A variable is an item of data used to store the state of objects.
- A variable has a:

  Data type = the data type indicates the type of value that the variable can hold.
  Name = the variable name must follow rules for identifiers.

Declaring and Initializing Variables

Declare one variable per line of code. For example, the variable declarations,

```
double exam=50;
double quiz=10;
int grade = 0;
```

is preferred over the declaration,

```
double exam=50, quiz=10, grade=0;
```

Outputting Variable Data:

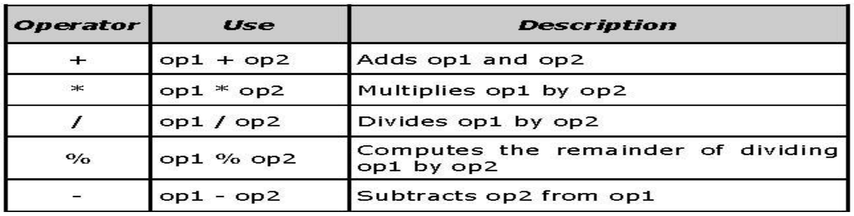
```
System.out.println( exam );
System.out.println( “The value of x= ” + x );
```

Operators

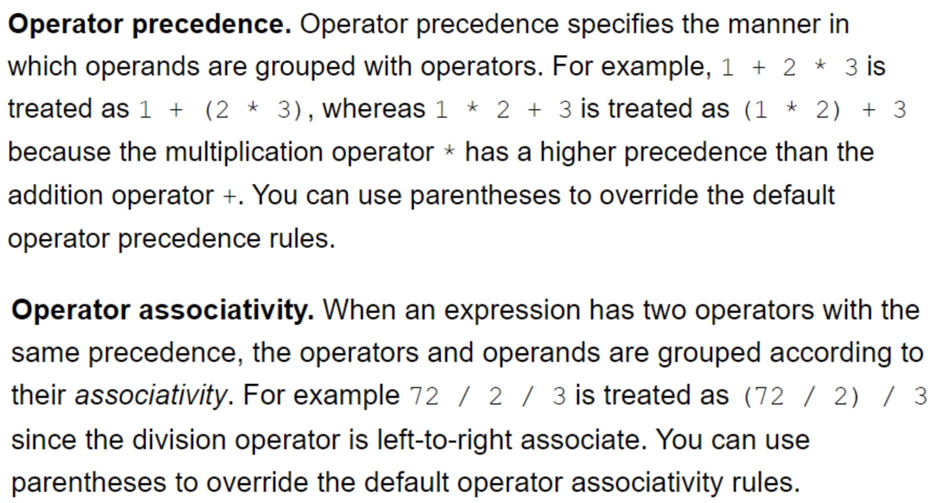
- Different types of operators:
  - Arithmetic operators
  - Relational operators
  - Logical operators
  - Conditional operators

- These operators follow a certain kind of **precedence** so that the compiler will know which operator to evaluate first in case multiple operators are used in one statement.

Arithmetic Operators



Operator	*Use*	*Description*
+	op1 + op2	Adds op1 and op2
*	op1 * op2	Multiplies op1 by op2
/	op1 / op2	Divides op1 by op2
%	op1 % op2	Computes the remainder of dividing op1 by op2
-	op1 - op2	Subtracts op2 from op1

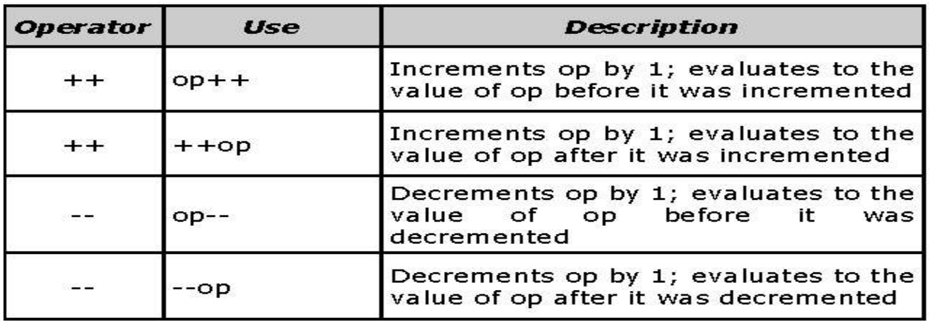


Operator precedence. Operator precedence specifies the manner in which operands are grouped with operators. For example, `1 + 2 * 3` is treated as `1 + (2 * 3)`, whereas `1 * 2 + 3` is treated as `(1 * 2) + 3` because the multiplication operator `*` has a higher precedence than the addition operator `+`. You can use parentheses to override the default operator precedence rules.

Operator associativity. When an expression has two operators with the same precedence, the operators and operands are grouped according to their *associativity*. For example `72 / 2 / 3` is treated as `(72 / 2) / 3` since the division operator is left-to-right associate. You can use parentheses to override the default operator associativity rules.

Increment and Decrement Operators

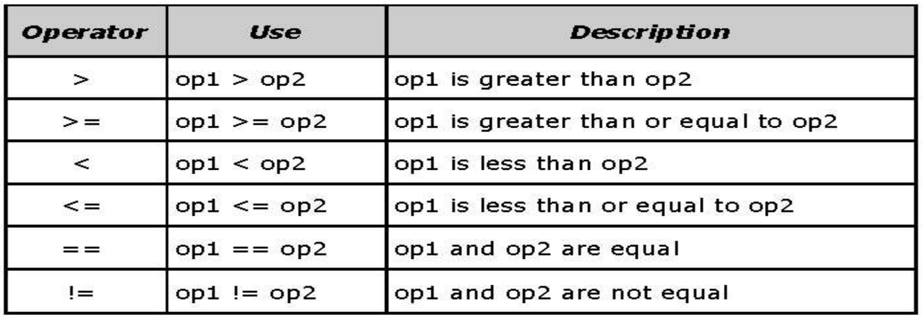
- ● unary increment operator (++)
- ● unary decrement operator (--)
- ● Increment and decrement operators increase and decrease a value stored in a number variable by 1.



Operator	Use	Description
++	op++	Increments op by 1; evaluates to the value of op before it was incremented
++	++op	Increments op by 1; evaluates to the value of op after it was incremented
--	op--	Decrements op by 1; evaluates to the value of op before it was decremented
--	--op	Decrements op by 1; evaluates to the value of op after it was decremented

Relational Operators

- ● Relational operators compare two values and determines the relationship between those values.
- ● The output of evaluation are the boolean values true or false.



Operator	Use	Description
>	op1 > op2	op1 is greater than op2
>=	op1 >= op2	op1 is greater than or equal to op2
<	op1 < op2	op1 is less than op2
<=	op1 <= op2	op1 is less than or equal to op2
==	op1 == op2	op1 and op2 are equal
!=	op1 != op2	op1 and op2 are not equal

Logical Operators

- Logical operators have one or two boolean operands that yield a boolean result.
- There are six logical operators:
  - && (logical AND)
  - & (boolean logical AND)
  - || (logical OR)
  - | (boolean logical inclusive OR)
  - ^ (boolean logical exclusive OR)
  - ! (logical NOT)

Operator Precedence: Coding Guidelines

- To avoid confusion in evaluating mathematical operations, keep your expressions simple and use parentheses.

B. SAMPLE MACHINE PROBLEM #2

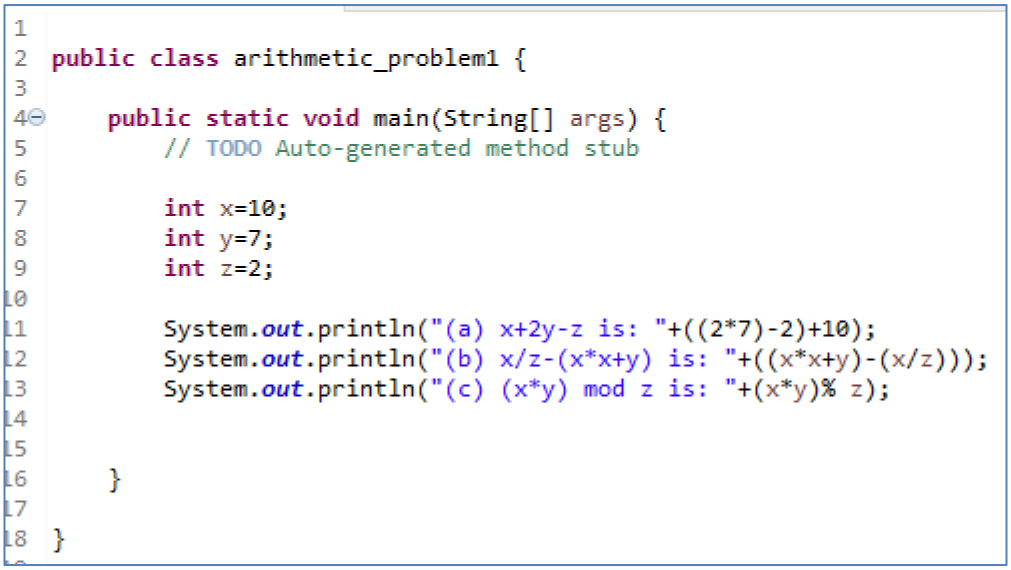
Project Name: machine_problem2

Class Name: example1

1. Given integer variables x,y and z with values 10,7,2 respectively. Determine the value of each of the following arithmetic expressions:

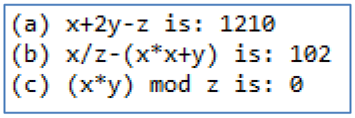
   a. x+2y-z (b) x/z-(x*x+y) (c) (x*y) mod z

Sample Code



```java

public class arithmetic_problem1 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        int x=10;
        int y=7;
        int z=2;

        System.out.println("(a) x+2y-z is: "+((2*7)-2)+10);
        System.out.println("(b) x/z-(x*x+y) is: "+((x*x+y)-(x/z)));
        System.out.println("(c) (x*y) mod z is: "+(x*y)% z);


    }

}
```

Sample output



```
(a) x+2y-z is: 1210
(b) x/z-(x*x+y) is: 102
(c) (x*y) mod z is: 0
```

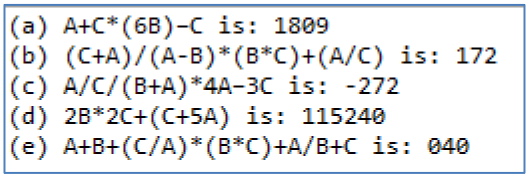
C. CHALLENGE MACHINE PROBLEM #2

Project Name: machine_problem2
Class Name: challenge2

1. Given integer variables A, B and C with values 22,12,5 respectively. Determine the value of each of the following arithmetic expressions: **Use MDAS rule.**

    a. A + C * (6B) – C
    b. (C + A) / (A-B) * (B*C) + (A / C)
    c. A / C / (B + A) * 4A – 3C
    d. 2B * 2C + (C + 5A)
    e. A + B + (C / A) * (B * C) + A / B +C

Answer:



```
(a) A+C*(6B)-C is: 1809
(b) (C+A)/(A-B)*(B*C)+(A/C) is: 172
(c) A/C/(B+A)*4A-3C is: -272
(d) 2B*2C+(C+5A) is: 115240
(e) A+B+(C/A)*(B*C)+A/B+C is: 040
```