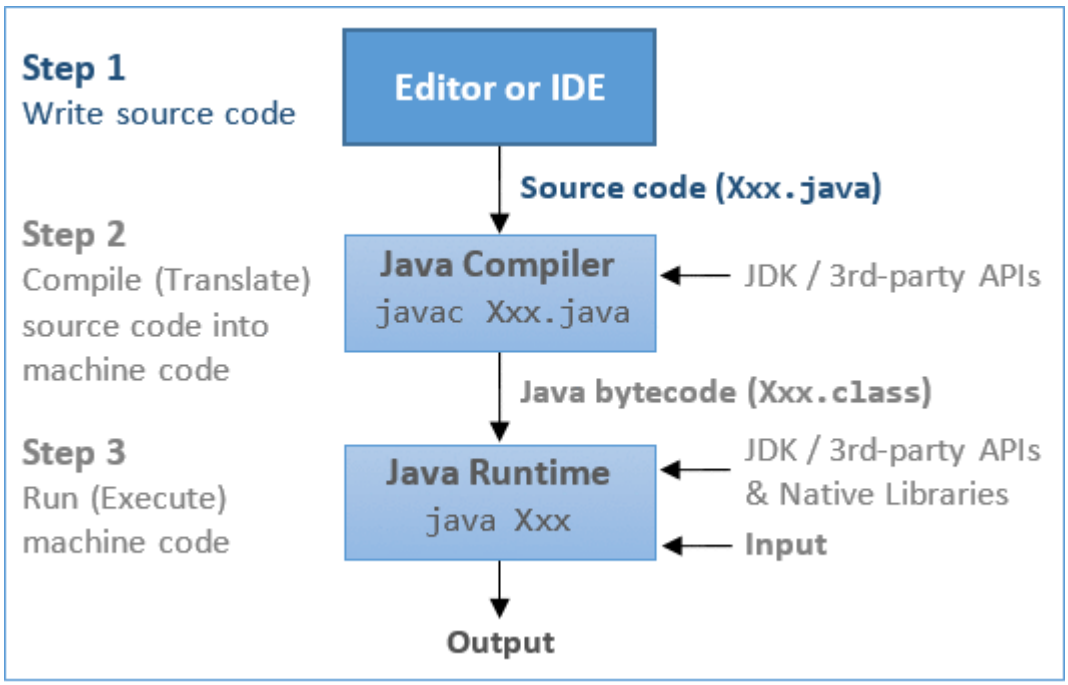
EXPERIMENT NO.-01

1. Basic Syntaxes

1.1 Steps in Writing a Java Program

The steps in writing a Java program is illustrated as follows:



Step 1: Write the source code Xxx.java using a programming text editor (such as Sublime Text, Atom, Notepad++, Textpad, gEdit) or an IDE (such as Eclipse or NetBeans).
Step 2: Compile the source code Xxx.java into Java portable bytecode Xxx.class using the JDK Compiler by issuing command:

```
javac Xxx.java
```

Step 3: Run the compiled bytecode Xxx.class with the input to produce the desired output, using the Java Runtime by issuing command:

```
java Xxx
```

1.2 Java Program Template

You can use the following *template* to write your Java programs. Choose a meaningful "*Classname*" that reflects the *purpose* of your program, and write your programming statements inside the body of the main() method. Don't worry about the other terms and keywords now. I will explain them in due course. Provide comments in your program!

Arithmetic:
Arithmetic Operators: They are used to perform simple arithmetic operations on primitive data types.

- *** :** Multiplication
- **/ :** Division
- **% :** Modulo
- **+ :** Addition
- **– :** Subtraction

```
// Java program to illustrate
// arithmetic operators
public class operators {
    public static void main(String[] args)
    {
        int a = 20, b = 10, c = 0, d = 20, e = 40, f = 30;
        String x = "Thank", y = "You";

        // + and - operator
        System.out.println("a + b = " + (a + b));
        System.out.println("a - b = " + (a - b));

        // + operator if used with strings
        // concatenates the given strings.
        System.out.println("x + y = " + x + y);

        // * and / operator
        System.out.println("a * b = " + (a * b));
        System.out.println("a / b = " + (a / b));

        // modulo operator gives remainder
        // on dividing first operand with second
        System.out.println("a % b = " + (a % b));

        // if denominator is 0 in division
        // then Arithmetic exception is thrown.
        // uncommenting below line would throw
        // an exception
        // System.out.println(a/c);
    }
}
```

Relational Operators : These operators are used to check for relations like equality, greater than, less than. They return boolean result after the comparison and are extensively used in looping statements as well as conditional if else statements. General format is,
variable **relation_operator** value
Some of the relational operators are-

- **==, Equal to :** returns true of left hand side is equal to right hand side.
- **!=, Not Equal to :** returns true of left hand side is not equal to right hand side.
- **<, less than :** returns true of left hand side is less than right hand side.
- **<=, less than or equal to :** returns true of left hand side is less than or equal to right hand side.
- **>, Greater than :** returns true of left hand side is greater than right hand side.
- **>=, Greater than or equal to:** returns true of left hand side is greater than or equal to right hand side.

```
// Java program to illustrate
// relational operators
public class operators {
    public static void main(String[] args)
```

```
    {
        int a = 20, b = 10;
        String x = "Thank", y = "Thank";
        int ar[] = { 1, 2, 3 };
        int br[] = { 1, 2, 3 };
        boolean condition = true;

        // various conditional operators
        System.out.println("a == b :" + (a == b));
        System.out.println("a < b :" + (a < b));
        System.out.println("a <= b :" + (a <= b));
        System.out.println("a > b :" + (a > b));
        System.out.println("a >= b :" + (a >= b));
        System.out.println("a != b :" + (a != b));

        // Arrays cannot be compared with
        // relational operators because objects
        // store references not the value
        System.out.println("x == y : " + (ar == br));

        System.out.println("condition==true :"
                           + (condition == true));
    }
}
```

Logical Operators : These operators are used to perform "logical AND" and "logical OR" operation, i.e. the function similar to AND gate and OR gate in digital electronics. One thing to keep in mind is the second condition is not evaluated if the first one is false, i.e. it has a short-circuiting effect. Used extensively to test for several conditions for making a decision. Conditional operators are-

- **&&, Logical AND :** returns true when both conditions are true.
- **||, Logical OR :** returns true if at least one condition is true.

Java Special Operators:

Java Unary Operator

The Java unary operators require only one operand. Unary operators are used to perform various operations i.e.:

- incrementing/decrementing a value by one
- negating an expression
- inverting the value of a boolean

Java Unary Operator Example: ++ and --

```
1. class OperatorExample{
2. public static void main(String args[]){
3. int x=10;
```

```
4. System.out.println(x++);//10 (11)
5. System.out.println(++x);//12
6. System.out.println(x--);//12 (11)
7. System.out.println(--x);//10
8. }}
```

Output:

```
10
12
12
10
```

Java AND Operator Example: Logical && and Bitwise &

The logical && operator doesn't check second condition if first condition is false. It checks second condition only if first one is true.

The bitwise & operator always checks both conditions whether first condition is true or false.

Java AND Operator Example: Logical && vs Bitwise &

```
1. class OperatorExample{
2. public static void main(String args[]){
3. int a=10;
4. int b=5;
5. int c=20;
6. System.out.println(a<b&&a++<c);//false && true = false
7. System.out.println(a);//10 because second condition is not checked
8. System.out.println(a<b&a++<c);//false && true = false
9. System.out.println(a);//11 because second condition is checked
10. }}
```

Output:

```
false
10
false
11
```

Java OR Operator Example: Logical || and Bitwise |

The logical || operator doesn't check second condition if first condition is true. It checks second condition only if first one is false.

The bitwise | operator always checks both conditions whether first condition is true or false.

```
1. class OperatorExample{
2. public static void main(String args[]){
3. int a=10;
4. int b=5;
5. int c=20;
6. System.out.println(a>b||a<c);//true || true = true
7. System.out.println(a>b|a<c);//true | true = true
8. //|| vs |
9. System.out.println(a>b||a++<c);//true || true = true
10. System.out.println(a);//10 because second condition is not checked
11. System.out.println(a>b|a++<c);//true | true = true
12. System.out.println(a);//11 because second condition is checked
13. }}
```

Output:

```
true
true
true
10
true
```

Java Assignment Operator

Java assignment operator is one of the most common operator. It is used to assign the value on its right to the operand on its left.

Java Assignment Operator Example

```
1. class OperatorExample{
2. public static void main(String[] args){
3. int a=10;
4. a+=3;//10+3
5. System.out.println(a);
6. a-=4;//13-4
7. System.out.println(a);
8. a*=2;//9*2
9. System.out.println(a);
10. a/=2;//18/2
11. System.out.println(a);
12. }}
```

Output:

```
13
9
18
```