



National University of Computer & Emerging Sciences,  
Karachi



Computer Science Department  
Spring 2023, Lab Manual – 02

Course Code: CL-1004	Course : Object Oriented Programming Lab
Instructor(s) :	Abeer Gauher, Hajra Ahmed, Shafique Rehman

## **LAB - 2**

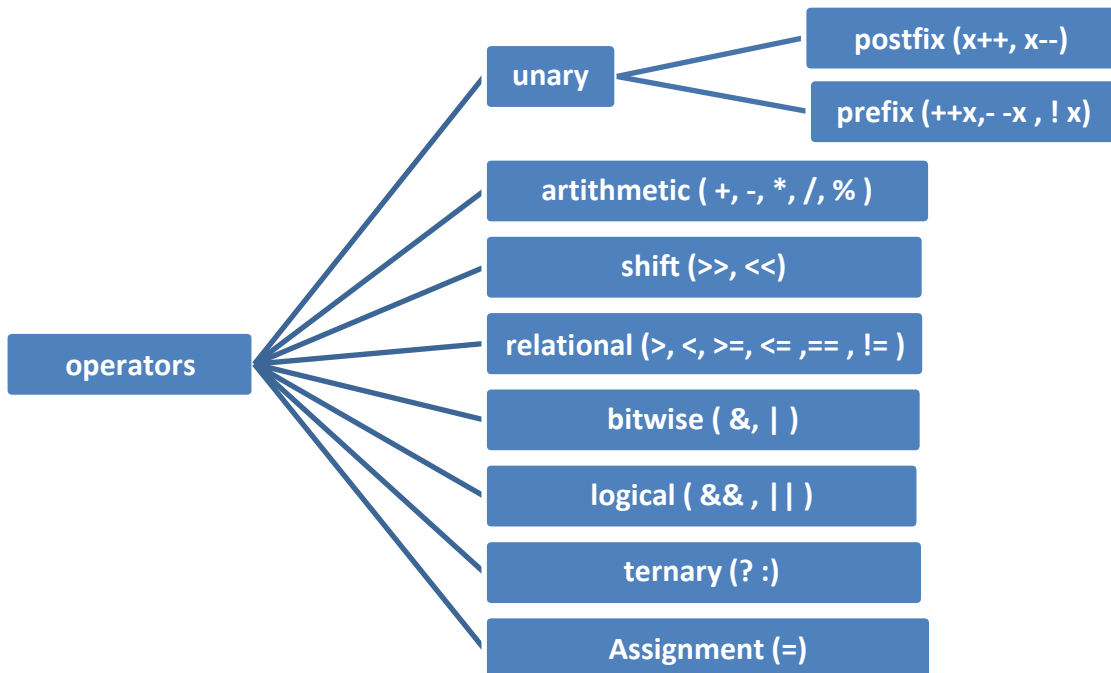
# **Operators, Control Structures, loops & Math Library Functions in Java**

## CONTENTS:

- Operators
- Control structures.
  - If statements
  - If else statements
  - If else if statements
  - For loop
  - While loop
  - Do while loop
- Java Math class

### Operators

Operators are classified into different types shown below:



```
1 package testing.project;
2
3 public class TestingProject {
4     public static void main(String[] args) {
5         // TODO code application logic here
6         int x=2;
7         int y=2;
8         boolean c= true;
9         System.out.println(++x); //pre increment
10        System.out.println(y++); //post increment
11        System.out.println(x>>2); //right shift
12        System.out.println(x<<2); //left shift
13        System.out.println(!c); //unary not operator
14        System.out.println(x&2); //bitwise and operator
15    }
16 }
```

```
Output - Run (Output) %
-----< com.mycompany:OOPLab1 >-----
Building OOPLab1 1.0-SNAPSHOT
-----[ jar ]-----
--- exec-maven-plugin:3.0.0:exec (default-cli) @ OOPLab1 ---
3
2
0
12
false
2
```

### Ternary operator:

bool data= age>18? "can vote": "cannot vote"

### Control structures:

Java compiler executes the code from top to bottom. The statements in the code are executed according to the order in which they appear. However, Java provides statements that can be used to control the flow of Java code. Such statements are called control flow statements. It is one of the fundamental features of Java, which provides a smooth flow of program.

- Decision making (if statements, if else statements, switch statements etc.)
- Loop statements (while, do while, for, foreach)
- Jump statements (break, continue)

### Decision making statements:

In Java, the "if" statement is used to evaluate a condition. The control of the program is diverted depending upon the specific condition. The condition of the If statement gives a Boolean value, either true or false. In Java, there are four types of if-statements, i.e. if statement, if-else statements, if else-if ladder, nested if statements.

```
package testing.project;
import java.util.Scanner;
public class TestingProject {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter Your age\n");
        int age=s.nextInt();

        if(age >= 18){
            System.out.println("Congratulations! You can vote!");
        }
    }
}
```

Figure 1: if statement

```
package testing.project;
import java.util.Scanner;
public class TestingProject {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter Your age\n");
        int age=s.nextInt();

        if(age > 18){
            System.out.println("Congratulations! You can vote!");
        }
        else{
            System.out.println("Sorry, you can't vote");
        }
    }
}
```

Figure 2:if else statements

```

package testing.project;
import java.util.Scanner;
public class TestingProject {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter Your age\n");
        int age=s.nextInt();
        if(age<18){
            System.out.println("Hello! junior");
        }
        else if(age >= 18 && age<=60){
            System.out.println("Hey! you are young");
        }
        else if(age >= 60){
            System.out.println("Hey! you are a senior citizen");
        }
        else{
            System.out.println("Sorry, you eneterd bad input");
        }
    }
}

```

Figure 3:if else if statements

```

package testing.project;
import java.util.Scanner;
public class TestingProject {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter Your age\n");
        int age=s.nextInt();
        if(age<=19){
            if(age>=13){
                System.out.println("Hey! welcome to teens");
            }
            else{
                System.out.println("you are not a teenager, wait for your 13th birthday");
            }
        }
        else{
            System.out.println("You are a responsible citizen now, know it and worth it...");
        }
    }
}

```

Figure 4: nested if statements

If an if statements is placed with in the else statement then it is termed as if-else if ladder. Consider it a DIY exercise.

```
package testing.project;
import java.util.Scanner;
public class TestingProject {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter a letter \n");
        String letter=s.next();
        switch(letter){
            case "A":
                System.out.println("vowel");
                break;
            case "E":
                System.out.println("vowel");
                break;
            case "I":
                System.out.println("vowel");
                break;
            case "O":
                System.out.println("vowel");
                break;
            case "U":
                System.out.println("vowel");
                break;
            default:
                System.out.println("not a vowel");
        }
    }
}
```

Figure 5: Switch case

We will cover break statement in next sections.

### **3.2 Loops:**

In programming, sometimes we need to execute the block of code repeatedly while some condition evaluates to true. However, loop statements are used to execute the set of instructions in a repeated order. The execution of the set of instructions depends upon a particular condition.

In Java, we have three types of loops namely for loop, while loop and do while loop.

```

1 package testing.project;
2 import java.util.Scanner;
3 public class TestingProject {
4     public static void main(String[] args) {
5         for(int i=0;i<3;i++){
6             System.out.println("hello Students, a good day to you all!");
7         }
8     }
9 }
10
testing.project.TestingProject > main >
Output - testing project (run) x
run:
hello Students, a good day to you all!
hello Students, a good day to you all!
hello Students, a good day to you all!
BUILD SUCCESSFUL (total time: 0 seconds)

```

Figure 6: for loop

```

1 package testing.project;
2 public class TestingProject {
3     public static void main(String[] args) {
4         for (int row=1;row<3;row++){
5             for(int col=1;col<3;col++){
6                 System.out.println("row = "+row+", col= "+col);
7             }
8             System.out.println("\n");
9         }
10    }
11 }
testing.project.TestingProject > main > for(int row = 1; row < 3; row++) >
Output - testing project (run) x
run:
row = 1, col= 1
row = 1, col= 2

row = 2, col= 1
row = 2, col= 2

BUILD SUCCESSFUL (total time: 0 seconds)

```

Figure 7:nested for loop

While loop will check the condition first and then execute the statements if condition is true.

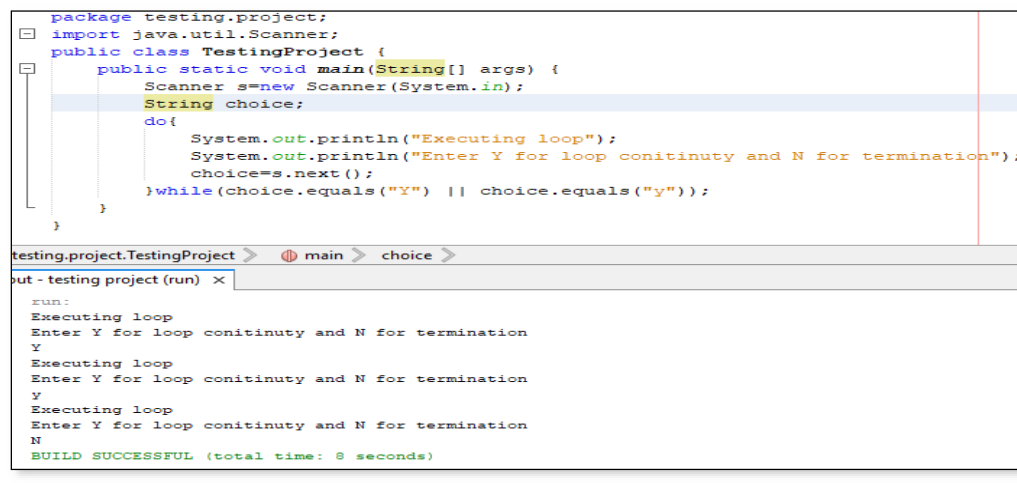
```

package testing.project;
import java.util.Scanner;
public class TestingProject {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        System.out.println("Enter Y for loop conitinity and N for termination");
        String choice=s.next();
        while(choice.equals("Y") || choice.equals("y")){
            System.out.println("Executing loop");
            System.out.println("Enter Y for loop conitinity and N for termination");
            choice=s.next();
        }
    }
}
testing.project.TestingProject > main > while (choice.equals("Y") || choice.equals("y")) >
Output - testing project (run) x
run:
Enter Y for loop conitinity and N for termination
y
Executing loop
Enter Y for loop conitinity and N for termination
Y
Executing loop
Enter Y for loop conitinity and N for termination
N
BUILD SUCCESSFUL (total time: 10 seconds)

```

Figure 8: while loop

Do while will execute loop body once and then check the condition.



```
package testing.project;
import java.util.Scanner;
public class TestingProject {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String choice;
        do{
            System.out.println("Executing loop");
            System.out.println("Enter Y for loop conitnuty and N for termination");
            choice=s.next();
        }while(choice.equals("Y") || choice.equals("y"));
    }
}
```

testing.project.TestingProject > main > choice >

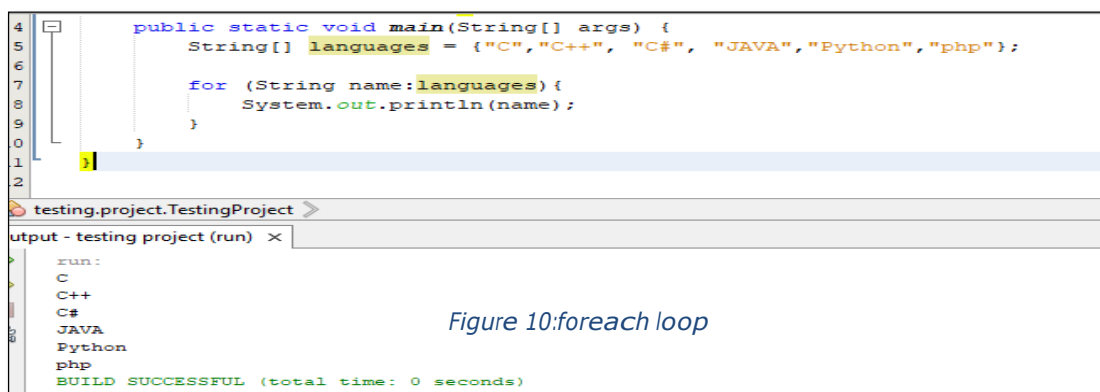
Output - testing project (run) x

run:

Executing loop  
Enter Y for loop conitnuty and N for termination  
Y  
Executing loop  
Enter Y for loop conitnuty and N for termination  
Y  
Executing loop  
Enter Y for loop conitnuty and N for termination  
N  
BUILD SUCCESSFUL (total time: 8 seconds)

Figure 9: do while loop

For-each is another array traversing technique like for loop, while loop, do-while loop introduced in Java5. It has a slightly different syntax than for loop shown below:



```
public static void main(String[] args) {
    String[] languages = {"C", "C++", "C#", "JAVA", "Python", "php"};
    for (String name:languages){
        System.out.println(name);
    }
}
```

testing.project.TestingProject >

Output - testing project (run) x

run:

C  
C++  
C#  
JAVA  
Python  
php  
BUILD SUCCESSFUL (total time: 0 seconds)

Figure 10:foreach loop

### 3.3 jump statements:

The break and continue statements are jump statements that are used to bypass some loop statements or finish the loop without verifying the test expression. These statements can be inserted into any loop, including for, while, and do-while loops.

The **break** statement in java is used to terminate from the loop immediately. When a break statement is encountered inside a loop, the loop iteration stops there, and control returns from the loop immediately to the first statement after the loop. Basically, break statements are used in situations when we are not sure about the actual number of iterations for the loop, or we want to terminate the loop based on some condition.



```
package testing.project;
public class TestingProject {
    public static void main(String[] args) {
        for (int row=1;row<5;row++){
            System.out.println("executing row number = "+row);
            if (row == 3){
                System.out.println("sorry for that, can't go any further");
                break;
            }
        }
    }
}

testing.project.TestingProject > main >
out - testing project (run) x
run:
executing row number = 1
executing row number = 2
executing row number = 3
sorry for that, can't go any further
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 11: break statement

The **continue** statement in Java is used to skip the current iteration of a loop. Generally, they are used in the situations when we want to continue the loop but do not want the remaining statement after the continue statement to be executed.

```
package testing.project;
public class TestingProject {
    public static void main(String[] args) {
        for (int row=1;row<5;row++){
            System.out.println("executing row number = "+row);
            if (row == 3){
                System.out.println("sorry for that, can't go any further");
                continue;
            }
            System.out.println("testing statement");
        }
    }
}

testing.project.TestingProject > main >
out - testing project (run) x
run:
executing row number = 1
testing statement
executing row number = 2
testing statement
executing row number = 3
sorry for that, can't go any further
executing row number = 4
testing statement
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 12: continue statement

There is no goto statement in Java but still goto is a reserved keyword. If they want to add it to the later versions. For now, the break statements use labels to jump to a specific line of code.

### Java Math Class:

Java Math class provides several methods to work on math calculations like min(), max(), avg(), sin(), cos(), tan(), round(), ceil(), floor(), abs() etc.

Function	Description
<b>Math.abs(x)</b>	It will return the Absolute value of the given value.
<b>Math.min(x,y)</b>	It returns the Largest of two values
<b>Math.max(x,y)</b>	It is used to return the Smallest of two values.
<b>Math.round(x)</b>	It is used to round of the decimal numbers to the nearest value.
<b>Math.sqrt(x)</b>	It is used to return the square root of a number.
<b>Math.cbrt(x)</b>	It is used to return the cube root of a number.
<b>Math.pow(x,y)</b>	It returns the value of first argument raised to the power to second argument.
<b>Math.signum(x)</b>	It is used to find the sign of a given value.
<b>Math.ceil(x)</b>	Compute ceiling of a number (round up forward to nearest decimal place)
<b>Math.floor(x)</b>	Compute floor of a number (round off backward to nearest decimal place)
<b>Math.random(x)</b>	generate a positive random number between 0 to 1.
<b>Math.sin(x)</b>	It is used to return the trigonometric Sine value of a Given double value.
<b>Math.log(x)</b>	It returns the natural logarithm of a double value.

```
package testing.project;
import java.lang.Math;
public class TestingProject {
    public static void main(String[] args) {
        int x=2;
        int y=4;
        double z=2.45;
        double t=45.56;
        int no=-3;
        System.out.println("absolute of x = "+ Math.abs(x));
        System.out.println("Maximum of x & y = "+ Math.max(x, y));
        System.out.println("Minimum of x & y = "+ Math.min(x, y));
        System.out.println("ceiling of z = "+ Math.ceil(z));
        System.out.println("floor of z = "+ Math.floor(z));
        System.out.println("Square root of t = "+Math.sqrt(t));
        System.out.println("cube root of t = " +Math.cbrt(t));
        System.out.println("round off t = "+Math.round(t));
        System.out.println("Sign of no = "+Math.signum(no));
        System.out.println("random no = "+Math.random());
    }
}
```

testing.project.TestingProject > main >

ut - testing project (run) x

```
run:
absolute of x = 2
Maximum of x & y = 4
Minimum of x & y = 2
ceiling of z = 3.0
floor of z = 2.0
Square root of t = 6.749814812274482
cube root of t = 3.571587040587451
round off t = 46
Sign of no = -1.0
random no = 0.39523829950752776
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 13: math library functions

## LAB TASKS:

### TASK – 01:

Your task is to design and implement an Invoice application in Java that will print the formatted invoice. Assume that you are designing this application to be used by a cashier at the POS (Point Of Sales).

- Your program starts by inputting items and their prices.
- It should then calculate the sub-total, tax, discount, and the final bill for a customer.
- Program will terminate if the user enters 'y' or 'Y'.

### Here is a sample interaction:

Enter items (first item's name and then price):

Chips 50

Hand Wash 200

Canned Almonds 125

Pickle 145

Apple juice 70

### Sample Output

Subtotal:	\$590.00
Discount percent:	10
Discount amount:	\$59.00
Total before tax:	\$531.00
Sales tax:	5
Invoice total:	\$557.55
Continue(y/n):	

### TASK – 02:

Write a program that reads a set of integers from 1 to 100, and then finds and prints the sum of the even and odd integers separately.

### TASK – 03:

Write a program that asks user an integer as input and print table of given integer up to 10.

### TASK – 04:

You are required to develop a program ProcessGrades to be used by the Examinations department.

- Your program should first prompt the user the number of students for whom she/he wishes to process grades.
- It should then read in a sequence of student numbers (note that loop has to be run for the number of students times, for example if user enters 10, then the program should read in number for 10 students) and computes the average, the number of students who pass (who obtain marks above or equals to 60) and the number of students who fail.
- Finally, it should display all information.

### TASK – 05:

Create a menu driven transactions' processing unit in a bank as long as user enters correct input. Initially, take the username and password as input from user. If that username and password is correct then the user will log in to the system and can perform any transaction as per the menu shown below.

Main Menu \*\* Welcome to Bank of Pakistan\*\*

1. Deposit Money
2. Withdraw Amount
3. Account status

Select your choice: \_\_\_\_\_

(After completing the selected transaction) Do you want to continue? [y/Y]

(goes to Main Menu, if y/Y is pressed)

Depending upon the user's choice, perform the transaction and display the remaining account balance along with the owner's username.