OOP IN JAVA

BCA 3RD SEMESTER

LAB REPORT

Er. Sital Prasad Mandal

Mechi Multiple Campus

Bhadrapur, Jhapa, Nepal

info.sitalmandal@gmail.com

https://ctaljava.blogspot.com/

JAVA PROGRAMMING LAB MANUAL

1. GUIDELINES TO STUDENTS

i. Equipment in the lab for the use of student community. Students need to maintain a proper decorum in the computer lab. Students must use the equipment with care. Any damage is caused is punishable.
ii. Students are instructed to come to lab in formal dresses only.
iii. Students are supposed to occupy the systems allotted to them and are not supposed to talk or make noise in the lab.
iv. Students are required to carry their observation book and lab records with completed exercises while entering the lab.
v. Lab records need to be submitted every week.
vi. Students are not supposed to use pen drives in the lab.

2. List Of Experiments As Per The University Curriculum

Marks Obtain Parameter: A=5, B=4, C=3, D=2

Lab. NO	EXP NO	EXPERIMENT NAME	Submited Mark Obt.	PAGE NO
1	Week – 1	Write a Program to print the text "Welcome to World of Java". Save it with name Welcome.java in your folder.	A	
2		Write a Program to print the area of triangle. Save it with name Area.java in your folder. (area = 1/2 (b × h)	A	
3		Write a java Program to check the number is Prime or not.	A	
4		Write a java Program to generate a Ladder of number.	A	
5		Bitwise Operator Shifting	A	
6		for-each loop **(a.)** Integer (**b.**) String	A	
7		Unary Operator Example: Complement (~)	A	
2. Classes and Objects, Constructors, parameterized constructor, Method overloading, constructor overloading				
8	Week – 2	Write a program to create a class Student with data 'name, city and age' along with method printData to display the data. Create the two objects s1 ,s2 to declare and access the values.	A	
9		Write a program to create a class Student2 along with two method getData(), printData() to get the value through argument and display the data in printData. Create the two objects s1 ,s2 to declare and access the values from class STtest.	A	
10		WAP using parameterized Employee constructor with two parameters id and name. While creating the objects obj1 and obj2 passed two arguments so that this constructor gets invoked after creation of obj1 and obj2.	A	
11		Write a program in JAVA to demonstrate the method and constructor overloading.	A	
12		Write a program in JAVA to create a class Bird also declares the different parameterized constructor to display the name of Birds.	A	
		Polymorphism		
13		Demonstrate Simple Calculator using Java. Here we have two definitions of the same method add() which add method would be called is determined by the parameter list at the C*ompile Time*.	A	
14		Demonstrate Simple Shape using Java. Here we have Three definitions of the same method draw( ) which different draw method, would be called is determined by their respected Shape at the *Run Time*.	A	
15	Week-3	Overriding Methods Example		

16		Dynamic Method Dispatch		
17		Using Object Class Method		
18		Using Packages Demo		

Q1. Lab 1

```
public class Lab1 {
    public static void main ( String args[] ) {
        System.out.println("welcome to world of Java");
    }
}
```

Q2. Lab 2

```
//Write a Program to print the area of triangle
public class Lab2 {
    public static void main ( String args[] ) {
        {
            int height = 10, base = 6;
            float area = 0.5F * base * height;
            System.out.println("area of triangle = " + area);
        }
    }
}
```

Q3. Lab 3

```
// Write a java Program to check the number is Prime or not.

import java.util.Scanner;
public class Lab3 {
    public static void main ( String arr[] ) {
        int c;
        Scanner in = new Scanner(System.in);
        System.out.println("Enter the number to be tested for prime ");
        int n = in.nextInt();
        for (c = 2; c <= n - 1; c++) {
            if (n % c == 0) {
                System.out.println(n + ">>>>> not prime");
                break;
            }
        }
        if (c == n)
            System.out.println(n + ">>>>Number is prime.");
    }
}
```

Q4. Lab 4

```
//Write a java Program to generate a Ladder of number.
import java.util.Scanner;
class Lab4 {
    public static void main ( String arr[] ) {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter the number of rows"); //5
        int a = in.nextInt();
        for (int i = 1; i <= a; i++) {
            for (int j = 1; j <= i; j++)
                System.out.print(j);
            for (int k = i - 1; k >= 1; k--)
                System.out.print(k);
            System.out.print("\n");
        }
    }
}
```

Output:

```
Enter the number of rows
5
1
121
12321
1234321
123454321
```

Q5. Lab 5

```
//Bitwise Operator Shifting
public class OperatorShifting {
    public static void main ( String[] args ) {
        byte l, r,b;
        l=10; //00001010
        r=10; //00001010
        System.out.println("Bitwise Left Shift: l<<2 = "+(l<<2)); //00101000 = 40
        System.out.println("Bitwise Right Shift: r>>2 = "+(r>>2)); //00000010 = 2
        // >>> Bitwise Zero Fill Right Shift
        b=32; //00100000
        System.out.println("b>>>2 = "+(b>>>2)); //00001000 = 8
        System.out.println("b>>>3 = "+(b>>>3)); //00000100 = 4
        System.out.println("b>>>4 = "+(b>>>4)); //00000010 = 2
    }
}
```

Output:

```
Bitwise Left Shift: l<<2 = 40
Bitwise Right Shift: r>>2 = 2
b>>>2 = 8
b>>>3 = 4
b>>>4 = 2
```

Q6. Lab 6

```
// a. ForEach Loop
public class Arraydemo {
    public static void main ( String[] args ) {
        int[] arr={10,20,30,40};
        for(int x:arr){
            System.out.println(x);
        }
    }
}
```

Output:

```
10
20
30
40
```

```
// b. ForEach Loop
import java.util.ArrayList;
import java.util.List;
public class ForEachDemo {
    public static void main ( String[] args ) {
        List<String> gamesList = new ArrayList<String>();
        gamesList.add("Football");
        gamesList.add("Cricket");
        gamesList.add("Chess");
        System.out.println("--Iterating by passing lambda expression--");
        // gamesList.forEach(System.out::print);
        for (String g : gamesList) {
            System.out.println(g);
        }
    }
}
```

Output

```
--Iterating by passing lambda expression--
Football
Cricket
Chess
```

Q7. Lab 7

```
// Unary Operator Example: Complement (~)
public class OperatorDemo {
    public static void main ( String[] args ) {
            int a=5;
            int b=-5;
            System.out.println(~a); //-6
       //(minus of total positive value which starts from 0)
    }
}
```

Illustration:
a = 5 [0101 in Binary]
result = ~5
This performs a bitwise complement of 5
~0101 = 1010 = 10 (in decimal)
Then the compiler will give 2's complement of that number.
2's complement of 10 will be -6.
result = -6

Viva Questions

Q1. Explain JDK, JRE and JVM? JDK vs JRE vs JVM		
JDK	**JRE**	**JVM**
It stands for Java Development Kit.	It stands for Java Runtime Environment.	It stands for Java Virtual Machine.
It is the tool necessary to compile, document and package Java programs.	JRE refers to a runtime environment in which Java bytecode can be executed.	It is an abstract machine. It is a specification that provides a run-time environment in which Java bytecode can be executed.
It contains JRE + development tools.	It's an implementation of the JVM which physically exists.	JVM follows three notations: Specification, Implementation, and Runtime Instance.

Q2. Explain public static void main(String args[]) in Java.

main() in Java is the entry point for any Java program. It is always written as public static void main(String[] args).

- **public**: Public is an access modifier, which is used to specify who can access this method. Public means that this Method will be accessible by any Class.
- **static**: It is a keyword in java which identifies it is class-based. main() is made static in Java so that it can be accessed without creating the instance of a Class. In case, main is not made static then the compiler will throw an error as main() is called by the JVM before any objects are made and only static methods can be directly invoked via the class.
- **void**: It is the return type of the method. Void defines the method which will not return any value.
- **main**: It is the name of the method which is searched by JVM as a starting point for an application with a particular signature only. It is the method where the main execution occurs.
- **String args[]:** It is the parameter passed to the main method.

Q3. Why Java is platform independent?

Java is called platform independent because of its byte codes which can run on any system irrespective of its underlying operating system.

Q.4 why is main method declared as static?

Ans: The main method is static because it keeps things simpler. Since the main method is static JVM (Java virtual Machine) can call it without creating any instance of a class which contains the main method. The main() method must be declared public, static, and void. It must accept a single argument that is an array of strings. This method can be declared as either:

public static void main(String[] args)

Q.5 Is JDK required on each machine to run a java program?

Ans. No, JDK (Java Development Kit) isn't required on each machine to run a Java program. Only JRE is required, it is an implementation of the Java Virtual machine (JVM), which actually executes Java programs. JDK is development Kit of Java and is required for development only. It is a bundle of software components that is used to develop Java based applications.

2. Classes and Objects, Constructors

Q.8 Write a program to create a class Student with data 'name, city and age' along with method printData to display the data. Create the two objects s1 ,s2 to declare and access the values.

```
//Q8. Lab Student Object
class Student {
    static int m;
    String name, city;
    int age;
    void printData () {
        System.out.println("Student name = " + name);
        System.out.println("Student city = " + city);
        System.out.println("Student age = " + age);
    }
}

class Lab8 {
    public static void main ( String args[] ) {
        Student s1 = new Student();
        Student s2 = new Student();
        s1.name = "Hari";
        s1.city = "KTM";
        s1.age = 22;
        s2.name = "Roshan";
        s2.city = "BDH";
        s2.age = 23;
        s2.printData();
        s1.printData();
        s1.m = 20;
        s2.m = 22;
        Student.m = 27;
        System.out.println("s1.m = " + s1.m);
        System.out.println("s2.m = " + s2.m);
        System.out.println("Student.m =" + Student.m);
```

```
    }
}
```

```
Student name = Roshan
Student city = BDH
Student age = 23
Student name = Hari
Student city = KTM
Student age = 22
s1.m = 27
s2.m = 27
Student.m =27
```

Q9. Write a program to create a class Student2 along with two method getData(), printData() to get the value through argument and display the data in printData. Create the two objects s1 ,s2 to declare and access the values from class STtest.

```
//Q9. Lab
class Student2 {
    private String name, city;
    private int age;

    public void getData ( String x, String y, int t ) {
        name = x;
        city = y;
        age = t;
    }

    public void printData () {
        System.out.println("Student name =" + name);
        System.out.println("Student city =" + city);
        System.out.println("Student age =" + age);
    }
}
// STtest
class Lab9 {
    public static void main ( String args[] ) {
        Student2 s1 = new Student2();
        Student2 s2 = new Student2();
        s2.getData("Hari", "BDH", 23);
        s2.printData();
        s1.getData("Roshan", "KTM", 22);
        s1.printData();
    }
}
```

Output:

```
Student name =Hari
Student city =BDH
Student age =23
Student name =Roshan
Student city =KTM
Student age =22
```

Q10. WAP using parameterized Employee constructor with two parameters id and name. While creating the objects obj1 and obj2 passed two arguments so that this constructor gets invoked after creation of obj1 and obj2.

```
//Q10. lab
class Employee {
    int empId;
    String empName;

    //parameterized constructor with two parameters
    Employee ( int id, String name ) {
        this.empId = id;
        this.empName = name;
    }

    public static void main ( String args[] ) {
        Employee obj1 = new Employee(10245, "Rahul");
        Employee obj2 = new Employee(92232, "Bibek");
        obj1.info();
        obj2.info();
    }

    void info () {
        System.out.println("Id: " + empId + " Name: " + empName);
    }
}
```

Output:

```
Id: 10245 Name: Rahul
Id: 92232 Name: Bibek
```

Q11. Write a program in JAVA to demonstrate the method and constructor overloading.

```
// Constructor Overlaoding
class Cs {
    int p, q;

    public Cs () {
    }
    public Cs ( int x, int y ) {
        p = x;
        q = y;
    }
    public int add ( int i, int j ) {
        return (i + j);
    }
    public int add ( int i, int j, int k ) {
        return (i + j + k);
    }
    public float add ( float f1, float f2 ) {
        return (f1 + f2);

    }

    public void printData () {
        System.out.print("p = " + p);
        System.out.println(" q = " + q);
    }
}

class ConstructorOverlaoding {
    public static void main ( String args[] ) {
        int x = 2, y = 3, z = 4;
        Cs c = new Cs();
        Cs c1 = new Cs(x, z);
        c1.printData();
        float m = 7.2F, n = 5.2F;
        int k = c.add(x, y);
        int t = c.add(x, y, z);
        float ft = c.add(m, n);
        System.out.println("k = " + k);
        System.out.println("t = " + t);
        System.out.println("ft = " + ft);
    }
}
```

```
p = 2 q = 4
k = 5
t = 9
ft = 12.4
```

Q12. Write a program in JAVA to create a class Bird also declares the different parameterized constructor to display the name of Birds.

```
class Bird {
    int age;
    String name;

    Bird () {
        System.out.println("this is the perrot");
    }

    Bird ( String x ) {
        name = x;
        System.out.println("this is the " + name);
    }

    Bird ( int y, String z ) {
        age = y;
        name = z;
        System.out.println("this is the " + age + "years\t" + name);
    }

    public static void main ( String arr[] ) {
        // call it self default constructor when Object is created
        Bird obj = new Bird();   // this is the perrot initialized
        //obj.Bird(); // error no need to call
        Bird b = new Bird("Duck");
        Bird c = new Bird(20, "sparrow");
    }
}
```

```
this is the perrot
this is the Duck
this is the 20years sparrow
```

Viva Questions

Q.1 What is a Constructor?

Ans. Constructors are used to initialize the object's state. Like methods, a constructor also contains collection of statements(i.e. instructions) that are executed at time of Object creation.

Q.2 What are the differences between C++ and Java?

Ans.The differences between C++ and Java are given in the following table.

Comparison Index	C++	Java
Platform-independent	C++ is platform-dependent.	Java is platform-independent.
Mainly used for	C++ is mainly used for system programming.	Java is mainly used for application programming. It is widely used in window, web-based, enterprise and mobile applications.
Design Goal	C++ was designed for systems and applications programming. It was an extension of C programming language	Java was designed and created as an interpreter for printing systems but later extended as a support network computing. It was designed with a goal of being easy to use and accessible to a broader audience.
Goto	C++ supports the goto statement.	Java doesn't support the goto statement.
Multiple inheritance	C++ supports multiple inheritance.	Java doesn't support multiple inheritance through class. It can be achieved by interfaces in java.
Operator Overloading	C++ supports operator overloading.	Java doesn't support operator overloading.
Pointers	C++ supports pointers. You can write pointer program in C++.	Java supports pointer internally. However, you can't write the pointer program in java. It means java has restricted pointer support in java.
Compiler and Interpreter	C++ uses compiler only. C++ is compiled and run using the compiler which converts source code into machine code so, C++ is platform dependent.	Java uses compiler and interpreter both. Java source code is converted into bytecode at compilation time. The interpreter executes this bytecode at runtime and produces output. Java is interpreted that is why it is platform independent.
Call by Value and Call by reference	C++ supports both call by value and call by reference.	Java supports call by value only. There is no call by reference in java.

Structure and Union	C++ supports structures and unions.	Java doesn't support structures and unions.
Thread Support	C++ doesn't have built-in support for threads. It relies on third-party libraries for thread support.	Java has built-in thread support.
Virtual Keyword	C++ supports virtual keyword so that we can decide whether or not override a function.	Java has no virtual keyword. We can override all non-static methods by default. In other words, non-static methods are virtual by default.
unsigned right shift >>>	C++ doesn't support >>> operator.	Java supports unsigned right shift >>> operator that fills zero at the top for the negative numbers. For positive numbers, it works same like >> operator.
Inheritance Tree	C++ creates a new inheritance tree always.	Java uses a single inheritance tree always because all classes are the child of Object class in java. The object class is the root of the inheritance tree in java.
Hardware	C++ is nearer to hardware.	Java is not so interactive with hardware.
Object-oriented	C++ is an object-oriented language. However, in C language, single root hierarchy is not possible.	Java is also an object-oriented language. However, everything (except fundamental types) is an object in Java. It is a single root hierarchy as everything gets derived from java.lang.Object.

Q3. Can we have a class with no Constructor in it ? What will happen during object creation ?

Ans. Yes, we can have a class with no constructor, When the compiler encounters a class with no constructor then it will automatically create a default constructor for you.

Q4.What is No-arg constructor?

Ans. Constructor without arguments is called no-arg constructor. Default constructor in java is always a no-arg constructor.

Q.5 If I don't provide any arguments on the command line, then what will the value stored in the String array passed into the main() method, empty or NULL?

Ans. It is empty, but not null.

Q.6 What is the constructor?

AnsThe constructor can be defined as the special type of method that is used to initialize the state of an object. It is invoked when the class is instantiated, and the memory is allocated for the object. Every time, an object is created using the new keyword, the default constructor of the class is called. The name of the constructor must be similar to the class name. The constructor must not have an explicit return type.

Q.7 How many types of constructors are used in Java?

Ans. Based on the parameters passed in the constructors, there are two types of constructors in Java.

- **Default Constructor**: default constructor is the one which does not accept any value. The default constructor is mainly used to initialize the instance variable with the default values. It can also be used for performing some useful task on object creation. A default constructor is invoked implicitly by the compiler if there is no constructor defined in the class.

- **Parameterized Constructor**: The parameterized constructor is the one which can initialize the instance variables with the given values. In other words, we can say that the constructors which can accept the arguments are called parameterized constructors.

Q.8 Is constructor inherited?

Ans.No, The constructor is not inherited.

Q.9 What are the differences between the constructors and methods?

Ans.There are many differences between constructors and methods. They are given below.

Java Constructor	Java Method
A constructor is used to initialize the state of an object.	A method is used to expose the behavior of an object.
A constructor must not have a return type.	A method must have a return type.
The constructor is invoked implicitly.	The method is invoked explicitly.
The Java compiler provides a default constructor if you don't have any constructor in a class.	The method is not provided by the compiler in any case.
The constructor name must be same as the class name.	The method name may or may not be same as class name.

Q.10 Can we have both Default Constructor and Parameterized Constructor in the same class?

Ans. Yes, we have both Default Constructor and Parameterized Constructor in the same class.

Q13. Demonstrate Simple Calculator using Java. Here we have two definitions of the same method **add()** which add method would be called is determined by the parameter list at the compile time.

```
// Lab.13 Static Polymorphism
class SimpleCalculator {
    int add ( int a, int b ) {
        return a + b;
    }

    int add ( int a, int b, int c ) {
        return a + b + c;
    }
}

public class Lab13 {
    public static void main ( String args[] ) {
        SimpleCalculator obj = new SimpleCalculator();
        System.out.println(obj.add(20, 20));
        System.out.println(obj.add(20, 20, 30));
    }
}
```

Output:

40
70

Q14. Demonstrate Simple Shape using Java. Here we have Threee definitions of the same method **draw( )** which different draw method, would be called is determined by their respected Shape at the Run time.

```
//Q14. Java Runtime Polymorphism Example: Shape
class Shape {
    void draw () {
        System.out.println("drawing...");
    }
}

class Rectangle extends Shape {
    void draw () {
        System.out.println("drawing rectangle...");
    }
}

class Circle extends Shape {
    void draw () {
        System.out.println("drawing circle...");
    }
}

class Triangle extends Shape {
    void draw () {
        System.out.println("drawing triangle...");
    }
}

class Lab14 {
    public static void main ( String args[] ) {
        Shape s;
        s = new Rectangle();
        s.draw();
        s = new Circle();
        s.draw();
        s = new Triangle();
        s.draw();
    }
}
```

Output:

drawing rectangle...
drawing circle...
drawing triangle...

Week-3

Q15. Overriding Methods Example:

Declaring a method in sub class which is already present in parent class is known as method overriding.

```
class Human{
    //Overridden method
    public void eat()
    {
        System.out.println("Human is eating");
    }
}
class Boy extends Human{
    //Overriding method
   /* public void eat(){  // Boy is eating
        System.out.println("Boy is eating");
    }*/
    public static void main( String args[]) {
        Human obj = new Boy();
        //This will call the child class version of eat()
        obj.eat(); //Human is eating
    }
}
```

Q16. Dynamic Method Dispatch

```
class ABC{
    //Overridden method
    public void disp()
    {
        System.out.println("disp() method of parent class");
    }

}
class Demo extends ABC{
    //Overriding method
    public void disp(){
        System.out.println("disp() method of Child class");
    }
    public void newMethod(){
        System.out.println("new method of child class");
    }
    public static void main( String args[]) {
        /* When Parent class reference refers to the parent class object
         * then in this case overridden method (the method of parent class)
         *  is called.
         */
        ABC obj = new ABC();
        obj.disp();

        /* When parent class reference refers to the child class object
         * then the overriding method (method of child class) is called.
         * This is called dynamic method dispatch and runtime polymorphism
         */
        ABC obj2 = new Demo();
        obj2.disp();
    //  obj2.newMethod(); //error Unresolved compilation, newMethod() is undefined for the type
ABC
    }
}
```

Output:

disp() method of parent class

disp() method of Child class

Q17. Object Class Method

//a. Clone( )

```
import java.util.*;
public class ObjectClassClone {
    public static void main(String[] args) {
        Date date = new Date();
        System.out.println(date.toString());
        Date date2 = (Date) date.clone();
        System.out.println(date2.toString());
    }
}
```

```
Output:
Tue Nov 08 09:01:03 NPT 2022
Tue Nov 08 09:01:03 NPT 2022
```

//b. Equals( )

```
public class ObjectClassEquals {
    public static void main ( String[] args ) {
        // get an integer, which is an object
        Integer x = new Integer(50);

        // get a float, which is an object as well
        Float y = new Float(50f);

        // check if these are equal,which is
        // false since they are different class
        System.out.println("" + x.equals(y));

        // check if x is equal with another int 50
        System.out.println("" + x.equals(50));
    }
}
```

Output:

false
true

//c. getClass( )

```
class Person1 {
    private String firstName;
    public static void main ( String[] args ) {
        Person1 person = new Person1();
        System.out.println(person.getClass());
    }
}

//output:
       class Unit4.Person1
```

Q18. Packages Demo

```
// Multiple.java

package Unit4.pakagesdemo;
public class Multiple {
    public int MultipleFun(int a, int b){
        int p;
        p = a * b;
        return p;
    }
}

// UsingPakage.java

import Unit4.pakagesdemo.Multiple;

public class UsingPakage{
    public static void main(String[] args) {
        Multiple m = new Multiple();
        int mul = m.MultipleFun(5,6);
        System.out.println(mul);
    }
}

O/P: 30
```