INDEX

Week & Date	Idx	Program	Sign
6	6.1	Sum of first n natural numbers using recursion	
	6.2	Factorial of given number using recursion	
	6.3	Generate Fibonacci Series using recursion	
	6.4	Constructor Overloading	
	6.5	Method overloading	
	6.6	This keyword usage for name conflict resolution	
	6.7	Constructor chaining	
	6.8	Method chaining	
	6.9	Access controls	

Problem 1

Problem: Sum of first n natural numbers using recursion

Java Solution

```
import java.util.Scanner;
class Sum
{
 public static void main(String args[])
 {
  Scanner sc= new Scanner(System.in);
  System.out.print(" Enter N Value : ");
  int n= sc.nextInt();
  int r=Sum_N(n);
  System.out.print(" Sum of first "+n+" natural numbers =  "+r);
 }
static int Sum_N(int n)
{
 if(n==1)
   return 1;
 return (n+Sum_N(n-1));
}
}
```

Sample Output

```
D:\AY2026_27\JAVA\LAB\Week6>javac Sum.java

D:\AY2026_27\JAVA\LAB\Week6>java Sum
 Enter N Value : 5
 Sum of first 5 natural numbers =  15
```

Problem 2

Problem: **Factorial of given number using recursion**

Java Solution

```
import java.util.Scanner;
class Factorial
{
  int Fact(int i)
  {
   if(i==0)
     return 1;
   return i*Fact(i-1);
  }
}

class FactDemo
{
 public static void main(String args[])
 {
  Scanner sc= new Scanner(System.in);
  System.out.print(" Enter N Value : ");
  int n= sc.nextInt();
  Factorial f=new Factorial();
  int r = f.Fact(n);
  System.out.print(" Factorial of "+n+" =  "+r);
 }
}
```

Sample Output

```
D:\AY2026_27\JAVA\LAB\Week6>javac FactDemo.java

D:\AY2026_27\JAVA\LAB\Week6>java FactDemo
 Enter N Value : 5
 Factorial of 5 =  120
```

Problem 3

Problem: Generate Fibonacci Series using recursion

Java Solution

```
import java.util.Scanner;
class Series
{
  int Fib(int n)
  {
   if((n==0)||(n==1))
     return n;
   return Fib(n-1)+Fib(n-2);
  }
}

class Fibonacci
{
 public static void main(String args[])
 {
  Scanner sc= new Scanner(System.in);
  System.out.print(" Enter Number of terms : ");
  int n= sc.nextInt();
  System.out.println("Fibonacci Series upto "+n+" terms is ");
  Series f=new Series();
  for(int i=0;i<=n;i++)
  {
    int r = f.Fib(i);
  System.out.print(" "+r);
 }
}
}
```

Sample Output

```
D:\AY2026_27\JAVA\LAB\Week6>javac Fibonacci.java

D:\AY2026_27\JAVA\LAB\Week6>java Fibonacci
 Enter Number of terms : 8
Fibonacci Series upto 8 terms is
 0 1 1 2 3 5 8 13 21
```

Problem 4

Problem: Find Area and Perimeter of Rectangle using Constructor overloading

---

Java Solution

```
class Rectangle {
    int length;
    int breadth;


    Rectangle() {
        length = 10;
        breadth = 5;
    }

    Rectangle(int side) {
        length = side;
        breadth = side;
    }

    Rectangle(int length, int breadth) {
        this.length = length;
        this.breadth = breadth;
    }

    void area() {
        System.out.println("Area = " + (length * breadth));
    }

    void perimeter() {
        System.out.println("Perimeter = " + (2 * (length + breadth)));
    }
}
public class ConstructorOverloadingDemo {
    public static void main(String[] args) {
```

```
        Rectangle r1 = new Rectangle();
        System.out.println("Rectangle 1:");
        r1.area();
        r1.perimeter();

        Rectangle r2 = new Rectangle(8);
        System.out.println("\nRectangle 2:");
        r2.area();
        r2.perimeter();


        Rectangle r3 = new Rectangle(10, 5);
        System.out.println("\nRectangle 3:");
        r3.area();
        r3.perimeter();
    }
}
```

Sample Output

```
D:\AY2026_27\JAVA\LAB\Week6>javac ConstructorOverloadingDemo.java

D:\AY2026_27\JAVA\LAB\Week6>java ConstructorOverloadingDemo
Rectangle 1:
Area = 50
Perimeter = 30

Rectangle 2:
Area = 64
Perimeter = 32

Rectangle 3:
Area = 50
Perimeter = 30
```

Problem 5

Problem: Adding numbers using method overloading

Java Solution

```
class Calculator {

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


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


    double add(double a, double b) {
        return a + b;
    }


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

public class MethodOverloadingDemo {
    public static void main(String[] args) {

        Calculator c = new Calculator();

        System.out.println("Addition of two integers: "
```

```
                + c.add(10, 20));

        System.out.println("Addition of three integers: "
                + c.add(10, 20, 30));

        System.out.println("Addition of two double values: "
                + c.add(10.5, 20.5));

        System.out.println("Addition of two integers and one double: "
                + c.add(10, 20, 5.5));
    }
}
```

Sample Output

```
D:\AY2026_27\JAVA\LAB\Week6>javac MethodOverloadingDemo.java

D:\AY2026_27\JAVA\LAB\Week6>java MethodOverloadingDemo
Addition of two integers: 30
Addition of three integers: 60
Addition of two double values: 31.0
Addition of two integers and one double: 35.5
```

Problem 6

Problem: Demonstrating *this* keyword for name conflict resolution between instance variables and formal parameters using Student data.

Java Solution

```
class Student {
    int rollNo = 0;
    String name = "Unknown";

    void setDetails_with(int rollNo, String name) {
        this.rollNo = rollNo;
        this.name = name;
    }

   void setDetails_without(int rollNo, String name) {
        rollNo = rollNo;
        name = name;
    }

    void display() {
        System.out.println("Roll No : " + rollNo);
        System.out.println("Name    : " + name);
    }
}

public class ThisDemo {
    public static void main(String[] args) {

        Student s = new Student();

        s.setDetails_without(101, "Ravi");
```

```
        System.out.println("without using this keyword calling setDetails():");
        s.display();

        s.setDetails_with(101, "Ravi");

        System.out.println("\nUsing this keyword calling setDetails():");
        s.display();
    }
}
```

Sample Output

```
D:\AY2026_27\JAVA\LAB\Week6>javac ThisDemo.java

D:\AY2026_27\JAVA\LAB\Week6>java ThisDemo
without using this keyword calling setDetails():
Roll No : 0
Name    : Unknown

Using this keyword calling setDetails():
Roll No : 101
Name    : Ravi
```

Problem 7

Problem: Demonstrating Constructor chaining

Java Solution

```
class BankAccount {
   int accountNo;
   String holderName;
   double balance;

     BankAccount() {
       this(1001, "Ravi", 5000);
       System.out.println("Default constructor");
   }


     BankAccount(int accountNo) {
        this.accountNo=accountNo;
       System.out.println("One-argument constructor");
   }


   BankAccount(int accountNo, String holderName) {
       this(accountNo);
        this.holderName=holderName;
       System.out.println("Two-argument constructor");
   }

   BankAccount(int accountNo, String holderName, double balance) {
     this(accountNo,holderName);
```

```
        this.balance = balance;
        System.out.println("Three-argument constructor");
    }

    void display() {
        System.out.println("Account No : " + accountNo);
        System.out.println("Name       : " + holderName);
        System.out.println("Balance    : " + balance);
    }
}

public class ConstructorChainingDemo {
    public static void main(String[] args) {

        BankAccount b1 = new BankAccount();

        System.out.println("\nAccount Details:");
        b1.display();
    }
}
```

Sample Output

```
D:\AY2026_27\JAVA\LAB\Week6>javac ConstructorChainingDemo.java

D:\AY2026_27\JAVA\LAB\Week6>java ConstructorChainingDemo
One-argument constructor
Two-argument constructor
Three-argument constructor
Default constructor

Account Details:
Account No : 1001
Name       : Ravi
Balance    : 5000.0
```

Problem 8

Problem: Demonstrating method chaining

Java Solution

```
class Car {
    String brand;
    String color;
    int speed;

    Car setBrand(String brand) {
        this.brand = brand;
        return this;
    }

    Car setColor(String color) {
        this.color = color;
        return this;
    }

    Car setSpeed(int speed) {
        this.speed = speed;
        return this;
    }

    Car accelerate(int increase) {
        speed = speed + increase;
```

```
        return this;
    }

    void display() {
        System.out.println("Brand : " + brand);
        System.out.println("Color : " + color);
        System.out.println("Speed : " + speed + " km/h");
    }
}

public class MethodChainingDemo {
    public static void main(String[] args) {

        Car c = new Car();

        c.setBrand("Toyota")
         .setColor("White")
         .setSpeed(40)
         .accelerate(20)
         .display();
    }
}
```

Sample Output

```
D:\AY2026_27\JAVA\LAB\Week6>javac MethodChainingDemo.java

D:\AY2026_27\JAVA\LAB\Week6>java MethodChainingDemo
Brand : Toyota
Color : White
Speed : 60 km/h
```

Problem 9

Problem: Demonstrating Access Controls in Java

Java Solution

```
class MobilePhone {

        private int battery = 80;
        String brand = "Samsung";
        protected String model = "Galaxy";
        public void display() {
        System.out.println("Brand   : " + brand);
        System.out.println("Model   : " + model);
        System.out.println("Battery : " + battery + "%");
    }


    public int getBattery() {
        return battery;
    }

    public void charge(int amount) {
        if (amount > 0 && battery + amount <= 100) {
            battery = battery + amount;
        } else {
```

```
            System.out.println("Invalid charging amount!");
        }
    }
}

public class AccessControlDemo {
    public static void main(String[] args) {

        MobilePhone phone = new MobilePhone();
        phone.display();
         phone.brand="Vivo";
        System.out.println("\nBrand: " + phone.brand);
         phone.model="x300";
        System.out.println("Model: " + phone.model);

       // Access private variable through public method
        System.out.println("Battery: " + phone.getBattery() + "%");

        // Modify private variable through public method
        phone.charge(15);

        System.out.println("\nAfter charging:");
        phone.display();
    }
}
```

Sample Output

```
D:\AY2026_27\JAVA\LAB\Week6>javac AccessControlDemo.java

D:\AY2026_27\JAVA\LAB\Week6>java AccessControlDemo
Brand   : Samsung
Model   : Galaxy
Battery : 80%

Brand: Vivo
Model: x300
Battery: 80%

After charging:
Brand   : Vivo
Model   : x300
Battery : 95%
```