ASSIGNMENT – 04

Name: - Aniket Audumbar Yadav

Roll No: - 18

PRN: - 2324000609

Subject: - Object Oriented Programming

Class: SY AIML

Div: - B

Batch: - B1

1. Write Java Program to understand and implement the concept of single inheritance in Java using Department as a parent class and Employee as a child class. Problem Statement - Create a class Department that contains department ID and department name. Derive a class Employee from Department that includes employee ID, name, and salary. Accept and display the data for an employee along with their department details using inheritance. Modify your code to add Phone Number as String in Employee class

```
//    Parent    Class

class Department {

int deptId;

   String deptName;


   Department(int deptId, String deptName) {

this.deptId = deptId;        this.deptName =

deptName;

   }


   void displayDepartment() {

       System.out.println("Department ID: " + deptId);
System.out.println("Department Name: " + deptName);

   }

}


// Child Class
```

```
class Employee extends Department {

int empId;

   String empName;

double salary;

        String phno;

        // add phone number as String


   Employee(int deptId, String deptName, int empId, String empName, double
salary,String phno) {

      super(deptId, deptName); // calling parent constructor

this.empId = empId;       this.empName = empName;

this.salary = salary;

                this.phno=phno;

   }


   void displayEmployee() {

      System.out.println("Employee ID: " + empId);
System.out.println("Employee Name: " + empName);

      System.out.println("Salary: " + salary);

                System.out.println("phno: " + phno);

      displayDepartment(); // accessing parent method

   }

}
```

```
// Main Class

public class ERPSystemDemo {

    public static void main(String[] args) {

        Employee e = new Employee(101, "Computer Science", 501, "Rahul Patil",
60000.0,"9087564321");

        e.displayEmployee();

                    // add phone number in parameter & Test it


    }

}
```

Output:

```
D:\javalabB18\Assignment 4 - Lab - Java Inheritance & Interface>javac
ERPSystemDemo.java

D:\javalabB18\Assignment 4 - Lab - Java Inheritance & Interface>java
ERPSystemDemo

Employee ID: 501

Employee Name: Rahul Patil

Salary: 60000.0 phno:

9087564321

Department ID: 101

Department Name: Computer Science
```

2. Java Program to implement multilevel inheritance in Java using a realworld academic structure: University → College → Department

```
// Base class class
University {
    String universityName;
    String location;

    void displayUniversityInfo() {
        System.out.println("University Name: " + universityName);
        System.out.println("Location: " + location);
    }
}

// Derived from University class
College extends University {
    String collegeName;
    String collegeCode;

    void displayCollegeInfo() {
        System.out.println("College Name: " + collegeName);
        System.out.println("College Code: " + collegeCode);
    }
}
```

```
// Derived from College class
Department extends College {
String departmentName;     int
studentCount;

   void displayDepartmentInfo() {
      System.out.println("Department: " + departmentName);
      System.out.println("Total Students: " + studentCount);
   }
}

// Main class to test the structure public
class EducationSystem {     public static
void main(String[] args) {
      Department dept = new Department();

      // Setting values
      dept.universityName = "Shivaji University";        dept.location = "Kolhapur";
      dept.collegeName = "KIT College of Engineering";
dept.collegeCode = "6267";
      dept.departmentName = "Computer Science & Engineering";
dept.studentCount = 180;
```

```
        // Displaying information
        System.out.println("--- Department Details ---");
dept.displayUniversityInfo();        dept.displayCollegeInfo();
dept.displayDepartmentInfo();
    }
}
```

Output:

```
D:\javalabB18\Assignment 4 - Lab - Java Inheritance & Interface>javac
EducationSystem.java

D:\javalabB18\Assignment 4 - Lab - Java Inheritance & Interface>java
EducationSystem

--- Department Details --- University

Name: Shivaji University

Location: Kolhapur

College Name: KIT College of Engineering

College Code: 6267

Department: Computer Science & Engineering

Total Students: 180
```

3.Write Program to implement Java method overloading concept. Create add method versions for integer values with below method signature. ✓ add(int a, int b) – returns the sum of two integers. ✓ add(int a, int b, int c) – returns the sum of three integers ✓ Add your logic to Source Code - AdditionTest.java

```
// Class demonstrating method overloading class

Addition {
    // Method to add two integers
int add(int a, int b) {
return a + b;
    }

    // Overloaded method to add three integers
int add(int a, int b, int c) {         return a + b
+ c;
    }

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


          // Overloaded method to add 4 integers and Test it
}

// Main class to test Addition public
class AdditionTest {
    public static void main(String[] args) {
        Addition adder = new Addition();
        // Calling both methods         int
sumTwo = adder.add(10, 20);         int
sumThree = adder.add(10, 20, 30);
                    int sumFour = adder.add(10, 20, 30,40);


        // Printing results
        System.out.println("Sum of two numbers (10 + 20): " + sumTwo);
        System.out.println("Sum of three numbers (10 + 20 + 30): " + sumThree);
  System.out.println("Sum of three numbers (10 + 20 + 30+40): " + sumFour);
    }
}
```

Output:

```
D:\javalabB18\Assignment 4 - Lab - Java Inheritance & Interface>javac
AdditionTest.java

D:\javalabB18\Assignment 4 - Lab - Java Inheritance & Interface>java
AdditionTest

Sum of two numbers (10 + 20): 30

Sum of three numbers (10 + 20 + 30): 60

Sum of three numbers (10 + 20 + 30+40): 100
```

4. Write Java Program to implement Java method overriding concept. Create calculate method versions to find square, squareRoot and Cube. Add your logic to Source Code - CalculatorTest.java

```
// Base class class
Calculator {
    void calculate(double number) {
                    double square = number * number;
        System.out.println("Square: " + square);
    }
}

// Derived class with overridden method class
FindSquareRoot extends Calculator {
    @Override
    void calculate(double number) {

        double squareRoot = Math.sqrt(number);
        System.out.println("Square Root: " + squareRoot);
    }
}

class FindCube extends Calculator {
    @Override
    void calculate(double number) {

        double squarecube = number*number*number;
        System.out.println("Square cube: " + squarecube);
    }
}

// Add FindCube extends Calculator logic to calculate the cube

// Main class to test method overriding
public class CalculatorTest {        public
static void main(String[] args) {
        Calculator calc = new FindSquareRoot(); // dynamic polymorphism
calc.calculate(16.5); // calling overridden method with a double value

                Calculator calc1 = new FindCube();
calc1.calculate(3);

    }
}
```

Output:

```
D:\javalabB18\Assignment 4 - Lab - Java Inheritance & Interface>javac
CalculatorTest.java

D:\javalabB18\Assignment 4 - Lab - Java Inheritance & Interface>java
CalculatorTest

Square Root: 4.06201920231798

Square cube: 27.0
```

5. Write Java Program to implement Java Abstract Class for below diagram.

Add your logic to Source Code - CommonOperations.java abstract class MyClass{ // this is abstract method abstract void calculate(double num);

```
}


class FindSqaure extends MyClass{
public void calculate(double num){
                System.out.println("Sqaure="+ (num*num));
        }
}
class FindSquareRoot extends MyClass {
   @Override
   void calculate(double number) {


      double squareRoot = Math.sqrt(number);
      System.out.println("Square Root: " + squareRoot);
   }
}


class FindCube extends MyClass {
   @Override
   void calculate(double number) {
        double squarecube = number*number*number;

        System.out.println("Square cube: " + squarecube);

    }

}
```

Output:

```
D:\javalabB18\Assignment 4 - Lab - Java Inheritance & Interface>javac
CommonOperations.java

D:\javalabB18\Assignment 4 - Lab - Java Inheritance & Interface>java
CommonOperations

Sqaure=9.0

Square Root: 4.898979485566356

Square cube: 262144.0
```

6. Write Java Program to implement Java Abstract Class for below diagram

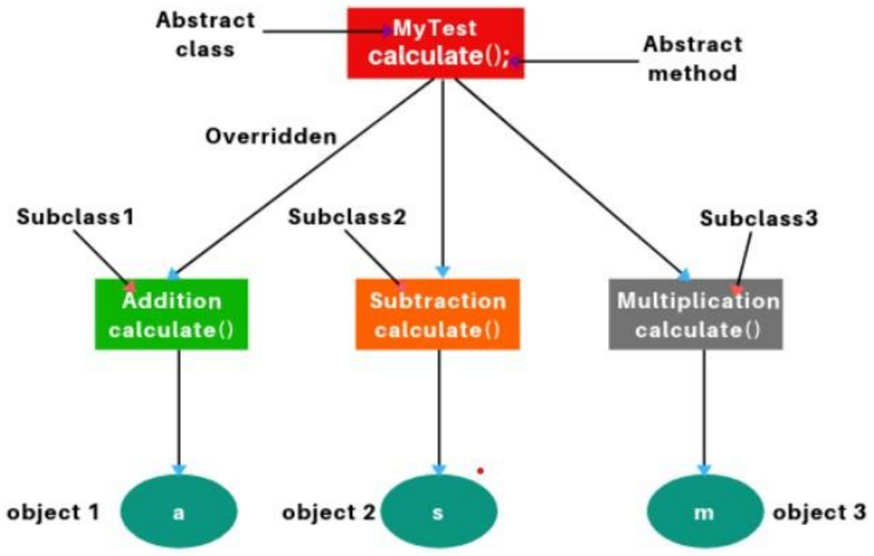

Fig: Abstract class and its subclasses

```
abstract class MyTest {
    abstract void calculate(int a, int b);
}

class Addition extends MyTest {
void calculate(int a, int b) {
        System.out.println("Addition: " + (a + b));
    }
}

class Subtraction extends MyTest {
    void calculate(int a, int b) {
        System.out.println("Subtraction: " + (a - b));
    }
}

class Multiplication extends MyTest {
void calculate(int a, int b) {
        System.out.println("Multiplication: " + (a * b));
    }
}
```

```
public class Mathsoperations {     public
static void main(String[] args) {
MyTest a = new Addition();
        MyTest s = new Subtraction();
        MyTest m = new Multiplication();

        int x = 20, y = 10;

        a.calculate(x, y);
        s.calculate(x, y);
        m.calculate(x, y);
    }
}
```

Output:

```
D:\OOP>javac Mathsoperations.java
D:\OOP>java Mathsoperations
Addition: 30
Subtraction: 10
Multiplication: 200
```

7. Write Java Program to implement Java Interface Concept. ✓ Test the PaymentSystem.java source code ✓ Add Paytm payment option in the source code

```
}

// Interface defining UPI payment structure
interface UPIPayment {     void
pay(double amount);     void
getPaymentDetails();
}

// Google Pay implementation class
GooglePay implements UPIPayment {
public void pay(double amount) {
        System.out.println("Payment of ₹" + amount + " done using Google Pay.");
    }

    public void getPaymentDetails() {
        System.out.println("Transaction ID: GPay12345");
    }
}

// PhonePe implementation class
PhonePe implements UPIPayment {
public void pay(double amount) {
        System.out.println("Payment of ₹" + amount + " done using PhonePe.");
    }
```

```
   public void getPaymentDetails() {
      System.out.println("Transaction ID: PhonePe98765");
   }
}
class Paytm implements UPIPayment {
public void pay(double amount) {
      System.out.println("Payment of ₹" + amount + " done using PhonePe.");
   }

   public void getPaymentDetails() {
      System.out.println("Transaction ID: Paytm00966");
   }
}

// add Paytm implementation logic here.

// Main class to test the interface public
class PaymentSystem {    public static
void main(String[] args) {
      // Google Pay transaction
```

```
        UPIPayment payment1 = new GooglePay();
payment1.pay(1500.0);         payment1.getPaymentDetails();

        System.out.println(); // spacer

        // PhonePe transaction
        UPIPayment payment2 = new PhonePe();
payment2.pay(2200.0);         payment2.getPaymentDetails();

                System.out.println(); // spacer

                // Paytm transaction
                UPIPayment payment3 = new Paytm();
        payment3.pay(2100.0);         payment3.getPaymentDetails();

                System.out.println();
   }
}
```

Output:

```
D:\javalabB18\Assignment 4 - Lab - Java Inheritance & Interface>javac
PaymentSystem.java

D:\javalabB18\Assignment 4 - Lab - Java Inheritance & Interface>java
PaymentSystem
```

```
Payment of ?1500.0 done using Google Pay.

Transaction ID: GPay12345

Payment of ?2200.0 done using PhonePe.

Transaction ID: PhonePe98765

Payment of ?2100.0 done using PhonePe.

Transaction ID: Paytm00966
```

8. Design java program to implement multiple Interface in Java Class Problem Source Code – MultipleInterfaceDemo.java ✓ Add PhonePe class implementation

```
// First interface for payment

interface UPIPayment {

void pay(double amount);

}


// Second interface for rewards interface

RewardPoints {

   void calculateReward(double amount);

}


// GooglePay implements both interfaces

class GooglePay implements UPIPayment, RewardPoints {
    public void pay(double amount) {
        System.out.println("Payment of ₹" + amount + " done using Google Pay.");
    }

    public void calculateReward(double amount) {          int
points = (int)(amount / 100); // 1 point per ₹100
System.out.println("Reward Points Earned: " + points);
    }
}

// PhonePe implements both interface


// Main class
public class MultipleInterfaceDemo {
    public static void main(String[] args) {
        GooglePay gpay = new GooglePay();

        gpay.pay(750.0);
gpay.calculateReward(750.0);

                    // PhonePe class Testing code
    }
```

```
}
```

Output:

```
D:\javalabB18\Assignment 4 - Lab - Java Inheritance & Interface>javac
MultipleInterfaceDemo.java

D:\javalabB18\Assignment 4 - Lab - Java Inheritance & Interface>java
MultipleInterfaceDemo

Payment of ?750.0 done using Google Pay.

Reward Points Earned: 7
```

9.Design java program to implement the Java Package. Package name = payment payment/UPIPayment.java payment/PhonePe.java PaymentTest.java (in default package) /YourProjectDirectory │ ├── PaymentTest.java └── payment/ ├── UPIPayment.java └── PhonePe.java Steps to Compile and Run the code 1. Create folder payment 2. Move (cut & Paste) two file - UPIPayment.java PhonePe.java 3. Run the below command javac -d . payment/UPIPayment.java payment/PhonePe.java PaymentTest.java

PhonePe.java

```
package payment;

public class PhonePe implements UPIPayment {

public void pay(double amount) {

        System.out.println("Paid ₹" + amount + " using PhonePe.");

    }

}
```

UPIPayment.java package

payment; public interface

```
UPIPayment {     void
pay(double amount);
}
```

PaymentTest.java import payment.PhonePe;

```
import payment.UPIPayment; public class
PaymentTest {     public static void
main(String[] args) {         UPIPayment
payment = new PhonePe();
payment.pay(2500.0);
    }
}

//Steps to Compile and Run the code
//1.     Create folder payment
//2.     Move ( cut & Paste )  two file - UPIPayment.java PhonePe.java
//3.     Run the below command
//     javac -d . payment/UPIPayment.java payment/PhonePe.java PaymentTest.java
//4.   java PaymentTest

//YourProjectDirectory
//│
//├── PaymentTest.java
```

```
// └── payment/
//    ├── UPIPayment.java
//    └── PhonePe.java
```
Output:

```
D:\OOP>javac -d . payment/UPIPayment.java payment/PhonePe.java

D:\OOP>javac PaymentTest.java

D:\OOP>java PaymentTest

Paid â??2500.0 using PhonePe.
```

10.Design and Write Java Program to create mathoperations package. ✓ Package: mathoperations ✓ Classes: • Addition • Subtraction • Multiplication • Division ✓ A main test class MathTest in the default package to use all these classes Addition.java package mathoperations;

```
public class Addition {
public int add(int a, int b) {
return a + b;
   }
}
```

Subtraction.java package

```
mathoperations; public class
Subtraction {     public int
subtract(int a, int b) {         return
a - b;
   }
```

}

Multiplication.java package

mathoperations; public class

Multiplication {   public int

multiply(int a, int b) {   return

a * b;

    }

} **Division.java** package

mathoperations; public class

Division {   public double

divide(int a, int b) {   return

(double) a / b;

    }

}

MathTest.java import

mathoperations.*; public

class MathTest {

    public static void main(String[] args) {

Addition add = new Addition();

        Subtraction sub = new Subtraction();

```
        Multiplication mul = new Multiplication();
Division div = new Division();        int a = 20,
b = 10;
        System.out.println("Addition: " + add.add(a, b));
        System.out.println("Subtraction: " + sub.subtract(a, b));
        System.out.println("Multiplication: " + mul.multiply(a, b));
        System.out.println("Division: " + div.divide(a, b));
    }
}
```

Output:

```
D:\OOP\mathoperations>javac Addition.java
D:\OOP\mathoperations>javac Subtraction.java
D:\OOP\mathoperations>javac Multiplication.java
D:\OOP\mathoperations>javac Division.java D:\OOP\mathoperations>cd..
D:\OOP>javac MathTest.java
D:\OOP>java MathTest
Addition: 30
Subtraction: 10
Multiplication: 200
Division: 2.0
```