

Name- Honey Jain

Sap id-

**OBJECT ORIENTED PROGRAMMING**

**LAB EXPERIMENT-4**

1. **Write a Java program to show that private member of a super class cannot be accessed from derived classes.**

**CODE:**

class Parent {

    int a;

    private int b;

    protected int c;

    Parent(int a, int b, int c) {

        this.a = a; this.b = b; this.c = c;}

    public void printa() {

        System.out.println("Public");}

    private void printb() {

        System.out.println("Private");}

    protected void printc() {

        System.out.println("Protected");}

class Child extends Parent{

    private int d;

    Child(int a, int b, int c, int d){

        super(a, b, c);

        this.d = d;}}

public class Exp1 {

    public static void main(String[] args) {

        Parent p = new Parent(1, 2, 3);

        p.printa();

        Child c = new Child(10, 20, 30, 40);

        c.printa();

    }

}

**OUTPUT:**



1. **Write a program in Java to create a Player class. Inherit the classes Cricket \_Player, Football \_Player and Hockey\_ Player from Player class.**

**CODE:**

class Player {

    protected String name;

    protected int age;

    protected double height;

    Player(String name, int age, double height) {

        this.name = name;

        this.age = age;

        this.height = height;

    }

    public void printDetails() {

        System.out.println("\nInfo of Player:");

        System.out.println("Name: "+name + "\n" +"Age: "+ age + "\n" + "Height: "+height);

    }

}

class Cricket\_Player extends Player {

    private String team;

    private String position;

    Cricket\_Player(String name, int age, double height, String team, String position) {

        super(name, age, height);

        this.team = team;

        this.position = position;

    }

    public void printDetails() {

        super.printDetails();

        System.out.println("Team: "+ team + "\n" +"Position: "+ position);

    }

}

class Football\_Player extends Player {

    private String team;

    private String position;

    Football\_Player(String name, int age, double height, String team, String position) {

        super(name, age, height);

        this.team = team;

        this.position = position;

    }

    public void printDetails() {

        super.printDetails();

        System.out.println("Team: "+team + "\n" +"Position: "+ position);

    }

}

class Hockey\_Player extends Player {

    private String team;

    private String position;

    Hockey\_Player(String name, int age, double height, String team, String position) {

        super(name, age, height);

        this.team = team;

        this.position = position;

    }

    public void printDetails() {

        super.printDetails();

        System.out.println("Team: "+team + "\n" +"Position: "+ position + "\n");

    }

}

public class Exp2 {

    public static void main(String[] args) {

        Player p = new Cricket\_Player("Bobby", 54, 145, "Team1", "Bowler");

        p.printDetails();

        p = new Football\_Player("Honey", 24, 335.5, "Team2", "Bowler");

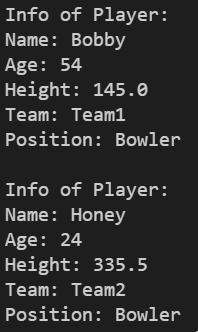
        p.printDetails();

        p = new Hockey\_Player("Sunny", 34, 64, "Team3", "Batsman");

    }

}

**OUTPUT:**



1. **Write a class Worker and derive classes DailyWorker and SalariedWorker from it. Every worker has a name and a salary rate. Write method ComPay (int hours) to compute the week pay of every worker. A Daily Worker is paid on the basis of the number of days he/she works. The Salaried Worker gets paid the wage for 40 hours a week no matter what the actual hours are. Test this program to calculate the pay of workers. You are expected to use the concept of polymorphism to write this program.**

**CODE:**

class Worker {

    protected String name;

    protected double salaryRate;

    Worker(String name, double salaryRate) {

        this.name = name;

        this.salaryRate = salaryRate;

    }

}

class SalariedWorker extends Worker {

    private int hoursWorked;

    SalariedWorker(String name, double salaryRate, int hoursWorked) {

        super(name, salaryRate);

        this.hoursWorked = hoursWorked;

    }

    public double ComPay() {

        double salary = 40 \* salaryRate;

        return salary;

    }

}

class DailyWorker extends Worker {

    private int daysWorked;

    DailyWorker(String name, double salaryRate, int daysWorked) {

        super(name, salaryRate);

        this.daysWorked = daysWorked;

    }

    public double ComPay() {

        double salary = daysWorked \* salaryRate;

        return salary;

    }

}

public class Exp3 {

    public static void main(String[] args) {

        SalariedWorker one = new SalariedWorker("1", 686, 86);

        DailyWorker two = new DailyWorker("2", 656, 86);

        System.out.println("Salaried : "+one.ComPay());

        System.out.println("Daily : "+two.ComPay());

    }

}

**OUTPUT:**



1. **Consider the trunk calls of a telephone exchange. A trunk call can be ordinary, urgent or lightning. The charges depend on the duration and the type of the call. Write a program using the concept of polymorphism in Java to calculate the charges.**

**CODE:**

class TrunkCall {

    protected int duration;

    TrunkCall(int duration) {this.duration = duration;}

    public int getDuration() {return this.duration;}

    public double cost() {return 0.0;}

}

class Ordinary extends TrunkCall {

    private static final double charge = 1.0;

    Ordinary(int duration) {super(duration);}

    public double cost() {return charge \* duration;}

}

class Urgent extends TrunkCall {private static final double charge = 3.0;

    Urgent(int duration) {

        super(duration);

    }

    public double cost() {

        return charge \* duration;

    }

}

class Lightning extends TrunkCall {

    private static final double charge = 5;

    Lightning(int duration) {

        super(duration);

    }

    public double cost() {

        return charge \* duration;

    }

}

public class Exp4 {

    public static void main(String[] args) {

        TrunkCall x = new Ordinary(25);

        TrunkCall f = new Urgent(25);

        TrunkCall j = new Lightning(25);

        System.out.println("Ordinary: "+x.cost());

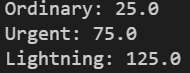
        System.out.println("Urgent: "+f.cost());

        System.out.println("Lightning: "+j.cost());

    }

}

**OUTPUT:**



1. **Design a class employee of an organization. An employee has a name, empid, and salary. Write the default constructor, a constructor with parameters (name, empid, and salary) and methods to return name and salary. Also write a method increaseSalary that raises the employee’s salary by a certain user specified percentage. Derive a subclass Manager from employee. Add an instance variable named department to the manager class. Supply a test program that uses theses classes and methods.**

**CODE:**

class Employee {

    protected String name;

    protected int empid;

    protected double salary;

    Employee(String name, int empid, double salary) {this.name = name;

        this.empid = empid;

        this.salary = salary;

    }

    public double getSalary() {return this.salary;}

    public String getName() {

        return this.name;}

    public void increaseSalary(double percentage) {this.salary += salary \* percentage / 100;}

}

class Manager extends Employee {private String department;

    Manager(String name, int empid, double salary, String department) {super(name, empid, salary);

        this.department = department;}

    public String getDepartment() {return this.department;}

}

public class Exp5 {

    public static void main(String[] args) {

        Employee A = new Employee("Mukesh", 576, 67.00);

        Manager B = new Manager("Suresh", 51, 86.96, "1");

        System.out.println("Person: "+A.getName()+"\nIncome: "+A.getSalary()+"\n\n");

        System.out.println("Person: "+B.getName()+"\nIncome: "+B.getSalary());

    }

}

**OUTPUT:**

