School of Computer Science

LABORATORY FILE

Object-Oriented Programming Systems

B.Tech. - IV Sem (2025)

Submitted by

Name:Soumya Aggarwal

SAP ID: 500124313

Batch: 1 IOT

Lab Experiment: 05 (Inheritance)

1. Write a Java program to demonstrate that a private member of a superclass cannot be accessed directly from a derived class.

fibo.java U | Inheritance_1.java 1, U | Sum.java | ESP-IDF: Search Error

Inheritance_1.java > Language Support for Java(TM) by Red Hat > Inheritance_1 > main(String[])

```java
class SuperClass {
    private int privateVar = 10;  // Private member

    public int getPrivateVar() {  // Getter method
        return privateVar;
    }
}

class SubClass extends SuperClass {
    public void display() {
        //System.out.println(privateVar); // This will cause an error (private access)
        System.out.println("Private variable accessed via method: " + getPrivateVar());
    }
}

public class Inheritance_1 {
    Run | Debug | Run main | Debug main
    public static void main(String[] args) {
        SubClass obj = new SubClass();
        obj.display();
    }
}
```

OUTPUT | GITLENS | SQL CONSOLE | ESP-IDF | DEBUG CONSOLE | PORTS | PROBLEMS 2

TERMINAL

```
soumyaaggarwal@Soumyas-MacBook-Air CODES %  /usr/bin/env /opt/homebrew/Cellar/openjdk/23.0.1/libexec/openjdk.jdk/Con
tents/Home/bin/java --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/soumyaaggarwal/Library/Appli
cation\ Support/Code/User/workspaceStorage/3f33e64210add55743701586641762b1/redhat.java/jdt_ws/CODES_b47bddb2/bin In
heritance_1
Private variable accessed via method: 10
soumyaaggarwal@Soumyas-MacBook-Air CODES %
```

2. Create a Java program with a Player class and derive three subclasses: Cricket_Player, Football_Player, and Hockey_Player. Implement attributes such as name, age, and position, and methods like play() and train() to represent these players.

```java
class Player {
    String name;
    int age;
    String position;
    public Player(String name, int age, String position) {
        this.name = name;
        this.age = age;
        this.position = position;}
    public void play() {
        System.out.println(name + " is playing.");}
    public void train() {
        System.out.println(name + " is training.");}}

class Cricket_Player extends Player {
    public Cricket_Player(String name, int age, String position) {
        super(name, age, position);}@Override
    public void play() {System.out.println(name + " is batting or bowling.");}}
class Football_Player extends Player {
    public Football_Player(String name, int age, String position) {
        super(name, age, position);}@Override
    public void play() {System.out.println(name + " is dribbling and shooting at " + position);}}

class Hockey_Player extends Player {
    public Hockey_Player(String name, int age, String position) { super(name, age, position);}
 @Override
    public void play() {
        System.out.println(name + " is handling the puck at " + position);}}
        public class PlayerDemo {
            public static void main(String[] args) {
        Cricket_Player cp = new Cricket_Player(name:"Virat", age:34, position:"Batsman");   // Cricket_Player this is class, Cricket_Player() = Pa
        Football_Player fp = new Football_Player(name:"Messi", age:36, position:"Forward");
        Hockey_Player hp = new Hockey_Player(name:"Dhyan", age:28, position:"Midfielder");
        cp.play();
        cp.train();
        fp.play();
        fp.train();
        hp.play();
        hp.train();}}
```

```
TERMINAL

soumyaaggarwal@Soumyas-MacBook-Air CODES %  /usr/bin/env /opt/homebrew/Cellar/openjdk/23.0.1/libexec/openjdk.jdk/Contents/Home/bin/java --enable-preview
 -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/soumyaaggarwal/Library/Application\ Support/Code/User/workspaceStorage/3f33e64210add5574370158664176
2b1/redhat.java/jdt_ws/CODES_b47bddb2/bin PlayerDemo
Virat is batting or bowling.
Virat is training.
Messi is dribbling and shooting at Forward
Messi is training.
Dhyan is handling the puck at Midfielder
Dhyan is training.
soumyaaggarwal@Soumyas-MacBook-Air CODES %  cd /Users/soumyaaggarwal/Documents/Sem\ 4/OOPs\ JAVA/CODES ; /usr/bin/env /opt/homebrew/Cellar/openjdk/23.0.
1/libexec/openjdk.jdk/Contents/Home/bin/java --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/soumyaaggarwal/Library/Application\ Sup
port/Code/User/workspaceStorage/3f33e64210add5574370158664176 2b1/redhat.java/jdt_ws/CODES_b47bddb2/bin PlayerDemo
Virat is batting or bowling.
Virat is training.
Messi is dribbling and shooting at Forward
Messi is training.
Dhyan is handling the puck at Midfielder
Dhyan is training.
soumyaaggarwal@Soumyas-MacBook-Air CODES %
```

3. Define a Worker class with DailyWorker and SalariedWorker as its subclasses. Each worker has a name and salary rate. Implement a method computePay(int hours) to compute weekly pay. DailyWorker is paid based on the number of days worked (assuming 8 hours per day), whereas SalariedWorker receives a fixed wage for 40 hours per week, regardless of actual hours worked. Use polymorphism to implement this program and test worker salary calculations.

WorkerDemo.java > Language Support for Java(TM) by Red Hat > WorkerDemo > main(String[])

```java
class Worker {
    String name;
    double salaryRate;

    public Worker(String name, double salaryRate) {
        this.name = name;
        this.salaryRate = salaryRate;
    }

    public double computePay(int hours) {
        return 0; // Overridden by subclasses
    }
}
class DailyWorker extends Worker {
    public DailyWorker(String name, double salaryRate) {
        super(name, salaryRate);
    }

    @Override
    public double computePay(int hours) {
        int daysWorked = hours / 8; // Assuming 8 hours per day
        return daysWorked * salaryRate;
    }
}
class SalariedWorker extends Worker {
    public SalariedWorker(String name, double salaryRate) {
        super(name, salaryRate);
    }
    @Override
    public double computePay(int hours) {
        return 40 * salaryRate; // Fixed pay for 40 hours per week
    }
}
public class WorkerDemo {
    Run | Debug | Run main | Debug main
    public static void main(String[] args) {
        Worker dw = new DailyWorker(name:"John", salaryRate:100); // Paid per day
        Worker sw = new SalariedWorker(name:"Alice", salaryRate:50); // Fixed weekly pay

        System.out.println(dw.name + "'s Weekly Pay: $" + dw.computePay(hours:48)); // 48 hours worked (6 days)
        System.out.println(sw.name + "'s Weekly Pay: $" + sw.computePay(hours:60)); // 60 hours but fixed salary
    }}
```

```
soumyaaggarwal@Soumyas-MacBook-Air CODES %  /usr/bin/env
 -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/soum
2b1/redhat.java/jdt_ws/CODES_b47bddb2/bin WorkerDemo
John's Weekly Pay: $600.0
Alice's Weekly Pay: $2000.0
soumyaaggarwal@Soumyas-MacBook-Air CODES %
```

3. Implement a Java program to calculate trunk call charges based on duration and type (Ordinary, Urgent, or Lightning). Use polymorphism to manage different charge rates for each type. Implement a Java program to calculate trunk call charges based on duration (in minutes) and type (Ordinary, Urgent, or Lightning). Use polymorphism to manage different charge rates for each type. The program should take user input for duration and type and display the total charge.

```java
import java.util.Scanner;
// Base class
abstract class TrunkCall {
    protected int duration;
    public TrunkCall(int duration) {
        this.duration = duration;}
    // Abstract method to calculate charges
    public abstract double calculateCharge();}
// Ordinary call class
class OrdinaryCall extends TrunkCall {
    private static final double RATE = 1.0; // Example rate per minute
    public OrdinaryCall(int duration) {
        super(duration);}
    @Override
    public double calculateCharge() {
        return duration * RATE;}}
// Urgent call class
class UrgentCall extends TrunkCall {
    private static final double RATE = 2.0;
    public UrgentCall(int duration) {
        super(duration);
    }
    @Override
    public double calculateCharge() {
        return duration * RATE; }}
// Lightning call class
class LightningCall extends TrunkCall {
    private static final double RATE = 3.0;
    public LightningCall(int duration) {
        super(duration); }
    @Override
    public double calculateCharge() {
        return duration * RATE;}}
// Main class
public class TrunkCallCalculator {
    Run | Debug | Run main | Debug main
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println(x:"Enter call duration in minutes:");
        int duration = scanner.nextInt();

        System.out.println(x:"Enter call type (Ordinary, Urgent, Lightning):");
        String callType = scanner.next();

        TrunkCall call;

        switch (callType.toLowerCase()) {
            case "ordinary":
                call = new OrdinaryCall(duration);
                break;
            case "urgent":
                call = new UrgentCall(duration);
                break;
            case "lightning":
                call = new LightningCall(duration);
                break;
            default:
                System.out.println(x:"Invalid call type!");
                scanner.close();
                return;
        }

        System.out.println("Total charge: Rs. " + call.calculateCharge());
        scanner.close();
    }
}
```

```
 /usr/bin/env /opt/homebrew/Cellar/openjdk/23.0.1/libexe
p /Users/soumyaaggarwal/Library/Application\ Support/Coc
runkCallCalculator
soumyaaggarwal@Soumyas-MacBook-Air CODES %  /usr/bin/env
 -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/soum
2b1/redhat.java/jdt_ws/CODES_b47bddb2/bin TrunkCallCalcu
Enter call duration in minutes:
45
Enter call type (Ordinary, Urgent, Lightning):
urgent
Total charge: Rs. 90.0
soumyaaggarwal@Soumyas-MacBook-Air CODES %
```

5. Design a Java class Employee with attributes name, empid, and salary. Implement a default constructor, a parameterized constructor, and methods to return the employee's name and salary. Add a method increaseSalary(double percentage) to raise the salary by a userspecified percentage. Create a subclass Manager with an additional instance variable department. Develop a test program to validate these functionalities.

EmployeeTest.java > ...

```java
import java.util.Scanner;

// Employee class
class Employee {
    protected String name;
    protected int empId;
    protected double salary;

    // Default constructor
    public Employee() {
        this.name = "Unknown";
        this.empId = 0;
        this.salary = 0.0;
    }

    // Parameterized constructor
    public Employee(String name, int empId, double salary) {
        this.name = name;
        this.empId = empId;
        this.salary = salary;
    }

    public String getName() {
        return name;
    }

    public double getSalary() {
        return salary;
    }

    public void increaseSalary(double percentage) {
        if (percentage > 0) {
            salary += salary * (percentage / 100);
        }
    }
}

// Manager subclass
class Manager extends Employee {
    private String department;

    public Manager(String name, int empId, double salary, String department) {
        super(name, empId, salary);
        this.department = department;
    }
```

```java
    public String getDepartment() {
        return department;
    }
}

// Test program
public class EmployeeTest {
    Run | Debug | Run main | Debug main
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println(x:"Enter Employee Name:");
        String name = scanner.nextLine();

        System.out.println(x:"Enter Employee ID:");
        int empId = scanner.nextInt();

        System.out.println(x:"Enter Employee Salary:");
        double salary = scanner.nextDouble();

        Employee emp = new Employee(name, empId, salary);

        System.out.println(x:"Enter percentage increase in salary:");
        double percentage = scanner.nextDouble();
        emp.increaseSalary(percentage);

        System.out.println("Updated Salary of " + emp.getName() + " is: Rs. " + emp.getSalary());

        // Testing Manager subclass
        scanner.nextLine(); // Consume newline
        System.out.println(x:"Enter Manager Department:");
        String department = scanner.nextLine();

        Manager manager = new Manager(name, empId, salary, department);
        System.out.println("Manager " + manager.getName() + " heads " + manager.getDepartment() + " department.");

        scanner.close();
    }
}
```

```
 /usr/bin/env /opt/homebrew/Cellar/openjdk/23.0.1/libexec/openj
p /Users/soumyaaggarwal/Library/Application\ Support/Code/User/
mployeeTest
soumyaaggarwal@Soumyas-MacBook-Air CODES %  /usr/bin/env /opt/h
 -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/soumyaaggar
2b1/redhat.java/jdt_ws/CODES_b47bddb2/bin EmployeeTest
Enter Employee Name:
soumya
Enter Employee ID:
500124313
Enter Employee Salary:
50000
Enter percentage increase in salary:
12
Updated Salary of soumya  is: Rs. 56000.0
Enter Manager Department:
vinod
Manager soumya  heads vinod department.
soumyaaggarwal@Soumyas-MacBook-Air CODES %
```

Additional Questions:

6. A vehicle manufacturing company produces different types of vehicles, such as cars and motorcycles. The base class Vehicle contains common properties like brand, model, and price. The class Car extends Vehicle by adding attributes like seatingCapacity and fuelType. Further, a subclass ElectricCar extends Car, introducing additional attributes like batteryCapacity and chargingTime. The Motorcycle class extends Vehicle and adds engineCapacity and type (e.g., "Sport", "Cruiser"). Implement this vehicle hierarchy system using multilevel inheritance in Java. Use constructor chaining to initialize attributes efficiently and demonstrate polymorphism by overriding a method displayDetails() in each Subclass.

```java
    public ElectricCar(String brand, String model, double price, int seatingCapacity, String fuelType, int batteryCapacity, double chargingTime)
        super(brand, model, price, seatingCapacity, fuelType);
        this.batteryCapacity = batteryCapacity;
        this.chargingTime = chargingTime;
    }

    @Override
    public void displayDetails() {
        super.displayDetails();
        System.out.println("Battery Capacity: " + batteryCapacity + " kWh, Charging Time: " + chargingTime + " hours");
    }
}

// Motorcycle class extending Vehicle
class Motorcycle extends Vehicle {
    private int engineCapacity;
    private String type;

    public Motorcycle(String brand, String model, double price, int engineCapacity, String type) {
        super(brand, model, price);
        this.engineCapacity = engineCapacity;
        this.type = type;
    }

    @Override
    public void displayDetails() {
        super.displayDetails();
        System.out.println("Engine Capacity: " + engineCapacity + " cc, Type: " + type);
    }
}

// Main class to test the hierarchy
public class VehicleTest {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Creating and displaying an ElectricCar
        ElectricCar tesla = new ElectricCar("Tesla", "Model S", 75000, 5, "Electric", 100, 1.5);
        System.out.println("Electric Car Details:");
        tesla.displayDetails();

        // Creating and displaying a Motorcycle
        Motorcycle yamaha = new Motorcycle("Yamaha", "R1", 20000, 998, "Sport");
```

```java
        // Creating and displaying an ElectricCar
        ElectricCar tesla = new ElectricCar(brand:"Tesla", model:"Model S", price:75000, seatingCapacity:5, fuelType:"Electric", batte
        System.out.println(x:"Electric Car Details:");
        tesla.displayDetails();

        // Creating and displaying a Motorcycle
        Motorcycle yamaha = new Motorcycle(brand:"Yamaha", model:"R1", price:20000, engineCapacity:998, type:"Sport");
        System.out.println(x:"\nMotorcycle Details:");
        yamaha.displayDetails();
    }
}
}
```

OUTPUT GITLENS SQL CONSOLE ESP-IDF DEBUG CONSOLE PORTS PROBLEMS 13

TERMINAL

```
 /usr/bin/env /opt/homebrew/Cellar/openjdk/23.0.1/libexec/openjdk.jdk/Contents/Home/bin/java --enable-preview -XX:+ShowCodeDetailsInExceptionMess
p /Users/soumyaaggarwal/Library/Application\ Support/Code/User/workspaceStorage/3f33e64210add55743701586641762b1/redhat.java/jdt_ws/CODES_b47bddb
ehicleTest
soumyaaggarwal@Soumyas-MacBook-Air CODES %  /usr/bin/env /opt/homebrew/Cellar/openjdk/23.0.1/libexec/openjdk.jdk/Contents/Home/bin/java --enable-
 -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/soumyaaggarwal/Library/Application\ Support/Code/User/workspaceStorage/3f33e64210add557437015
2b1/redhat.java/jdt_ws/CODES_b47bddb2/bin VehicleTest
Electric Car Details:
Brand: Tesla, Model: Model S, Price: Rs. 75000.0
Seating Capacity: 5, Fuel Type: Electric
Battery Capacity: 100 kWh, Charging Time: 1.5 hours

Motorcycle Details:
Brand: Yamaha, Model: R1, Price: Rs. 20000.0
Engine Capacity: 998 cc, Type: Sport
soumyaaggarwal@Soumyas-MacBook-Air CODES %
```