Experiment Number : 05 **Date:**

INHERITANCE

Aim:

To understand and implement inheritance concepts in Java.

PRE LAB EXERCISE

QUESTIONS

✓ What is inheritance?

Inheritance is an object-oriented programming concept where a **child (subclass)** acquires the **properties and methods** of a **parent (superclass)**.
It helps in building new classes using existing ones.

✓ What is code reusability?

Code reusability means **using existing code again** without rewriting it.
It saves **time**, reduces **errors**, and makes programs **easy to maintain**.

✓ What is the use of extends keyword?

The **extends keyword** is used to:

- Create a **subclass** from a **superclass**
- Inherit **variables and methods** of the parent class
- Achieve **inheritance** in Java

IN LAB EXERCISE

Objective:

To implement all types of inheritance.

PROGRAMS:

Student Result System (Single Inheritance)

Question:
A school wants to store student details and calculate marks. Create a base class Student and a derived class Result.

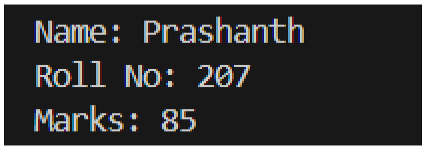
Code:

```
class Student {
    String name;
    int rollNo;

    void getDetails() {
        name = "Prashanth";
        rollNo = 207;
    }
}

class Result extends Student {
    int marks = 85;

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

```
public class Main {
    public static void main(String[] args) {
        Result r = new Result();
        r.getDetails();
        r.display();
    }
}
```

Output:

```
Name: Prashanth
Roll No: 207
Marks: 85
```



```
Name: Prashanth
Roll No: 207
Marks: 85
```

2. **Bank Account System (Hierarchical Inheritance)**

Question:
A bank has Savings and Current accounts. Both inherit from a common Account class.

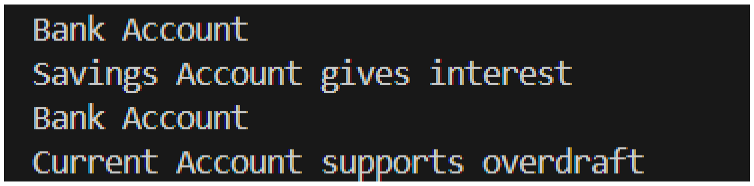
Code:

```
class Account {
    void showAccountType() {
        System.out.println("Bank Account");
    }
}
```

```
class SavingsAccount extends Account {
   void interest() {
      System.out.println("Savings Account gives interest");
   }
}

class CurrentAccount extends Account {
   void overdraft() {
      System.out.println("Current Account supports overdraft");
   }
}

public class Main {
   public static void main(String[] args) {
      SavingsAccount s = new SavingsAccount();
      CurrentAccount c = new CurrentAccount();

      s.showAccountType();
      s.interest();

      c.showAccountType();
      c.overdraft();
   }
}
```

Output:

```
Bank Account

Savings Account gives interest

Bank Account

Current Account supports overdraft
```



```
Bank Account
Savings Account gives interest
Bank Account
Current Account supports overdraft
```

3. Vehicle System (Multilevel Inheritance)

Question:
A company classifies vehicles as Vehicle → Car → ElectricCar.

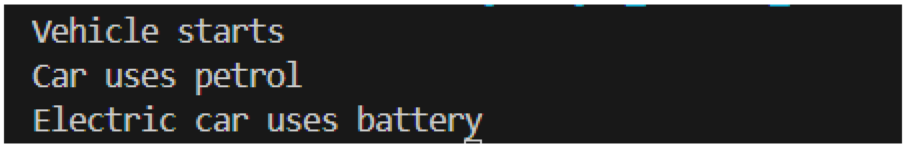
Code:

```
class Vehicle {

   void start() {

       System.out.println("Vehicle starts");

   }

}


class Car extends Vehicle {

   void fuelType() {

       System.out.println("Car uses petrol");

   }

}


class ElectricCar extends Car {

   void battery() {

       System.out.println("Electric car uses battery");
```

```
    }
}

public class Main {
    public static void main(String[] args) {
        ElectricCar e = new ElectricCar();
        e.start();
        e.fuelType();
        e.battery();
    }
}
```

Output:

```
Vehicle starts
Car uses petrol
Electric car uses battery
```



```
Vehicle starts
Car uses petrol
Electric car uses battery
```

POST LAB EXERCISE

- ✓ Why Java does not support multiple inheritance using classes and how it is implemented?

Java does **not support multiple inheritance using classes** to avoid **ambiguity and complexity**, mainly the **Diamond Problem** (when two parent classes have the same method and the compiler can't decide which one to use).

How it is implemented:

Java achieves multiple inheritance using **interfaces**, where:

- Methods are **abstract by default**
- A class can implement **multiple interfaces**

  This avoids ambiguity and ensures clear method implementation.

✓ What is the role of the super keyword? Give examples.

The **super keyword** is used to refer to the **parent class object**.

Roles of super:

- Access parent class variables
- Call parent class methods
- Invoke parent class constructors

```
class Parent {

   int a = 10;

}



class Child extends Parent {

   void show() {

      System.out.println(super.a);
```

```
    }
}
```

- ✓ Can a child class access private members of the parent class? Why?

No, a child class **cannot directly access private members** of the parent class.

Why?

- private members are accessible **only within the same class**
- This maintains **data encapsulation and security**

However, private members can be accessed **indirectly** using **public or protected getter methods**.

- ✓ Explain why hybrid inheritance is not supported in Java.

Java does **not support hybrid inheritance using classes** because it combines **multiple and multilevel inheritance**, which can cause:

- **Method ambiguity**
- Increased complexity
- Diamond Problem

To keep the language **simple and secure**, Java restricts hybrid inheritance with classes and supports it **only through interfaces**

Result:

Thus the different types of inheritance were implemented and executed successfully.

ASSESSMENT

Description	**Max Marks**	**Marks Awarded**
Pre Lab Exercise	**5**	
In Lab Exercise	**10**	
Post Lab Exercise	**5**	
Viva	**10**	
Total	**30**	
	Faculty Signature	