Experiment Number : 05 **Date: 4.2.2026**

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 one class acquires the properties and methods of another class. The class that inherits is called the **child (sub) class**, and the class being inherited from is called the **parent (super) class**. Inheritance helps in reducing code duplication and improving program structure.

- ✓ What is code reusability?

  Code reusability means **using the same code multiple times** without writing it again. Instead of creating new code, existing classes, methods, or functions are reused. This saves time, reduces errors, and makes the program easier to maintain.

- ✓ What is the use of extends keyword?

  The extends keyword is used to **implement inheritance in Java**. It allows one class to inherit the properties and methods of another class. By using extends, the child class can reuse the code of the parent class and also add new features or override existing ones.

  Example:

  ```
  class Child extends Parent {
  }
  ```

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.

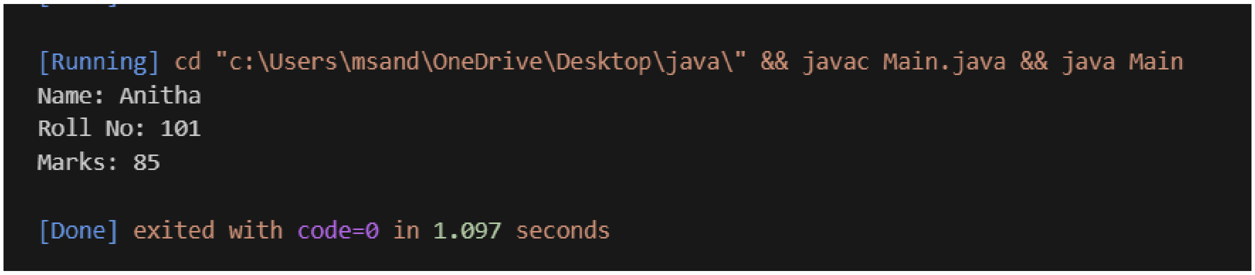
Code:

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

    void getDetails() {
        name = "Anitha";
        rollNo = 101;
    }
}

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: Anitha
Roll No: 101
Marks: 85
```



```
[Running] cd "c:\Users\msand\OneDrive\Desktop\java\" && javac Main.java && java Main
Name: Anitha
Roll No: 101
Marks: 85

[Done] exited with code=0 in 1.097 seconds
```

2. Bank Account System (Hierarchical Inheritance)

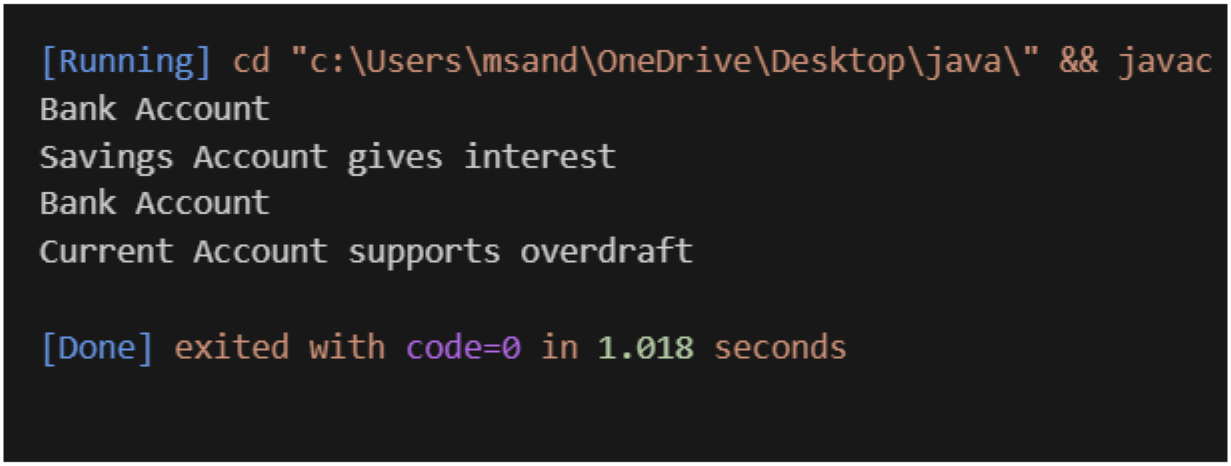
Question:
A bank has Savings and Current accounts. Both inherit from a common Account class.

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
```



```
[Running] cd "c:\Users\msand\OneDrive\Desktop\java\" && javac
Bank Account
Savings Account gives interest
Bank Account
Current Account supports overdraft

[Done] exited with code=0 in 1.018 seconds
```

3. Vehicle System (Multilevel Inheritance)

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

Code:

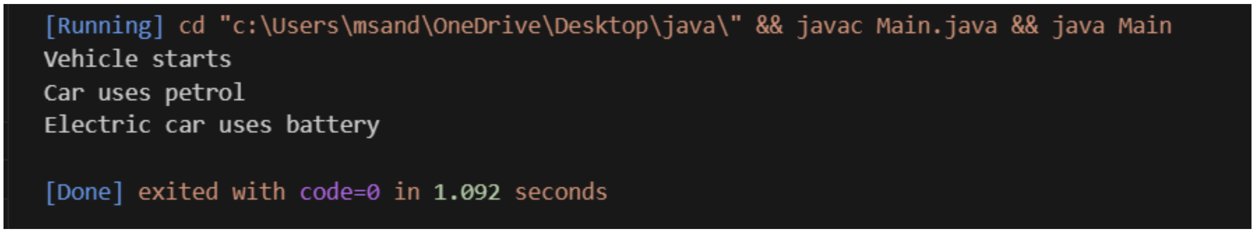
```
class Vehicle {
    void start() {
        System.out.println("Vehicle starts");
    }
}

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



```
[Running] cd "c:\Users\msand\OneDrive\Desktop\java\" && javac Main.java && java Main
Vehicle starts
Car uses petrol
Electric car uses battery

[Done] exited with code=0 in 1.092 seconds
```

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.
- It prevents the *diamond problem*.
- Method name conflicts can confuse the compiler.
- Java implements multiple inheritance using **interfaces**.
- Interfaces provide a clear and safe way to achieve multiple inheritance.

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

- Refers to the immediate parent class object.
- Used to access parent class variables.
- Used to call parent class methods.

- Used to invoke parent class constructor.
- Avoids confusion between parent and child class members.

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

- No, child class cannot access private members directly.
- `private` members are accessible only within the same class.
- This ensures data security and encapsulation.
- Private members can be accessed indirectly using public or protected methods.

✓ Explain why hybrid inheritance is not supported in Java.

- Hybrid inheritance involves multiple inheritance.
- Multiple inheritance causes ambiguity problems.
- Java avoids complexity and confusion.
- Hence, hybrid inheritance is not supported using classes.
- Interfaces are used instead to achieve similar functionality.

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	