Experiment Number : 05 **Date: 05.02.2026**

INHERITANCE

Aim:

To understand and implement inheritance concepts in Java.

PRE LAB EXERCISE

QUESTIONS

✓ What is inheritance?

- Inheritance is an object-oriented concept where one class acquires the properties and methods of another class.
- The existing class is called the parent or superclass.
- The new class is called the child or subclass.
- It helps in code reusability and better organization.

✓ What is code reusability?

- Code reusability means using existing code again without rewriting it.
- It reduces redundancy in programs.
- It saves time and effort in development.
- It makes the program easier to maintain.

✓ What is the use of extends keyword?

- The extends keyword is used to inherit a class in Java.
- It allows a subclass to access properties and methods of the superclass.
- It supports code reusability.
- It represents an *is-a* relationship between classes.

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.

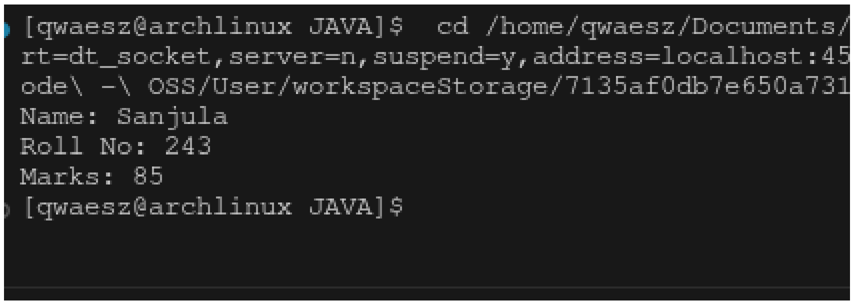
Code:

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

   void getDetails() {
      name = "Sanjula";
      rollNo = 243;
   }
}

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: Sanjula
Roll No: 243
Marks: 85
```



```
[qwaesz@archlinux JAVA]$  cd /home/qwaesz/Documents/
rt=dt_socket,server=n,suspend=y,address=localhost:45
ode\ -\ OSS/User/workspaceStorage/7135af0db7e650a731
Name: Sanjula
Roll No: 243
Marks: 85
[qwaesz@archlinux JAVA]$
```

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

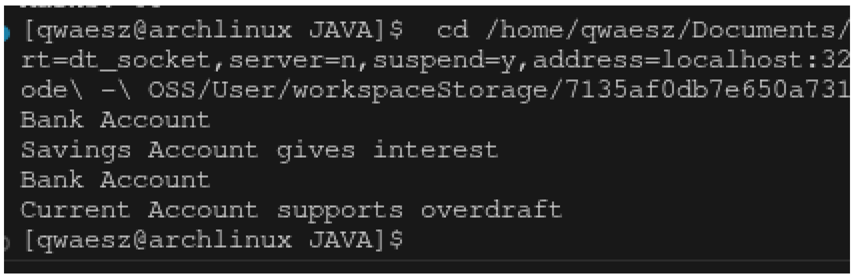
Question:
A bank has Savings and Current accounts. Both inherit from a common Account class.

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
```



```
[qwaesz@archlinux JAVA]$  cd /home/qwaesz/Documents/
rt=dt_socket,server=n,suspend=y,address=localhost:32
ode\ -\ OSS/User/workspaceStorage/7135af0db7e650a731
Bank Account
Savings Account gives interest
Bank Account
Current Account supports overdraft
[qwaesz@archlinux JAVA]$
```

3. Vehicle System (Multilevel Inheritance)

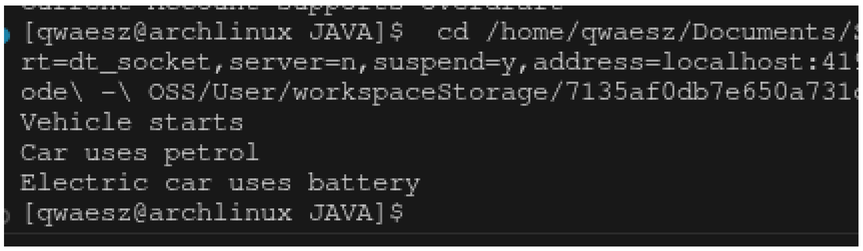
Question:
A company classifies vehicles as Vehicle → Car → ElectricCar.

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
```



```
[qwaesz@archlinux JAVA]$  cd /home/qwaesz/Documents/
rt=dt_socket,server=n,suspend=y,address=localhost:41
ode\ -\ OSS/User/workspaceStorage/7135af0db7e650a731
Vehicle starts
Car uses petrol
Electric car uses battery
[qwaesz@archlinux JAVA]$
```

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.
- If two parent classes have methods with the same name, the compiler cannot decide which one to inherit (diamond problem).
- This can lead to complexity and confusion in method execution.
- Java achieves multiple inheritance using **interfaces**, where method conflicts are handled using implementation rules.

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

- The super keyword is used to refer to the immediate parent class object.
- It is used to access parent class variables.
- It is used to call parent class methods.
- It is used to invoke the parent class constructor.

```
class Parent {
   int x = 10;
}

class Child extends Parent {
   int x = 20;
   void display() {
      System.out.println(super.x); // accesses parent variable
   }
}
```

✓ 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.
- Private members are accessible only within the same class.
- This supports data hiding and encapsulation.
- They can be accessed indirectly using public or protected methods.

✓ Explain why hybrid inheritance is not supported in Java.

- Hybrid inheritance involves a combination of multiple inheritance types.
- It can create ambiguity when two parent classes have the same method.
- This leads to the diamond problem.
- To maintain simplicity and avoid confusion, Java does not support hybrid inheritance using classes.

- However, it can be achieved using 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		