DAY4_ASSIGNMENT:

1.Create multilevel inheritance for

```
//Vehicle
//Four_wheeler
//Petrol_Four_Wheeler
//FiveSeater_Petrol_Four_Wheeler
//Baleno_FiveSeater_Petrol_Four_Wheeler
```

A. **package** Day4;

```
class Vehicle {

   void type() {

      System.out.println("Vehicle");

   }

}

class Four_wheeler extends Vehicle {

   void wheels() {

      System.out.println("4 wheelers");

   }

}

class Petrol_Four_Wheeler extends Four_wheeler {

   void fuel() {

      System.out.println("Petrol fuel");

   }

}

class FiveSeater_Petrol_Four_Wheeler extends Petrol_Four_Wheeler {

   void seats() {

      System.out.println("Five seater");

   }
```

```
}

class Baleno_FiveSeater_Petrol_Four_Wheeler extends
FiveSeater_Petrol_Four_Wheeler {

   void model() {

      System.out.println("Baleno model");

   }

}

public class Multilevel_Inheritance {

        public static void main(String[] args) {

                Baleno_FiveSeater_Petrol_Four_Wheeler baleno = new
Baleno_FiveSeater_Petrol_Four_Wheeler();

      baleno.type();

      baleno.wheels();

      baleno.fuel();

      baleno.seats();

      baleno.model();

        }
```

2.Demonstrate the use of the super keyword.

A. The super keyword in Java is used to refer to the immediate parent class of a subclass. It helps you access members (variables, methods, constructors) from the parent class when they are hidden or overridden in the child class.

Use:

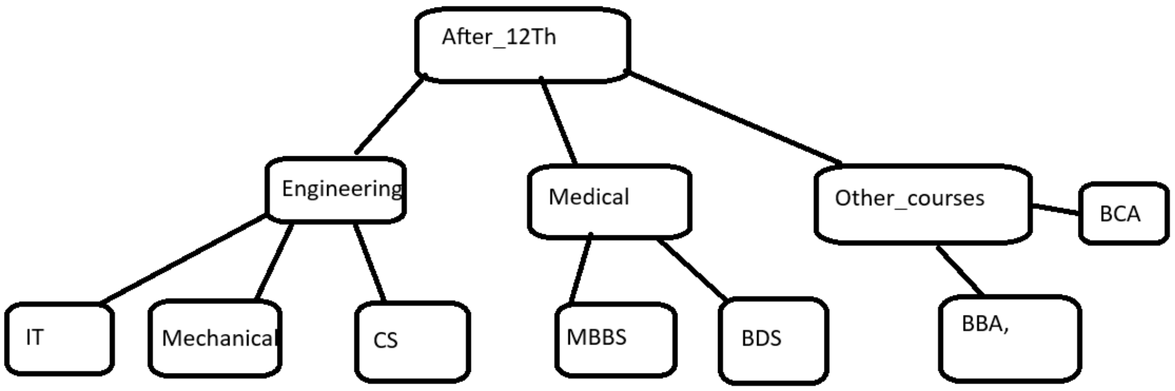
- To avoid method/variable overriding confusion.
- To invoke parent class constructor explicitly.
- To reuse parent class methods and fields.
- Must be the first statement in a constructor.
- Cannot be used in static methods.
- Improves code reusability and clarity.

3.Create Hospital super class and access this class inside the patient child class and access properties from Hospital class.

A.
```
package Day4;
class Hospital {
    void hospitalName() {
        System.out.println("City Care Hospital");
    }
    void location() {
        System.out.println("Hyderabad");
    }
}
class Patient extends Hospital {
    void patientName() {
        System.out.println("Suma");
    }
    void age() {
        System.out.println("23");
    }
}
public class HospitalDemo_superclass {

    public static void main(String[] args) {
        Patient p = new Patient();
        p.patientName();
        p.age();
        p.hospitalName();
        p.location();
    }
}
```

4.Create Hierarchical inheritance



A. package interface_p;

```
interface Doctor {
   void operation();
   void ODP();
}
interface Nurse {
   void dailyCheck();
   void documentation();
}
interface Accountant {
   void payment();
   void query();
}
class Patient implements Doctor, Nurse, Accountant {
   public void operation() {
      System.out.println("Doctor is performing operation.");
   }
   public void ODP() {
      System.out.println("Doctor is doing OPD duty.");
   }
   public void dailyCheck() {
      System.out.println("Nurse is doing daily check-up.");
   }
   public void documentation() {
      System.out.println("Nurse is updating documents.");
   }
   public void payment() {
      System.out.println("Accountant is processing payment.");
   }
   public void query() {
      System.out.println("Accountant is handling billing queries.");
   }
}
public class Hospitaldemo {
    public static void main(String[] args) {
        Patient p = new Patient();
        p.operation();
        p.ODP();
        p.dailyCheck();
        p.documentation();
        p.payment();
        p.query();
    }
}
```

Polymorphism:

1.Create a class Calculator with the following overloaded add()

1.add(int a, int b)

2.add(int a, int b, int c)

3.add(double a, double b)

A. **package** Day4;

```
public class Calculator {
    int add(int a, int b) {
        return a + b;
    }
    int add(int a, int b, int c) {
        return a + b + c;
    }
    double add(double a, double b) {
        return a + b;
    }

    public static void main(String[] args) {
        Calculator calc = new Calculator();
        System.out.println("add(int, int): " + calc.add(5, 10));
        System.out.println("add(int, int, int): " + calc.add(5, 10, 15));
        System.out.println("add(double, double): " + calc.add(5.5, 10.5));
    }
}
```

2.Create a base class Shape with a method area() that prints a message. Then create two subclasses Circle→override area() to calculator and print area of circle Rectangle→ override area() to calculate and print area of a rectangle.

A. **package** Day4;

```
class Shape {
    void area() {
        System.out.println("Calculating area...");
    }
}
class Circle extends Shape {
    double radius = 5.0;
    @Override
    void area() {
        double result = 3.14 * radius * radius;
        System.out.println("Area of Circle: " + result);
    }
}
class Rectangle extends Shape {
    double length = 10.0;
    double width = 5.0;
    @Override
    void area() {
        double result = length * width;
        System.out.println("Area of Rectangle: " + result);
    }
}
```

```
public class Main {
   public static void main(String[] args) {
      Shape shape;
      shape = new Circle();
      shape.area();
      shape = new Rectangle();
      shape.area();
   }
}
```

3.Create a Bank class with a method getInterestRate() create subclasses:
SBI→return 6.7% ICICI→return 7.0%
HDFC→return 7.5%

A.
```
package Day4;
class Bank {
   double getInterestRate() {
      return 0.0;
   }
}
class SBI extends Bank {
   @Override
   double getInterestRate() {
      return 6.7;
   }
}
class ICICI extends Bank {
   @Override
   double getInterestRate() {
      return 7.0;
   }
}
class HDFC extends Bank {
   @Override
```

```
    double getInterestRate() {
        return 7.5;
    }
}
public class BankTest {
    public static void main(String[] args) {
        Bank bank;
        bank = new SBI();
        System.out.println("SBI Interest Rate: " + bank.getInterestRate() + "%");
        bank = new ICICI();
        System.out.println("ICICI Interest Rate: " + bank.getInterestRate() + "%");
        bank = new HDFC();
        System.out.println("HDFC Interest Rate: " + bank.getInterestRate() + "%");
    }
}
```

Combined question:

1.Create an abstract class SmartDevice with methods like turnOn(), turnOff(), and performFunction().
Create child classes:

- SmartPhone: performs calling and browsing.
- SmartWatch: tracks fitness and time.
- SmartSpeaker: plays music and responds to voice commands.
- Write code to store all objects in an array and use polymorphism to invoke their performFunction().

A. **package** Day4;

```
abstract class SmartDevice {
    public void turnOn() {
        System.out.println("Device is turning on...");
    }
    public void turnOff() {
        System.out.println("Device is turning off...");
    }
    public abstract void performFunction();
}
class SmartPhone extends SmartDevice {
    public void performFunction() {
        System.out.println("SmartPhone is making a call and browsing the internet.");
    }
}
class SmartWatch extends SmartDevice {
    public void performFunction() {
        System.out.println("SmartWatch is tracking fitness and showing time.");
    }
}
class SmartSpeaker extends SmartDevice {
    public void performFunction() {
        System.out.println("SmartSpeaker is playing music and responding to voice
commands.");
    }
}
public class SmartDevice_p {
    public static void main(String[] args) {
        SmartDevice[] devices = new SmartDevice[3];
        devices[0] = new SmartPhone();
        devices[1] = new SmartWatch();
        devices[2] = new SmartSpeaker();
        for (SmartDevice device : devices) {
            device.turnOn();
            device.performFunction();
            device.turnOff();
            System.out.println();
        }
    }
}
```

2.Design an interface Bank with methods deposit(), withdraw(), and getBalance().
Implement this in SavingsAccount and CurrentAccount classes.

- Use inheritance to create a base Account class.
- Demonstrate method overriding with customized logic for withdrawal (e.g., minimum balance in SavingsAccount).

A.
```
package Day4;
public interface Bank {
   void deposit(double amount);
   void withdraw(double amount);
   double getBalance();
}
public abstract class Account implements Bank {
   protected double balance;
   protected String accountNumber;
   public Account(String accountNumber, double initialBalance) {
      this.accountNumber = accountNumber;
      this.balance = initialBalance;
   }
   @Override
   public void deposit(double amount) {
      if (amount > 0) {
         balance += amount;
         System.out.println("Deposited: " + amount);
      } else {
         System.out.println("Invalid deposit amount.");
      }
   }
   @Override
   public double getBalance() {
      return balance;
   }
```

```
    @Override
    public abstract void withdraw(double amount);
}
public class SavingsAccount extends Account {
    private final double MIN_BALANCE = 500;
    public SavingsAccount(String accountNumber, double initialBalance) {
        super(accountNumber, initialBalance);
    }
    @Override
    public void withdraw(double amount) {
        if (balance - amount >= MIN_BALANCE) {
            balance -= amount;
            System.out.println("Withdrew: " + amount);
        } else {
            System.out.println("Withdrawal denied. Minimum balance must be
maintained.");
        }
    }
}
public class CurrentAccount extends Account {
    public CurrentAccount(String accountNumber, double initialBalance) {
        super(accountNumber, initialBalance);
    }
    @Override
    public void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
            System.out.println("Withdrew: " + amount);
        } else {
            System.out.println("Insufficient balance.");
        }
    }
```

```
}
public class Main {
    public static void main(String[] args) {
        Bank savings = new SavingsAccount("SA001", 1000);
        Bank current = new CurrentAccount("CA001", 1000);
        System.out.println("--- Savings Account ---");
        savings.deposit(500);
        savings.withdraw(900);
        savings.withdraw(200);
        System.out.println("Savings Balance: " + savings.getBalance());
        System.out.println("Current Account");
        current.deposit(300);
        current.withdraw(1200);
        current.withdraw(1000);
        System.out.println("Current Balance: " + current.getBalance());
    }
}
```

3.Create a base class Vehicle with method start().
Derive Car, Bike, and Truck from it and override the start() method.

- Create a static method that accepts Vehicle type and calls start().
- Pass different vehicle objects to test polymorphism.

A.
```
package Day4;
class Vehicle {
    public void start() {
        System.out.println("Starting the vehicle...");
    }
}
class Car extends Vehicle {
    @Override
    public void start() {
        System.out.println("Car is starting with a key ignition...");
    }
}
class Bike extends Vehicle {
   @Override
   public void start() {
      System.out.println("Bike is starting with a kick start...");
   }
}
class Truck extends Vehicle {
   @Override
   public void start() {
      System.out.println("Truck is starting with an air brake system...");
   }
}
class VehicleStarter {
   public static void startVehicle(Vehicle vehicle) {
      vehicle.start();  // Polymorphism: resolves to the correct overridden method
   }
}
public class Vehicle_main {
   public static void main(String[] args) {
      Vehicle car = new Car();
      Vehicle bike = new Bike();
      Vehicle truck = new Truck();
      VehicleStarter.startVehicle(car);
      VehicleStarter.startVehicle(bike);
      VehicleStarter.startVehicle(truck);
   }
}
```

4.Design an abstract class Person with fields like name, age, and abstract method getRoleInfo().
Create subclasses:

- Student: has course and roll number.
- Professor: has subject and salary.
- TeachingAssistant: extends Student and implements getRoleInfo() in a hybrid way.
- Create and print info for all roles using overridden getRoleInfo().

A. **package** Day4;

```
abstract class Person {
    protected String name;
    protected int age;
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
    public abstract void getRoleInfo();
}
class Student extends Person {
    protected String course;
    protected int rollNumber;
    public Student(String name, int age, String course, int rollNumber) {
        super(name, age);
        this.course = course;
        this.rollNumber = rollNumber;
    }
    @Override
    public void getRoleInfo() {
        System.out.println("Student Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Course: " + course);
        System.out.println("Roll Number: " + rollNumber);
    }
}
class Professor extends Person {
    private String subject;
    private double salary;
    public Professor(String name, int age, String subject, double salary) {
        super(name, age);
        this.subject = subject;
        this.salary = salary;
    }
    @Override
    public void getRoleInfo() {
        System.out.println("Professor Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Subject: " + subject);
        System.out.println("Salary: $" + salary);
    }
}
class TeachingAssistant extends Student {
    private String assistantFor;
    public TeachingAssistant(String name, int age, String course, int rollNumber,
String assistantFor) {
        super(name, age, course, rollNumber);
        this.assistantFor = assistantFor;
    }
    @Override
    public void getRoleInfo() {
        System.out.println("Teaching Assistant Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Course: " + course);
        System.out.println("Roll Number: " + rollNumber);
        System.out.println("Assisting For: " + assistantFor);
    }
}
public class RoleInfoMain {
    public static void main(String[] args) {
        Person student = new Student("Alice", 20, "Computer Science", 101);
        Person professor = new Professor("Dr. Smith", 45, "Mathematics", 95000);
        Person ta = new TeachingAssistant("Bob", 22, "Data Structures", 202,
"Algorithms");
        student.getRoleInfo();
        System.out.println();
        professor.getRoleInfo();
        System.out.println();
        ta.getRoleInfo();
    }
}
```

5.Create:

- Interface Drawable with method draw()
- Abstract class Shape with abstract method area()
  Subclasses: Circle, Rectangle, and Triangle.
- Calculate area using appropriate formulas.
- Demonstrate how interface and abstract class work together.

A.
```
package polymorphism;
interface Drawable {
    void draw();
}
abstract class Shape implements Drawable {
    public abstract double area();  // Abstract method for area
}
class Circle extends Shape {
    private double radius;

    public Circle(double radius) {
```

```java
        this.radius = radius;
    }
    @Override
    public void draw() {
        System.out.println("Drawing a Circle");
    }
    @Override
    public double area() {
        return Math.PI * radius * radius;
    }
}
class Rectangle extends Shape {
    private double length;
    private double width;

    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }
    @Override
    public void draw() {
        System.out.println("Drawing a Rectangle");
    }
    @Override
    public double area() {
        return length * width;
    }
}
class Triangle extends Shape {
    private double base;
    private double height;
    public Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }
    @Override
    public void draw() {
        System.out.println("Drawing a Triangle");
    }
    @Override
    public double area() {
        return 0.5 * base * height;
    }
}
public class ShapeDemo {
    public static void main(String[] args) {
        Shape circle = new Circle(5);
        Shape rectangle = new Rectangle(4, 6);
        Shape triangle = new Triangle(3, 8);
        Shape[] shapes = {circle, rectangle, triangle};
        for (Shape shape : shapes) {
            shape.draw();
            System.out.println("Area: " + shape.area());
            System.out.println();
        }
    }
}
```