Inheritance :

1. Create Multilevel Inheritance for Vehicle

Program :

```
package Day4_Inheritance;
class Vehicle {
  void vehicle() {
      System.out.println("Vehicle");
   }
}
class FourWheeler extends Vehicle{
   void wheels() {
      System.out.println("Four Wheeler");
   }
}
class PetrolFourWheeler extends FourWheeler {
   void type() {
      System.out.println("Petrol Four Wheeler");
   }
}
class FiveSeaterPetrolFourWheeler extends PetrolFourWheeler {
   void seats() {
      System.out.println("Five Seater Petrol Four Wheeler");
   }
}
class Baleno extends FiveSeaterPetrolFourWheeler {
   void brand() {
      System.out.println("Baleno");
   }
}
public class Vehicle_In {
```

```
	public static void main(String[] args) {
		Baleno b = new Baleno();
		b.vehicle();
		b.type();
		b.type();
		b.seats();
		b.brand();
	}
}
```

Output : Vehicle

Petrol Four Wheeler

Petrol Four Wheeler

Five Seater Petrol Four Wheeler

Baleno

2. Demonstrate the use of the super keyword?

Program :

```
package Day4_Inheritance;
class Animal {
	Animal() {
		System.out.println("animal constructor");
	}
	void eat () {
		System.out.println(" Eats " );
	}
}
class Dog extends Animal{
	Dog() {
		super();
		System.out.println("dog constructor");
	}
```

```
        void eat() {
            super.eat();
            System.out.println("royal canin");
        }
}
public class Super_key {
    public static void main(String[] args) {
            // TODO Auto-generated method stub
            Dog d = new Dog();
            d.eat();
    }
}
```

Ouput : animal constructor

dog constructor

Eats

royal canin

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

Program :

```
package Day4_Inheritance;
class Hospital {
   String name = "Apollo Hospital";
   void location() {
      System.out.println("Located in Hyderabad");
   }
}
class Patient extends Hospital {
   void details() {
      System.out.println("Patient admitted in: " + name);
      location();
```

```
    }
}
public class Hospital_In {
        public static void main(String[] args) {
                Patient p = new Patient();
                p.details();
        }
}
```

Output : Patient admitted in: Apollo Hospital

Located in Hyderabad

4. Hierarchical inheritance about Education ?

Program :

```
package Day4_Inheritance;
class After_12th {
        void options() {
                System.out.println("Courses after 12th : ");
        }
}
class Engineering extends After_12th{
        void branches() {
                System.out.println("Engineering branches :  IT, Mechanical, CS");
        }
}
class IT extends Engineering{
        void it() {
                System.out.println("choosen IT ");
        }
}
class Mech extends Engineering{
        void mech() {
```

```
                System.out.println("choosen Mechanical");
        }
}
class CS extends Engineering{
        void cs() {
                System.out.println("choosen CS");
        }
}
class Medical extends After_12th{
        void branches() {
                System.out.println("Medical courses : MBBS, BDS");
        }
}
class MBBS extends Medical{
        void mbbs() {
                System.out.println("choosen MBBS ");
        }
}
class BDS extends Medical{
        void bds() {
                System.out.println("choosen BDS");
        }
}
class Other_Courses extends After_12th{
        void branches() {
                System.out.println("Medical courses : MBBS, BDS");
        }
}
class BCA extends Other_Courses{
        void bca() {
                System.out.println("choosen BCA ");
```

```
	}
}
class BBA extends Other_Courses{
	void bba() {
		System.out.println("choosen BBA");
	}
}

public class Education_Heirarchy {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		System.out.println("Engineering path");
		IT i = new IT();
		i.options();
		i.branches();
		i.it();
		System.out.println("Medical path");
		MBBS m = new MBBS();
		m.options();
		m.branches();
		m.mbbs();
		System.out.println("Other Courses path");
		BCA b = new BCA();
		b.options();
		b.branches();
		b.bca();
	}
}
```

Output : Engineering path

Courses after 12th :

Engineering branches : IT, Mechanical, CS

choosen IT

Medical path

Courses after 12th :

Medical courses : MBBS, BDS

choosen MBBS

Other Courses path

Courses after 12th :

Medical courses : MBBS, BDS

choosen BCA

5. Hospital Heirarchy ?

Program :

```
package Day4_Inheritance;
class Hospital1 {
	void open(String S) {
		System.out.println("Hospital is "+ S);
	}
	void close(String S) {
		System.out.println("Hospital is "+ S);
	}
}
class Doctor extends Hospital1{
	void timings (String tm) {
		System.out.println("Doctor Timings are : " + tm);
	}
}
class Nurse extends Hospital1 {
	void shift(String shift) {
		System.out.println("Nurse Shift :" + shift);
	}
```

```
}
class Accountant extends Hospital1{
	void workType(String type) {
		System.out.println("Accountant Work : " + type);
	}
}
class Gynac extends Doctor {
	void Exp (int exp) {
		System.out.println("Doctor Experience : " + exp);
	}
}
class Endo extends Doctor {
	void Exp (int exp) {
		System.out.println("Doctor Experience : " + exp);
	}
}
class Cardiac extends Doctor {
	void Exp (int exp) {
		System.out.println("Doctor Experience : " + exp);
	}
}
class Payments extends Accountant{
	void bill(int amt) {
		System.out.println("Total Bill : " + amt);
	}
}
class Documentation extends Accountant{
	void reports(String rep) {
		System.out.println("Reports are : " + rep);
	}
}
```

```
class Operation extends Cardiac {
	void surgery(String sur) {
		System.out.println("Surgery is  : " + sur);
	}
}
class OPD extends Cardiac {
	void days(String days) {
		System.out.println("OPD days : " + days);
	}
}
public class Hospital_Heirarchy {
	public static void main(String[] args) {
		Gynac g = new Gynac();
		g.open("Opened");
		g.Exp(5);
		g.timings("5-9");

		Endo e = new Endo();
		e.Exp(6);
		e.timings("6-12");


		Cardiac c = new Cardiac();
		c.Exp(6);
		c.timings("12-4");

		Nurse n = new Nurse();
		n.shift("Night");

		Payments p = new Payments();
		p.workType("Billing");
```

```
                p.bill(50000);

                Documentation d = new Documentation();
                d.reports("Ready");

                Operation o = new Operation();
                o.surgery("Bypass Surgery");

                OPD  op= new OPD();
                op.days("Mon-Tue-Wed");
        }
}
```

Output : Hospital is Opened

Doctor Experience : 5

Doctor Timings are : 5-9

Doctor Experience : 6

Doctor Timings are : 6-12

Doctor Experience : 6

Doctor Timings are : 12-4

Nurse Shift :Night

Accountant Work : Billing

Total Bill : 50000

Reports are : Ready

Surgery is : Bypass Surgery

OPD days : Mon-Tue-Wed

Polymorphism :

1. 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)

Program :

```
package Day4_Polymorphism;
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 class Calculator_P {
        public static void main(String[] args) {
                Calculator c = new Calculator();
                System.out.println(c.add(3, 5));
                System.out.println(c.add(3, 5,10));
                System.out.println(c.add(3.0, 5.0));
}
}
```

Output : 8

18

8.0

2. Create a shape then two subclasses Circle, Rectangle

Program :

```
package Day4_Polymorphism;
class Shape {
   void area() {
      System.out.println("Area of shape");
```

```
    }
}
class Circle extends Shape {
    double radius = 5;
    void area() {
        System.out.println("Area of circle: " + (Math.PI * radius * radius));
    }
}
class Rectangle extends Shape {
    int length = 10, breadth = 5;
    void area() {
        System.out.println("Area of rectangle: " + (length * breadth));
    }
}
public class Shape_P {
        public static void main(String[] args) {
                // TODO Auto-generated method stub
                Circle c = new Circle();
                c.area();
                Rectangle r = new Rectangle();
                r.area();
        }
}
```

Output : Area of circle: 78.53981633974483

Area of rectangle: 50

3. Create a Bank class with sub class SBI, ICICI, HDFC

Program :

```
package Day4_Polymorphism;
class Bank {
    double getInterestRate() {
```

```
        return 0;
    }
}
class SBI extends Bank {
    double getInterestRate() {
        return 6.7;
    }
}
class ICICI extends Bank {
    double getInterestRate() {
        return 7.0;
    }
}
class HDFC extends Bank {
    double getInterestRate() {
        return 7.5;
    }
}
public class Bank_P {
        public static void main(String[] args) {
                // TODO Auto-generated method stub
                SBI s = new SBI();
                System.out.println("SBI : " + s.getInterestRate() + "%");
                ICICI i = new ICICI();
                System.out.println("ICICI : "+i.getInterestRate() + "%");
                HDFC h = new HDFC();
                System.out.println("HDFC : "+h.getInterestRate() + "%");
        }
}
```

Output : SBI: 6.7%

ICICI: 7.0%

HDFC: 7.5%

4. Runtime polymorphism with constructor chaining .

Program :

```
package Day4_Polymorphism;
class Vehicle {
   Vehicle() {
      System.out.println("Vehicle Created");
   }
   void run() {
      System.out.println("Vehicle runs");
   }
}
class Bike extends Vehicle {
   Bike() {
      super();
      System.out.println("Bike Created");
   }
   void run() {
      System.out.println("Bike runs");
   }
}
public class Vehicle_P {
        public static void main(String[] args) {
                Bike b = new Bike();
                b.run();
        }
}
```

Output : Vehicle Created

Bike Created

Bike runs

Combined Questions :

1. Create abstract class for smart device with turnOn, turnOff, performFunction

Program :

```
package Day4_Abstraction;
abstract class SmartDevice {
   abstract void turnOn();
   abstract void turnOff();
   abstract void performFunction();
}
class SmartPhone extends SmartDevice {
   void turnOn() {
         System.out.println("Phone ON");
   }
   void turnOff() {
         System.out.println("Phone OFF");
   }
   void performFunction() {
      System.out.println("Calling and Browsing");
   }
}
class SmartWatch extends SmartDevice {
   void turnOn() {
         System.out.println("Watch ON");
   }
   void turnOff() {
         System.out.println("Watch OFF");
   }
   void performFunction() {
      System.out.println("Tracking fitness and time");
   }
}
```

```
class SmartSpeaker extends SmartDevice {
   void turnOn() {
         System.out.println("Speaker ON");
   }
   void turnOff() {
         System.out.println("Speaker OFF");
   }
   void performFunction() {
      System.out.println("Playing music and voice commands");
   }
}
public class Phone_A {
         public static void main(String[] args) {
                  // TODO Auto-generated method stub
                  SmartDevice[] devices = {
                              new SmartPhone(),
                              new SmartWatch(),
                              new SmartSpeaker()
                           };
                           for (SmartDevice d : devices) d.performFunction();
         }
}
```

Output : Calling and Browsing

Tracking fitness and time

Playing music and voice commands

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

Program :

```
package Day4_Abstraction;
```

```
interface BankInterface {
    void deposit(double amt);
    void withdraw(double amt);
    double getBalance();
}
class Account {
    double balance = 0;
}
class SavingsAccount extends Account implements BankInterface {
    public void deposit(double amt) {
        if (amt > 0) {
            balance += amt;
            System.out.println("Savings Account - Deposited: " + amt);
        }
    }
    public void withdraw(double amt) {
        if (balance - amt >= 500) {
            balance -= amt;
            System.out.println("Savings Account - Withdrawn: " + amt);
        } else {
            System.out.println("Savings Account - Minimum balance of 500 required!");
        }
    }
    public double getBalance() {
        return balance;
    }
}
class CurrentAccount extends Account implements BankInterface {
    public void deposit(double amt) {
        if (amt > 0) {
            balance += amt;
```

```
            System.out.println("Current Account - Deposited: " + amt);
        }
    }
    public void withdraw(double amt) {
        if (amt > 0 && amt <= balance) {
            balance -= amt;
            System.out.println("Current Account - Withdrawn: " + amt);
        } else {
            System.out.println("Current Account - Invalid withdrawal amount!");
        }
    }
    public double getBalance() {
        return balance;
    }
}
public class Bank_A {
    public static void main(String[] args) {
        SavingsAccount s = new SavingsAccount();
        s.deposit(1000);
        s.withdraw(400);
        s.withdraw(300);
        System.out.println("Savings Account Balance: " + s.getBalance());
        System.out.println("-------------------------");
        CurrentAccount c = new CurrentAccount();
        c.deposit(2000);
        c.withdraw(1500);
        c.withdraw(600);
        System.out.println("Current Account Balance: " + c.getBalance());
    }
}
```

Output : Savings Account - Deposited: 1000.0

Savings Account - Withdrawn: 400.0

Savings Account - Minimum balance of 500 required!

Savings Account Balance: 600.0

Current Account - Deposited: 2000.0

Current Account - Withdrawn: 1500.0

Current Account - Invalid withdrawal amount!

Current Account Balance: 500.0

3. Create a base class Vehicle with method start().

Program :

```
package Day4_Abstraction;
class Vehicle {
    void start() {
        System.out.println("Vehicle started");
    }
}
class Car extends Vehicle {
    void start() {
        System.out.println("Car started");
    }
}
class Bike extends Vehicle {
    void start() {
        System.out.println("Bike started");
    }
}
class Truck extends Vehicle {
    void start() {
        System.out.println("Truck started");
    }
```

```
}
public class Vehicle_A {
	static void startVehicle(Vehicle v) {
	   v.start();
	}
	public static void main(String[] args) {
		startVehicle(new Car());
		startVehicle(new Bike());
		startVehicle(new Truck());
	}
}
```

Output : Car started

Bike started

Truck started

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

Program :

```
package Day4_Abstraction;
abstract class Person {
   String name;
   int age;
   Person(String name, int age) {
      this.name = name;
      this.age = age;
   }
   abstract void getRoleInfo();
}
class Student extends Person {
   String course;
   int rollNo;
```

```
    Student(String name, int age, String course, int rollNo) {
        super(name, age);
        this.course = course;
        this.rollNo = rollNo;
    }
    void getRoleInfo() {
        System.out.println("Student: " + name + ", Age: " + age + ", Course: " + course + ", Roll No: " +
rollNo);
    }
}
class Professor extends Person {
    String subject;
    double salary;
    Professor(String name, int age, String subject, double salary) {
        super(name, age);
        this.subject = subject;
        this.salary = salary;
    }
    void getRoleInfo() {
        System.out.println("Professor: " + name + ", Age: " + age + ", Subject: " + subject + ", Salary: " +
salary);
    }
}
class TeachingAssistant extends Student {
    TeachingAssistant(String name, int age, String course, int rollNo) {
        super(name, age, course, rollNo);
    }
    void getRoleInfo() {
        System.out.println("Teaching Assistant: " + name + ", Age: " + age + ", Course: " + course + ", Roll
No: " + rollNo);
    }
}
```

```
public class Person_A {
    public static void main(String[] args) {
        Person s = new Student("Alice", 20, "CS", 101);
        Person p = new Professor("Dr. Smith", 45, "Math", 75000);
        Person t = new TeachingAssistant("Bob", 22, "CS", 102);
        s.getRoleInfo();
        p.getRoleInfo();
        t.getRoleInfo();
    }
}
```

Output : Student: Alice, Age: 20, Course: CS, Roll No: 101

Professor: Dr. Smith, Age: 45, Subject: Math, Salary: 75000.0

Teaching Assistant: Bob, Age: 22, Course: CS, Roll No: 102

5. Create Interface Drawable with method draw().

Program :

```
package Day4_Abstraction;
interface Drawable {
    void draw();
}
abstract class Shape implements Drawable {
    abstract double area();
}
class Circle extends Shape {
    double radius;
    Circle(double radius) {
        this.radius = radius;
    }
    public void draw() {
        System.out.println("Drawing Circle");
    }
```

```
    double area() {
        return Math.PI * radius * radius;
    }
}
class Rectangle extends Shape {
    double length, breadth;
    Rectangle(double length, double breadth) {
        this.length = length;
        this.breadth = breadth;
    }
    public void draw() {
        System.out.println("Drawing Rectangle");
    }
    double area() {
        return length * breadth;
    }
}
class Triangle extends Shape {
    double base, height;
    Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }
    public void draw() {
        System.out.println("Drawing Triangle");
    }
    double area() {
        return 0.5 * base * height;
    }
}
public class Shape_A {
```

```
    public static void main(String[] args) {
        Circle c = new Circle(5);
        c.draw();
        System.out.println("Area of Circle: " + c.area());
        Rectangle r = new Rectangle(4, 6);
        r.draw();
        System.out.println("Area of Rectangle: " + r.area());
        Triangle t = new Triangle(3, 4);
        t.draw();
        System.out.println("Area of Triangle: " + t.area());
    }
}
```

Output : Drawing Circle

Area of Circle: 78.53981633974483

Drawing Rectangle

Area of Rectangle: 24.0

Drawing Triangle

Area of Triangle: 6.0