1. create multilevel inheritance for

```
package Day_4;
class Vehicle {
  void type() {
    System.out.println("This is a vehicle");
  }
}

class Four_Wheeler extends Vehicle {
  void wheels() {
    System.out.println("Has four wheels");
  }
}

class Petrol_Four_Wheeler extends Four_Wheeler {
  void fuelType() {
    System.out.println("Runs on petrol");
  }
}

class FiveSeater_Petrol_Four_Wheeler extends Petrol_Four_Wheeler {
  void seatingCapacity() {
    System.out.println("Has seating capacity of 5");
  }
}

class Baleno_FiveSeater_Petrol_Four_Wheeler extends FiveSeater_Petrol_Four_Wheeler {
  void modelName() {
    System.out.println("Model: Baleno");
  }
}
public class Multilevel_Inheritance {
    public static void main(String[] args) {
      Baleno_FiveSeater_Petrol_Four_Wheeler car = new
Baleno_FiveSeater_Petrol_Four_Wheeler();
      car.type();
      car.wheels();
      car.fuelType();
      car.seatingCapacity();
      car.modelName();
  }
}
```

Output:

```
This is a vehicle
Has four wheels
```

Runs on petrol
Has seating capacity of 5
Model: Baleno

2. Demonstrate the use of the super keyword

```
package Day_4;

class Parent {
   String name = "Parent Class";

   Parent() {
      System.out.println("Parent constructor called");
   }

   void display() {
      System.out.println("Display method in Parent");
   }
}

class Child extends Parent {
   String name = "Child Class";

   Child() {
      super();
      System.out.println("Child constructor called");
   }

   void display() {
      super.display();
      System.out.println("Display method in Child");
      System.out.println("Parent name: " + super.name);
      System.out.println("Child name: " + name);
   }
}

public class SuperKeywordDemo {
   public static void main(String[] args) {
      Child c = new Child();
      c.display();
   }
}
```

Output:
Parent constructor called
Child constructor called
Display method in Parent

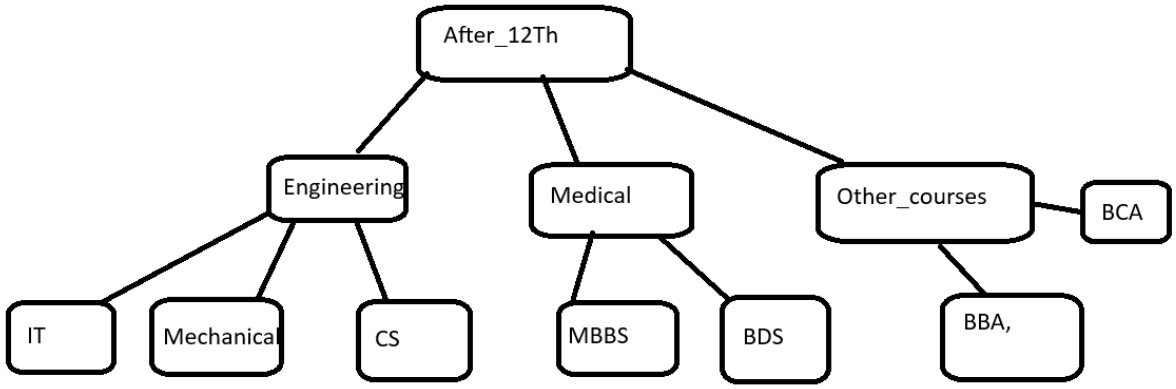
Display method in Child
Parent name: Parent Class
Child name: Child Class

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

```
package Day_4;

class Hospital {
   String hospitalName = "City Hospital";
   String location = "Pune";

   void hospitalInfo() {
      System.out.println(hospitalName + " located in " + location);
   }
}

class Patient extends Hospital {
   String patientName = "Rahul";
   int age = 30;

   void patientInfo() {
      hospitalInfo(); // From Hospital class
      System.out.println("Patient Name: " + patientName);
      System.out.println("Age: " + age);
   }
}

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

Output:
City Hospital located in Pune
Patient Name: Rahul
Age: 30

4. Create Hierarchical inheritance




```
package Day_4;

class Animal {
   void eat() {
      System.out.println("Animal eats food");
   }
}

class Dog extends Animal {
   void bark() {
      System.out.println("Dog barks");
   }
}

class Cat extends Animal {
   void meow() {
      System.out.println("Cat meows");
   }
}

public class HierarchicalInheritanceDemo {
   public static void main(String[] args) {
      Dog d = new Dog();
      d.eat();
      d.bark();

      Cat c = new Cat();
      c.eat();
      c.meow();
   }
}
```

Output:

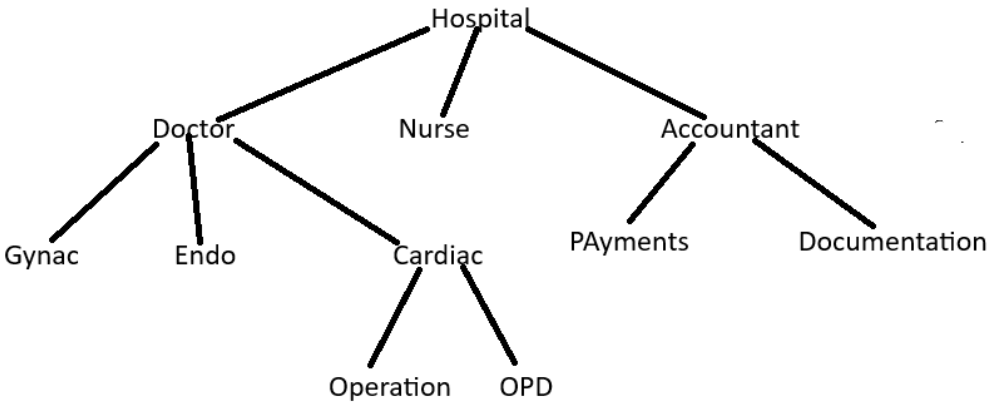
Animal eats food

Dog barks

Animal eats food

Cat meows

5. Create practice on this




```
package Day_4_Assignment;


class Hospital1
{
        void showHospital()
        {
           System.out.println("Welcome to the Hospital.");
        }
}

class Doctor extends Hospital1 {
 void showDoctor() {
    System.out.println("Doctor Department.");
 }
}

class Nurse extends Hospital1 {
 void showNurse() {
    System.out.println("Nursing Department.");
 }
}

class Accountant extends Hospital1 {
 void showAccountant() {
    System.out.println("Accounts Department.");
 }
}

//Level 2 - Under Doctor
```

```
class Gynac extends Doctor {
 void showGynac() {
    System.out.println("Gynaecology Section.");
 }
}

class Endo extends Doctor {
 void showEndo() {
    System.out.println("Endocrinology Section.");
 }
}

class Cardiac extends Doctor {
 void showCardiac() {
    System.out.println("Cardiology Section.");
 }
}

//Level 3 - Under Cardiac
class Operation extends Cardiac {
 void showOperation() {
    System.out.println("Cardiac Operation Section.");
 }
}

class OPD extends Cardiac {
 void showOPD() {
    System.out.println("Cardiac OPD Section.");
 }
}

//Level 2 - Under Accountant
class Payments extends Accountant {
 void showPayments() {
    System.out.println("Payment Processing Section.");
 }
}

class Documentation extends Accountant {
 void showDocs() {
    System.out.println("Document & Billing Section.");
 }
}

//Main class
```

```
public class HospitalStructure {
 public static void main(String[] args) {

    System.out.println("----- Cardiac OPD Flow -----");
    OPD opd = new OPD();
    opd.showHospital();
    opd.showDoctor();
    opd.showCardiac();
    opd.showOPD();

    System.out.println("\n----- Endocrinology Department -----");
    Endo endo = new Endo();
    endo.showHospital();
    endo.showDoctor();
    endo.showEndo();

    System.out.println("\n----- Accounts and Documentation -----");
    Documentation doc = new Documentation();
    doc.showHospital();
    doc.showAccountant();
    doc.showDocs();

    System.out.println("\n----- Nurse Section -----");
    Nurse nurse = new Nurse();
    nurse.showHospital();
    nurse.showNurse();

    System.out.println("\n----- Cardiac Operation Room -----");
    Operation op = new Operation();
    op.showHospital();
    op.showDoctor();
    op.showCardiac();
    op.showOperation();
 }
}
```

Output:

```
----- Cardiac OPD Flow -----
Welcome to the Hospital.
Doctor Department.
Cardiology Section.
Cardiac OPD Section.

----- Endocrinology Department -----
Welcome to the Hospital.
Doctor Department.
```

Endocrinology Section.

----- Accounts and Documentation -----
Welcome to the Hospital.
Accounts Department.
Document & Billing Section.

----- Nurse Section -----
Welcome to the Hospital.
Nursing Department.

----- Cardiac Operation Room -----
Welcome to the Hospital.
Doctor Department.
Cardiology Section.
Cardiac Operation Section.

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)

```
package Day_4_Assignment;

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();
```

```
        int sum1 = calc.add(10, 20);
        int sum2 = calc.add(5, 15, 25);
        double sum3 = calc.add(12.5, 7.3);

        System.out.println("Sum of 2 integers: " + sum1);
        System.out.println("Sum of 3 integers: " + sum2);
        System.out.println("Sum of 2 doubles: " + sum3);

    }
}
```

Output:

```
Sum of 2 integers: 30
Sum of 3 integers: 45
Sum of 2 doubles: 19.8
```

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

```
package Day_4_Assignment;

class Shape
{
   void area()
   {
      System.out.println("This is a shape");
   }
}
class Circle extends Shape
{
   void area()
   {

      System.out.println("This is a Circle");
   }
}
class Rectangle extends Shape
{
   void area()
   {
      System.out.println("This is a Rectangle");
   }
}
public class Shapes
{
        public static void main(String[] args)
```

```
        {
            Shape s;

            s = new Circle();
            s.area();

            s = new Rectangle();
            s.area();
          }
    }
```

Output:

```
This is a Circle
This is a Rectangle
```

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

```
package Day_4_Assignment;

class Bank {
   double getInterestRate() {
      return 0.0;
   }
}

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 BankDemo {
   public static void main(String[] args) {
```

```
        Bank b;

        b = new SBI();
        System.out.println("SBI Interest Rate: " + b.getInterestRate() + "%");

        b = new ICICI();
        System.out.println("ICICI Interest Rate: " + b.getInterestRate() + "%");

        b = new HDFC();
        System.out.println("HDFC Interest Rate: " + b.getInterestRate() + "%");
    }
}
```

Output:

```
SBI Interest Rate: 6.7%
ICICI Interest Rate: 7.0%
HDFC Interest Rate: 7.5%
```

4. Runtime Polymorphism with constructor Chaining create a class vehicle with a constructor that prints “Vehicle Created”
   Create a subclass Bike that override a method and uses super() in constructor

```
package Day_4_Assignment;

class Vehicle1 {
    Vehicle1() {
        System.out.println("Vehicle Created");
    }

    void start() {
        System.out.println("Vehicle is starting...");
    }
}

class Bike extends Vehicle1 {
    Bike() {
        super();
        System.out.println("Bike Created");
    }

    void start() {
        System.out.println("Bike is starting...");
    }
}

public class Runtime_Vehicle
```

```
{
        public static void main(String[] args)
        {
    Vehicle1 v = new Bike();
    v.start();
  }
}
```

Output:

Vehicle Created

Bike Created

Bike is starting...

5. 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.

```
package Day_4_Assignment;

abstract class SmartDevice {
  abstract void turnOn();
  abstract void turnOff();
  abstract void performFunction();
}

class SmartPhone extends SmartDevice {
  void turnOn() {
    System.out.println("SmartPhone ON");
  }

  void turnOff() {
    System.out.println("SmartPhone OFF");
  }

  void performFunction() {
    System.out.println("Calling and Browsing");
  }
}

class SmartWatch extends SmartDevice {
  void turnOn() {
    System.out.println("SmartWatch ON");
  }
```

```
    void turnOff() {
        System.out.println("SmartWatch OFF");
    }

    void performFunction() {
        System.out.println("Fitness Tracking and Time Display");
    }
}

class SmartSpeaker extends SmartDevice {
    void turnOn() {
        System.out.println("SmartSpeaker ON");
    }

    void turnOff() {
        System.out.println("SmartSpeaker OFF");
    }

    void performFunction() {
        System.out.println("Playing Music and Voice Commands");
    }
}

public class Device {
    public static void main(String[] args) {
        SmartDevice[] devices = {
            new SmartPhone(),
            new SmartWatch(),
            new SmartSpeaker()
        };

        for (SmartDevice device : devices) {
            device.performFunction();
        }
    }
}
```

Output:

Calling and Browsing

Fitness Tracking and Time Display

Playing Music and Voice Commands

6. Write code to store all objects in an array and use polymorphism to invoke their performFunction().
7. 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)

```
package Day_4_Assignment;

interface Bank1 {
   void deposit(double amount);
   void withdraw(double amount);
   double getBalance();
}

class Account {
   String accountHolder;
   double balance;

   void setAccount(String name, double initialBalance) {
      accountHolder = name;
      balance = initialBalance;
   }

   void displayInfo() {
      System.out.println("Account Holder: " + accountHolder);
      System.out.println("Balance: ₹" + balance);
   }
}

class SavingsAccount extends Account implements Bank1 {
   final double minBalance = 500;

   public void deposit(double amount) {
      balance += amount;
   }

   public void withdraw(double amount) {
      if (balance - amount >= minBalance) {
         balance -= amount;
      } else {
         System.out.println("Withdrawal denied: Minimum balance ₹500 required.");
      }
   }

   public double getBalance() {
      return balance;
   }
}
```

```
class CurrentAccount extends Account implements Bank1 {
    public void deposit(double amount) {
        balance += amount;
    }

    public void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
        } else {
            System.out.println("Withdrawal denied: Insufficient funds.");
        }
    }

    public double getBalance() {
        return balance;
    }
}

public class BankInterface {
    public static void main(String[] args) {
        SavingsAccount sa = new SavingsAccount();
        sa.setAccount("Ravi", 1000);
        sa.deposit(500);
        sa.withdraw(1200);
        sa.displayInfo();

        System.out.println();

        CurrentAccount ca = new CurrentAccount();
        ca.setAccount("Priya", 2000);
        ca.deposit(1000);
        ca.withdraw(2500);
        ca.displayInfo();
    }
}
```

Output:
Withdrawal denied: Minimum balance ₹500 required.
Account Holder: Ravi
Balance: ₹1500.0

Account Holder: Priya
Balance: ₹500.0

8. 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.

```
package Day_4_Assignment;

class Vehicles {
   void start() {
      System.out.println("Vehicle is starting");
   }
}

class Car extends Vehicles {
   void start() {
      System.out.println("Car is starting");
   }
}

class Bikes extends Vehicles {
   void start() {
      System.out.println("Bike is starting");
   }
}

class Truck extends Vehicles {
   void start() {
      System.out.println("Truck is starting");
   }
}

public class TransportDemo {
   static void beginRide(Vehicles v) {
      v.start();
   }

   public static void main(String[] args) {
      Vehicles car = new Car();
      Vehicles bike = new Bikes();
      Vehicles truck = new Truck();

      beginRide(car);
      beginRide(bike);
      beginRide(truck);
   }
}
```

Output:
Car is starting
Bike is starting
Truck is starting

9. 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().

```
package Day_4_Assignment;

abstract class Person {
   String name;
   int age;

   Person(String n, int a) {
      name = n;
      age = a;
   }

   abstract void getRoleInfo();
}

class Student extends Person {
   String course;
   int rollNo;

   Student(String n, int a, String c, int r) {
      super(n, a);
      course = c;
      rollNo = r;
   }

   void getRoleInfo() {
      System.out.println("Student: " + name + ", Age: " + age + ", Course: " + course + ", Roll No: " +
rollNo);
   }
}

class Professor extends Person {
   String subject;
   double salary;
```

```
    Professor(String n, int a, String s, double sal) {
        super(n, a);
        subject = s;
        salary = sal;
    }

    void getRoleInfo() {
        System.out.println("Professor: " + name + ", Age: " + age + ", Subject: " + subject + ", Salary: ₹" +
salary);
    }
}

class TeachingAssistant extends Student {
    String assistArea;

    TeachingAssistant(String n, int a, String c, int r, String area) {
        super(n, a, c, r);
        assistArea = area;
    }

    void getRoleInfo() {
        System.out.println("Teaching Assistant: " + name + ", Age: " + age + ", Course: " + course + ", Roll
No: " + rollNo + ", Assists: " + assistArea);
    }
}


public class College {
            public static void main(String[] args) {
        Person s = new Student("Amit", 20, "B.Sc CS", 101);
        Person p = new Professor("Dr. Meena", 45, "Physics", 85000);
        Person ta = new TeachingAssistant("Sneha", 23, "M.Sc CS", 202, "Lab");

        s.getRoleInfo();
        p.getRoleInfo();
        ta.getRoleInfo();
    }
}
```

Output:

Student: Amit, Age: 20, Course: B.Sc CS, Roll No: 101

Professor: Dr. Meena, Age: 45, Subject: Physics, Salary: ₹85000.0

Teaching Assistant: Sneha, Age: 23, Course: M.Sc CS, Roll No: 202, Assists: Lab

10. 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.

```
package Day_4_Assignment;
interface Drawable {
   void draw();
}

abstract class Shape1 implements Drawable {
   abstract void area();
}

class Circles extends Shape1 {
   public void draw() {
      System.out.println("Circle");
   }

   void area() {
      System.out.println("Area: " + (3.14 * 3 * 3));
   }
}

class Rectangles extends Shape1 {
   public void draw() {
      System.out.println("Rectangle");
   }

   void area() {
      System.out.println("Area: " + (4 * 5));
   }
}

class Triangle extends Shape1 {
   public void draw() {
      System.out.println("Triangle");
   }

   void area() {
      System.out.println("Area: " + (0.5 * 6 * 2));
   }
}


public class Shapes_Interface {
```

```
        public static void main(String[] args) {
                Shape1 s1 = new Circles();
                Shape1 s2 = new Rectangles();
                Shape1 s3 = new Triangle();

            s1.draw();
            s1.area();

            s2.draw();
            s2.area();

            s3.draw();
            s3.area();
          }
}
```

Output:

```
Circle
Area: 28.259999999999998
Rectangle
Area: 20
Triangle
Area: 6.0
```