1. Create multilevel inheritance for

//Vehicle

//Four\_wheeler

//Petrol\_Four\_Wheeler

//FiveSeater\_Petrol\_Four\_Wheeler

//Baleno\_FiveSeater\_Petrol\_Four\_Wheeler

Code:

package Practice\_assign;

class Vehicle {

void start() {

System.***out***.println("Vehicle is starting");

}

}

class Four\_wheeler extends Vehicle {

void type() {

System.***out***.println("This is a four wheeler");

}

}

class Petrol\_Four\_Wheeler extends Four\_wheeler {

void fuel() {

System.***out***.println("Runs on Petrol");

}

}

class FiveSeater\_Petrol\_Four\_Wheeler extends Petrol\_Four\_Wheeler {

void seats() {

System.***out***.println("It has 5 seats");

}

}

class Baleno\_FiveSeater\_Petrol\_Four\_Wheeler extends FiveSeater\_Petrol\_Four\_Wheeler {

void model() {

System.***out***.println("Model name: Baleno");

}

}

public class Assign\_5 {

public static void main(String[] args) {

Baleno\_FiveSeater\_Petrol\_Four\_Wheeler obj = new Baleno\_FiveSeater\_Petrol\_Four\_Wheeler();

obj.start();

obj.type();

obj.fuel();

obj.seats();

obj.model();

}

}

Output:

Vehicle is starting

This is a four wheeler

Runs on Petrol

It has 5 seats

Model name: Baleno

1. Demonstrate the use of the super keyword

Code:

package Practice\_assign;

class Hospital {

String hospitalName = "City Hospital";

String location = "Bangalore";

void showHospitalDetails() {

System.***out***.println("Hospital Name: " + hospitalName);

System.***out***.println("Location: " + location);

}

}

class Patient extends Hospital {

String patientName = "Ramesh";

int age = 45;

void showPatientDetails() {

System.***out***.println("Patient Name: " + patientName);

System.***out***.println("Age: " + age);

}

}

public class Assign\_6 {

public static void main(String[] args) {

Patient p = new Patient();

p.showHospitalDetails(); // Access method from Hospital class

p.showPatientDetails(); // Access method from Patient class

}

}

Output:

Hospital Name: City Hospital

Location: Bangalore

Patient Name: Ramesh

Age: 45

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

Code:

package Practice\_assign;

class Hospital {

String hospitalName = "City Hospital";

String location = "Hyderabad";

void displayHospital() {

System.out.println("Hospital Name: " + hospitalName);

System.out.println("Location: " + location);

}

}

class Patient extends Hospital {

String patientName = "Ravi";

int age = 30;

void displayPatient() {

System.out.println("Patient Name: " + patientName);

System.out.println("Age: " + age);

}

}

public class Assign\_7 {

public static void main(String[] args) {

Patient p = new Patient();

p.displayHospital(); // accessing Hospital class method

p.displayPatient(); // accessing Patient class method

}

}

Output:

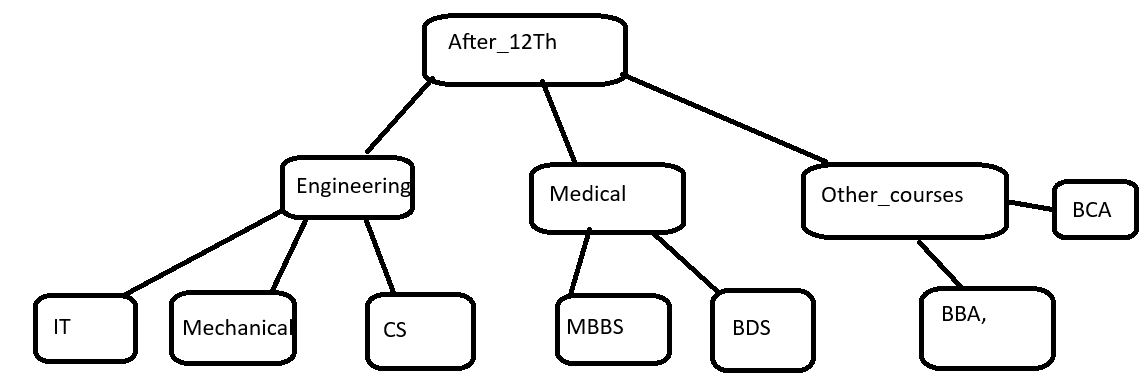
Hospital Name: City Hospital

Location: Hyderabad

Patient Name: Ravi

Age: 30

1. Create Hierarchical inheritance



Code:

package Practice\_assign;

class After\_12Th {

void display() {

System.out.println("Courses available after 12th:");

}

}

class Engineering extends After\_12Th {

void engCourses() {

System.out.println("Engineering Branches: IT, Mechanical, CS");

}

}

class Medical extends After\_12Th {

void medCourses() {

System.out.println("Medical Branches: MBBS, BDS");

}

}

class Other\_courses extends After\_12Th {

void otherCourses() {

System.out.println("Other Courses: BBA, BCA");

}

}

public class Assign\_8 {

public static void main(String[] args) {

Engineering e = new Engineering();

e.display();

e.engCourses();

Medical m = new Medical();

m.display();

m.medCourses();

Other\_courses o = new Other\_courses();

o.display();

o.otherCourses();

}

}

Output:

Courses available after 12th:

Engineering Branches: IT, Mechanical, CS

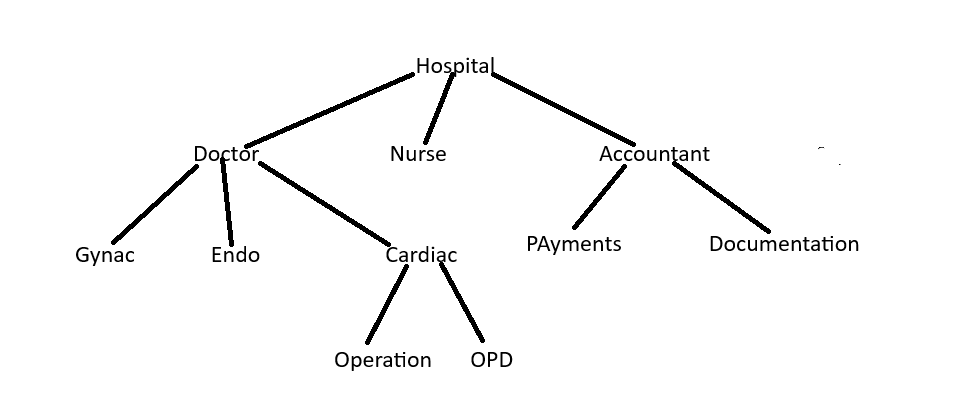
Courses available after 12th:

Medical Branches: MBBS, BDS

Courses available after 12th:

Other Courses: BBA, BCA

1. Create practice on this



Code:

package Practice\_assign;

class After\_12Th {

void display() {

System.out.println("Courses available after 12th:");

}

}

class Engineering extends After\_12Th {

void engCourses() {

System.out.println("Engineering Branches: IT, Mechanical, CS");

}

}

class Medical extends After\_12Th {

void medCourses() {

System.out.println("Medical Branches: MBBS, BDS");

}

}

class Other\_courses extends After\_12Th {

void otherCourses() {

System.out.println("Other Courses: BBA, BCA");

}

}

public class Assign\_8 {

public static void main(String[] args) {

Engineering e = new Engineering();

e.display();

e.engCourses();

Medical m = new Medical();

m.display();

m.medCourses();

Other\_courses o = new Other\_courses();

o.display();

o.otherCourses();

}

}

Output:

Courses available after 12th:

Engineering Branches: IT, Mechanical, CS

Courses available after 12th:

Medical Branches: MBBS, BDS

Courses available after 12th:

Other Courses: BBA, BCA

**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)

**Code:**

package Practice\_assign;

class Calculator {

void add(int a, int b) {

System.out.println("Sum of two integers: " + (a + b));

}

void add(int a, int b, int c) {

System.out.println("Sum of three integers: " + (a + b + c));

}

void add(double a, double b) {

System.out.println("Sum of two doubles: " + (a + b));

}

}

public class Assign\_9 {

public static void main(String[] args) {

Calculator c = new Calculator();

c.add(5, 10);

c.add(2, 4, 6);

c.add(3.5, 2.5);

}

}

**Output:**

Sum of two integers: 15

Sum of three integers: 12

Sum of two doubles: 6.0

1. Create a base class Shape with a method area() that prints a message. Then create two subclasses Circleàoverride area() to calculator and print area of circle Rectangleà override area() to calculate and print area of a rectangle

Code:

package Practice\_assign;

class Shape {

void area() {

System.out.println("Calculating area...");

}

}

class Circle extends Shape {

double radius;

Circle(double r) {

radius = r;

}

@Override

void area() {

double result = 3.14 \* radius \* radius;

System.out.println("Area of Circle: " + result);

}

}

class Rectangle extends Shape {

double length, width;

Rectangle(double l, double w) {

length = l;

width = w;

}

@Override

void area() {

double result = length \* width;

System.out.println("Area of Rectangle: " + result);

}

}

public class Assign\_10 {

public static void main(String[] args) {

Shape s1 = new Circle(5);

s1.area();

Shape s2 = new Rectangle(4, 6);

s2.area();

}

}

Output:

Area of Circle: 78.5

Area of Rectangle: 24.0

1. Create a Bank class with a method getInterestRate() create subclasses: SBIàreturn 6.7% ICICIàreturn 7.0% HDFCàreturn 7.5%

Code:

package Practice\_assign;

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 Assign\_11 {

public static void main(String[] args) {

Bank b1 = new SBI();

System.out.println("SBI Interest Rate: " + b1.getInterestRate() + "%");

Bank b2 = new ICICI();

System.out.println("ICICI Interest Rate: " + b2.getInterestRate() + "%");

Bank b3 = new HDFC();

System.out.println("HDFC Interest Rate: " + b3.getInterestRate() + "%");

}

}

Output:

SBI Interest Rate: 6.7%

ICICI Interest Rate: 7.0%

HDFC Interest Rate: 7.5%

1. 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

Code:

package Practice\_assign;

class Vehicle {

Vehicle() {

System.out.println("Vehicle Created");

}

void run() {

System.out.println("Vehicle is running");

}

}

class Bike extends Vehicle {

Bike() {

super(); // calls Vehicle constructor

System.out.println("Bike Created");

}

@Override

void run() {

System.out.println("Bike is running safely");

}

}

public class Assign\_12 {

public static void main(String[] args) {

Vehicle v = new Bike(); // runtime polymorphism

v.run();

}

}

Output:

Vehicle Created

Bike Created

Bike is running safely

**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().

Code:

package Practice\_assign;

abstract class SmartDevice {

abstract void turnOn();

abstract void turnOff();

abstract void performFunction();

}

class SmartPhone extends SmartDevice {

void turnOn() {

System.out.println("SmartPhone is turning on");

}

void turnOff() {

System.out.println("SmartPhone is turning off");

}

void performFunction() {

System.out.println("SmartPhone is calling and browsing");

}

}

class SmartWatch extends SmartDevice {

void turnOn() {

System.out.println("SmartWatch is turning on");

}

void turnOff() {

System.out.println("SmartWatch is turning off");

}

void performFunction() {

System.out.println("SmartWatch is tracking fitness and time");

}

}

class SmartSpeaker extends SmartDevice {

void turnOn() {

System.out.println("SmartSpeaker is turning on");

}

void turnOff() {

System.out.println("SmartSpeaker is turning off");

}

void performFunction() {

System.out.println("SmartSpeaker is playing music and responding to voice commands");

}

}

public class Assign\_13 {

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 d : devices) {

d.turnOn();

d.performFunction();

d.turnOff();

System.out.println();

}

}

}

Output:

SmartPhone is turning on

SmartPhone is calling and browsing

SmartPhone is turning off

SmartWatch is turning on

SmartWatch is tracking fitness and time

SmartWatch is turning off

SmartSpeaker is turning on

SmartSpeaker is playing music and responding to voice commands

SmartSpeaker is turning off

**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).

Code:

package Practice\_assign;

interface Bank {

void deposit(double amount);

void withdraw(double amount);

double getBalance();

}

class Account implements Bank {

protected double balance;

public Account(double initialBalance) {

this.balance = initialBalance;

}

@Override

public void deposit(double amount) {

if(amount > 0) {

balance += amount;

System.out.println("Deposited: " + amount);

} else {

System.out.println("Deposit amount must be positive");

}

}

@Override

public void withdraw(double amount) {

if(amount > 0 && amount <= balance) {

balance -= amount;

System.out.println("Withdrawn: " + amount);

} else {

System.out.println("Withdrawal failed: Insufficient balance or invalid amount");

}

}

@Override

public double getBalance() {

return balance;

}

}

class SavingsAccount extends Account {

private static final double MIN\_BALANCE = 500;

public SavingsAccount(double initialBalance) {

super(initialBalance);

}

@Override

public void withdraw(double amount) {

if(amount > 0 && (balance - amount) >= MIN\_BALANCE) {

balance -= amount;

System.out.println("Savings Account Withdrawn: " + amount);

} else {

System.out.println("Withdrawal denied: Minimum balance of " + MIN\_BALANCE + " must be maintained");

}

}

}

class CurrentAccount extends Account {

public CurrentAccount(double initialBalance) {

super(initialBalance);

}

@Override

public void withdraw(double amount) {

if(amount > 0 && amount <= balance) {

balance -= amount;

System.out.println("Current Account Withdrawn: " + amount);

} else {

System.out.println("Withdrawal failed: Insufficient balance or invalid amount");

}

}

}

public class Assign\_14 {

public static void main(String[] args) {

SavingsAccount savings = new SavingsAccount(1000);

CurrentAccount current = new CurrentAccount(1000);

System.out.println("Savings Account Balance: " + savings.getBalance());

savings.withdraw(600); // Should fail because it breaks minimum balance rule

savings.withdraw(400); // Should succeed

System.out.println("Savings Account Balance after withdrawal: " + savings.getBalance());

System.out.println();

System.out.println("Current Account Balance: " + current.getBalance());

current.withdraw(900); // Should succeed

System.out.println("Current Account Balance after withdrawal: " + current.getBalance());

}

}

Output:

Savings Account Balance: 1000.0

Withdrawal denied: Minimum balance of 500.0 must be maintained

Savings Account Withdrawn: 400.0

Savings Account Balance after withdrawal: 600.0

Current Account Balance: 1000.0

Current Account Withdrawn: 900.0

Current Account Balance after withdrawal: 100.0

**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.

**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().

Code:

package Practice\_assign;

class Vehicle {

public void start() {

System.out.println("Vehicle is starting");

}

}

class Car extends Vehicle {

@Override

public void start() {

System.out.println("Car is starting with ignition key");

}

}

class Bike extends Vehicle {

@Override

public void start() {

System.out.println("Bike is starting with kick start");

}

}

class Truck extends Vehicle {

@Override

public void start() {

System.out.println("Truck is starting with heavy-duty ignition");

}

}

public class Assign\_15 {

// Static method that accepts Vehicle and calls start()

public static void startVehicle(Vehicle vehicle) {

vehicle.start();

}

public static void main(String[] args) {

Vehicle myCar = new Car();

Vehicle myBike = new Bike();

Vehicle myTruck = new Truck();

startVehicle(myCar); // Car's start()

startVehicle(myBike); // Bike's start()

startVehicle(myTruck); // Truck's start()

}

}

Output:

Car is starting with ignition key

Bike is starting with kick start

Truck is starting with heavy-duty ignition

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.

Code:

package Practice\_assign;

interface Drawable {

void draw();

}

abstract class Shape implements Drawable {

abstract double area();

}

class Circle extends Shape {

private double radius;

public Circle(double radius) {

this.radius = radius;

}

@Override

public double area() {

return Math.PI \* radius \* radius;

}

@Override

public void draw() {

System.out.println("Drawing Circle with radius " + radius);

}

}

class Rectangle extends Shape {

private double length, width;

public Rectangle(double length, double width) {

this.length = length;

this.width = width;

}

@Override

public double area() {

return length \* width;

}

@Override

public void draw() {

System.out.println("Drawing Rectangle with length " + length + " and width " + width);

}

}

class Triangle extends Shape {

private double base, height;

public Triangle(double base, double height) {

this.base = base;

this.height = height;

}

@Override

public double area() {

return 0.5 \* base \* height;

}

@Override

public void draw() {

System.out.println("Drawing Triangle with base " + base + " and height " + height);

}

}

public class Assign\_16 {

public static void main(String[] args) {

Shape[] shapes = {

new Circle(5),

new Rectangle(4, 6),

new Triangle(3, 7)

};

for (Shape shape : shapes) {

shape.draw();

System.out.println("Area: " + shape.area());

System.out.println();

}

}

}

Output;

Drawing Circle with radius 5.0

Area: 78.53981633974483

Drawing Rectangle with length 4.0 and width 6.0

Area: 24.0

Drawing Triangle with base 3.0 and height 7.0

Area: 10.5