cy

Problem Statement

Teena is launching a new airline, Boeing747, and needs to calculate the total revenue generated from ticket sales based on the ticket cost and seat availability. Teena's airline offers two types of seats: regular and premium. The ticket cost and seat availability for both types of seats need to be considered for revenue calculation.

```
import java.util.Scanner;

// Base class

class Airline {

  private double cost;


  public void setCost(double cost) {

    this.cost = cost;

  }


  public double getCost() {

    return cost;

  }

}


// Intermediate class

class Indigo extends Airline {

  private int seatAvailability;


  public void setSeatAvailability(int seatAvailability) {

    this.seatAvailability = seatAvailability;
```

```
    }

    public int getSeatAvailability() {
        return seatAvailability;
    }
}

// Derived class
class Boeing747 extends Indigo {
    public double calculateTotalRevenue() {
        return getCost() * getSeatAvailability();
    }
}

// Main class
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Boeing747 plane = new Boeing747();

        double ticketCost = scanner.nextDouble();
        plane.setCost(ticketCost);
        int seatAvailability = scanner.nextInt();
        plane.setSeatAvailability(seatAvailability);

        System.out.printf("Ticket Cost: Rs. %.1f\n", plane.getCost());
```

```
        System.out.println("Seat Availability: " + plane.getSeatAvailability() + " seats");
        System.out.printf("Total Revenue: Rs. %.1f\n", plane.calculateTotalRevenue());

        scanner.close();
    }
}
```

Problem Statement

A bank provides two types of deposit schemes: Fixed Deposits (FD) and Recurring Deposits (RD). Customers want to calculate the interest they can earn based on their selected scheme.

```
import java.util.Scanner;
// Base class
class Account {
    protected String accountHolder;
    protected double principalAmount;

    public Account(String accountHolder, double principalAmount) {
        this.accountHolder = accountHolder;
        this.principalAmount = principalAmount;
    }

    public double calculateInterest() {
        return 0;
    }
}
```

```
// Fixed Deposit class
class FixedDeposit extends Account {
    private int durationInYears;
    private double rateOfInterest;

    public FixedDeposit(String accountHolder, double principalAmount, int durationInYears, double rateOfInterest) {
        super(accountHolder, principalAmount);
        this.durationInYears = durationInYears;
        this.rateOfInterest = rateOfInterest;
    }

    @Override
    public double calculateInterest() {
        return (principalAmount * durationInYears * rateOfInterest) / 100;
    }
}

// Recurring Deposit class
class RecurringDeposit extends Account {
    private int durationInMonths;
    private double rateOfInterest;

    public RecurringDeposit(String accountHolder, double monthlyDeposit, int durationInMonths, double rateOfInterest) {
        super(accountHolder, monthlyDeposit);
```

```
        this.durationInMonths = durationInMonths;
        this.rateOfInterest = rateOfInterest;
    }

    @Override
    public double calculateInterest() {
        double maturityAmount = principalAmount * durationInMonths; // principalAmount is monthly deposit
        return (maturityAmount * durationInMonths * rateOfInterest) / (12 * 100);
    }
}

// Main class
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int choice = sc.nextInt();

        switch (choice) {
            case 1:
                sc.nextLine(); // consume newline
                String fdName = sc.nextLine();
                double fdPrincipal = sc.nextDouble();
                int fdDuration = sc.nextInt();
                double fdRate = sc.nextDouble();
```

```
            FixedDeposit fd = new FixedDeposit(fdName, fdPrincipal, fdDuration, fdRate);
            System.out.printf("Interest for FD: %.1f", fd.calculateInterest());
            break;

        case 2:
            sc.nextLine();
            String rdName = sc.nextLine();
            int rdDeposit = sc.nextInt();
            int rdDuration = sc.nextInt();
            double rdRate = sc.nextDouble();

            RecurringDeposit rd = new RecurringDeposit(rdName, rdDeposit, rdDuration,
rdRate);
            System.out.printf("Interest for RD: %.1f", rd.calculateInterest());
            break;

        default:
            System.out.println("Invalid Choice");
    }

    sc.close();
  }
}
```

Problem Statement

Joy, a house-building constructor, seeks assistance from his friend, a developer.

```
import java.util.Scanner;
// Base class
class Property {
    protected double length;
    protected double width;

    public void setDimensions(double length, double width) {
        this.length = length;
        this.width = width;
    }
}

// Derived class for 2BHK
class TwoBHK extends Property {
    public double TBHKSquareRoot() {
        return Math.sqrt(length * width);
    }
}

// Derived class for 5BHK
class FiveBHK extends Property {
    public double FBHKSquareRoot() {
        return Math.sqrt(length * width);
```

```
    }
}

// Main class
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        TwoBHK twoBHK = new TwoBHK();
        double twoBHKLength = scanner.nextDouble();
        double twoBHKWidth = scanner.nextDouble();
        twoBHK.setDimensions(twoBHKLength, twoBHKWidth);

        FiveBHK fiveBHK = new FiveBHK();
        double fiveBHKLength = scanner.nextDouble();
        double fiveBHKWidth = scanner.nextDouble();
        fiveBHK.setDimensions(fiveBHKLength, fiveBHKWidth);

        scanner.close();

        System.out.printf("2BHK Square root: %.2f%n", twoBHK.TBHKSquareRoot());
        System.out.printf("5BHK Square root: %.2f%n", fiveBHK.FBHKSquareRoot());
    }
}
```

Problem Statement

Raghav is working on a university payroll system that categorizes employees into four types: Programmer, Assistant Professor, Associate Professor, and Professor. Each employee has basic details, including name, ID, address, email ID, and mobile number. All four roles inherit these details from the base class Employee.

```
class Programmer extends Employee {
   void processSalary() {
      da = 0.97 * bp;
      hra = 0.10 * bp;
      pf = 0.12 * bp;
      staffClub = 0.001 * bp;
      gross = bp + da + hra;
      net = gross - (pf + staffClub);
   }
}

class AssistantProfessor extends Employee {
   void processSalary() {
      da = 0.97 * bp;
      hra = 0.10 * bp;
      pf = 0.12 * bp;
      staffClub = 0.001 * bp;
      gross = bp + da + hra;
      net = gross - (pf + staffClub);
   }
}
```

```
class AssociateProfessor extends Employee {
   void processSalary() {
      da = 0.97 * bp;
      hra = 0.10 * bp;
      pf = 0.12 * bp;
      staffClub = 0.001 * bp;
      gross = bp + da + hra;
      net = gross - (pf + staffClub);
   }
}

class Professor extends Employee {
   void processSalary() {
      da = 0.97 * bp;
      hra = 0.10 * bp;
      pf = 0.12 * bp;
      staffClub = 0.001 * bp;
      gross = bp + da + hra;
      net = gross - (pf + staffClub);
   }
}
```

Problem Statement

Design a Java program to model a basic vehicle system using inheritance. Implement a base class Vehicle with attributes for distance and time. Then, create a derived class Car that includes a brand name, calculates speed using the formula Speed = Distance / Time, and adds extra utility.

```
import java.util.Scanner;
class Vehicle {
   protected double distance;
   protected double time;

   public Vehicle(double distance, double time) {
      this.distance = distance;
      this.time = time;
   }
}

class Car extends Vehicle {
   private final String brand;
   private double speed;

   public Car(String brand, double distance, double time) {
      super(distance, time);
      this.brand = brand;
   }

   public void calc() {
      speed = distance / time;
   }

   public final void display() {
```

```
        String category;
        if (speed < 20) category = "Slow";
        else if (speed <= 60) category = "Average";
        else category = "Fast";

        System.out.println("Brand: " + brand);
        System.out.printf("Speed: %.2f km/hr (%s)%n", speed, category);
    }

    public double predictDistance(double newTime) {
        return speed * newTime;
    }

    public double predictTime(double newDistance) {
        return newDistance / speed;
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        double distance = scanner.nextDouble();
        double time = scanner.nextDouble();
        double newTime = scanner.nextDouble();
        double newDistance = scanner.nextDouble();
```

```
        Car car = new Car("Toyota", distance, time);
        car.calc();
        car.display();

        System.out.printf("Distance covered in %.1f hours: %.2f km%n", newTime,
car.predictDistance(newTime));
        System.out.printf("Time taken to cover %.1f km: %.2f hours%n", newDistance,
car.predictTime(newDistance));

        scanner.close();
    }
}
```