Assignment

Create one Employee class, mention one instance, create methods in it and create instance and reference in it.

```
package org.java.example;

import java.util.Scanner;


public class Employee {
// Fields for employee details
private String name;
private String department;
private long contact;
private String location;
private int employeeID;

Scanner scanner = new Scanner(System.in);


public void acceptEmployeeDetails() {
System.out.print("Enter Employee ID: ");
employeeID = scanner.nextInt();
scanner.nextLine();

System.out.print("Enter Employee Name: ");
name = scanner.nextLine();

System.out.print("Enter Department: ");
department = scanner.nextLine();

System.out.print("Enter Contact Number: ");
contact = scanner.nextLong();
scanner.nextLine();

System.out.print("Enter Location: ");
location = scanner.nextLine();
}
```

```
// Method to display employee details
public void displayEmployeeDetails() {
System.out.println("\nEmployee Details:");
System.out.println("ID: " + employeeID);
System.out.println("Name: " + name);
System.out.println("Department: " + department);
System.out.println("Contact: " + contact);
System.out.println("Location: " + location);
}
}

package org.java.example;

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

Employee emp1 = new Employee();
emp1.acceptEmployeeDetails();


emp1.displayEmployeeDetails();

Employee emp2 = new Employee();

emp2.acceptEmployeeDetails();


emp2.displayEmployeeDetails();
}
}
```

```
OUTPUT
Employee Details:
ID: 101
Name: Alia Gupta
Department: IT
Contact: 1234567890
Location: New Delhi
```

Employee Details:
ID: 102
Name: Janvi Kumari
Department: HR
Contact: 9876543210
Location: Nagpur

1. Loan Amortization Calculator

Implement a system to calculate and display the monthly payments for a mortgage loan. The system should:

1. Accept the principal amount (loan amount), annual interest rate, and loan term (in years) from the user.

2. Calculate the monthly payment using the standard mortgage formula:

Monthly Payment Calculation:

monthly Payment principal (monthlyInterestRate (1+ monthlyInterest Rate) (numberOfMonths)) / ((1+ monthly Interest Rate) (numberOfMonths)-1)

Where monthlyInterest Rate annualinterestRate/12/100 and numberOfMonths loanTerm 12

Note: Here means power and to find it you can use Math.pow() method.

```java
package com.loanamortization;

public class LoanAmortizationCalculator {

// Fields
private double principal;
private double annualInterestRate;
private int loanTerm;

// Constructor
public LoanAmortizationCalculator(double principal, double
annualInterestRate, int loanTerm) {
this.principal = principal;
this.annualInterestRate = annualInterestRate;
this.loanTerm = loanTerm;
}

// Getter and Setter methods
public double getPrincipal() {
return principal;
}

public void setPrincipal(double principal) {
this.principal = principal;
}

public double getAnnualInterestRate() {
return annualInterestRate;
}

public void setAnnualInterestRate(double annualInterestRate) {
this.annualInterestRate = annualInterestRate;
}

public int getLoanTerm() {
return loanTerm;
}
```

```
public void setLoanTerm(int loanTerm) {
this.loanTerm = loanTerm;
}

// Business logic to calculate monthly payment
public double calculateMonthlyPayment() {
double monthlyInterestRate = (annualInterestRate / 12) / 100;
int numberOfMonths = loanTerm * 12;
return (principal * monthlyInterestRate * Math.pow(1 +
monthlyInterestRate, numberOfMonths))
/ (Math.pow(1 + monthlyInterestRate, numberOfMonths) - 1);
}

// Business logic to calculate the total payment over the loan term
public double calculateTotalPayment() {
return calculateMonthlyPayment() * loanTerm * 12;
}

@Override
public String toString() {
return "LoanAmortizationCalculator [Principal Amount = ₹" + principal
+
", Annual Interest Rate = " + annualInterestRate + "%" +
", Loan Term = " + loanTerm + " years]";
}
}
```

```
package com.loanamortization;

import java.util.Scanner;
public class LoanAmortizationCalculatorUtil {

public LoanAmortizationCalculator acceptRecord() {
Scanner scanner = new Scanner(System.in);

System.out.println("Enter Principal Amount (₹): ");
double principal = scanner.nextDouble();
```

```
System.out.println("Enter Annual Interest Rate (%): ");
double annualInterestRate = scanner.nextDouble();

System.out.println("Enter Loan Term (years): ");
int loanTerm = scanner.nextInt();

return new LoanAmortizationCalculator(principal, annualInterestRate,
loanTerm);
}

public void printRecord(LoanAmortizationCalculator loanCalculator) {
double monthlyPayment = loanCalculator.calculateMonthlyPayment();
double totalPayment = loanCalculator.calculateTotalPayment();

System.out.println("\nLoan Details: ");
System.out.println(loanCalculator.toString());
System.out.printf("Monthly Payment: ₹%.2f%n", monthlyPayment);
System.out.printf("Total Amount Paid Over Life of Loan: ₹%.2f%n",
totalPayment);
}

// Method to display the menu
public void menuList() {
System.out.println("\n*** Loan Amortization Calculator ***");
System.out.println("1. Enter Loan Details");
System.out.println("2. Calculate and Display Monthly Payment and Total
Payment");
System.out.println("3. Exit");
}
}
```

```
package com.loanamortization;

import java.util.Scanner;

public class Program {
```

```
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
LoanAmortizationCalculatorUtil util = new
LoanAmortizationCalculatorUtil();
LoanAmortizationCalculator loanCalculator = null;

int choice;
do {
util.menuList();
System.out.println("Enter your choice: ");
choice = scanner.nextInt();

switch (choice) {
case 1:
loanCalculator = util.acceptRecord();
break;

case 2:
if (loanCalculator != null) {
util.printRecord(loanCalculator);
} else {
System.out.println("Please enter loan details first.");
}
break;

case 3:
System.out.println("Exiting program...");
break;

default:
System.out.println("Invalid choice, please try again.");
}

} while (choice != 3);

scanner.close();
}
}
```

2. Compound Interest Calculator for Investment

Develop a system to compute the future value of an investment with compound interest. The system should:

1. Accept the initial investment amount, annual interest rate, number of times the interest is compounded per year, and investment duration (in years) from the user.

2. Calculate the future value of the investment using the formula:

Future Value Calculation:

FutureValue principal (1 annual Interest Rate /

numberOfCompounds) (numberOfCompounds years)

Total Interest Earned: totalInterest futureValue principal

3. Display the future value and the total interest earned, in Indian Rupees (*).

Define the class CompoundInterestCalculator with fields, an appropriate constructor, getter and setter methods, a tostring method and business logic methods. Define the class CompoundInterestCalculatorUtil with methods accept Record, printRecord, and menulist. Define the class Program with a main method to test the functionality of the utility class.

```
package com.compoundinterest;

public class CompoundInterestCalculator {

 // Fields
```

```
 private double principal;
 private double annualInterestRate;
 private int numberOfCompounds;
 private int years;

 // Constructor
 public CompoundInterestCalculator(double principal, double
annualInterestRate, int numberOfCompounds, int years) {
 this.principal = principal;
 this.annualInterestRate = annualInterestRate;
 this.numberOfCompounds = numberOfCompounds;
 this.years = years;
 }

 // Getter and Setter methods
 public double getPrincipal() {
 return principal;
 }

 public void setPrincipal(double principal) {
 this.principal = principal;
 }

 public double getAnnualInterestRate() {
 return annualInterestRate;
 }

 public void setAnnualInterestRate(double annualInterestRate) {
 this.annualInterestRate = annualInterestRate;
 }

 public int getNumberOfCompounds() {
 return numberOfCompounds;
 }

 public void setNumberOfCompounds(int numberOfCompounds) {
 this.numberOfCompounds = numberOfCompounds;
 }
```

```
 public int getYears() {
 return years;
 }

 public void setYears(int years) {
 this.years = years;
 }

 // to calculate the future value
 public double calculateFutureValue() {
 return principal * Math.pow(1 + (annualInterestRate /
numberOfCompounds / 100), numberOfCompounds * years);
 }

 // to calculate the total interest earned
 public double calculateTotalInterest() {
 return calculateFutureValue() - principal;
 }

 @Override
 public String toString() {
 return "CompoundInterestCalculator [Principal Amount = ₹" + principal
+
 ", Annual Interest Rate = " + annualInterestRate + "%" +
 ", Compounded " + numberOfCompounds + " times per year, " +
 "Investment Duration = " + years + " years]";
 }
}


package com.compoundinterest;

import java.util.Scanner;

public class CompoundInterestCalculatorUtil {

public CompoundInterestCalculator acceptRecord() {
 Scanner scanner = new Scanner(System.in);
```

```
 System.out.println("Enter Principal Amount (₹): ");
 double principal = scanner.nextDouble();

 System.out.println("Enter Annual Interest Rate (%): ");
 double annualInterestRate = scanner.nextDouble();

 System.out.println("Enter Number of Times Interest is Compounded Per
Year: ");
 int numberOfCompounds = scanner.nextInt();

 System.out.println("Enter Investment Duration (Years): ");
 int years = scanner.nextInt();

 return new CompoundInterestCalculator(principal, annualInterestRate,
numberOfCompounds, years);
 }

 // to print investment details and results
 public void printRecord(CompoundInterestCalculator calculator) {
 double futureValue = calculator.calculateFutureValue();
 double totalInterest = calculator.calculateTotalInterest();

 System.out.println("\nInvestment Details: ");
 System.out.println(calculator.toString());
 System.out.printf("Future Value: ₹%.2f%n", futureValue);
 System.out.printf("Total Interest Earned: ₹%.2f%n", totalInterest);
 }

 // to display the menu
 public void menuList() {
 System.out.println("\n*** Compound Interest Calculator ***");
 System.out.println("1. Enter Investment Details");
 System.out.println("2. Calculate and Display Future Value and Total
Interest");
 System.out.println("3. Exit");
 }
}
```

```java
package com.compoundinterest;

import java.util.Scanner;

public class Program {

public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
CompoundInterestCalculatorUtil util = new
CompoundInterestCalculatorUtil();
CompoundInterestCalculator calculator = null;

int choice;
do {
util.menuList();
System.out.println("Enter your choice: ");
choice = scanner.nextInt();

switch (choice) {
case 1:
calculator = util.acceptRecord();
break;

case 2:
if (calculator != null) {
util.printRecord(calculator);
} else {
System.out.println("Please enter investment details first.");
}
break;

case 3:
System.out.println("Exiting program...");
break;

default:
System.out.println("Invalid choice, please try again.");
}
```

```
} while (choice != 3);

scanner.close();
}
}
```

3. BMI (Body Mass Index) Tracker

Create a system to calculate and classify Body Mass Index (BMI). The system should:

1. Accept weight (in kilograms) and height (in meters) from the user.

2. Calculate the BMI using the formula:

BMI Calculation: BMI weight/ (height height)

3. Classify the BMI into one of the following categories:

Underweight: BMI<18.5

Normal weight: 18.5 <BMI<24.9

Overweight: 25<BMI<29.9

Obese: BMI>30

4. Display the BMI value and its classification.

Define the class BMITracker with fields, an appropriate constructor, getter and setter methods, a tostring method, and business logic methods. Define the class EMITrackerUtil with methods acceptRecord, printRecord, and menuList. Define the class Program with a main method to test the functionality of the utility class.

```java
package com.bmitracker;

public class BMITracker {

// Fields for weight and height
private double weight; // in kilograms
private double height; // in meters

// Constructor using this keyword to refer to instance variables
public BMITracker(double weight, double height) {
this.weight = weight;
this.height = height;
}

// Getter and Setter for weight with this keyword
public double getWeight() {
return this.weight;
}

public void setWeight(double weight) {
this.weight = weight;
}

// Getter and Setter for height with this keyword
public double getHeight() {
return this.height;
}

public void setHeight(double height) {
this.height = height;
}

// Method to calculate BMI
public double calculateBMI() {
return this.weight / (this.height * this.height);
}
```

```
// Method to classify the BMI value
public String classifyBMI() {
double bmi = this.calculateBMI();
if (bmi < 18.5) {
return "Underweight";
} else if (bmi >= 18.5 && bmi < 24.9) {
return "Normal weight";
} else if (bmi >= 25 && bmi < 29.9) {
return "Overweight";
} else {
return "Obese";
}
}

// toString method to display weight and height
@Override
public String toString() {
return "BMITracker [Weight = " + this.weight + " kg, Height = " +
this.height + " meters]";
}
}
```

```
package com.bmitracker;

import java.util.Scanner;

public class BMITrackerUtil {

// Method to accept user input and create a BMITracker object
public BMITracker acceptRecord() {
Scanner scanner = new Scanner(System.in);

System.out.println("Enter weight (in kilograms): ");
double weight = scanner.nextDouble();

System.out.println("Enter height (in meters): ");
double height = scanner.nextDouble();
```

```
// Using the constructor to create a BMITracker object
return new BMITracker(weight, height);
}

// Method to print the BMI value and its classification
public void printRecord(BMITracker bmiTracker) {
double bmi = bmiTracker.calculateBMI();
String classification = bmiTracker.classifyBMI();

System.out.println("\nBMI Details: ");
System.out.println(bmiTracker.toString()); // Calling toString method
System.out.printf("Calculated BMI: %.2f%n", bmi);
System.out.println("Classification: " + classification);
}

// Method to display the menu
public void menuList() {
System.out.println("\n*** BMI Tracker ***");
System.out.println("1. Enter Weight and Height");
System.out.println("2. Calculate and Display BMI and Classification");
System.out.println("3. Exit");
}
}

package com.bmitracker;

import java.util.Scanner;

public class Program {

public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
BMITrackerUtil util = new BMITrackerUtil();
BMITracker bmiTracker = null;

int choice;
do {
util.menuList(); // Display the menu
```

```
System.out.println("Enter your choice: ");
choice = scanner.nextInt();

switch (choice) {
case 1:
bmiTracker = util.acceptRecord(); // Accept user input
break;

case 2:
if (bmiTracker != null) {
util.printRecord(bmiTracker); // Display BMI and classification
} else {
System.out.println("Please enter weight and height first.");
}
break;

case 3:
System.out.println("Exiting program...");
break;

default:
System.out.println("Invalid choice, please try again.");
}

} while (choice != 3); // Exit when the user chooses option 3

scanner.close();
}
}
```

4. Discount Calculation for Retail Sales

Design a system to calculate the final price of an item after applying a discount. The system should:

1. Accept the original price of an item and the discount percentage from the user.

2. Calculate the discount amount and the final price using the following formulas:
Discount Amount Calculation: discount.Amount originalPrice. (discountRate / 100)

• Final Price Calculation: finalPrice originalPrice discount Amount

3. Display the discount amount and the final price of the item, in Indian Rupees (4).

Define the class DiscountCalculator with fields, an appropriate constructor, getter and setter methods, a tostring method, and business logic methods. Define the class DiscountCalculatorUtil with methods acceptRecord, printRecord, and menuList.

Define the class Program with a main method to test the functionality of the utility class.

```
package com.discountcalculator;

public class DiscountCalculator {

// Fields for original price and discount rate
private double originalPrice;
private double discountRate;

// Constructor using this keyword to refer to instance variables
public DiscountCalculator(double originalPrice, double discountRate) {
this.originalPrice = originalPrice;
this.discountRate = discountRate;
}

// Getter and Setter for originalPrice with this keyword
public double getOriginalPrice() {
return this.originalPrice;
}

public void setOriginalPrice(double originalPrice) {
```

```
this.originalPrice = originalPrice;
}

// Getter and Setter for discountRate with this keyword
public double getDiscountRate() {
return this.discountRate;
}

public void setDiscountRate(double discountRate) {
this.discountRate = discountRate;
}

// Method to calculate the discount amount
public double calculateDiscountAmount() {
return this.originalPrice * (this.discountRate / 100);
}

// Method to calculate the final price
public double calculateFinalPrice() {
return this.originalPrice - this.calculateDiscountAmount();
}

// toString method to display original price and discount rate
@Override
public String toString() {
return "DiscountCalculator [Original Price = ₹" + this.originalPrice +
", Discount Rate = " + this.discountRate + "%]";
}
}
```

```
package com.discountcalculator;

import java.util.Scanner;

public class DiscountCalculatorUtil {

// Method to accept user input and create a DiscountCalculator object
public DiscountCalculator acceptRecord() {
```

```
Scanner scanner = new Scanner(System.in);

System.out.println("Enter the original price of the item (₹): ");
double originalPrice = scanner.nextDouble();

System.out.println("Enter the discount rate (%): ");
double discountRate = scanner.nextDouble();

// Using the constructor to create a DiscountCalculator object
return new DiscountCalculator(originalPrice, discountRate);
}

// Method to print the discount amount and final price
public void printRecord(DiscountCalculator discountCalculator) {
double discountAmount = discountCalculator.calculateDiscountAmount();
double finalPrice = discountCalculator.calculateFinalPrice();

System.out.println("\nDiscount Details: ");
System.out.println(discountCalculator.toString()); // Calling toString
method
System.out.printf("Discount Amount: ₹%.2f%n", discountAmount);
System.out.printf("Final Price: ₹%.2f%n", finalPrice);
}

// Method to display the menu
public void menuList() {
System.out.println("\n*** Discount Calculator ***");
System.out.println("1. Enter Original Price and Discount Rate");
System.out.println("2. Calculate and Display Discount Amount and Final
Price");
System.out.println("3. Exit");
}
}
```

```
package com.discountcalculator;

import java.util.Scanner;
```

```
public class Program {

public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
DiscountCalculatorUtil util = new DiscountCalculatorUtil();
DiscountCalculator discountCalculator = null;

int choice;
do {
util.menuList(); // Display the menu
System.out.println("Enter your choice: ");
choice = scanner.nextInt();

switch (choice) {
case 1:
discountCalculator = util.acceptRecord(); // Accept user input
break;

case 2:
if (discountCalculator != null) {
util.printRecord(discountCalculator); // Display discount and final
price
} else {
System.out.println("Please enter original price and discount rate
first.");
}
break;

case 3:
System.out.println("Exiting program...");
break;

default:
System.out.println("Invalid choice, please try again.");
}
```

5. Toll Booth Revenue Management

Develop a system to simulate a toll booth for collecting revenue. The system should:

1. Allow the user to set toll rates for different vehicle types: Car, Truck, and Motorcycle.

2. Accept the number of vehicles of each type passing through the toll booth.

3. Calculate the total revenue based on the toll rates and number of vehicles.

4. Display the total number of vehicles and the total revenue collected, in Indian Rupees (*).

Toll Rate Examples:

Car: ₹50.00

• Truck: ₹100.00

• Motorcycle: ₹30.00

Define the class TollBoothRevenueManager with fields, an appropriate constructor, getter and setter methods, a tostring method, and business logic methods. Define the class FollBoothRevenue ManagerUtil with methods acceptRecord, printRecord, and menulist. Define the class Program with a main method to test the functionality of the utility class.

```
package com.tollbooth;

import java.util.Scanner;

public class Program {

public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
TollBoothRevenueManagerUtil util = new TollBoothRevenueManagerUtil();
```

```
TollBoothRevenueManager tollBooth = null;

int choice;
do {
util.menuList(); // Display the menu
System.out.println("Enter your choice: ");
choice = scanner.nextInt();

switch (choice) {
case 1:
tollBooth = util.acceptRecord(); // Set toll rates
break;

case 2:
if (tollBooth != null) {
util.acceptVehicleCounts(tollBooth); // Enter vehicle counts
} else {
System.out.println("Please set the toll rates first.");
}
break;

case 3:
if (tollBooth != null) {
util.printRecord(tollBooth); // Display total revenue and vehicles
} else {
System.out.println("Please set the toll rates and vehicle counts
first.");
}
break;

case 4:
System.out.println("Exiting program...");
break;

default:
System.out.println("Invalid choice, please try again.");
}

} while (choice != 4); // Exit when the user chooses option 4
```

```
scanner.close();
}
}
```

```
package com.tollbooth;

public class TollBoothRevenueManager {

// Fields for toll rates and vehicle counts
private double carTollRate;
private double truckTollRate;
private double motorcycleTollRate;
private int carCount;
private int truckCount;
private int motorcycleCount;

// Constructor using this keyword to refer to instance variables
public TollBoothRevenueManager(double carTollRate, double
truckTollRate, double motorcycleTollRate) {
this.carTollRate = carTollRate;
this.truckTollRate = truckTollRate;
this.motorcycleTollRate = motorcycleTollRate;
}

// Getters and Setters for toll rates
public double getCarTollRate() {
return this.carTollRate;
}

public void setCarTollRate(double carTollRate) {
this.carTollRate = carTollRate;
}

public double getTruckTollRate() {
return this.truckTollRate;
```

```
}

public void setTruckTollRate(double truckTollRate) {
this.truckTollRate = truckTollRate;
}

public double getMotorcycleTollRate() {
return this.motorcycleTollRate;
}

public void setMotorcycleTollRate(double motorcycleTollRate) {
this.motorcycleTollRate = motorcycleTollRate;
}

// Getters and Setters for vehicle counts
public int getCarCount() {
return this.carCount;
}

public void setCarCount(int carCount) {
this.carCount = carCount;
}

public int getTruckCount() {
return this.truckCount;
}

public void setTruckCount(int truckCount) {
this.truckCount = truckCount;
}

public int getMotorcycleCount() {
return this.motorcycleCount;
}

public void setMotorcycleCount(int motorcycleCount) {
this.motorcycleCount = motorcycleCount;
}
```

```
// Business logic methods
public double calculateTotalRevenue() {
double carRevenue = this.carCount * this.carTollRate;
double truckRevenue = this.truckCount * this.truckTollRate;
double motorcycleRevenue = this.motorcycleCount *
this.motorcycleTollRate;
return carRevenue + truckRevenue + motorcycleRevenue;
}

public int calculateTotalVehicles() {
return this.carCount + this.truckCount + this.motorcycleCount;
}

// toString method to display toll rates and vehicle counts
@Override
public String toString() {
return "TollBoothRevenueManager [Car Toll Rate = ₹" + this.carTollRate
+
", Truck Toll Rate = ₹" + this.truckTollRate +
", Motorcycle Toll Rate = ₹" + this.motorcycleTollRate + "]";
}
}
```

```
package com.tollbooth;

import java.util.Scanner;

public class TollBoothRevenueManagerUtil {

// Method to accept user input and create a TollBoothRevenueManager
object
public TollBoothRevenueManager acceptRecord() {
Scanner scanner = new Scanner(System.in);

System.out.println("Enter the toll rate for Cars (₹): ");
double carTollRate = scanner.nextDouble();
```

```
System.out.println("Enter the toll rate for Trucks (₹): ");
double truckTollRate = scanner.nextDouble();

System.out.println("Enter the toll rate for Motorcycles (₹): ");
double motorcycleTollRate = scanner.nextDouble();

// Create and return TollBoothRevenueManager object
return new TollBoothRevenueManager(carTollRate, truckTollRate,
motorcycleTollRate);
}

// Method to accept vehicle counts
public void acceptVehicleCounts(TollBoothRevenueManager tollBooth) {
Scanner scanner = new Scanner(System.in);

System.out.println("Enter the number of Cars: ");
int carCount = scanner.nextInt();
tollBooth.setCarCount(carCount);

System.out.println("Enter the number of Trucks: ");
int truckCount = scanner.nextInt();
tollBooth.setTruckCount(truckCount);

System.out.println("Enter the number of Motorcycles: ");
int motorcycleCount = scanner.nextInt();
tollBooth.setMotorcycleCount(motorcycleCount);
}

// Method to print the total vehicles and total revenue
public void printRecord(TollBoothRevenueManager tollBooth) {
int totalVehicles = tollBooth.calculateTotalVehicles();
double totalRevenue = tollBooth.calculateTotalRevenue();

System.out.println("\nToll Booth Summary:");
System.out.println(tollBooth.toString()); // Display toll rates using
toString
System.out.println("Total Vehicles: " + totalVehicles);
System.out.printf("Total Revenue Collected: ₹%.2f%n", totalRevenue);
}
```

```
// Method to display the menu
public void menuList() {
System.out.println("\n*** Toll Booth Revenue Manager ***");
System.out.println("1. Set Toll Rates");
System.out.println("2. Enter Vehicle Counts");
System.out.println("3. Calculate and Display Total Revenue and
Vehicles");
System.out.println("4. Exit");
}
}
```