**Note:**

* The assignment is designed to practice constructor, getter/setter and toString method.
* Create a separate project for each question and create separate file for each class.
* Try to test the functionality by using menu-driven program.

**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:**
     + monthlyPayment = principal \* (monthlyInterestRate \* (1 + monthlyInterestRate)^(numberOfMonths)) / ((1 + monthlyInterestRate)^(numberOfMonths) - 1)
     + Where monthlyInterestRate = annualInterestRate / 12 / 100 and numberOfMonths = loanTerm \* 12
     + Note: Here ^ means power and to find it you can use Math.pow( ) method
3. Display the monthly payment and the total amount paid over the life of the loan, in Indian Rupees (₹).

Define the class LoanAmortizationCalculator with fields, an appropriate constructor, getter and setter methods, a toString method and business logic methods. Define the class LoanAmortizationCalculatorUtil with methods acceptRecord, printRecord, and menuList. Define the class Program with a main method and test the functionality of the utility class.

**package** org.example2;

**import** java.util.Scanner;

**class** LoanAmortizationCalculator{

**private** **double** Principal;

**private** **double** Annualinterestrate;

**private** **double** Loanterm;

**private** **double** monthlypayment;

**public** LoanAmortizationCalculator( ) {

**this**.Principal = 0;

**this**.Annualinterestrate = 0;

**this**.Loanterm = 0;

**this**.monthlypayment = 0;

}

**public** **void** setInputs(**double** Principal, **double** Annualinterestrate, **double** Loanterm) {

**this**.Principal = Principal;

**this**.Annualinterestrate = Annualinterestrate;

**this**.Loanterm = Loanterm;

}

**public** **void** printRecords() {

System.***out***.println("Principle Amount: " + Principal);

System.***out***.println("Annual Interest Rate: " + Annualinterestrate);

System.***out***.println("Loan term: " + Loanterm);

}

**public** **double** CalMonthlypayment() {

**double** Nummonths = **this**.NumofMonths();

**double** monthlyInterestAmt = **this**.monthlyInterestAmt();

**return** Principal \* (monthlyInterestAmt \* Math.*pow*(1 + monthlyInterestAmt, Nummonths)) /

(Math.*pow*(1 + monthlyInterestAmt, Nummonths) - 1);

}

**public** **double** monthlyInterestAmt() {

**return** Annualinterestrate / 12 / 100;

}

**public** **double** NumofMonths() {

**return** Loanterm \* 12;

}

**public** **void** getValues() {

**this**.monthlypayment = **this**.CalMonthlypayment();

**this**.printRecords();

System.***out***.println("Monthly Payment: " + monthlypayment);

}

}

**public** **class** LoanCal {

**public** **static** **void** main(String[] args) {

LoanAmortizationCalculator lc = **new** LoanAmortizationCalculator();

lc.printRecords();

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Principal Amt: ");

**double** P = sc.nextDouble();

System.***out***.print("Annual Interest rate: ");

**double** A = sc.nextDouble();

System.***out***.print("Loan term : ");

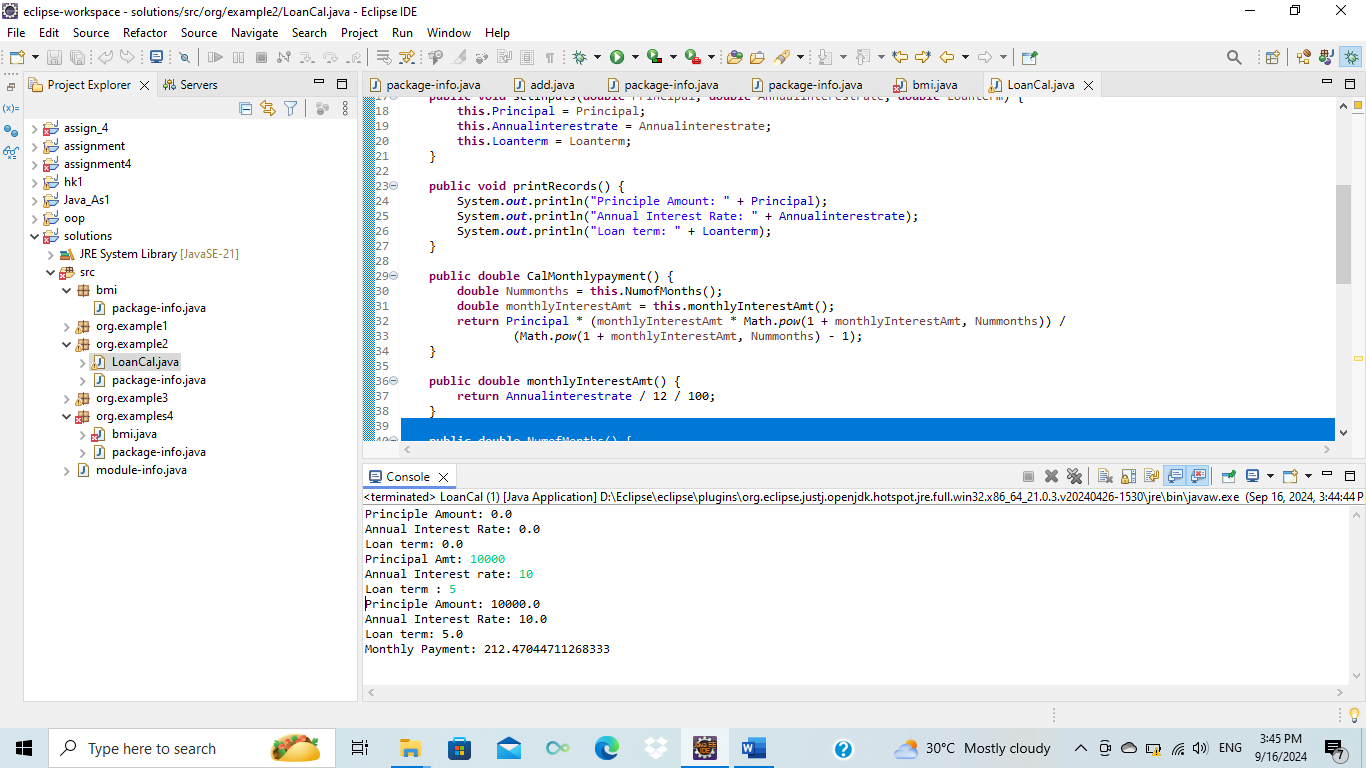
**double** L = sc.nextDouble();

lc.setInputs(P, A, L);

lc.getValues();

}

}



**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 + annualInterestRate / 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 acceptRecord, printRecord, and menuList. Define the class Program with a main method to test the functionality of the utility class.

**package** org.example3;

**import** java.util.Scanner;

**class** CompoundInterestCalculator{

**private** **double** principal;

**private** **double** annualInterestRate;

**private** **int** numberOfCompounds;

**private** **int** years;

**private** **double** futureValue;

**private** **double** totalInterest;

**public** CompoundInterestCalculator( ) {

**this**.principal=0;

**this**.annualInterestRate=0;

**this**.numberOfCompounds=0;

**this**.years=0;

}

**public** **double** getFutureValueCal() {

**this**.futureValue = principal \* Math.*pow*(1 + (annualInterestRate / numberOfCompounds / 100), numberOfCompounds \* years);

**return** futureValue;

}

**public** **double** getTotalIntCal() {

**this**.totalInterest = futureValue - principal;

**return** totalInterest;

}

**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;

}

**public** **void** printRecord() {

**this**.getTotalIntCal();

**this**.getFutureValueCal();

System.***out***.println("Initial Investment : " + principal);

System.***out***.println("Annual Interest Rate: " + annualInterestRate);

System.***out***.println("Future Value: " +futureValue );

System.***out***.println("Total Interest Earned: " + totalInterest);

}

}

**public** **class** CompoundITR {

**public** **static** **void** main(String[] args) {

CompoundInterestCalculator lc = **new** CompoundInterestCalculator();

lc.setPrincipal(10000);

lc.setAnnualInterestRate(10);

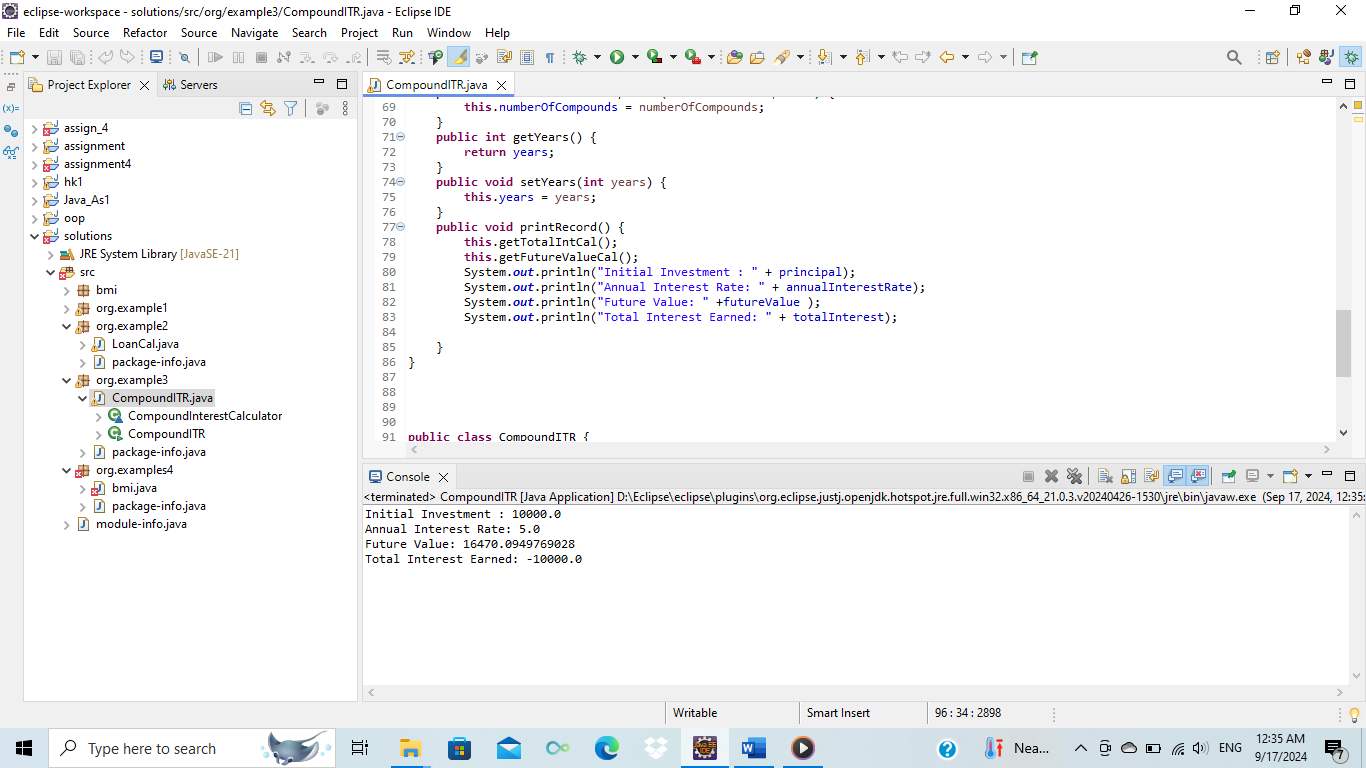
lc.setNumberOfCompounds(12);

lc.setYears(10)

lc.printRecord();

}

}



**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 BMITrackerUtil with methods acceptRecord, printRecord, and menuList. Define the class Program with a main method to test the functionality of the utility class.

**package** org.examples4;

**import** java.util.Scanner;

**class** BMITracker{

**private** **double** weight;

**private** **double** height;

**private** **double** bmi;

**public** BMITracker(**double** weight, **double** height) {

**this**.weight = weight;

**this**.height = height;

**this**.bmi = calculateBMI();

}

**private** **double** calculateBMI() {

**if** (height > 0) {

**return** weight / (height \* height);

}

**return** 0;

}

**public** **double** getWeight() {

**return** weight;

}

**public** **void** setWeight(**double** weight) {

**this**.weight = weight;

**this**.bmi = calculateBMI();

}

**public** **double** getHeight() {

**return** height;

}

**public** **void** setHeight(**double** height) {

**this**.height = height;

**this**.bmi = calculateBMI();

}

**public** **double** getBMI() {

**return** bmi;

}

**public** String checkBMI() {

**if**(bmi<18.5) {

**return** " You are Underweight.";

}

**else** **if**(bmi>=18.5 && bmi<=24.9) {

**return** "YOu have a normal weight";

}

**else** **if**(bmi>=25 && bmi <=29.9) {

**return** "You are overweight";

}

**else** {

**return** "You are Obese";

}

}

@Override

**public** String toString() {

**return** "Weight: " + weight + " kg, Height: " + height + " m, BMI: " + String.*format*("%.2f", bmi) + ", Classification: " + checkBMI();

}

}

**class** BMITrackerUtil {

**private** **static** Scanner *scanner* = **new** Scanner(System.***in***);

**public** **static** BMITracker acceptRecord() {

System.***out***.println("Enter your weight in kilograms: ");

**double** weight = *scanner*.nextDouble();

System.***out***.println("Enter your height in meters: ");

**double** height = *scanner*.nextDouble();

**return** **new** BMITracker(weight, height);

}

**public** **static** **void** printRecord(BMITracker tracker) {

System.***out***.println(tracker.toString());

}

**public** **static** **void** menuList() {

System.***out***.println("1. Calculate BMI");

System.***out***.println("2. Exit");

}

}

**public** **class** bmi {

**public** **static** **void** main(String[] args) {

BMITracker tracker = **null**;

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

**int** choice;

**do** {

BMITrackerUtil.*menuList*();

System.***out***.println("Enter your choice: ");

choice = scanner.nextInt();

**switch** (choice) {

**case** 1:

tracker = BMITrackerUtil.*acceptRecord*();

BMITrackerUtil.*printRecord*(tracker);

**break**;

**case** 2:

System.***out***.println("Exiting...");

**break**;

**default**:

System.***out***.println("Invalid choice! Please try again.");

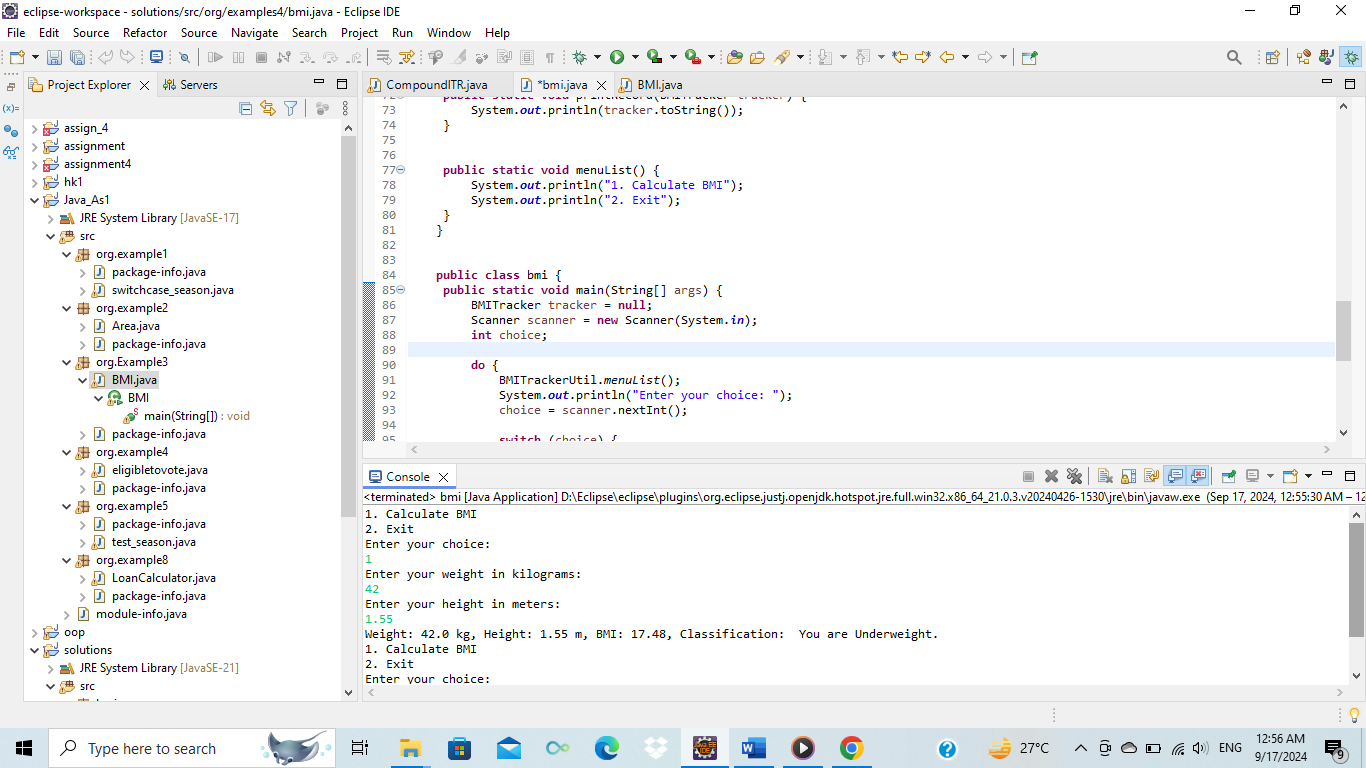
}

} **while** (choice != 2);

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:** discountAmount = originalPrice \* (discountRate / 100)
   * **Final Price Calculation:** finalPrice = originalPrice - discountAmount
3. Display the discount amount and the final price of the item, in Indian Rupees (₹).

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.

**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 TollBoothRevenueManagerUtil with methods acceptRecord, printRecord, and menuList. Define the class Program with a main method to test the functionality of the utility class.

**package** org.example7;

**import** java.util.Scanner;

**class** TollBoothRevenueManager {

**private** **double** carTollRate;

**private** **double** truckTollRate;

**private** **double** motorcycleTollRate;

**private** **int** carCount;

**private** **int** truckCount;

**private** **int** motorcycleCount;

**public** TollBoothRevenueManager(**double** carTollRate, **double** truckTollRate, **double** motorcycleTollRate) {

**super**();

**this**.carTollRate = carTollRate;

**this**.truckTollRate = truckTollRate;

**this**.motorcycleTollRate = motorcycleTollRate;

**this**.carCount = 0;

**this**.truckCount = 0;

**this**.motorcycleCount = 0;

}

**public** **double** getCarTollRate() {

**return** carTollRate;

}

**public** **void** setCarTollRate(**double** carTollRate) {

**this**.carTollRate = carTollRate;

}

**public** **double** getMotorcycleTollRate() {

**return** motorcycleTollRate;

}

**public** **void** setMotorcycleTollRate(**double** motorcycleTollRate) {

**this**.motorcycleTollRate = motorcycleTollRate;

}

**public** **int** getCarCount() {

**return** carCount;

}

**public** **void** setCarCount(**int** carCount) {

**this**.carCount = carCount;

}

**public** **int** getMotorcycleCount() {

**return** motorcycleCount;

}

**public** **void** setMotorcycleCount(**int** motorcycleCount) {

**this**.motorcycleCount = motorcycleCount;

}

**public** **double** getTruckTollRate() {

**return** truckTollRate;

}

**public** **void** setTruckTollRate(**double** truckTollRate) {

**this**.truckTollRate = truckTollRate;

}

**public** **int** getTruckCount() {

**return** truckCount;

}

**public** **void** setTruckCount(**int** truckCount) {

**this**.truckCount = truckCount;

}

**public** **double** calculateTotalRevenue() {

**return** (carTollRate \* carCount) + (truckTollRate \* truckCount) + (motorcycleTollRate \* motorcycleCount);

}

**public** **int** getTotalVehicleCount() {

**return** carCount + truckCount + motorcycleCount;

}

@Override

**public** String toString() {

**return** "calculateTotalRevenue()=" + calculateTotalRevenue()+

" getTotalVehicleCount()=" + getTotalVehicleCount() ;

}

}

**class** TollBoothRevenueManagerUtil{

//Scanner sc = new Scanner(System.in);

**public** **void** acceptRecord(TollBoothRevenueManager tm,Scanner sc) {

System.***out***.println("Enter the number of cars: ");

tm.setCarCount(sc.nextInt());

System.***out***.println("Enter the number of trucks: ");

tm.setTruckCount(sc.nextInt());

System.***out***.println("Enter the number of motorcycles: ");

tm.setMotorcycleCount(sc.nextInt());

}

**public** **void** printRecord(TollBoothRevenueManager tm) {

System.***out***.println(tm);

}

**public** **void** menuList() {

System.***out***.println("1. Enter vehicle count");

System.***out***.println("2. Display total vehicles and revenue");

System.***out***.println("3. Exit");

}

}

**public** **class** Program {

**public** **static** **void** main(String[] args) {

TollBoothRevenueManager tm = **new** TollBoothRevenueManager(50, 100, 30);

TollBoothRevenueManagerUtil util = **new** TollBoothRevenueManagerUtil();

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

**int** choice;

**do** {

util.menuList();

System.***out***.println("Enter your choice: ");

choice = scanner.nextInt();

**switch** (choice) {

**case** 1:

util.acceptRecord(tm, scanner);

**break**;

**case** 2:

util.printRecord(tm);

**break**;

**case** 3:

System.***out***.println("Exiting...");

**break**;

**default**:

System.***out***.println("Invalid choice, please try again.");

}

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

}

// scanner.close();

}