1. **Date and Time Converter**

   Accept date and time from user. You are required to write a Java program to convert dates and times between different formats.

   The program should be able to convert dates between the following formats:
   a. dd/mm/yyyy (e.g., 31/12/2022)
   b. mm/dd/yyyy (e.g., 12/31/2022)
   c. yyyy/mm/dd (e.g., 2022/12/31)

   The program should be able to convert times between the following formats:
   a. hh:mm:ss (e.g., 23:59:59)
   b. hh:mm:ss a (e.g., 11:59:59 PM)
   c. hh:mm (e.g., 23:59)

   The program should be able to convert dates and times between the following formats:
   a. dd/mm/yyyy hh:mm:ss (e.g., 31/12/2022 23:59:59)
   b. mm/dd/yyyy hh:mm:ss a (e.g., 12/31/2022 11:59:59 PM)
   c. yyyy/mm/dd hh:mm (e.g., 2022/12/31 23:59)

2. **BMI Calculator**

   You have been asked to write a Java program to implement a BMI (Body Mass Index) calculator. The program should have the following functionality:

   The BMI calculator class should have the following fields:
   a. height: A double field to store the height of the person in meters.
   b. weight: A double field to store the weight of the person in kilograms.

   The BMI calculator class should have the following methods:
   a. A constructor to initialize the height and weight fields of the BMI calculator object.
   b. Getter and setter methods.
   c. Double calculateBMI(): A method to calculate the BMI of the person using the following formula:

   $$BMI = weight / (height * height).$$

   Write a Java program that creates an object of the BMI calculator class, prompts the user to input their height and weight, sets the height and weight fields of the BMI calculator object using the setter methods, calculates the BMI using the calculateBMI() method, and prints the calculated BMI to the console.

3. **Book Inventory Management System**

You are required to write a Java program to implement a Book class that can be used to manage a book inventory system.

The Book class should have the following fields:

    a. title: A string field to store the title of the book.

    b. author: A string field to store the name of the author of the book.

    c. publisher: A string field to store the name of the publisher of the book.

    d. isbn: A string field to store the ISBN number of the book.

    e. year: An integer field to store the year in which the book was published.

    f. price: A double field to store the price of the book.

    g. quantity: An integer field to store the quantity of the book in the inventory.

The Book class should have following methods

    a. Constructors,

    b. Getter & setter methods

    c. Business Logic methods

        I.   increaseQuantity(int quantity): A method to increase the quantity of the book in the inventory by the specified amount.

        II.   decreaseQuantity(int quantity): A method to decrease the quantity of the book in the inventory by the specified amount.

        III.   getInventoryValue(): A method to calculate the total value of the inventory of the book, which is the product of the price and the quantity of the book.

4. **Electricity Bill Calculation**

    Create a class named "ElectricityBill" that has the following instance variables:

    a. customerName (String)

    b. unitsConsumed (double)

    c. billAmount (double)

Create a constructor that initializes the customerName and unitsConsumed instance variables.

Define a method named "calculateBillAmount" that calculates the bill amount based on the number of units consumed. The formula for calculating the bill amount is:

    a. For the first 100 units: Rs. 5 per unit

    b. For the next 200 units: Rs. 7 per unit

    c. For the remaining units: Rs. 10 per unit

Implement the "calculateBillAmount" method in the "ElectricityBill" class.

Define a main method that creates an object of the "ElectricityBill" class and sets the customerName and unitsConsumed instance variables. Then, call the "calculateBillAmount" method to calculate the bill amount and display the customerName, unitsConsumed, and billAmount.

5. **Telephone Bill Calculation**

You are required to write a Java program to calculate the telephone bill for a given customer based on their usage. The program should take the following inputs from the user:
   a. Customer name
   b. Phone number
   c. Number of calls made
   d. Duration of calls (in minutes)

The program should then calculate the bill for the customer based on the following criteria:
   a. The first 100 calls are charged at a rate of 50 cents per call.
   b. Calls beyond the first 100 are charged at a rate of 25 cents per call.
   c. All calls are subject to a minimum duration of 1 minute.
   d. Calls with a duration less than 1 minute are rounded up to 1 minute.
   e. There is a flat rate of $10 per month for all customers.

6. **Bank Account Management System**

You are required to write a Java program to manage bank accounts. The program should be able to perform the following operations:
   a. Create a new account
   b. Deposit money into an account
   c. Withdraw money from an account
   d. Display the account balance
   e. Display the account holder's information

Each account should have the following information:
   a. Account holder's name (String)
   b. Account number (int)

    c. Account balance (double)

The program should be able to perform the following operations:

a. Create a new account: The program should prompt the user to enter the account holder's name, and generate a unique account number for the new account. The initial account balance should be zero.

b. Deposit money into an account: The program should prompt the user to enter the account number and the amount to be deposited. If the account number is valid, the program should add the amount to the account balance. If the account number is not valid, the program should display an error message.

c. Withdraw money from an account: The program should prompt the user to enter the account number and the amount to be withdrawn. If the account number is valid and the account balance is sufficient, the program should deduct the amount from the account balance. If the account number is not valid or the account balance is insufficient, the program should display an error message.

d. Display the account balance: The program should prompt the user to enter the account number and display the current balance for that account. If the account number is not valid, the program should display an error message.

e. Display the account holder's information: The program should prompt the user to enter the account number and display the account holder's name and current balance for that account. If the account number is not valid, the program should display an error message.

7. Toll Booth Bill Generator

You are required to write a Java program to implement a Toll Booth Bill Generator. The program should be able to calculate the toll fee for each vehicle based on the number of axles and the distance travelled. The program should also be able to generate a bill for each vehicle.

Each vehicle should have the following information:

a. Vehicle type (String)

b. Number of axles (int)

c. Distance Information

Each toll booth should have the following information:

a. Distance travelled (double)

b. Toll Fee Calculation

Core Java Assignment 2

The program should be able to calculate the toll fee for each vehicle based on the following criteria:

    a. Cars, vans, and buses pay a base rate of $0.25 per mile for each axle.

    b. Trucks pay a base rate of $0.50 per mile for each axle.

The program should be able to generate a bill for each vehicle based on the following criteria:

    a. The bill should include the vehicle type, number of axles, distance travelled, toll fee, and total amount due.

    b. The total amount due should include a $2.00 processing fee.

    c. The toll fee calculation and total amount due should not be accessible outside of the class.

The program should have the following methods:

    a. calculateTollFee(): A method to calculate the toll fee for a given vehicle based on the number of axles and distance travelled.

    b. generateBill(): A method to generate a bill for a given vehicle based on the toll fee and total amount due.

    c. showMenu(): A method to show the menu options for the user to input the vehicle information.

The program should have the following fields:

    a. vehicleType: A string field to store the type of vehicle.

    b. numAxles: An integer field to store the number of axles.

    c. distanceTraveled: A double field to store the distance traveled.

    d. tollFee: A double field to store the calculated toll fee.

    e. totalAmountDue: A double field to store the total amount due.

The program should show the following menu options:

    a. Enter vehicle type (car, van, bus, or truck)

    b. Enter number of axles

    c. Enter distance travelled

    d. Calculate toll fee

    e. Generate bill

    f. Exit

8. **Rational Number Calculator**

You are required to write a Java program to perform arithmetic operations on rational numbers. Rational numbers are numbers that can be expressed as a fraction of two integers (i.e., numerator and denominator). The program should take the following inputs from the user:

Two rational numbers (i.e., two pairs of integers representing the numerator and denominator of each number)

The program should then perform the arithmetic operation on the two rational numbers and output the result in the form of a reduced fraction (i.e., the numerator and denominator should be as small as possible).

Example Input

    Enter the first rational number:

    Numerator: 2

    Denominator: 3

    Enter the second rational number:

    Numerator: 1

    Denominator: 6

    Enter the arithmetic operation (+, -, *, /): *

9. **Date Class**

You are required to write a Java program to implement a Date class.
The Date class should have the following fields:
    a. day: An integer field to store the day of the month.
    b. month: An integer field to store the month of the year.
    c. year: An integer field to store the year.

The Date class should have the following methods:
    a. isValid(): A method to check if the date is valid or not. A date is considered valid if it is a valid date of the Gregorian calendar, and has a day, month, and year that are within a reasonable range.
    b. getDayOfWeek(): A method to return the day of the week for the given date, where Sunday is represented by 0, Monday by 1, and so on.
    c. isLeapYear(): A method to check if the year of the given date is a leap year or not.

d. getNextDay(): A method to return the date of the next day.

e. getPreviousDay(): A method to return the date of the previous day.

Example Usage

Date date = new Date(31, 12, 2022);

System.out.println(date.isValid()); // true

System.out.println(date.getDayOfWeek()); // 6 (Saturday)

System.out.println(date.isLeapYear()); // false

Date nextDay = date.getNextDay();

System.out.println(nextDay); // 01-01-2023

Date previousDay = date.getPreviousDay();

System.out.println(previousDay); // 30-12-2022

10. Credit Score Calculator Practice Question

You have been asked to write a Java program to implement a credit score calculator.

The credit score calculator class should have the following fields:

a. creditHistory: An int to represent the length of the individual's credit history.

b. creditUtilization: A double to represent the percentage of available credit the individual is using.

c. paymentHistory: A boolean to represent whether the individual has a good payment history or not.

The credit score calculator class should have the following methods:

a. Constructors

b. Getter and setter methods

c. int calculateCreditScore(): A method to calculate the credit score based on the provided information. The credit score should be calculated using the following formula:

I. If the individual has a good payment history, the credit score should be calculated as follows:

creditScore = (creditHistory * 15) + (int)(creditUtilization * 30) + 55

II. If the individual has a bad payment history, the credit score should be calculated as follows:

creditScore = (creditHistory * 15) + (int)(creditUtilization * 30) + 35