□□□□□□Assignment 5

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)

Soln:-
```
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Scanner;

public class DateTime{
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter date and time (format: dd/mm/yyyy hh:mm:ss): ");
        String input = scanner.nextLine();


        DateTimeFormatter inputFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");
        DateTimeFormatter outputFormatter;


        if (input.contains("/")) {
            if (input.contains(":")) {
                outputFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");
            } else {
                outputFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
            }
        } else if (input.contains(":")) {
            if (input.contains("AM") || input.contains("PM")) {
                outputFormatter = DateTimeFormatter.ofPattern("hh:mm:ss a");
            } else {
                outputFormatter = DateTimeFormatter.ofPattern("HH:mm:ss");
            }
        } else {
            outputFormatter = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");
        }


        LocalDateTime dateTime = LocalDateTime.parse(input, inputFormatter);
        String output = dateTime.format(outputFormatter);
        System.out.println("Converted: " + output);
    }
}
```

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.

Soln:-
```
import java.util.Scanner;

class BMICalculator {
   private double height;
   private double weight;


   public BMICalculator(double height, double weight) {
      this.height = height;
      this.weight = weight;
   }


   public double getHeight() {
      return height;
   }

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

   public double getWeight() {
      return weight;
   }

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

   public double calculateBMI() {
      return weight / (height * height);
   }
}

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


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

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


        BMICalculator bmiCalculator = new BMICalculator(height, weight);


        double bmi = bmiCalculator.calculateBMI();


        System.out.println("Your BMI is: " + bmi);

        scanner.close();
    }
}
```

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

Soln:-
```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class BankAccountManagementSystem {
    static Map<Integer, BankAccount> accounts = new HashMap<>();
    static int nextAccountNumber = 1;

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

        while (true) {
            System.out.println("\nBank Account Management System");
            System.out.println("1. Create a new account");
            System.out.println("2. Deposit money into an account");
            System.out.println("3. Withdraw money from an account");
            System.out.println("4. Display the account balance");
            System.out.println("5. Display the account holder's information");
            System.out.println("6. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine();
            switch (choice) {
                case 1:
                    createAccount(scanner);
                    break;
                case 2:
                    deposit(scanner);
                    break;
                case 3:
                    withdraw(scanner);
                    break;
                case 4:
                    displayBalance(scanner);
                    break;
                case 5:
                    displayAccountInfo(scanner);
                    break;
                case 6:
                    System.out.println("Exiting program...");
                    scanner.close();
                    System.exit(0);
                default:
                    System.out.println("Invalid choice. Please enter a number from 1 to 6.");
            }
        }
    }

      public static void createAccount(Scanner scanner) {
        System.out.print("Enter account holder's name: ");
        String name = scanner.nextLine();

        BankAccount account = new BankAccount(name, nextAccountNumber++);
        accounts.put(account.getAccountNumber(), account);

        System.out.println("Account created successfully. Account number: " + account.getAccountNumber());
    }

      public static void deposit(Scanner scanner) {
        System.out.print("Enter account number: ");
        int accountNumber = scanner.nextInt();

        if (accounts.containsKey(accountNumber)) {
            System.out.print("Enter amount to deposit: ");
            double amount = scanner.nextDouble();

            BankAccount account = accounts.get(accountNumber);
            account.deposit(amount);

            System.out.println("Deposit successful. New balance: $" + account.getBalance());
        } else {
            System.out.println("Error: Account not found.");
        }
    }
```

```
        public static void withdraw(Scanner scanner) {
        System.out.print("Enter account number: ");
        int accountNumber = scanner.nextInt();

        if (accounts.containsKey(accountNumber)) {
            System.out.print("Enter amount to withdraw: ");
            double amount = scanner.nextDouble();

            BankAccount account = accounts.get(accountNumber);
            if (account.withdraw(amount)) {
                System.out.println("Withdrawal successful. New balance: $" + account.getBalance());
            } else {
                System.out.println("Error: Insufficient funds.");
            }
        } else {
            System.out.println("Error: Account not found.");
        }
    }


    public static void displayBalance(Scanner scanner) {
        System.out.print("Enter account number: ");
        int accountNumber = scanner.nextInt();

        if (accounts.containsKey(accountNumber)) {
            BankAccount account = accounts.get(accountNumber);
            System.out.println("Account balance: $" + account.getBalance());
        } else {
            System.out.println("Error: Account not found.");
        }
    }

        public static void displayAccountInfo(Scanner scanner) {
        System.out.print("Enter account number: ");
        int accountNumber = scanner.nextInt();

        if (accounts.containsKey(accountNumber)) {
            BankAccount account = accounts.get(accountNumber);
            System.out.println("Account holder's name: " + account.getAccountHolderName());
            System.out.println("Account balance: $" + account.getBalance());
        } else {
            System.out.println("Error: Account not found.");
        }
    }
}

class BankAccount {
    private String accountHolderName;
    private int accountNumber;
    private double balance;

    public BankAccount(String accountHolderName, int accountNumber) {
        this.accountHolderName = accountHolderName;
        this.accountNumber = accountNumber;
        this.balance = 0.0;
    }

    public String getAccountHolderName() {
        return accountHolderName;
    }
    public int getAccountNumber() {
        return accountNumber;
    }

    public double getBalance() {
        return balance;
    }

    public void deposit(double amount) {
        balance += amount;
    }

    public boolean withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            return true;
        }
        return false;
    }
}
```

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
Core Java Assignment 2
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.

Soln:-
```
public class ElectricityBill {
    private String customerName;
    private double unitsConsumed;
    private double billAmount;

    public ElectricityBill(String customerName, double unitsConsumed) {
        this.customerName = customerName;
        this.unitsConsumed = unitsConsumed;
    }


    public void calculateBillAmount() {
        if (unitsConsumed <= 100) {
            billAmount = unitsConsumed * 5;
        } else if (unitsConsumed <= 300) {
            billAmount = 100 * 5 + (unitsConsumed - 100) * 7;
        } else {
            billAmount = 100 * 5 + 200 * 7 + (unitsConsumed - 300) * 10;
        }
    }


    public double getBillAmount() {
        return billAmount;
    }

     public static void main(String[] args) {

        ElectricityBill bill = new ElectricityBill("Omkar", 250);

        bill.calculateBillAmount();

        System.out.println("Customer Name: " + bill.customerName);
        System.out.println("Units Consumed: " + bill.unitsConsumed);
        System.out.println("Bill Amount: Rs. " + bill.getBillAmount());
    }
}
```

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.

Soln:-
```
import java.util.Scanner;

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


        System.out.println("Enter customer name: ");
        String customerName = scanner.nextLine();

        System.out.println("Enter phone number: ");
        String phoneNumber = scanner.nextLine();

        System.out.println("Enter number of calls made: ");
        int numCalls = scanner.nextInt();

        System.out.println("Enter total duration of calls (in minutes): ");
        int totalDuration = scanner.nextInt();

        scanner.close();


        double flatRate = 10.0; // Flat rate of $10 per month
        double totalCost;

        double callCost = (numCalls <= 100) ? numCalls * 0.5 : (100 * 0.5) + ((numCalls - 100) * 0.25);


        totalDuration = Math.max(totalDuration, numCalls); // Consider minimum duration of 1 minute per call
        totalCost = callCost + flatRate;


        System.out.println("\nCustomer Name: " + customerName);
        System.out.println("Phone Number: " + phoneNumber);
        System.out.println("Total Calls Made: " + numCalls);
        System.out.println("Total Duration of Calls (in minutes): " + totalDuration);
        System.out.println("Flat Rate for the Month: $" + flatRate);
        System.out.println("Total Bill Amount: $" + totalCost);
    }
}
```

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.
Soln:-

```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class BankAccountManagementSystem {
    static Map<Integer, BankAccount> accounts = new HashMap<>();
    static int nextAccountNumber = 1;
```

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

    while (true) {
        System.out.println("\nBank Account Management System");
        System.out.println("1. Create a new account");
        System.out.println("2. Deposit money into an account");
        System.out.println("3. Withdraw money from an account");
        System.out.println("4. Display the account balance");
        System.out.println("5. Display the account holder's information");
        System.out.println("6. Exit");
        System.out.print("Enter your choice: ");
        int choice = scanner.nextInt();
        scanner.nextLine();

        switch (choice) {
            case 1:
                createAccount(scanner);
                break;
            case 2:
                deposit(scanner);
                break;
            case 3:
                withdraw(scanner);
                break;
            case 4:
                displayBalance(scanner);
                break;
            case 5:
                displayAccountInfo(scanner);
                break;
            case 6:
                System.out.println("Exiting program...");
                scanner.close();
                System.exit(0);
            default:
                System.out.println("Invalid choice. Please enter a number from 1 to 6.");
        }
    }
}

    public static void createAccount(Scanner scanner) {
    System.out.print("Enter account holder's name: ");
    String name = scanner.nextLine();

    BankAccount account = new BankAccount(name, nextAccountNumber++);
    accounts.put(account.getAccountNumber(), account);

    System.out.println("Account created successfully. Account number: " + account.getAccountNumber());
}


public static void deposit(Scanner scanner) {
    System.out.print("Enter account number: ");
    int accountNumber = scanner.nextInt();

    if (accounts.containsKey(accountNumber)) {
        System.out.print("Enter amount to deposit: ");
        double amount = scanner.nextDouble();
```

```
            BankAccount account = accounts.get(accountNumber);
            account.deposit(amount);

            System.out.println("Deposit successful. New balance: $" + account.getBalance());
        } else {
            System.out.println("Error: Account not found.");
        }
    }


    public static void withdraw(Scanner scanner) {
        System.out.print("Enter account number: ");
        int accountNumber = scanner.nextInt();

        if (accounts.containsKey(accountNumber)) {
            System.out.print("Enter amount to withdraw: ");
            double amount = scanner.nextDouble();

            BankAccount account = accounts.get(accountNumber);
            if (account.withdraw(amount)) {
                System.out.println("Withdrawal successful. New balance: $" + account.getBalance());
            } else {
                System.out.println("Error: Insufficient funds.");
            }
        } else {
            System.out.println("Error: Account not found.");
        }
    }


    public static void displayBalance(Scanner scanner) {
        System.out.print("Enter account number: ");
        int accountNumber = scanner.nextInt();

        if (accounts.containsKey(accountNumber)) {
            BankAccount account = accounts.get(accountNumber);
            System.out.println("Account balance: $" + account.getBalance());
        } else {
            System.out.println("Error: Account not found.");
        }
    }


    public static void displayAccountInfo(Scanner scanner) {
        System.out.print("Enter account number: ");
        int accountNumber = scanner.nextInt();

        if (accounts.containsKey(accountNumber)) {
            BankAccount account = accounts.get(accountNumber);
            System.out.println("Account holder's name: " + account.getAccountHolderName());
            System.out.println("Account balance: $" + account.getBalance());
        } else {
            System.out.println("Error: Account not found.");
        }
    }
}

class BankAccount {
    private String accountHolderName;
    private int accountNumber;
    private double balance;

    public BankAccount(String accountHolderName, int accountNumber) {
        this.accountHolderName = accountHolderName;
        this.accountNumber = accountNumber;
        this.balance = 0.0;
    }

    public String getAccountHolderName() {
        return accountHolderName;
    }

    public int getAccountNumber() {
        return accountNumber;
    }

    public double getBalance() {
        return balance;
    }

    public void deposit(double amount) {
        balance += amount;
    }

    public boolean withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            return true;
        }
        return false;
    }
}
```

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
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

Soln:-
```
import java.util.Scanner;

public class TollBoothBillGenerator {
    private String vehicleType;
    private int numAxles;
    private double distanceTraveled;
    private double tollFee;
    private double totalAmountDue;

    public static void main(String[] args) {
        TollBoothBillGenerator tollBooth = new TollBoothBillGenerator();
        tollBooth.showMenu();
    }

    public void showMenu() {
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\nToll Booth Bill Generator");
            System.out.println("1. Enter vehicle type (car, van, bus, or truck)");
            System.out.println("2. Enter number of axles");
            System.out.println("3. Enter distance traveled");
            System.out.println("4. Calculate toll fee");
            System.out.println("5. Generate bill");
            System.out.println("6. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine();

            switch (choice) {
                case 1:
                    System.out.print("Enter vehicle type (car, van, bus, or truck): ");
                    vehicleType = scanner.nextLine();
                    break;
                case 2:
                    System.out.print("Enter number of axles: ");
                    numAxles = scanner.nextInt();
                    break;
                case 3:
                    System.out.print("Enter distance traveled: ");
                    distanceTraveled = scanner.nextDouble();
                    break;
                case 4:
                    calculateTollFee();
                    break;
                case 5:
                    generateBill();
                    break;
                case 6:
                    System.out.println("Exiting program...");
                    scanner.close();
                    System.exit(0);
                default:
                    System.out.println("Invalid choice. Please enter a number from 1 to 6.");
            }
        }
    }

    private void calculateTollFee() {
        if (vehicleType != null && numAxles != 0 && distanceTraveled != 0) {
            if (vehicleType.equalsIgnoreCase("car") || vehicleType.equalsIgnoreCase("van") || vehicleType.equalsIgnoreC
                tollFee = 0.25 * numAxles * distanceTraveled;
            } else if (vehicleType.equalsIgnoreCase("truck")) {
                tollFee = 0.50 * numAxles * distanceTraveled;
            } else {
                System.out.println("Invalid vehicle type. Please enter car, van, bus, or truck.");
                return;
            }
            System.out.println("Toll fee calculated: " + tollFee);
        } else {
            System.out.println("Error: Vehicle information incomplete. Please enter vehicle type, number of axles, and dis
        }
    }

    private void generateBill() {
        if (tollFee != 0) {
            totalAmountDue = tollFee + 2.00;
            System.out.println("\nBill Generated:");
            System.out.println("Vehicle Type: " + vehicleType);
            System.out.println("Number of Axles: " + numAxles);
            System.out.println("Distance Traveled: " + distanceTraveled + " miles");
            System.out.println("Toll Fee: $" + tollFee);
            System.out.println("Total Amount Due: $" + totalAmountDue);
        } else {
            System.out.println("Error: Toll fee not calculated. Please calculate toll fee first.");
        }
    }
}
```

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 (+, -, *, /): *
```

Soln:-

```java
import java.util.Scanner;

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


        System.out.println("Enter the first rational number:");
        int num1 = scanner.nextInt();
        int denom1 = scanner.nextInt();

        System.out.println("Enter the second rational number:");
        int num2 = scanner.nextInt();
        int denom2 = scanner.nextInt();


        System.out.println("Enter the arithmetic operation (+, -, *, /):");
        char operation = scanner.next().charAt(0);


        switch (operation) {
            case '+':
                add(num1, denom1, num2, denom2);
                break;
            case '-':
                subtract(num1, denom1, num2, denom2);
                break;
            case '*':
                multiply(num1, denom1, num2, denom2);
                break;
            case '/':
                divide(num1, denom1, num2, denom2);
                break;
            default:
                System.out.println("Invalid operation.");
        }

        scanner.close();
    }


    private static int gcd(int a, int b) {
        while (b != 0) {
            int temp = b;
            b = a % b;
            a = temp;
        }
        return a;
    }
    private static void reduceFraction(int numerator, int denominator) {
        int gcd = gcd(numerator, denominator);
        numerator /= gcd;
        denominator /= gcd;
        System.out.println("Result: " + numerator + "/" + denominator);
    }


    private static void add(int num1, int denom1, int num2, int denom2) {
        int numerator = num1 * denom2 + num2 * denom1;
        int denominator = denom1 * denom2;
        reduceFraction(numerator, denominator);
    }


    private static void subtract(int num1, int denom1, int num2, int denom2) {
        int numerator = num1 * denom2 - num2 * denom1;
        int denominator = denom1 * denom2;
        reduceFraction(numerator, denominator);
    }

        private static void multiply(int num1, int denom1, int num2, int denom2) {
        int numerator = num1 * num2;
        int denominator = denom1 * denom2;
        reduceFraction(numerator, denominator);
    }


    private static void divide(int num1, int denom1, int num2, int denom2) {
        int numerator = num1 * denom2;
        int denominator = denom1 * num2;
        reduceFraction(numerator, denominator);
    }
}
```

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
```

Soln:-

```
public class Date {
    private int day;
    private int month;
    private int year;

        public Date(int day, int month, int year) {
        this.day = day;
        this.month = month;
        this.year = year;
    }

        public boolean isValid() {
        if (year < 1 || month < 1 || month > 12 || day < 1)
            return false;
        int maxDay = 31;
        if (month == 4 || month == 6 || month == 9 || month == 11)
            maxDay = 30;
        else if (month == 2) {
            maxDay = isLeapYear() ? 29 : 28;
        }
        return day <= maxDay;
    }


    public int getDayOfWeek() {
        int y0 = year - (14 - month) / 12;
        int x = y0 + y0/4 - y0/100 + y0/400;
        int m0 = month + 12 * ((14 - month) / 12) - 2;
        return (day + x + (31*m0)/12) % 7;
    }


    public boolean isLeapYear() {
        return (year % 400 == 0) || ((year % 4 == 0) && (year % 100 != 0));
    }


    public Date getNextDay() {
        int nextDay = day + 1;
        int nextMonth = month;
        int nextYear = year;
        int maxDay = 31;
        if (month == 4 || month == 6 || month == 9 || month == 11)
            maxDay = 30;
        else if (month == 2) {
            maxDay = isLeapYear() ? 29 : 28;
        }
        if (nextDay > maxDay) {
            nextDay = 1;
            nextMonth++;
            if (nextMonth > 12) {
                nextMonth = 1;
                nextYear++;
            }
        }
        return new Date(nextDay, nextMonth, nextYear);
    }
    public Date getPreviousDay() {
        int prevDay = day - 1;
        int prevMonth = month;
        int prevYear = year;
        if (prevDay < 1) {
            prevMonth--;
            if (prevMonth < 1) {
                prevMonth = 12;
                prevYear--;
            }
            int maxDay = 31;
            if (prevMonth == 4 || prevMonth == 6 || prevMonth == 9 || prevMonth == 11)
                maxDay = 30;
            else if (prevMonth == 2) {
                maxDay = isLeapYear() ? 29 : 28;
            }
            prevDay = maxDay;
        }
        return new Date(prevDay, prevMonth, prevYear);
    }


    @Override
    public String toString() {
        return String.format("%02d-%02d-%04d", day, month, year);
    }

    // Example Usage
    public static void main(String[] args) {
        Date date = new Date(31, 12, 2022);
        System.out.println("Is Entered date is valid: "+date.isValid());
        System.out.println("The Day of Week is: "+date.getDayOfWeek());
        System.out.println("Is Enter year is leapyear: "+date.isLeapYear());
        Date nextDay = date.getNextDay();
        System.out.println("Next Date of Entered Date: "+nextDay);
        Date previousDay = date.getPreviousDay();
        System.out.println("Previousday of Entered Date: "+previousDay);
    }
}
```

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.
II.
If the individual has a good payment history, the credit score should be calculated as follows:
creditScore = (creditHistory * 15) + (int)(creditUtilization * 30) + 55

If the individual has a bad payment history, the credit score should be calculated as follows:
creditScore = (creditHistory * 15) + (int)(creditUtilization * 30) + 35
Soln:-

```
public class CreditScoreCalculator {
    private int creditHistory;
    private double creditUtilization;
    private boolean paymentHistory;


    public CreditScoreCalculator(int creditHistory, double creditUtilization, boolean paymentHistory) {
        this.creditHistory = creditHistory;
        this.creditUtilization = creditUtilization;
        this.paymentHistory = paymentHistory;
    }


    public int getCreditHistory() {
        return creditHistory;
    }

    public void setCreditHistory(int creditHistory) {
        this.creditHistory = creditHistory;
    }

    public double getCreditUtilization() {
        return creditUtilization;
    }

    public void setCreditUtilization(double creditUtilization) {
        this.creditUtilization = creditUtilization;
    }

    public boolean isPaymentHistory() {
        return paymentHistory;
    }

    public void setPaymentHistory(boolean paymentHistory) {
        this.paymentHistory = paymentHistory;
    }


    public int calculateCreditScore() {
        int creditScore;
        if (paymentHistory) {
            creditScore = (creditHistory * 15) + (int)(creditUtilization * 30) + 55;
        } else {
            creditScore = (creditHistory * 15) + (int)(creditUtilization * 30) + 35;
        }
        return creditScore;
    }


    public static void main(String[] args) {

        CreditScoreCalculator calculator = new CreditScoreCalculator(5, 0.2, true);
        int score = calculator.calculateCreditScore();
        System.out.println("Credit Score: " + score);
    }
}
```