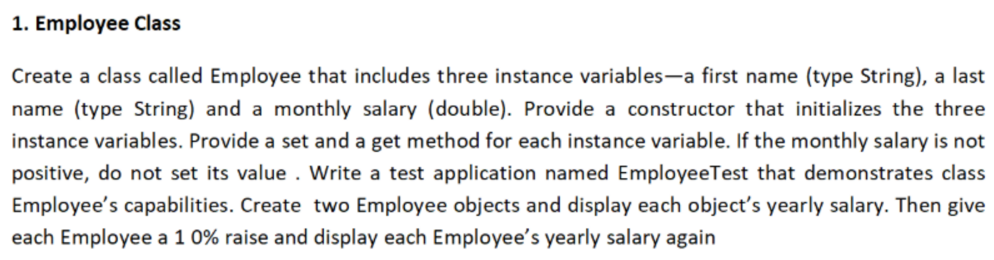
Assignment 7

Contributor: Ngo Thanh Trung - 1677469

1. Employee class



1. Employee Class

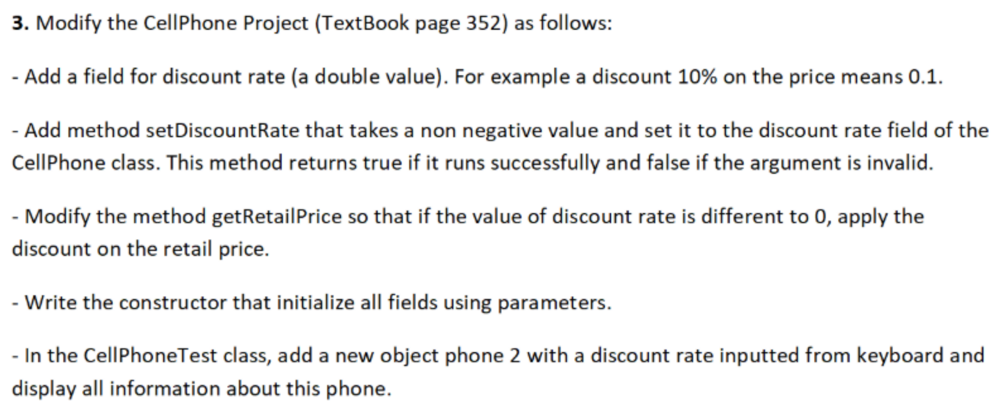
Create a class called Employee that includes three instance variables—a first name (type String), a last name (type String) and a monthly salary (double). Provide a constructor that initializes the three instance variables. Provide a set and a get method for each instance variable. If the monthly salary is not positive, do not set its value . Write a test application named EmployeeTest that demonstrates class Employee's capabilities. Create two Employee objects and display each object's yearly salary. Then give each Employee a 1 0% raise and display each Employee's yearly salary again

Figure 1: Problem 1 Task Requirement

```java
public class Employee {
    private String firstName;
    private String lastname;
    private double monthySalary;

    public Employee(String firstName, String lastname, double monthySalary) {
        this.firstName = firstName;
        this.lastname = lastname;
        if (monthySalary >= 0) {
            this.monthySalary = monthySalary;
        } else {
            this.monthySalary = 0.0;
        }
    }

    public String getFirstName() {
        return firstName;
    }
    public String getLastName() {
        return lastname;
    }
    public double getMonthySalary() {
        return monthySalary;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public void setLastname(String lastname) {
        this.lastname = lastname;
    }
    public void setMonthySalary(double monthySalary) {
        if (monthySalary >= 0) {
            this.monthySalary = monthySalary;
        }
    }
    public double getYearlySalary() {
        // yearly bonus
        return monthySalary * 13;
    }
    public void giveRaise(double raisePercentage) {
        this.monthySalary += monthySalary * raisePercentage / 100;
    }
}

public class EmployeeTest {
    public static void main(String[] args) {
        Employee emp1101 = new Employee("Nguyen", "A", 90);
        Employee emp1100 = new Employee("Dao", "B", 1000);

        System.out.println("Salary of " + emp1101.getFirstName() + " " + emp1101.
            getLastName() + ": " + emp1101.getMonthySalary());
        System.out.println("Salary of " + emp1100.getFirstName() + " " + emp1100.
            getLastName() + ": " + emp1100.getMonthySalary());

        // give each employee a 10% raise
        emp1101.giveRaise(10);
        emp1100.giveRaise(10);

        emp1100.setMonthySalary(-90);
        emp1100.setFirstName("Tien");
        emp1101.setLastname("T");

        System.out.println("Salary of " + emp1101.getFirstName() + " " + emp1101.
            getLastName() + " after 10% raise: " + emp1101.getMonthySalary());
        System.out.println("Salary of " + emp1100.getFirstName() + " " + emp1100.
            getLastName() + " after 10% raise: " + emp1100.getMonthySalary());

        // display yearly salary after the raise
        System.out.println("Yearly salary of " + emp1101.getFirstName() + " " +
            emp1101.getLastName() + " after 10% raise: " + emp1101.getYearlySalary
            ());
        System.out.println("Yearly salary of " + emp1100.getFirstName() + " " +
            emp1100.getLastName() + " after 10% raise: " + emp1100.getYearlySalary
            ());
    }
}
```

1: Employee and EmployeeTest Classes

3. Modify the CellPhone Project



3. Modify the CellPhone Project (TextBook page 352) as follows:

- Add a field for discount rate (a double value). For example a discount 10% on the price means 0.1.

- Add method setDiscountRate that takes a non negative value and set it to the discount rate field of the CellPhone class. This method returns true if it runs successfully and false if the argument is invalid.

- Modify the method getRetailPrice so that if the value of discount rate is different to 0, apply the discount on the retail price.

- Write the constructor that initialize all fields using parameters.

- In the CellPhoneTest class, add a new object phone 2 with a discount rate inputted from keyboard and display all information about this phone.

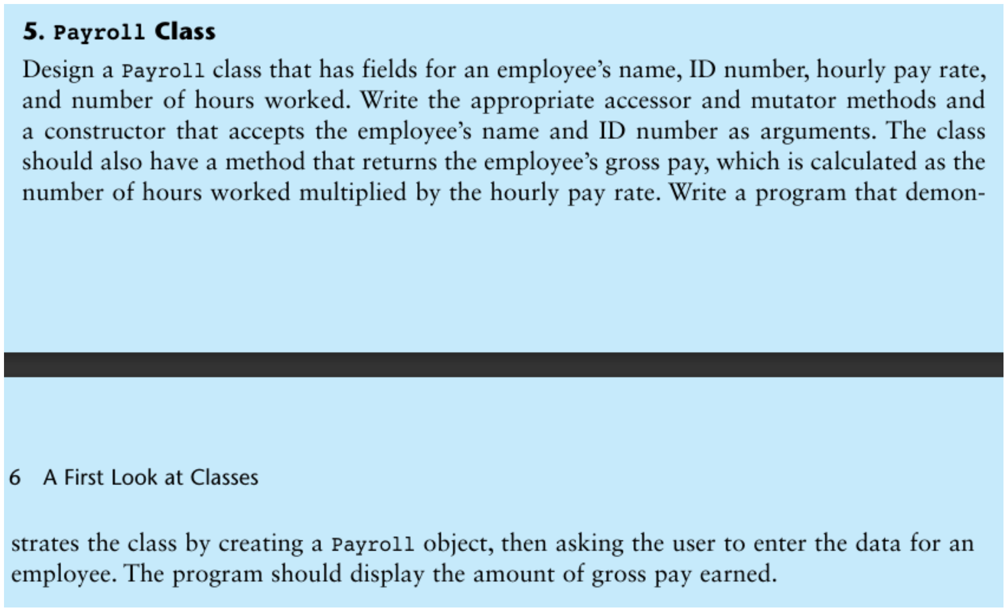
Figure 2: Problem 3 Task Requirement

```java
/**
 * The CellPhone class holds data about a cell phone.
 */

public class CellPhone {

    // Fields
    private String manufact; // Manufacturer
    private String model;    // Model
    private double retailPrice; // Retail price

    private double discountRate;

    /**
     * Constructor
     * @param man The phone's manufacturer.
     * @param mod The phone's model number.
     * @param price The phone's retail price.
     */
    public CellPhone(String man, String mod, double price) {
        manufact = man;
        model = mod;
        retailPrice = price;
    }

    public CellPhone(String man, String mod, double price, double discount) {
        manufact = man;
        model = mod;
        retailPrice = price;
        discountRate = discount;
    }

    /**
     * The setManufact method sets the phone's manufacturer name.
     * @param man The phone's manufacturer.
     */
    public void setManufact(String man) {
        manufact = man;
    }

    /**
     * The setModel method sets the phone's model number.
     * @param mod The phone's model number.
     */
    public void setModel(String mod) {
        model = mod;
    }

    /**
     * The setRetailPrice method sets the phone's retail price.
     * @param price The phone's retail price.
     */
    public void setRetailPrice(double price) {
        retailPrice = price;
    }

    /**
     * getManufact method
     * @return The name of the phone's manufacturer.
     */
    public String getManufact() {
        return manufact;
    }

    /**
     * getModel method
     * @return The phone's model number.
     */
    public String getModel() {
        return model;
    }

    /**
     * getRetailPrice method
     * @return The phone's retail price.
     */
    public double getRetailPrice() {
        if (discountRate > 0){
            retailPrice -= retailPrice * discountRate;
        }
        return retailPrice;
    }

    public boolean setDiscountRate(double d) {
        if(d > 0 && d < 1) {
            this.discountRate = d;
            return true;
        }
        System.out.println("Invalid discount rate");
        return false;
    }

    public double getDiscountRate() {
        return discountRate;
    }
}
```

```java
import java.util.Scanner;

/**
 * This program runs a simple test of the CellPhone class.
 */
public class CellPhoneTest {

    public static void main(String[] args) {
        String testMan; // To hold a manufacturer
        String testMod; // To hold a model number
        double testPrice; // To hold a price
        double discount;

        // Create a Scanner object for keyboard input.
        Scanner keyboard = new Scanner(System.in);

        // Get the manufacturer name.
        System.out.print("Enter the manufacturer: ");
        testMan = keyboard.nextLine();

        // Get the model number.
        System.out.print("Enter the model number: ");
        testMod = keyboard.nextLine();

        // Get the retail price.
        System.out.print("Enter the retail price: ");
        testPrice = keyboard.nextDouble();

        // Create an instance of the CellPhone class,
        // passing the data that was entered as arguments
        // to the constructor.
        CellPhone phone = new CellPhone(testMan, testMod, testPrice);
        CellPhone phone2 = new CellPhone("Apple", "Iphone 20", 9000);

        // Get the data from the phone and display it.
        System.out.println();
        System.out.println("Here is the data that you provided:");
        System.out.println("Manufacturer: " + phone.getManufact());
        System.out.println("Model number: " + phone.getModel());
        System.out.println("Retail price: " + phone.getRetailPrice());

        System.out.println();
        System.out.print("Enter the discount for phone 2: ");
        discount = keyboard.nextDouble();
        phone2.setDiscountRate(discount);

        System.out.println();
        System.out.println("Here is the data that you provided:");
        System.out.println("Manufacturer: " + phone2.getManufact());
        System.out.println("Model number: " + phone2.getModel());
        System.out.println("Discount rate: " + phone2.getDiscountRate());
        System.out.println("Retail price: " + phone2.getRetailPrice());

        keyboard.close();
    }
}
```

2: CellPhone and CellPhoneTest Classes

6.5



5. Payroll Class

Design a Payroll class that has fields for an employee's name, ID number, hourly pay rate, and number of hours worked. Write the appropriate accessor and mutator methods and a constructor that accepts the employee's name and ID number as arguments. The class should also have a method that returns the employee's gross pay, which is calculated as the number of hours worked multiplied by the hourly pay rate. Write a program that demon-

6 A First Look at Classes

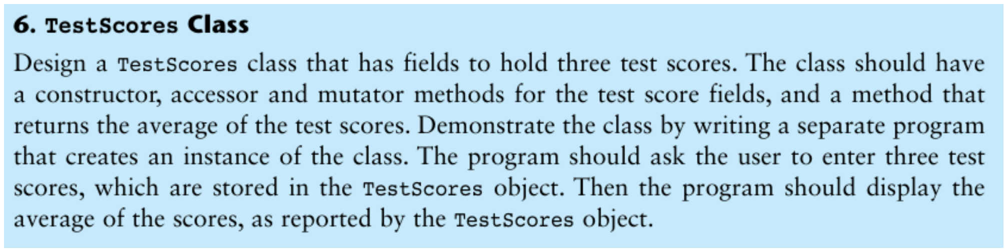
strates the class by creating a Payroll object, then asking the user to enter the data for an employee. The program should display the amount of gross pay earned.

Figure 3: 6.5 Task Requirement

```java
public class payRoll {
    private String employeeName;
    private String id;
    private double hourlyRate = 0.0;
    private int hourWorked = 0;

    payRoll(String employeeName, String id) {
        this.employeeName = employeeName;
        this.id = id;
    }

    public String getName() {
        return employeeName;
    }

    public String getId() {
        return id;
    }

    public double getHourlyRate() {
        return hourlyRate;
    }

    public int getHourWorked() {
        return hourWorked;
    }

    public double getGrossPay() {
        return hourlyRate * hourWorked;
    }

    public void setEmployeeName(String employeeName) {
        this.employeeName = employeeName;
    }

    public void setId(String id) {
        this.id = id;
    }

    public void setHourlyRate(double hourlyRate) {
        if (hourlyRate > 0) {
            this.hourlyRate = hourlyRate;
        }
    }

    public void setHourWorked(int hourWorked) {
        if (hourWorked > 0) {
            this.hourWorked = hourWorked;
        }
    }
}

import java.util.Scanner;

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

        payRoll PRex = new payRoll("test", "0000");

        Scanner sc = new Scanner(System.in);

        System.out.println();
        System.out.println("Please enter the information for a new employee: ");
        System.out.print("Name: ");
        PRex.setEmployeeName(sc.nextLine());

        System.out.print("Employee id: ");
        PRex.setId(sc.nextLine());

        System.out.print("Employee's hourly rate: ");
        PRex.setHourlyRate(sc.nextDouble());

        System.out.print("How many hours have they worked? ");
        PRex.setHourWorked(sc.nextInt());

        System.out.println("Employee " + PRex.getName() + " gross pay: " + PRex.
            getGrossPay());

        sc.close();
    }
}
```

3: payRoll and payRollTest Classes

6.6


6. TestScores Class

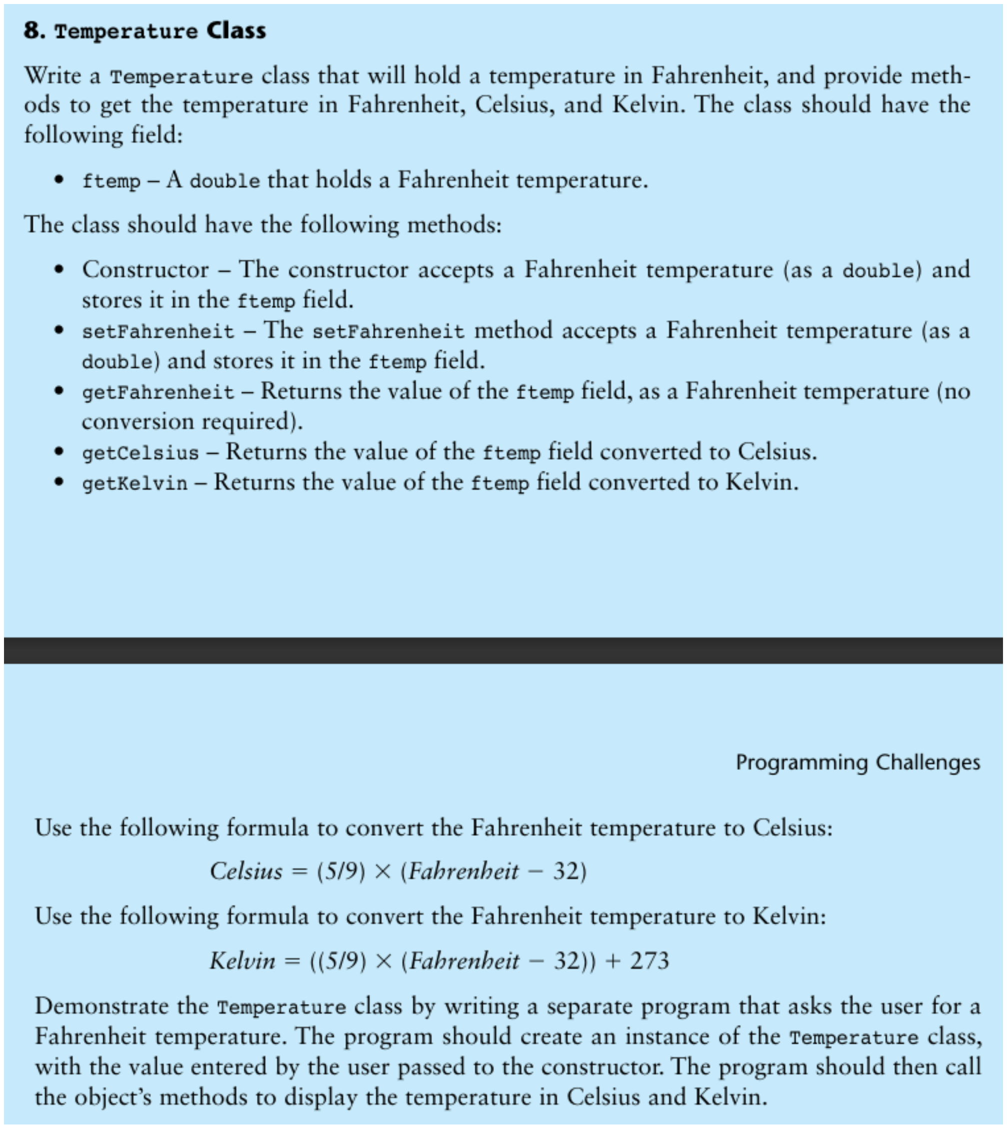
Design a TestScores class that has fields to hold three test scores. The class should have a constructor, accessor and mutator methods for the test score fields, and a method that returns the average of the test scores. Demonstrate the class by writing a separate program that creates an instance of the class. The program should ask the user to enter three test scores, which are stored in the TestScores object. Then the program should display the average of the scores, as reported by the TestScores object.

Figure 4: 6.6 Task Requirement

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        double CS310Score;
        double CS365Score;
        double CS360Score;
        Scanner sc = new Scanner(System.in);

        TestScore TrungTS = new TestScore(10, 10, 10);
        System.out.println(TrungTS.getAverageScore());

        System.out.print("Enter the first subject score: ");
        CS310Score = sc.nextDouble();

        System.out.print("Enter the second subject score: ");
        CS365Score = sc.nextDouble();

        System.out.print("Enter the third subject score: ");
        CS360Score = sc.nextDouble();

        TestScore Fall2024TrungTS = new TestScore(CS310Score, CS365Score,
            CS360Score);
        System.out.println("Your Fall 2024 average tests score: " +
            Fall2024TrungTS.getAverageScore());

        sc.close();
    }
}

public class TestScore {
    private double testScore1;
    private double testScore2;
    private double testScore3;

    TestScore(double testScore1, double testScore2, double testScore3) {
        if ((testScore1 >= 0 && testScore2 >= 0 && testScore3 >= 0) &&
            (testScore1 <= 10 && testScore2 <= 10 && testScore3 <= 10)) {
            this.testScore1 = testScore1;
            this.testScore2 = testScore2;
            this.testScore3 = testScore3;
        }
    }
    public double getTestScore1() {
        return testScore1;
    }

    public double getTestScore2() {
        return testScore2;
    }

    public double getTestScore3() {
        return testScore3;
    }

    public void setTestScore1(double testScore1) {
        if (testScore1 >= 0 && testScore1 <= 10) {
            this.testScore1 = testScore1;
        }
    }

    public void setTestScore2(double testScore2) {
        if (testScore2 >= 0 && testScore2 <= 10) {
            this.testScore2 = testScore2;
        }
    }

    public void setTestScore3(double testScore3) {
        if (testScore3 >= 0 && testScore3 <= 10) {
            this.testScore3 = testScore3;
        }
    }

    public double getAverageScore() {
        return (this.testScore1 + this.testScore2 + this.testScore3) / 3;
    }
}
```

4: Main and TestScore Classes

6.8



8. Temperature Class

Write a `Temperature` class that will hold a temperature in Fahrenheit, and provide methods to get the temperature in Fahrenheit, Celsius, and Kelvin. The class should have the following field:

- `ftemp` – A `double` that holds a Fahrenheit temperature.

The class should have the following methods:

- Constructor – The constructor accepts a Fahrenheit temperature (as a `double`) and stores it in the `ftemp` field.
- `setFahrenheit` – The `setFahrenheit` method accepts a Fahrenheit temperature (as a `double`) and stores it in the `ftemp` field.
- `getFahrenheit` – Returns the value of the `ftemp` field, as a Fahrenheit temperature (no conversion required).
- `getCelsius` – Returns the value of the `ftemp` field converted to Celsius.
- `getKelvin` – Returns the value of the `ftemp` field converted to Kelvin.

Programming Challenges

Use the following formula to convert the Fahrenheit temperature to Celsius:

$$Celsius = (5/9) \times (Fahrenheit - 32)$$

Use the following formula to convert the Fahrenheit temperature to Kelvin:

$$Kelvin = ((5/9) \times (Fahrenheit - 32)) + 273$$

Demonstrate the `Temperature` class by writing a separate program that asks the user for a Fahrenheit temperature. The program should create an instance of the `Temperature` class, with the value entered by the user passed to the constructor. The program should then call the object's methods to display the temperature in Celsius and Kelvin.

Figure 5: 6.8 Task Requirement

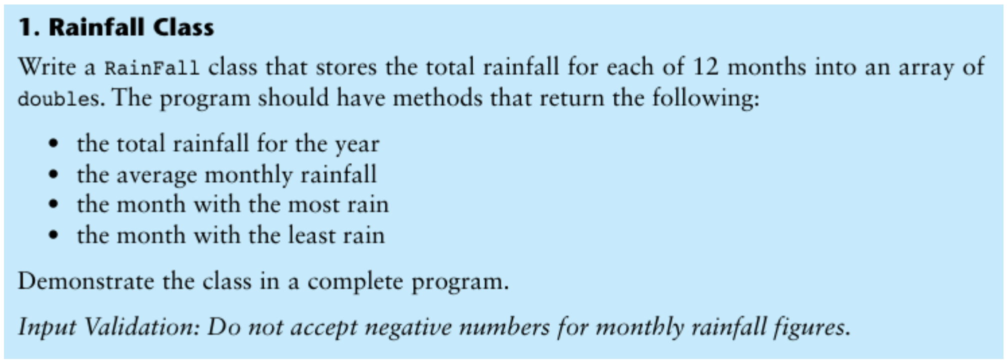
```java
import java.util.Scanner;

public class Temperature {
    private double ftemp;

    Temperature(double ftemp) {
        this.ftemp = ftemp;
    }

    public double getFahrenheit() {
        return ftemp;
    }

    public double getCelsius() {
        return (5.0 / 9) * (ftemp - 32);
    }

    public double getKelvin() {
        return ((5.0 / 9) * (ftemp - 32)) + 273;
    }

    public void setFahrenheit(double Fahrenheit) {
        this.ftemp = Fahrenheit;
    }

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

        System.out.print("Enter a temperature in Fahrenheit: ");
        double ftemp = sc.nextDouble();

        Temperature temp = new Temperature(ftemp);

        System.out.println("Temperature in Fahrenheit: " + temp.getFahrenheit());
        System.out.printf("Temperature in Celsius: %.2f\n", temp.getCelsius());
        System.out.printf("Temperature in Kelvin: %.2f\n", temp.getKelvin());

        sc.close();
    }
}
```

5: Temperature Class

Assignment 8

Contributor: Nguyen Quang Huy - 1677664

7.1



1. Rainfall Class

Write a `RainFall` class that stores the total rainfall for each of 12 months into an array of `double`s. The program should have methods that return the following:

- the total rainfall for the year
- the average monthly rainfall
- the month with the most rain
- the month with the least rain

Demonstrate the class in a complete program.

Input Validation: Do not accept negative numbers for monthly rainfall figures.

Figure 6: 7.1 Task Requirement

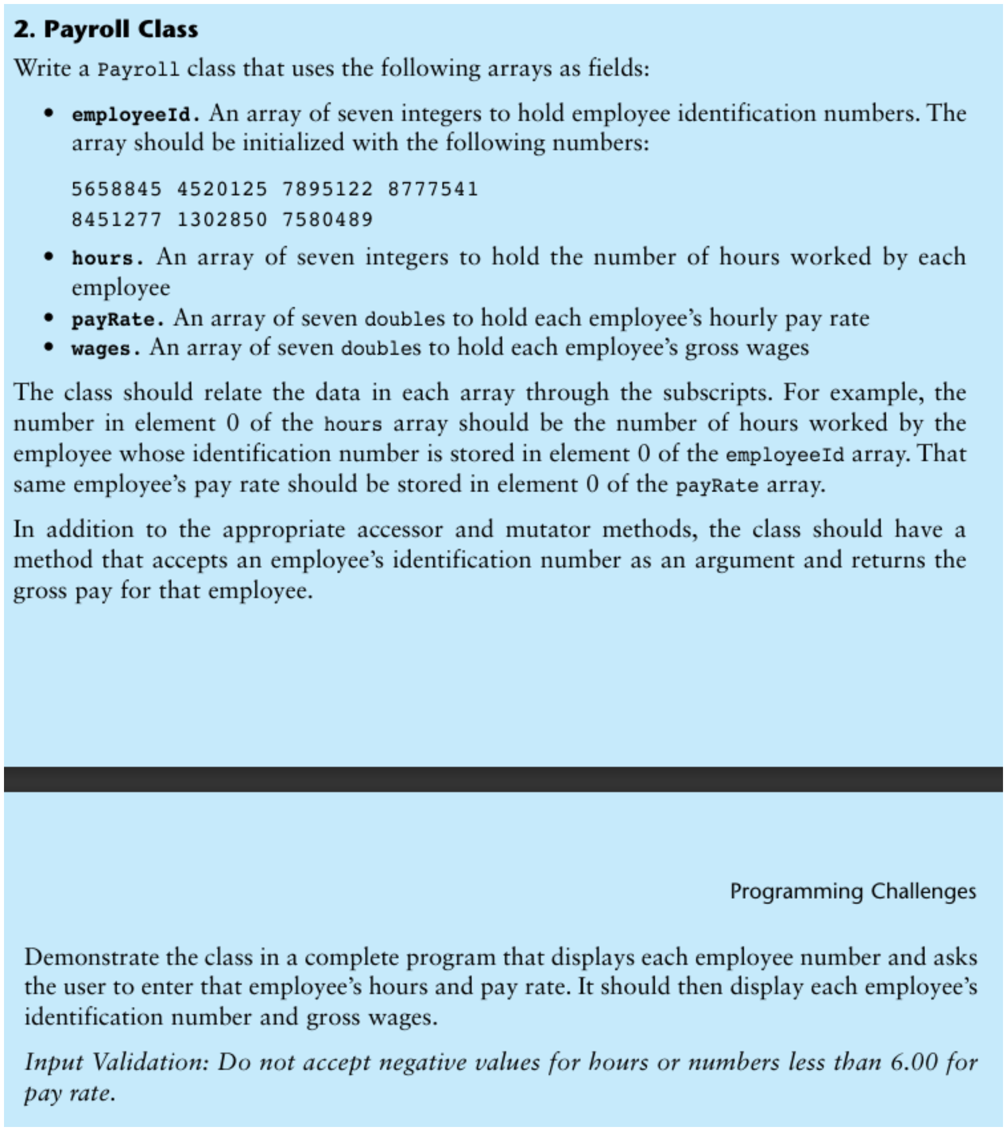
```java
// Rainfall.java
public class Rainfall {
    private double[] monthlyRainfall;

    // Constructor with input validation
    public Rainfall(double[] rainfall) {
        monthlyRainfall = new double[12];
        for (int i = 0; i < 12; i++) {
            if (rainfall[i] < 0) {
                throw new IllegalArgumentException("Rainfall figures cannot be
                    negative.");
            }
            monthlyRainfall[i] = rainfall[i];
        }
    }

    // Calculate total rainfall
    public double getTotalRainfall() {
        double total = 0;
        for (double rain : monthlyRainfall) {
            total += rain;
        }
        return total;
    }

    // Calculate average monthly rainfall
    public double getAverageRainfall() {
        return getTotalRainfall() / monthlyRainfall.length;
    }

    // Find the month with the most rain
    public int getMonthWithMostRain() {
        int month = 0;
        double maxRain = monthlyRainfall[0];
        for (int i = 1; i < monthlyRainfall.length; i++) {
            if (monthlyRainfall[i] > maxRain) {
                maxRain = monthlyRainfall[i];
                month = i;
            }
        }
        return month + 1; // Return month (1-12)
    }

    // Find the month with the least rain
    public int getMonthWithLeastRain() {
        int month = 0;
        double minRain = monthlyRainfall[0];
        for (int i = 1; i < monthlyRainfall.length; i++) {
            if (monthlyRainfall[i] < minRain) {
                minRain = monthlyRainfall[i];
                month = i;
            }
        }
        return month + 1; // Return month (1-12)
    }
}
```

6: Rainfall.java

```java
// RainfallDemo.java
import java.util.Scanner;

public class RainfallDemo {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        double[] rainfall = new double[12];

        System.out.println("Enter the total rainfall for each month:");

        // Input rainfall data with validation
        for (int i = 0; i < 12; i++) {
            while (true) {
                System.out.print("Month " + (i + 1) + ": ");
                rainfall[i] = input.nextDouble();
                if (rainfall[i] < 0) {
                    System.out.println("Rainfall cannot be negative. Please enter
                        a valid number.");
                } else {
                    break;
                }
            }
        }

        // Create Rainfall object and call methods
        Rainfall rainData = new Rainfall(rainfall);

        // Display results
        System.out.printf("Total Rainfall: %.2f\n", rainData.getTotalRainfall());
        System.out.printf("Average Monthly Rainfall: %.2f\n", rainData.
            getAverageRainfall());
        System.out.println("Month with Most Rain: " + rainData.
            getMonthWithMostRain());
        System.out.println("Month with Least Rain: " + rainData.
            getMonthWithLeastRain());

        input.close();
    }
}
```

7: RainFallDemo.java

7.2



2. Payroll Class

Write a `Payroll` class that uses the following arrays as fields:

- **`employeeId.`** An array of seven integers to hold employee identification numbers. The array should be initialized with the following numbers:

  ```
  5658845 4520125 7895122 8777541
  8451277 1302850 7580489
  ```
- **`hours.`** An array of seven integers to hold the number of hours worked by each employee
- **`payRate.`** An array of seven `double`s to hold each employee's hourly pay rate
- **`wages.`** An array of seven `double`s to hold each employee's gross wages

The class should relate the data in each array through the subscripts. For example, the number in element 0 of the `hours` array should be the number of hours worked by the employee whose identification number is stored in element 0 of the `employeeId` array. That same employee's pay rate should be stored in element 0 of the `payRate` array.

In addition to the appropriate accessor and mutator methods, the class should have a method that accepts an employee's identification number as an argument and returns the gross pay for that employee.

Programming Challenges

Demonstrate the class in a complete program that displays each employee number and asks the user to enter that employee's hours and pay rate. It should then display each employee's identification number and gross wages.

Input Validation: Do not accept negative values for hours or numbers less than 6.00 for pay rate.

Figure 7: 7.2 Task Requirement

```java
// Payroll.java
public class Payroll {
    // Declare arrays for fields
    private int[] employeeId = { 5658845, 4520125, 7895122, 8777541, 8451277,
        1302850, 7580489 };
    private int[] hours = new int[7];        // Hours worked by each employee
    private double[] payRate = new double[7]; // Hourly pay rate
    private double[] wages = new double[7];   // Total wages for each employee

    // Set hours worked with input validation
    public void setHours(int index, int hoursWorked) {
        if (hoursWorked < 0) {
            throw new IllegalArgumentException("Hours cannot be negative.");
        }
```

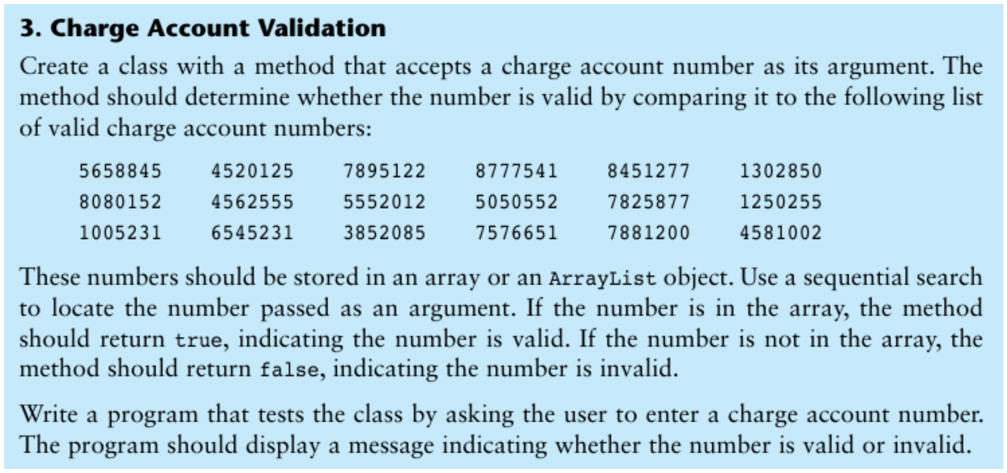
```java
        hours[index] = hoursWorked;
    }

    // Set pay rate with input validation
    public void setPayRate(int index, double rate) {
        if (rate < 6.00) {
            throw new IllegalArgumentException("Pay rate must be at least 6.00.");
        }
        payRate[index] = rate;
    }

    // Calculate wages for each employee
    public void calculateWages() {
        for (int i = 0; i < employeeId.length; i++) {
            wages[i] = hours[i] * payRate[i];
        }
    }

    // Return employee ID
    public int getEmployeeId(int index) {
        return employeeId[index];
    }

    // Return total wages for an employee
    public double getWages(int index) {
        return wages[index];
    }
}
```

8: Payroll.java

```java
// PayrollDemo.java
import java.util.Scanner;

public class PayrollDemo {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        Payroll payroll = new Payroll();

        System.out.println("Enter hours worked and pay rate for each employee:");

        // Loop to input data for each employee
        for (int i = 0; i < 7; i++) {
            int hoursWorked;
            double payRate;

            // Display employee ID
            System.out.println("Employee ID: " + payroll.getEmployeeId(i));

            // Input hours worked with validation
            while (true) {
                System.out.print("Enter hours worked: ");
                hoursWorked = input.nextInt();
                if (hoursWorked >= 0) {
                    break;
                }
                System.out.println("Hours cannot be negative. Please try again.");
            }
            payroll.setHours(i, hoursWorked);
            // Input hourly pay rate with validation
            while (true) {
                System.out.print("Enter hourly pay rate: ");
                payRate = input.nextDouble();
                if (payRate >= 6.00) {
                    break;
                }
                System.out.println("Pay rate must be at least 6.00. Please try again.");
            }
            payroll.setPayRate(i, payRate);
        }

        // Calculate wages
        payroll.calculateWages();

        // Display results
        System.out.println("\nEmployee Wages:");
        for (int i = 0; i < 7; i++) {
            System.out.printf("Employee ID: %d | Gross Wages: %.2f\n",
                              payroll.getEmployeeId(i), payroll.getWages(i));
        }

        input.close();
    }
}
```

9: PayrollDemo.java

7.3



3. Charge Account Validation

Create a class with a method that accepts a charge account number as its argument. The method should determine whether the number is valid by comparing it to the following list of valid charge account numbers:

5658845	4520125	7895122	8777541	8451277	1302850
8080152	4562555	5552012	5050552	7825877	1250255
1005231	6545231	3852085	7576651	7881200	4581002

These numbers should be stored in an array or an `ArrayList` object. Use a sequential search to locate the number passed as an argument. If the number is in the array, the method should return `true`, indicating the number is valid. If the number is not in the array, the method should return `false`, indicating the number is invalid.

Write a program that tests the class by asking the user to enter a charge account number. The program should display a message indicating whether the number is valid or invalid.

Figure 8: 7.3 Task Requirement

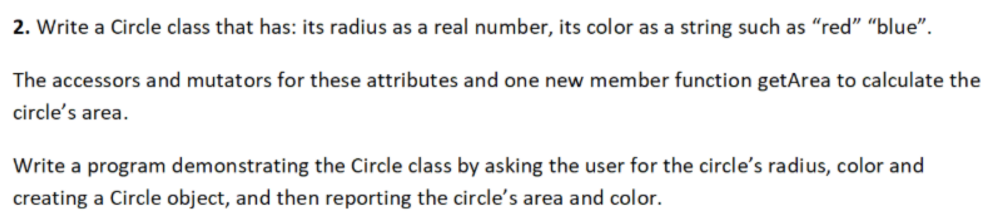
```java
public class ChargeAccount {
    private int[] validAccountNumbers = {
        5658845, 4520125, 7895122, 8777541, 8451277, 1302850,
        8080152, 4562555, 5552012, 5050552, 7825877, 1250255,
        1005231, 6545231, 3852085, 7576651, 7881200, 4581002
    };
    public boolean isValidAccount(int accountNumber) {
        for (int validNumber : validAccountNumbers) {
            if (validNumber == accountNumber) {
                return true;
            }
        }
        return false;
    }
}
```

10: ChargeAccount.java

```java
import java.util.Scanner;

public class ChargeAccountDemo {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        ChargeAccount accountValidator = new ChargeAccount();


        System.out.print("Enter a charge account number to validate: ");
        int accountNumber = input.nextInt();

        if (accountValidator.isValidAccount(accountNumber)) {
            System.out.println("The account number " + accountNumber + " is VALID.
                ");
        } else {
            System.out.println("The account number " + accountNumber + " is
                INVALID.");
        }

        input.close();
    }
}
```

11: ChargeAccountDemo.java

2. Circle class



2. Write a Circle class that has: its radius as a real number, its color as a string such as “red” “blue”.

The accessors and mutators for these attributes and one new member function getArea to calculate the circle’s area.

Write a program demonstrating the Circle class by asking the user for the circle’s radius, color and creating a Circle object, and then reporting the circle’s area and color.

Figure 9: Problem 2 Task Requirement

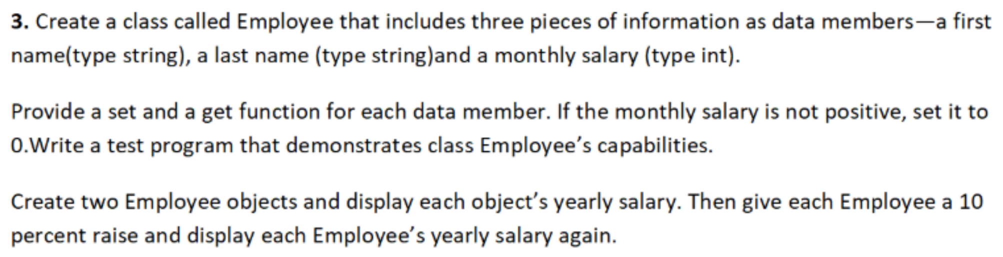
```java
public class Circle {

    private double radius;
    private String color;
    public Circle(double radius, String color) {
        this.radius = radius;
        this.color = color;
    }

    public double getRadius() {
        return radius;
    }

    public String getColor() {
        return color;
    }

    public void setRadius(double radius) {
        this.radius = radius;
    }

    public void setColor(String color) {
        this.color = color;
    }

    public double getArea() {
        return Math.PI * Math.pow(radius, 2);
    }
}
```

12: Circle.java

```java
import java.util.Scanner;

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

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter radius: ");
        double radius = scanner.nextDouble();
        scanner.nextLine();

        String color = "";
        boolean validColor = false;
        while (!validColor) {
            System.out.print("Enter color (red or blue): ");
            color = scanner.nextLine().toLowerCase();
            if (color.equals("red") || color.equals("blue")) {
                validColor = true;
            } else {
                System.out.println("Invalid color. Please enter 'red' or 'blue'.")
                    ;
            }
        }

        Circle circle = new Circle(radius, color);

        System.out.printf("The area is: %.2f\n", circle.getArea());
        System.out.println("The color is: " + circle.getColor());

        scanner.close();
    }
}
```

13: CircleTest.java

3. Employee class



3. Create a class called Employee that includes three pieces of information as data members—a first name(type string), a last name (type string)and a monthly salary (type int).

Provide a set and a get function for each data member. If the monthly salary is not positive, set it to 0.Write a test program that demonstrates class Employee's capabilities.

Create two Employee objects and display each object's yearly salary. Then give each Employee a 10 percent raise and display each Employee's yearly salary again.

Figure 10: Problem 3 Requirement

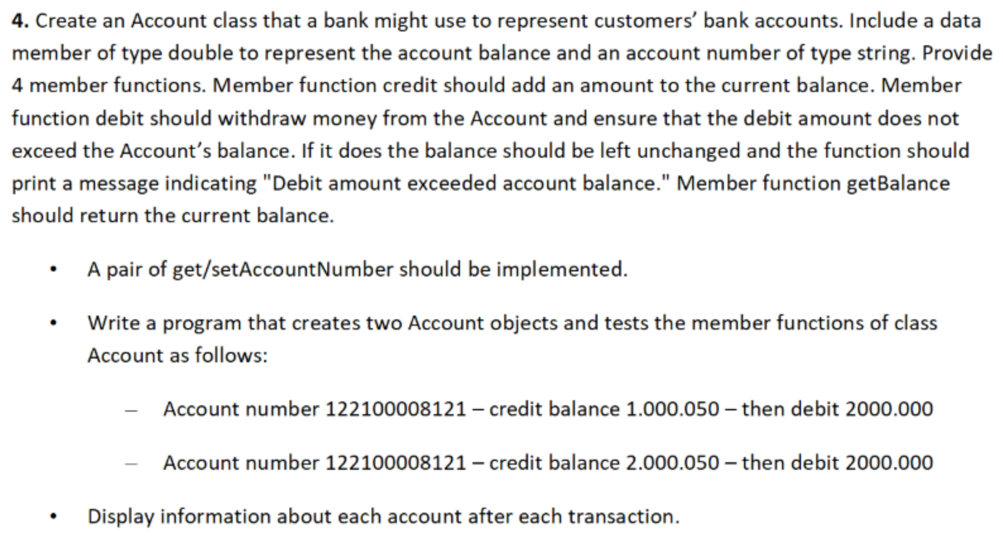
```java
public class Employee {

    private String firstName;
    private String lastName;
    private int monthlySalary;

    public Employee(String firstName, String lastName, int monthlySalary) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.monthlySalary = monthlySalary;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public int getMonthlySalary() {
        return monthlySalary;
    }

    public void setMonthlySalary(int monthlySalary) {
        if (monthlySalary > 0) {
            this.monthlySalary = monthlySalary;
        } else {
            this.monthlySalary = 0;
        }
    }

    public int getYearlySalary() {
        return monthlySalary * 12;
    }

    public void applyRaise() {
        monthlySalary += monthlySalary * 0.10;
    }
}
```

14: Employee.java

```java
public class EmployeeTest {
    public static void main(String[] args) {
        Employee employee1 = new Employee("Quang", "Huy", 3000);
        Employee employee2 = new Employee("Naruto", "Sasuke", 2500);

        System.out.println(employee1.getFirstName() + " " + employee1.getLastName
            () +
            "'s yearly salary: " + employee1.getYearlySalary());
        System.out.println(employee2.getFirstName() + " " + employee2.getLastName
            () +
            "'s yearly salary: " + employee2.getYearlySalary());

        employee1.applyRaise();
        employee2.applyRaise();

        System.out.println("After 10% raise:");
        System.out.println(employee1.getFirstName() + " " + employee1.getLastName
            () +
            "'s yearly salary: " + employee1.getYearlySalary());
        System.out.println(employee2.getFirstName() + " " + employee2.getLastName
            () +
            "'s yearly salary: " + employee2.getYearlySalary());
    }
}
```

15: EmployeeTest.java

4. Account class



4. Create an Account class that a bank might use to represent customers' bank accounts. Include a data member of type double to represent the account balance and an account number of type string. Provide 4 member functions. Member function credit should add an amount to the current balance. Member function debit should withdraw money from the Account and ensure that the debit amount does not exceed the Account's balance. If it does the balance should be left unchanged and the function should print a message indicating "Debit amount exceeded account balance." Member function getBalance should return the current balance.

- A pair of get/setAccountNumber should be implemented.
- Write a program that creates two Account objects and tests the member functions of class Account as follows:
  - Account number 122100008121 – credit balance 1.000.050 – then debit 2000.000
  - Account number 122100008121 – credit balance 2.000.050 – then debit 2000.000
- Display information about each account after each transaction.

Figure 11: Problem 4 Requirement

```java
public class Account {
    private double balance;
    private String accountNumber;

    public Account(String accNumber, double initialBalance) {
        accountNumber = accNumber;
        balance = initialBalance;
    }

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

    public void debit(double amount) {
        if (amount > balance) {
            System.out.println("Debit amount exceeded account balance.");
        } else {
            balance -= amount;
        }
    }

    public double getBalance() {
        return balance;
    }

    public String getAccountNumber() {
        return accountNumber;
    }

    public void setAccountNumber(String accNumber) {
        accountNumber = accNumber;
    }
    public void displayAccountInfo() {
        System.out.println("Account number: " + accountNumber);
        System.out.println("Balance: " + balance);
    }
}
```

16: Account.java

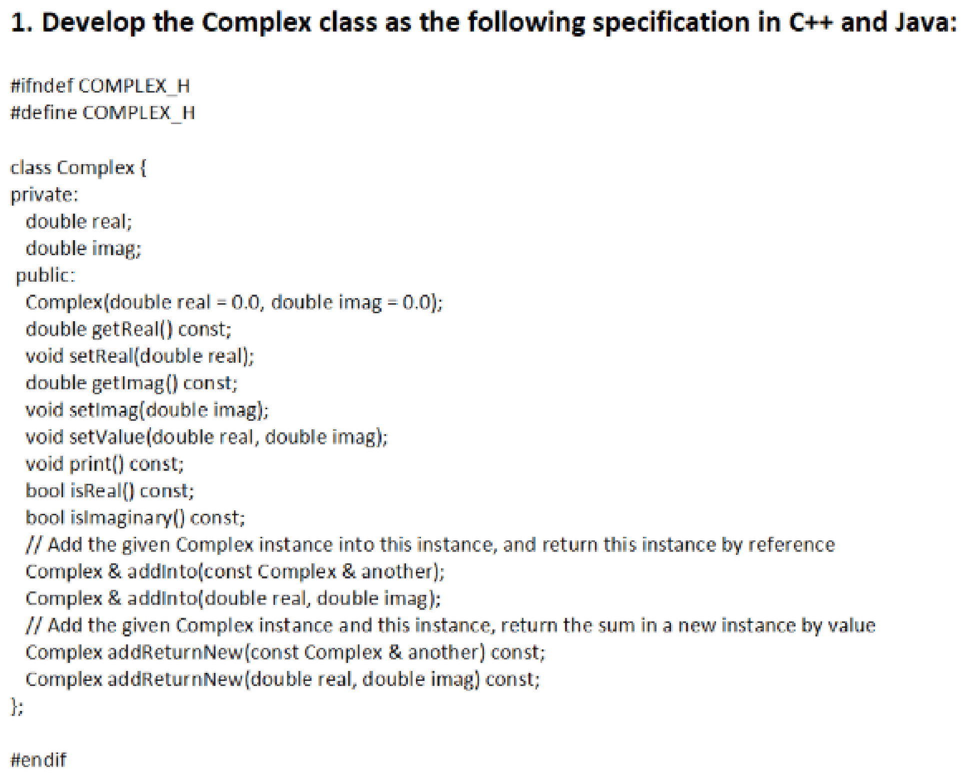
```java
public class ATM {
    public static void main(String[] args) {
        Account account1 = new Account("122100008121", 1000050);
        Account account2 = new Account("122100008121", 2000050);

        System.out.println("Before transactions:");
        account1.displayAccountInfo();
        account2.displayAccountInfo();

        account1.credit(1000050);
        account1.debit(2000000);

        account2.credit(2000050);
        account2.debit(2000000);

        System.out.println("After transaction:");
        account1.displayAccountInfo();
        account2.displayAccountInfo();
    }
}
```

17: ATM.java

Assignment 10

Contributor: Nguyen Duy Khoi - 1677395

1. Complex class



1. Develop the Complex class as the following specification in C++ and Java:

```
#ifndef COMPLEX_H
#define COMPLEX_H

class Complex {
private:
  double real;
  double imag;
 public:
  Complex(double real = 0.0, double imag = 0.0);
  double getReal() const;
  void setReal(double real);
  double getImag() const;
  void setImag(double imag);
  void setValue(double real, double imag);
  void print() const;
  bool isReal() const;
  bool isImaginary() const;
  // Add the given Complex instance into this instance, and return this instance by reference
  Complex & addInto(const Complex & another);
  Complex & addInto(double real, double imag);
  // Add the given Complex instance and this instance, return the sum in a new instance by value
  Complex addReturnNew(const Complex & another) const;
  Complex addReturnNew(double real, double imag) const;
};

#endif
```

Figure 12: Problem 1

1. C++

```cpp
#include "Complex.h"
#include <iostream>

using namespace std;

Complex::Complex(double real, double imag) : real(real), imag(imag) {}

double Complex::getReal() const {
    return real;
}

void Complex::setReal(double real) {
    this->real = real;
}

double Complex::getImag() const {
    return imag;
}

void Complex::setImag(double imag) {
    this->imag = imag;
}

void Complex::setValue(double real, double imag) {
    this->real = real;
    this->imag = imag;
}

void Complex::print() const {
    cout << real;
    if (imag >= 0) {
        cout << " + " << imag << "i" << endl;
    } else {
        cout << " - " << -imag << "i" << endl;
    }
}

bool Complex::isReal() const {
    return imag == 0.0;
}

bool Complex::isImaginary() const {
    return real == 0.0;
}

Complex & Complex::addInto(const Complex & another) {
    this->real += another.real;
    this->imag += another.imag;
    return *this;
}

Complex & Complex::addInto(double real, double imag) {
    this->real += real;
    this->imag += imag;
    return *this;
}

Complex Complex::addReturnNew(const Complex & another) const {
    return Complex(this->real + another.real, this->imag + another.imag);
}

Complex Complex::addReturnNew(double real, double imag) const {
    return Complex(this->real + real, this->imag + imag);
}
```

18: Complex.cpp

```cpp
#ifndef COMPLEX_H
#define COMPLEX_H

class Complex {
private:
    double real;
    double imag;

public:
    Complex(double real = 0.0, double imag = 0.0);

    double getReal() const;
    void setReal(double real);

    double getImag() const;
    void setImag(double imag);
    void setValue(double real, double imag);

    void print() const;

    bool isReal() const;
    bool isImaginary() const;

    Complex & addInto(const Complex & another);
    Complex & addInto(double real, double imag);

    Complex addReturnNew(const Complex & another) const;
    Complex addReturnNew(double real, double imag) const;
};

#endif
```

19: Complex.h

```cpp
#include <iostream>
#include "Complex.h"

int main() {
    Complex c1(2.0, 3.0);
    Complex c2(1.0, 4.0);
    Complex c3 = c1.addReturnNew(c2);
    c1.print();
    c2.print();
    c3.print();

    c1.addInto(c2);
    c1.print();

    return 0;
}
```

20: Main.cpp

1. **Java**

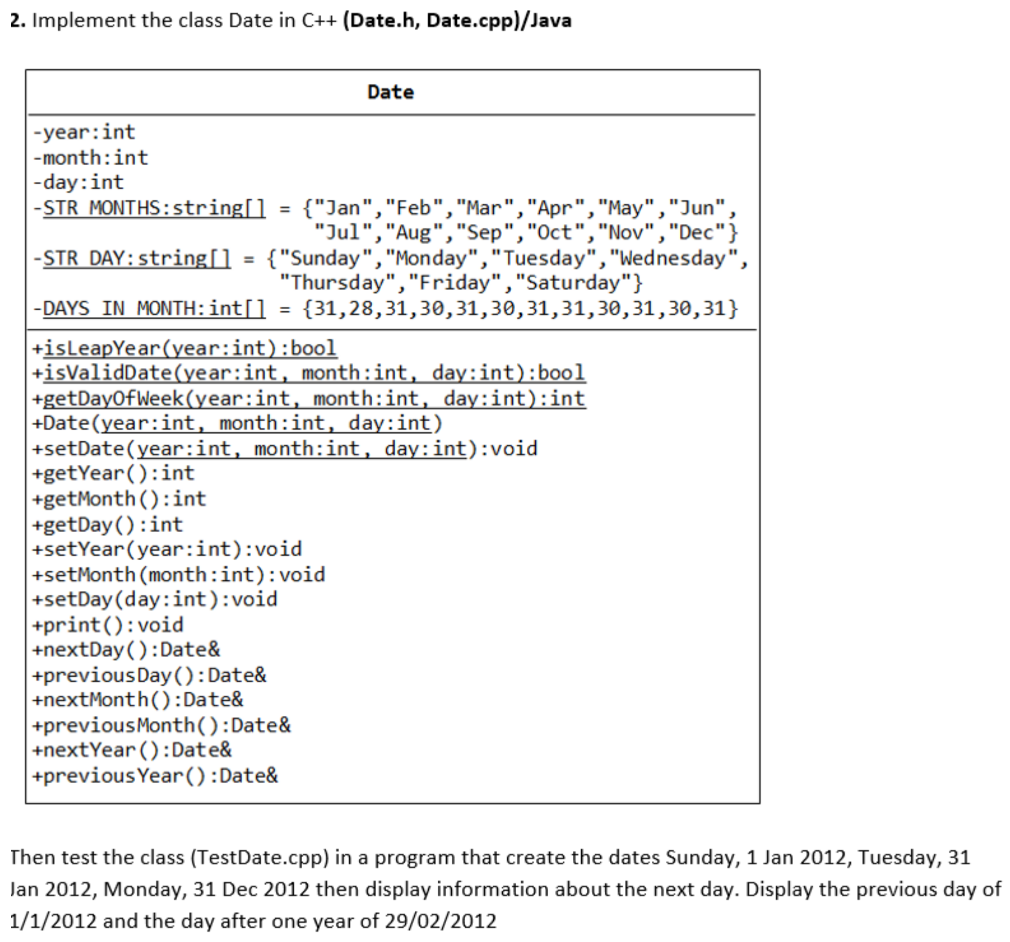
```java
public class Complex {
    private double real;
    private double imag;

    public Complex(double real, double imag) {
        this.real = real;
        this.imag = imag;
    }

    public Complex() {
        this(0.0, 0.0);
    }

    public double getReal() {
        return real;
    }

    public void setReal(double real) {
        this.real = real;
    }

    public double getImag() {
        return imag;
    }

    public void setImag(double imag) {
        this.imag = imag;
    }

    public void setValue(double real, double imag) {
        this.real = real;
        this.imag = imag;
    }

    public void print() {
        System.out.print(real);
        if (imag >= 0) {
            System.out.println(" + " + imag + "i");
        } else {
            System.out.println(" - " + -imag + "i");
        }
    }

    public boolean isReal() {
        return imag == 0.0;
    }

    public boolean isImaginary() {
        return real == 0.0;
    }

    public Complex addInto(Complex another) {
        this.real += another.real;
        this.imag += another.imag;
        return this;
    }

    public Complex addInto(double real, double imag) {
        this.real += real;
        this.imag += imag;
        return this;
    }

    public Complex addReturnNew(Complex another) {
        return new Complex(this.real + another.real, this.imag + another.imag);
    }

    public Complex addReturnNew(double real, double imag) {
        return new Complex(this.real + real, this.imag + imag);
    }
}
```

21: Complex.java

```java
public class TestComplex {
    public static void main(String[] args) {
        Complex c1 = new Complex(2.0, 3.0);
        Complex c2 = new Complex(1.0, 4.0);
        Complex c3 = c1.addReturnNew(c2);
        c1.print();
        c2.print();
        c3.print();

        c1.addInto(c2);
        c1.print();
    }
}
```

22: TestComplex.java

2. Date class



2. Implement the class Date in C++ **(Date.h, Date.cpp)/Java**

```
Date
-year:int
-month:int
-day:int
-STR_MONTHS:string[] = {"Jan","Feb","Mar","Apr","May","Jun",
                         "Jul","Aug","Sep","Oct","Nov","Dec"}
-STR_DAY:string[] = {"Sunday","Monday","Tuesday","Wednesday",
                      "Thursday","Friday","Saturday"}
-DAYS_IN_MONTH:int[] = {31,28,31,30,31,30,31,31,30,31,30,31}
+isLeapYear(year:int):bool
+isValidDate(year:int, month:int, day:int):bool
+getDayOfWeek(year:int, month:int, day:int):int
+Date(year:int, month:int, day:int)
+setDate(year:int, month:int, day:int):void
+getYear():int
+getMonth():int
+getDay():int
+setYear(year:int):void
+setMonth(month:int):void
+setDay(day:int):void
+print():void
+nextDay():Date&
+previousDay():Date&
+nextMonth():Date&
+previousMonth():Date&
+nextYear():Date&
+previousYear():Date&
```

Then test the class (TestDate.cpp) in a program that create the dates Sunday, 1 Jan 2012, Tuesday, 31 Jan 2012, Monday, 31 Dec 2012 then display information about the next day. Display the previous day of 1/1/2012 and the day after one year of 29/02/2012

Figure 13: Problem 2

2. C++

```
#include "Date.h"
#include <iostream>
#include <string>
```

```cpp
Date::Date(int day, int month, int year) : day(day), month(month), year(year) {}

int Date::getDay() const {
    return day;
}

int Date::getMonth() const {
    return month;
}

int Date::getYear() const {
    return year;
}

void Date::setDay(int day) {
    this->day = day;
}

void Date::setMonth(int month) {
    this->month = month;
}

void Date::setYear(int year) {
    this->year = year;
}

bool Date::isValid() const {
    if (month < 1 || month > 12 || day < 1 || day > 31) {
        return false;
    }

    if (month == 2) {
        if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
        }
        return day <= 28;
    }

    if (month == 4 || month == 6 || month == 9 || month == 11) {
        return day <= 30;
    }

    return true;
}

std::string Date::toString() const {
    return std::to_string(day) + "/" + std::to_string(month) + "/" + std::to_
       string(year);
}
```

23: Date.cpp

```cpp
#ifndef DATE_H
#define DATE_H
#include <string>

class Date {
private:
    int day;
    int month;
    int year;

public:
    Date(int day, int month, int year);

    int getDay() const;
    int getMonth() const;
    int getYear() const;

    void setDay(int day);
    void setMonth(int month);
    void setYear(int year);

    bool isValid() const;

    std::string toString() const;
};

#endif
```

24: Date.h

```cpp
#include <iostream>
#include "Date.h"

Date getNextDay(const Date &d) {
    int day = d.getDay();
    int month = d.getMonth();
    int year = d.getYear();

    day++;
    Date temp(day, month, year);

    if (!temp.isValid()) {
        day = 1;
        month++;
        temp.setDay(day);
        temp.setMonth(month);

        if (!temp.isValid()) {
            month = 1;
            year++;
            temp.setMonth(month);
            temp.setYear(year);
        }
    }
    return temp;
}

Date getPreviousDay(const Date &d) {
    int day = d.getDay();
    int month = d.getMonth();
    int year = d.getYear();

    day--;
    Date temp(day, month, year);

    if (!temp.isValid()) {
        month--;
        if (month < 1) {
            month = 12;
            year--;
        }

        temp.setMonth(month);
        temp.setYear(year);
        if (month == 2) {
            temp.setDay((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0) ?
                29 : 28);
        } else if (month == 4 || month == 6 || month == 9 || month == 11) {
            temp.setDay(30);
        } else {
            temp.setDay(31);
        }
    }
    return temp;
}

Date getDateAfterOneYear(const Date &d) {
    int day = d.getDay();
    int month = d.getMonth();
    int year = d.getYear();

    year++;
    Date temp(day, month, year);

    if (!temp.isValid()) {
        if (month == 2 && day == 29 && !(year % 4 == 0 && year % 100 != 0) && !(
            year % 400 == 0)) {
            temp.setDay(28);
        }
    }
    return temp;
}
```

25: DateFunctions.cpp

```cpp
#include <iostream>
#include "Date.h"

int main() {
    Date date1(1, 1, 2012);
    Date date2(31, 1, 2012);
    Date date3(31, 12, 2012);

    std::cout << "Current Date: " << date1.toString() << " | Next Day: " <<
        getNextDay(date1).toString() << std::endl;
    std::cout << "Current Date: " << date2.toString() << " | Next Day: " <<
        getNextDay(date2).toString() << std::endl;
    std::cout << "Current Date: " << date3.toString() << " | Next Day: " <<
        getNextDay(date3).toString() << std::endl;

    Date prevDay = getPreviousDay(date1);
    std::cout << "Previous Day of 1/1/2012: " << prevDay.toString() << std::endl;

    Date leapYearDate(29, 2, 2012);
    Date nextYearDate = getDateAfterOneYear(leapYearDate);
    std::cout << "One year after 29/2/2012: " << nextYearDate.toString() << std::
        endl;

    return 0;
}
```

26: main.cpp

2. Java

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

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

    // Getters
    public int getDay() { return day; }
    public int getMonth() { return month; }
    public int getYear() { return year; }

    // Setters
    public void setDay(int day) { this.day = day; }
    public void setMonth(int month) { this.month = month; }
    public void setYear(int year) { this.year = year; }

    // Display method
    public void display() {
        System.out.println(day + "/" + month + "/" + year);
    }

    // Convert to String
    public String toString() {
        return day + "/" + month + "/" + year;
    }

    // Validation method
    public boolean isValid() {
        if (month < 1 || month > 12 || day < 1 || day > 31) {
            return false;
        }
        if (month == 2) {
            if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
                return day <= 29;
            }
            return day <= 28;
        }
        if (month == 4 || month == 6 || month == 9 || month == 11) {
            return day <= 30;
        }
        return true;
    }
}
```

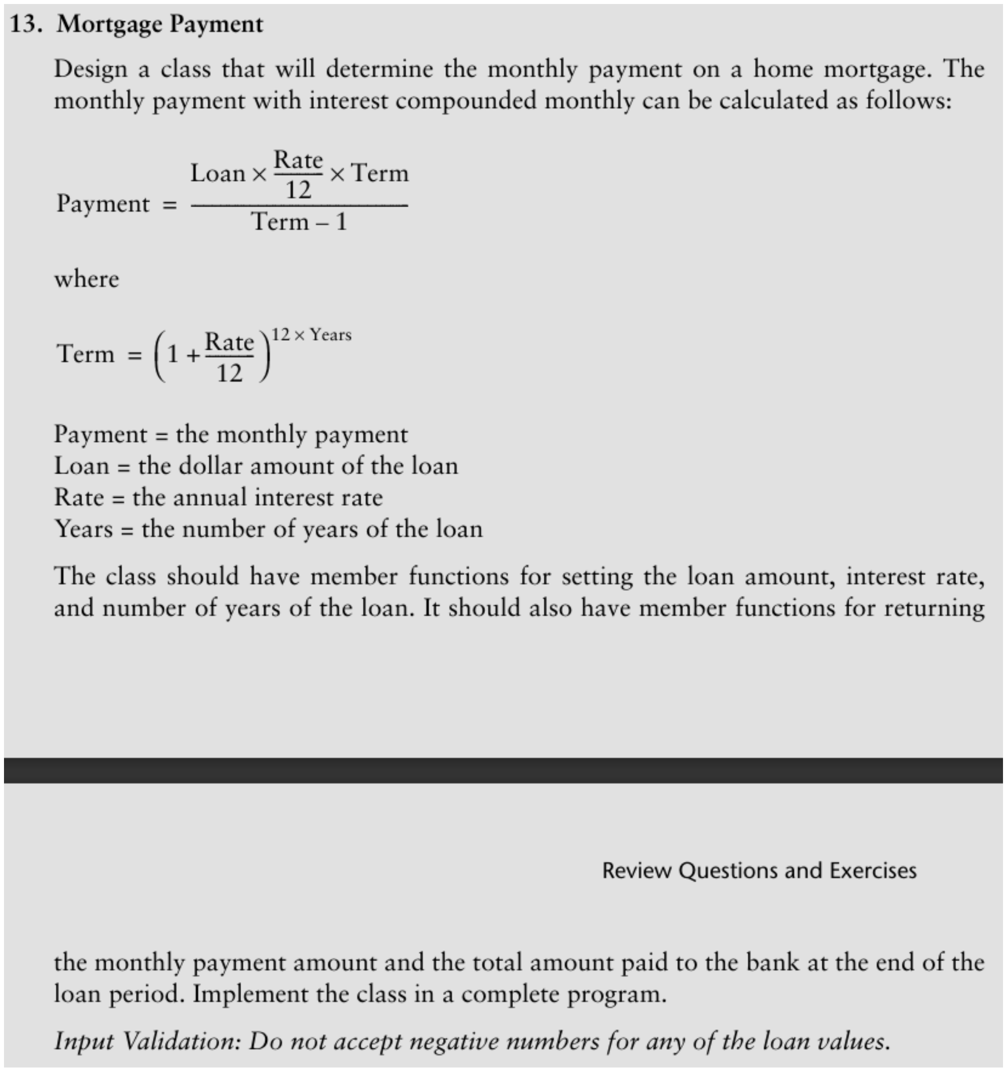
27: Date.java

```
public class TestDate {

    public static Date getNextDay(Date d) {
        int day = d.getDay();
        int month = d.getMonth();
        int year = d.getYear();
        day++;
        Date temp = new Date(day, month, year);

        if (!temp.isValid()) {
            day = 1;
            month++;
            temp.setDay(day);
            temp.setMonth(month);

            if (!temp.isValid()) {
                month = 1;
                year++;
                temp.setMonth(month);
                temp.setYear(year);
            }
        }
        return temp;
    }

    public static Date getPreviousDay(Date d) {
        int day = d.getDay();
        int month = d.getMonth();
        int year = d.getYear();
        day--;
        Date temp = new Date(day, month, year);

        if (!temp.isValid()) {
            month--;
            if (month < 1) {
                month = 12;
                year--;
            }
            temp.setMonth(month);
            temp.setYear(year);
            if (month == 2) {
                temp.setDay((year % 4 == 0 && year % 100 != 0) || (year % 400 ==
                    0) ? 29 : 28);
            } else if (month == 4 || month == 6 || month == 9 || month == 11) {
                temp.setDay(30);
            } else {
                temp.setDay(31);
            }
        }
        return temp;
    }

    public static Date getDateAfterOneYear(Date d) {
        int day = d.getDay();
        int month = d.getMonth();
```

```java
        int year = d.getYear();
        year++;
        Date temp = new Date(day, month, year);

        if (!temp.isValid()) {
            if (month == 2 && day == 29 && !(year % 4 == 0 && year % 100 != 0) &&
                !(year % 400 == 0)) {
                temp.setDay(28);
            }
        }
        return temp;
    }

    public static void main(String[] args) {
        Date date1 = new Date(1, 1, 2012);
        Date date2 = new Date(31, 1, 2012);
        Date date3 = new Date(31, 12, 2012);

        System.out.println("Current Date: " + date1.toString() + " | Next Day: " +
             getNextDay(date1).toString());
        System.out.println("Current Date: " + date2.toString() + " | Next Day: " +
             getNextDay(date2).toString());
        System.out.println("Current Date: " + date3.toString() + " | Next Day: " +
             getNextDay(date3).toString());

        Date prevDay = getPreviousDay(date1);
        System.out.println("Previous Day of 1/1/2012: " + prevDay.toString());

        Date leapYearDate = new Date(29, 2, 2012);
        Date nextYearDate = getDateAfterOneYear(leapYearDate);
        System.out.println("One year after 29/2/2012: " + nextYearDate.toString())
            ;
    }
}
```

28: TestDate.java

13.13



13. Mortgage Payment

Design a class that will determine the monthly payment on a home mortgage. The monthly payment with interest compounded monthly can be calculated as follows:

$$\text{Payment} = \frac{\text{Loan} \times \frac{\text{Rate}}{12} \times \text{Term}}{\text{Term} - 1}$$

where

$$\text{Term} = \left(1 + \frac{\text{Rate}}{12}\right)^{12 \times \text{Years}}$$

Payment = the monthly payment
Loan = the dollar amount of the loan
Rate = the annual interest rate
Years = the number of years of the loan

The class should have member functions for setting the loan amount, interest rate, and number of years of the loan. It should also have member functions for returning

Review Questions and Exercises

the monthly payment amount and the total amount paid to the bank at the end of the loan period. Implement the class in a complete program.

Input Validation: Do not accept negative numbers for any of the loan values.

Figure 14: 13.13 Task Requirement

13.13. C++

```cpp
#include <iostream>
#include <cmath>
#include "Mortgage.h"

using namespace std;

Mortgage::Mortgage(double loan, double rate, int years) {
    this->loan = loan;
    this->rate = rate;
    this->years = years;
}

void Mortgage::setLoan(double loan) {
    this->loan = loan;
}

void Mortgage::setRate(double rate) {
    this->rate = rate;
}

void Mortgage::setYears(int years) {
    this->years = years;
}

double Mortgage::calculateMonthlyPayment() {
    double monthlyRate = rate / 100 / 12;
    int totalMonths = years * 12;
    double term = pow(1 + monthlyRate, totalMonths);

    double payment = (loan * monthlyRate * term) / (term - 1);
    return payment;
}

double Mortgage::calculateTotalAmountPaid() {
    double monthlyPayment = calculateMonthlyPayment();
    int totalMonths = years * 12;
    return monthlyPayment * totalMonths;
}

void Mortgage::displayLoanDetails() {
    double monthlyPayment = calculateMonthlyPayment();
    double totalPaid = calculateTotalAmountPaid();

    cout << "Loan Amount: $" << loan << endl;
    cout << "Annual Interest Rate: " << rate << "%" << endl;
    cout << "Loan Term: " << years << " years" << endl;
    cout << "Monthly Payment: $" << monthlyPayment << endl;
    cout << "Total Amount Paid: $" << totalPaid << endl;
}
```

29: Mortage.cpp

```cpp
#ifndef MORTGAGE_H
#define MORTGAGE_H

class Mortgage {
private:
    double loan;
    double rate;
    int years;

public:
    Mortgage(double loan, double rate, int years);

    void setLoan(double loan);
    void setRate(double rate);
    void setYears(int years);

    double calculateMonthlyPayment();
    double calculateTotalAmountPaid();
    void displayLoanDetails();
};
```

```
#endif
```

30: Mortage.h

```
#include <iostream>
#include "Mortgage.h"
using namespace std;

int main() {
    double loan;
    double rate;
    int years;

    do {
        cout << "Enter the loan amount (positive number): ";
        cin >> loan;
        if (loan <= 0) {
            cout << "Invalid input! Loan amount must be positive." << endl;
        }
    } while (loan <= 0);

    do {
        cout << "Enter the annual interest rate (positive number): ";
        cin >> rate;
        if (rate <= 0) {
            cout << "Invalid input! Interest rate must be positive." << endl;
        }
    } while (rate <= 0);

    do {
        cout << "Enter the number of years of the loan (positive number): ";
        cin >> years;
        if (years <= 0) {
            cout << "Invalid input! Number of years must be positive." << endl;
        }
    } while (years <= 0);

    Mortgage mortgage(loan, rate, years);
    mortgage.displayLoanDetails();

    return 0;
}
```

31: main.cpp

13.13. Java

```
// Mortgage.java
public class Mortgage {
    private double loan;
    private double rate;
    private int years;

    // Constructor
    public Mortgage(double loan, double rate, int years) {
        this.loan = loan;
        this.rate = rate;
        this.years = years;
```

```java
    }

    // Setters
    public void setLoan(double loan) {
        this.loan = loan;
    }

    public void setRate(double rate) {
        this.rate = rate;
    }

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

    // Calculate Monthly Payment
    public double calculateMonthlyPayment() {
        double monthlyRate = rate / 100 / 12;
        int totalMonths = years * 12;
        double term = Math.pow(1 + monthlyRate, totalMonths);

        double payment = (loan * monthlyRate * term) / (term - 1);
        return payment;
    }

    // Calculate Total Amount Paid
    public double calculateTotalAmountPaid() {
        double monthlyPayment = calculateMonthlyPayment();
        int totalMonths = years * 12;
        return monthlyPayment * totalMonths;
    }

    // Display Loan Details
    public void displayLoanDetails() {
        double monthlyPayment = calculateMonthlyPayment();
        double totalPaid = calculateTotalAmountPaid();

        System.out.printf("Loan Amount: $%.2f\n", loan);
        System.out.printf("Annual Interest Rate: %.2f%%\n", rate);
        System.out.printf("Loan Term: %d years\n", years);
        System.out.printf("Monthly Payment: $%.2f\n", monthlyPayment);
        System.out.printf("Total Amount Paid: $%.2f\n", totalPaid);
    }
}
```

32: Mortage.java

```java
// MortgageTest.java
import java.util.Scanner;

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

        // Input loan amount
        double loan = -1;
        while (loan <= 0) {
            System.out.print("Enter the loan amount (positive number): ");
            loan = scanner.nextDouble();
            if (loan <= 0) {
                System.out.println("Invalid input! Loan amount must be positive.");
            }
        }

        // Input interest rate
        double rate = -1;
        while (rate <= 0) {
            System.out.print("Enter the annual interest rate (positive number): ");
            rate = scanner.nextDouble();
            if (rate <= 0) {
                System.out.println("Invalid input! Interest rate must be positive.");
            }
        }

        // Input number of years
        int years = -1;
        while (years <= 0) {
            System.out.print("Enter the number of years of the loan (positive number): ");
            years = scanner.nextInt();
            if (years <= 0) {
                System.out.println("Invalid input! Number of years must be positive.");
            }
        }

        // Create mortgage object and display details
        Mortgage mortgage = new Mortgage(loan, rate, years);
        mortgage.displayLoanDetails();

        scanner.close();
    }
}
```

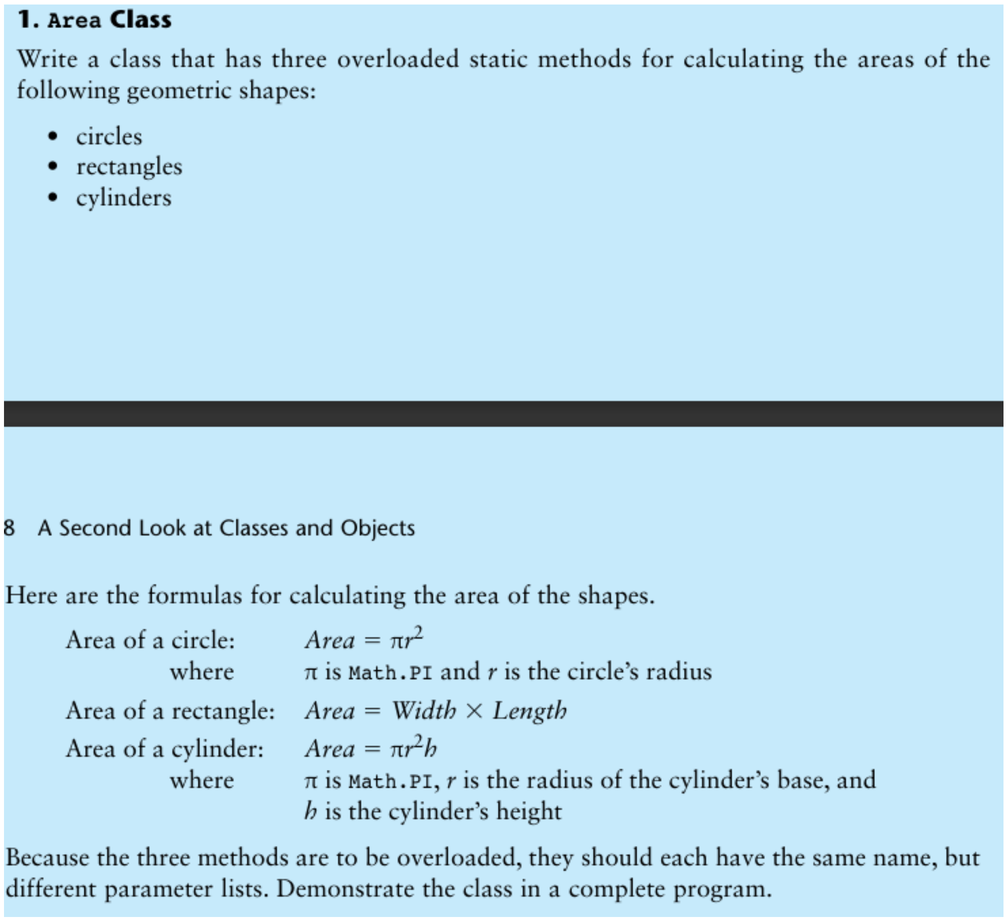
33: MortageTest.java

Assignment 11

Contributor: Nguyen Duy Duc - 1624838

Java

8.1



1. Area Class

Write a class that has three overloaded static methods for calculating the areas of the following geometric shapes:

- circles
- rectangles
- cylinders

8 A Second Look at Classes and Objects

Here are the formulas for calculating the area of the shapes.

Area of a circle:	*Area* = πr^2
where	π is `Math.PI` and r is the circle's radius
Area of a rectangle:	*Area* = *Width* $\times$ *Length*
Area of a cylinder:	*Area* = $\pi r^2 h$
where	π is `Math.PI`, r is the radius of the cylinder's base, and h is the cylinder's height

Because the three methods are to be overloaded, they should each have the same name, but different parameter lists. Demonstrate the class in a complete program.

Figure 15: 8.1 Task Requirement

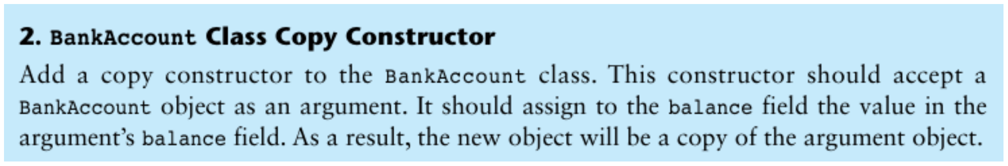
```java
public class Area {

    // Area of a circle
    public static double calculateArea(double radius) {
        return Math.PI * radius * radius;
    }

    // Area of a rectangle
    public static double calculateArea(double width, double length) {
        return width * length;
    }

    // Area of a cylinder
    public static double calculateArea(double radius, double height, boolean
        isCylinder) {
        return Math.PI * radius * radius * height;
    }
}
```

34: Area.java

```java
import java.util.Scanner;

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

        System.out.println("Select a shape to calculate the area:");
        System.out.println("1. Circle");
        System.out.println("2. Rectangle");
        System.out.println("3. Cylinder");

        int choice = scanner.nextInt();

        switch (choice) {
            case 1:
                // Circle
                System.out.print("Enter the radius of the circle: ");
                double radiusCircle = scanner.nextDouble();
                double circleArea = Area.calculateArea(radiusCircle);
                System.out.printf("The area of the circle is: %.2f\n", circleArea)
                    ;
                break;

            case 2:
                // Rectangle
                System.out.print("Enter the width of the rectangle: ");
                double width = scanner.nextDouble();
                System.out.print("Enter the length of the rectangle: ");
                double length = scanner.nextDouble();
                double rectangleArea = Area.calculateArea(width, length);
                System.out.printf("The area of the rectangle is: %.2f\n",
                    rectangleArea);
                break;

            case 3:
                // Cylinder
                System.out.print("Enter the radius of the cylinder: ");
                double radiusCylinder = scanner.nextDouble();
                System.out.print("Enter the height of the cylinder: ");
                double height = scanner.nextDouble();
                double cylinderArea = Area.calculateArea(radiusCylinder, height,
                    true);
                System.out.printf("The area of the cylinder is: %.2f\n",
                    cylinderArea);
                break;

            default:
                System.out.println("Invalid choice! Please select 1, 2, or 3.");
        }

        scanner.close();
    }
}
```

35: AreaDemo.java

8.2



2. `BankAccount` Class Copy Constructor

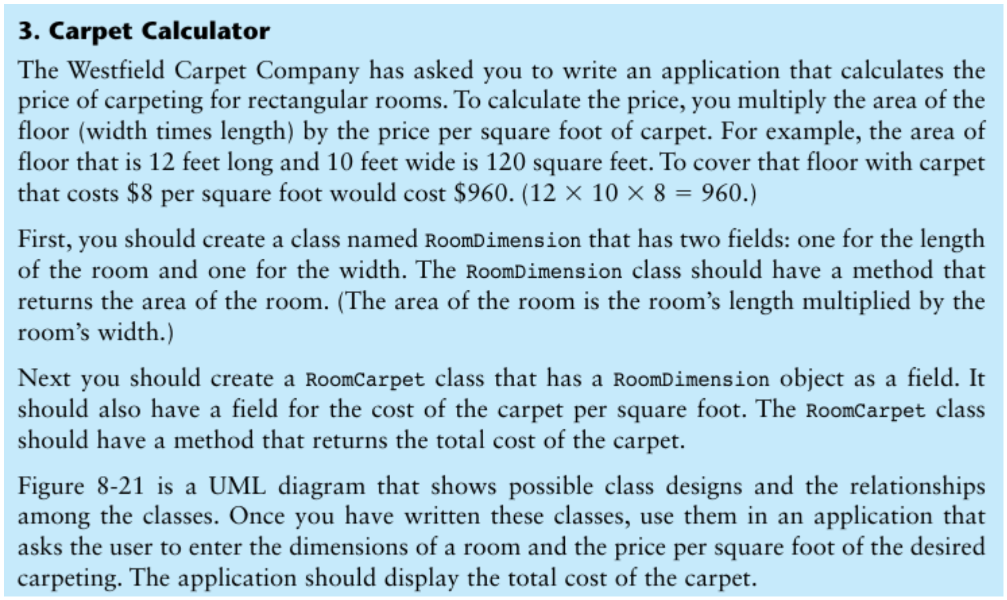
Add a copy constructor to the `BankAccount` class. This constructor should accept a `BankAccount` object as an argument. It should assign to the `balance` field the value in the argument's `balance` field. As a result, the new object will be a copy of the argument object.

Figure 16: 8.2 Task Requirement

```java
public class BankAccount {
    private double balance;

    // Constructor
    public BankAccount(double initialBalance) {
        balance = initialBalance;
    }

    // Copy constructor
    public BankAccount(BankAccount account) {
        this.balance = account.balance;
    }

    public double getBalance() {
        return balance;
    }

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

    public void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
        } else {
            System.out.println("Insufficient funds.");
        }
    }

    public static void main(String[] args) {
        // Create a BankAccount object
        BankAccount account1 = new BankAccount(1000.0);

        // Create a copy of account1
        BankAccount account2 = new BankAccount(account1);

        // Display the balances
        System.out.println("Balance of account1: " + account1.getBalance());
        System.out.println("Balance of account2: " + account2.getBalance());
    }
}
```

36: BankAccount.java

8.3



3. Carpet Calculator

The Westfield Carpet Company has asked you to write an application that calculates the price of carpeting for rectangular rooms. To calculate the price, you multiply the area of the floor (width times length) by the price per square foot of carpet. For example, the area of floor that is 12 feet long and 10 feet wide is 120 square feet. To cover that floor with carpet that costs $8 per square foot would cost $960. ($12 \times 10 \times 8 = 960$.)

First, you should create a class named `RoomDimension` that has two fields: one for the length of the room and one for the width. The `RoomDimension` class should have a method that returns the area of the room. (The area of the room is the room's length multiplied by the room's width.)

Next you should create a `RoomCarpet` class that has a `RoomDimension` object as a field. It should also have a field for the cost of the carpet per square foot. The `RoomCarpet` class should have a method that returns the total cost of the carpet.

Figure 8-21 is a UML diagram that shows possible class designs and the relationships among the classes. Once you have written these classes, use them in an application that asks the user to enter the dimensions of a room and the price per square foot of the desired carpeting. The application should display the total cost of the carpet.

Figure 17: 8.3 Task Requirement

```java
import java.util.Scanner;

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

        // Get the dimensions
        System.out.print("Enter the length of the room in feet: ");
        double length = scanner.nextDouble();

        System.out.print("Enter the width of the room in feet: ");
        double width = scanner.nextDouble();

        // Create a RoomDimension object
        RoomDimension roomDimension = new RoomDimension(length, width);

        // Get the cost per square foot of the carpet
        System.out.print("Enter the cost per square foot of the carpet: ");
        double costPerSquareFoot = scanner.nextDouble();

        // Create a RoomCarpet object
        RoomCarpet roomCarpet = new RoomCarpet(roomDimension, costPerSquareFoot);

        // Display the total cost
        System.out.println("The total cost of the carpet: $" + roomCarpet.
            getTotalCost());
    }
}
```

37: CarpetCalculator.java

```java
class RoomCarpet {
    private RoomDimension dimension;
    private double costPerSquareFoot;

    // Constructor
    public RoomCarpet(RoomDimension dimension, double costPerSquareFoot) {
        this.dimension = dimension;
        this.costPerSquareFoot = costPerSquareFoot;
    }

    public double getTotalCost() {
        return dimension.getArea() * costPerSquareFoot;
    }
}
```

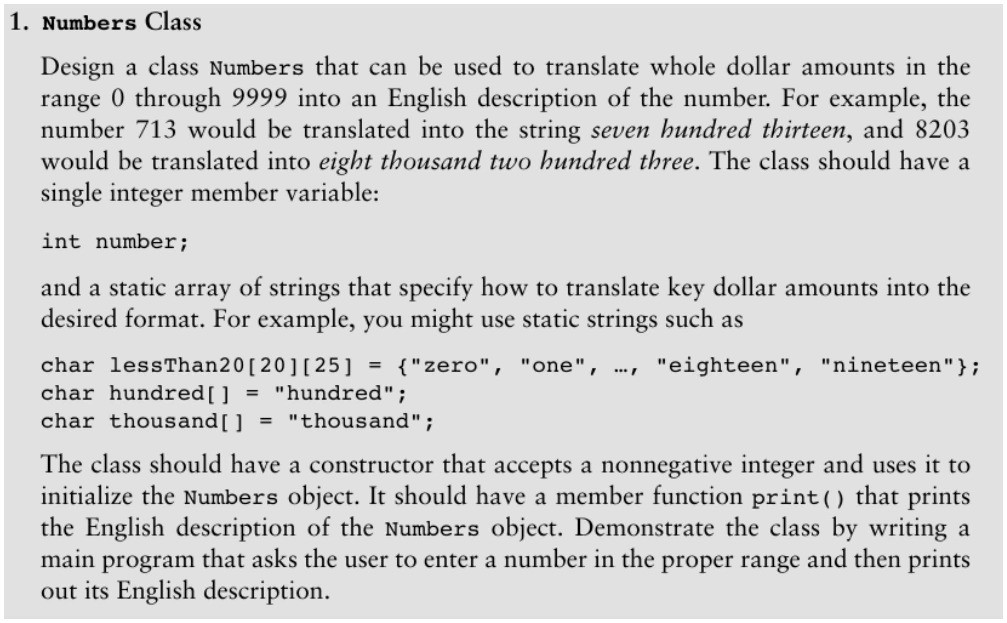
38: RoomCarpet.java

```java
class RoomDimension {
    private double length;
    private double width;

    // Constructor
    public RoomDimension(double length, double width) {
        this.length = length;
        this.width = width;
    }

    public double getArea() {
        return length * width;
    }
}
```

39: RoomDimension.java

C++

14.1



1. `Numbers` Class

Design a class `Numbers` that can be used to translate whole dollar amounts in the range 0 through 9999 into an English description of the number. For example, the number 713 would be translated into the string *seven hundred thirteen*, and 8203 would be translated into *eight thousand two hundred three*. The class should have a single integer member variable:

```
int number;
```

and a static array of strings that specify how to translate key dollar amounts into the desired format. For example, you might use static strings such as

```
char lessThan20[20][25] = {"zero", "one", …, "eighteen", "nineteen"};
char hundred[] = "hundred";
char thousand[] = "thousand";
```

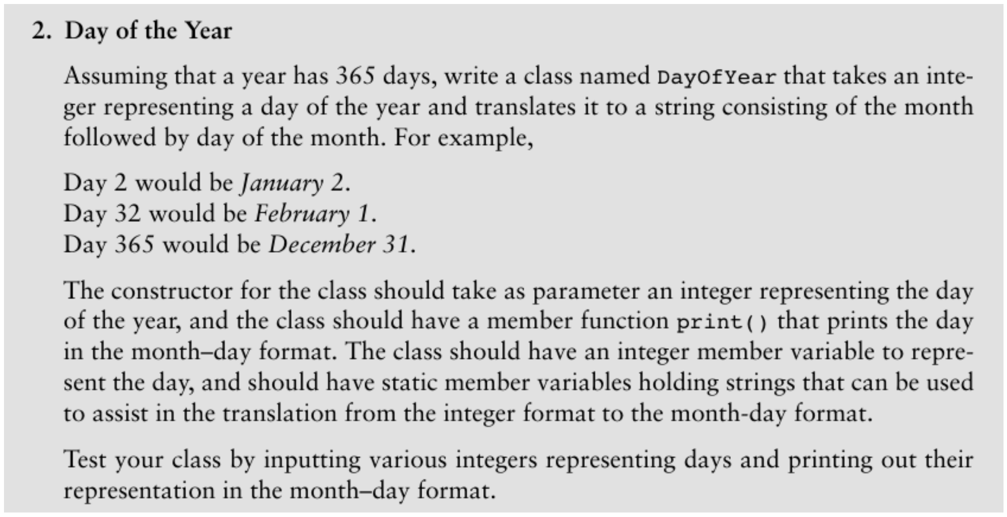
The class should have a constructor that accepts a nonnegative integer and uses it to initialize the `Numbers` object. It should have a member function `print()` that prints the English description of the `Numbers` object. Demonstrate the class by writing a main program that asks the user to enter a number in the proper range and then prints out its English description.

Figure 18: 14.1 Task Requirement

```cpp
#include <iostream>
#include <string>

using namespace std;

class Numbers {
private:
    int number;
    static string lessThan20[20];
    static string tens[10];
    static string hundred;
    static string thousand;

public:
    // Constructor
    Numbers(int num) : number(num) {}

    // Function to print the English description of the number
    void print() const {
        if (number == 0) {
            cout << lessThan20[0] << endl;
            return;
        }

        string result;
        int tempNumber = number;

        if (tempNumber >= 1000) {
            result += lessThan20[tempNumber / 1000] + " " + thousand + " ";
            tempNumber %= 1000;
        }
        if (tempNumber >= 100) {
            result += lessThan20[tempNumber / 100] + " " + hundred + " ";
            tempNumber %= 100;
        }
        if (tempNumber >= 20) {
            result += tens[tempNumber / 10] + " ";
            tempNumber %= 10;
        }
        if (tempNumber > 0) {
            result += lessThan20[tempNumber] + " ";
        }

        cout << result << endl;
    }
};

// Static member initialization
string Numbers::lessThan20[20] = {"zero", "one", "two", "three", "four", "five", "
    six", "seven", "eight", "nine", "ten", "eleven", "twelve", "thirteen", "
    fourteen", "fifteen", "sixteen", "seventeen", "eighteen", "nineteen"};
string Numbers::tens[10] = {"", "", "twenty", "thirty", "forty", "fifty", "sixty",
     "seventy", "eighty", "ninety"};
string Numbers::hundred = "hundred";
string Numbers::thousand = "thousand";

int main() {
    int num;

    // Loop until a valid number is entered
    do {
        cout << "Enter a number between 0 and 9999: ";
        cin >> num;

        if (num < 0 || num > 9999) {
            cout << "Number out of range! Please try again." << endl;
        }
    } while (num < 0 || num > 9999);

    Numbers number(num);
    number.print();

    return 0;
}
```

40: Numbers.cpp

14.2



2. Day of the Year

Assuming that a year has 365 days, write a class named `DayOfYear` that takes an integer representing a day of the year and translates it to a string consisting of the month followed by day of the month. For example,

Day 2 would be *January 2*.
Day 32 would be *February 1*.
Day 365 would be *December 31*.

The constructor for the class should take as parameter an integer representing the day of the year, and the class should have a member function `print()` that prints the day in the month–day format. The class should have an integer member variable to represent the day, and should have static member variables holding strings that can be used to assist in the translation from the integer format to the month-day format.

Test your class by inputting various integers representing days and printing out their representation in the month–day format.

Figure 19: 14.2 Task Requirement

```cpp
#include <iostream>
#include <string>

using namespace std;

class DayOfYear {
private:
    int day;
    static string months[12];
    static int daysInMonth[12];

public:
    // Constructor
    DayOfYear(int dayOfYear) : day(dayOfYear) {}

    // Function to print the day in month-day format
    void print() const {
        int month = 0;
        int dayOfMonth = day;

        while (dayOfMonth > daysInMonth[month]) {
            dayOfMonth -= daysInMonth[month];
            month++;
        }

        cout << months[month] << " " << dayOfMonth << endl;
    }
};

// Static member initialization
string DayOfYear::months[12] = {"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"};
int DayOfYear::daysInMonth[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

int main() {
    int dayOfYear;

    // Loop until a valid day is entered
    do {
        cout << "Enter a day of the year (1-365): ";
        cin >> dayOfYear;

        if (dayOfYear < 1 || dayOfYear > 365) {
            cout << "Day out of range! Please try again." << endl;
        }
    } while (dayOfYear < 1 || dayOfYear > 365);

    DayOfYear day(dayOfYear);
    day.print();

    return 0;
}
```

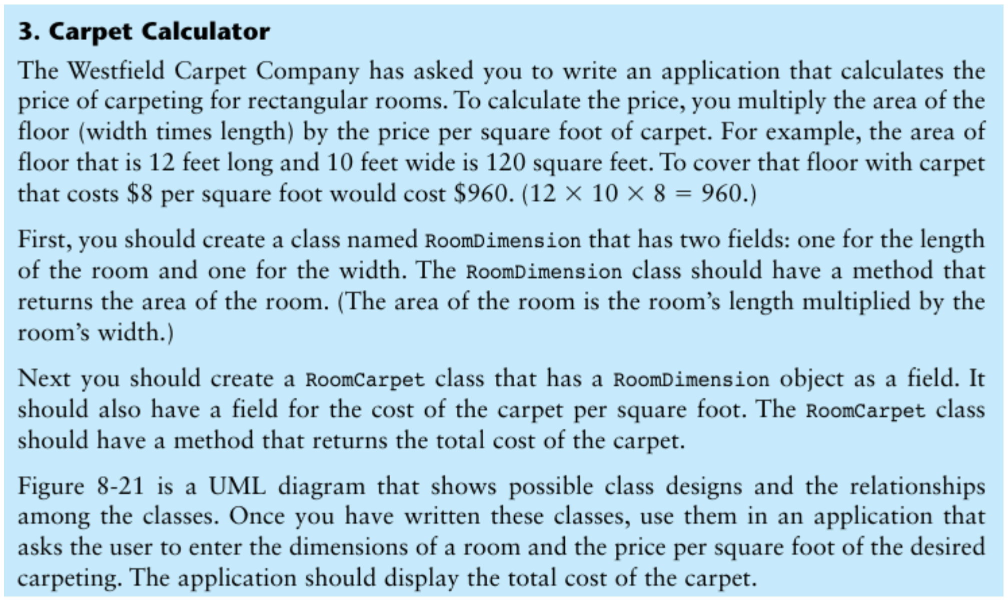
41: DayOfYear.cpp

Assignment 12

Contributor: Ngo Thanh Trung - 1677469

Java

8.3



3. Carpet Calculator

The Westfield Carpet Company has asked you to write an application that calculates the price of carpeting for rectangular rooms. To calculate the price, you multiply the area of the floor (width times length) by the price per square foot of carpet. For example, the area of floor that is 12 feet long and 10 feet wide is 120 square feet. To cover that floor with carpet that costs $8 per square foot would cost $960. ($12 \times 10 \times 8 = 960$.)

First, you should create a class named `RoomDimension` that has two fields: one for the length of the room and one for the width. The `RoomDimension` class should have a method that returns the area of the room. (The area of the room is the room's length multiplied by the room's width.)

Next you should create a `RoomCarpet` class that has a `RoomDimension` object as a field. It should also have a field for the cost of the carpet per square foot. The `RoomCarpet` class should have a method that returns the total cost of the carpet.

Figure 8-21 is a UML diagram that shows possible class designs and the relationships among the classes. Once you have written these classes, use them in an application that asks the user to enter the dimensions of a room and the price per square foot of the desired carpeting. The application should display the total cost of the carpet.

Figure 20: 8.3 Task Requirement

```
package carpetCalculator.Code;
import java.util.Scanner;

public class CarpetCalculator {
```

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

    System.out.print("Enter the length of the room (m): ");
    double length = sc.nextDouble();
    System.out.print("Enter the width of the room (m): ");
    double width = sc.nextDouble();

    RoomDimension myRoom = new RoomDimension(length, width);
    System.out.println("Room entered area (m^2): " + myRoom.getRoomArea());

    System.out.print("Enter the price per square (   ): ");
    double pricePerSquare = sc.nextDouble();

    RoomCarpet myCarpet = new RoomCarpet(myRoom, pricePerSquare);
    System.out.println("Price for " + myRoom.getRoomArea() + " room area: " +
        myCarpet.getTotalCost() + "    ");
    }
}
```

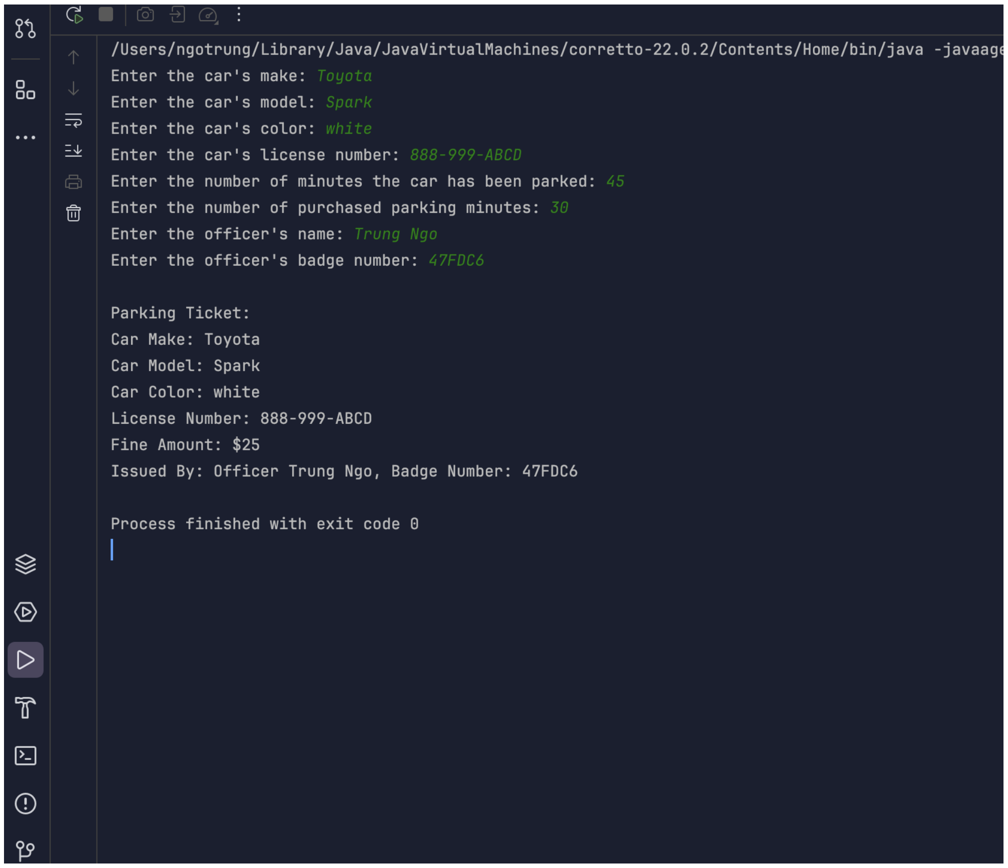
42: CarpetCalculator.java

```java
package carpetCalculator.Code;

public class RoomCarpet {
    private RoomDimension roomDimension;
    private double costPerSqF = 0;

    public RoomCarpet(RoomDimension roomDimension, double costPerSqF) {
        this.roomDimension = roomDimension;
        if (costPerSqF > 0) {
            this.costPerSqF = costPerSqF;
        }
    }

    public double getTotalCost() {
        return costPerSqF * roomDimension.getRoomArea();
    }
}
```

43: RoomCarpet.java

```java
package carpetCalculator.Code;

public class RoomDimension {
    private double roomLength = 0;
    private double roomWidth = 0;

    RoomDimension(double roomLength, double roomWidth) {
        if (roomLength > 0) {
            this.roomLength = roomLength;
        }
        if (roomWidth > 0) {
            this.roomWidth = roomWidth;
        }
    }

    public double getRoomArea() {
        return roomLength * roomWidth;
    }
}
```

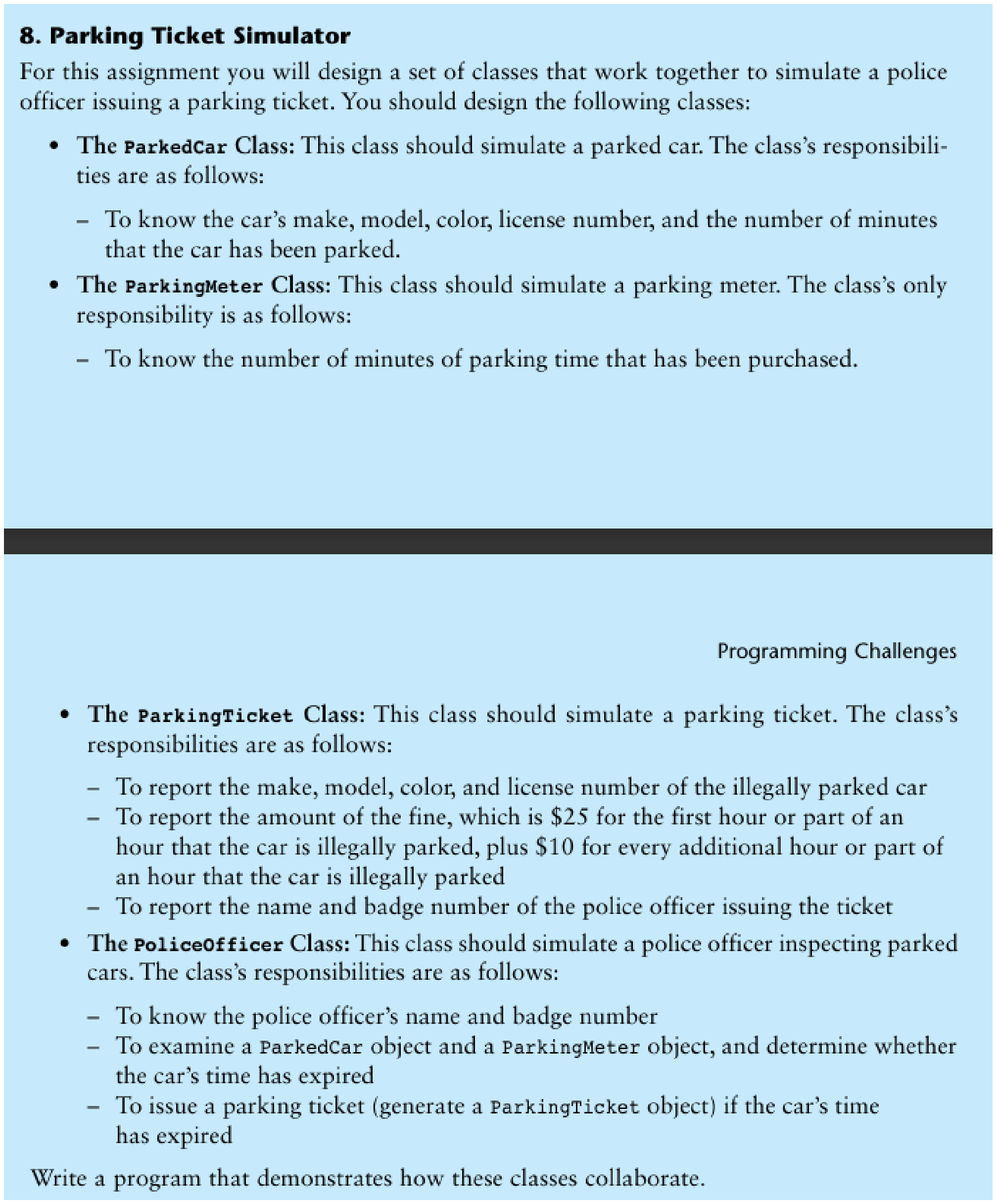
44: RoomDimension.java



```
/Users/ngotrung/Library/Java/JavaVirtualMachines/corretto-22.0.2/Contents/Home/bin/java -javaage
Enter the car's make: Toyota
Enter the car's model: Spark
Enter the car's color: white
Enter the car's license number: 888-999-ABCD
Enter the number of minutes the car has been parked: 45
Enter the number of purchased parking minutes: 30
Enter the officer's name: Trung Ngo
Enter the officer's badge number: 47FDC6

Parking Ticket:
Car Make: Toyota
Car Model: Spark
Car Color: white
License Number: 888-999-ABCD
Fine Amount: $25
Issued By: Officer Trung Ngo, Badge Number: 47FDC6

Process finished with exit code 0
```

Figure 21: 8.3 Result

8.8



8. Parking Ticket Simulator

For this assignment you will design a set of classes that work together to simulate a police officer issuing a parking ticket. You should design the following classes:

- **The `ParkedCar` Class:** This class should simulate a parked car. The class's responsibilities are as follows:
  - To know the car's make, model, color, license number, and the number of minutes that the car has been parked.
- **The `ParkingMeter` Class:** This class should simulate a parking meter. The class's only responsibility is as follows:
  - To know the number of minutes of parking time that has been purchased.

Programming Challenges

- **The `ParkingTicket` Class:** This class should simulate a parking ticket. The class's responsibilities are as follows:
  - To report the make, model, color, and license number of the illegally parked car
  - To report the amount of the fine, which is $25 for the first hour or part of an hour that the car is illegally parked, plus $10 for every additional hour or part of an hour that the car is illegally parked
  - To report the name and badge number of the police officer issuing the ticket
- **The `PoliceOfficer` Class:** This class should simulate a police officer inspecting parked cars. The class's responsibilities are as follows:
  - To know the police officer's name and badge number
  - To examine a `ParkedCar` object and a `ParkingMeter` object, and determine whether the car's time has expired
  - To issue a parking ticket (generate a `ParkingTicket` object) if the car's time has expired

Write a program that demonstrates how these classes collaborate.

Figure 22: 8.8 Task Requirement

```java
package Code;

public class ParkedCar {
    private String make;
    private String model;
    private String color;
    private String licenseNumber;
    private int parkedMinutes;

    public ParkedCar(String make, String model, String color, String licenseNumber
        , int parkedMinutes) {
```

```java
        this.make = make;
        this.model = model;
        this.color = color;
        this.licenseNumber = licenseNumber;
        if (parkedMinutes >= 0) {
            this.parkedMinutes = parkedMinutes;
        } else {
            this.parkedMinutes = 0;
        }
    }

    public String getMake() {
        return make;
    }

    public String getModel() {
        return model;
    }

    public String getColor() {
        return color;
    }

    public String getLicenseNumber() {
        return licenseNumber;
    }

    public int getParkedMinutes() {
        return parkedMinutes;
    }
}
```

45: ParkedCar.java

```java
package Code;

public class ParkingMeter {
    private int purchasedMinutes;

    public ParkingMeter(int purchasedMinutes) {
        if (purchasedMinutes >= 0) {
            this.purchasedMinutes = purchasedMinutes;
        } else {
            this.purchasedMinutes = 0;
        }
    }

    public int getPurchasedMinutes() {
        return purchasedMinutes;
    }
}
```

46: ParkingMeter.java

```java
package Code;

public class ParkingTicket {
    private String carMake;
    private String carModel;
    private String carColor;
    private String licenseNumber;
    private String officerName;
    private String officerBadgeNumber;
    private int fine;

    public ParkingTicket(ParkedCar car, String officerName, String
        officerBadgeNumber, int overParkedMinutes) {
        this.carMake = car.getMake();
        this.carModel = car.getModel();
        this.carColor = car.getColor();
        this.licenseNumber = car.getLicenseNumber();
        this.officerName = officerName;
        this.officerBadgeNumber = officerBadgeNumber;

        calculateFine(overParkedMinutes);
    }

    private void calculateFine(int overParkedMinutes) {
        int hours = (int) Math.ceil(overParkedMinutes / 60.0);
        fine = 25 + (hours - 1) * 10;
    }

    public String getTicketDetails() {
        return  "\n" +
                "Parking Ticket:\n" +
                "Car Make: " + carMake + "\n" +
                "Car Model: " + carModel + "\n" +
                "Car Color: " + carColor + "\n" +
                "License Number: " + licenseNumber + "\n" +
                "Fine Amount: $" + fine + "\n" +
                "Issued By: Officer " + officerName + ", Badge Number: " +
                    officerBadgeNumber;
    }
}
```

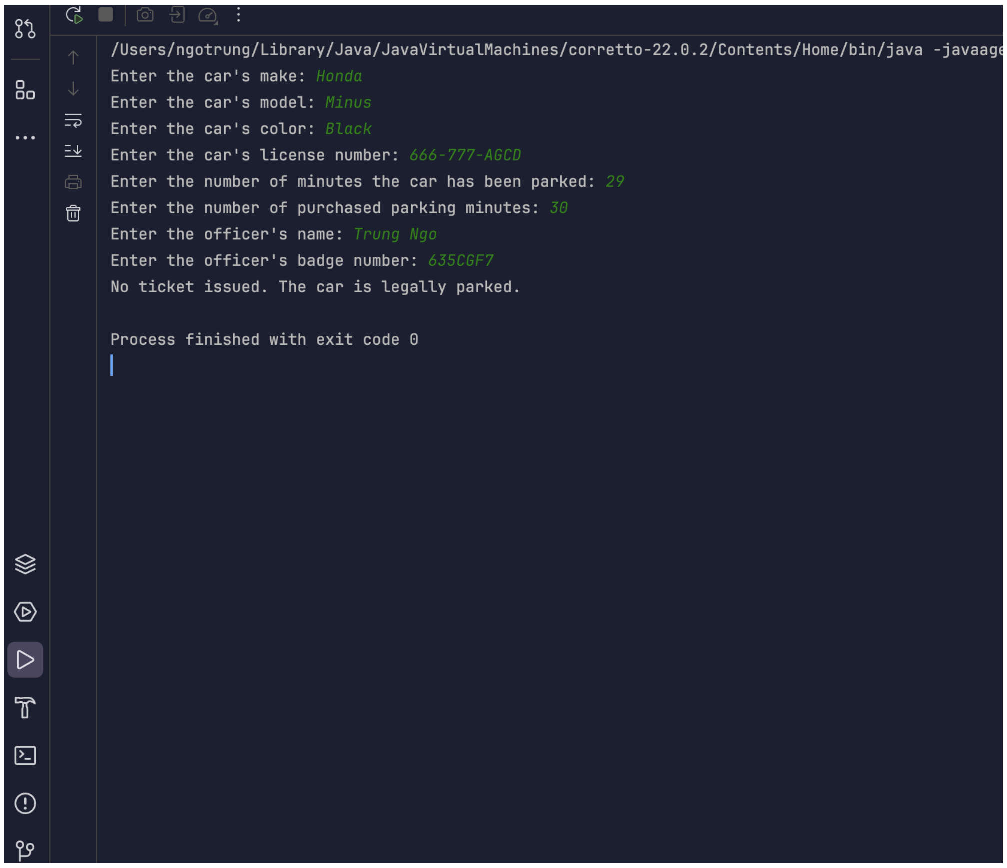
47: ParkingTicket.java

```java
package Code;

import java.util.Scanner;

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

        System.out.print("Enter the car's make: ");
        String make = sc.nextLine();
        System.out.print("Enter the car's model: ");
        String model = sc.nextLine();
        System.out.print("Enter the car's color: ");
        String color = sc.nextLine();
        System.out.print("Enter the car's license number: ");
        String licenseNumber = sc.nextLine();
        System.out.print("Enter the number of minutes the car has been parked: ");
        int parkedMinutes = sc.nextInt();

        ParkedCar car = new ParkedCar(make, model, color, licenseNumber,
            parkedMinutes);
        System.out.print("Enter the number of purchased parking minutes: ");
        int purchasedMinutes = sc.nextInt();
        ParkingMeter meter = new ParkingMeter(purchasedMinutes);

        System.out.print("Enter the officer's name: ");
        sc.nextLine();
        String officerName = sc.nextLine();
        System.out.print("Enter the officer's badge number: ");
        String badgeNumber = sc.nextLine();
        PoliceOfficer officer = new PoliceOfficer(officerName, badgeNumber);

        ParkingTicket ticket = officer.inspectCar(car, meter);
        if (ticket != null) {
            System.out.println(ticket.getTicketDetails());
        } else {
            System.out.println("No ticket issued. The car is legally parked.");
        }

        sc.close();
    }
}
```

48: ParkingTicketSimulator.java

```java
package Code;

public class PoliceOfficer {
    private String name;
    private String badgeNumber;

    public PoliceOfficer(String name, String badgeNumber) {
        this.name = name;
        this.badgeNumber = badgeNumber;
    }

    public ParkingTicket inspectCar(ParkedCar car, ParkingMeter meter) {
        int overParkedMinutes = car.getParkedMinutes() - meter.getPurchasedMinutes
            ();
        if (overParkedMinutes > 0) {
            return new ParkingTicket(car, name, badgeNumber, overParkedMinutes);
        }
        return null;
    }
}
```

49: PoliceOfficer.java

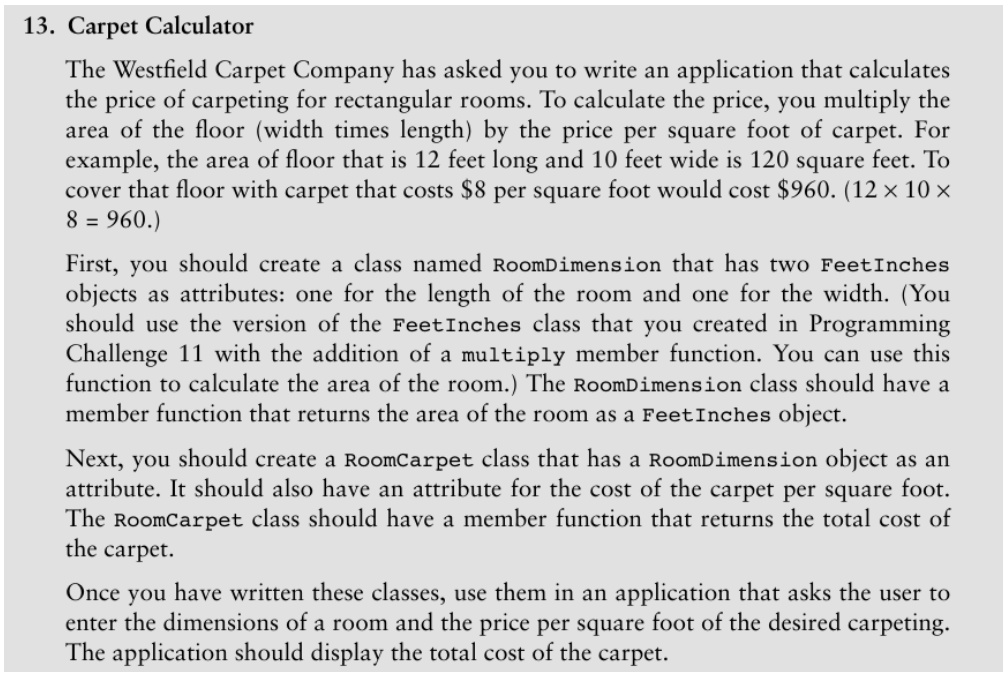


```
/Users/ngotrung/Library/Java/JavaVirtualMachines/corretto-22.0.2/Contents/Home/bin/java -javaage
Enter the car's make: Honda
Enter the car's model: Minus
Enter the car's color: Black
Enter the car's license number: 666-777-AGCD
Enter the number of minutes the car has been parked: 29
Enter the number of purchased parking minutes: 30
Enter the officer's name: Trung Ngo
Enter the officer's badge number: 635CGF7
No ticket issued. The car is legally parked.

Process finished with exit code 0
```

Figure 23: 8.8 Result

C++

14.13



13. Carpet Calculator

The Westfield Carpet Company has asked you to write an application that calculates the price of carpeting for rectangular rooms. To calculate the price, you multiply the area of the floor (width times length) by the price per square foot of carpet. For example, the area of floor that is 12 feet long and 10 feet wide is 120 square feet. To cover that floor with carpet that costs $8 per square foot would cost $960. (12 × 10 × 8 = 960.)

First, you should create a class named `RoomDimension` that has two `FeetInches` objects as attributes: one for the length of the room and one for the width. (You should use the version of the `FeetInches` class that you created in Programming Challenge 11 with the addition of a `multiply` member function. You can use this function to calculate the area of the room.) The `RoomDimension` class should have a member function that returns the area of the room as a `FeetInches` object.

Next, you should create a `RoomCarpet` class that has a `RoomDimension` object as an attribute. It should also have an attribute for the cost of the carpet per square foot. The `RoomCarpet` class should have a member function that returns the total cost of the carpet.

Once you have written these classes, use them in an application that asks the user to enter the dimensions of a room and the price per square foot of the desired carpeting. The application should display the total cost of the carpet.

Figure 24: 14.13 Task Requirement

```cpp
#ifndef FEETINCHES_H
#define FEETINCHES_H

class FeetInches {
private:
    int feet;
    int inches;
    void simplify();

public:
    FeetInches(int f = 0, int i = 0);
    FeetInches multiply(const FeetInches& other);
    double toDouble() const;
    void display() const;
};

#endif
```

50: FeetInches.h

```cpp
#include "FeetInches.h"
#include <iostream>
using namespace std;

void FeetInches::simplify() {
    if (inches >= 12) {
        feet += inches / 12;
        inches %= 12;
    }
}

FeetInches::FeetInches(int f, int i) : feet(f), inches(i) {
    simplify();
}

FeetInches FeetInches::multiply(const FeetInches& other) {
    int total_inches_1 = feet * 12 + inches;
    int total_inches_2 = other.feet * 12 + other.inches;

    int total_square_inches = total_inches_1 * total_inches_2;

    int result_feet = total_square_inches / 144;
    int result_inches = (total_square_inches % 144) / 12;

    return FeetInches(result_feet, result_inches);
}

double FeetInches::toDouble() const {
    return feet + inches / 12.0;
}
void FeetInches::display() const {
    cout << feet << " feet " << inches << " inches";
}
```

51: FeetInches.cpp

```
#ifndef ROOMDIMENSION_H
#define ROOMDIMENSION_H

#include "FeetInches.h"

class RoomDimension {
private:
    FeetInches length;
    FeetInches width;

public:
    RoomDimension(FeetInches l, FeetInches w);
    FeetInches getArea();
    void display();
};

#endif
```

52: RoomDimension.h

```
#include "RoomDimension.h"
#include <iostream>
using namespace std;

RoomDimension::RoomDimension(FeetInches l, FeetInches w) : length(l), width(w) {}

FeetInches RoomDimension::getArea() {
    return length.multiply(width);
}

void RoomDimension::display() {
    cout << "Room Dimensions: ";
    length.display();
    cout << " x ";
    width.display();
    cout << endl;
}
```

53: RoomDimension.cpp

```
#ifndef ROOMCARPET_H
#define ROOMCARPET_H

#include "RoomDimension.h"

class RoomCarpet {
private:
    RoomDimension room;
    double pricePerSquareFoot;

public:
    RoomCarpet(RoomDimension r, double price);
    double getTotalCost();
};

#endif
```

54: RoomCarpet.h

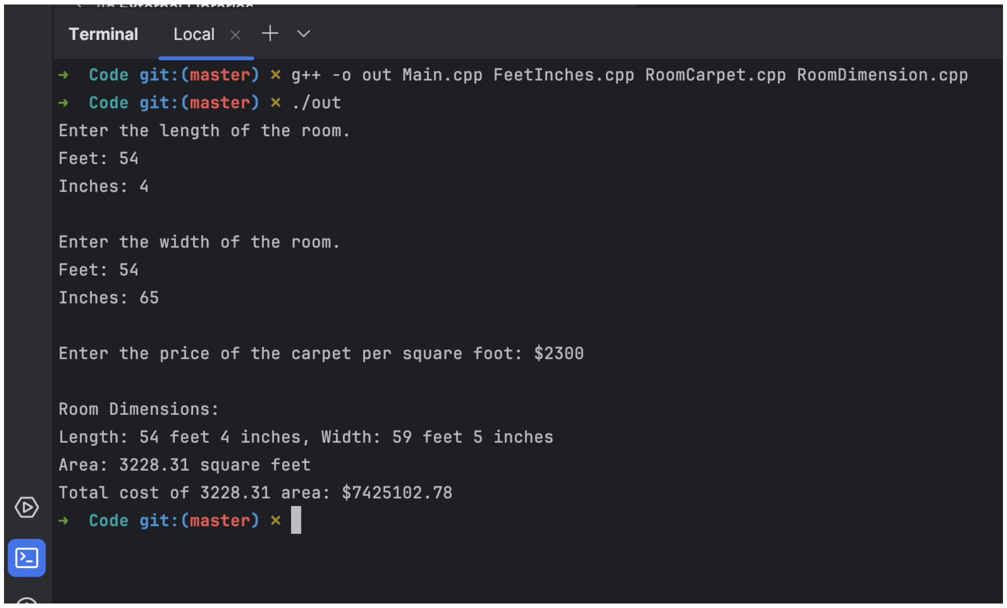
```
#include "RoomCarpet.h"

RoomCarpet::RoomCarpet(RoomDimension r, double price) : room(r),
    pricePerSquareFoot(price) {}

double RoomCarpet::getTotalCost() {
    return room.getArea().toDouble() * pricePerSquareFoot;
}
```

55: RoomCarpet.cpp

```
#include <iostream>
#include "FeetInches.h"
#include "RoomDimension.h"
#include "RoomCarpet.h"

using namespace std;

int main() {
    int lengthFeet, lengthInches, widthFeet, widthInches;
    double price;

    cout << "Enter the length of the room (feet inches): ";
    cin >> lengthFeet >> lengthInches;
    cout << "Enter the width of the room (feet inches): ";
    cin >> widthFeet >> widthInches;
    cout << "Enter the price per square foot: $";
    cin >> price;

    FeetInches length(lengthFeet, lengthInches);
    FeetInches width(widthFeet, widthInches);
```

```cpp
    RoomDimension roomDim(length, width);
    RoomCarpet carpet(roomDim, price);

    roomDim.display();
    cout << "Total cost of carpet: $" << carpet.getTotalCost() << endl;

    return 0;
}
```

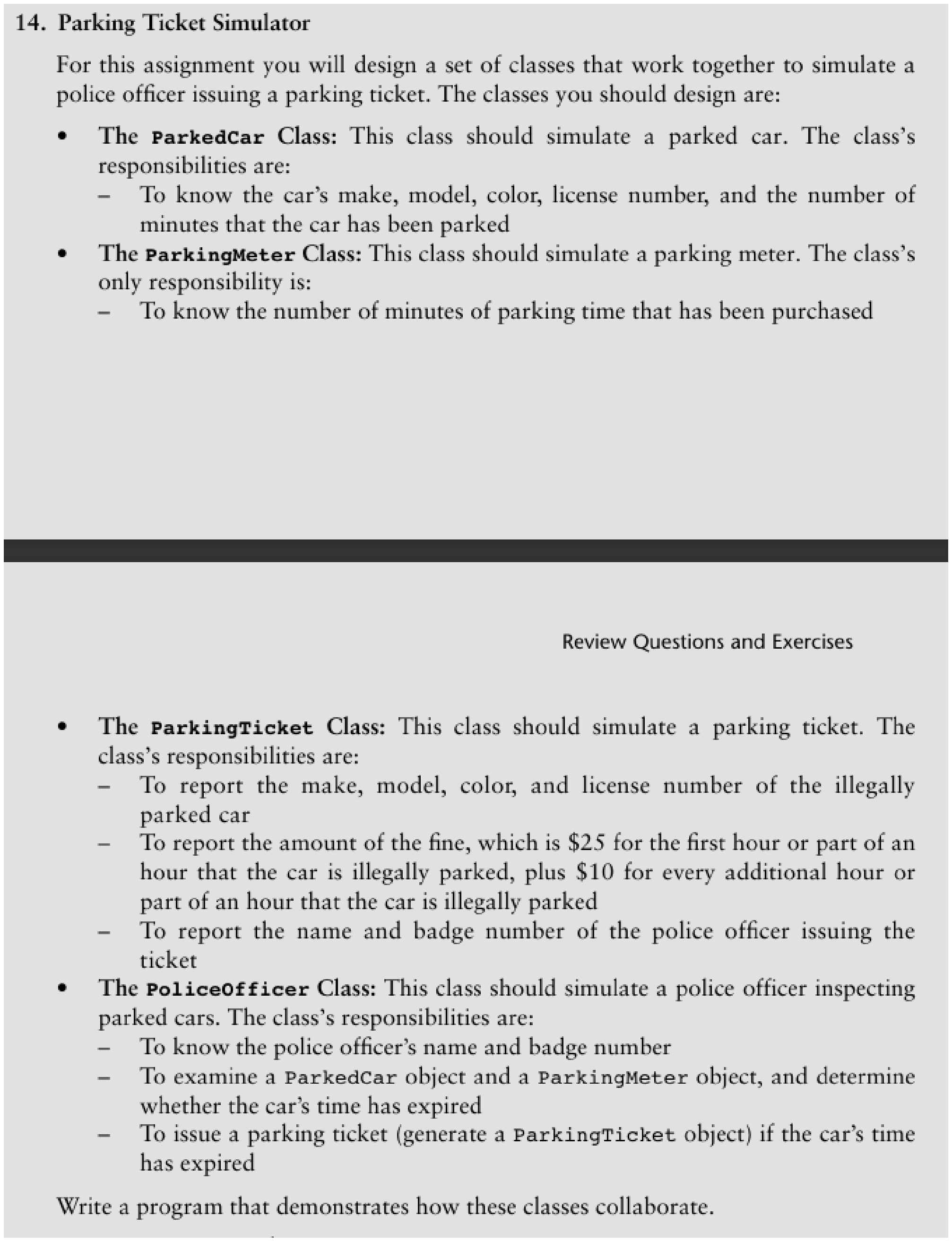
56: main.cpp



```
➜  Code git:(master) ✗ g++ -o out Main.cpp FeetInches.cpp RoomCarpet.cpp RoomDimension.cpp
➜  Code git:(master) ✗ ./out
Enter the length of the room.
Feet: 54
Inches: 4

Enter the width of the room.
Feet: 54
Inches: 65

Enter the price of the carpet per square foot: $2300

Room Dimensions:
Length: 54 feet 4 inches, Width: 59 feet 5 inches
Area: 3228.31 square feet
Total cost of 3228.31 area: $7425102.78
➜  Code git:(master) ✗
```

Figure 25: 14.13 Result

14.14



14. Parking Ticket Simulator

For this assignment you will design a set of classes that work together to simulate a police officer issuing a parking ticket. The classes you should design are:

- **The `ParkedCar` Class:** This class should simulate a parked car. The class's responsibilities are:
  - To know the car's make, model, color, license number, and the number of minutes that the car has been parked
- **The `ParkingMeter` Class:** This class should simulate a parking meter. The class's only responsibility is:
  - To know the number of minutes of parking time that has been purchased

Review Questions and Exercises

- **The `ParkingTicket` Class:** This class should simulate a parking ticket. The class's responsibilities are:
  - To report the make, model, color, and license number of the illegally parked car
  - To report the amount of the fine, which is $25 for the first hour or part of an hour that the car is illegally parked, plus $10 for every additional hour or part of an hour that the car is illegally parked
  - To report the name and badge number of the police officer issuing the ticket
- **The `PoliceOfficer` Class:** This class should simulate a police officer inspecting parked cars. The class's responsibilities are:
  - To know the police officer's name and badge number
  - To examine a `ParkedCar` object and a `ParkingMeter` object, and determine whether the car's time has expired
  - To issue a parking ticket (generate a `ParkingTicket` object) if the car's time has expired

Write a program that demonstrates how these classes collaborate.

Figure 26: 14.14 Task Requirement

```
#ifndef PARKEDCAR_H
#define PARKEDCAR_H

#include <string>
using namespace std;

class ParkedCar {
private:
```

```cpp
    string make;
    string model;
    string color;
    string licenseNumber;
    int parkedMinutes;

public:
    ParkedCar(const string& carMake, const string& carModel, const string&
        carColor, const string& license, int minutesParked);
    string getMake() const;
    string getModel() const;
    string getColor() const;
    string getLicenseNumber() const;
    int getParkedMinutes() const;
};

#endif
```

57: ParkedCar.h

```cpp
#ifndef PARKINGMETER_H
#define PARKINGMETER_H

#include <string>
using namespace std;

class ParkingMeter {
private:
    int purchasedMinutes;

public:
    ParkingMeter(int minutesPurchased);
    int getPurchasedMinutes() const;
};

#endif
```

58: ParkingMeter.h

```cpp
#ifndef PARKINGTICKET_H
#define PARKINGTICKET_H

#include "ParkedCar.h"
#include <string>
using namespace std;

class ParkingTicket {
private:
    string carMake;
    string carModel;
    string carColor;
    string licenseNumber;
    string officerName;
    string officerBadgeNumber;
    int fine;
    void calculateFine(int overParkedMinutes);

public:
    ParkingTicket(const ParkedCar& car, const string& officer, const string& badge
        , int overParkedMinutes);
    void printTicket() const;
};

#endif
```

59: ParkingTicket.h

```
#ifndef POLICEOFFICER_H
#define POLICEOFFICER_H

#include "ParkedCar.h"
#include "ParkingMeter.h"
#include "ParkingTicket.h"
#include <string>
using namespace std;

class PoliceOfficer {
private:
    string name;
    string badgeNumber;

public:
    PoliceOfficer(const string& officerName, const string& badge);
    ParkingTicket* inspectCar(const ParkedCar& car, const ParkingMeter& meter)
        const;
};

#endif
```

60: PoliceOfficer.h

```
#include "ParkingTicket.h"
#include <iostream>
#include <cmath>

ParkingTicket::ParkingTicket(const ParkedCar& car, const string& officer, const
    string& badge, int overParkedMinutes)
    : carMake(car.getMake()), carModel(car.getModel()), carColor(car.getColor()),
      licenseNumber(car.getLicenseNumber()), officerName(officer),
          officerBadgeNumber(badge) {
    calculateFine(overParkedMinutes);
}

void ParkingTicket::calculateFine(int overParkedMinutes) {
    int hours = static_cast<int>(ceil(overParkedMinutes / 60.0)); // rounding up
    fine = 25 + (hours - 1) * 10;
}

void ParkingTicket::printTicket() const {
    std::cout << "Parking Ticket:\n"
              << "Car Make: " << carMake << "\n"
              << "Car Model: " << carModel << "\n"
              << "Car Color: " << carColor << "\n"
              << "License Number: " << licenseNumber << "\n"
              << "Fine Amount: $" << fine << "\n"
              << "Issued By: Officer " << officerName << " (Badge " <<
                  officerBadgeNumber << ")\n";
```

```cpp
}
```

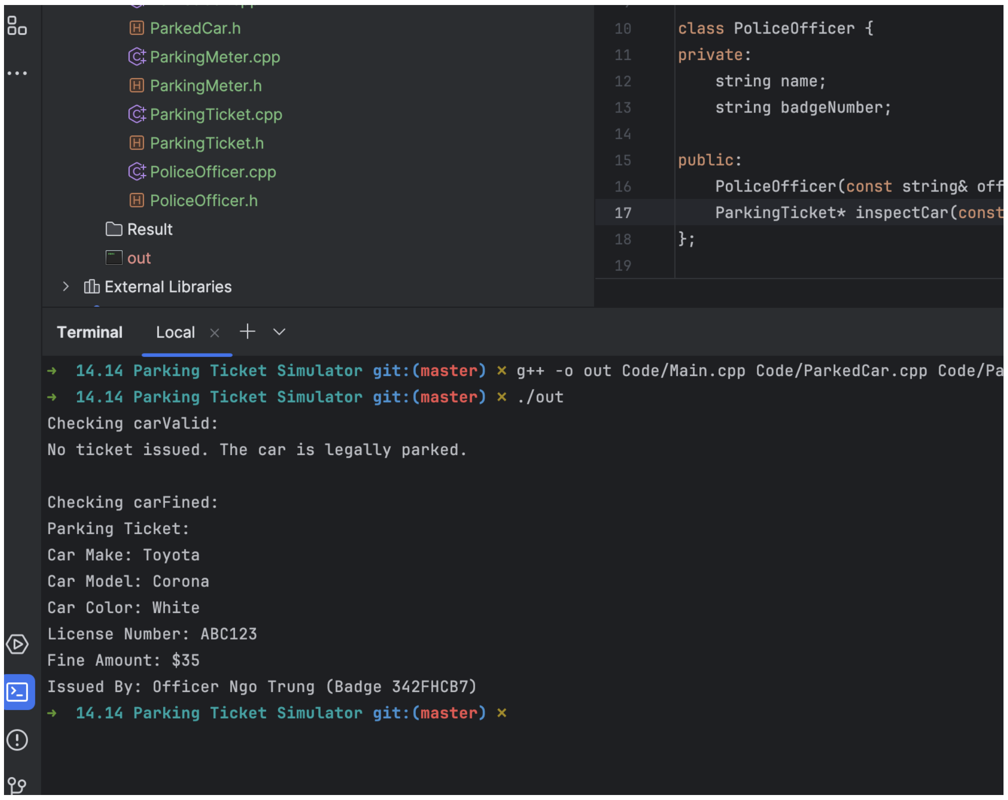
61: ParkingTicket.cpp

```cpp
#include "PoliceOfficer.h"

PoliceOfficer::PoliceOfficer(const string& officerName, const string& badge) :
    name(officerName), badgeNumber(badge) {}

ParkingTicket* PoliceOfficer::inspectCar(const ParkedCar& car, const ParkingMeter&
     meter) const {
    int overParkedMinutes = car.getParkedMinutes() - meter.getPurchasedMinutes();
    if (overParkedMinutes > 0) {
        return new ParkingTicket(car, name, badgeNumber, overParkedMinutes);
    }
    return nullptr;
}
```

62: PoliceOfficer.cpp

```cpp
#include "ParkedCar.h"
#include "ParkingMeter.h"
#include "ParkingTicket.h"
#include "PoliceOfficer.h"
#include <iostream>
using namespace std;

void checkTicket(const ParkedCar& car, const ParkingMeter& meter, const
    PoliceOfficer& officer) {
    ParkingTicket* ticket = officer.inspectCar(car, meter);
    if (ticket != nullptr) {
        ticket->printTicket();
        delete ticket;
    } else {
        cout << "No ticket issued. The car is legally parked.\n";
    }
}

int main() {
    ParkedCar carValid("Toyota", "Corona", "White", "ABC123", 59);
    ParkedCar carFined("Toyota", "Corona", "White", "ABC124", 125);
    ParkingMeter meter(60);
    PoliceOfficer officer("Ngo Trung", "342FHCB7");

    checkTicket(carValid, meter, officer);
    cout << "\n";
    checkTicket(carFined, meter, officer);

    return 0;
}
```

63: Main Program



```cpp
class PoliceOfficer {
private:
    string name;
    string badgeNumber;

public:
    PoliceOfficer(const string& off
    ParkingTicket* inspectCar(const
};
```

```
➜  14.14 Parking Ticket Simulator git:(master) ✗ g++ -o out Code/Main.cpp Code/ParkedCar.cpp Code/Pa
➜  14.14 Parking Ticket Simulator git:(master) ✗ ./out
Checking carValid:
No ticket issued. The car is legally parked.

Checking carFined:
Parking Ticket:
Car Make: Toyota
Car Model: Corona
Car Color: White
License Number: ABC123
Fine Amount: $35
Issued By: Officer Ngo Trung (Badge 342FHCB7)
➜  14.14 Parking Ticket Simulator git:(master) ✗
```

Figure 27: 14.14 Result