Introduction to programming

Homework 4

1. (The Account class) Design a class named Account that contains:

■ A private int data field named id for the account.

■ A private float data field named balance for the account.

■ A private float data field named annualInterestRate that stores the current interest rate.

■ A constructor that creates an account with the specified id (default 0), initial balance (default 100), and annual interest rate (default 0).

■ The accessor and mutator methods for id, balance, and annualInterestRate.

■ A method named getMonthlyInterestRate() that returns the monthly interest rate.

■ A method named getMonthlyInterest() that returns the monthly interest.

■ A method named withdraw that withdraws a specified amount from the account.

■ A method named deposit that deposits a specified amount to the account.

Draw the UML diagram for the class, and then implement the class. (Hint: The method getMonthlyInterest() is to return the monthly interest amount, not the interest rate. Use this formula to calculate the monthly interest: balance \* monthlyInterestRate. monthlyInterestRate is annualInterestRate / 12. Note that annualInterestRate is a percent (like 4.5%). You need to divide it by 100.)

Write a test program that creates an Account object with an account id of 1122, a balance of $20,000, and an annual interest rate of 4.5%. Use the withdraw method to withdraw $2,500, use the deposit method to deposit $3,000, and print the id, balance, monthly interest rate, and monthly interest.

1. (Stopwatch) Design a class named StopWatch. The class contains:

■ The private data fields startTime and endTime with get methods.

■ A constructor that initializes startTime with the current time.

■ A method named start() that resets the startTime to the current time.

■ A method named stop() that sets the endTime to the current time.

■ A method named getElapsedTime() that returns the elapsed time for the stop watch in milliseconds.

Draw the UML diagram for the class, and then implement the class.

Write a test program that measures the execution time of adding numbers from 1 to 1,000,000.

1. (The Time class) Design a class named Time. The class contains:

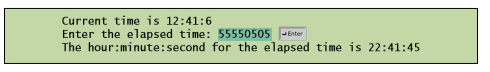
■ The private data fields hour, minute, and second that represent a time.

■ A constructor that constructs a Time object that initializes hour, minute, and second using the current time.

■ The get methods for the data fields hour, minute, and second, respectively.

■ A method named setTime(elapseTime) that sets a new time for the object using the elapsed time in seconds. For example, if the elapsed time is 555550 seconds, the hour is 10, the minute is 19, and the second is 12.

Draw the UML diagram for the class, and then implement the class. Write a test program that creates a Time object and displays its hour, minute, and second. Your program then prompts the user to enter an elapsed time, sets its elapsed time in the Time object, and displays its hour, minute, and second. Here is a sample run:



(Hint: The initializer will extract the hour, minute, and second from the elapsed time. The current elapsed time can be obtained using time.time())

1. (The Rectangle class) Design a class named Rectangle to represent a rectangle. The class contains:

■ Two data fields named width and height.

■ A constructor that creates a rectangle with the specified width and height. The default values are 1 and 2 for the width and height, respectively.

■ A method named getArea() that returns the area of this rectangle.

■ A method named getPerimeter() that returns the perimeter.

Draw the UML diagram for the class, and then implement the class. Write a test program that creates two Rectangle objects—one with width 4 and height 40 and the other with width 3.5 and height 35.7. Display the width, height, area, and perimeter of each rectangle in this order.

1. (The Stock class) Design a class named Stock to represent a company’s stock that contains:

■ A private string data field named symbol for the stock’s symbol.

■ A private string data field named name for the stock’s name.

■ A private float data field named previousClosingPrice that stores the stock price for the previous day.

■ A private float data field named currentPrice that stores the stock price for the current time.

■ A constructor that creates a stock with the specified symbol, name, previous price, and current price.

■ A get method for returning the stock name.

■ A get method for returning the stock symbol.

■ Get and set methods for getting/setting the stock’s previous price.

■ Get and set methods for getting/setting the stock’s current price.

■ A method named getChangePercent() that returns the percentage changed from previousClosingPrice to currentPrice.

Draw the UML diagram for the class, and then implement the class. Write a test program that creates a Stock object with the stock symbol INTC, the name Intel Corporation, the previous closing price of 20.5, and the new current price of 20.35, and display the price-change percentage.

1. (The Fan class) Design a class named Fan to represent a fan. The class contains:

■ Three constants named SLOW, MEDIUM, and FAST with the values 1, 2, and 3 to denote the fan speed.

■ A private int data field named speed that specifies the speed of the fan.

■ A private bool data field named on that specifies whether the fan is on (the default is False).

■ A private float data field named radius that specifies the radius of the fan.

■ A private string data field named color that specifies the color of the fan.

■ The accessor and mutator methods for all four data fields.

■ A constructor that creates a fan with the specified speed (default SLOW), radius (default 5), color (default blue), and on (default False).

Draw the UML diagram for the class and then implement the class. Write a test program that creates two Fan objects. For the first object, assign the maximum speed, radius 10, color yellow, and turn it on. Assign medium speed, radius 5, color blue, and turn it off for the second object. Display each object’s speed, radius, color, and on properties.

1. (Algebra: quadratic equations) Design a class named QuadraticEquation for a quadratic equation The class contains:

■ The private data fields a, b, and c that represent three coefficients.

■ A constructor for the arguments for a, b, and c.

■ Three get methods for a, b, and c.

■ A method named getDiscriminant() that returns the discriminant, which is b^2 - 4ac.

■ The methods named getRoot1()and getRoot2()for returning the two roots of the equation using these formulas:



These methods are useful only if the discriminant is nonnegative. Let these methods return 0 if the discriminant is negative.

Draw the UML diagram for the class, and then implement the class. Write a test program that prompts the user to enter values for a, b, and c and displays the result based on the discriminant. If the discriminant is positive, display the two roots. If the discriminant is 0, display the one root. Otherwise, display “The equation has no roots.”