CS-521 Homework 6

**Instructions**

* Please read the Assignment Directions below.

**Assignment Directions**

Complete the following problems from the textbook: 7.1, 7.3, 7.5, 7.7 (pages 237- 238); 12.1, 12.3 (page 428-429).

**Where to submit?**

Click Assignments in the Navigation Area and then click on the title of the assignment to enter the submission area and upload your response.

7.1

(*The* *Rectangle class*) Following the example of the *Circle* class in [**Section 7.2**](https://jigsaw.vitalsource.com/books/9780133464498/epub/OPS/xhtml/fileP700047855200000000000000000A72D.xhtml#P700047855200000000000000000A741), 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.

7.3

(*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.

\*7.5

(*Geometry: n-sided regular polygon)* An *n*-sided regular polygon’s sides all have the same length and all of its angles have the same degree (i.e., the polygon is both equilateral and equiangular). Design a class named *RegularPolygon* that contains:

* + A private *int* data field named *n* that defines the number of sides in the polygon.
  + A private float data field named *side* that stores the length of the side.
  + A private float data field named *x* that defines the *x*-coordinate of the center of the polygon with default value *0* .
  + A private float data field named *y* that defines the *y*-coordinate of the center of the polygon with default value *0* .
  + A constructor that creates a regular polygon with the specified *n* (default *3* ), side (default *1* ), *x* (default *0* ), and *y* (default *0* ).
  + The accessor and mutator methods for all data fields.
  + The method *getPerimeter()* that returns the perimeter of the polygon.
  + The method *getArea()* that returns the area of the polygon. The formula for computing the area of a regular polygon is **Area** Area=n×s24×tan(πn)Area = n × s24 × tan(πn)

Draw the UML diagram for the class, and then implement the class. Write a test program that creates three *RegularPolygon* objects, created using *RegularPolygon()* , using *RegularPolygon(6, 4)* and *RegularPolygon(10, 4, 5.6, 7.8)* . For each object, display its perimeter and area.

\*7.7

(*Algebra: 2 × 2 linear equations*) Design a class named *LinearEquation* for a 2 × 2 system of linear equations:

ax+by=ecx+dy=fx=ed−bfad−bcy=af−ecad−bcax + by = ecx + dy = fx = ed − bfad − bcy = af − ecad − bc

The class contains:

* The private data fields *a* , *b* , *c* , *d* , *e* , and *f* with get methods.
* A constructor with the arguments for *a* , *b* , *c* , *d* , *e* , and *f* .
* Six *get* methods for *a* , *b* , *c* , *d* , *e* , and *f* .
* A method named *isSolvable()* that returns true if **ad − bc** is not *0* .
* The methods *getX()* and *getY()* that return the solution for the equation.

Draw the UML diagram for the class, and then implement the class. Write a test program that prompts the user to enter *a* , *b* , *c* , *d* , *e* , and *f* and displays the result. If **ad − bc** is *0* , report that “The equation has no solution.” See Exercise 4.3 for sample runs.

12.1

(*The Triangle class*) Design a class named *Triangle* that extends the *GeometricObject* class. The *Triangle* class contains:

* Three float data fields named *side1* , *side2* , and *side3* to denote the three sides of the triangle.
* A constructor that creates a triangle with the specified *side1* , *side2* , and *side3* with default values *1.0* .
* The accessor methods for all three data fields.
* A method named *getArea()* that returns the area of this triangle.
* A method named *getPerimeter()* that returns the perimeter of this triangle.
* A method named *\_\_str\_\_()* that returns a string description for the triangle.

For the formula to compute the area of a triangle, see Exercise 2.14. The *\_\_str\_\_()* method is implemented as follows:

*return*  ***"Triangle: side1 = "***  + str(side1) +  ***" side2 = "***  +

str(side2) +  ***" side3 = "***  + str(side3)

Draw the UML diagrams for the classes *Triangle* and *GeometricObject* . Implement the *Triangle* class. Write a test program that prompts the user to enter the three sides of the triangle, a color, and *1* or *0* to indicate whether the triangle is filled. The program should create a *Triangle* object with these sides and set the color and filled properties using the input. The program should display the triangle’s area, perimeter, color, and *True* or *False* to indicate whether the triangle is filled or not.

\*\*12.3

(*Game: ATM machine*) Use the *Account* class created in Exercise 7.3 to simulate an ATM machine. Create ten accounts in a list with the ids *0* , *1* , ..., *9* , and an initial balance of $100. The system prompts the user to enter an id. If the id is entered incorrectly, ask the user to enter a correct id. Once an id is accepted, the main menu is displayed as shown in the sample run. You can enter a choice of *1* for viewing the current balance, *2* for withdrawing money, *3* for depositing money, and *4* for exiting the main menu. Once you exit, the system will prompt for an id again. So, once the system starts, it won’t stop.

Enter an account id: *4*

Main menu

1: check balance

2: withdraw

3: deposit

4: exit

Enter a choice: *1*

The balance is 100.00

Main menu

1: check balance

2: withdraw

3: deposit

4: exit

Enter a choice: *2*

Enter an amount to withdraw: *3*

Main menu

1: check balance

2: withdraw

3: deposit

4: exit

Enter a choice: *1*

The balance is 97.00

Main menu

1: check balance

2: withdraw

3: deposit

4: exit

Enter a choice: *3*

Enter an amount to deposit: *10*

Main menu

1: check balance

2: withdraw 3: deposit 4: exit Enter a choice: *1*

The balance is 107.00

Main menu

1: check balance

2: withdraw

3: deposit

4: exit

Enter a choice: *4*

Enter an account id: