**Project: The Triangle Class**

Problem Description:

Design a class named Triangle that extends GeometricObject. The class contains:

* Three double data fields named side1, side2, and side3 with default values 1.0 to denote three sides of the triangle.
* A no-arg constructor that creates a default triangle.
* A constructor that creates a triangle with the specified side1, side2, and side3.
* 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 toString() that returns a string description for the triangle.

For the formula to compute the area of a triangle, see Exercise 5.19. The toString() method is implemented as follows:

**return** "Triangle: side1 = " + side1 + " side2 = " + side2 +

" side3 = " + side3;

Design the classes Triangle and GeometricObject. Implement the class. Write a test program that creates a Triangle object with sides 1, 1.5, 1, color yellow and filled true, and displays the area, perimeter, color, and whether filled or not.

Coding:

public class Exercise10\_1 {

public static void main(String[] args) {

Triangle triangle = new Triangle(1, 1.5, 1);

triangle.setColor("yellow");

triangle.setFilled(true);

System.out.println(triangle);

System.out.println("The area is " + triangle.getArea());

System.out.println("The perimeter is "

+ triangle.getPerimeter());

System.out.println(triangle);

}

}

class GeometricObject {

// designed it by yourself

}

class Triangle extends GeometricObject {

// Implement it

}

**Problem statement:**

*Write an application that computes the total charges for the overdue library books. For each library book, the user enters the due date and (optionally) the overdue charge per day,the maximum charge, and the title. If the optional values are not entered, then the preset default values are used. A complete list of book information is displayed when the user finishes entering the input data.The user can enter different return dates to compare the overdue charges.*

Overall Plan

* Tasks:
  1. Get the information for all books
  2. Display the entered book information
  3. Ask for the return date and display the total charge. Repeat this step until the user quits.

Development Steps

* We will develop this program in five steps:

1. Define the basic LibraryBook class.

2. Explore the given BookTracker class and integrate it with the LibraryBook class.

3. Define the top-level OverdueChecker class. Implement the complete input routines.

4. Complete the LibraryBook class by fully implementing the overdue charge computation.

5. Finalize the program by tying up loose ends.

Step 1 Design

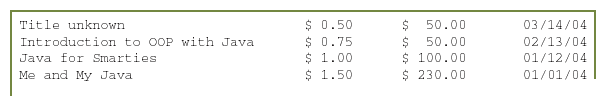
* Develop the basic LibraryBook class.
* The key design task is to identify the data members for storing relevant information.
* We will include multiple constructors for ease of creating LibraryBook objects.
  + Make sure that an instance will be initiated correctly no matter which constructor is used.

Step 1 Code

LibraryBook.java

Step 1 Test

* In the testing phase, we run the test main program and confirm that we get the expected output:



Step 2 Design

* Explore the helper BookTracker class and incorporate it into the program.
* Adjust the LibraryBook class to make it compatible with the BookTracker class.

Step 2 Code

Source Files: LibraryBook.java

Step 2 Test

* In the testing phase, we run the test main program and confirm that we get the expected output.
* We run the program multiple times trying different variations each time.

Step 3 Design

* We implement the top-level control class OverdueChecker.
* The top-level controller manages a single BookTracker object and multiple LibraryBook objects.
* The top-level controller manages the input and output routines
  + If the input and output routines are complex, then we would consider designing separate classes to delegate the I/O tasks.

Step 3 Pseudocode

GregorianCalendar returnDate;

String reply, table;

double totalCharge;

inputBooks(); //read in all book information

table = bookTracker.getList();

System.out.println(table);

//try different return dates

do {

returnDate = *read return date* ;

totalCharge = bookTracker.getCharge(returnDate);

displayTotalCharge(totalCharge);

reply = *prompt the user to continue or not*;

} while ( *reply is yes* );

Step 3 Code

Source Files: OverdueChecker.java

Step 3 Test

* Now we run the program multiple times, trying different input types and values.
* We confirm that all control loops are implemented and working correctly.
  + At this point, the code to compute the overdue charge is still a stub, so we will always get the same overdue charge for the same number of books.
* After we verify that everything is working as expected,we proceed to the next step.

Step 4: Compute the Charge

* To compute the overdue charge, we need two dates: the due date and the date the books are or to be returned.
* The getTimeInMillis method returns the time elasped since the epoch to the date in milliseconds.
* By subtracting this since-the-epoch milliseconds value of the due date from the same of the return date, we can find the difference between the two.
  + If the difference is negative, then it’s not past due, so there’s no charge.
  + If the difference is positive, then we convert the milliseconds to the equivalent number of days and multiply it by the per-day charge to compute the total charge.

Step 4 Code

Source Files: OverdueChecker.java

Step 4 Test

* We run the program multiple times again, possibly using the same set of input data.
* We enter different input variations to try out all possible cases for the computeCharge method.
  + Try cases such as the return date and due date are the same, the return date occurs before the due date, the charge is beyond the maximum, and so forth.
* After we verify the program,we move on to the next step.

Step 5: Finalize / Extend

* Program Review
  + Are all the possible cases handled?
  + Are the input routines easy to use?
  + Will it be better if we allow different formats for entering the date information?
* Possible Extensions
  + Warn the user, say, by popping a warning window or ringing an alarm, when the due date is approaching.
  + Provide a special form window to enter data

(Note: To implement these extensions, we need techniques not covered yet.)