Project 4A: Student Invoice Program

Due: Monday, September 17, 2012 by 11:59 PM

Deliverables

The following project files must be submitted to the "graded" assignment by the due date and time specified above (see the Lab Guidelines for information on submitting project files). You should plan to start on the project not later than Wednesday in lab. To ensure that your classes and methods are named correctly, you should make sure that your file is successfully submitted to the "ungraded" assignment in Web-CAT during lab on Wednesday. **Projects sent via e-mail past the deadline at 11:59 PM will not be accepted without a university-approved excuse.**

Files to submit to Web-CAT:

- StudentInvoiceApp.java
- StudentInvoice.java

Specifications

Overview: You will write a program this week that is composed of two classes, one that defines StudentInvoice objects and the other, StudentInvoiceApp, which inputs data, creates a StudentInvoice object, and then prints invoice.

- **StudentInvoice.java**

  Requirements: Create StudentInvoice class that stores student name, number, tuition and fees (one item), and scholarships. It also includes methods to set and get name and number, and methods to update each of the cost items.

  Design: The StudentInvoice class has fields, a constructor, and methods as outlined below.

  (1) **Fields** (or instance variables): student name and number which are String objects, and tuitionFees and scholarhips which are of type double. These instance variables should be private so that they are not directly accessible from outside of the StudentInvoice class.

  (2) **Constructor**: Your StudentInvoice class must contain a constructor that accepts four parameters representing the student name, number, tuition and fees (one item), and scholarships. Below is an example of how the constructor could be used to create a StudentInvoice object:

```
StudentInvoice student1 = new StudentInvoice(studentName, studentNumber,
                                             tuitionFees, scholarships);
```

  The values for studentName and studentNumber should be trimmed of leading and trailing spaces prior to setting the fields.

(3) **Methods**: **Usually** a class provides methods to access (or read) and modify each of its instance variables (known as get and set methods) along with any other required methods. The methods for StudentInvoice are described below.

- `getName`: Accepts no parameters and returns a String representing the student's name. .
- `setName`: Takes a String parameter and returns a boolean. If the string parameter (the new student's name) is null, then the method returns false and the name is not set. Otherwise, the "trimmed" name is set to the parameter value and the method returns true.
- `getStudentNumber`: Accepts no parameters and returns a String representing the student's number.
- `setStudentNumber`: Takes a String parameter and returns a boolean. If the string parameter (the new student's number) is null, then the method returns false and the student number is not set. Otherwise, the "trimmed" student number is set to the parameter value and the method returns true.
- `getTuitionFees`: Accepts no parameters and returns a double representing the student's tuition and fees.
- `setTuitionFees`: Accepts a double parameter, sets the tutionFee field, and returns nothing.
- `getScholarships`: Accepts no parameters and returns a double representing the student's scholarships.
- `setScholarships`: Accepts a double parameter, sets the scholarships field, and returns nothing.
- `adjustTuitionFees`: Accepts a double parameter, adds it to the tutionFee field, and returns the value of the updated field.
- `adjustScholarships`: Accepts a double parameter, adds it to the scholarships field, and returns the value of the updated field.
- `toString`: Returns a String containing the information in the StudentInvoice object as shown below. Note that you will need to include newline escape sequences and formatting for the dollar values.

```
Name: Pat Smith
ID Number: 012345
Tuition & Fees: $4,300.00
Scholarships: $500.00

You owe: $3,800.00
```

Or, if the scholarship amount exceeds the tuition and fees:

```
Name: Pat Smith
ID Number: 012345
Tuition & Fees: $4,300.00
Scholarships: $5,000.00

Please pick up your check for: $700.00
```

Code and Test: As you implement your StudentInvoice class, you should compile it and then test it using interactions. For example, as soon you have implemented and successfully compiled the constructor, you should create an instance of StudentInvoice in interactions (see the example above). Remember that have an instance on the workbench, you can unfold it to see its values. After you have implemented and compiled one or more methods, create a StudentInvoice object and invoke each of your methods on the object to make sure the methods is working as intended.

- **StudentInvoiceApp.java**

Requirements: Create StudentInvoiceApp class with a main method that reads in student info, creates a StudentInvoice object, and prints the invoice.

Design: The main method should prompt the user to enter the student name, student number, tuition and fees (one item), and scholarships. After the values have been read in, a StudentInvoice object should be created at printed. Below is an example where the student owes money. Your program output should be **exactly** as follows (replace everything in italics with your own words):

Line #	Program output
1	`Prompt the for name: Pat Smith` (*Prompt the for name*)
2	`Prompt the student number last name: 012345` (*Prompt the student number last name*)
3	`Prompt for tuition and fees: 4300` (*Prompt for tuition and fees*)
4	`Prompt for scholarships: 500` (*Prompt for scholarships*)
5	
6	`Name: Pat Smith`
7	`ID Number: 012345`
8	`Tuition & Fees: $4,300.00`
9	`Scholarships: $500.00`
10	
11	`You owe: $3,800.00`
12	

Code: Your program should use the nextLine method of the Scanner class to read user input. Note that this method returns the input as a String. You can use the parseDouble method in the Double wrapper class to convert a String to a double when a double value is desired. For example: `Double.parseDouble(s)` will convert s to a double.

Test: You should test several sets of data to make sure that your program is working correctly. Although your main method does not use all the methods in StudentInvoice, you should ensure that all of your method work according to the specification. You can either user interactions in jGRASP or you can write another class and main method to exercise the methods. Web-CAT will test all methods to determine your project grade.