Project 5: Student Invoice List App

Due: Monday, October 1, 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:

- StudentInvoiceListApp.java
- StudentInvoiceList.java
- StudentInvoice.java

Specifications

Overview: You will write a program this week that is composed of three classes: the first class defines StudentInvoice objects, the second class defines a StudentInvoiceList, and the third, StudentInvoiceApp, which reads in a file name from the user then reads invoice data from the file, creates a StudentInvoice objects and stores them in a ArrayList; creates a StudentInvoiceList object with the ArrayList, and prints the StudentInvoiceList object, and then prints summary information about the StudentInvoiceList object.

- **StudentInvoice.java** (this is an updated version from Project 4A)

  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 adjust 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 class, and they should be set to appropriate default values. These are the only instance variables this class should have.

  (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 in main to create a StudentInvoice object:

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

  The parameter for studentName and studentNumber should be checked for null and not set if null or if equal to an empty String after being trimmed of leading and trailing spaces.

(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 or if it has a length of zero, then the method returns false and the name field is not set. Otherwise, the "trimmed" parameter value is set to the name field 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 or if it has a length of zero, then the method returns false and the student number field is not set. Otherwise, the "trimmed" parameter value is set to the student number field 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 and returns boolean. Sets the tutionFee field and returns true if the parameter is greater than or equal to zero; otherwise returns false without changing the field.
- `getScholarships`: Accepts no parameters and returns a double representing the student's scholarships.
- `setScholarships`: Accepts a double parameter and returns boolean. Sets the scholarships field and returns true if the parameter is greater than or equal to zero; otherwise returns false without changing the field.
- `adjustTuitionFees`: Accepts a double parameter and returns double. Adds the parameter to the tutionFee field if the result would be greater than or equal to zero. Returns the value of the tutionFee field.
- `adjustScholarships`: Accepts a double parameter and returns double. Adds the parameter to the scholarships field if the result would be greater than or equal to zero. Returns the value of the scholarships field.
- `toString`: Returns a String containing the information in the StudentInvoice object as shown below, including 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 when you 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.

- **StudentInvoiceList.java**

Requirements: Create StudentInvoiceList class that stores an ArrayList of StudentInvoice objects. It also includes methods that return the average of all tuition and fees in the list, the percentage of students with scholarships in the list, the average of the scholarship for those who have scholarships, and a toString method.

Design: The StudentInvoiceList class has a field, a constructor, and methods as outlined below.

(1) Field (or instance variable): an ArrayList of StudentInvoice objects. This is the only field (or instance variable) that this class should have, and it should be initialized to a new ArrayList of StudentInvoice objects.

```
... = new ArrayList<StudentInvoice>();
```

(2) Constructor: Your StudentInvoiceList class must contain a constructor that accepts a single parameter of type ArrayList<StudentInvoice> representing the list of StudentInvoice objects. The parameter should be used to assign the field described above.

(3) Methods: The methods for StudentInvoiceList are described below.

- `avgTuitionFees`: Returns a double representing the average of all the tuitionFee fields of the StudentInvoice objects in the list. If there are zero StudentInvoice objects in the list, an average of zero should be returned.
- `percentScholarships`: Returns a double representing the percentage of StudentInvoice objects in the list that have a scholarship greater than zero. .
- `avgScholarships`: Returns a double representing the average of all the scholarship fields of the StudentInvoice objects in the list. If there are zero StudentInvoice objects in the list, an average of zero should be returned.
- `toString`: Returns a String containing the information in each of the StudentInvoice objects in the list as shown below. To create the return result, the toString() method should call the toString() method for each StudentInvoice object in the list. [*The result should **not** include any of the following: average tuition and fees, percentage of students with scholarships, and the average scholarship.*]

Code and Test: Each of the methods above requires that you use a loop to read the objects in the ArrayList. As you implement your StudentInvoiceList class, you should compile it and then test it using interactions.

- **StudentInvoiceListApp.java**

  Requirements: Create a StudentInvoiceApp class with a main method that reads in invoice data from the file, creates StudentInvoice objects, stores them in a ArrayList. creates a StudentInvoiceList object with the ArrayList, and then prints the StudentInvoiceList object followed summary information about the StudentInvoiceList object.

  Design: The main method should prompt the user to enter a file name then it should read invoice data from the file, creating StudentInvoice objects and storing them in a ArrayList. After the file has been read in and the ArrayList created, the main method should create a StudentInvoiceList object with the ArrayList as parameter. It should then print the StudentInvoiceList object followed summary information about the StudentInvoiceList object (i.e., average tuition and fees, percentage of students with scholarships, and the average scholarship among those that have a scholarship).

  Below is output produced after reading in the sample input file (invoice.dat). Your program output should be **exactly** as follows:

Line #	Program output
1	`Enter file name: invoice.dat`
2	
3	`Name: Pat Smith`
4	`ID Number: 0123`
5	`Tuition & Fees: $12,000.00`
6	`Scholarships: $0.00`
7	`You owe: $12,000.00`
8	
9	`Name: Sam Jones`
10	`ID Number: 0124`
11	`Tuition & Fees: $11,000.00`
12	`Scholarships: $2,500.00`
13	`You owe: $8,500.00`
14	
15	`Name: Bee Bonnett`
16	`ID Number: 0125`
17	`Tuition & Fees: $10,000.00`
18	`Scholarships: $12,500.00`
19	`Please pick up your check for: $2,500.00`
20	
21	`Average Tuition and Fees: $11,000.00`
22	`Percentage of Students with Scholarships: 66.67%`
23	`Average Scholarship: $7,500.00`
24	

Code: After your program reads in the file name, it should read in the file using a Scanner object that was created on a file using the file name entered by the user.

```
... = new Scanner(new File(fileName));
```

A while loop should be used to read in the file. You can assume that each set of four lines contains the data from which a StudentInvoice object can be created. As each line is read from the file, local variables for name, number, tuitionFees, and scholarship should be assigned, after which the StudentInvoice object should be created and added to an ArrayList. After the file has been processed, the ArrayList should be used to create a StudentInvoiceList object. The StudentInvoiceList object is should then be printed. Finally, the summary information is printed by calling the appropriate instance methods on the StudentInvoiceList object.

Test: You should test your program minimally by reading in the sample file (invoice.dat), which should produce the output above. Although your program may not use all of the methods in StudentInvoice, you should ensure that all of your methods 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.