Project: Buyer – Part 2

Deliverables

Your project files should be submitted to Web-CAT by the due date and time specified. Note that there is also an optional Skeleton Code assignment which will indicate level of coverage your tests have achieved (there is no late penalty since the skeleton code assignment is ungraded for this project). The files you submit to skeleton code assignment may be incomplete in the sense that method bodies have at least a return statement if applicable or they may be essentially completed files. To avoid a late penalty for the project, you must submit your completed code files to Web-CAT no later than 11:59 PM on the due date for the completed code assignment. If you are unable to submit to Web-CAT directly or via jGRASP, you should e-mail your project Java files in a zip file to your TA before the deadline. Test files are **not** required for this project. If submitted, you will be able to see your code coverage, but this will not be counted as part of your grade.

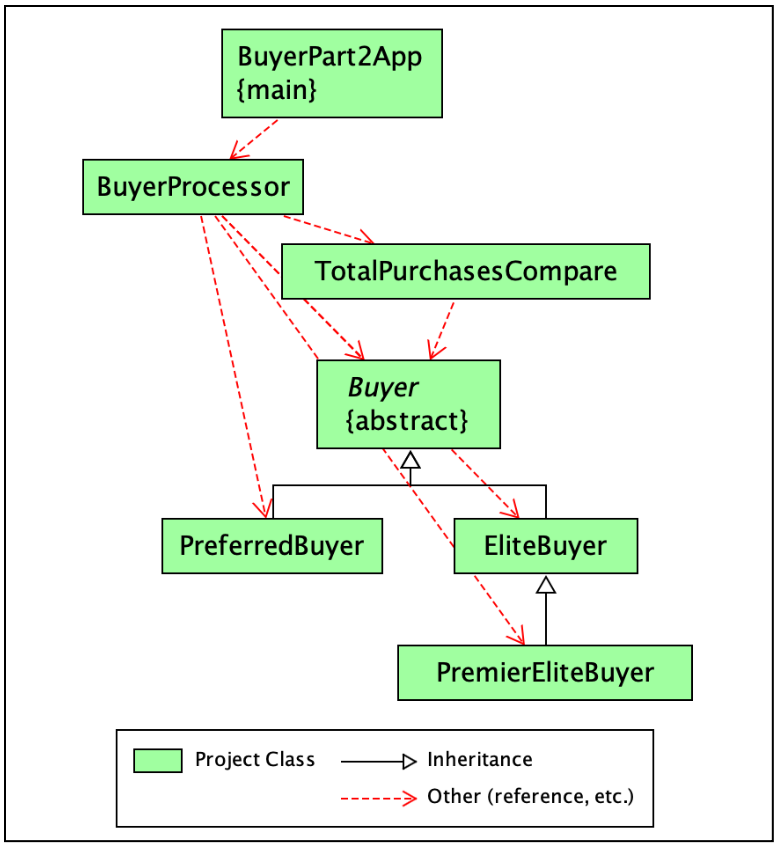
Files to submit to Web-CAT (*test files are optional*):

From Part 1

- Buyer.java
- PreferredBuyer.java, *PreferredBuyerTest.java*
- EliteBuyer.java, *EliteBuyerTest.java*
- PremierEliteBuyer.java, *PremierEliteBuyerTest.java*

New in Part 2

- TotalPurchasesCompare
- BuyerProcessor.java, *BuyerProcessorTest.java*
- BuyerPart2App.java, *BuyerPart2AppTest.java*



Recommendations

You should create new folder for Part 2 and copy your relevant Part 1 source and optional test files to it. You should create a jGRASP project with these files in it, and then add the new source and optional test files as they are created.

Specifications – Use arrays in this project; ArrayLists are not allowed!

Overview: This project is the second of three to process the monthly purchases made by members of the "Buyers" online store. In Part 2, you will modify the Buyer class and you will create two additional classes: (1) BuyerProcessor that includes a method to read in a Buyer data file and methods to generate several reports, (2) TotalPurchasesCompare which implements the Comparator interface for Buyer; and (3) BuyerPart2App which includes the main method for the program. You should create new folder for Part 2 and copy your Part 1source and test files to it. You should create a jGRASP project and add the class and test files as they are created. As you develop and debug your program, you may find it helpful to use the "viewer canvas" feature in conjunction with the debugger or interactions.

- **Buyer, PreferredBuyer, EliteBuyer, PremierEliteBuyer**

  Requirements and Design: In addition to the specifications in Part 1, the Buyer class should implement the Comparable interface for type Buyer and, thus, should add a compareTo method which takes a Buyer object as the parameter. In the `compareTo` method, the comparison is based on the name field; otherwise, there are no changes to these classes.

- **BuyerProcessor.java**

  Requirements: The BuyerProcessor class provides methods for reading in the data file and generating the reports.

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

  (1) **Fields:** An array of Buyer elements and an array of String elements to hold any invalid records read from the data file. [The second array will be used in Part 3.] Note that there are no fields for the number elements in each array. In this project, the size of the array should be the same as the number of elements in the array. These two fields should be private.
  (2) **Constructor:** The constructor has no parameters and initializes the Buyer array and String array in the fields to arrays of length 0.
  (3) **Methods:** Usually a class provides methods to access and modify each of its instance variables (i.e., getters and setters) along with any other required methods. The methods for Buyer are described below.
    - `getBuyerArray` returns an array of type Buyer representing the Buyers array field.
    - `getInvalidRecordsArray` returns an array of type String representing the invalid records array field.
    - `addBuyer` has no return value, accepts a Buyer object, increases the capacity of the Buyer array by one, and adds the Buyer in the last position of the Buyer array.
    - `addInvalidRecord` has no return value, accepts a String, increases the capacity of the invalidRecords array by one, and adds the String in the last position of the invalidRecords array. This method will be used in Part 3, but it still needs to be tested in this project.
    - `readBuyerFile` has no return value, accepts the data file name as a String and throws a FileNotFoundException, which should be included as an import at the beginning of the file. This method creates a Scanner object to read in the file one line at a time. As each line is read, a separate Scanner on the line should be created to read the values in the line. For each line read in, the appropriate Buyer object is created and added to the Buyer array field. The data file has space-delimited text records containing category, account number, last name, and first name, followed by one or more purchases. The category codes are 1 for Preferred Buyer, 2 for Elite Buyer, and 3 for Premier Elite Buyer. Below are examples data records:

```
1 10001 Smith Sam 34.5 100.0 63.50 350.0
2 10002 Jones Pat 34.5 100.0 63.50 300.0
3 10003 King Kelly 34.5 100.0 63.50 300.0
3 10004 Jenkins Jordan 34.5 100.0 63.50 300.0 100.0
```

- `generateReport` processes the Buyer array using the original order from the file to produce the Buyer Report and then returns the report as String. The title lines should be included at the beginning of the report. See example result in output for BuyerPart2App beginning on page 5.
- `generateReportByName` sorts the Buyer array by its natural ordering and processes the Buyer array to produce the Buyer Report (by Name), then returns the report as a String. See example result in output for BuyerPart2App beginning on page 5.
- `generateReportByTotalPurchases` sorts the Buyer array by total purchases and processes the Buyer array to produce the Buyer Report (by Total Purchases) and then returns the report as String. See example result in output for BuyerPart2App beginning on page 5.

Code and Test: See examples of file reading and sorting (using Arrays.sort) in the class notes. The natural sorting order for Buyer objects is determined by the compareTo method when the Comparable interface is implemented.

```
        Arrays.sort(buyerArray);
```

The sorting order based on a Buyer's total purchases is determined by the TotalPurchasesCompare class which implements the Comparator interface (described below).

```
        Arrays.sort(buyerArray, new TotalPurchasesCompare());
```

In your JUnit test methods for the generate reports methods above, you may want to use the following assertion to avoid having to match the return result exactly (where actual result is the result of the method call and expected_result is part of what you think it should contain.

```
        Assert.assertTrue(actual_result.contains(expected_result));
```

- **TotalPurchasesCompare.java**

  Requirements and Design: The TotalPurchasesCompare class implements the Comparator interface for Buyer objects. Hence, it implements the method `compare(Buyer b1, Buyer b2)` that defines the ordering from highest to lowest based on the total purchases of each Buyer. See examples in class notes.

- **BuyerPart2App.java**

  Requirements: The BuyerPart2App class contains the main method for running the program. Note error handling will be added in Part 3.

  Design: The BuyerPart2App class has fields, constructor, and methods as outlined below.
  (1) **Fields:** none
  (2) **Constructor:** none.
  (3) **Method:**

- `main` does the following.
  1. Accepts a file name as a command line argument and throws a FileNotFoundException, which should be included as an import at the beginning of the file (note: if the file name passed in is not in the local folder, a FileNotFoundException will be thrown in the readBuyerFile method and propagated to main, which you catch in the next project (i.e., Buyer – Part 3).
  2. Creates a BuyerProcessor object.
  3. Invokes its readBuyerFile method to read the file and process the buyer records.
  4. Invokes its three generate report methods to print the three reports as shown in the example output beginning on page 5.
  5. If no command line argument is provided, the program should indicate this, and end as shown in the first example output on page 5. An example data file can be downloaded from the assignment page in Canvas.

Code and Test: In your JUnit test file for the BuyerPart2App class, you should have at least two test methods for the main method. One test method should invoke BuyerPart2App.main(args) where args is an empty String array, and the other test method should invoke BuyerPart2App.main(args) where args[0] is the String representing the data file name. Depending on how you implemented the main method, these two methods should cover the code in main. As for the assertion in the test method, since SALES_TAX_RATE is a public class variable in Buyer, you could assert that `Buyer.SALES_TAX_RATE` equals `.08` in each test methods.

In the first test method, you can invoke main with no command line argument as follows:

```
// If you are checking for args.length == 0
// in BuyerPart2App, the following should exercise
// the code for true.
String[] args1 = {};  // an empty String[]
BuyerPart2App.main(args1);
```

In the second test method, you can invoke main as follows with the file name as the first (and only) command line argument:

```
String[] args2 = {"Buyer_data_1.txt"}; // element 0 is the file
name
BuyerPart2App.main(args2);
```

If Web-CAT complains the default constructor for `BuyerPart2App` has not been covered, you should include the following line of code in one of your test methods.

```
// to exercise the default constructor
BuyerPart2App app = new BuyerPart2App();
```

Example Output when no file name is provided as a command line argument

```
----jGRASP exec: java BuyerPart2App
File name expected as command line argument.
Program ending.

 ----jGRASP: operation complete.
```

Example Output when *Buyer_data_1.txt* is the command line argument

```
----jGRASP exec: java BuyerPart2App Buyer_data_1.txt
----------------------------
Buyer Report
----------------------------
Preferred Buyer
AcctNo/Name: 10001 Smith, Sam
Subtotal: $548.00
Tax: $43.84
Total: $591.84
Award Points: 548

Elite Buyer
AcctNo/Name: 10002 Jones, Pat
Subtotal: $473.10
Tax: $37.85
Total: $510.95
Award Points: 4730
(includes 0.05 discount rate applied to Subtotal)

Premier Elite Buyer
AcctNo/Name: 10003 King, Kelly
Subtotal: $423.30
Tax: $33.86
Total: $457.16
Award Points: 8460
(includes 0.15 discount rate applied to Subtotal)

Premier Elite Buyer
AcctNo/Name: 10004 Jenkins, Jordan
Subtotal: $508.30
Tax: $40.66
Total: $548.96
Award Points: 11160
(includes 0.15 discount rate applied to Subtotal)
(includes 1000 bonus points added to Award Points)


--------------------------------------
Buyer Report by Name
--------------------------------------
Premier Elite Buyer
AcctNo/Name: 10004 Jenkins, Jordan
Subtotal: $508.30
Tax: $40.66
```

```
Total: $548.96
Award Points: 11160
(includes 0.15 discount rate applied to Subtotal)
(includes 1000 bonus points added to Award Points)

Elite Buyer
AcctNo/Name: 10002 Jones, Pat
Subtotal: $473.10
Tax: $37.85
Total: $510.95
Award Points: 4730
(includes 0.05 discount rate applied to Subtotal)

Premier Elite Buyer
AcctNo/Name: 10003 King, Kelly
Subtotal: $423.30
Tax: $33.86
Total: $457.16
Award Points: 8460
(includes 0.15 discount rate applied to Subtotal)

Preferred Buyer
AcctNo/Name: 10001 Smith, Sam
Subtotal: $548.00
Tax: $43.84
Total: $591.84
Award Points: 548


---------------------------------------
Buyer Report by Total Purchases
---------------------------------------
Preferred Buyer
AcctNo/Name: 10001 Smith, Sam
Subtotal: $548.00
Tax: $43.84
Total: $591.84
Award Points: 548

Premier Elite Buyer
AcctNo/Name: 10004 Jenkins, Jordan
Subtotal: $508.30
Tax: $40.66
Total: $548.96
Award Points: 11160
(includes 0.15 discount rate applied to Subtotal)
(includes 1000 bonus points added to Award Points)

Elite Buyer
AcctNo/Name: 10002 Jones, Pat
Subtotal: $473.10
Tax: $37.85
Total: $510.95
Award Points: 4730
(includes 0.05 discount rate applied to Subtotal)
```

```
Premier Elite Buyer
AcctNo/Name: 10003 King, Kelly
Subtotal: $423.30
Tax: $33.86
Total: $457.16
Award Points: 8460
(includes 0.15 discount rate applied to Subtotal)



 ----jGRASP: operation complete.
```