Regis University CC&IS
CS310 Data Structures
Programming Assignment 2: ArrayLists

Problem Scenario

The charity was very impressed with your work from last week.
The IT director now needs you to be able to store the data about Donors and Donations in ArrayLists.

Program Requirements

Last week, you created your **Donor** and **Donation** classes. This week you will be implementing ArrayLists containing objects from those classes.

Two ArrayLists will be implemented:

An ArrayList, ***ordered*** by donor ids, containing Donor objects.
An ArrayList, unordered, containing Donation objects.

The program must follow the **CS310 Coding Standards** from Content section 1.9.

Input Data File

This **csv** file input file for this program will be called "assn2input.txt", and will be modified slightly from last week. The file still will contain lines of data about Donors and Donations. But an action field will be added to each line, as the second field, to indicate whether to add or remove the specified donor/donation.

If the action is "ADD", then the remaining data fields will be the same as in assignment 1. But if the action is "DEL", the data line will contain only a donor id, and for donations, a donation id.

The format will be (any of the following data lines, in any order):

```
DONOR,ADD,donorId,firstName,lastName,emailAddress,phoneNumber

DONOR,DEL,donorId

DONATION,ADD,donationId,donorId,description,amount,date,taxDeductible(Y/N),
checkNumber

DONATION,DEL,donationId,donorId
```

Note that a DONOR, ADD line must appear before any lines to add Donations for that Donor, because a Donation can only be added for existing Donors. See the sample input file provided in **Appendix A** below, to see how the data lines will look. You will need to create your own test input data files to test your program.

Implementation Classes

The actual implementation of the Donor and Donation classes will not occur in the **CS310<lastname>.java** file. You will add separate ***implementation*** classes to perform this work.

Your **main** class file will not know what data structure was used. In fact, this **main** class file will be used with future assignments, and will process the same (or a similar) input file format. And the ***implementation*** classes will be modified each week, to reflect whichever data structure is being used that week.

This week you will create the following classes for implementation of the ArrayLists: **DonorLogImpl** and **DonationLogImpl**.

Note that appending "Impl" to the end of a class name is a standard for naming implementation classes. We have also added "Log" for now, to aid in your understanding of what is being implemented.

The **DonorLogImpl** class will be used to create and manage an ArrayList that holds a list of Donor objects.
The class will have one private attribute, an ArrayList of Donor objects.
The class will minimally contain the following methods:

```
public ArrayList<Donor> getDonorList  // return the ArrayList attribute

public void add(Object obj)           // add donor to ordered list

public boolean remove (int donorId)   // remove donor with donorId from list
                                      // and return true if successful

public boolean isIdUnique(int Id)     // test if donor with Id exists in list
```

Since the ArrayList of Donors is an ***ordered*** list, when adding an object to the list, you will need to search for the correct index to add it to (i.e. the index ***after*** the index containing the largest donor id that is smaller than the donor id being inserted).

The **DonationLogImpl** class will be used to create and manage an ArrayList that holds Donation objects.
The class will have one private attribute, an ArrayList of Donation objects.
The class will minimally contain the following methods:

```
public ArrayList<Donation> getDonationList   // return the ArrayList attribute

public boolean add(Object obj)               // add donation to list
                                             // and return true if successful

  // remove donations with donorId from list and return true if any donations deleted
public boolean remove(int donorId)

  // overloaded - remove donation with both donorId and donationId from list
public boolean remove(int donorId, int donationId)

public boolean isIdUnique(int donationId)    // test if donation with id exists in list

public int numberOfDonations()               // return count of all donations
public int numberOfDonations(int donorId)    // overloaded -
                                             // return count of donations with donorId

public float totalDonationAmount()             // return sum of all donations
public float totalDonationAmount(int donorId) // overloaded -
                                              // return sum of donations with donorId
```

Since the ArrayList of Donations is an ***unordered*** list, when adding an object to the list, you can just add to the end of the list.

The program will also have a **PrintImpl** class, which will generate a report (more below).
You can also add any other additional methods your think you need to your implementations.

The main class

When you created your project last week, NetBeans created the initial source file which contained the **main** method (**CS310<lastname>.java**). We will refer to this as the "main class".

All implementations will be instantiated as objects in the main class. For example:

```
public static DonorLogImpl donorLogImpl = new DonorLogImpl();
```

In this example, the `donorLogImpl` object has a single attribute, which is the ArrayList of donors.

In addition to the ***implementation*** instantiations, the **main *class*** will contain the following static methods:

- Method to read the data file and process each data line
  - Try to open the input data file and throw/handle an exception if the file cannot be opened.
  - If the file opens, for each line in the data file:
    - Read the data line.
    - Use **split**() to parse the line to extract the individual data fields into an array.
    - Depending on the String values in the array at:
      - index 0 (DONOR or DONATION)
      - index 1 (ADD or DEL)

      Call a method to do the addition/deletion, passing in the parsed String array as a parameter.

- Method to process a Donor addition to the list
  - Create a new Donor object using the String array parameter values.
  - Validate the email address (contains "@") using the method you wrote for Assn 1.
    - If not valid, display an error message that includes the donor id and email address. (Donors with invalid email addresses will still be added to the Donor ArrayList)
  - Check to see if the donor id is unique using the Donor isIdUnique() method.
  - If the donor id is unique,
    - Call Donor add to add the Donor object to the Donor ArrayList data structure.

    Otherwise,
    - Donors with non-unique donor ids will ***not*** be added to the Donor ArrayList. So display an error message stating the Donor will not be added to the list. Include the use of the **toString**() method to display the object data.

- Method to process a Donation addition to the list
  - Create a new Donation object using the String array parameter values.
  - Insure the check number is valid using the method you wrote for Assn 1.
    - If not valid, display an error message that includes the donation id and check number. (Donations with invalid check numbers will still be added to the Donation ArrayList)
  - Check if the donor id is unique (i.e. when NOT unique, the donor is in the Donor list) using the Donor isIdUnique() method.
  - Check if the donation id is unique using the Donation isIdUnique() method.
  - If the donor id is ***not*** unique and donation id ***is*** unique,
    - Call Donation add to add the Donation object to the Donation ArrayList data structure.

    Otherwise,
    - Donations with unique donor ids or non-unique donation ids will ***not*** be added to the Donation ArrayList. So display an error message stating the Donor will not be added to the list. Include the use of the **toString**() method to display the object data.

- Method to process a Donor deletion
  - Check if the donor id is NOT unique (i.e. the donor **is** in the Donor list) using the Donor isIdUnique() method.
  - If the donor id is in the list:
    - Call Donor remove to find the Donor object with the correct donor id and remove that Donor object from the Donor ArrayList data structure.
    - Call Donation remove to delete all donations with the donor id that has been deleted.
    - Display a message confirming which donor id and its donations were deleted.
  - Otherwise issue an error message that the donor id was not found.

- Method to process a Donation deletion
    - Check if the donation id is NOT unique (i.e. the donation **is** in the Donation list) using the Donation isIdUnique() method.
    - If the donation id is in the list:
        - Call Donation remove to:
            - Find the Donation object with the correct donation id
            - Remove that Donation object from the Donation ArrayList data structure.
        - Display a message confirming which donation id was deleted.
    - Otherwise issue an error message that the donation id was not found.

- Method to create a report

The main class will also (obviously) still contain a **main *method***. This method will:

Call the method to read and process the data file.
Call the method to create the report.
(NOTE: Be sure to delete all the testing code from Assn 1 from the **main** method).

Output Report

After all input has been read and processed, the charity would like you to create a report. You will use a print implementation class, **PrintImpl**, to do this. The **PrintImpl** class will contain a single method that will create the report.

The report will be written to an output file named "assn2report.txt", which will be located in the output folder of the project. Your program should create a constant to hold this filename,
as follows:

```
final String OUTPUT_FILENAME = "output/assn2report.txt";
```

To create the report:

For each donor in the Donor ArrayList, the code will examine each donation in the Donation ArrayList and find donor ids that match the donor id of the Donor object. HINT: Use nested loops.

So for each donor, the report will include the following:

```
DonorId  DonorLastname, DonorFirstname   GOLDSTAR*
         Donations
             DonationId  DonationDate  Description  Amount  CheckNumber  Tax Deductible*

         Number of donations for donor: totalNum
         Total amount of donations for donor:  $ totalValue
```

(*display if true)
A GoldStar donor is one who has donated more than $10,000.00 in cash and property donations.

At the end of the report, the program will also display the total number of donations for ***all*** donors, and the total value of donations for ***all*** donors.

See the sample output in the **Appendix A** below, to see how the formatting will look.

Deliverables

- Create **Javadoc headers**, and generate **Javadoc files** for each of your new ***implementation*** classes and for all new static methods in the main class. You are responsible for completing the comment headers created.
- Your input data file will still be read from the **input** folder in your project.

  Place all test data files that you create to test your program in the **input** folder of your project, and name them as follows:
  `assn2input1.txt`
  `assn2input2.txt`
  (i.e. number each data file after the filename of **`assn2input.txt`**)

  Together, all of your test data files should demonstrate that you have tested every possible execution path within your code, including erroneous data which causes errors or exceptions.

- You will need to add another folder to your project for this assignment, called "**output**".
  The report your program creates will be written to the **output** folder.

- Add screen shots of **clean compile** of your classes to the documentation folder.
  - Remember, if you have compile errors, a red error symbol is placed on the file and folder name tabs. Be sure to clear all compile errors before capturing the screen shot.

  WARNING: Submittals without the clean compile will not be accepted.

Program Submission

This programming assignment is due by midnight of the date listed on the **Course Assignments by Week** page of the Content.

- Export your project from NetBeans (same as Assn 1):
  - Highlight the project name.
  - Click on **File** from the top menu, and select **Export Project**.
  - Select **To ZIP**
  - Name your export file in the following format:
    `CS310<lastname>Assn<x>.zip`

    For example:
    `CS310SmithAssn2.zip`

    NOTE: Save this zip file to some other directory, not your project directory.

- Submit your **.zip** file to the **Prog Assn 2** dropbox (located under **Dropbox** tab in online course).

  Warning: Only NetBeans export files will be accepted.
  Do not use any other kind of archive or zip utility.

Grading

This program will be graded using the **rubric** that is linked under **Student Resources** page.

WARNING:
*Programs submitted more than 5 days past the due date will **not** be accepted,*
and will receive a grade of 0.

Appendix – Sample Input and Output

Sample Input File

```
DONOR,ADD,1,George,Washington,gwashington@mtvernon.com,202-123-4567
DONATION,ADD,101,1,Payroll deduction,22.22,07/04/1776,Y,1001
DONATION,ADD,102,1,Event contribution,200.00,07/05/1776,Y,99
DONATION,ADD,103,1,Contribution,100.00,08/10/1776,N,1002
DONOR,ADD,1,John,Smith,jsmith@gmail.com,111-111-1111
DONOR,ADD,3,Thomas,Jefferson,tjeff#monticello.com,303-333-3333
DONATION,ADD,301,3,Signing donation,10000.00,07/04/1776,Y,2222
DONATION,ADD,302,3,Payroll deduction,22.22,09/01/1810,Y,2223
DONATION,ADD,303,3,Anniversary contribution,111.00,07/04/1777,N,2244
DONATION,DEL,302,3
```

Sample Display Output

```
Reading data from file input/assn2Input.txt
  ERROR: For donation 102, check number 99 is not valid.
  ERROR: Donor ID 1 is not unique.
         Donor{donorId=1, firstName=John, lastName=Smith, phoneNumber=111-111-1111, emailAddress=jsmith@gmail.com}
         will NOT be added to list.
  ERROR: For donor 3, email tjeff#monticello.com is not valid.
  Donation ID 302 has been removed from the donation list
Done reading file. 10 lines read
Creating report...done!
```

Sample Report Output

```
1     Washington, George
      Donations
          101  07/04/1776  Payroll deduction              22.22  1001  Tax deductible
          102  07/05/1776  Event contribution            200.00    99  Tax deductible
          103  08/10/1776  Contribution                  100.00  1002
      Number of donations for donor: 3
      Total amount of donations for donor: $ 322.22

3     Jefferson, Thomas   GOLDSTAR donor
      Donations
          301  07/04/1776  Signing donation            10000.00  2222  Tax deductible
          303  07/04/1777  Anniversary contribution      111.00  2244
      Number of donations for donor: 2
      Total amount of donations for donor: $ 10111.00

Total Number of Donations (all donors) = 5
Total Value of Donations (all donors) = $ 10433.22
```

Appendix B – Partial Sample Code outline

Sample main class

```
public class CS310Lastname {
    static DonorLogImpl donorLogImpl = new DonorLogImpl ();
    static DonationLogImpl donationLogImpl = new DonationLogImpl ();
    static PrintImpl printImpl = new PrintImpl();

    public static void main(String[] args) {
        processFile();
        createReport();
    }

    public static void processDonorAddition (String [] inputLineValues) {
      :
    }

    public static void processDonationAddition (String []inputLineValues) {
      :
    }

    public static void processDonorDeletion(String [] inputLineValues) {
      :
    }

    public static void processDonationDeletion(String [] inputLineValues) {
      :
    }

    private static void processFile() {
       // Try to open file and throw exception if file not found
       // Loop:
       //   Read and parse data line
       //   Call processDonorAddition, processDonationAddition, processDonorDeletion, or processDonationDeletion
    }

    public static void createReport() {
       :
    }

    // Any other static methods that you would like to define
      :

}
```