Regis University CC&IS
CS310 Data Structures
Programming Assignment 1: Create class domains in NetBeans and test them

Problem Scenario

A local charity is in need of a Java programmer to assist them with some of their needs. You have decided to show off your new Java skills, and have graciously volunteered to assist them.
The charity needs to be able to track their donors, as well as the type of donations.

For each **Donor**, the charity needs to know:

- Donor Id (integer identification number - should be a unique value)
- Donor first name (String)
- Donor last name (String)
- Donor phone number (String)
- Donor email address (String)

The charity has had issues with incorrect email addresses, so they are requesting a way to be able to determine if the email address is correct. At this time, the charity just wants to make sure there is a "@" in the email address.

For each **Donation**, the charity wants to track:

- Donation Id (integer identification number - should be a unique value)
- Donor Id of the donor (integer identification number – should match a donor id in the donor list)
- Description of the donation (String – up to length 25)
- Amount of the donation (double)
- Date of the donation (String)
- An indication if the donation is tax deductible (boolean)
- A check number (integer)

Currently the charity only accepts donations by check (no cash). But the charity would like to prevent fraud. So as a fraud detection measure, only check numbers greater than 100, but less than 5000 will be considered valid.

Program Requirements

The assignment this week is to create some of the classes for the problem scenario above. These classes will be used to implement data structures in later assignments.

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

First create a main **class** for this program, named as follows:

`CS310Lastname`

For example, for this program: **`CS310Smith`**

This is the class that will contain a **main** method.
Then create two additional classes:

- a Donor class
- a Donation class

These classes will be used for creating and manipulating the Donor and Donation objects, and will ***not*** contain a **main** method.

Create your classes in NetBeans. Use the tips provided in the NetBeans reader to generate:

constructors
getters/setters
equals()
toString()

methods for the Donor and Donation classes.

Both Donor and Donation classes should include two constructors:

an empty constructor
a constructor that accepts all of the attributes of the class as parameters

While NetBeans will create the basic methods, you will also need to add two additional methods to:

- check the donor email address contains the "@" character
- if a cash donation, check the check number is between 100 and 5000 ***exclusive***

These two methods will simply check the proper attribute and return a **true** or **false**.

Input Data File

You will use a simple **csv** file input file for this program (see the online Content for information on how to read a **csv** file). The file will contain lines of data about Donors and Donations.

This week's input file, **assn1input.txt**, will contain ***only 2 data lines***, in the following format:

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

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

Sample Input File Lines

```
DONOR,1,George,Washington,gwashington@mtvernon.com,202-123-4567
DONATION,101,1,Leadership contribution,100.00,7/4/1776,Y,1001
```

This file will be placed in an **input** directory within your project (see Deliverables section below for how to create new directories within your project). Within your code, define a constant to hold this filename, as follows:

```
final String INPUT_FILENAME = "input/assn1input.txt";
```

Testing

This week, the **main** method in the **`CS310Lastname`** class will be used to test your Donor and Donation classes. The **main** method will run 3 sets of tests.

NOTE: Before each test, display a message stating which test is being run.
Also display a blank line between each of the tests.

Test Set 1

The first set of tests will test the **constructors *with parameters for each attribute*** and the **toString**() method for each class.

Test 1a:

Instantiate a **Donor** object by hardcoding argument values of your choice in the call to the constructor. Then use the **toString**() instance method to print the attribute values to the console.

Test 1b:
Instantiate a **Donation** object by hardcoding argument values of your choice in the call to the constructor. Then use the **toString**() instance method to print the attribute values to the console.

Example:

```
MyClass myObjectName = new MyClass("abc", 123, false);
System.out.println( myObjectName.toString() );
```

Test Set 2
The second set of tests will test the ***empty*** **constructors**, **setters** and **getters**, and will also test reading data from the input data file.

Before running the tests, try to open the input data file.

- Throw an exception of the data file cannot be found.
- The exception handler should display an error message including the name of the file that could not be found.

If the file opens successfully, run the rest of the tests.

Test 2a:
Instantiate a **Donor** object, using the ***empty*** constructor.
Read the ***first*** line of data from the input data file.
Use **split()** to parse the line read into an array of String values.
Call a static **setDonorAttributes**() method. This method will:

- Pass in the Donor object and array of String values as parameters
- Use the setters to set each attribute value
- After setting the attributes, validate the email address
  - If the email address does not contain an '@' character, display an error message
- Return the Donor object with all attributes set (even if the email address is invalid).

Call a static **getDonorAttributes**() method. This method will:
Pass in the Donor object as a parameter.
Use the getters to display each attribute value, one per line.

Test 2b:
Instantiate a **Donation** object, using the ***empty*** constructor.
Read the ***second*** line of data from the input data file.
Use **split()** to parse the line read into an array of String values.
Call a static **setDonationAttributes**() method. This method will:

- Pass in the Donation object and array of String values as parameters
- Use the setters to set each attribute value
- After setting the attributes, validate the check number
  - If the check number is not between 100 and 5000, display an error message.
- Return the Donation object with all attributes set (even if the check number is invalid).

Call a static **getDonationAttributes**() method. This method will:
Pass in the Donation object as a parameter.
Use the getters to display each attribute value, one per line.

Close the input data file when you have completed the tests in Test Set 2.

Test Set 3
The third set of tests will test the **equals**() methods for both classes.

Test 3a:
Use the **equals**() method to compare the two **Donor** objects, and display a message stating whether they are equal or not.

Test 3b:
Use the **equals**() method to compare the two **Donation** objects, and display a message stating whether they are equal or not.

Notes on Testing
In order to validate your code, you will need to run your test code multiple times, each time using different sets of data. For example:

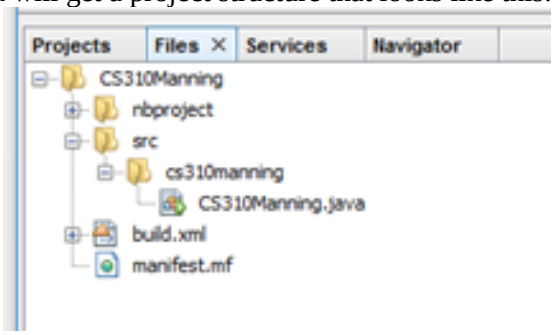
Run 1: No test data input file in input folder.
Run 2: File email is invalid, file check number is invalid, one of the donations is not deductible, and objects are not equal (at least one attribute value differs).
Run 3: All data is valid, one of the donations is deductible, and the objects are equal (all attribute values are the same).

Sample Output Displayed

```
Running Test 1a:
Donor{donorId=2, firstName=John, lastName=Smith, phoneNumber=303-333-3333,
emailAddress=jsmith@gmail.com}

Running Test 1b:
Donation{donationId=202, donorId=2, description=Payroll deduction,
amount=22.22, date=03/01/2016, taxDeductible=true, checkNumber=2222}

Running Test 2a:
ERROR:  Invalide e-mail address is missing @: gwashington#mtvernon.com
1
George
Washington
gwashington#mtvernon.com
202-123-4567

Running Test 2b:
ERROR:  Invalid check number: 99
101
1
Leadership contribution
100.00
7/4/1776
false
99

Running Test 3a:
Donor objects are NOT equal

Running Test 3b:
Donation objects NOT are equal
```

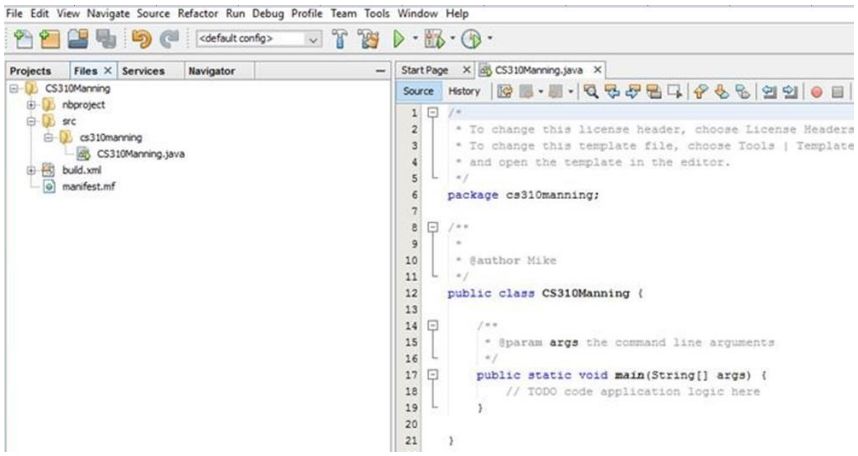
Deliverables

- Create a new project in NetBeans, and name it **CS310 + last name** (e.g. CS310Manning).
    - NetBeans will create the default folders.
      You will get a project structure that looks like this:



    - NetBeans will create the **main** method in the **CS310<lastname>.java** file. This is where you will add your tests.
    - All of the additional class files you create for this assignment should will be placed in the same **CS310<lastname>** folder as the **CS310<lastname>.java** file.

- Use NetBeans to build comment headers. You are responsible for completing the comment headers created. Refer to the NetBeans reader on how to use NetBeans to insert your comment headers.
- Add an **input** folder. To create a new folder in NetBeans:
    - Right click on the project name (top level folder).
    - On the next dialog box, select **New**.
    - Another dialog box will pop up. Look for "Folder" (the order of the list varies based on what the last actions you have done). Select **Folder**.
    - When prompted, name the new folder "**input**".
- Place your input test data file(s) in the **input** folder.
- Use the same method as above to add a **documentation** folder.
- Add screen shots of a clean compile of your classes to the **documentation** folder:



- 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.
- You can paste your image file into the documentation folder, as follows:
  - Right click on the **documentation** folder name, and select Paste.

WARNING: Submittals without the clean compile screenshots 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. To submit your project:

- Export your project from NetBeans.
  To do this:
  - 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:
    `CS310SmithAssn1.zip`

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

- Submit your **.zip** file to the **Prog Assn 1** 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.*