Nassau Community College **Summer I 2016**
CSC 120 CA: Computer Science I **Prof. Christopher R. Merlo**

Lab Assignment #6: Writing the PhoneBookEntry Class
Due Wednesday, 8 June 2016 10:05 am

Please help NCC conserve resources by not printing this document on campus.

Objective

- Practice writing and testing programmer-defined classes

Partners

You may choose your own partner for this assignment. You may also choose to work alone, but this is not recommended.

Overview

The `PhoneBookEntry` class will contain directory information for a single entry in a phone book – first and last names, phone number, date of birth, and an entry representing the person's relationship to you. Member methods will allow the caller to retrieve each piece of data individually, update the last name, return the state of the entry, and compare two entries.

Assignment

1. Create a new project named `Lab6` containing two classes, `PhoneBookEntry` and `Lab6App`. Be sure to place an appropriate JavaDoc comment at the top of each, leaving room for each class' description.

2. Declare the following *instance variables*:

   - `firstName`: To store the person's first name
   - `lastName`: To store the person's last name
   - `phoneNumber`: To store the phone number in the format "`xxx-xxx-xxxx`"
   - `birthDate`: To store the person's date of birth in the format "`mm/dd`"
   - `relationship`: To store `'F'` to indicate a friend, `'M'` for a member of your family, or `'B'` for a business relationship

3. Write a *default constructor* that sets all of the data fields to values that indicate that they haven't been initialized yet: each name to "None", the phone number to "516-555-5555", the birth date to "01/01/2013", and the relationship to 'X'.

4. Write a `toString` method which returns a reference to a `String` containing four lines of information. (Remember to add `\n` to your string to generate a new line.) The first line should contain the person's name in "Last, First" format. The second line should contain one of "Friend", "Family Member", "Business Associate", or "Not Specified". The last two lines should contain the phone number and date of birth, in that order. So, calling `toString` on an object created with the default constructor should generate the following `String`:

   ```
   None, None
   516-555-5555
   01/01/2013
   Not Specified
   ```

5. In the application class, create a `PhoneBookEntry` object, instantiate it, and display its state, after a line that says "`Testing default constructor:`". What output should you expect? Don't move on until you get the expected output.

6. Write a *parameterized constructor* for the `PhoneBookEntry` class, which accepts values for each of the instance variables. Parameter order matters; please follow the order specified in Step 2 above.

   Note: While Java syntax allows you to name the formal parameters with the same identifiers you used to name the instance variables, doing so can very easily lead to great confusion. This is ***not*** recommended.

7. In the application class, declare a second `PhoneBookEntry` object, and instantiate it with the parameterized constructor so that its state (what gets returned from `toString`) is:

   ```
   Smith, Mary
   631-123-4567
   03/15/1990
   Friend
   ```

   Be sure to display a line that says "`Testing parameterized constructor:`", followed by the new object's state.

8. Write an *accessor method* for each of the instance variables. Be sure to name these methods according to Java naming conventions (`get`*`VariableName`*). Write code to test these methods in the application class. The result of testing these methods should be:

```
Testing accessor methods:
First name: Mary
Last name: Smith
Phone number: 631-123-4567
Date of birth: 03/15/1990
Relationship: F
```

9. Write a *mutator method* for the last name field, so that an object's last name can be changed after the object is instantiated. Java naming conventions specify that mutator methods should be named similarly to accessors, except that mutator names start with "set". (How many parameters does this method need? What should the return type be?)

   In the main method, create a new object using the default constructor, set its last name to "Doe", and then display its state. You should generate this output:

```
Testing mutator method for the last name on default object:
Doe, None
516-555-5555
01/01/2013
Not Specified
```

> ***Note:*** (You can read this after class if you're pressed for time, but make sure you read it at some point.) One of the reasons that you need to write and test small amounts of code, and not go on until what you've written is working properly, is because this makes it easier for you to find later bugs. If you've been working on these tasks in order – and I hope you have been – then at this point *you already know* that the `toString` method works. By process of elimination, then, if you don't see "`Doe`" here, then the bug *must* be in the mutator.
>
> However, if you've been skipping around, or avoiding testing your earlier code, incorrect output here may have been caused by any of several previous steps, and you will find that it's frustrating to not know which one.

10. Write a mutator method for the phone number field. Testing it on the object you created in Step #9 should generate this output:

```
Testing mutator method for the phone number on default object:
Doe, None
516-123-4567
01/01/2013
Not Specified
```

11. Write an instance method called `equals` that determines whether the object passed to this method identifies the same person as this object. Return true if the two objects contain the same first name and last name, or false otherwise.

12. In the application, create another `PhoneBookEntry` object with the name Mary Smith, a business associate, born on April 10th, 1985, with the phone number 516-572-7383. Compare this object to the first Mary Smith object, so that this output is displayed:

```
Testing equals method:
Smith, Mary
631-123-4567
03/15/1990
Friend
IS THE SAME PERSON AS
Smith, Mary
516-572-7383
04/10/1985
Business Associate
```

Now create yet another `PhoneBookEntry` object for family member John Smith, 516-572-7700, 1/20/1995. Compare John with the first Mary:

```
Smith, Mary
631-123-4567
03/15/1990
Friend
IS NOT THE SAME PERSON AS
Smith, John
516-572-7700
01/20/1995
Family Member
```

13. Write an instance method called `calculateAge`, which returns how old the person will turn on his or her birthday in the year 2025. Test it with the appropriate objects to generate this output:

```
Testing calculateAge method:
None Doe will be 12 years old in 2025.
Mary Smith will be 35 years old in 2025.
Mary Smith will be 40 years old in 2025.
John Smith will be 30 years old in 2025.
```

14. Now that you know what these classes do, go back to the top and finish writing the comments.

Expected Output

Once you're done, your program should generate this output. If it doesn't, you aren't done. If you need help determining why your output doesn't look like this, ask for help.

```
Testing default constructor:
None, None
516-555-5555
01/01/2013
Not Specified

Testing parameterized constructor:
Smith, Mary
631-123-4567
03/15/1990
Friend

Testing accessor methods:
First name: Mary
Last name: Smith
Phone number: 631-123-4567
Date of birth: 03/15/1990
Relationship: F

Testing mutator method for the last name on default object:
Doe, None
516-555-5555
01/01/2013
Not Specified

Testing mutator method for the phone number on default object:
Doe, None
516-123-4567
01/01/2013
Not Specified

Testing equals method:
Smith, Mary
631-123-4567
03/15/1990
Friend
IS THE SAME PERSON AS
Smith, Mary
516-572-7383
04/10/1985
```

```
Business Associate

Smith, Mary
631-123-4567
03/15/1990
Friend
IS NOT THE SAME PERSON AS
Smith, John
516-572-7700
01/20/1995
Family Member

Testing calculateAge method:
None Doe will be 12 years old in 2025.
Mary Smith will be 35 years old in 2025.
Mary Smith will be 40 years old in 2025.
John Smith will be 30 years old in 2025.
```

Submission

Submit both source code files to the auto grader, and also upload them to my web site. In the text box on my web site, either write the names of your partners, or indicate that you worked alone. Please remember that I won't see your code on my site if you leave this text box blank. Remember that these submissions are due by the start of the next class; late submissions will not be accepted.