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

Lab Assignment #4: Understanding Programmer-Defined Classes
Due Thursday, 2 June 2016 at 10:05 am

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

Objectives

- Understand programmer-defined classes
- Practice testing a programmer-defined class by writing an application class
- Practice reading JavaDoc documentation in order to write methods in a programmer-defined class

Partners

- **Daniel Adrien** and **Tom Marszal**
- **Robim Ambooken** and **Ryan Feraren**
- **Justin Barratt** and **Heather Bo**
- **Christian Burns** and **Janani Thiruvangadam**
- **Jonathon Carrera** and **Kevin Lopez**
- **Derick Hansraj** and **Re'and Ward**
- **Tyler Litz** and **Chuck Okonko Aguolu**
- **Wahab Quazi** and **Sean Rowland**
- **Ali Sharif** and **Liam Turner**
- **Matthew Sixt** and **Emma Zorfass**

Start Eclipse and Create a Project

Call the package `lab4`.

Creating Multiple Classes

Create a new class called `Card`. Be sure to specify `lab4` as the package name. Then create another new class called `Lab4App`. Once again, specify the package name `lab4`.

If you don't properly specify the package name for both files as you create them, bad things will happen.

Please notice that there are now two tabs above the editing pane, one for each source code file you're going to write. You may click on these tabs to switch back and forth between files.

Copy the Classes

I have written some code for the `Card` class, which you will need to import into your project and then edit. There are two different ways to import code into your project; ***choose one*** of these:

- Download `Card.java` from the web page. ***Do not double click it, or open it in Eclipse.*** Right-click the file and choose "Open in Notepad++". Copy the code contained in that file, and paste it in to your `Card.java` file in Eclipse. You may close the version you downloaded.
- Alternately, you can copy the file in the Explorer window, navigate to your workspace (should be something like `j:\workspace\lab4\src\lab4\`), and paste the file into that directory. Clicking on the `Card.java` tab in Eclipse should prompt you to accept the update.

Now do the same thing for the `Lab4App` class.

JavaDoc

Documentation is critical in programming. You will need to read your source code someday. Even more importantly, other people (your professors, your co-workers, your boss, your clients) will need to read your code. Programmers create documentation to simplify this process. We write documentation to explain the code.

The Java language provides a very nice feature called **JavaDoc** which generates *external documentation* (in HTML) based upon carefully-crafted comments. Whereas normal internal multi-line comments are found between the symbols `/*` and `*/`, JavaDoc comments are delineated by `/**` and `*/`.

Within JavaDoc documentation, certain symbols have special meaning. For instance, notice in the documentation at the top of the file that there's a line with `@author` in it.

Whatever information follows that word will be presented in a specially-formatted way within the documentation, so that it's easy to find the author's name.

Also, notice that some of the JavaDoc comments for method definitions include `@param`, after which you document one parameter required by the method, or `@return` where you document the meaning of the method's return value. (Please notice that a method that takes multiple parameters requires multiple `@param` lines.)

JavaDoc comments should only appear at the top of the class definition, and at the top of each method definition. Any other comments should be of the `/* ... */` style or the `// ... <EOL>` style. Also, JavaDoc comments do *not* typically appear above methods in the application class (although the Web-CAT auto-grader will complain about this, so please include them there, too).

You can view the JavaDoc documentation for your class as it would appear on the Web by clicking inside the comment and then clicking the JavaDoc tab at the bottom of the screen. (Try it out now.)

Assignment

Take a look at the `Card` class. Some of the methods are written already, but you will have to write code for the others. Methods that need to return a value have placeholders so that the code doesn't have syntax errors, but these placeholders are not necessarily the values that should be returned. (These are called method stubs, http://en.wikipedia.org/wiki/Method_stub.) Submit your answers to the numbered questions on the web site. (You should write your answers in a text file, and then transfer them all at once, when you know you're logged in to the site.)

Remember, don't write too much code all at once. Each time you add code to `Lab4App`, run it to make sure you get the **Expected Output**. If your output doesn't match the expected output, *fix your code right away*.

1. Run the program the way it is, and make sure you and your partner understand why you get the output you get.
2. Answer Questions #1 and #2 now.

In the `Card` Class:

3. Read the JavaDoc documentation for the `isAFaceCard` method. Since the method is just a stub, it always returns false. Fix the code so that it returns true if the card is a face card, and false otherwise.

In the `Lab4App` Class:

4. Call the `isAFaceCard` method on the object `card1`, and display "`Card 1 is a face card: `", followed by the return value from `isAFaceCard`. (See **Expected Output**.) Display a blank line after this.

5. Instantiate another `Card` object, called `card2`, passing the number 7 as the parameter. Call its `toString` method and display the result following the label "`Card 2: `". Then, as you did with `card1`, call the `isAFaceCard` method on `card2` and display the results. Once again, make sure your program's output matches the **Expected Output**. Display a blank line after these lines.

In the `Card` Class:

6. Read the JavaDoc documentation for the `getPointValue` method. Since the method is just a stub, it always returns 0. Fix the code so that it returns the appropriate point value for every card.

In the `Lab4App` Class:

7. Call the `getPointValue` method on the objects `card1` and `card2`, and display them with the prompts you see in **Expected Output**: "`Card 1's point value is:`". Display a blank line afterward.

In the `Card` Class:

8. Read the JavaDoc documentation for the `isRed` method. Since the method is just a stub, it always returns false. Fix the code so that it returns true if the card is a red card (a heart or a diamond), and false otherwise.

In the `Lab4App` Class:

9. To test the `isRed` method, it will be most useful to display a card's suit along with the return value from the method. Display the string "`Card 1 is `", followed by the card's suit. Retrieve the suit using a method; don't just hardcode the suit name into your code. After this, display "` - color is red: `", followed by the return value from calling `isRed` on `card1`. See the **Expected Output**.

10. One call to a method like `isRed` is not sufficient to convince yourself that you wrote the method correctly. Experienced programmers learn to test all possible conditions before deciding that a program works. So, create three more `Card` objects, using the values 14, 26, and 37, and then display these cards' suits and redness as you did with `card1`. Display a blank line afterward.

11. Call the `toString` method on the three new `Card` objects you just created, and answer Question #3.

In the `Card` Class:

12. Change the `toString` method so that it returns a string with a number in it, if the card is a number card, but a string with "Ace" or "Jack" or "Queen" or "King" as appropriate also. Re-run your program and make sure you're getting the **Expected Output**, for face cards *and* for number cards.

13. Read the JavaDoc documentation for the `equalValue` and `equalSuit` methods. Notice that these method stubs just return false. Fix the code so that they perform the proper comparisons.

In the `Lab4App` Class:

14. Test the `equalValue` method twice each: once on `card1` and `card5`, and once on `card2` and `card3`. Then test `equalSuit` on `card2` and `card3`, and then again on `card4` and `card5`. Make sure not only to format your output according to the `Expected Output`, but also that these methods return the right answers.

15. Answer question 4.

Questions

1. What are the `Card` class' instance variables?

2. What discrepancy do you see between your three lines of output and the first three lines of the expected output? Note: this discrepancy will exist for a while, and that's ok. You'll fix it later.

3. What's weird about the return values from those three calls to `toString`? What do you think should be changed?

4. What happens when you compare two `String` reference variables using the `==` operator?

5. Which step of the instructions did you have to complete to remove the discrepancy you wrote about in Question #2?

Expected Output

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

```
Card 1: Jack of diamonds
Card 1's value is: 11
Card 1's suit is: diamonds
```

```
Card 1 is a face card: true

Card 2: 7 of clubs
Card 2 is a face card: false

Card 1's point value is: 10
Card 2's point value is: 7

Card 1 is diamonds - color is red: true
Card 3 is diamonds - color is red: true
Card 4 is hearts - color is red: true
Card 5 is hearts - color is red: true

Ace of diamonds
King of hearts
Jack of hearts

Jack of diamonds and Jack of hearts have the same value.
7 of clubs and Ace of diamonds do not have the same value.
7 of clubs and Ace of diamonds do not have the same suit.
King of hearts and Jack of hearts have the same suit.
```

Submission

When you think you're ready to submit, call me over to have a quick look, just to make sure you've done everything you need to have done.

Web-CAT

Please notice that ***there are two current assignments*** on Web-CAT. Each one is set up to accept one of the two files you've created. Please upload `Card.java` and `Lab4App.java` to the appropriate assignments.

My Website

Please upload `Card.java` and `Lab4App.java` to the website.

Questions

Please enter the answers to the numbered questions in the text box on the web site.

Please note that these files must be uploaded, and these questions answered, before the start of our next class, and that each student must upload separately. Late submissions will not be accepted.