CS603 Programming Assignment 1

Due by 4:30 PM on Tuesday, 9/12

1. Getting Started

The reading assignment in the syllabus should always be completed before coming to class. It is a good idea to review the assigned reading again before beginning on the programming assignment. When reading the textbook, use the review questions to test your understanding of the material. Answers to those questions are available at http://www.cs.armstrong.edu/liang/intro9e/. You should also review the handouts and the code examples posted to Blackboard.

Be sure to carefully read through the programming assignment before beginning it. Also make sure you understand and adhere to the **academic honesty policy** regarding homework assignments in this course (included in the Course Description document). Note in particular that **you cannot collaborate** on any assignments, and the **code you submit must be yours alone**. The assignments play an extremely important role, as they are designed to help you master the concepts introduced in the lectures and reinforced in the reading.

Plan on starting on your programming assignments as soon after class as possible. Beginning an assignment a day or two before it is due is never a good idea, as it may take you longer to complete than the time you've allotted, and the late penalties are severe. Many students find that spending a few hours each day is the best way to tackle assignments.

Paying careful attention to details is an essential skill for a good programmer, so be sure your code meets all of the requirements of the assignment before submitting it. Thorough testing is essential for ensuring the proper functionality of your code. Your assignment grade will also include points for style. As with any written document, clear communication is critical. Following programming conventions and including descriptive comments will make it possible for others to understand how your code works and what it accomplishes. People supporting code in production are typically not the same ones who wrote that code. No matter how skilled a coder you may be(come), you are not a good programmer if you are the only one who can make sense of your program!

General requirements for all assignments:

1. Your program must compile without errors to receive credit. Programs that fail to compile are awarded 0 points.
2. The inputs to and outputs from the program should appear in the **exact order** that is shown in the sample interactions included in the assignment description.
3. You can use any of the methods you learn in class for prompting for input and showing output, including methods from the **`Scanner`** class for reading input, the **`println()`** method for displaying output, and input and output methods in the **`JOptionPane`** class. Note that you can also use **print()** in prompting for input, in which case the prompt and the user's entry will appear on the same line; **do NOT** use this method for printing output from your program, however, as the operating system may not handle it correctly.
4. Eclipse may add a **Package statement** to the start of your file if your code is not in the default folder, or "package." Please **comment out this statement** prior to submitting if it appears in your file. While it is needed in the Eclipse environment for running your code if it has been added for you, it will not work in my testing environment.

5. Save your file and **close Eclipse** before submitting via Blackboard. Otherwise, unexpected changes may occur to your code. Please use **Google Chrome**, as other browsers may cause errors in the submission process.

Style requirements:

Your grade on an assignment will include points for style. Be sure to do the following:

- Include introductory comments listing your name and a description of the purpose of the program.
- Provide comments within the code to describe the functionality and to clarify any segments that would benefit from further explanation. (You should also remove any comments added by Eclipse that aren't relevant to your program.)
- Use variable names that reflect their purpose and add comments if more information would be helpful. For example, if a variable has a valid data range, that should be noted in a comment.
- Use uppercase for the first letter in a class name and lowercase for the first letter of a method or variable name.
- Use named constants, declared with the keyword **final** and named with all uppercase letters, as appropriate. Note that only you, the coder, can change the value of a constant; it cannot be overwritten during program execution. In addition, named constants make code easier to follow.
- **Indent code** within curly brackets, methods, and control structures (if statements, loops, etc.). This will improve readability and will aid in debugging, as the layout will match the functionality. In **Eclipse**, select **Source > Correct Indentation** for selected code or **Source > Format** for formatting the entire file. You may need to do a bit of cleanup to remove any unnecessary line breaks resulting from the automated formatting.
- Use whitespace to separate blocks of code.

2. Programming Assignment

This assignment consists of the following tasks:

1. Follow the directions posted to the ***Course Notes and Code > Week 1*** folder on Blackboard for installing **Java** and **Eclipse** to your laptop. Another handout posted there describes how to use Eclipse. Make sure you can create and run a Java project using Eclipse before proceeding.
2. For this assignment, you will be creating, compiling, and running two Java programs, as described below. The first problem draws on the contents of the first lecture in conjunction with Chapters 1 and 2, while the second also makes use of material from our second class and from Chapter 3 on the use of selection (*if…else*) statements.
3. When you are satisfied that your programs meet all requirements, submit each **.java** file electronically by following the appropriate submission link from the Blackboard ***Assignments page***. Note there is a separate link for each problem. While it is always preferable to be able to submit each file only once, you can submit multiple times up until the time the assignment is due if you find you need to make a change. Blackboard can be temperamental with this, and you may receive a warning message even when it works, so be sure to check what has been submitted.

Problem 1: I need a break! (10 points)

Write a program that computes the start and end time of a break that occurs midway through a class as well as the end time for that class. The program must prompt the user to enter the starting hour and starting minute of the class, in **that exact order**, based on a 24-hour clock.

Assume that the length of the class is 2 hours, 20 minutes, and the break is 10 minutes long. Your code must work even if those numbers were to change.

You may assume that no classes start later than 8:00 pm (i.e., 20:00) and all classes end by end-of-day (23:59). Additionally, the user always inputs valid data of type integer that fall within the allowed ranges.

Additional requirement:

- Use **named constants** for the **length of the class** and the **length of the break** (as well as any other values you deem necessary). Those constants must be used in your calculations so that, if they were to change, the only changes required in your program would be to modify the values of those constants. For full credit, use the naming convention for constants described in the Style requirements section of this assignment

  Note that the values given to you for those lengths are currently both even numbers. If one or the other changed to an odd number, then the time before the break may be calculated as one minute shorter or one minute longer than the time after the break. Either would be fine, so no additional logic is needed.

Following is a sample interaction that demonstrates how your program should work. The user input appears in **boldface in the handout** (it will **not** appear that way in Eclipse). Your output must include appropriate spacing between literal text and input and output values.

Sample interaction:

```
This program prints the start and end times for a mid-class break and the end
time of the class.

Enter starting hour: 17
Enter starting minute: 25

Break start hour: 18
Break start minute: 30
Break end hour: 18
Break end minute: 40
Class end hour: 19
Class end minute: 45
```

The program will be awarded 8 points for the correct functionality and 2 points for proper style.

Problem 2: Shipping Costs (10 points)

Write a program that computes shipping costs according to the following table:

Delivery Type	Cost	Discount with coupon on total price
1 - Express	$5.50 flat rate plus 1.20 per pound	7%
2 - Regular	$0.98 per pound	5%

The program must first prompt the user to enter the number of pounds and the number of ounces for the shipment (integer values). Next, the user must be prompted for the type of delivery, with 1 for express or any other number for regular. Lastly, the user is prompted to enter a 1 if they have a coupon or any other number if they do not. **Be sure to prompt in this exact order** or your program will fail in the automated test environment. You may assume that the user always enters valid data. Your program must output the total cost, the amount of the discount given, and the total due after the discount, all as dollar amounts (including a dollar sign).

Additional requirements:

- Use **named constants** for all cost components, including the discount rate, as well as anywhere else they would be appropriate. Then **use those constants**, rather than their numeric values, in any calculations involving them. Once again, follow the naming convention for constants..
- **Round currency values** to **at most 2 decimal places** in your output. You do not need to add an additional 0 to output that has only one digit after the decimal point, such as $15.4.

Following are sample interactions that demonstrate how your program should work, with user input again shown in **boldface**. Be sure to test with a variety of inputs.

Sample interaction 1:

```
This program calculates shipping costs.

Enter the number of pounds: 5
Enter the number of ounces: 12
Enter 1 for express delivery or any other number for regular: 1
Enter 1 if you have a coupon or any other number if you do not: 0

Total cost before discount: $12.4
Amount of discount: $0.0
Final cost: $12.4
```

Sample interaction 2:

```
This program calculates shipping costs.

Enter the number of pounds: 10
Enter the number of ounces: 5
Enter 1 for express delivery or any other number for regular: 0
Enter 1 if you have a coupon or any other number if you do not: 1

Total cost before discount: $10.11
Amount of discount: $0.51
Final cost: $9.6
```

The program will be awarded 8 points for functionality and 2 points for proper style.