CIS 255 • Java Programming
Laboratory Assignment 7: Methods with Return Values

In this laboratory assignment you will work with Java method syntax to calculate the cost of an order of candy bars for a fundraiser.

Background

A local middle school band is selling candy bars to raise funds for a trip. Small candy bars sell for $2.50; medium, for $3.75; and large, for $5.00 each. Your program should use a `void` method to display the menu of candy sizes, and it should use a value-returning method to calculate the order total based on size and quantity.

Step 1: Download the Starter File and Launch TextPad

On Blackboard, click the link for the starter file to download it. Make sure you insert your first initial and last name where indicated (**FLast_Lab07.java**) and that you specify the file location. Once you have downloaded the file, open it in TextPad or a similar Java code editor.

Step 2: Customize Your Code and Add Import Statements

Edit the comments at the top of the code to include your name, the course number, the laboratory assignment number, and a brief description of the program's contents. You may also wish to include the date (perhaps after the laboratory assignment number). Also, modify the class name to match your file name. As you continue your code, you may go ahead and add comments to the statements or groups of statements that you believe deserve additional explanation, or you may wait until after you have written your entire program to add comments.

This program will deal with interactive input via the console, so add the `import` statement that corresponds to input from the keyboard. Also, you will need to decide how to handle formatting the numbers: either using the `printf` method, or using the `DecimalFormat` class (the latter requires an extra `import` statement).

Step 3: Declare Variables as Necessary

Several variables have already been declared for you in the `main` method. Make sure that you understand the type and purpose of each variable. If you are using the `DecimalFormat` class for number formatting, declare an object of that type with the proper formatting `String` argument (require two digits after the decimal point).

Step 4: Write the `void` Method Definition

The menu of choices for the candy bar size could be a substantial amount of code. Often a programmer chooses to put the output statements for a menu in a separate method.

After the closing bracket of the `main` method, but before the closing bracket of the class, leave a blank line; then, write the header for the method to display the menu. This method has no parameters, and it does not return a value.

After the header for this new method, add another set of curly brackets. Inside these curly brackets, add several output statements to display the options available to the user. Choose values to represent each size according to what you plan on having the user type as a response:

```
Thank you for supporting our school!
Please choose a candy bar size:
S = Small
M = Medium
L = Large
```

Since this is a `void` method, and since the `Scanner` object is local to `main`, don't call the input method in this method! The statements that obtain the input are written in `main`.

Step 5: Call the `void` Method

Back in the `main` method, proceed to a new line after the variable declarations. Call the `void` method you just defined by typing its name, an empty set of parentheses, and a semicolon.

After the method call, note the statements used to get input from the user for the candy bar size and the number of candy bars. Make sure that you understand these statements before proceeding.

There is no `nextChar` method defined for `Scanner`. In its place, you must read the value using `nextLine` and then obtain the first character of the `String` using the `charAt(0)` method call.

Step 6: Write the Value-Returning Method Definition

The other method in the program should accept the candy bar size and the number of candy bars requested and should return the cost of the order. After the closing bracket of your new `void` method, but before the closing bracket of your class, write the header for this method.

Don't nest one method definition inside the body of another method definition! Although one method definition may contain a call to another, the definitions must be separate.

This header will be similar to that of the other two methods, but the return type should not be `void`: it should indicate that the value to be returned is a real number (the amount of the order). The method name should indicate that it is calculating the amount of the order. Also, the method header needs two parameters in parentheses: the first should be a variable to receive the value the user entered for the candy bar size (a single character), and the second should be a variable to receive the number of candy bars requested (a whole number).

Remember that parameter variables are separated by commas, not semicolons.

Each parameter variable must be listed with a type and a name. Don't list one type at the very beginning and assume that it applies to all of the variables that follow.

Don't include a separate parameter for each size. The user has entered only one character for size, so the method needs only one parameter to receive this size.

After the header, add a set of curly brackets for the method body. Inside these brackets, declare a variable to store the order amount (a real number). You may also declare constants to store the price of each candy bar size (all listed in the **Background** section on the first page), but this is optional.

Each method has its own set of variables (including the parameters). These variables may have the same name as the variables in other methods, but there is no direct relationship between them.

Don't give a variable the same name as a method in your program. The compiler will be confused as to which of the two should be selected.

Determining the order amount involves a decision structure. If the user ordered small candy bars, multiply the quantity by the price of a small candy bar; if medium, multiply by the price of a medium candy bar; if large, multiply by that price. Assign the result of the multiplication to the local variable for the order amount.

Don't declare a separate order amount variable for each candy bar size. Although there are three sizes, only one order amount is being calculated, and only one value can be returned by the method.

Finally, the method must have a `return` statement for the order amount. This statement will include the keyword `return`, the name of the method's order amount variable, and a semicolon.

If a method's return type is `void`, don't try to return a value. You may include a plain `return` statement without a value, but this is not required. If a method should return a value, make sure the value in the `return` statement matches the return type.

Step 7: Call the Value-Returning Method and Display the Results in `main`

Back in the `main` method, add a call to the value-returning method on a new line after the input statements. Since this method returns a value, you must assign the method call to the variable declared in `main` to store the order amount. The method call should include the method name and the argument names in parentheses (you don't need to include the types).

Don't use the parameter names in the method call. The argument names should be the variables declared inside of `main`. (These may have the same name, but if they don't, use the variable names from `main`, not those from the other method.)

Finally, use your chosen output method to display the order amount in currency format (two digits after the decimal). Display it with appropriate labeling:

```
Your order total comes to $(formatted order amount variable)
Please make your check payable to the school.  Thanks again!
```

Step 8: Compile Your Code and Execute the Byte Code

To determine if there are any syntax errors in your source code, and to generate the byte code once your source code is free of syntax errors, select the item **Compile Java** from the **External Tools** submenu of the **Tools** menu. The results of the build process will appear in the **Tool Output** area below your code. If the Tool Output area indicates that there are syntax errors in your code, you may double-click the line that describes each syntax error to see the line that contains the error (or, in some cases, the line after the error).

Once you receive a message in the Tool Output area that the tool completed successfully, you're ready to test your program. In the **External Tools** submenu of the **Tools** menu, select **Run Java Application**. Enter some test values to verify that the results match what you believe they should be.

To test multiple values, you'll need to run the program multiple times. You could nest all of this logic inside a loop to allow for multiple nickname requests, but this is not required.

Step 9: Add Inline Comments and Submit Your Code

Make sure that your code has **three or more inline comments** near the statements that you believe warrant the additional description (especially those that deal with methods). These can be simple line comments at the end of a line of code that needs explanation, or block comments where each precedes a series of statements described in the comment.

Follow the instructions on the course website to attach and submit your source code.

Make sure you complete the submission process! If you attach your file without submitting the assignment, your file will not be received!