**MoneyApp**

1. Open a new project. Call it MoneyApp. In this project you will have a client file called MoneyApp, and a class file called Money. Reminder: Java is case-sensitive. The name of your class file *must* begin with a capital letter. The name of your client file *should* begin with a capital letter.
2. In the Money file, define a class named Money whose objects represent amounts of US money. The class should have two instance variables of type int for the dollars and cents in the amount of money. Mark the instance variables as private.
3. Write three constructors for the Money class.
   1. Write a no-argument, or default constructor with zero as the instance values.
   2. Write a constructor that takes two arguments of type int.
   3. Write a constructor that takes one argument of type int for the dollar value. The cents value is set to zero.
4. Write a toString() method with appropriate formatting for US money amounts.
5. Test what you’ve written so far. In the client file, construct several objects of type Money. Output the following money values to the console.

$99.99

$5.05

$0.00

$0.50

1. Write accessor methods for the dollars and cents members.
2. Write mutator methods for the dollars and cents members.
3. Write two static methods in the client file called add and minus. The methods should calculate the sum and difference between two Money objects. Each method should take two Money objects as arguments and return a Money object. (Hint: write one method and get it to work. Then write the other.)
4. Test your new static methods in the client code. Make sure your code can handle the following:

$99.99 + $0.01

$100.00 - $0.01

The above should not be your only tests. What other values should you test?

Remember to have more than “number only” output. For example, the output from the above tests should look like this:

$99.99 + $0.01 = $100.00

$100.00 - $0.01 = $99.99

1. Without changing your static methods add and minus, now write class methods using the same method names which perform the same type of calculations. These methods should use a calling object and a single argument. Your methods will still return objects of type Money. Note: You will now have two versions of add and minus. Test these new methods in your client code.