CSC 207 **Assignment 4: Bank Accounts Redux** **Spring 2010**

Assigned: Tuesday 23 February 2010

Due: Tuesday 2 March 2010

Topics: Inheritance, design patterns

Collaboration: This homework assignment is to be completed individually.

Submission: Follow the instructions for submitting programs via P-Web and handing in a printed copy. Be sure to generate unit tests and a statement justifying their sufficiency.

In this assignment, we will revisit our banking program for an environment where things are slightly more complicated. There is a federal tax of 5% that needs to be withheld from interest payments. (That is, 5% of the gain due to interest.) Where should the code for this tax go? Hopefully, you know it wouldn't go in your individual account classes. That would create a great deal of repeated code (especially if we had many more account types). It shouldn't really be in the root `Account` class, either. Why? Well, a tax isn't really a property of an account, so that's not a very accurate object model.

Where should we put it? If it doesn't go in the individual account class or the parent class, the only answer left is: in a new class. How should such a class compute the tax? Well, we could have a method that computes it given the relevant parameters. But why should a caller (i.e., an account) have to know what the relevant parameters are? The answer: it shouldn't! Instead, we should make use of the *visitor* pattern. That is, the account should send a reference to itself to a tax object so that the tax man (er, object) can probe the account for whatever information is needed in calculating the withholdings.

Of course, being bankrupt, the states now want in on the action, too. Our bank has account holders in Iowa, Nebraska, and Wyoming, and each of these has a different method for taxing interest:

Iowa	2% of interest earnings, and any monthly account fees are deductible (i.e. subtracted from the taxable amount before the tax is calculated)
Nebraska	1% of interest earnings
Wyoming	No interest tax

Our bank has plans for growth, so our design must scale to the possibility of 50 states! (Fortunately, for this assignment we only need to worry about three.) Where should the *state* tax calculator reside? How should we determine which formula to use?

We could have one state tax calculator class and use `if` statements to determine which formula to use. This seems problematic. If a state were to change its rules, we'd have to hunt down the formula in a (potentially) huge function and make the change. This could also jeopardize the correctness of all the other states' tax calculations. Instead, we're better off with a larger number of (relatively small) individual state tax calculator classes.

How do we determine which method/class to call? We don't really want a big case statement to clutter our account class(es). Instead, we can use a special class that and returns a polymorphic reference to an appropriate state tax calculator (the *factory* pattern!) object. With this, an account method can deduct the appropriate state tax from an interest payment using just two lines of code: one (presumably in the constructor) that gets a reference to an appropriate state tax calculator object, and another to call the actual tax computation (the *state* pattern!) that uses information from the the account (a la the *visitor* pattern!). This code should be the same for every account, regardless of its state of origin.

Furthermore, we don't really need more than one instance of a state tax calculator for any given state, so these should use the *singleton* pattern. (Since many accounts will share references to the same state tax calculators—you will have a *flyweight* pattern!).

Re-design your banking system from assignment 2 so that it now incorporates the federal and state tax changes described above. Create a short program that generates sample statements showing taxed amounts for several accounts of each type, demonstrating the various aspects of your system (states, fees, etc.).

In summary, you will need:

- A federal tax calculator
- Several state tax calculators (each a singleton)
- A state tax calculator factory
- Account updates that withhold taxes from interest payments appropriately

You may use the solution to Assignment 2 found in `/home/weinman/courses/CSC207/bank/` as your starting point.