

Name: _____

APCS Problem Set 2: Object-Oriented Programming

2 Analyze a Basic Class

5. Draw a UML class diagram¹ for the `BankAccount` class.

5 New and Improved `BankAccount` & `BankAccountDriver` Classes

In the earlier sections, you were stepped through the process of taking a simple class and making it conform to *best practices* in class design. Specifically, you took on the task of *data hiding* or *encapsulation*: Making information inside a class only indirectly available outside. This means that the designer of the `BankAccount` class has tight control over what can and cannot be stored in the field variables, and just how data can be retrieved from `BankAccount` objects.

You are now to update the classes `BankAccount` and `BankAccountDriver` so that they abide by these best practices:

- make all field variables `private`
- include *getter* and *setter* methods (accessors and mutators, respectively)
- update the driver class so it uses the getters and setters when interacting with `BankAccounts`

Your updates need to happen in **two** places: On the paper form and in the files you've been working with so far. The Java source files will be used again in the next problem set.

¹Refer to the UML diagram of the `Person` class in the *Lesson #11* slides at <http://feromax.com/apcs/lessons/>.

6.1.2 SavingsAccount class

1. On the paper form, write the code for the `SavingsAccount` class.

6.2 Creating a new pair of accounts

Now that we have defined just what checking and savings accounts are, we can create instances of each. Update `BankAccountDriver.java` to create the following accounts:

CheckingAccount called chkgAcct1	
balance	3200
ownerSSN	882-61-2486
minBalance	100
monthlyFee	5

SavingsAccount called svgsAcct1	
balance	15980
ownerSSN	882-61-2486
interestRate	4
maxNumWithdrawals	6

On the paper form, write the code for `BankAccountDriver.java`, which creates the accounts in the tables above.