**Lab 4, Part E**

**[March 2017 Programming Exam Question]**

In a company, employees may have multiple bank accounts: zero or more savings accounts and zero or more checking accounts. Each checking account has an account id, a balance, and a monthly fee. Each savings account has an account id, a balance, and an interest rate associated with the particular type of savings account. It is possible to read the current balance in any of these accounts, but it is also possible to determine the balance after interest or monthly fee is applied by calling the computeUpdatedBalance method on the account.

An administrator has access to all employee records and from time to time computes the total balance across all employee-owned accounts; for each account, the balance that is needed in this computation is the *updated balance*. This computation is performed in the static method

computeUpdatedBalanceSum

in the Admin class.

Below is a class diagram showing the classes involved and relationships between them. A sequence diagram for the operation computeUpdatedBalanceSum is also provided. Your task in this problem is to write Java code that implements the classes and relationships shown in the diagram. Shells for the Admin and the Employee classes have been provided in your workspace. Also, a Main class (with a main method) has been provided for you to test your code (in the launch package) – the code in the main method has been commented out; when you are ready to test your code, you can uncomment it.

The method computeUpdatedBalance in CheckingAccount does the following computation to obtain the return value:

balance – monthlyFee.

The method computeUpdatedBalance in SavingsAccount performs the following computation to obtain the return value:

balance + (interestRate \* balance).

Points to notice about the class diagram:

1. CheckingAccount and SavingsAccount are subclasses of Account. Also, Account has several abstract methods which must be implemented in its subclasses.
2. There is a one-way association from Employee to Account. It is important that your code reflects and maintains this association.
3. The diagram has a mix of dependencies and associations; make sure your code distinguishes between these properly.

You may not modify the signatures or qualifiers of the methods contained in the Admin or Employee classes that have been provided. You will need to create all the other classes mentioned in the diagrams.

.



