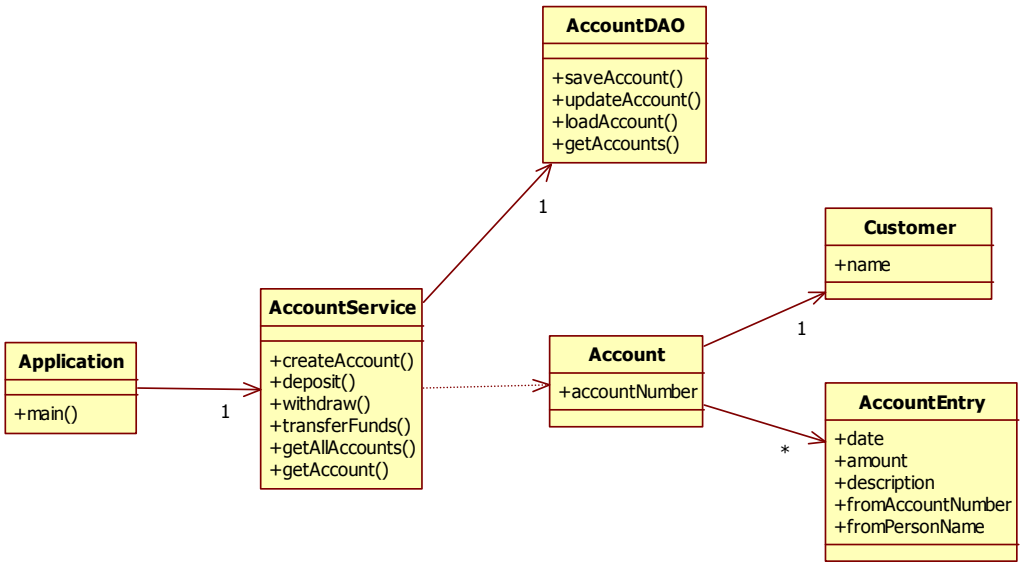
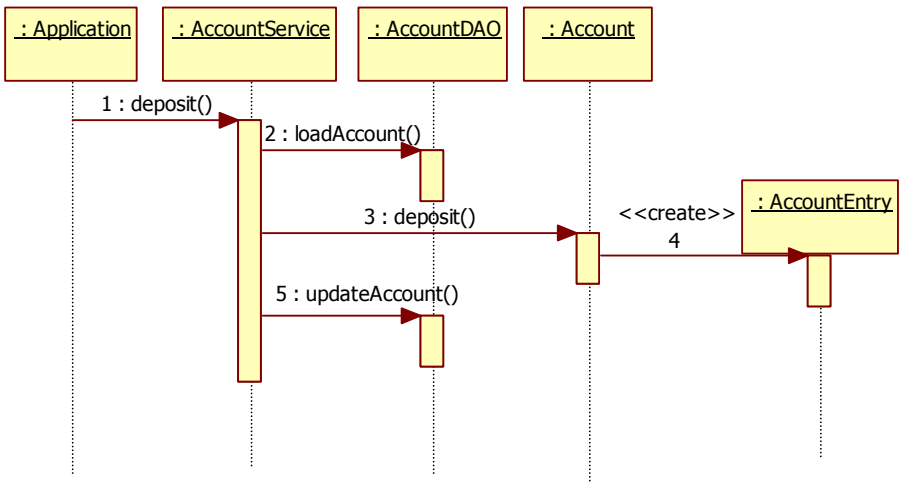
Explanation of the service class.

In the labs for the Observer, Command and Strategy pattern we started with the following simple bank application:





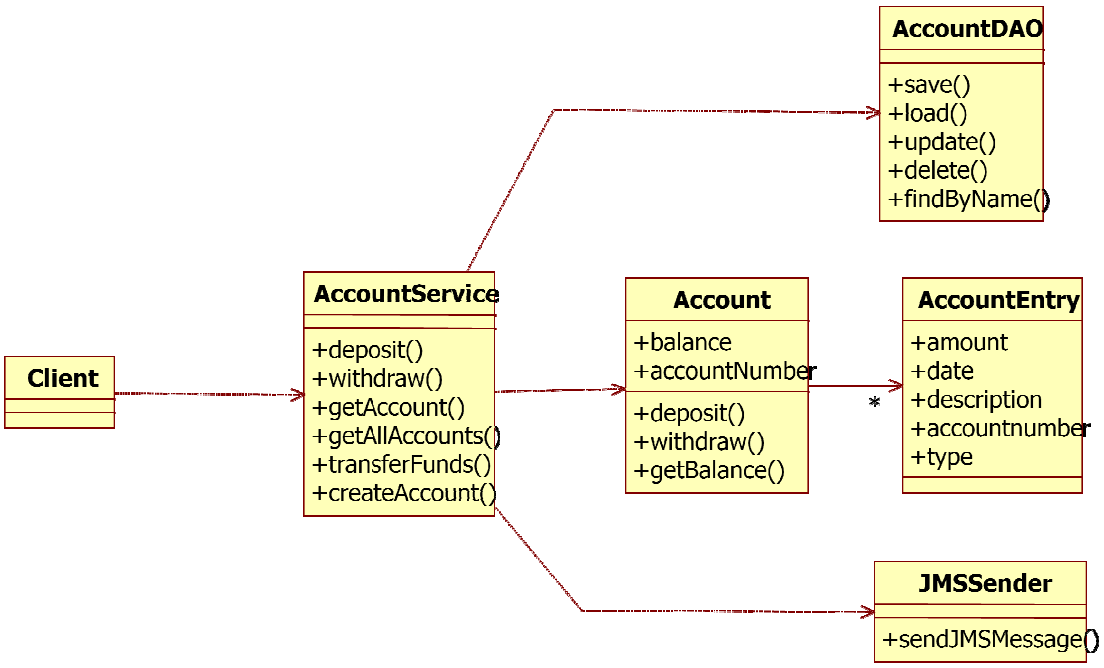
The AccountDOA class contains all necessary data access logic to save, update, delete and retrieve accounts to and from the database.

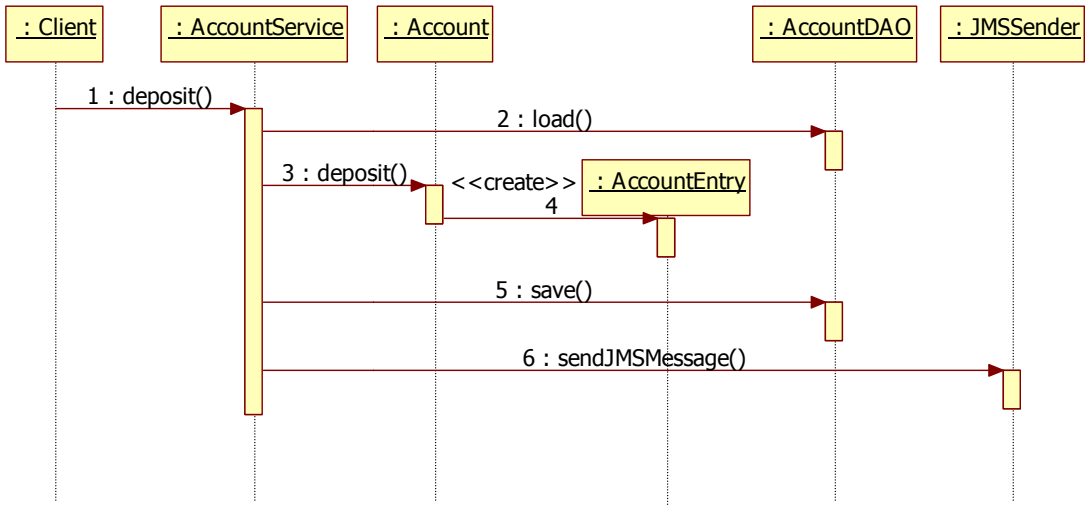
The AccountService class has 2 purposes:

1. The AccountService class is the façade for the account functionality. We can make an encapsulated account component, and the AccountService is the uniform entry class for this component (Façade pattern).

2. The AccountService class connects the business logic (Account, AccountEntry, Customer) with the technical 'plumbing' classes, in this case the AccountDAO class. Notice that none of the business logic classes call the AccountDAO. The business logic classes do not even know that there is a database.

To illustrate this better, see the following modified example:

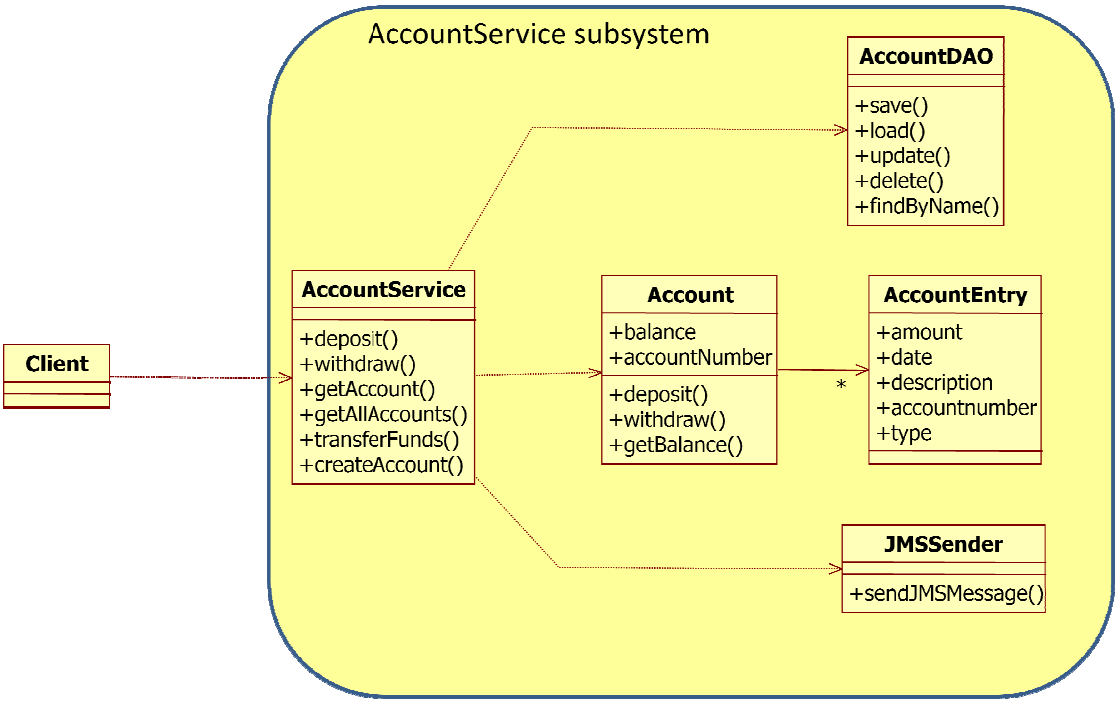




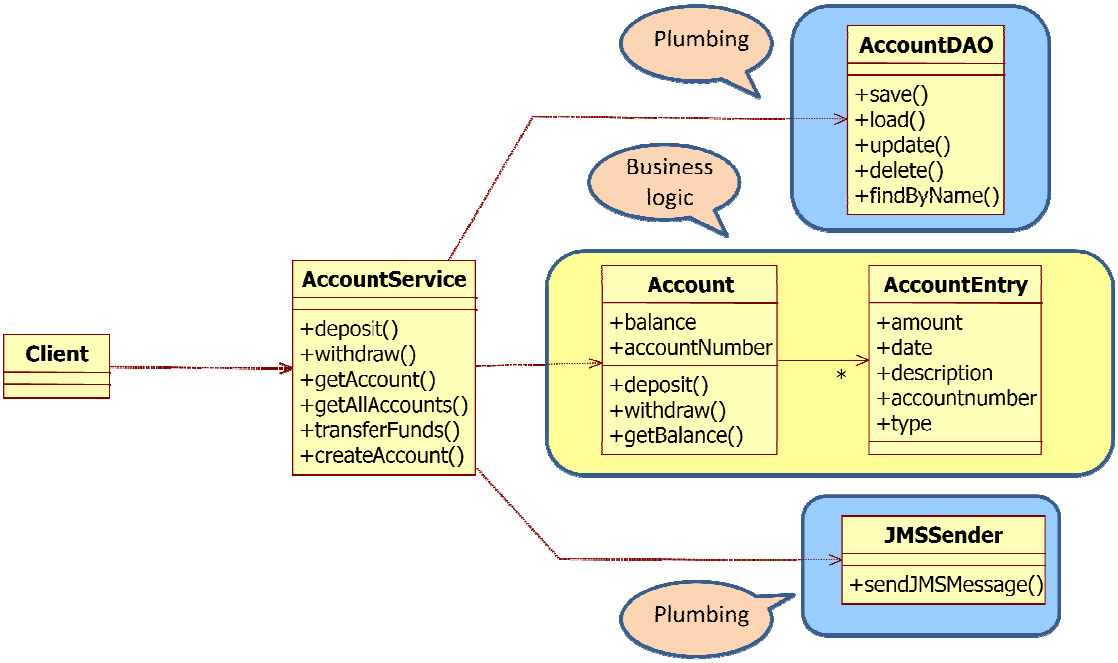
The AccountService is the only class that calls the 'plumbing' classes (AccountDAO and JMSSender). The Account class contains the business logic. Suppose you get $1 free as bonus if you deposit more than $1000. This business rule will be implemented in the Account class and NOT in the AccountService class. Remember, a Façade class is like the receptionist for a company, let's say an

insurance company. The receptionist itself does not perform insurance related work. The receptionist will send you to the right person in the company depending on your needs.

So the Accountservice acts as a Façade:



And the AccountService is the only class that calls plumbing classes so that the business logic is independent of the technical plumbing classes. This way the business logic classes are easier to understand, easier to write and easier to test.



Here you see that the service classes are in its own application layer: the service layer.