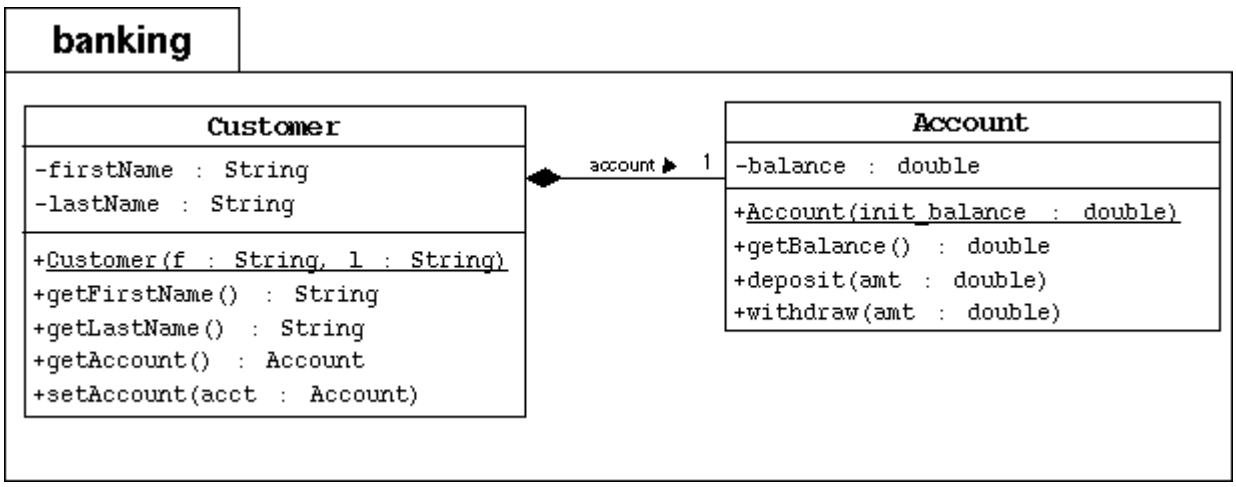
5. Objects and Classes (2)

Lab Exercise 5-1: Create Customer Accounts

Objective

In this exercise you will expand the Banking project by adding a `Customer` class. A customer will contain one `Account` object.



Note: You should know the UML notations to understand this diagram. For example, the `account`relationship can be implemented as a data attribute in the `Customer` class; even though, it is not mentioned in the "attributes" section of the `Customer` class block.

Directions

Start by changing your working directory to `lab05/exercise1` on your computer.

1. Make the `banking` directory.
2. Copy the Banking project files from the previous Banking lab to this directory.
3. Create the `Customer` class in the file `Customer.java` under the `banking` directory. This directory represents the Java package structure for your program. This class must implement the model in the above UML diagram.
    1. declare three private object attributes: `firstName`, `lastName`, and `account`
    2. declare a public constructor that takes two parameters (`f` and `l`) that populate the object attributes
    3. declare two public accessors for the object attributes; these methods `getFirstName` and `getLastName` simply return the appropriate attribute
    4. declare the `setAccount` method to assign the `account` attribute

5. declare the `account` method to retrieve the `account` attribute

4. In the main `exercise1` directory, compile and run the `TestBanking` program. You shoud see the following output:

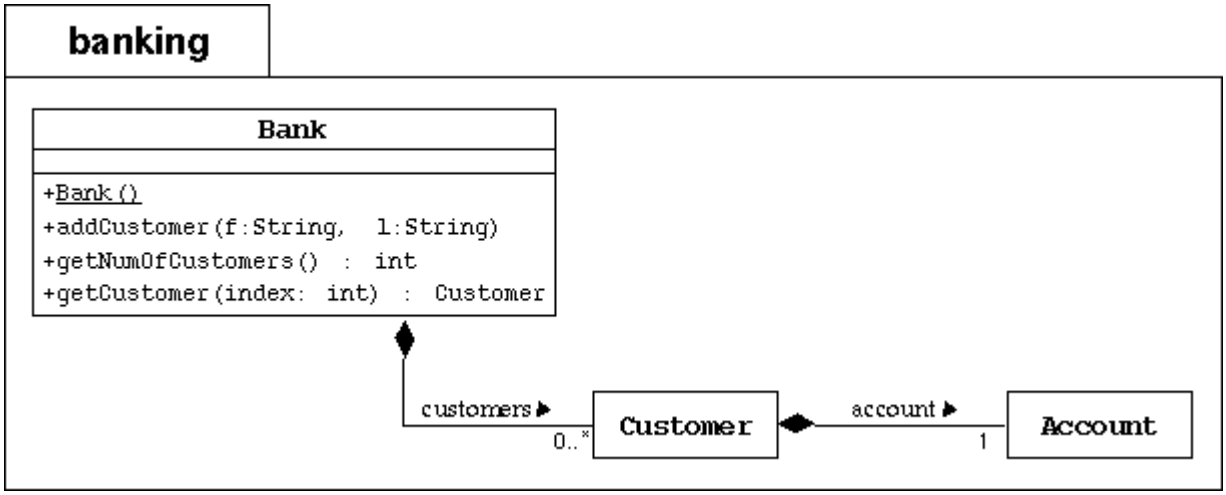
```
  Creating the customer Jane Smith.
  Creating her account with a 500.00 balance.
  Withdraw 150.00
  Deposit 22.50
  Withdraw 47.62
Customer [Smith, Jane] has a balance of 324.88
```

5. Objects and Classes (2)

Lab Exercise 5-2: Use Arrays to Represent Associations

Objective

In this exercise you will use arrays to implement the multiplicity on the association between a bank and its customers.



Directions

To the `banking` package, you will add the `Bank` class as modeled by the UML diagram above. A bank object simply keeps track of an association between itself and its customers. We will implement this aggregate association with an array of `Customer` objects. We will also need to keep a integer attribute that keeps track of how many customers currently exist in the bank.

1. Create the `banking` directory. Copy the previous Banking project files in this package directory.
2. Add two attributes to the `Bank` class: `customers` (an array of `Customer` objects) and `numberOfCustomers` (an integer that keeps track of the next`customers` array index).

3. Add a public constructor that initializes the `customers` array with some appropriate maximum size (at least bigger than 5).
4. Add the `addCustomer` method. This method must construct a new `Customer` object from the parameters (first name, last name) and place it on the`customers` array. It must also increment the `numberOfCustomers` attribute.
5. Add the `getNumOfCustomers` accessor method, which returns the `numberOfCustomers` attribute.
6. Add the `getCustomer` method. This method returns the customer associated with the given `index` parameter.
7. Compile and run the `TestBanking` program. You shoud see the following output:

```
8.   Customer [1] is Simms, Jane
9.   Customer [2] is Bryant, Owen
10.  Customer [3] is Soley, Tim
   Customer [4] is Soley, Maria
```