# Chap 5 – Business Objects that hold Data Structures

For our next Lab we want to make it so that the Customer class will contain a List of the Accounts that they own.

Currently the Customer class has the Properties:

id

pw

firstName

lastName

address &

email

We want to add a list of Accounts to these properties. Because a Customer has a lastName, etc, but also has a List of Accounts.

So we need to add a new Property to the Customer class. We could use a simple Array of Accounts, but my suggestion is that we build a new class that holds and Array of Accounts.(Yu could also use ArrayList instead of an Array, both will be acceptatble).

* The AccountList Class

The AccountList class will have 2 Properties and 2 Methods:

//Properties

public int count; //the number of Accounts in the list

public Account accArr[] = new Account[10];

//Methods

public void addAccount(Account a1)…//this method adds 1 account to our list

public void displayList()…//this method will display the whole list of accounts

class AccountList { // do not extend from the Account class

//======== Properties =========

public int count; //the number of Accounts in the list

public Account accArr[] = new Account[10]; //the list

//======== Methods =========

public void addAccount(Account a1){

accArr[count] = a1; //put the new Account object a1 into the accArr

count++; //increase the count by 1 since we just added one

} //end addAccount

public void displayList() {

for (int x=1; x<count; x++) {

accArr[x].display(); //the Account class has a display() method

} //end for

}//end displayList()

// main for Testing the AccountList class before we add it to the Customer class

public static void main(String args[]) {

AccountList alist = new AccountList();

Account a = new Account(“7777”, “3001”,”SAV”,500.00);

Account b = new Account(“8888”, “3002”,”CHK”,700.00);

alist.addAccount(a);

alist.addAccount(b);

alist.displayList(); //prints the list which should contain 2 Accounts

}

}//end class

* Adding the AccountList to the Customer class

* + Add the AccountList Property

package Business;

import java.sql.\*;

public class Customer {

// ====================== Properties =============================

private String cid;

private String cpw;

private String fn, ln, addr, email;

**ADD 🡪**

public AccountList aList=new AccountList(); //new Property for Customer

* + Modify display() method to display the AccountList

public void display() {

System.out.println("Customer ID = "+getCid());

System.out.println("Customer PW = "+getCpw());

System.out.println("Customer FirstName = "+getFname());

System.out.println("Customer LastName = "+getLname());

System.out.println("Customer Address = "+getAddr());

System.out.println("Customer Email = "+getEmail());

**ADD 🡪**

aList.displayList(); // 🡨 add so the account list will be displayed

}

* + Add Database code to get the Customer’s Accounts from the Database

This is a little complicated, but not too bad.

The Accounts will come from the Accounts Table, not the Customer table.

The SQL code is as follows:

String sql = “Select acctNo from Accounts where Cid=’”+getCustID()+”’”;

Next we need to execute this SQL code.

**ADD 🡪** To do this I add a new method called getAccounts().

This method includes the 6 database steps to execute the above SQL code.

Then I call the getAccounts(); method as the last line of code in the selectDB() method.

} //end catch

**ADD 🡪**

getAccounts(); // calls getAccounts() method

} //end selectDB() method

So the following code, that is in the main(), same testing code as for Lab #3:

Customer c1 = new Customer();

c1.selectDB(“3001”);

c1.display(); // 🡨 displays all cust data include the list of accounts

* + Code in the getAccounts() method to get each account and add it to the aList

// DB Steps #1 through 4 – copy from selectDB method and modify for new SQL code above

//STEP #5

String an;

Account a1;

while rs.next() {

an = rs.getString(1);

a1 = new Account();

a1.selectDB(an);

aList.addAccount(a1);

}

//Step #6 – close connection