IDS 401

# Assignment 3

# Instructions

### Deadline

In order to receive full credit, this assignment must be submitted by the deadline on Blackboard. Submitting your assignment early is recommended, in case problems arise with the submission process. Late submissions will be accepted (but penalized 10pts for each day late) up to one week after the submission deadline. After that, assignments will not be accepted.

### Assignment

The object of this assignment is to construct a mini-banking system that helps us manage banking data. Notice that you can copy and paste your code from previous assignments. There are multiple functions in this system. First, you should be able to create an array of checking account objects (think of this as your data). Then you should be able to search the array to find indexes/objects that match your search value.

The class name should be Checking3, an underscore, and your NetID run together; for example, if your NetID is abcde6, your class name would be Checking3\_abcde6. For this assignment, we will create text-based tools for creating, deleting, and sorting checking account objects. All requirements from the previous assignment remain in effect (meaning you should still keep the two data members and three methods from last assignment). The new features for this application are:

1) ***Outside*** your main class, Checking3\_abcde6, create another class named *Checking.* Similar to the Checking2 class you created from last assignment, you have two data members AccNum and Balance. It also has two methods from last assignment, Deposit and Withdraw.

2) Checking3\_abcde6 data members:

a) a static *Checking* array named *AccInfo.*

b) a static int variable *NumAcc*.

c) add a static *Checking* array named *p*. This *p* is the “active Checking” array and should be used to store temporary data, such as a newly-created *Checking* but not yet stored permanently in the program.

3) Checking3\_abcde6 methods:

a) *main*:

1) First ask the number of checking accounts users want to record in the program. Then create an array for *AccInfo*, the data member of Checking3\_abcde6 class. The size of this array (*NumAcc*) is determined by the number of checking accounts user just entered.

2) And then call the new object’s *getInput* method (3.b below)

b) *getInput*: a loop that processes user commands via the console. This method allows users to select different functions of the system. The following commands should be accepted: create, type search, and exit within the loop. The first two should call the corresponding methods (3.c and 3.d below) and an exit command should end the program. In other words, if users do not type exit, the program will keep running This *getInput* method should take no parameter and return nothing.

c) *create*: This method helps create an array of checking account for the system to manage, and assign the created array to *AccInfo*. Runs a loop to request the account number and balance of each checking account from users. That input is used to create an array of *Checking* object, which the method returns. You should also store the account number and balance data of each checking account in corresponding *Checking* object. This new array should be received by *AccInfo* as an array. For example, below is how you should create this array using create() method:

*AccInfo = new Checking[n]; AccInfo = create();*

Notice n in previous line is the number of *Checking* in *AccInfo*.

Particularly, pay attention to how we declare an array of object and then create these objects of this array one by one.

d) *BalanceSearch*: This function helps search for checking accounts with certain number of balance. Receives no parameters; now requests input via the console. If *AccInfo* is null or size zero, the method should immediately tell the user “No checking accounts to view!”. Else, asks for the balance to group. The method will display a list of all indexes from *AccInfo* with the input balance; if none exist, a “No matches found!” message will be displayed. The method will return an array containing all elements with the input number (a size zero array if no matches are found), which should overwrite *p*. For example, we have five checking account objects in the array, and three of them have balance of 100. The *BalanceSearch* method will print the index of these three checking accounts within *AccInfo* array, and also create and return a new Checking array with all these three checking accounts. For example:

*p = BalanceSearch ();*

### Additional Constraints

The program should be implemented as a single file containing the two Java classes Checking3\_abcde6 and Checking. For Checking3\_abcde6 class, all data members should be declared as private; all methods should be declared without any visibility modifiers.