REVIEW OF JAVA PROGRAMMING: ARRAY MANIPULATION

1. OBJECTIVES

(a) To review techniques on Java programming, the pre-requisite of CSC/CPE105.
(b) To review techniques of performing input / output from a plain text file.
(c) To review techniques of class/structure declaration and instantiation.
(d) To review techniques of array manipulation.

2. LABORATORY

This lab will be conducted in the Computing Lab 1 (N4-B1-8) in SCE. This is an individual experiment. You cannot use the classes that implement the **Collection** interface (please refer to http://java.sun.com/docs/books/tutorial/collections/), except for Array class. We are going to implement these classes (like Vector, LinkedList, Stack, TreeSet, etc.) in CPE/CSC105, instead of using them directly from Java.

No make-up is allowed for students who have missed their stipulated lab classes without any acceptable excuse like having a valid leave of absence from the school or on medical leave. The students will be deemed to have failed the particular lab work. In case you have valid reasons to do makeup, you must inform the lecturer in charge. The makeup should be done within the same week, during the lab sessions attended by other groups.

3. EQUIPMENT

Hardware: The PCs running under the LINUX environment in Computing Lab 1.
Software: NetBeans, SUN JAVA compiler (**javac**) and interpreter (**java**). They can be downloaded from http://java.sun.com/javase/downloads/index.jsp. Choose "JDK 6 Update 10 with NetBeans 6.5". Your programming will only be tested by the lab markers using javac and java on the lab PCs. None other is accepted.

4. EXPERIMENT

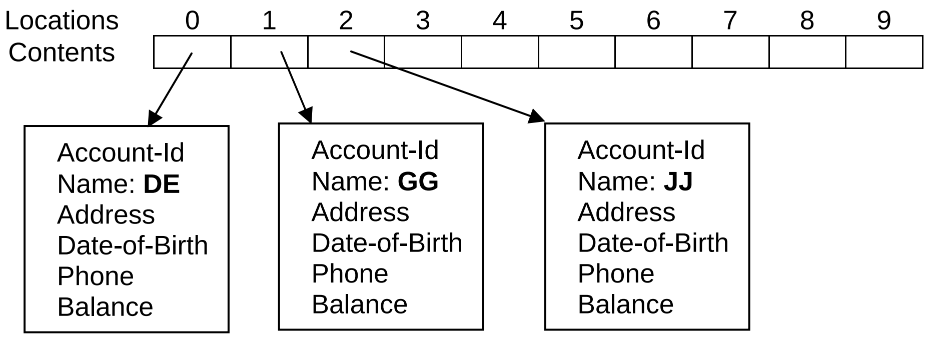
The ABC Bank has a set of customers who deposit or withdraw cash from their bank accounts regularly. Information about the customers is stored persistently as a text file on the bank's hard disk. At the beginning of each month, all the customer records are read into the computer from the hard disk. At the end of the month, all customer records will be saved back into the file. This experiment handles these 2 operations monthly. The details include:

- The input (output) method of customer records from (onto) a text file.
- The storage of records into an array in ascending order.

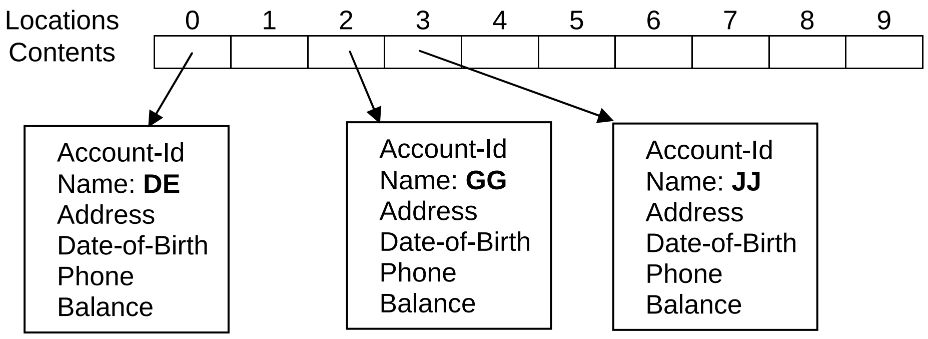
(a) A customer record contains the following attributes: a unique **account id** (an integer type number), **name** (of at most 20 characters), **address** (of at most 80 characters), **date of birth** (of at most 10 characters), **phone number** (of at most 8 characters), and an account **balance** (a double type number). Declare a suitable class called **Customer** to represent each customer record. One can assume that there are at most 30 customer records.

(b) Write a method **readFile** that reads customer records from a text file and stores them, in ascending order according to the **name**, in an array. The letters of each name must be converted to the **upper case letters**. The method reports the number of customer records successfully read as its return value. No duplicate record (according to the account id) should be read in.

The array stores customer records in ascending order of **name** and is compact at all times. This implies that when inserting a new record, some records must be right-shifted to make space for the new record. Suppose we want to insert a record with name **FOX** and the array is currently as follows:



We show only the names (i.e. **DE**, **GG**, and **JJ**) in records here. Records starting with name **GG** and onwards to the right should be shifted by one slot to the right so that an empty slot at location 1 is available for the new record as shown below.



(c) Write a method **displayRecords** that prints customer records on the screen. The method should first print the name, and then the balance and account id, followed by the rest of the information in any arbitrary order. We do not compute interests in lab 2. Interests will be deal with in labs 4 and 5.

(d) Write a method **writeFile** that writes the original customer records back to a separate text file, **newCustomers.txt**. The written records should be in ascending order of customer names. The method returns the number of customer records successfully written.

(e) Write a method **menu** that displays all the above methods for test run. The lab markers can select method 1, 2, 3, or q (or Q) to input data, display data, output data and quit in any order such that it is possible for the markers to input or output data more than once. When command 1 is selected, the program should prompt for the name of input file.

Although each of the above tasks is to be implemented as a single method, it may be necessary to create additional sub-method(s) to handle portions of the method. This is especially true if the original method is too long or contains more than one functionally related group of statements. When you write a method, remember that this method is to work for all possible inputs. Not on just your test inputs. You must test for all conditions that might possibly arise; print out error messages as needed.

A text file **customers.txt** of customer records has been created in the directory **/home/staff1/csc105/lab2** (see also appendix A) to assist the development and testing of your program. Its contents will be changed (while still adhering to the same data format) during grading in order to test your programs' robustness. The objective is to ensure that you do not hardcode your program to work exclusively for the given sample customer records. Your program should read input of menu choices from the keyboard. However, one can save typing by using redirection as "**java Lab2 < commands**" where **Lab2** is the name of your class file and **commands** is a file which records your desired input of menu choices from the keyboard. A typical **commands** file is provided for your convenience in appendix B. Check also the sample output file and a **readme** file in **/home/staff1/csc105/lab2** for any last minute hints or changes.

Create a sub-directory called **cpe105/lab2** or **csc105/lab2** (depending on whether you are taking CPE105 or CSC105) in your home directory. Please note that the grader will only look into this sub-directory to find, compile (like "`javac Lab2.java`") and test run (like "`java Lab2 < commands`") your program. You should provide a readme file that gives instructions to compile and run your program. You should not have irrelevant source files in the directory.

5. ERROR HANDLING

We will first test your work with error-free inputs from file and keyboard. It would be ideal if your programs can do appropriate error handling like

- Input name is longer than 20 characters
- Duplicate records are read in
- More than 30 records are read in

Duplicate records should be ignored and appropriate error statement should be printed. However, the program should still be run as usual. Apart from these, you do not need to worry about other types of errors.

6. REPORT

This lab is the first part of a larger project composed of labs 2, 4 and 5. You do not need to submit report for this lab. Be prepared to submit the final report of lab 5. You can refer to lab 5 for report format.

7. ACADEMIC HONESTY AND COLLABORATION

Cooperation is recommended in understanding various concepts and system features. But the actual solution of the assignments, the programming and debugging must be your individual work, except for what you specifically credit to other sources. (Your grade will be based on your own contribution.) For example, copying without attribution any part of someone else's program is plagiarism, even if you modify it and even if the source is a textbook. You can document the credit to other sources at the start of your program code listing. The University takes acts of cheating and plagiarism very seriously: first time violators may fail the coursework component of CPE105 / CSC105. Any wholly (or partly) copied (or being copied) programs will receive zero mark.

8. REFERENCES

[1] Text and reference books for CSC/CPE 105.

[2] http://java.sun.com/docs/index.html; http://java.sun.com/j2se/

Appendix A: customers.txt

```
Account Id = 123
Name = Matt Damon
Address = 465 Ripley Boulevard, Oscar Mansion, Singapore 7666322
DOB = 10-10-1970
Phone Number = 790-3233
Account Balance = 405600.00

Account Id = 126
Name = Ben Affleck
Address = 200 Hunting Street, Singapore 784563
DOB = 25-10-1968
Phone Number = 432-4579
Account Balance = 530045.00

Account Id = 65
Name = Salma Hayek
Address = 45 Mexican Boulevard, Hotel California, Singapore 467822
DOB = 06-04-73
Phone Number = 790-0000
Account Balance = 2345.00

Account Id = 78
Name = Phua Chu Kang
Address = 50 PCK Avenue, Singapore 639798
DOB = 11-08-64
Phone Number = 345-6780
Account Balance = 0.00

Account Id = 234
Name = Zoe Tay
Address = 100 Blue Eyed St, Singapore 456872
DOB = 15-02-68
Phone Number = 456-1234
Account Balance = 600.00
```

Appendix B: The commands file

```
1
customers.txt
2
3
1
customers2.txt
2
3
Q
```