**Database Management Systems**

**Homework #1**

**Matthew S. Hartstein & Sailesh Sai Sirigineedi**

**Objectives:**

The objectives of this assignment are (1) to implement a simple, file-based database using a sorted file of fixed length and (2) to understand and practice file management techniques to implement a database system. This program was developed with Java and includes the following libraries: “java.io.\*”, “java.util.\*”, “java.nio.file.\*”, “java.nio.file.StandardOpenOption.\*”, and “java.nio.charset.StandardCharsets”. In total, the program required approximately 25 hours to complete and was developed by two University of Arkansas undergraduate students - Matthew S. Hartstein and Sailesh Sai Sirigineedi - for Database Management Systems (CSCE 4523, Spring 2020).

**Approach:**

* How did you implement this? What languages? Strategies? Design?

We implemented this problem using Java programming language. Our strategies were to break each individual task into different methods/functions. We would have the states relating to the database and files as static variables of the class in order to access them from different functions.

* What format did you choose for your record? What size and order of fields?

For formatting our records, we found the maximum length for each field and added that in as whitespace and then separated each field of the record with a comma and added a comma at the end as well. The order of the fields was the same as the csv file as we didn’t want to complicate things.

* What delimiter? What total record size? Show a sample record.

The delimiter we used was a comma. The total record size was 81 bytes. The following is a sample record from the record file.

94 ,3M ,MAPLEWOOD ,MN,55144,91584 ,

* What did you use the config file for? Show the config file.

We used the config file to show the field names as well as the maximum length for each field and the number of records. The following is the config file:

500

RANK,NAME,CITY,STATE,ZIP,EMPLOYEES

3

38

19

2

5

7

* Did you use sample code?

We used sample code from the class record for getting records and binary search and we looked up some stuff for reading files as this was all new to us. However we had to understand which parts to use and couldn’t just use the sample code word for word as it wouldn’t have worked exactly for our project.

* How did you handle the overflow file (methods)?

We had addRecord method and deleteRecord method to handle the overflow methods.

**Results:**

* Discuss error handling, efficiency.

For error handling we would debug each individual method one at a time to figure out where the bug was occuring. This worked effectively because we could narrow it down easier and the changes would be small rather than changing the entire program just to fix one bug.

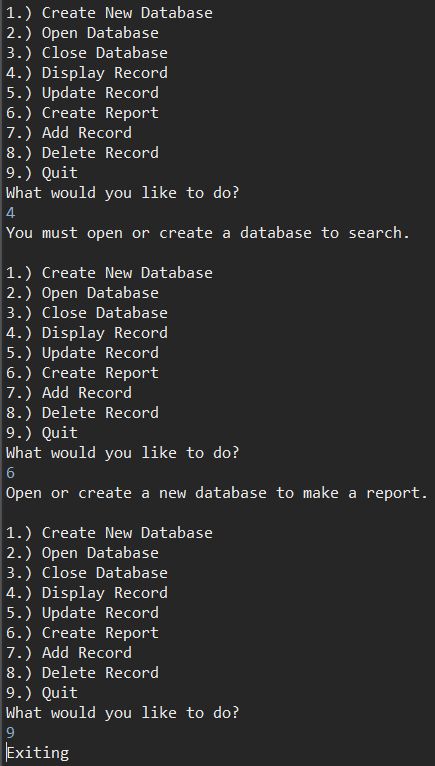
* What worked well? What didn’t?

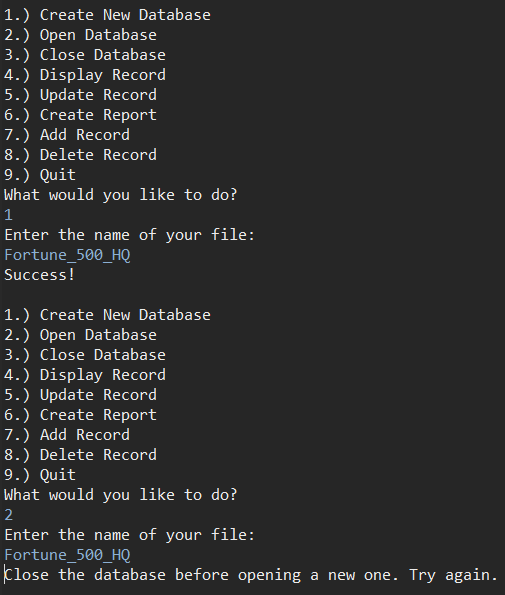
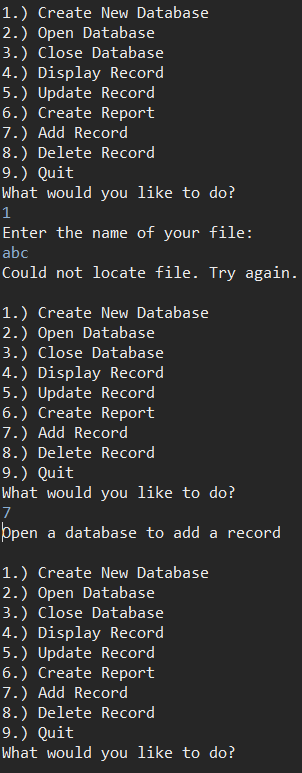
Creating a database, opening a database, closing a database, displaying a record, updating a record, creating a report all work pretty well. Adding and deleting a record

**Testing:**

* Describe how you tested your program.

We tested our program in various ways to ensure proper error checking occured as the user loops through the program (programLoop()). If the user wants to open a database before it is created, he/she will be sent back to the menu of operations. If the user wants to close a database, there must be a currently opened database to close. If there is an opened database, the program successfully allows the user to close their database. The displayRecord() function will not display a record if there isn’t an opened database, but if there is, it will display a given record from that database. The same rule applies for the updateRecord() function, as well. If the user wants to create a report, a database must be opened and successfully created. If successful, an output file (output.txt) is generated with the first 10 records (sorted by key; name) nicely formatted to the text file body. Moving on, if the user wants to add and/or delete a record, they must also have an opened database to add/remove from. If a database isn’t opened, the user will not be allowed to perform these operations. Even though we were not able to successfully implement these two functions, we were still able to add error checking to verify if a database is opened or not. Located below are several screenshots of our program displaying it’s error checking capabilities:





**Typescript:**

* Include the typescript here (or upload it separately).

username@turing:~$ script

username@turing:~$ javac Homework1.java

username@turing:~$ java Homework1

1

Fortune\_500\_HQ

4

3m

5

Walmart

Rank

2

3