CS22 Coursework Report

For testing my phonebook program I chose using few different methods. Testing was done as following:

Benchmarking on various sizes of the text files

Performing operations on an empty directory from all the directory implementations

Code:

```
System.out.println(directory.deleteEntry(1) + " expected false, deleting by num empty dir");
System.out.println(directory.deleteEntry("Mike") + " expected false, deleting by name in
empty dir");
System.out.println(directory.changeNumber("Mike", 5) + " expected false, change num in empty
dir");
System.out.println(directory.lookup("asd")   + " expected -1 (eg not found), empty dir");
System.out.println(directory.insertEntry("Johnson", "MS", 1234) + " expected false-Initial
not matching surname");
System.out.println(directory.insertEntry("Johnson", "MJ", 12344)+ " expected false-More than
4 digits");
System.out.println(directory.insertEntry("Johnson", "MJ", 1244) + " expected true" + " after
inserting ");
System.out.println(directory.insertEntry("Johnson", "MJ", 1244) + " expected true" + " after
inserting");
System.out.println();
System.out.println("Print directory");
System.out.println(directory.printDirectory());
System.out.println(directory.deleteEntry(1244)+ " expected true, delete by number");
System.out.println(directory.deleteEntry("Johnson") +      " expected true, deleting last
entry by surname");
System.out.println(directory.printDirectory());
```

Output:

```
false expected false, deleting by num empty dir
false expected false, deleting by name in empty dir
false expected false, change num in empty dir
-1 expected -1 (eg not found), empty dir
false expected false-Initial not matching surname
false expected false-More than 4 digits
true expected true after inserting
true expected true after inserting

Print directory
Johnson MJ      1244
Johnson MJ      1244

true expected true, delete by number
true expected true, deleting last entry by surname
Empty directory
```

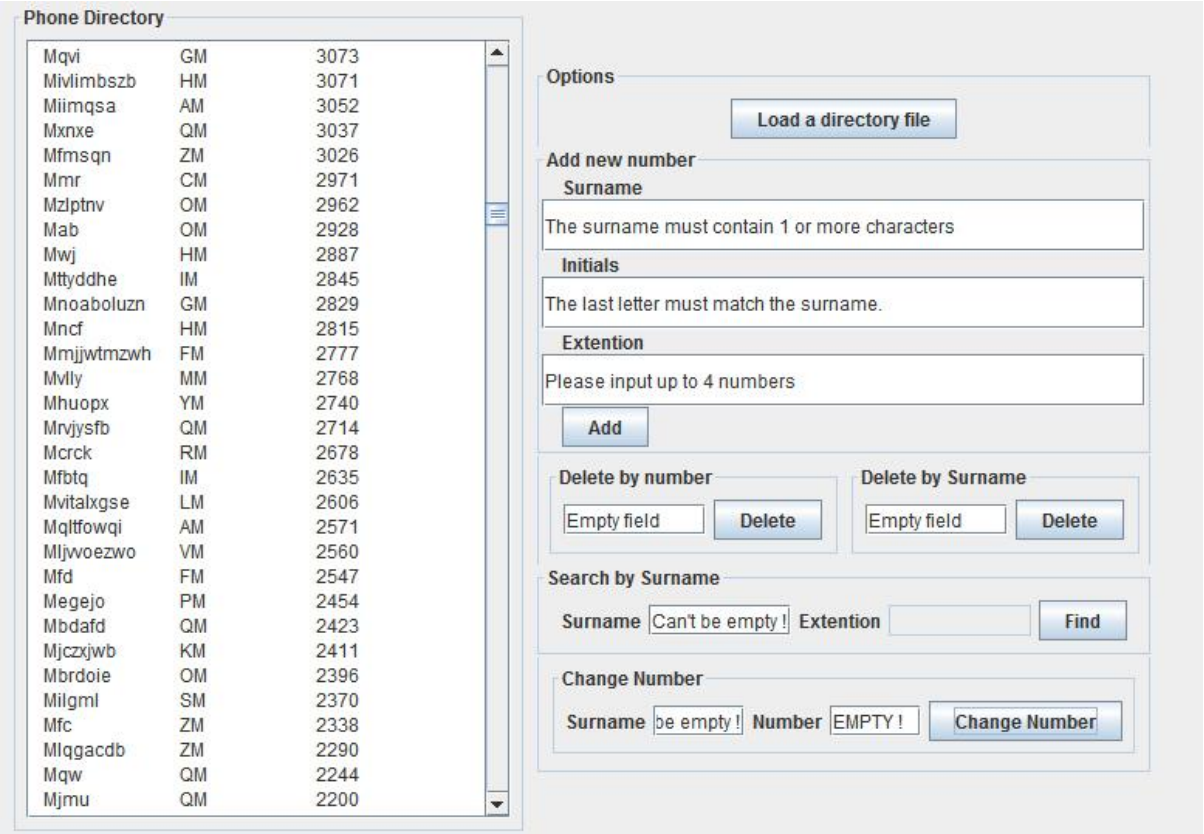
The output is taken from TestMore.java

Additional benchmarks

The program comes with test classes: TestArray.java, TestHash.java and TestLinked.java where further tests have been done such as deleting 1000 random numbers, 1000 random names, inserting in the middle, end, beginning, looking up random people and so on.

Performing operations on different length of text files:

3 different implementations of the directory work without a problem with the GUI, just change the structure type. Loading directory performs as expected and the GUI outputs to the user if anything goes wrong.



The program performs as expected with a file with 10 entries, 5000 entries and 55000 entries.

Timing results

Array directory:

```
52 730 133 after reading 2 000 entries from file
200 839 424 after inserting 1 000 entries
45 673 357 after deleting random 1000 names that exist
16 567 356 after deleting 1000 by number
21 341 539 after deleting 1000 names from the beginning
19 821 208 after deleting 1000 names from the end
14 832 901 after deleting 1000 names with a miss
8 9102 after last lookup
102 164 after first lookup (depends on the implementation)
6 970 475 after changing the numbers of the first 1000 people
9 232 076 after changing the numbers of the last 1000 people
       ======== Performance Report      =======
409 590 723
```

```
List directory:
```

```
48 048 785 after reading 2 000 entries from file
61 268 056 after inserting 1 000 entries
112 921 969 after deleting random 1000 names that exist
12 787 289 after deleting 1000 by number
33 000 360 after deleting 1000 names from the beginning
28 372 659 after deleting 1000 names from the end
38 221 452 after deleting 1000 names with a miss
595 723 after last lookup
695 554 after first lookup (depends on the implementation)
20 052 8733 after changing the numbers of the first 1000 people
97 731 260 after changing the numbers of the last 1000 people
       ======== Performance Report      =======
657 803 904
```

Hash directory:

```
61 971 540 after reading 2 000 entries from file
52 959 652 after inserting 1 000 entries
66 033 841 after deleting random 1000 names that exist
54 648 390 after deleting 1000 by number
709 083 after deleting 1000 names from the beginning
1 360 787 after deleting 1000 names from the end
116 159 after deleting 1000 names with a miss
12 129 after last lookup
98 899 after first lookup (depends on the implementation)
658 701 after changing the numbers of the first 1000 people
11 189 986 after changing the numbers of the last 1000 people
       ======== Performance Report      =======
251 151 210
```

The time is in nanoseconds.

Reading from file is relatively slow because of the read speed. The list directory has higher performance, because it inserts instantly, whereas the array has to resize before add is performed.

When inserting 1000 entries, hash and linked list implementations are significantly more efficient due to the nature of the data structure (instant addition without resizing). Hash directory is slightly faster because it performs hashing.

Deleting 1000 random names is more efficient in array directory, because of the immediate random access that allows for binary search (a very efficient algorithm). The hash directory implementation perform reasonably, because of the hashing algorithm that speeds up any sort of searching. The linked directory performs poorly because if there is a miss, it has to iterate till the end of the whole directory due to the structure of the implementation.

Deleting 1000 by number is very inefficient in all implementations, because finding happens by brute force.

Hash directory is very efficient when deleting by surname. Worst case scenario it is 26 times faster. It is still considerably faster in comparison to array directory despite its usage of binary search. The array directory lacks the speed because it has to resize every time when an entry is deleted.

The same characteristics can be observed with the rest of the methods where finding certain information is very slow when performed by linked directory. The hash directory is often the fastest way to handle the tasks where in certain corner cases array directory performs better.