CSC1022 Coursework 2018-19

Documentation

This java project has an interface called Directory which has three different implementations. The first implementation of the interface uses an array as data structure, the second implementation makes use of linked lists, and the third implementation, which is the most complex one out of these three, uses an array of linked lists and also creates and manipulates hash codes of the objects in order to increase the performance of the program. The graphical user interface is user-friendly as it tolerates mistakes such as invalid data format, surname that starts with lower case letter, etc. Program informs the user about successful and unsuccessful operations and also asks to re-enter data if input-related errors are detected. The three classes that implement Directory interface does not have exception handling, therefore if the methods that are in these classes are tested separately, they can be more vulnerable in terms of fault tolerance.

Every implementation of the project can be tested in classes App.java, TestArrayDirectory.java, TestListDirectory.java, and TestHashDirectory.java respectively. For testing purposes, I created text file with various different entries. The file is called “inputData.txt”.

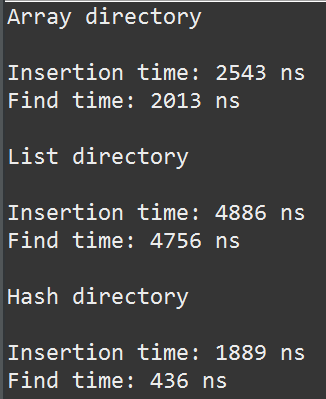
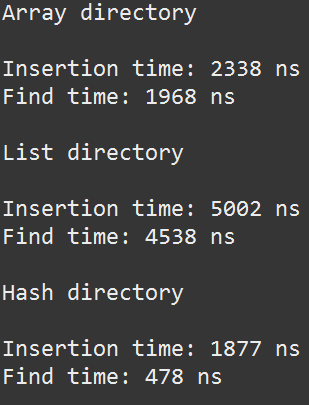
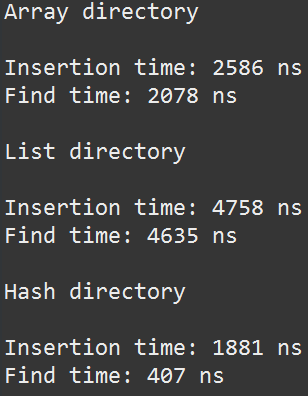
Instead of using void methods to insert or delete entries, or change number on existing entry, I made use of Boolean methods that return true or false depending on whether operation was successful or not. This strategy helped me with the implementation of the graphical user interface as I can easily check if operation has been successful or not and inform the user about it. In addition to this, using Boolean methods instead of void, helped me to avoid complexity of exception handling.

Testing

ArrayDirectory vs ListDirectory vs HashDirectory

ArrayDirectory class has medium performance in comparison with the other two classes that implement Directory interface. ListDirectory has the lowest speed for insert and findExNumber methods. HashDirectory has the best performance in terms of speed and reasonable amount of code and its complexity.

Performance is evidenced in the 3 screenshots below:

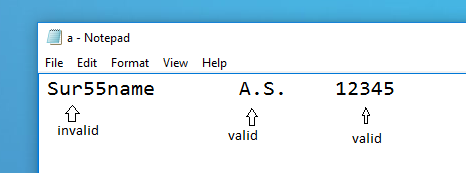
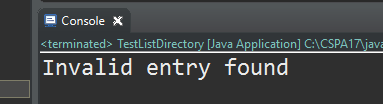
As it can be seen in the screenshots above, HashDirectory implementation has the best performance in terms of speed. The program was tested by creating 10,000 random strings that represent surname, therefore the performance differs slightly every time the program is run.

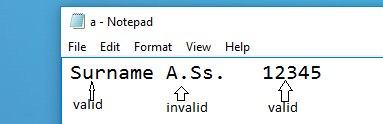
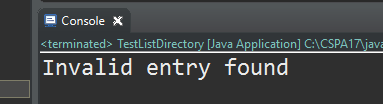
The implementation with arrays has quite reasonable speed because there are no complicated processes taking place such as referencing objects, etc. Arrays are not very efficient because if we want to insert a new object, we need to shift all the other objects to make space for the new one. Look up method is not very efficient as well as it needs to iterate the array until element is found.

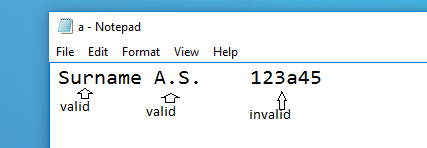
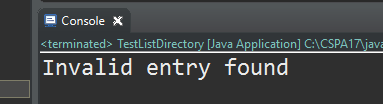
The implementation with linked list has the slowest speed because objects are chained to each other through references therefore it takes considerable amount of time to either insert a new entry or look up for extension number. When new entry is inserted there are processes of changing the references taking place which cost time. When look up method is invoked, program iterates through the list until it finds surname. If surname is in the beginning of the list, we are lucky, if it is at the end of the list it will take time to go there and match the surname.

The implementation of hashing is the fastest as it uses hash codes to tell the program which particular list of the array keeps information about particular entry. In this way program does not need to iterate every list to access data when surname is given. It needs to refer to particular list and it saves huge amount of time. When it comes to insert a new entry, it is always faster as it refers to particular list rather than iterating one big list.

If we try to read in invalid entry, program stops and prints message about the invalid entry to the console window.

 = 

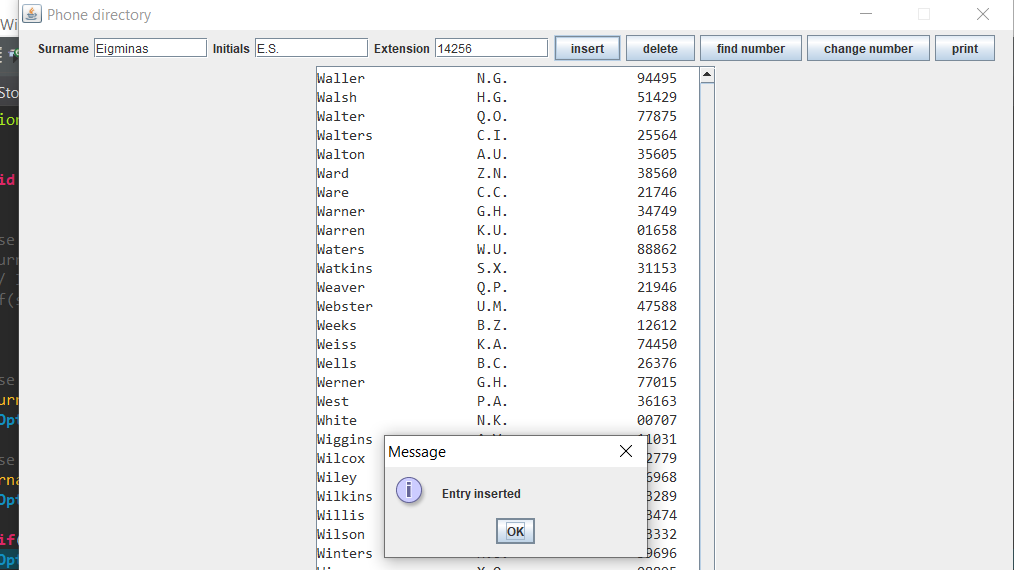
 = 

 = 

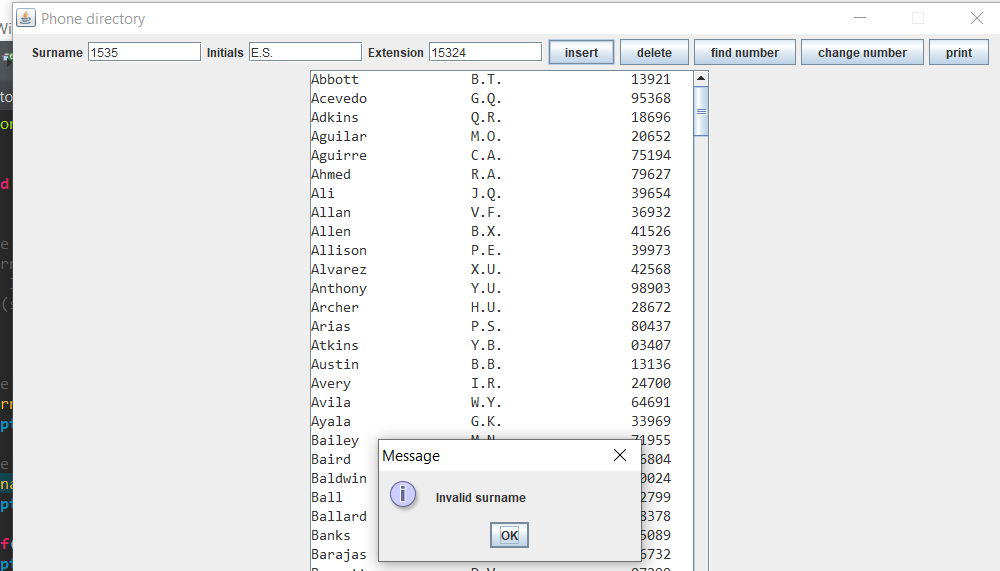
App

Insert testing

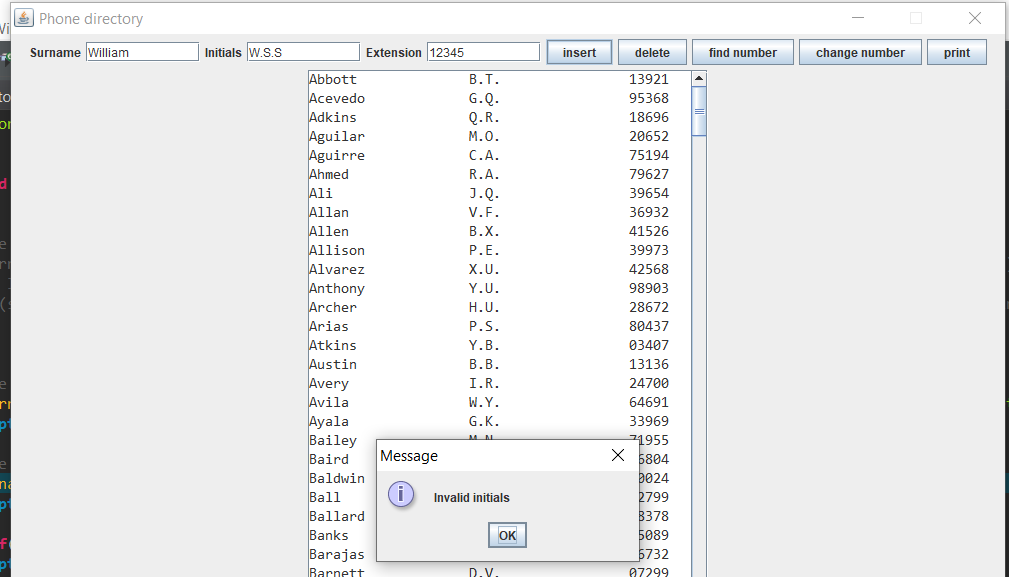
If valid data is entered, entry can be inserted:



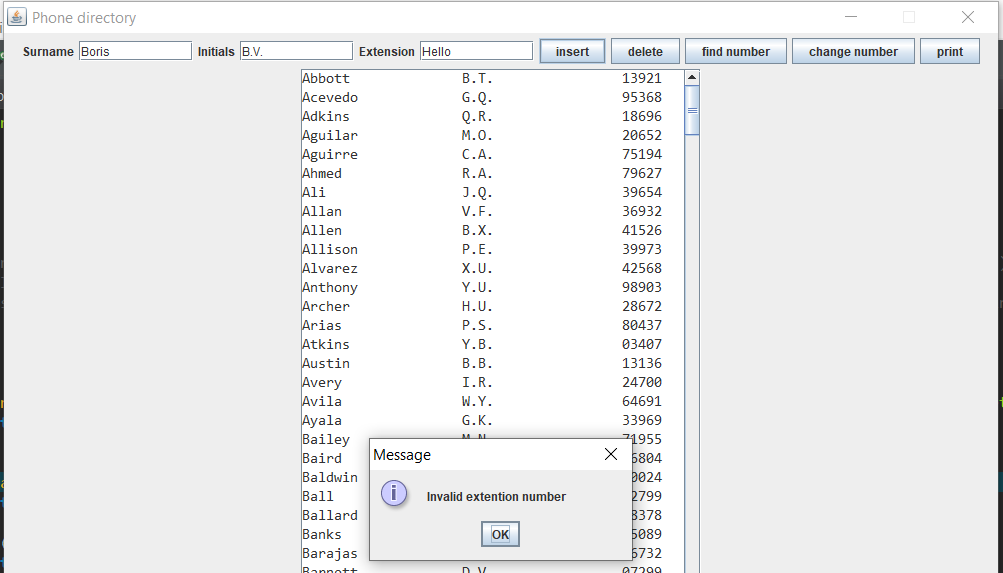
If invalid surname is entered:



If invalid initials are entered:

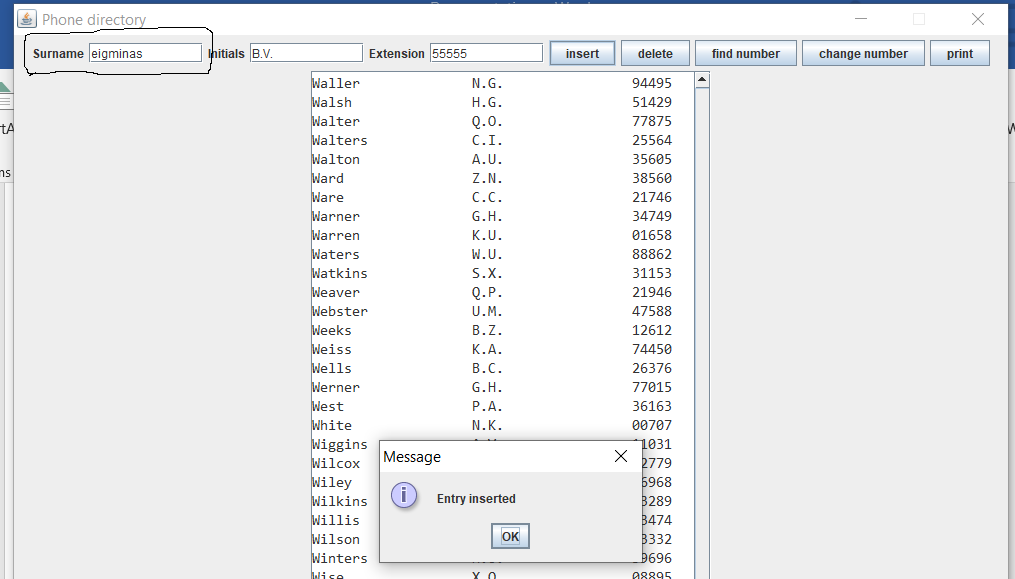


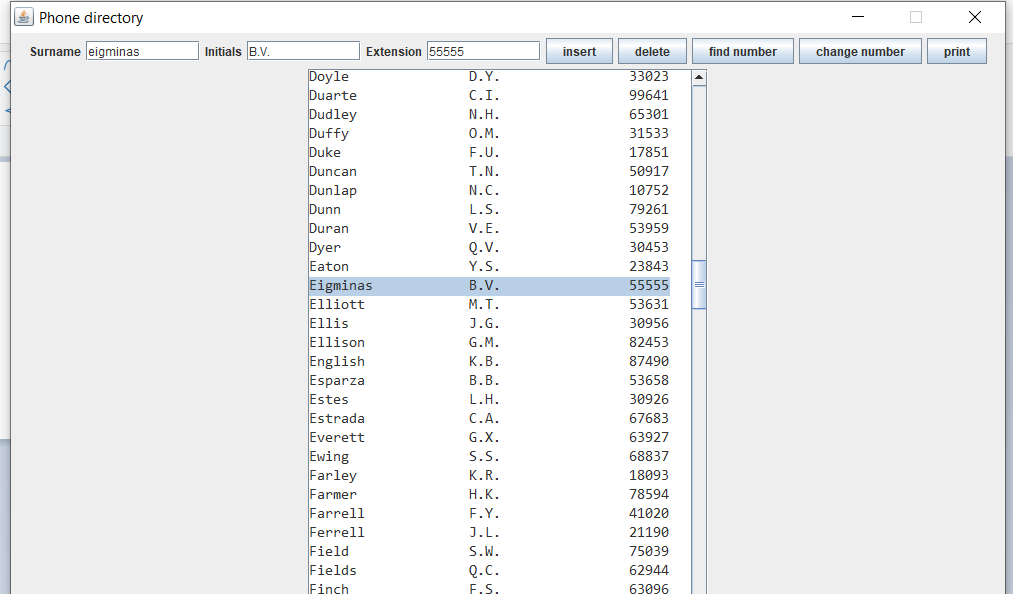
If invalid extension number is entered:



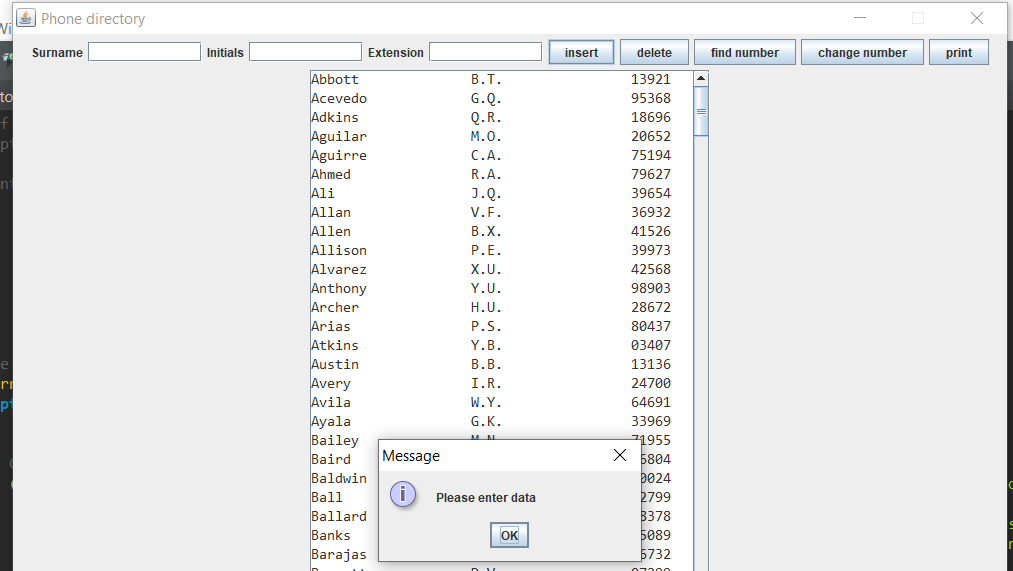
Program tolerates surnames that start with lower case letter and it automatically converts the surname to valid format.

For example: “ali” will be converted to “Ali”



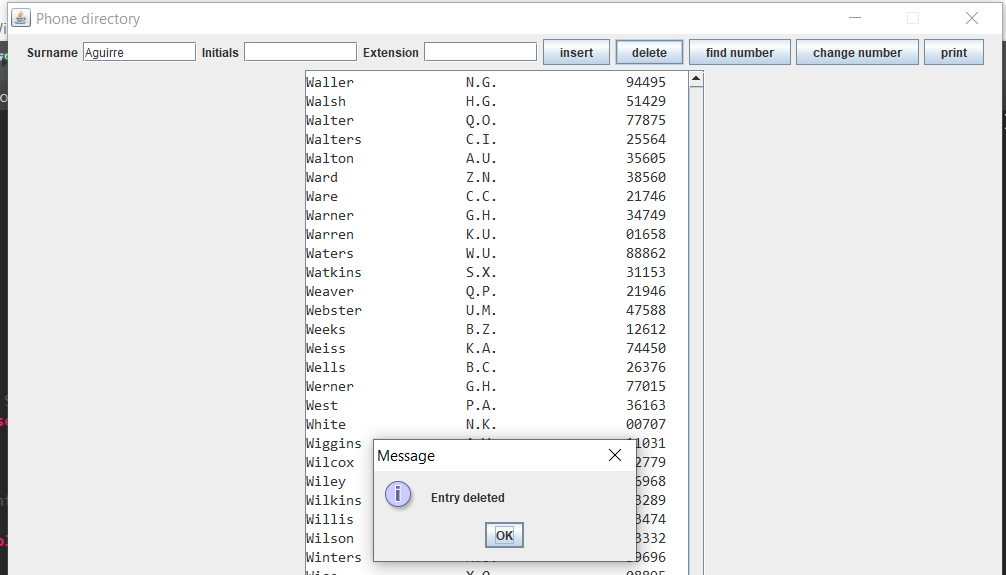


If no data is entered:

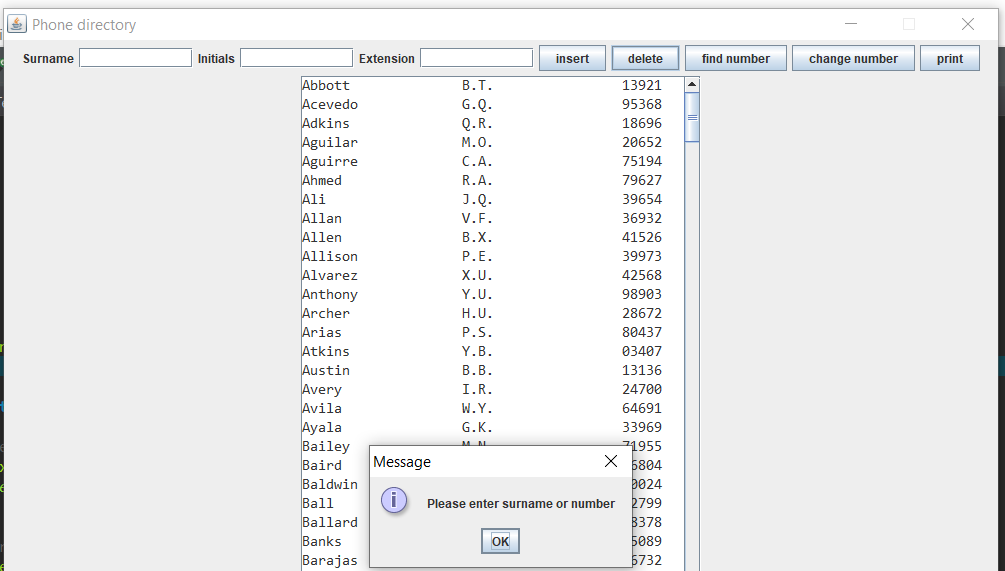


Delete testing

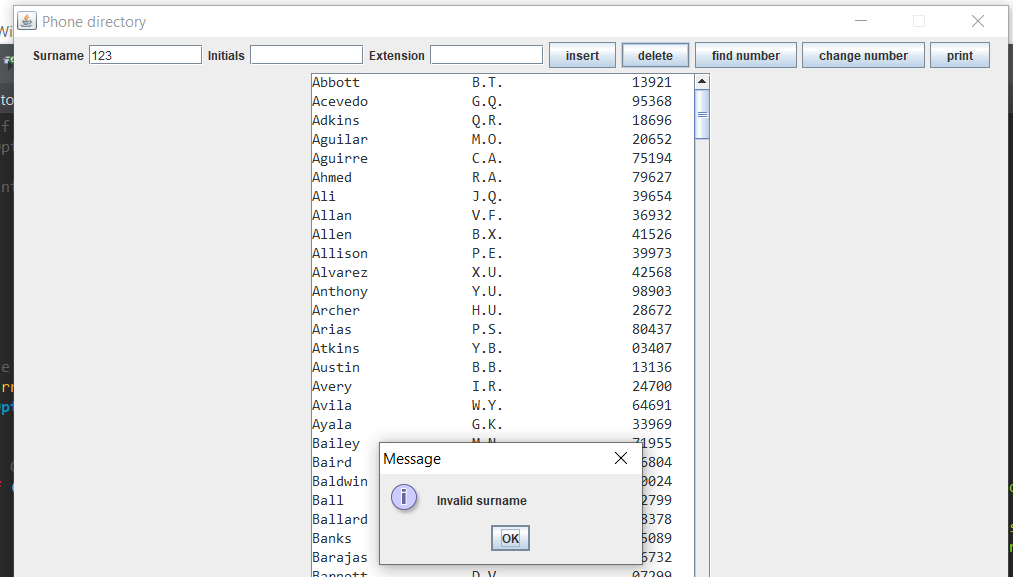
If valid surname, which exists in the directory, is entered, it gets deleted



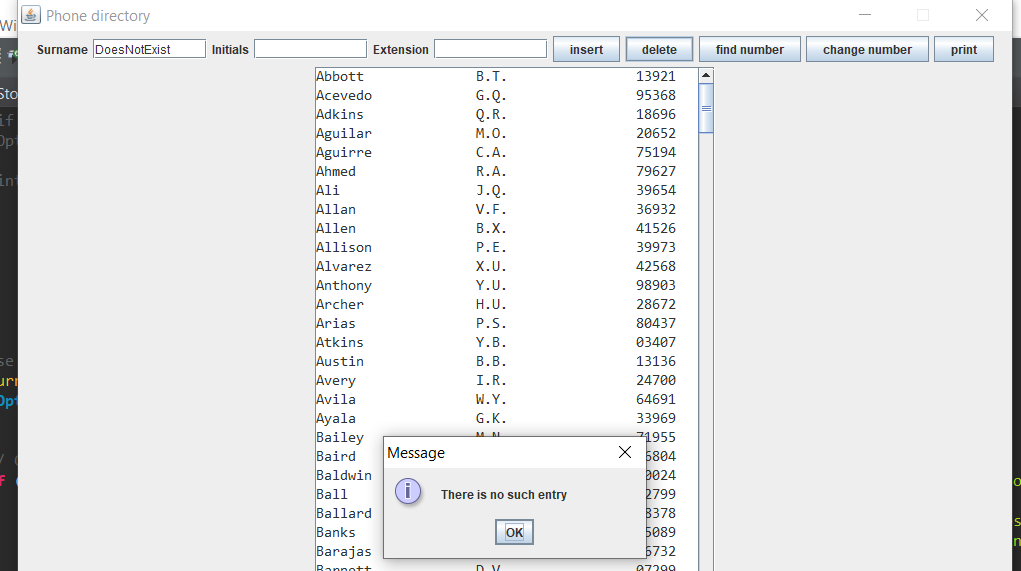
If no data is entered:



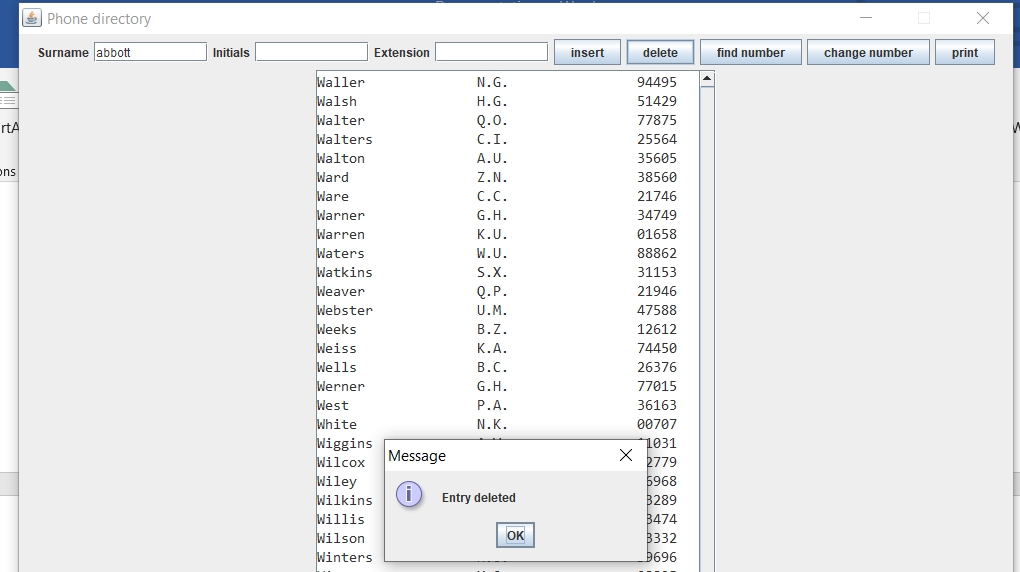
If invalid surname is entered:



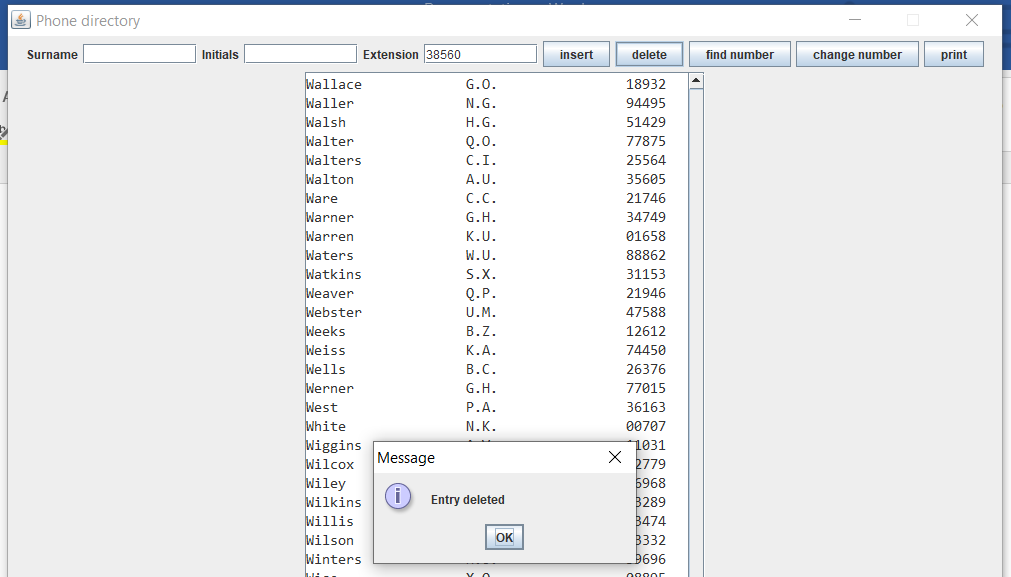
If surname, which does not exist in the directory, is entered:



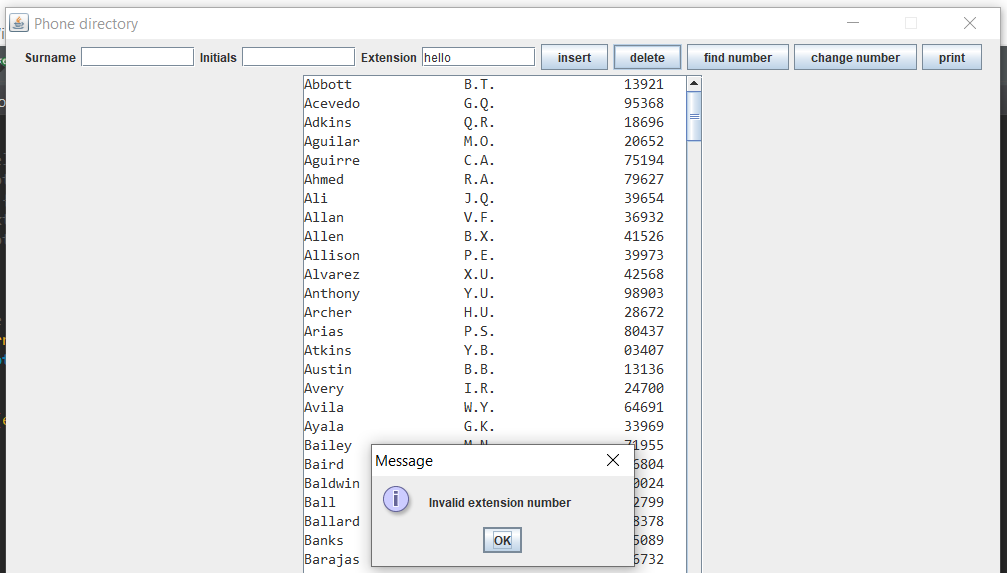
If a user enters surname which starts from lower case letter, entry will still be deleted:

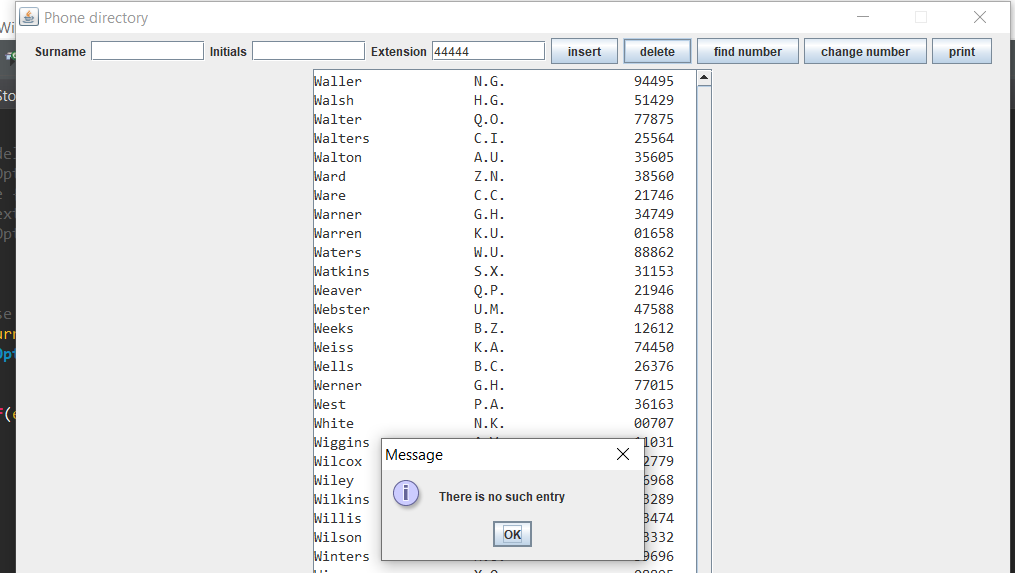


If valid number that exists in the directory is entered, entry will be deleted:



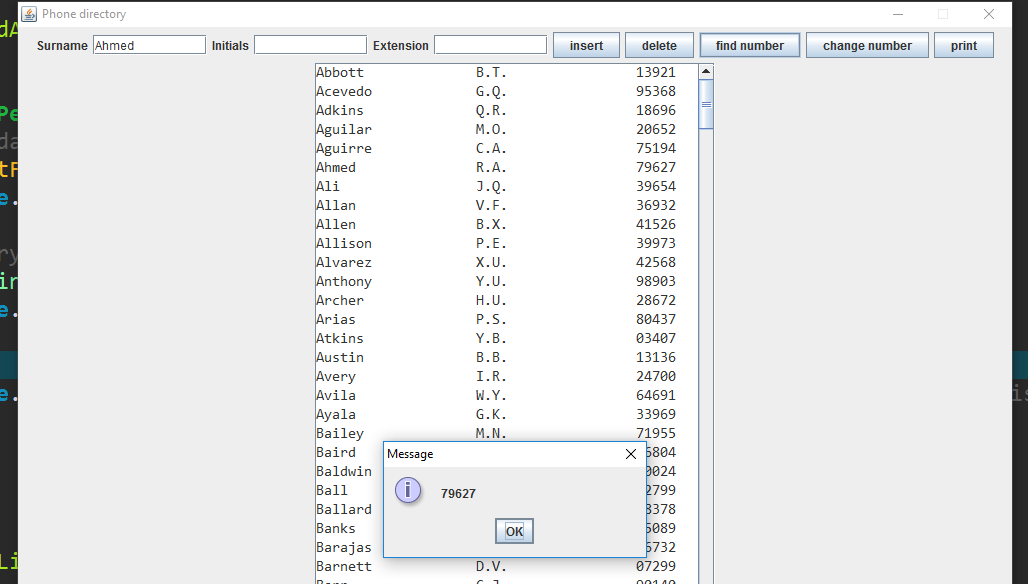
If invalid number is entered:



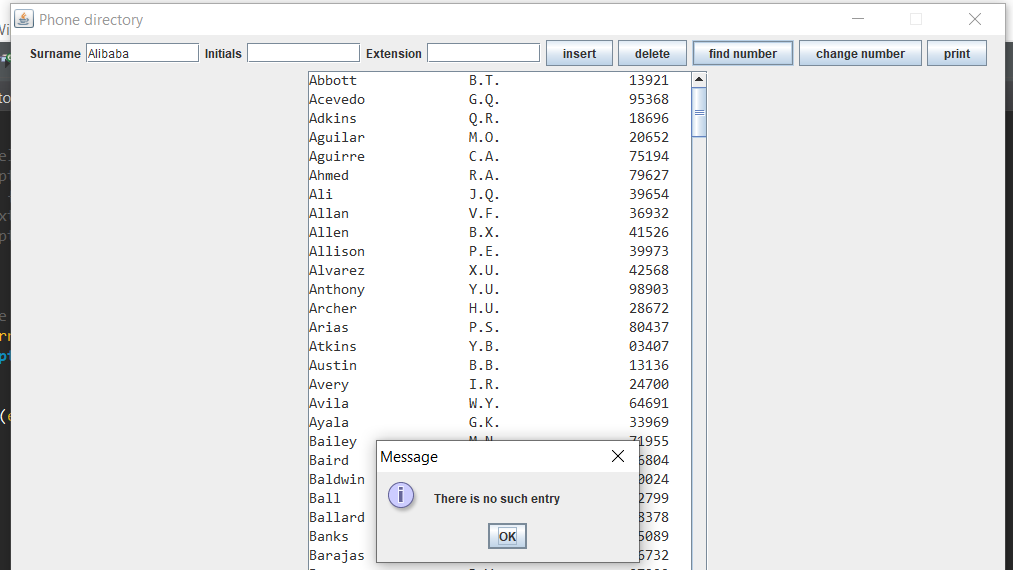
If valid number is entered but entry does not exist in the directory:  


Find number Testing

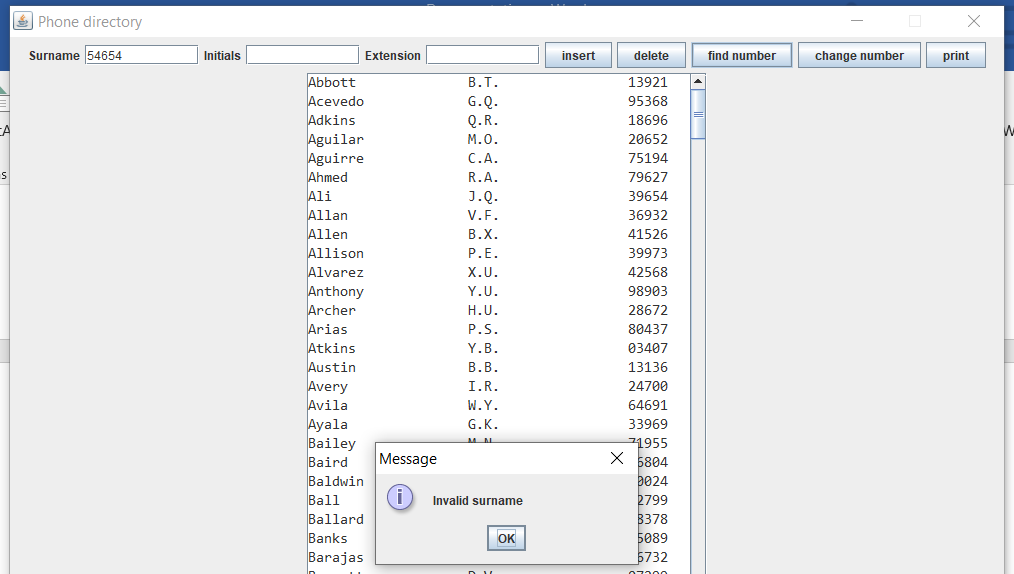
If valid surname, which exist in the directory, is entered, extension number is printed.



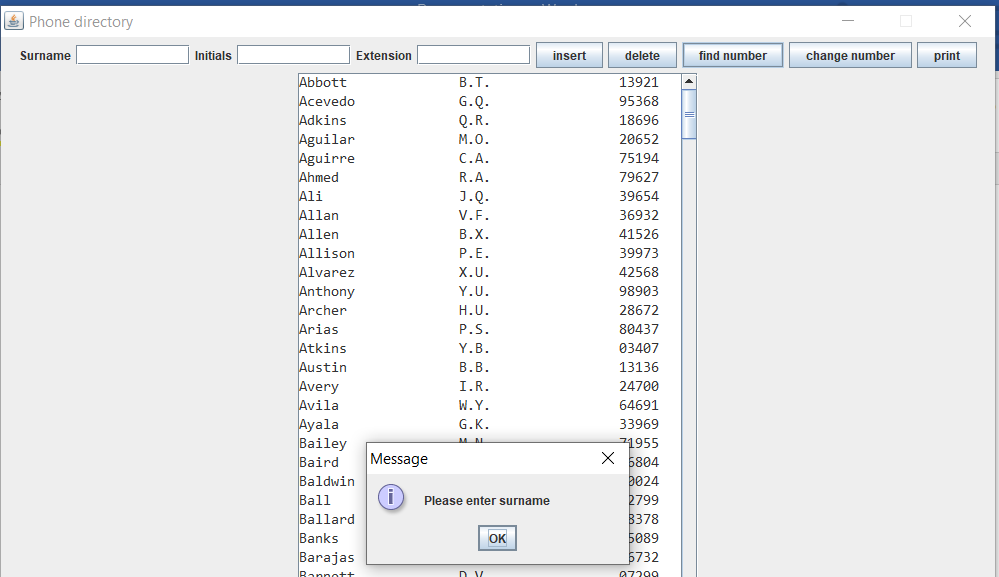
If valid surname, which does not exist in the directory, is entered:



If invalid surname is entered:

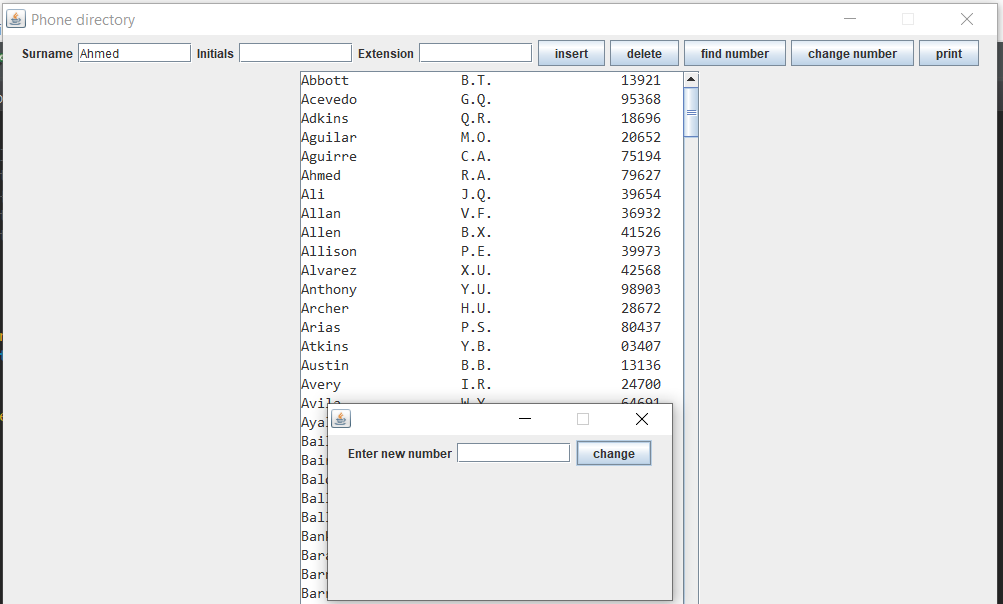


If no data is entered:

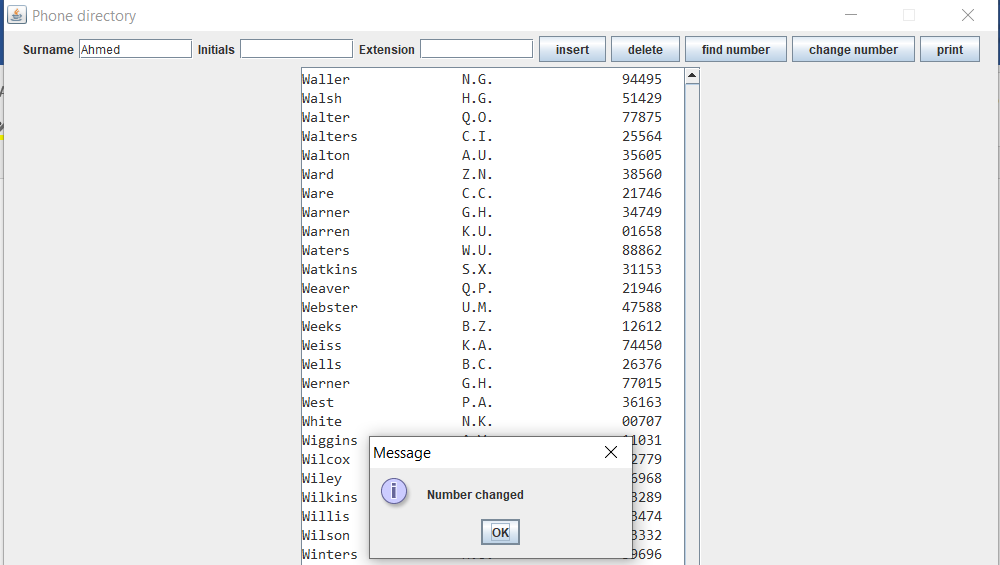


Change number Testing

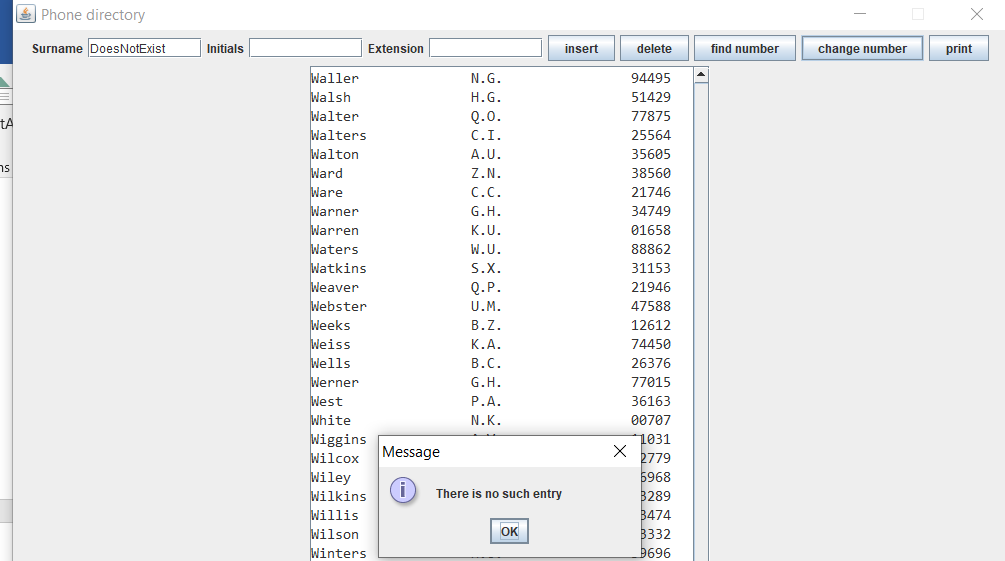
Number can be changed by entering the surname and clicking “change number” button. After button is clicked, new frame will appear and there will be the text field asking user to enter new number. Then user needs to click button change (the button is on the new frame). If number is successfully changed, user will be informed about it. If new number is invalid, program will ask user to re-enter the number which is valid.



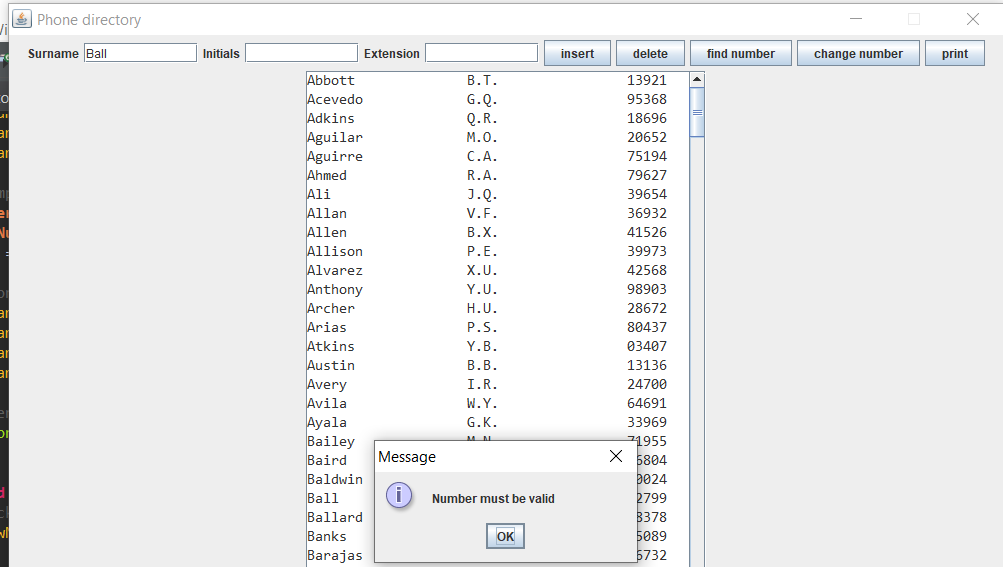
If surname exists in the directory and new number is valid format (only digits 0-9 allowed), number will be changed:



If surname does not exist in the directory:



If new number is invalid



Print Testing

The function of “print” button is to print the directory in the new frame. This button has no exception handling because it is hard to think of the scenario when clicking “print” button could potentially crash the program.

