Lab 6 in TND002

Mark Eric Dieckmann

February 22, 2023

Summary: You implement a phone book using a graphical user interface. You create the classes **Person** and **PhoneBook** that will be used by the class **GUI**. This class should provide the GUI with methods to search for persons, add persons, delete persons, load the phone book from the file *PhoneList.txt* and save it to a file with a name you type into the GUI.

1 Task: Implement and test Person and Phonebook

Each person has one *givenName*, one *surname*, and one phone number.

The instance variables are initialized with the constructor arguments.

The three *get..* methods return the surname, the full name (given name and surname separated by space) and the phone number.

Person
−givenName : String
−surname : String
−phoneNumber : int
+Person(String, String, int)
+getSurname() : String
+getFullName() : String
+getPhoneNumber() : int

Each line from the external file "PhoneList" defines one instance of **Person**. The persons are managed by an instance of **PhoneBook**, which stores all persons in a dynamic array and provides methods to manage them.

listOfNumbers stores all instances of **Person** that are listed in the phone book. It is initialized in the constructor.

load(arg) opens the file named *arg*, creates with every line in the file one instance of **Person**, adds it to *listOfNumbers* and closes the file. Use a try-catch block to prevent errors. Return "Phone book loaded" if it succeeded and "Try again" otherwise.

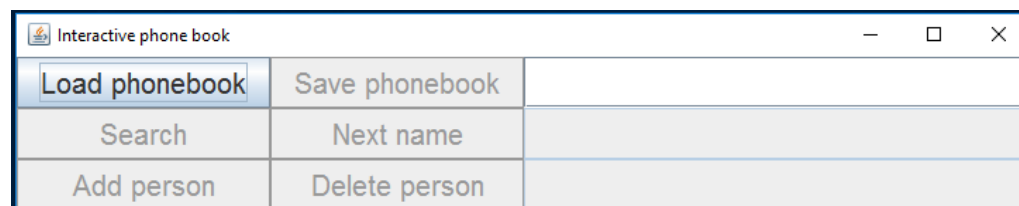| PhoneBook |
| --- |
| −listOfNumbers : ArrayList<Person> |
| +PhoneBook() |
| +load(String) : String |
| +search(String) : ArrayList<Person> |
| +deletePerson(String, int) : String |
| +addPerson(String, int) : boolean |
| +save(String) : String |

search(arg) checks if *listOfNumbers* contains persons specified by *arg*. *arg* can be the surname or the phone number. We create a dynamic array in the search method, add to it all the persons we found (several persons can have the same surname) and return it.

deletePerson(arg1, arg2) deletes a person that has the full name *arg1* and the phone number *arg2* (both must be given and match those of the person). You can use *search(arg)* with the phone number *arg* and compare the name of the returned person with *arg1*. The method returns either "The person/number does not exist" or "Person deleted".

addPerson(arg1, arg2) adds a person with the full name *arg1* and the phone number *arg2*. The person is not added if a person with that phone number already exists or if *arg1* does not contain exactly two words. It returns the boolean *true* if the person could be added and *false* if it could not be added.

save(arg) saves all persons in *listOfNumbers* in the external file named *arg* in the same format as *PhoneList*. Catch IO errors in this method. The full name is saved in a column of width 20 and the phone number in a column of width 5. *save(arg)* should return the string "Saved " + listOfNumbers.size() + " people to the file." if the saving succeeded. If the saving failed it should return the string "Could not save to the file".

2   Task: Create a graphical user interface (GUI)

Extend with **GUI** the class **JFrame** and create in *main()* an instance of **GUI**. Set the title "Interactive phone book" and define the font that you will use for all text in the GUI (buttons and text fields): plain sansserif with the font size 20. Create 6 buttons called load, save, search, next, add and delete and enable only the load button. Label the buttons as displayed in the figure. Create 3 text fields called searchField, nameField and numberField. Use a combination of gridlayout's to get a GUI that looks like the one above. The search field is the editable top text field, the name field is the middle one and the number field is the bottom text field. The size of the window is determined by *pack()*. Make the GUI visible and set the default close operation to "EXIT_ON_CLOSE".

3   Task: Connect the Phonebook and the GUI

Update **GUI** such that pressing either the active load button or 'return' in the search field extracts the string from the search field, clears the search field and calls with the extracted string the *load(arg)* method of **PhoneBook**. If a file with the name *arg* can not be opened, then the the text "Try again" should be written to the name field. Type in the name of the existing file ("PhoneList.txt") into the search field and press 'return'

or the load button. The content of the file is loaded into the phone book and the string "Phone book loaded" should be written to the name field. All buttons except the next button should be unlocked and the GUI should look like

Update **GUI** such that pressing the "Save phonebook" button reads the text string in the search field and clears the search field afterwards. If the text string is empty, then the text "Provide a file name" should be written to the name field. If the text string is not empty, call with it the save method of **Phonebook** and write its return value to the name field. Do not write the phone book to "PhoneList.txt", because you need this file and you may mess it up if your code is not working correctly.

Update the **GUI** such that pressing the "Search" button reads the string in the search field and clears the search field afterwards. The string *arg* that was read in is used to call the method *search(arg)* of **PhoneBook**. If the size of the dynamic array that is returned by the call is 0, then the text "Provide a name" should be written to the name field and "" to the number field. If the size of the dynamic array is 1, then the full name of the returned person is written to the name field and the phone number to the number field. If the size of the returned array list exceeds 1, then you enable the "Next name" button and set a person counter to 0. The next figure shows the case where you searched for the surname "Walker", which exists twice.

If you press the "Next name" button the value of the person counter should be increased by 1 and the GUI should display the next person in the dynamic array that was returned by the search. If you searched for Walker and pressed next once, your result should be that

When you reach the end of the dynamic array your person counter should be reset to zero and the "Next name" button should be locked again.

Update **GUI** such that the "Delete person" button removes the person with the name in the name field and the phone number in the number field. The return value from the delete method should be written into the search field. The result of pressing the "Delete person" button is shown in

Clicking the "Save phonebook" button should now save 49 people to the file.

Update **GUI** so that pressing the "Add person" button writes "Type in name and phone number" on the search field and enables the name field and the phone field. I typed in "This works" as the name of a person and a 5-digit phone number (we do not check that the length is 5).

After pressing the "Add person" button a second time, the add method of **PhoneBook** should be called with the name and phone number, the name and phone number should be removed from the text fields, the name field and the number field should no longer be editable and the text "Person added" should be written to the search field.