ITNP001 Principles and Practice of Programming
Practical Sheet 8: Telephone directory

For your practicals in week starting 16 November 2015

Remember to register your practical attendance in the way that you did before by double clicking on My Computer\Groups on Wide\ITNP001\Register

This example application maintains a simple telephone directory as an array of records. Your task is to understand the program and then to make various improvements. It will give you practice with strings, arrays and records.

Work with the initial version of the code in practical folder `V:\ITNP001\Practicals\Practical8\DirectoryInitial`. The files `Directory.java`, `DirectoryGUI.java` and `Entry.java` are already written. Only `PhoneBook.java` needs some work to finish the program.

1. You should enter names and phone numbers for a few people (e.g. Mary is 4136), clicking the Enter button after each one. (It would be a good idea to write down your sample names and numbers, so you can check the program is working properly.) Now enter a known name and click the Find button. This person's phone number should appear. At this stage, the Delete button doesn't do anything. In general, the status message in the *window title* tells you what just happened. Note that the directory entries are not stored permanently, so if you quit the program the information will have to be re-entered.

2. Just glance through `Directory.java` and `DirectoryGUI.java` since they are already written. The graphical parts of method `createGUI` should be familiar to you. This `DirectoryGUI` constructor finishes by creating an instance of `PhoneBook` to handle the list of telephone numbers. Note how `actionPerformed` calls methods of the `phoneBook` object when some of the buttons are clicked.

3. Now turn your attention to `Entry.java`. Class `Entry` describes a single phone directory entry. There are only `get` methods for fields since entries are created in their entirety.

4. Now turn your attention to `PhoneBook.java`. Class `PhoneBook` describes the directory as a whole with `size` entries in the array `directory`. Recall that an array is fixed in size, so what does it mean to use only part of an array? For the directory, this is done by maintaining the index of the `next` directory entry to use. Initially `next` is 0. After you add an entry, it becomes 1 and so on. At any point, the directory has been used from index 0 up to but not including index `next`. From `next` onwards, there is nothing useful in the directory. It follows that a loop to scan the directory must stop on reaching index `next`. You could think of next as being a 'water level' that tells you how far the directory 'bucket' has been filled. When you later consider deleting an entry from the directory, you will have to decrement `next` since there will be one less entry used.

5. Look at the methods `addEntry` and `findEntry`. Be sure you understand these before you start making changes.

   Why do you think that phone numbers are strings and not integers?

   There is a (deliberate!) problem in `addEntry`. Can you see what it is? This is mentioned later if you don't spot it immediately.

6. You may find it useful to inspect and explore the phone book storage using BlueJ's interactive debugger. Set a breakpoint somewhere useful, such as at the end of `addEntry` in `PhoneBook`, then explore the `next` and `directory` variables after adding an entry (then click **Continue** to resume execution).

7. `findEntry` looks for exactly the same name. What if the name in the directory is ‘Mary’ but the user asks for ‘mary’? Change `findEntry` so it does not care about upper or lower case letters. (Use `equalsIgnoreCase()`.) Check that using ‘mary’ finds the entry.

8. It would be even nicer if the full name was unnecessary when trying to find a user. So if the directory contains ‘Joseph Brown’, trying to find ‘Joseph’ or ‘Brown’ should both get the entry. Extend `findEntry` to look for the search string in only part of the directory name. (Convert both names to lower case with `toLowerCase` and then use `indexOf`.) Check that ‘Jo’, ‘Joseph’, ‘Brown’, and ‘Bro’ all find the entry.

9. `findEntry` doesn’t bother to check if the name entered is empty. Did this matter before? It matters now, *since an empty string is always part of any name!* If the name is empty (check for `length()` being zero) then do not check any of the directory entries; just return -1 as if the name was not found. Try finding an empty name field to make sure the request is unsuccessful.

10. The directory size is five entries. What happens when the directory becomes full? Satisfy yourself that a sixth entry is ignored.

11. Try making a directory entry where the name or the phone number field is empty. What happens? Change `addEntry` to not add the new entry and to return false if the name or the phone number is empty (or if the directory becomes full). Run the program and enter an empty name or phone number to make sure the request is unsuccessful.

Checkpoint Directory: Show a demonstrator your Directory application functioning correctly, and answer any questions.
Then continue…

12. You need to be able to delete a directory entry, perhaps because someone has moved house. When the Delete button is clicked, `actionPerformed` in `DirectoryGUI.java` calls `findEntry`. If the entry isn’t found, this is reported in the status message. If the entry is found, the name and phone number for this entry are displayed. Then the `deleteEntry` method is called; you need to complete this. For simplicity, this should ‘delete’ an entry by setting it to a new one with empty name and phone number strings. Run the program, make some entries, delete one of these and then try to find the name again; of course it should not be found.

13. `deleteEntry` isn’t ideal as a deleted entry’s space in the array is now ‘lost’ and cannot be re-used. One solution is to move all later entries down to where the entry was deleted. So if entry 2 is deleted, copy entry 3 to 2, 4 to 3, etc. Stop when you have copied the last entry (i.e. the one numbered `next-1`). Write a loop that achieves this. Finally, decrement `next` since the deletion has freed up the last place in the directory.

    A nasty thought: will the copying procedure work if the first entry, only entry, or last entry in the directory is deleted? Think it through and try testing various cases. (These kinds of cases are easy to get wrong, and always need to be tested carefully.)

SBJ November 2015