Project document

Yann Jorelle 652034, 1st year student Computer Science bachelor student – 24.4.2019

General description

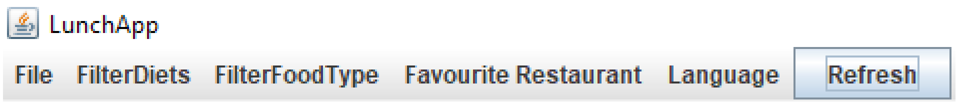
I've created a lunch app that gathers the daily menus of student restaurants in Otaniemi and displays them in a centralized manner. I completed the project on the intermediate difficulty level (with a GUI).

User interface

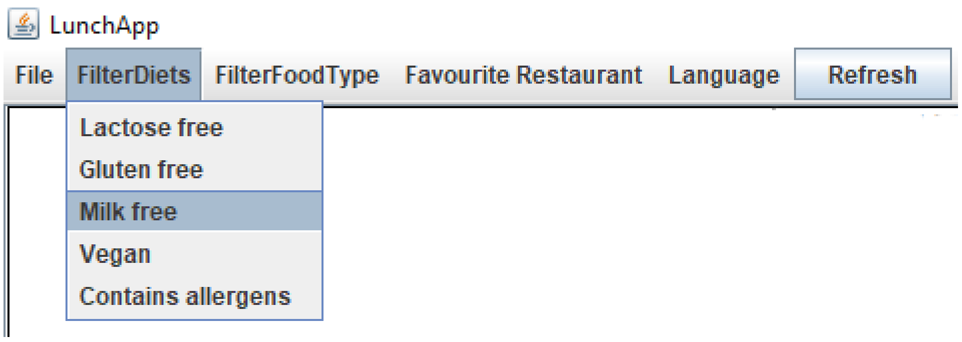
You can start the program by running the LunchApp.scala file (found in src/LunchApp). You will then be greeted by a grid table that lists the menus of each of the restaurants. Make sure to maximize the window so you can see all the scrollbars. Up in the top of the interface there are some options. You can filter the menus so that the app only shows menus that respect your diet or allergens. You can also filter the menus so that they only show the ones containing some food type, for example chicken, fish etc. You also have the option to set a favorite restaurant, so that the next time you open the app, that restaurant will have a pink background indicating that it is your favorite. It is also possible to change the language from Finnish to English or vice versa. The app will also remember this the next time you launch it. Finally there is a refresh button, for refreshing the menus.

Here are some screenshots to get a better idea:

The top panel:



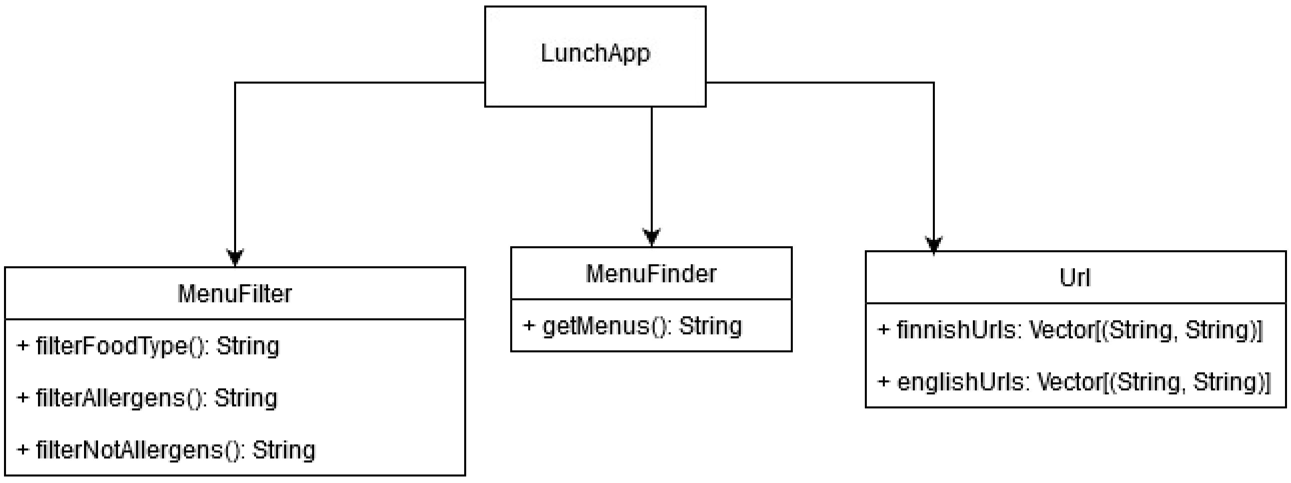
Example of how to filter:



Program structure

The program is split into three main sub-parts. The LunchApp class is the user interface. It uses classes MenuFinder and MenuFilter to get the menus from the restaurants' websites and filter the menus respectively. There is also a URL object which contains all of the restaurants' URLS. I could technically have implemented all of the functionality in one single class but I think the way I've done it improves readability, by making different classes responsible for different domains of the program.

UML diagram:



LunchApp
MenuFilter
+ filterFoodType(): String
+ filterAllergens(): String
+ filterNotAllergens(): String
MenuFinder
+ getMenus(): String
Url
+ finnishUrls: Vector[(String, String)]
+ englishUrls: Vector[(String, String)]

The MenuFinder class includes the key method getMenus which can be used to get the menus from the restaurants' websites. The MenuFilter class has key methods filterFoodType, filterAllergens and filterNotAllergens which are all used for filtering the menus based on different criteria.

Algorithms

The MenuFinder class makes use of the json4s library to parse the json data from the restaurants' websites. It then does some string manipulation to make the menus more readable. The MenuFilter class looks through the menus and looks for specific elements to then filter the menus accordingly. For example the diet info of the menus (lactose free, gluten free, vegan etc.) are always found in a parenthesis after the menus, which makes it quite easy to filter them. For filtering based on food type, it just looks at the menus that contain any of some words like fish, trout, salmon etc. and then filters them accordingly.

Data structures

I only used Scala's library data structures in this program, like Arrays, Buffers, Vectors and Sets. I didn't think too much about which data structures to use, as I don't think it bears much relevance in this case. Maybe you could make the program slightly more efficient with more thoughtful choices of data structures though.

Files and network access

The program uses some text files to store the favorite restaurant of the user and his/her language preferences. The data is presented in string format, for example "Täffä" to set the favorite restaurant to Täffä or "FI" to set the language to Finnish. The favorite restaurant is stored in the favRes.txt file and the language info in language.txt.

Testing

Since the program is relatively simple, the testing was carried out by manually testing all the essential methods. I tested the filter methods to make sure they worked and discovered a few bugs along the way. I also made sure no errors occurred while retrieving the menus (for example no internet connection) by using a try – catch clause. I also noticed while testing that sometimes the menus had extra line breaks and tabs, which made them look weird. This is something I fixed later.

Known bugs and missing features

One bug that I noticed while testing has to do with filtering based on food type. For example if you want to filter by fish, the app will look for menus containing the word "kala" among others. Of course this means it accepts words like "rans**kala**iset" which it is not supposed to. Another bug is that after choosing your favorite restaurant, the GUI doesn't update the restaurant's background to pink until you have restarted the app. Unfortunately I couldn't come up with a solution to this and I generally had trouble getting the GUI to update the menu texts (my solution is not very elegant).

One missing feature is to be able to show the menus for a whole week. I didn’t think about this feature early enough but it would certainly be doable. The Fazer restaurants’ urls only give menus for the remaining days of the week, which makes showing weekly menus more challenging. You could maybe get the menus in the beginning of the week and then save them into text files or something like that.

Strengths and weaknesses

I don't know of any particular strengths, however one extra feature is the choice of language. Other than the weaknesses discussed in the bugs and missing features section I can’t come up with any other major weaknesses.

Deviations from plan, realized process and schedule

I pretty much followed my initial plan regarding the order of implementation. I went a bit over my time estimates in some parts, but that was to be expected, because the time estimates were quite rough.

Final evaluation

All in all I am quite satisfied with how the program turned out. Of course there is still quite a lot of room for improvement. For example adding the option to see a whole weeks’ menus. You could also improve the app by making the GUI look nicer and easier to use (it looks quite bland right now). The code that deals with how the GUI reacts to changes could also be improved.
I think the choice of methods, data structures and class structure works quite fine for the purposes of this program. The structure makes it quite easy to add new restaurants for example.
If I started the project again from the beginning I would try to think a little more and make a good solution on the first try, instead of just writing something and then changing it 100 times after.

References

I used the json4s library to parse the json data from the websites. I also used the API for the Scala swing library to understand how it works. The json4s library's jars can be found in the projects' "libs" folder.

Links can be found here:

Json4s: http://json4s.org/ or https://github.com/json4s/json4s (github link)
Scala swing API: https://www.scala-lang.org/api/2.12.8/scala-swing/scala/swing/