Angeline Oktaviana Eka Pratiwi Jayanegara (706977)
14 February 2020

Technical plan

1. **Class structure**

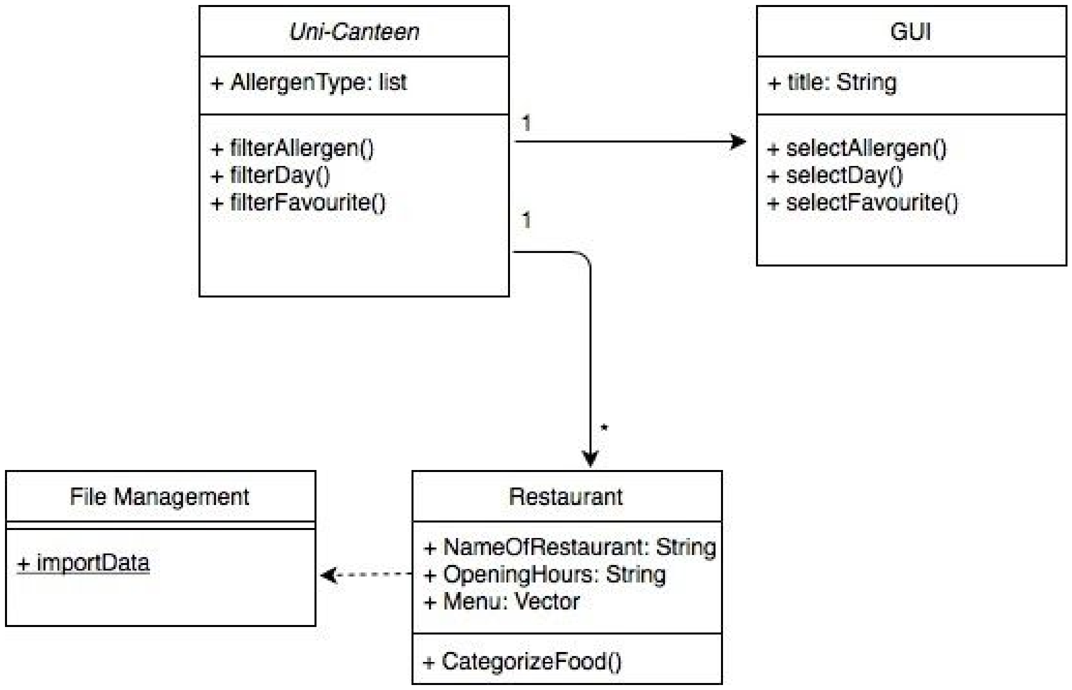


Figure 1. UML of Uni-Canteen

Uni-Canteen is the main class of the program. All of the important functions are run here. The method *filterAllergen* takes input from the user and filters the menu from the *Restaurant*. It should return the list of menu which the user has chosen to filter out as a parameter, for example, if the user chooses lactose free, the method will then filter out all the food with lactose and returns the list of lactose free menu, *filterDay* takes a parameter from the user and it should filter the menu on the day selected and return only the daily menu, *filterFavourite* takes parameter from the user, and it should adjust the position of the lists, so that it returns a new list with favoured restaurants in the beginning.

File Management class is where the data is processed and imported. After the importing is successful, it creates a restaurant object with the list of menus, which is then connected to the *Restaurant* class.

In *Restaurant* class each menu is then categorized if they are lactose free, including allergen, gluten free, or vegetarian depending on the list from the imported file.

At last *GUI* will be implemented and mainly connected to the main class, *Uni-Canteen.* Users will be able to click buttons for the day or check the tick boxes for filtering the Allergens and choose their favorite restaurants.

2. **Use case description**

   The user opens the program and is shown today's lunch list from a few student restaurants. There will be choices of other days' lunch lists, according to dates in a week(like in Kanttiinit), or for a shorter period of time, yesterday, today, and tomorrow (as shown as in GUI figure example in general plan). The user then is able to navigate the other days' list and is shown the menu on the particular day he has chosen. There will also be choices for filtering the food according to the allergen; lactose free, gluten free, and vegan. The user could choose which filter he wants to add and the list should be updated with a new list of menu according to the filter. Favourite option is an available choice in the program as well. The favourite restaurant will be shown before the other restaurants. There will be information about menu items that contain certain allergen.

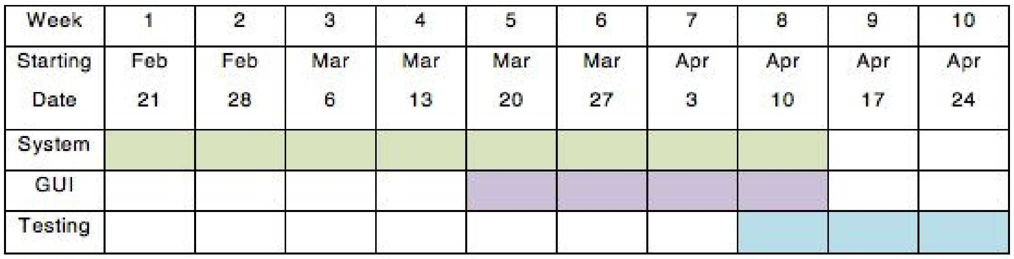
3. **Algorithms**

   Filtering allergies and favourite will be done using the Scala filter method from the raw lunch menu data, using predicate (input from the user) to filter raw collection elements, the filter then evaluate that element, and return true to keep the element in the new collection, or false to filter it out. Then it will be assigned to a new variable. The other algorithm such as binary search could be considered if the filter takes longer time than expected, or if there is more data added.

4. **Data structures**

   Data will be obtained using JSON from the sources available for the restaurants menus. The data of the lunch list is stored in *Mutable Buffers,* which is an easy structure for handling mutable collections, or if the data is immutable lists of units, they will be stored in *Vectors*. *Array* could be another option as well. If the GUI is implemented, the logo of restaurants, and other images will be in jpeg or png format.

5. **Schedule**



Week	1	2	3	4	5	6	7	8	9	10
Starting Date	Feb 21	Feb 28	Mar 6	Mar 13	Mar 20	Mar 27	Apr 3	Apr 10	Apr 17	Apr 24
System	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		
GUI					🟪	🟪	🟪	🟪		
Testing								🟦	🟦	🟦

Rough estimation of the working hours to implement the program(system) will be 1-2 hours per day during the weekday (5-10 hours per week). If the system is successfully implemented after 4 - 5 weeks, then GUI is going to be designed and implemented. The last 3 weeks of the project will be spent on testing and trying the finalized project.

6. **Unit testing plan**

First test will be on reading/importing the file and loading the data. If the text file has some missing data, the program should ignore and still create the lists. Project is also tested by comparing whether the data provided and the lunch list on the app are synchronised. The method *filterAllergen* should return the list of menu which the user has chosen to filter out as a parameter, for example, if the user choose lactose free, the method will then filter out all the food with lactose and returns the list of lactose free menu, *filterDay* should filter the menu on the day selected and return only the daily menu, *filterFavourite* should adjust the position of the lists, so that it returns a new list with favoured restaurants in the beginning. The app should provide the correct information as well, when the user gives correct inputs from the additional features. The GUI will be tested by trying to run the program and see if it works accordingly.

7. **References and links**

- Alvinalexander.com. (2020). *How to use the 'filter' method to filter a Scala collection | alvinalexander.com*. [online] Available at: https://alvinalexander.com/scala/how-to-use-filter-method-scala-collections-cookbook [Accessed 14 Feb. 2020].
- Anon, (2020). [ebook] Available at: https://cordis.europa.eu/docs/projects/cnect/5/215455/080/deliverables/ROADIDEA-D3-1-Data-filtering-methods-V1-1.pdf [Accessed 14 Feb. 2020].
- Kanttiinit.fi. (2020). *Kanttiinit*. [online] Available at: https://kanttiinit.fi [Accessed 14 Feb. 2020].