Programming 2: Studio
Lunch List Application, Project Document

1. Personal Information

Jeheon Kim (716054), Data Science second year
Last modified date of the document: 26th of April 2020

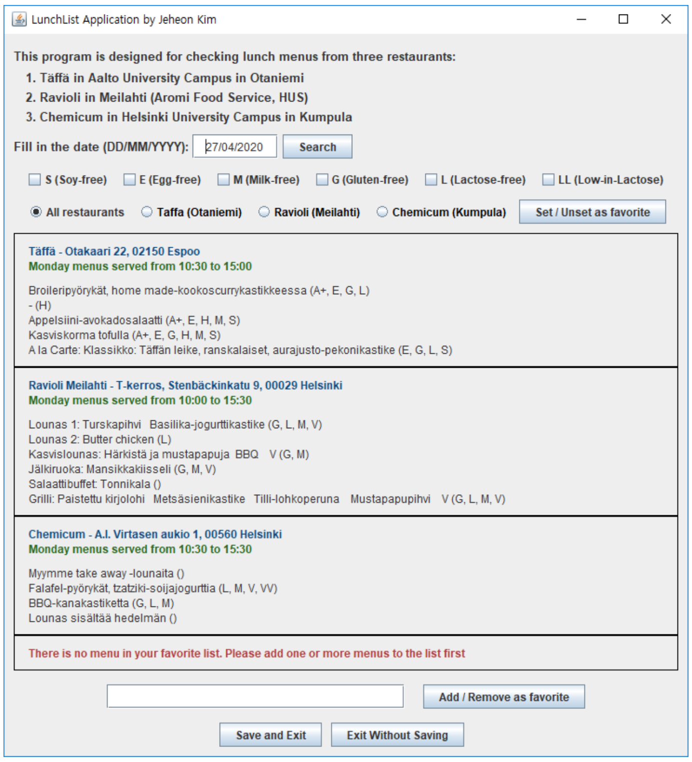
2. General Description

The project is lunch list application which provides menu information of three student restaurants. The initial idea was to cover only the restaurants in the Otaniemi campus, but due to ongoing virus situation, three restaurants are selected: Täffä (Otaniemi), Ravioli (Meilhati), and Chemicum (Kumpula), regardless of their location. It isn't difficult at all to include more restaurants, but for the conciseness of the project, it covers aforementioned restaurants only.

The user can check menus of three restaurant on any date. (As long as the restaurant provides / keep the JSON data) The program also allows users to filter allergens or specific restaurant based on their preference. Also, it is possible to save their preference (restaurant, menu or ingredient). The program will remember it and be provides the result based on the saved preference.

Based on the given information the project is at the Intermediate level (the highest possible of the lunch list project) which requires the Graphical User Interface to be implemented. The project also implemented all three additional features: "Filter menu based on allergen", "Support for choice of favorite restaurants", and "Inform user about menu items that contain a dish or a component (like fish) that the user has reported to like".

3. User Interface


LunchList Application by Jeheon Kim

This program is designed for checking lunch menus from three restaurants:
1. Täffä in Aalto University Campus in Otaniemi
2. Ravioli in Meilahti (Aromi Food Service, HUS)
3. Chemicum in Helsinki University Campus in Kumpula

Fill in the date (DD/MM/YYYY): 27/04/2020

Täffä - Otakaari 22, 02150 Espoo
Monday menus served from 10:30 to 15:00

Broileripyörykät, home made-kookoscurrykastikkeessa (A+, E, G, L)
- (H)
Appelsiini-avokadosalaatti (A+, E, H, M, S)
Kasviskorma tofulla (A+, E, G, H, M, S)
A la Carte: Klassikko: Täffän leike, ranskalaiset, aurajusto-pekonikastike (E, G, L, S)

Ravioli Meilahti - T-kerros, Stenbäckinkatu 9, 00029 Helsinki
Monday menus served from 10:00 to 15:30

Lounas 1: Turskapihvi Basilika-jogurttikastike (G, L, M, V)
Lounas 2: Butter chicken (L)
Kasvislounas: Härkistä ja mustapapuja BBQ V (G, M)
Jälkiruoka: Mansikkakiisseli (G, M, V)
Salaattibuffet: Tonnikala ()
Grilli: Paistettu kirjolohi Metsäsienikastike Tilli-lohkoperuna Mustapapupihvi V (G, L, M, V)

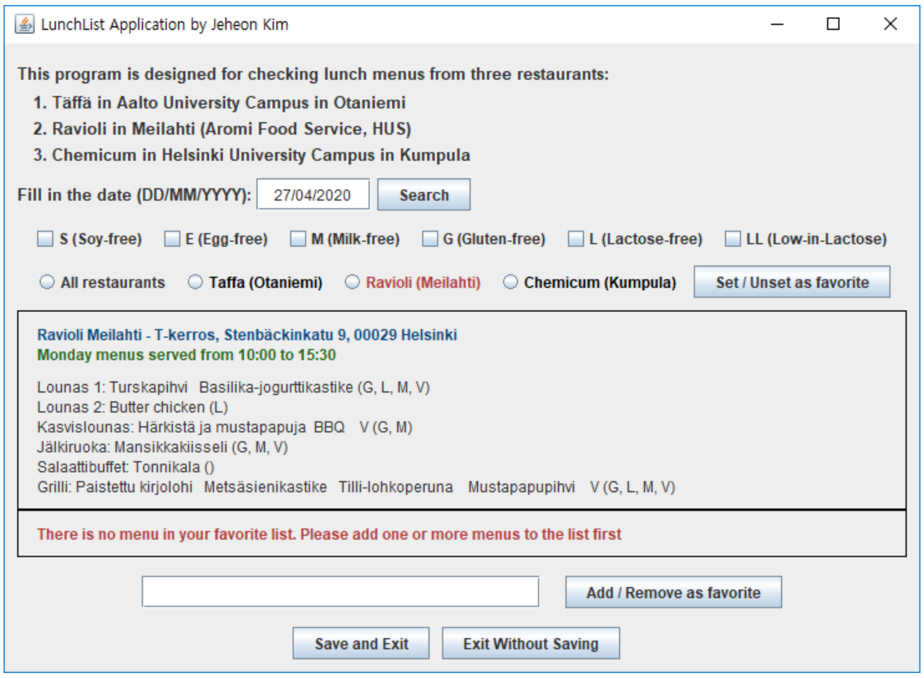
Chemicum - A.I. Virtasen aukio 1, 00560 Helsinki
Monday menus served from 10:30 to 15:30

Myymme take away -lounaita ()
Falafel-pyörykät, tzatziki-soijajogurttia (L, M, V, VV)
BBQ-kanakastiketta (G, L, M)
Lounas sisältää hedelmän ()

There is no menu in your favorite list. Please add one or more menus to the list first

Figure 1. Program's starting screen with default setting

When program started, current date menus of all three restaurants are displayed as default. However, it can be changed by adding one or more restaurants to the favorite list. Then the program will display only the menus of favorite restaurants when started as following:



This program is designed for checking lunch menus from three restaurants:
1. Täffä in Aalto University Campus in Otaniemi
2. Ravioli in Meilahti (Aromi Food Service, HUS)
3. Chemicum in Helsinki University Campus in Kumpula

Fill in the date (DD/MM/YYYY): 27/04/2020

Ravioli Meilahti - T-kerros, Stenbäckinkatu 9, 00029 Helsinki
Monday menus served from 10:00 to 15:30

Lounas 1: Turskapihvi Basilika-jogurttikastike (G, L, M, V)
Lounas 2: Butter chicken (L)
Kasvislounas: Härkistä ja mustapapuja BBQ V (G, M)
Jälkiruoka: Mansikkakiisseli (G, M, V)
Salaattibuffet: Tonnikala ()
Grilli: Paistettu kirjolohi Metsäsienikastike Tilli-lohkoperuna Mustapapupihvi V (G, L, M, V)

There is no menu in your favorite list. Please add one or more menus to the list first

Figure 2. Starting screen of the program with favorite restaurant setting

The figure 2 above is the case that the restaurant "Ravioli" is selected and saved as favorite restaurant. Unlike the figure 1 with default setting, it shows the menus of the user's favorite restaurant "Ravioli" only. Also, none of the restaurants (RadioButton) is selected and the restaurant name "Ravioli (Meilahti)" is colored in red. (Indicating that it is selected as favorite)

To search the menus of different date, fill in the date field with any date (In the format of DD/MM/YYYY) and click "Search" button. Because of the Swing's error with the "editDone" function, searching by hitting the "Enter" key is temporarily not supported. (It is possible to make it work by uncomment the "editDone" part in the "UI.scala", line 324 ~ 334.

Exceptions will be thrown if the input date is wrong in following cases:
1. The number digit is incorrect (ex: year '20201')

→ Day and Month should be 2 digits, and the Year should be 4 digits

2. Dates are divided by different symbol than "/" (ex: 26-04-2020)
3. Input date is incorrect (For example, month 13 or day 35)

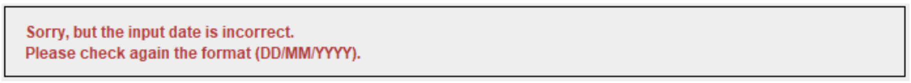
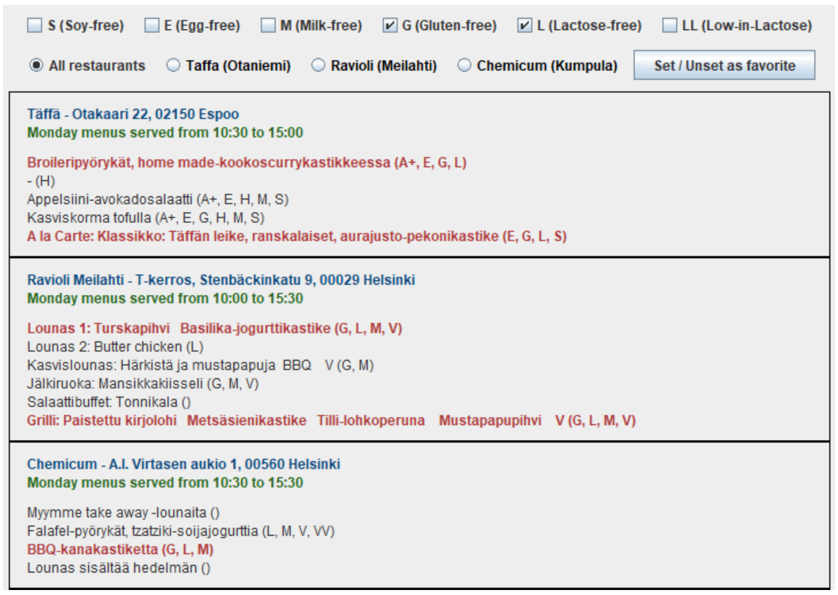

Sorry, but the input date is incorrect.
Please check again the format (DD/MM/YYYY).

Figure 3. Error text by the thrown exception from the wrong input date

The program allows users to filter allergens by marking the checkbox of each allergen and clicking the "Search" button to refresh the screen. Menus that satisfy the selection of allergens will be indicated by red color. For the simplicity of the demonstration, the program supports only 6 most common and important allergens. (Soy-free, Egg-free, Milk-free, Gluten-free, Lactose-free, and Low-in-Lactose) Because all three restaurants belong to different food service company, there was a great variation of the allergens and had to be narrowed down.


☐ S (Soy-free) ☐ E (Egg-free) ☐ M (Milk-free) ☑ G (Gluten-free) ☑ L (Lactose-free) ☐ LL (Low-in-Lactose)
● All restaurants ○ Taffa (Otaniemi) ○ Ravioli (Meilahti) ○ Chemicum (Kumpula)
Set / Unset as favorite

Täffä - Otakaari 22, 02150 Espoo
Monday menus served from 10:30 to 15:00

Broileripyörykät, home made-kookoscurrykastikkeessa (A+, E, G, L)
- (H)
Appelsiini-avokadosalaatti (A+, E, H, M, S)
Kasviskorma tofulla (A+, E, G, H, M, S)
A la Carte: Klassikko: Täffän leike, ranskalaiset, aurajusto-pekonikastike (E, G, L, S)

Ravioli Meilahti - T-kerros, Stenbäckinkatu 9, 00029 Helsinki
Monday menus served from 10:00 to 15:30

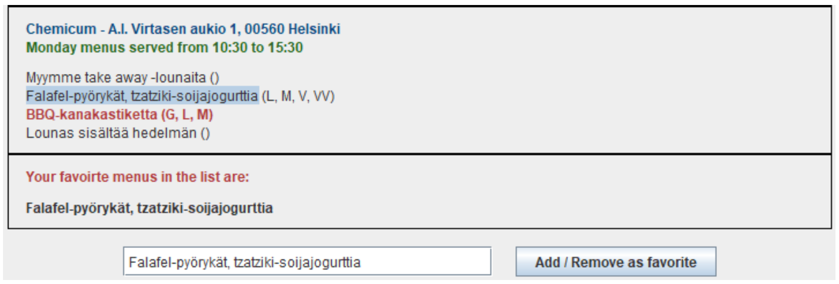
Lounas 1: Turskapihvi Basilika-jogurttikastike (G, L, M, V)
Lounas 2: Butter chicken (L)
Kasvislounas: Härkistä ja mustapapuja BBQ V (G, M)
Jälkiruoka: Mansikkakiisseli (G, M, V)
Salaattibuffet: Tonnikala ()
Grilli: Paistettu kirjolohi Metsäsienikastike Tilli-lohkoperuna Mustapapupihvi V (G, L, M, V)

Chemicum - A.I. Virtasen aukio 1, 00560 Helsinki
Monday menus served from 10:30 to 15:30

Myymme take away -lounaita ()
Falafel-pyörykät, tzatziki-soijajogurttia (L, M, V, VV)
BBQ-kanakastiketta (G, L, M)
Lounas sisältää hedelmän ()

Figure 4. Screen in which menus are filtered based on the selected allergens

Figure 4 shows the case where "Gluten-free" and "Lactose-free" are selected as preferred allergens. As above, only the menus that satisfy all selected allergens will be filtered if more than one allergen is selected. (Conjunctive addition)


Chemicum - A.I. Virtasen aukio 1, 00560 Helsinki
Monday menus served from 10:30 to 15:30

Myymme take away -lounaita ()
Falafel-pyörykät, tzatziki-soijajogurttia (L, M, V, VV)
BBQ-kanakastiketta (G, L, M)
Lounas sisältää hedelmän ()

Your favoirte menus in the list are:

Falafel-pyörykät, tzatziki-soijajogurttia

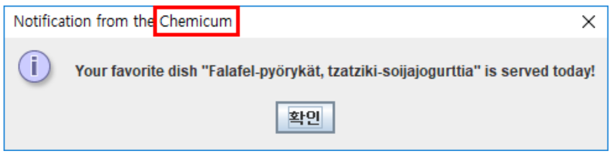
Falafel-pyörykät, tzatziki-soijajogurttia

Figure 5. Adding a menu (partial or whole name) to the favorite menu list

The program also allows users to select and add menus to their favorite menu list. Like the Figure 5 Above, users can easily drag, copy, and paste the menu name and add it to the list. (The input string will be lower-cased and trimmed, therefore the input is white-space insensitive and case insensitive) Then, the program will remember their preference and notify them (by pop-up dialog) when the added menu is served on the current date. For the conciseness of the project, the pop-up dialog is activated on the current date menu only.

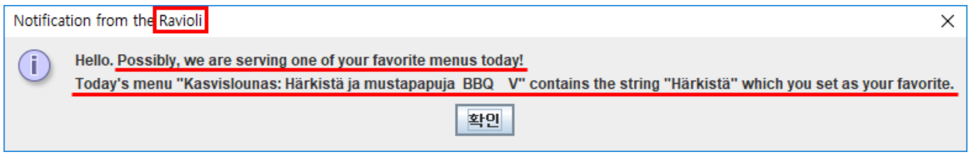
If today's menus match one or more of users' favorite menus (String), the pop-up dialog will be activated when program started like Figure 6 and 7 below. The restaurant that serves the matching menu will be indicated by the restaurant name in the title of the dialog.

The pop-up dialog has two cases where the added menu name matches exactly the name of the menu served on the day (Figure 6), and where the added menu name partially matches the name of the menu served on the day (Figure 7).


Notification from the Chemicum

Your favorite dish "Falafel-pyörykät, tzatziki-soijajogurttia" is served today!

Figure 6. Pop-up notification case 1

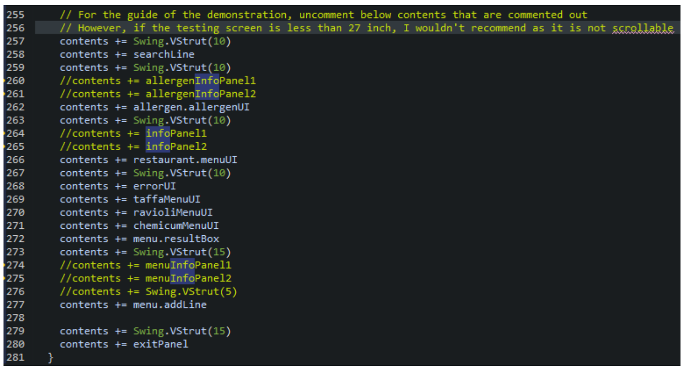

Notification from the Ravioli

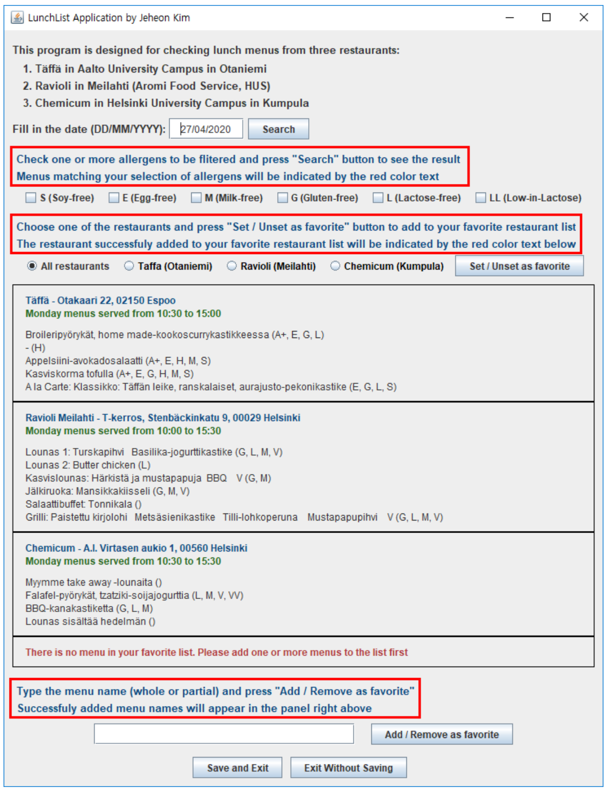
Hello. Possibly, we are serving one of your favorite menus today!
Today's menu "Kasvislounas: Härkistä ja mustapapuja BBQ   V" contains the string "Härkistä" which you set as your favorite.

Figure 7. Figure 6. Pop-up notification case 2

Also, there are hidden info panels to guide the teaching assistant to test the program. They are commented out as default due to the size of the program (The program is not scrollable because it was initially designed for two restaurants which requires small size. Instead, the program is designed to be responsive [resize automatically based on the size of contents], and it will be small for monitors with size less than 27 inches when all info-panels are activated. For this reason, the author recommends teaching assistants to activates info-panels only if the testing monitor is, at least, 27 inches. Otherwise, reading through this document is recommended)

To activate the info-panels, please uncomment the contents in the UI.scala: line 260, 261, 264, 265, 274, 275. Then, all info-panels will be displayed in the program like Figure 9 below.


```
255     // For the guide of the demonstration, uncomment below contents that are commented out
256     // However, if the testing screen is less than 27 inch, I wouldn't recommend as it is not scrollable
257     contents += Swing.VStrut(10)
258     contents += searchLine
259     contents += Swing.VStrut(10)
260     //contents += allergenInfoPanel1
261     //contents += allergenInfoPanel2
262     contents += allergen.allergenUI
263     contents += Swing.VStrut(10)
264     //contents += infoPanel1
265     //contents += infoPanel2
266     contents += restaurant.menuUI
267     contents += Swing.VStrut(10)
268     contents += errorUI
269     contents += taffaMenuUI
270     contents += ravioliMenuUI
271     contents += chemicumMenuUI
272     contents += menu.resultBox
273     contents += Swing.VStrut(15)
274     //contents += menuInfoPanel1
275     //contents += menuInfoPanel2
276     //contents += Swing.VStrut(5)
277     contents += menu.addLine
278
279     contents += Swing.VStrut(15)
280     contents += exitPanel
281   }
```
Figure 8. Info-panels that are commented out for the size of the GUI


LunchList Application by Jeheon Kim

This program is designed for checking lunch menus from three restaurants:
1. Täffä in Aalto University Campus in Otaniemi
2. Ravioli in Meilahti (Aromi Food Service, HUS)
3. Chemicum in Helsinki University Campus in Kumpula

Fill in the date (DD/MM/YYYY): 27/04/2020

Check one or more allergens to be flitered and press "Search" button to see the result
Menus matching your selection of allergens will be indicated by the red color text

☐ S (Soy-free) ☐ E (Egg-free) ☐ M (Milk-free) ☐ G (Gluten-free) ☐ L (Lactose-free) ☐ LL (Low-in-Lactose)

Choose one of the restaurants and press "Set / Unset as favorite" button to add to your favorite restaurant list
The restaurant successfuly added to your favorite restaurant list will be indicated by the red color text below

[✓ All restaurants] Taffa (Otaniemi) Ravioli (Meilahti) Chemicum (Kumpula)

Täffä - Otakaari 22, 02150 Espoo
Monday menus served from 10:30 to 15:00

Broileripyörykät, home made-kookoscurrykastikkeessa (A+, E, G, L)
- (H)
Appelsiini-avokadosalaatti (A+, E, H, M, S)
Kasviskorma tofulla (A+, E, G, H, M, S)
A la Carte: Klassikko: Täffän leike, ranskalaiset, aurajusto-pekonikastike (E, G, L, S)

Ravioli Meilahti - T-kerros, Stenbäckinkatu 9, 00029 Helsinki
Monday menus served from 10:00 to 15:30

Lounas 1: Turskapihvi   Basilika-jogurttikastike (G, L, M, V)
Lounas 2: Butter chicken (L)
Kasvislounas: Härkistä ja mustapapuja BBQ   V (G, M)
Jälkiruoka: Mansikkakiisseli (G, M, V)
Salaattibuffet: Tonnikala ()
Grilli: Paistettu kirjolohi   Metsäsienikastike   Tilli-lohkoperuna   Mustapapupihvi   V (G, L, M, V)

Chemicum - A.I. Virtasen aukio 1, 00560 Helsinki
Monday menus served from 10:30 to 15:30

Myymme take away -lounaita ()
Falafel-pyörykät, tzatziki-soijajogurttia (L, M, V, VV)
BBQ-kanakastiketta (G, L, M)
Lounas sisältää hedelmän ()

There is no menu in your favorite list. Please add one or more menus to the list first

Type the menu name (whole or partial) and press "Add / Remove as favorite"
Successfuly added menu names will appear in the panel right above

Figure 9. The GUI with all info-panels designed to guide the testing

4. Program Structure

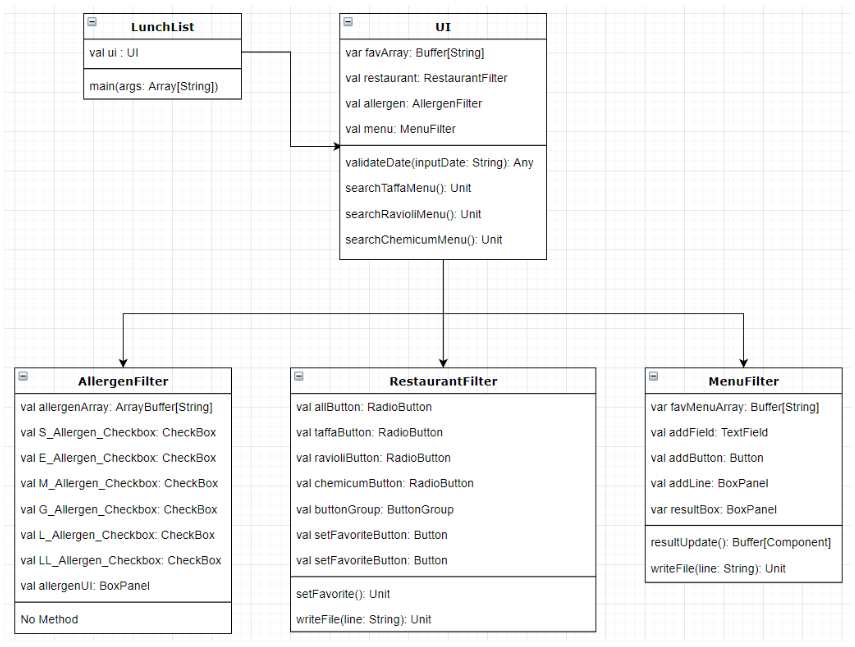

LunchList
val ui : UI
main(args: Array[String])
UI
var favArray: Buffer[String]
val restaurant: RestaurantFilter
val allergen: AllergenFilter
val menu: MenuFilter
validateDate(inputDate: String): Any
searchTaffaMenu(): Unit
searchRavioliMenu(): Unit
searchChemicumMenu(): Unit
AllergenFilter
val allergenArray: ArrayBuffer[String]
val S_Allergen_Checkbox: CheckBox
val E_Allergen_Checkbox: CheckBox
val M_Allergen_Checkbox: CheckBox
val G_Allergen_Checkbox: CheckBox
val L_Allergen_Checkbox: CheckBox
val LL_Allergen_Checkbox: CheckBox
val allergenUI: BoxPanel
No Method
RestaurantFilter
val allButton: RadioButton
val taffaButton: RadioButton
val ravioliButton: RadioButton
val chemicumButton: RadioButton
val buttonGroup: ButtonGroup
val setFavoriteButton: Button
val setFavoriteButton: Button
setFavorite(): Unit
writeFile(line: String): Unit
MenuFilter
var favMenuArray: Buffer[String]
val addField: TextField
val addButton: Button
val addLine: BoxPanel
var resultBox: BoxPanel
resultUpdate(): Buffer[Component]
writeFile(line: String): Unit

Figure 10. UML Class Diagram of the project

The class "LunchList" and "UI" are the main object and class of this project. Rest are for the additional features which filters the data and shows the result based on users' preference.

As mentioned, the class "DateCheck" and "JSON_Parsing" are not activated due to Exception in thread "main" java.lang.StackOverflowError, and are not included in the UML diagram above. The author left those classes (commented out) on purpose because they are errors that were faced during the project. All contents of these classes are included in the UI class instead.

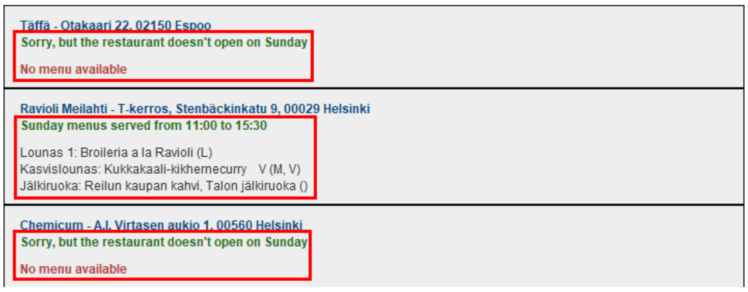
Unfortunately, the resulting classes are a lot different from the original planning due to the sudden change of the API (from Fazer and Sodex to Kanttiinit) and included restaurants.

5. Algorithms

This chapter will cover only main algorithms that are used the features.

✓ Restaurant Menu JSON Parsing

: The program receives the date from the text field and the input date is validated by the function "validateDate". The "validateDate" function checks whether the input date is in format of "DD/MM/YYYY". As mentioned earlier, it checks whether the date really exists (ex: Month 13 will be denied), whether each number is divided by the suggested symbol "/", and lastly whether each number have correct number of digits. (Day: 2 digits, Month: 2 digits, Year: 4 digits). Then the program will receive JSON data from the Kanttiinit API based on the input date. The program will check the day, opening hours, and whether the menu exists or not. Then it will parse the result to GUI based on the filtered data. (ex: Opening hours of the restaurant will be changed based on day; Notification will be displayed if the menu data is empty) JSON parsing functions are divided into three restaurants and therefore the program allows users to choose specific restaurant they would like to see menus from. (Also, it is possible to see all menus from thee restaurants, and all three functions are called in this case)



Täffä - Otakaari 22, 02150 Espoo
Sorry, but the restaurant doesn't open on Sunday

No menu available

Ravioli Meilahti - T-kerros, Stenbäckinkatu 9, 00029 Helsinki
Sunday menus served from 11:00 to 15:30

Lounas 1: Broileria a la Ravioli (L)
Kasvislounas: Kukkakaali-kikhernecurry V (M, V)
Jälkiruoka: Reilun kaupan kahvi, Talon jälkiruoka ()

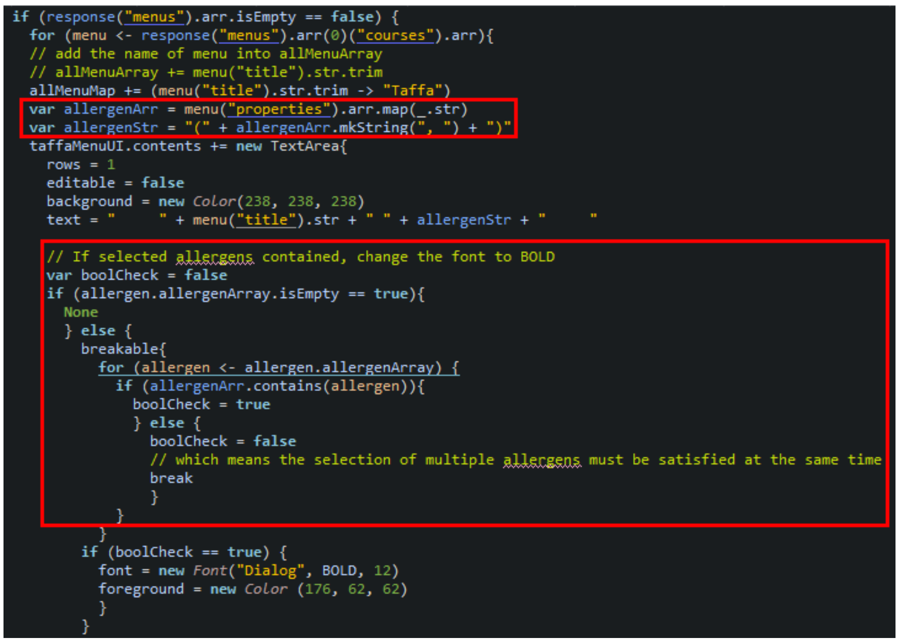
Chemicum - A.I. Virtasen aukio 1, 00560 Helsinki
Sorry, but the restaurant doesn't open on Sunday

No menu available

Figure 11. Different results based on the received JSON data

✓ Allergen Filter Function

Allergen filtering is performed in the process of JSON parsing. For this purpose, the Boolean is used to check whether the menu satisfies all selected allergens at the same time (Conjunctive addition) as following:



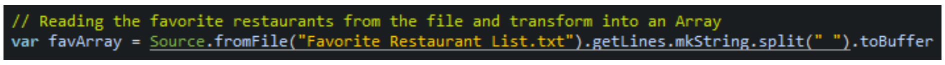
```
if (response("menus").arr.isEmpty == false) {
  for (menu <- response("menus").arr(0)("courses").arr){
  // add the name of menu into allMenuArray
  // allMenuArray += menu("title").str.trim
  allMenuMap += (menu("title").str.trim -> "Taffa")
  var allergenArr = menu("properties").arr.map(_.str)
  var allergenStr = "(" + allergenArr.mkString(", ") + ")"
  taffaMenuUI.contents += new TextArea{
    rows = 1
    editable = false
    background = new Color(238, 238, 238)
    text = "      " + menu("title").str + " " + allergenStr + "      "

    // If selected allergens contained, change the font to BOLD
    var boolCheck = false
    if (allergen.allergenArray.isEmpty == true){
      None
      } else {
        breakable{
          for (allergen <- allergen.allergenArray) {
            if (allergenArr.contains(allergen)){
              boolCheck = true
              } else {
                boolCheck = false
                // which means the selection of multiple allergens must be satisfied at the same time
                break
                }
            }
          }
        if (boolCheck == true) {
          font = new Font("Dialog", BOLD, 12)
          foreground = new Color (176, 62, 62)
          }
        }
```

Figure 12. Algorithm for filtering selected allergens

If one or more allergens are not included in their allergen properties array, the loop will break, like in the Figure 12, and the Boolean value will be remained as “false”. This algorithm guarantees that only the menus that satisfy all conditions will be displayed in BOLD and RED color text.

✓ Restaurant Filter Function

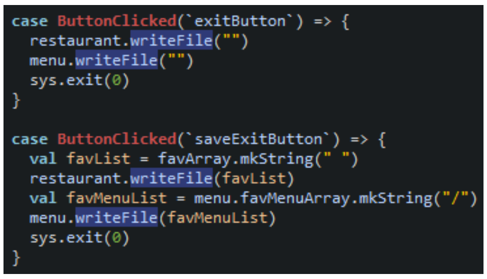
Restaurant filtering is done with the help of separate text file. First the program will create the ArrayBuffer based on the contents of the file "Favorite_Restaurant_List.txt".



```
// Reading the favorite restaurants from the file and transform into an Array
var favArray = Source.fromFile("Favorite_Restaurant_List.txt").getLines.mkString.split(" ").toBuffer
```

Figure 13. Creating ArrayBuffer based on the separate text file

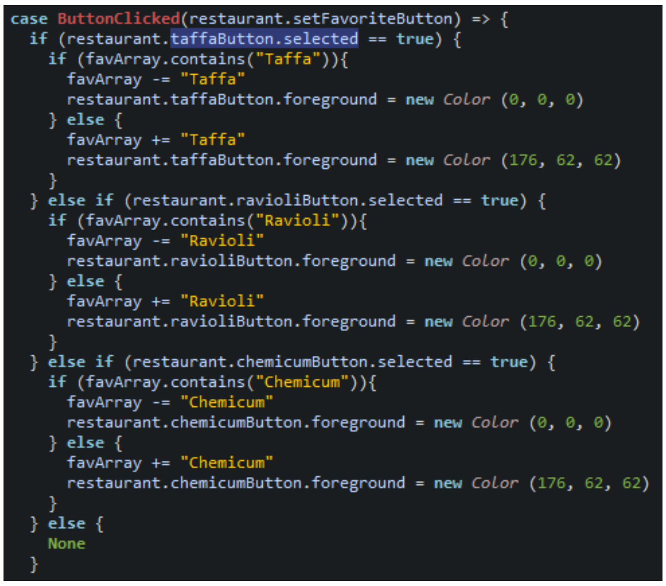
Then the program works with this array (Add / Remove based on the buttonClicked of “Set / Unset as favorite” button) until the program is closed. Users can choose to save the list (By clicking “Save and Exit”) and, in this case, the program will write the list of the BufferArray into the textfile as a string so that the program can remember users’s choice next time when program runs. However, if the user chooses not to save the list, empty string will be written on the text file instead.



```
case ButtonClicked(`exitButton`) => {
  restaurant.writeFile("")
  menu.writeFile("")
  sys.exit(0)
}

case ButtonClicked(`saveExitButton`) => {
  val favList = favArray.mkString(" ")
  restaurant.writeFile(favList)
  val favMenuList = menu.favMenuArray.mkString("/")
  menu.writeFile(favMenuList)
  sys.exit(0)
}
```

Figure 14. Two cases of handling user’s preference

Figure 15 below shows how the selected restaurant is added to the favorite list and indicated by different text color. The data structure “ArrayBuffer” is used for favorite list because it provides convenient way of adding and removing elements in the list.

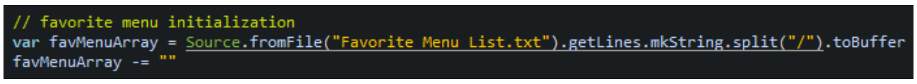


```
case ButtonClicked(restaurant.setFavoriteButton) => {
  if (restaurant.taffaButton.selected == true) {
    if (favArray.contains("Taffa")){
      favArray -= "Taffa"
      restaurant.taffaButton.foreground = new Color (0, 0, 0)
    } else {
      favArray += "Taffa"
      restaurant.taffaButton.foreground = new Color (176, 62, 62)
    }
  } else if (restaurant.ravioliButton.selected == true) {
    if (favArray.contains("Ravioli")){
      favArray -= "Ravioli"
      restaurant.ravioliButton.foreground = new Color (0, 0, 0)
    } else {
      favArray += "Ravioli"
      restaurant.ravioliButton.foreground = new Color (176, 62, 62)
    }
  } else if (restaurant.chemicumButton.selected == true) {
    if (favArray.contains("Chemicum")){
      favArray -= "Chemicum"
      restaurant.chemicumButton.foreground = new Color (0, 0, 0)
    } else {
      favArray += "Chemicum"
      restaurant.chemicumButton.foreground = new Color (176, 62, 62)
    }
  } else {
    None
  }
```

Figure 15. The algorithm for handling favorite restaurant list

✓ Menu Filter Function

Menu filtering is done in a similar way as the restaurant filtering. It also uses the separate text file “Favorite_Menu_List.txt”. First of all , the ArrayBuffer is created based on the list in the text file when program started.



```
// favorite menu initialization
var favMenuArray = Source.fromFile("Favorite_Menu_List.txt").getLines.mkString.split("/").toBuffer
favMenuArray -= ""
```

Figure 16. Initialization of the favorite menu array

Then, the program works with this array until the program is closed. The list of the favorite menu will be displayed in the BoxPanel like the figures below and both cases are handled when the list is empty and not empty.

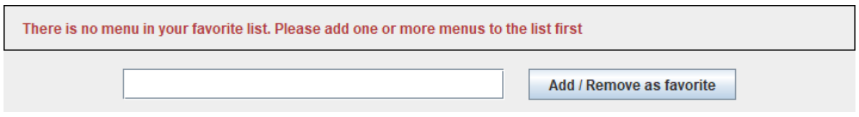
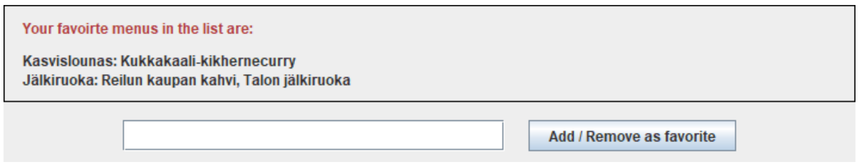

There is no menu in your favorite list. Please add one or more menus to the list first

Figure 17. The case where the favorite menu list is empty


Your favoirte menus in the list are:

Kasvislounas: Kukkakaali-kikhernecurry
Jälkiruoka: Reilun kaupan kahvi, Talon jälkiruoka

Figure 18. The case where the favorite menu list contains one or more elements

The box panel gets updated in real time by the algorithm presented in the figure 19. It updates the box panel based on the “favMenuArray” that is handled in the UI.scala.

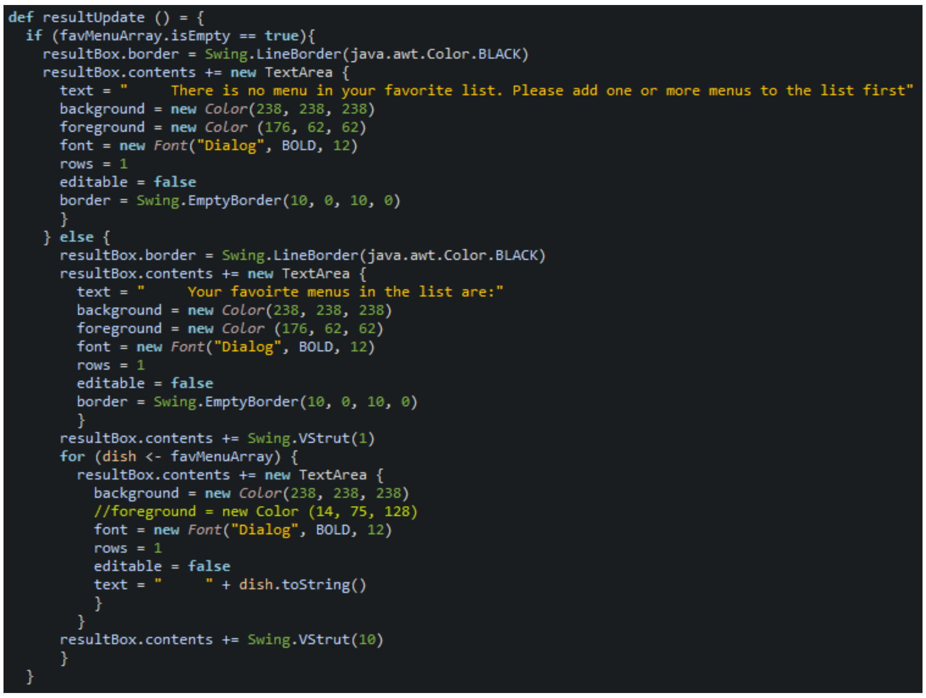

```scala
def resultUpdate () = {
  if (favMenuArray.isEmpty == true){
    resultBox.border = Swing.LineBorder(java.awt.Color.BLACK)
    resultBox.contents += new TextArea {
      text = "     There is no menu in your favorite list. Please add one or more menus to the list first"
      background = new Color(238, 238, 238)
      foreground = new Color (176, 62, 62)
      font = new Font("Dialog", BOLD, 12)
      rows = 1
      editable = false
      border = Swing.EmptyBorder(10, 0, 10, 0)
      }
    } else {
      resultBox.border = Swing.LineBorder(java.awt.Color.BLACK)
      resultBox.contents += new TextArea {
        text = "     Your favoirte menus in the list are:"
        background = new Color(238, 238, 238)
        foreground = new Color (176, 62, 62)
        font = new Font("Dialog", BOLD, 12)
        rows = 1
        editable = false
        border = Swing.EmptyBorder(10, 0, 10, 0)
        }
      resultBox.contents += Swing.VStrut(1)
      for (dish <- favMenuArray) {
        resultBox.contents += new TextArea {
          background = new Color(238, 238, 238)
          //foreground = new Color (14, 75, 128)
          font = new Font("Dialog", BOLD, 12)
          rows = 1
          editable = false
          text = "     " + dish.toString()
          }
        }
      resultBox.contents += Swing.VStrut(10)
      }
  }
```

Figure 19. The algorithm for updating the favorite list and its panel

The function is activated when the user clicks the “Add / Remove as favorite” button.

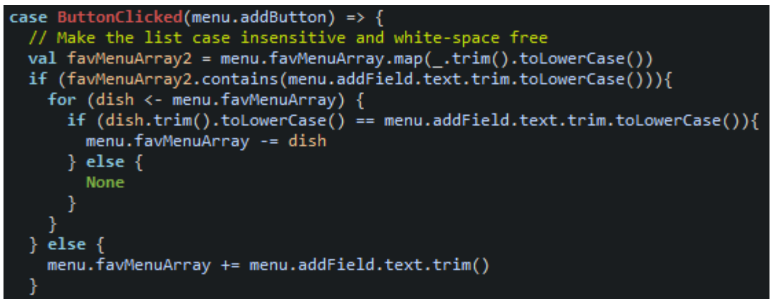

```scala
case ButtonClicked(menu.addButton) => {
  // Make the list case insensitive and white-space free
  val favMenuArray2 = menu.favMenuArray.map(_.trim().toLowerCase())
  if (favMenuArray2.contains(menu.addField.text.trim.toLowerCase())){
    for (dish <- menu.favMenuArray) {
      if (dish.trim().toLowerCase() == menu.addField.text.trim.toLowerCase()){
        menu.favMenuArray -= dish
      } else {
        None
      }
    }
  } else {
    menu.favMenuArray += menu.addField.text.trim()
  }
```

Figure 20. The algorithm for calling the update function when the designated button clicked

The input string will be trimmed so that possible unintended white space from copying and pasting the menu name will not affect the result. Also, it will be compared as lower-cased with elements in the array “favMenuArray”, in order to make the list case sensitive. For example, the string “Jälkiruoka: Mansikkakiisseli” and the “JälkiRUOKA: MansiKKakiiSSEli   ” are considered same string.

Since the menu filter accepts the string as an input and use the function "contains()" to find the match, both partial and whole name can be used to find related favorite menu. For example, for the menu "Jälkiruoka: Mansikkakiisseli" both complete string "Jälkiruoka: Mansikkakiisseli" (White space or case of letters does not matter) and "Jälkiruoka" or "Mansikkakiisseli" would find the menu in question correctly.

But the representation of the notification is different as depicted in figure 6 and 7.

6. Data Structures

The ArrayBuffer is widely used for the list in this project because of it provides convenient way of adding and removing elements by "+=" and "-=". Mutable and immutable data structures are selectively used based on its purpose. For the MenuFilter class, "Map" is used to get the corresponding restaurant name for each menu. That is how the pop-up dialog title could contain the correct name of the restaurant where the favorite menu is served. (Figure 6 and 7)

7. Testing

The initial plan of testing by classmates was cancelled due to the virus situation. Instead, the testing was done by author himself and his wife who has a great sense of design. The codes are created and tested by classes one by one. The program was tested in different environments (Mac, Windows) and passed all tests (Especially, regards to the features) before the submission.

8. Known bugs and missing features

The "editDone" reactions of the text field malfunctions as mentioned earlier (Clicking the area besides the text field activate the reaction), but it is commented out for smooth testing. Also, the author tried to separate the JSON parsing functions ("searchTaffaMenu", "searchRavioliMenu", and "searchChemicumMenu") and validateDate function into designated classes, but they ended up in Exception in thread "main" java.lang.StackOverflowError. For this reason, JSON parsing functions and "validateDate" function are remained in the UI.scala file.

Apart from these, there are no other known errors and all features are implemented as described.

9. Best sides and Weaknesses

Best sides:

1) Responsive Design: Resize automatically based on the included contents
2) Intuitiveness: Easy to test the program and results are presented clearly [Color, Text style] (Also, separate info panels are prepared to help users to test the program easier)
3) Versatility: Covers all additional features and responsive to many different possible cases (For example, the pop-up dialog that covers both cases where the name of the favorite menu matches the whole / partial name of the menu on the day)

Weaknesses:

1) Relatively slow loading: Possibly due to amount and the complexity of codes
2) Non-scrollable panels: It can cause a problem for small monitors (Less than 13 inches) (However, it would not be a problem in most cases if info-panels are deactivated)

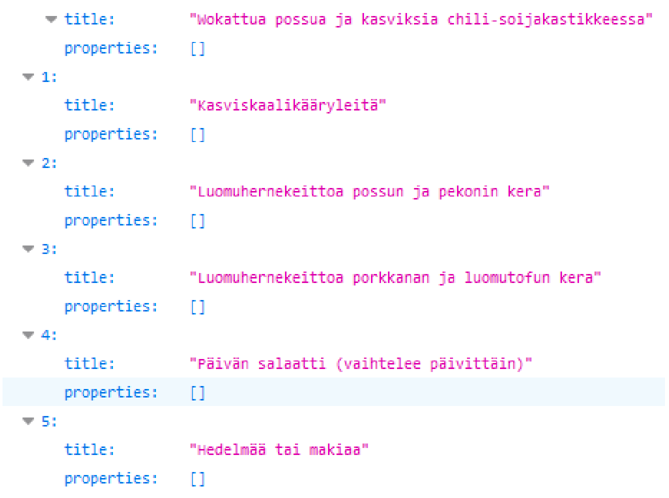
10. Deviations from the plan, realized process and schedule

The project did not go well with the initial plan because of unexpected situations. Most of all, the virus situation made restaurants closed and the API used in the beginning no longer worked. Therefore, the program had to be re-designed with the Kanttiinit API and this slowed down the progress significantly. Therefore, most of the works are done at the last two weeks after all other exams are ended. Through this opportunity, the author could realize the importance of having back-up plans and the ability to adapt to situation when things do not go according to the plan.

11. Final evaluation

In general, the author is satisfied with the result of the project. There were sudden changes and difficulties derived from the virus situation but the program and all suggested functions are successfully implemented and works as it should be.

Because the program is designed for three restaurants only, it is designed to be unscrollable but responsive instead. The reason for this decision is because the restaurant “Taffa”, “Ravioli” and “Chemicum” are only ones that are serving till this day and contain complete data. The restaurant “Maukau” also opens to this day, but it did not provide allergens list in JSON data.



```
  ▼ title:        "Wokattua possua ja kasviksia chili-soijakastikkeessa"
    properties:   []
▼ 1:
    title:        "Kasviskaalikääryleitä"
    properties:   []
▼ 2:
    title:        "Luomuhernekeittoa possun ja pekonin kera"
    properties:   []
▼ 3:
    title:        "Luomuhernekeittoa porkkanan ja luomutofun kera"
    properties:   []
▼ 4:
    title:        "Päivän salaatti (vaihtelee päivittäin)"
    properties:   []
▼ 5:
    title:        "Hedelmää tai makiaa"
    properties:   []
```

Figure 21. Empty allergen properties in JSON data from the restaurant Maukau

For this reason, aforementioned three restaurants are selected for the better visualization of the program. (So that the starting screen [menus of the current date] of the program will not have empty menus) And therefore, the author did not feel necessity to make panels scrollable.

Thus, for the future situation when all restaurants operate normally, the program can be improved to be scrollable so that it can contain any amount of data from all restaurants.

Also, if I get to start the project again from the beginning, I will start by defining and organizing the classes more detailed and precisely. And work on them one by one systematically according to the general plan so that the process can better adapt to any possible changes. To be honest, the project was tweaked significantly due to sudden malfunctions of Fazer and Sodexo API.

Otherwise, the resulting program is satisfiable

12. References

SCALA Tutorials by Alvin Alexander
https://alvinalexander.com/scala/

SCALA GUI Programming by Otfried Cheong
http://otfried.org/scala/gui.html

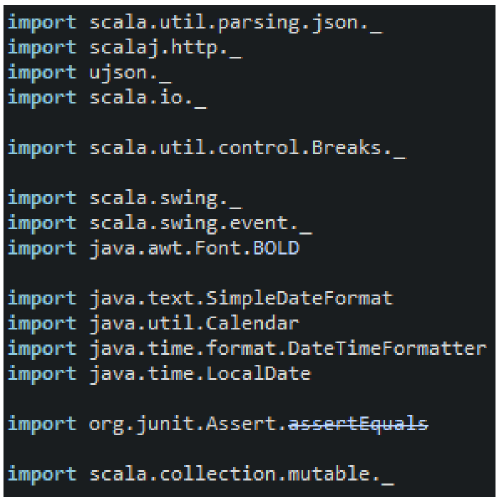
SCALA Swing Official Document
https://www.scala-lang.org/api/2.9.1/scala/swing/package.html

JAVA Swing Tutorial
https://www.javatpoint.com/java-swing

Oracle's JAVA Swing Tutorial
https://docs.oracle.com/javase/tutorial/uiswing/index.html

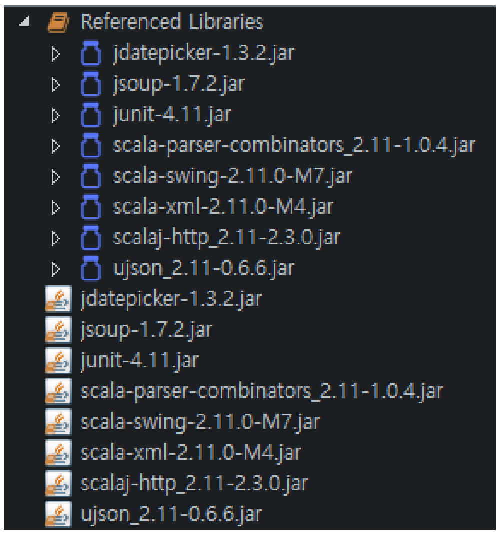
And the Stack Overflow when faced problems and errors

13. Appendixes



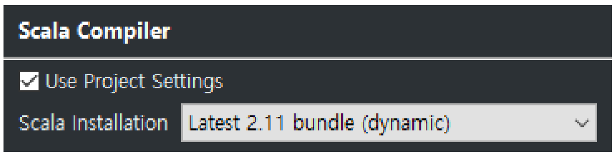
```scala
import scala.util.parsing.json._
import scalaj.http._
import ujson._
import scala.io._

import scala.util.control.Breaks._

import scala.swing._
import scala.swing.event._
import java.awt.Font.BOLD

import java.text.SimpleDateFormat
import java.util.Calendar
import java.time.format.DateTimeFormatter
import java.time.LocalDate

import org.junit.Assert.~~assertEquals~~

import scala.collection.mutable._
```

Appendix 1. Imported libraries for the project



```
Referenced Libraries
  jdatepicker-1.3.2.jar
  jsoup-1.7.2.jar
  junit-4.11.jar
  scala-parser-combinators_2.11-1.0.4.jar
  scala-swing-2.11.0-M7.jar
  scala-xml-2.11.0-M4.jar
  scalaj-http_2.11-2.3.0.jar
  ujson_2.11-0.6.6.jar
jdatepicker-1.3.2.jar
jsoup-1.7.2.jar
junit-4.11.jar
scala-parser-combinators_2.11-1.0.4.jar
scala-swing-2.11.0-M7.jar
scala-xml-2.11.0-M4.jar
scalaj-http_2.11-2.3.0.jar
ujson_2.11-0.6.6.jar
```

Appendix 2. Referenced Libraries

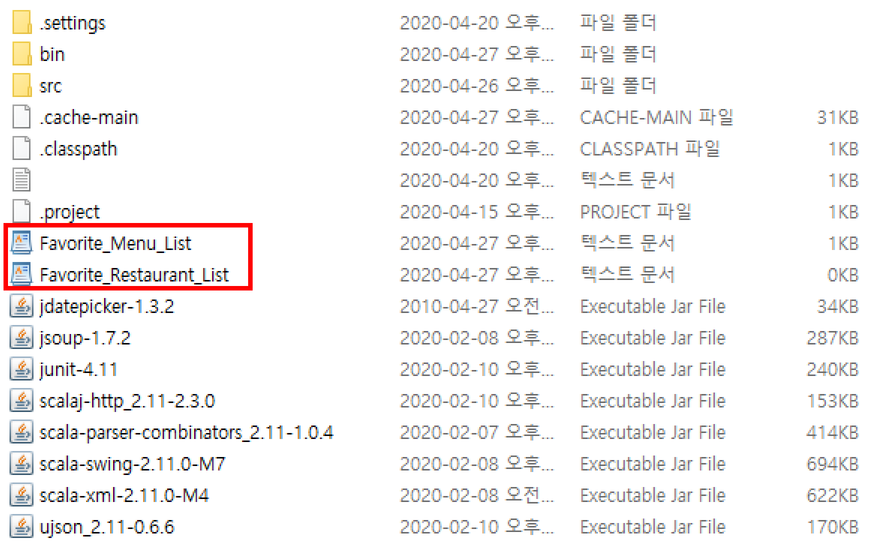


Scala Compiler

[✓] Use Project Settings

Scala Installation: Latest 2.11 bundle (dynamic)

Appendix 3. Scala version of the project (For Swing Library)



.settings	2020-04-20 오후...	파일 폴더	
bin	2020-04-27 오후...	파일 폴더	
src	2020-04-26 오후...	파일 폴더	
.cache-main	2020-04-27 오후...	CACHE-MAIN 파일	31KB
.classpath	2020-04-20 오후...	CLASSPATH 파일	1KB
	2020-04-20 오후...	텍스트 문서	1KB
.project	2020-04-15 오후...	PROJECT 파일	1KB
Favorite_Menu_List	2020-04-27 오후...	텍스트 문서	1KB
Favorite_Restaurant_List	2020-04-27 오후...	텍스트 문서	0KB
jdatepicker-1.3.2	2010-04-27 오전...	Executable Jar File	34KB
jsoup-1.7.2	2020-02-08 오후...	Executable Jar File	287KB
junit-4.11	2020-02-10 오후...	Executable Jar File	240KB
scalaj-http_2.11-2.3.0	2020-02-10 오후...	Executable Jar File	153KB
scala-parser-combinators_2.11-1.0.4	2020-02-07 오후...	Executable Jar File	414KB
scala-swing-2.11.0-M7	2020-02-08 오후...	Executable Jar File	694KB
scala-xml-2.11.0-M4	2020-02-08 오전...	Executable Jar File	622KB
ujson_2.11-0.6.6	2020-02-10 오후...	Executable Jar File	170KB

Appendix 4. Location of text files for write function