* Project title: CAU-IIKH
* List of team members: <Class 2 - Team 1>  
  20182705 고주형, 20185784 김호성, 20182610 손희승, 20162874 이준협, 20142611 이하람
* Presentation speaker name: 고주형
* Brief project description (summary)  
  Efficient kitchen helper that is used for managing a lot of recipes, and planning daily meals like breakfast, lunch or dinner.  
  By using our kitchen helper, you can SAVE/DELETE/SEARCH for recipes and manage your meal plans.  
  We also put ascii arts inside our program. Pretty UI makes program livelier and fresher. 😊
* How to compile and execute

1. Change directory to Source Code Package Folder
2. Open the Solution File with Visual Studio 2019
3. Compile (Ctrl-F5) and Execute

* And also, an executable file is in Release Folder so you can directly execute our project file. (How to use executable file is in README.txt)
* System requirement for compilation and execution  
  Target OS: Windows 7 / 10  
  System Recommendation: Same as Visual Studio 2019 system requirement

1. Description on functionality that was implemented in your SW system:

* Add recipe: Add a new recipe to our Database.
* Delete recipe: Delete existing recipe from our Database.
* Search recipe (Print all recipes):

Search existing recipe from our Database by recipe name. It is automatically printed in alphabetical order.  
Also, The containing word(Keyword) Search is supported(If you search “pie”, all kinds of recipes whose name includes substring “pie” will be searched. For Example, “apple pie”, “raspberry pie” …)

* Add meal plan: Add new meal plan to our Database.
* Print meal plan: Print all meal plans’ title and breakfast, lunch, dinner’s menu. It is automatically printed in alphabetical order.
* Delete meal plan: Delete existing meal plan from our Database.
* Pretty Ascii art: Maybe it will make user happy and make program fancier.

1. How you implemented (important implementation issues):

* At first, we tried to identify the project given to us before designing. What we were trying to create was IIKH, which needed to create and read a database of recipes and plans. Therefore, we focused on the database when designing the program. Recipes and plans have a database that contains recipe data or plan data. We needed to save the database as a file to load and save, and we chose the file format of the database to be csv. We chose csv format because the delimiter of csv was ','. We thought it would be easy to convert.   
  After that, the story came up to here, there were several things we had to do. One is to create the ability to read and write data using the csv file format, the other is to build the internal structure of the data and the database, and the other is to use the data to retrieve, display, or add data within the program. Based on this, we completed the division of roles, and after some communication, integrated the functions and finally completed the program.
* Important Issues

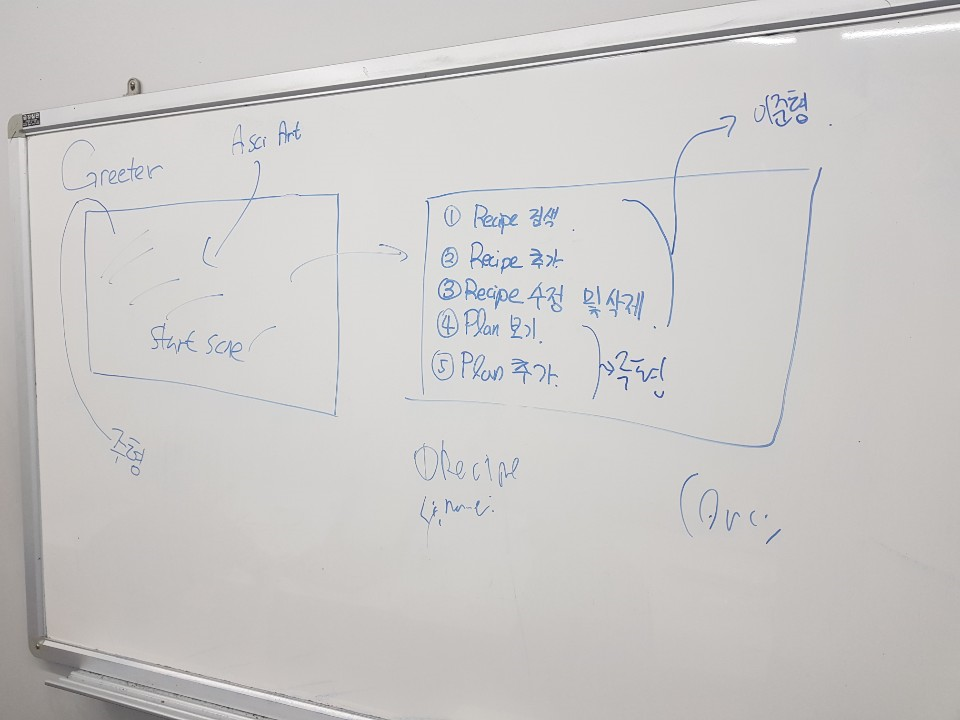
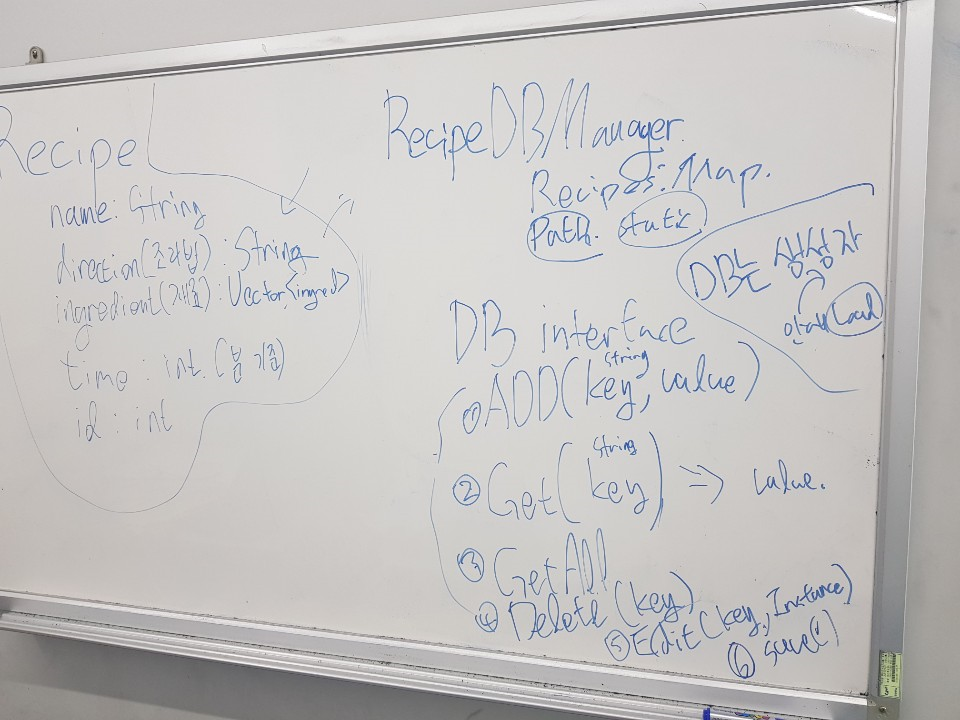
● Issue01: Problem was ambiguous.

Due to ambiguous specification, we had to talked a lot about what is this program, what is our target, what should we implement and how we’ll implement this system. Though we had clarified our objective (what we’ll implement). The Problems was that the overall program design which is about how we’ll implement this program, was chosen by just talking. There was no document or logs about what we talked. Also, after some coding we found out everyone was thinking slightly different design.

By that problem when we are talking about our program, we had many issues such as, Team Member\_A thought it is better to make Database Manager for all objects (recipeDBManager, planDBManager, mealDBManaer). But other Team Member\_B thought it is better to make integrated Database Manager that can handle all of the objects. By this mismatched when someone coded a Database header, few team members could understand it.

Also, the Plan class was confusing. It was not descripted correctly and specification was made by just talking. For instance, our one team member thought date should be addressed in plan but other team member thought date will be addressed in meal class. Such kinds of situation led our design more complicated and finally became not understandable. So, we had an emergency meeting and unified all of our design. After that we could start programming our given part for each peacefully and confidently.

Below images are meeting logs taken when we were doing Emergency Meeting.

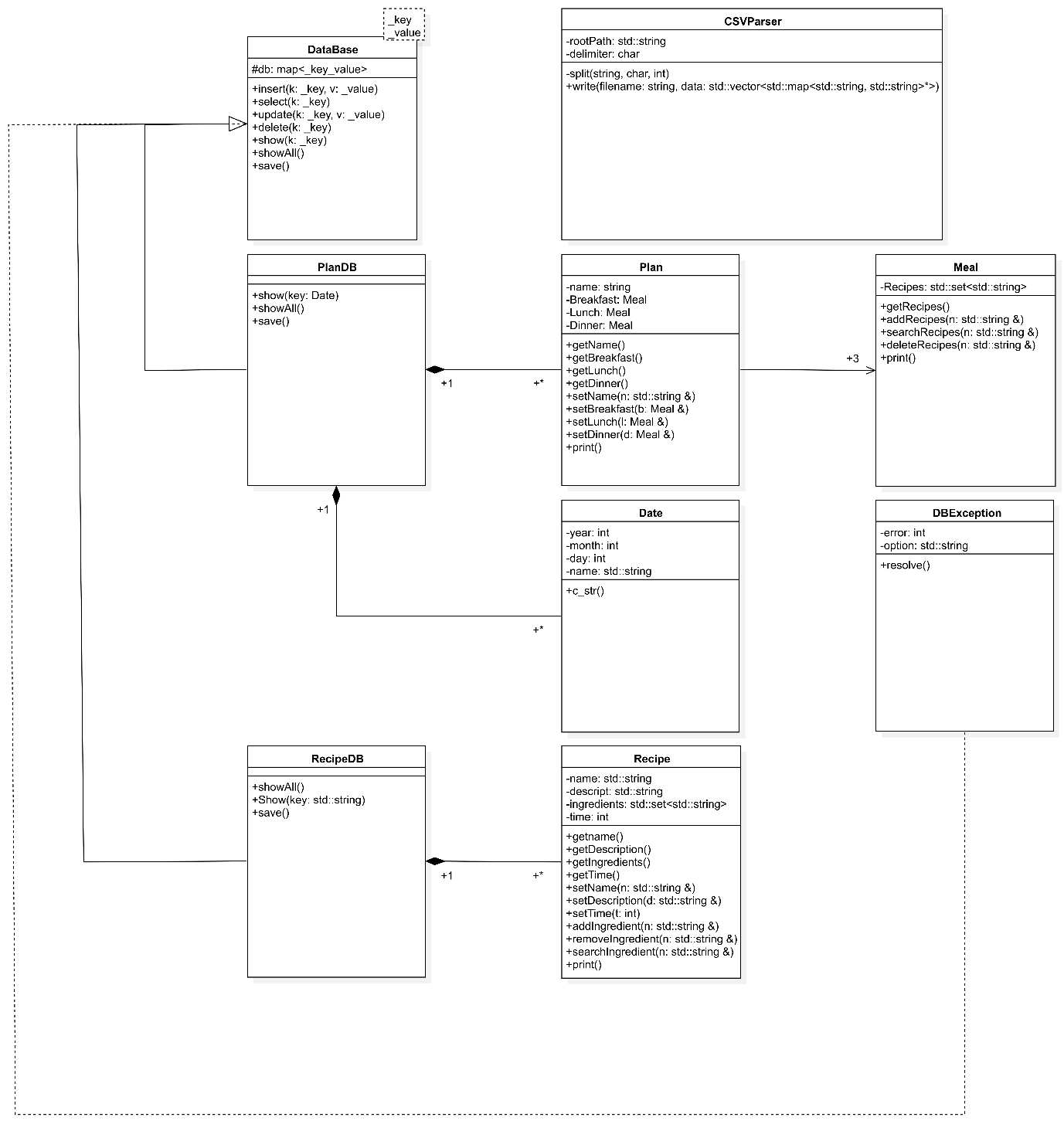


● Issue02: Designing component and giving responsibilities to specific class.

We argued a lot about responsibilities. If a specific class controls every class, the dependency becomes high. This was bad. Bad to debug or understand or extend as we learned.

For example, meal information was needed in everywhere. A Planner had to know meals for daily meal plan, Database Manager should contain every meal for saving, and meal should contain date for searching meal plan by date. After some coding meal class became too huge, and dependent. So, we lowered the dependency of class by giving responsibility to each class and requesting to other class when other class’ responsibility is need.

1. The result of SW system design [UML]:



1. Execution results: show real examples of program execution.

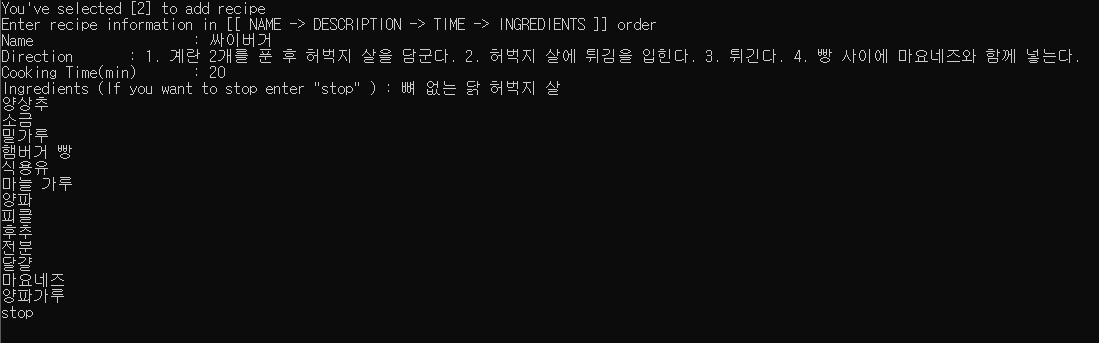
(use screen capture)

Show that each function of the SW system is working correctly.

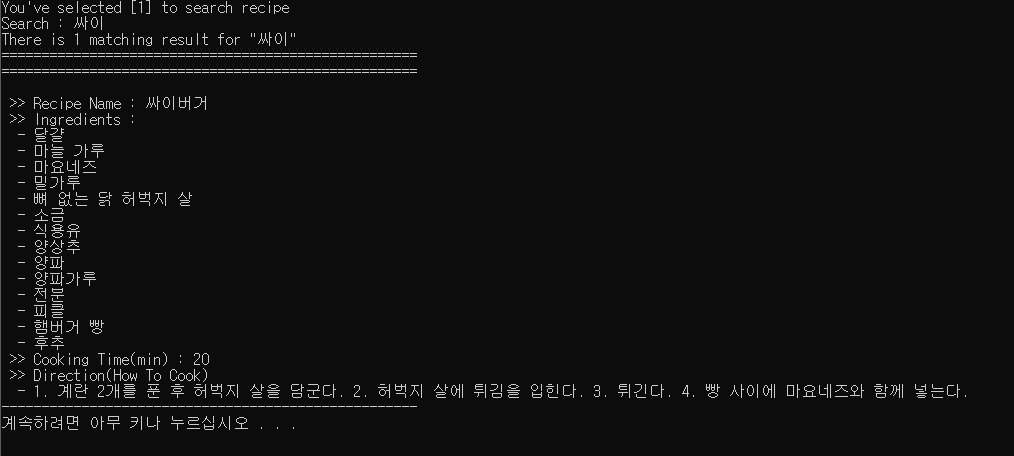
1. Start Scene



2. Adding Thigh Burger Recipe (Menu 2)



3. Searching Thigh Burger Recipe that I just added. (Menu 1)



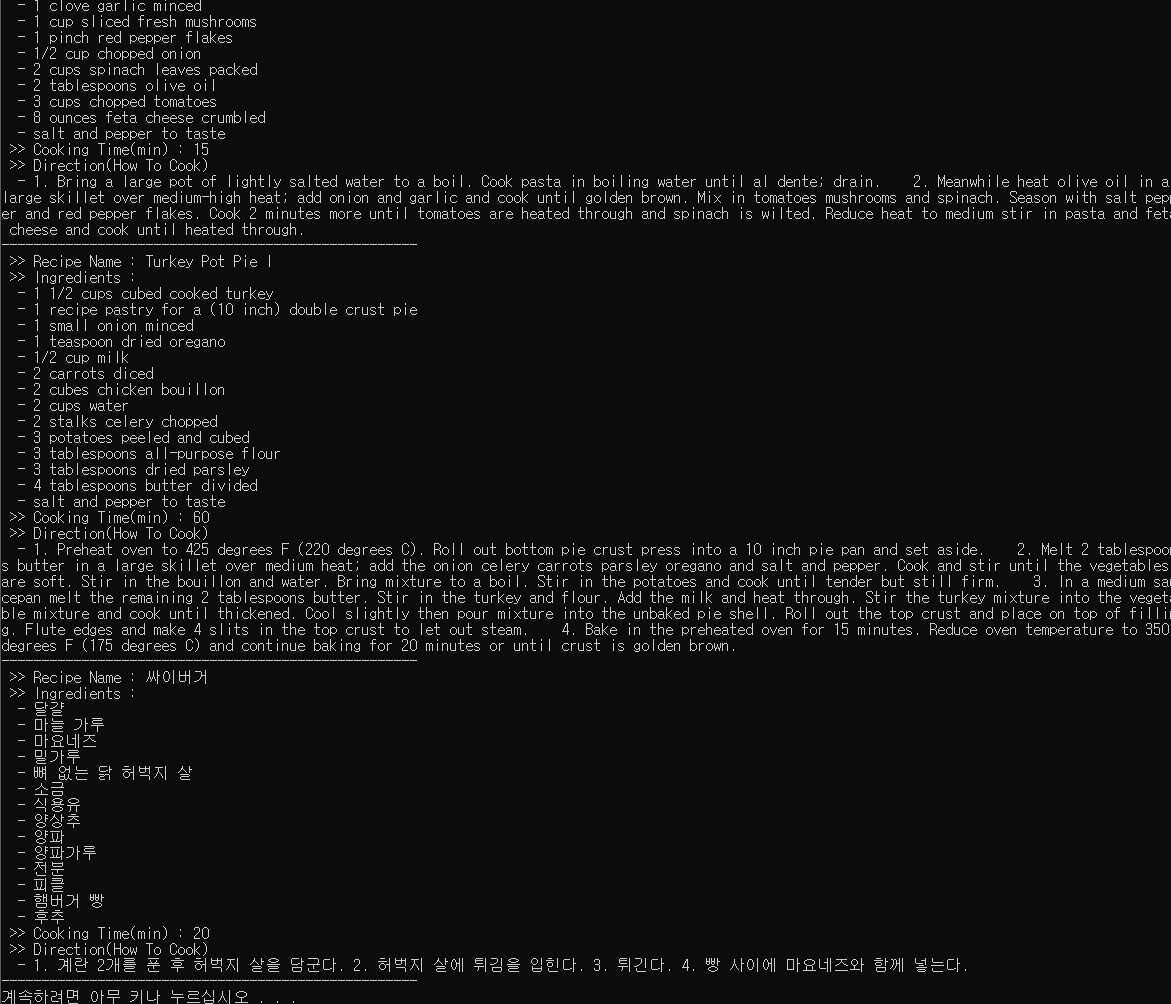
* You can see all Recipes containing searched word were being printed.



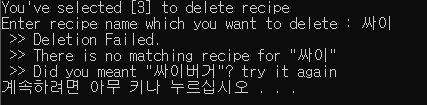
(This Screen capture is about keyword search)

As you see, by searching “Chicken”, all recipes about chicken were printed out.

4. See all Recipes including Thigh Burger Recipe that I just added. (Menu 4)

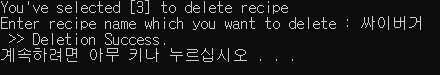


5-1. Trying to Delete Thigh Burger Recipes that I just added. (Menu 3)

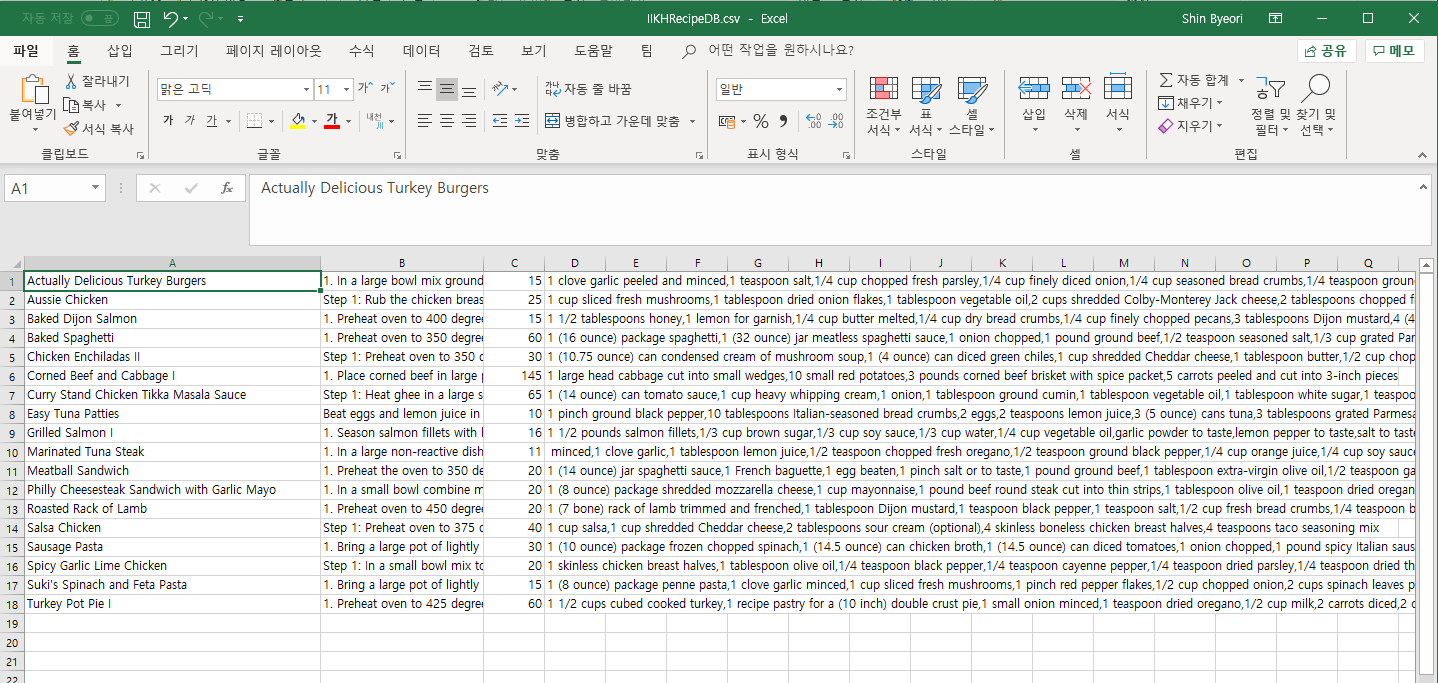


* Deletion Failed because I didn’t wrote full name. It gives recommendations.

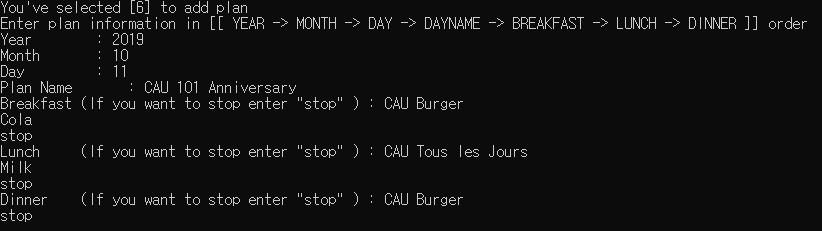
5-2. Deleting Thigh Burger Recipe that I just added. (Menu 3)



* Deleted Thigh Burger because I wrote full name exactly.
* For convenience, we used CSV format so you can see and edit Recipes in the Excel.



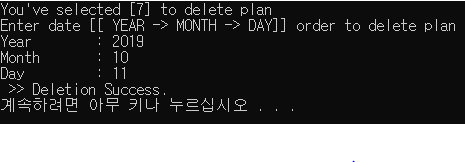
6. Setting Meal Plan for 2019/10/11 (Menu 6)



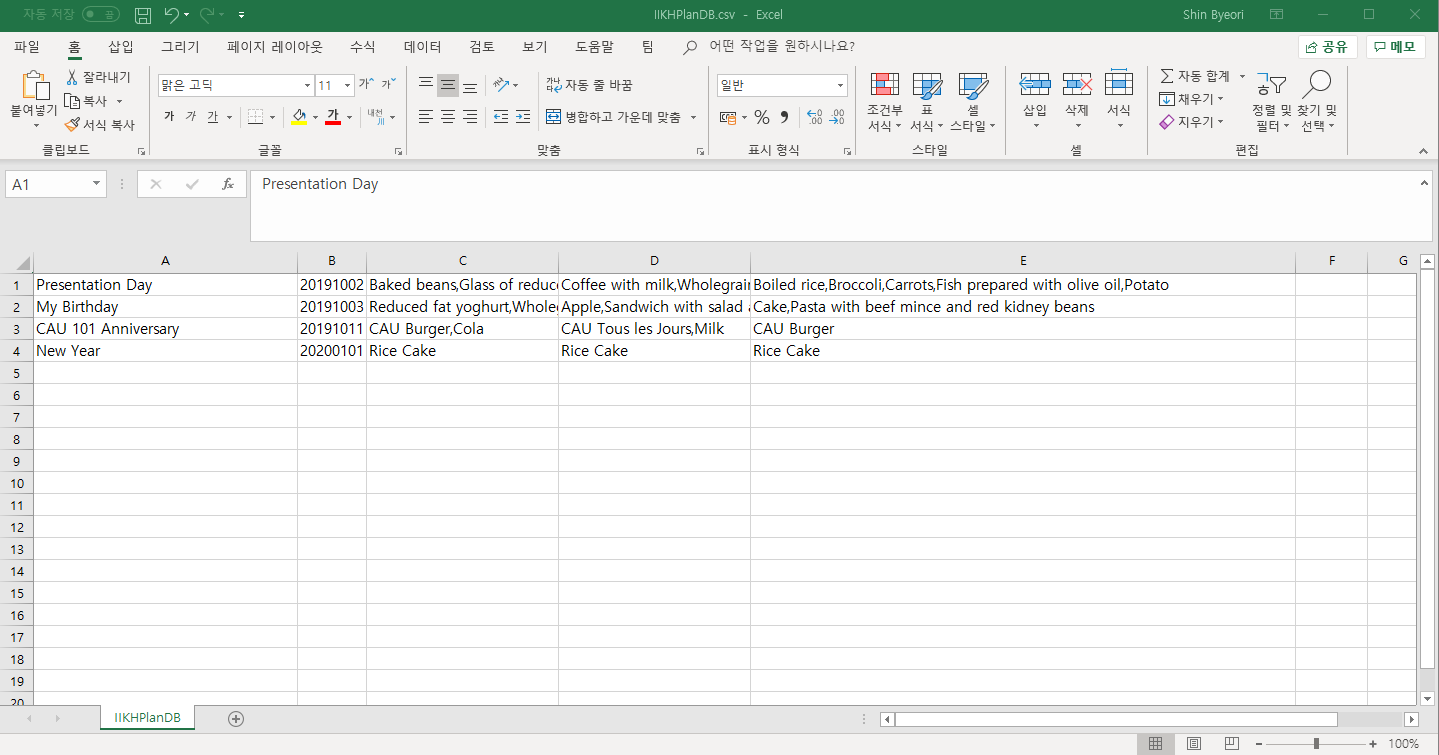
7. You can see all Meal Plans, including Plan you just added (Menu 8)



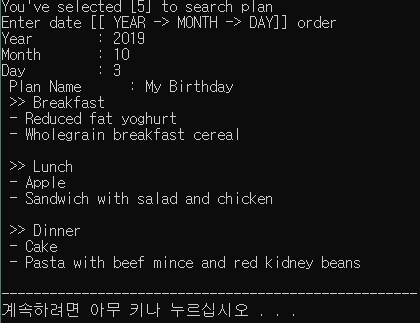
8. You can delete Meal Plan with Date. I deleted CAU 101 Anniversary Plan that I just added. (Menu 7)



* For convenience, we used CSV format so you can see and edit Plans in the Excel.

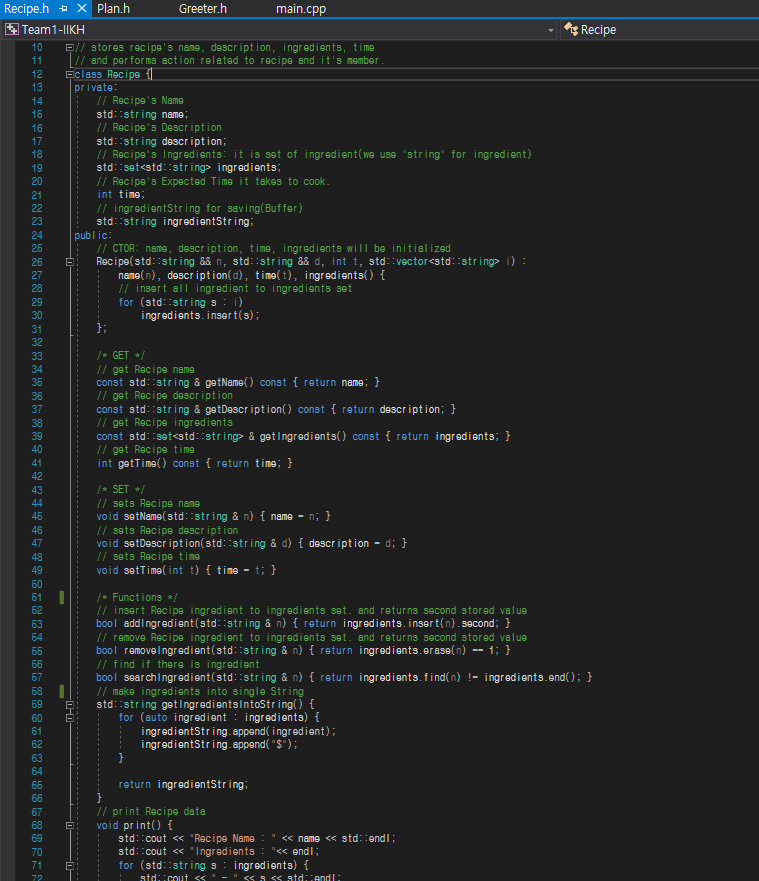


9. You can search Plan by Date.



1. Explain how you applied object-oriented concepts to the development for your project.

1. [Encapsulation] In our OOP class we learned how to hide unnecessary details. We always started from private and when it has to be exposed, we made it public. Plus, we designed the class with their appropriate responsibility. So, our class has only required variables or functions that are related with its responsibility.

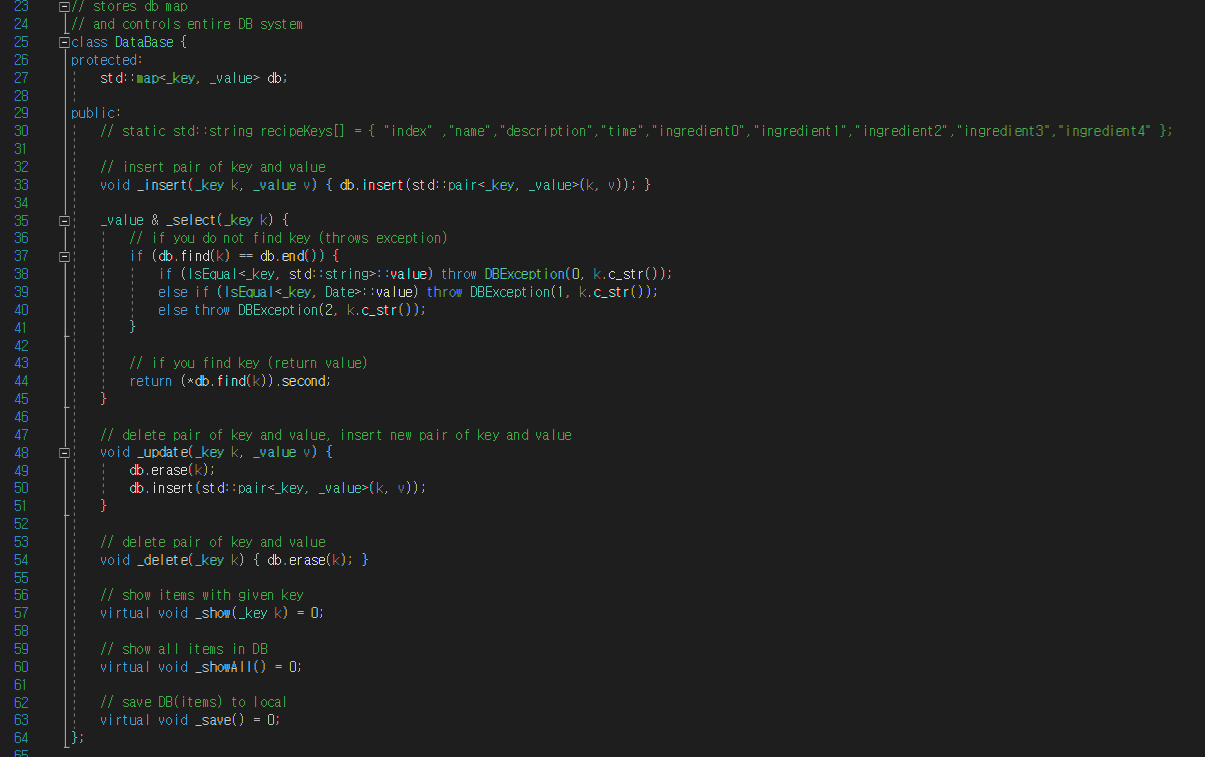


2. [Inheritance] We looked at the objects relation carefully and made programming more efficient.

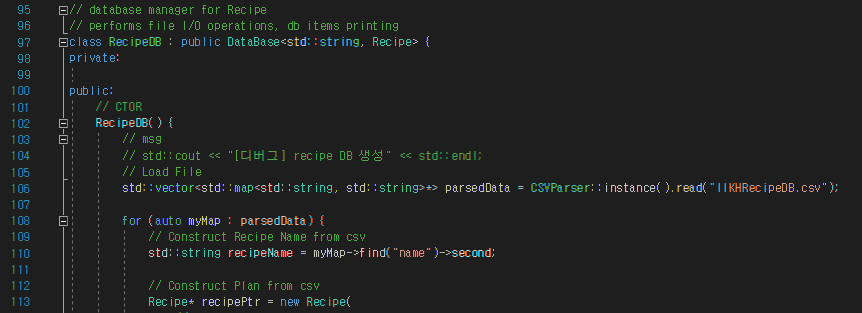
For Instance, in case of database manager, many kinds of database managers were needed. We needed Plan Database manager, Meal Database Manager, Recipe Database Manager. Each Database managers was similar to each other and also the code was almost the same but saving format or few functions details were little bit different. So, we made DataBase [parent class] which had all overlapping details of all kinds of Database Managers. Inheriting Database [parent class] traits, making other Database Managers were easier (High Reusability). Also, we could eliminate the same codes.

In addition, with virtual keyword we implemented different details in the Database Managers’ functions. That keyword acted like interface and all Database Managers [child class] had implemented mandatory details.

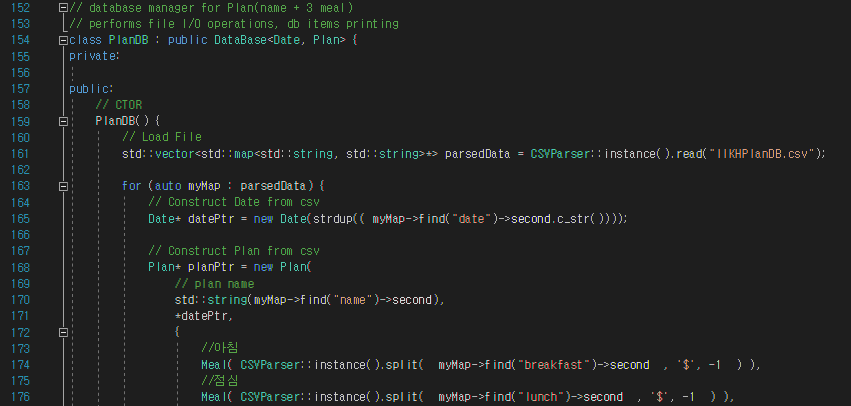
* DataBase [Parent]



* RecipeDB [Child]



* PlanDB [Child]



* Explain what you felt and learned from the project.

[고주형] It was nice to experience object-oriented programming as a team project. It was good chance to think deeply about object-oriented programming and design.

I was team leader so I kept track of project progress. I had to give everyone’s work equally to until the program is fully implemented. Also, I had to think in Bottom Up manner. This gave me enlightenment about what is good design and why we use object-oriented programming. I had to rapidly gathered all of our team member’s code and I integrated it into our master branch (we used git). When convention was different, I had hard time understanding it. But, unified convention and nicely encapsulated code was easy to use and I didn’t confuse much what to use because unneeded functions or variables were forbidden.

[김호성] I've never had a chance to do a project on a team-by-team basis before, and it's been great to be able to do it through this team project, and the process of working together was fun.

[손희승Coding with others was hard at first but after few days later, it became comfortable. I learned few tips. I know what Github is but didn’t use it much. In this project I got chance to know about Git like, how to cooperate using Git, how to invite collaborator. With kind team members, I think I learnt a lot about cooperating.

[이준협] It was interesting to think in various ways because no implementation or details were restricted. And I felt again that teamwork is important.

[이하람] Through this project, I learned and knew about what an Object-oriented programming in the abstract. Also, it was very interesting to discuss together for making the programming because it was the first time for me to learn about c++.

1. Conclusion

Our team project was successful. We argued a lot about what is more good design or what data structure we should use for many models. We learned from each other by talking, arguing and cooperating. We learned how to cooperate in bottom up style code. This kind of experience will be helpful in the future.

Thank You😊