* Project title: [아스키 아트??] 아직 미정 – 컨셉 정해야 됨
* List of team members:   
  20182705 고주형, 20185784 김호성, 20182610 손희승, 20162874 이준협, 20142611 이하람
* Presentation speaker name: [2명 가능한지]
* 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/EDIT/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 executable file is in Release Folder so you can directly execute our project file.
* System requirement for compilation and execution  
  Target OS: Windows 7 / 10  
  System Requirement: Same as Visual Studio 2019 system requirement

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

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

Search existing recipe from our Database by recipe name.   
(구현 실패 시 삭제) Containing Search is supported(If you search “pie”, all kinds of recipes whose name includes substring “pie” will be searched. Like, “apple pie”, “raspberry pie”, …)

* Add meal plan: Add new meal plan.
* Print meal plan: Prints meal plan’s title and breakfast, lunch, dinner.
* Pretty Ascii art: make you happy and makes program fancier.

1. How you implemented (important implementation issues):

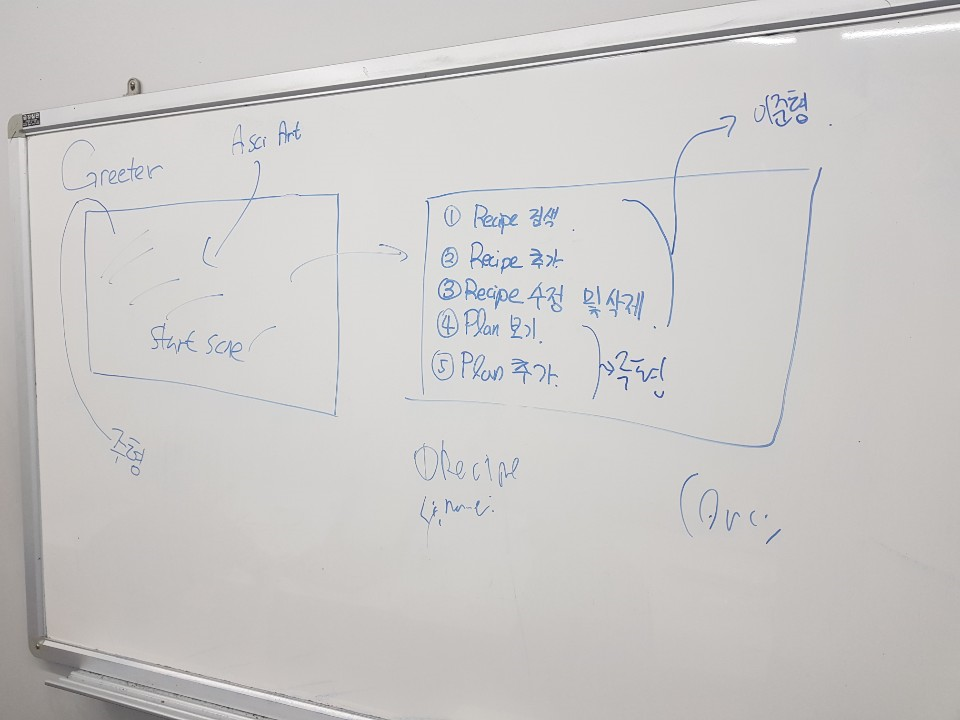
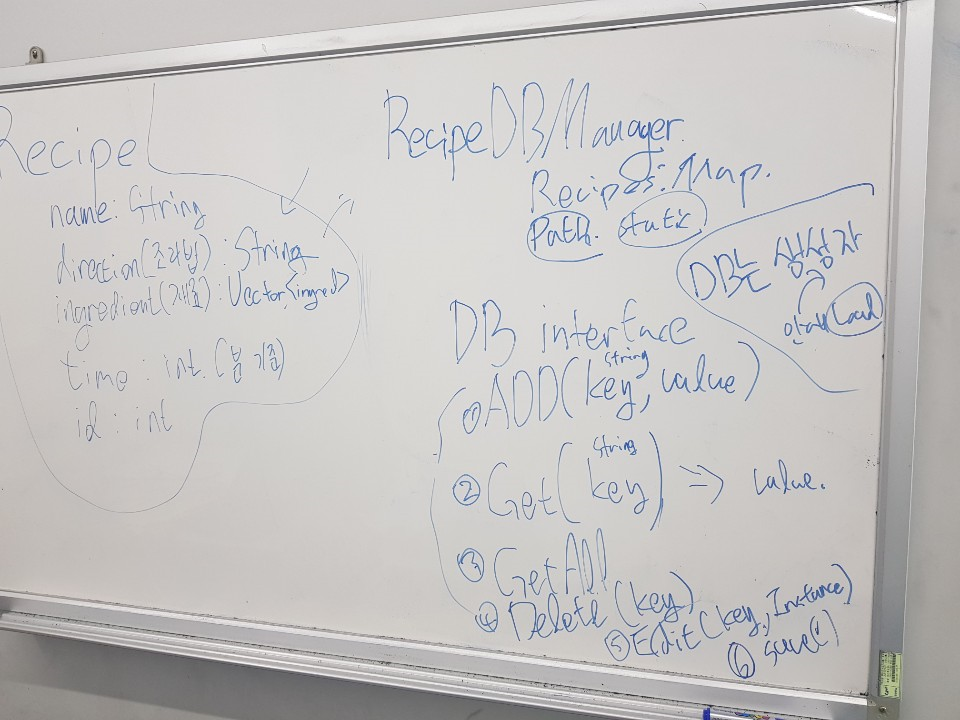
● Issue01: Problem was ambiguous.

Due to ambiguous specification, we talked a lot about what is this program, what is our target, what should we implement and how we’ll implement this system. After clarifying our objective (what we’ll implement) there still difficulties. Problem was that 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. 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 we was making Database Manager for all objects (recipeDBManager, planDBManager, mealDBManaer) but other Team Member\_B thought we was making integrated Database Manager that can handle all of the objects. By this mismatched thought when someone coded Database header, few team members couldn’t understand.

Also, the Plan class was confusing. It was not descripted correctly and specification was made by just talking. One team member thought date will be addressed in plan but other team member thought date will be addressed in meal class. This kind of situation lead our design more complicated and finally become not understandable. So, we had an emergency meeting and unified all of our design. After that we started programmed our given part 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 specific class controls every class dependency become high. This was bad. Bad to debug or understand or extend… as we learnt.

For example, meal information was needed everywhere. Planner had to know meals for daily meal plan, Database Manager should contain every meal for saving, meal should contain date for searching meal plan by date. After some coding meal class became too huge, and dependent. We lowered the dependency 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]:

준협이 형 - UML여기에

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

(use screen capture)

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

되면 데모 만들면서 캡처하기

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

1. [Encapsulation] In our OOP class we learnt 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 is related with its responsibility.

예시 코드 캡쳐해서 여기에 ㄱㄱ

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 were similar to each other and 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]’ trait, making other Database Managers were more easy(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.

--- [Parent Class: Database] UML 넣어영---

● 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] 불평불만

[손희승] Too E-----asy.

[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]

1. Conclusion

Our team project was successful. We argued a lot about what is more good design or what data structure should we use for many models. We learnt from each other by talking. We learnt how to cooperate in bottom up style code.

This kind of experience will be helpful in the future.

Thank You😊