MunchFast

Our Team

Our team consists of Cristian Racila and Florence K. Neflas.

Feature Selection*: How did you determine which features to include in your app? How did you decide which features to exclude?*

So, we decide to implement a Singleton, specifically for Restaurant, since we decided to only use one instance of this class. Additionally, we initially thought of implementing Factory design pattern: we thought of creating a Menu class that is Abstract Factory, and from that class, we would create other factories depending on what Menu type (Breakfast, Lunch, Dinner) we called inside the AbstractMenuFactory class:

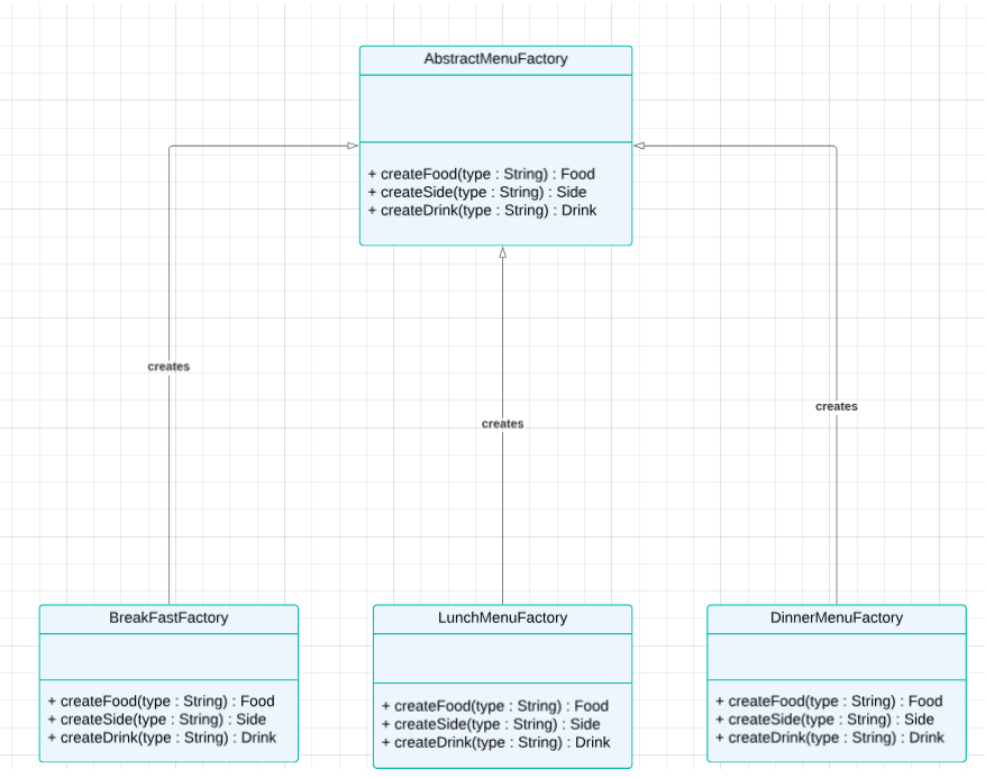

Figure 1: This was our initial plan for (Abstract) Factory design pattern

From the BreakfastFactory, LunchMenuFactory and DinnerMenuFactory, we would then create interfaces (again, according to our old plan) and from the interfaces, we would create the concrete menu items as shown in the following snapshot:

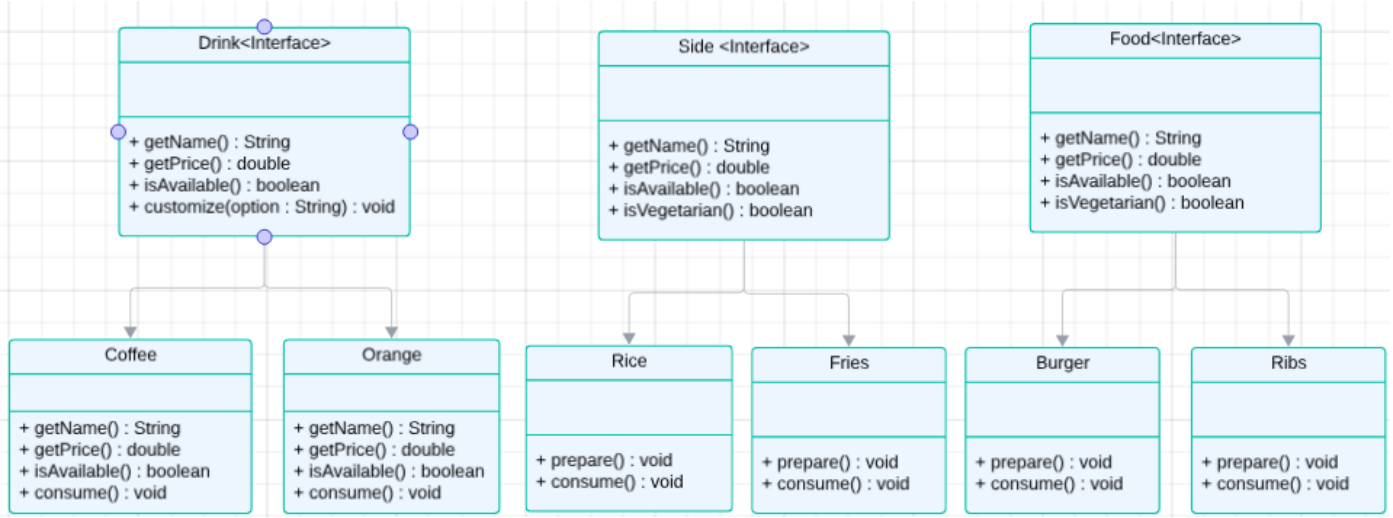

Figure 2: Interfaces created from the Factories and Concrete Item (Classes) that implements the Interfaces

It was only when we started implementing the classes, that we thought of simplifying our classes inside the project that we completely discarded this design pattern and instead implemented MVC and DAO design patterns. In our opinion, it was more simpler to implement the code since most of the implementation was query operations (for the DAO classes, specifically) and most of the code was copy paste for the Controller classes (you may refer to the diagram to view the MVC plan: https://lucid.app/lucidchart/e2d047eb-5313-4e4c-842e-bafc7b34a625/edit?viewport_loc=-13110%2C-3468%2C14583%2C6268%2C0_0&invitationId=inv_e3a1b352-272a-4faf-a8f9-98cb392d299d)

1. ***Technology Choices**: How did you choose the technologies to use for your project?*

   So, we chose NetBeans IDE to easily use the GUI designer since we're familiar with it. Additionally, we used IntelliJ for the code

implementation since we are familiar with it and it also has the autocomplete feature for us, so the code implementation was quicker on this IDE compared to NetBeans.

2. ***Work Division****: How did you divide the work among your team members? Be sure to specify who worked on which parts.*

   Whilst Florence took care of the database connections and all the needed methods to access the database, Cristian took care of the code to link the forms and link them together.

3. ***Challenges Faced****: What were some major difficulties you had to overcome during the development process?*

   We had a few difficulties with merging our code, since we ran into conflicts, but we took care of that, but we would lose some code because Florence had lost power 3 times the night before presentation day. We also had difficulties with synchronizing the menu schedules (Breakfast, lunch..) with our Order class logic. We also had many difficulties regarding the Factory design pattern for this project: in our opinion, we overcomplicated the idea of creating the Menu, so we needed a way to simplify this and we completely discarded the Factory design pattern and implemented DAO and MVC for Menu items, Order class and Customer class to keep our code consistent and simple. Lastly, there was some difficulties with finding an efficient wat to use internationalization throughout the application.

4. ***Unexpected Ease****: What aspects of the project went easier than expected?*

   It was surprisingly easy to implement the code for DAO, which is a design pattern that we had to research for this project: we planned and organized how we would include this design to our project, and to be honest, it was quite simple. Additionally, it was simple to implement our Singleton (Restaurant class) to our application.

5. ***Regrets****: Are there any decisions you regret making during the project?*

   We deeply regretted spending so much time overcomplicating the idea of our project. At first, we were under the impression that we should handle all types of users without considering the impact this would have on the project.

6. ***Git Conflicts****: How did you deal with Git conflicts and version control issues?*

   So, when we dealt with Git conflicts, especially with merging conflicts since it was our big issue with using Git, one of us had to check which changes to keep inside the files that had merging conflicts. Sadly, some wanted changes were lost because we would accidentally choose the wrong changes to keep and discard the changes we wanted to keep.