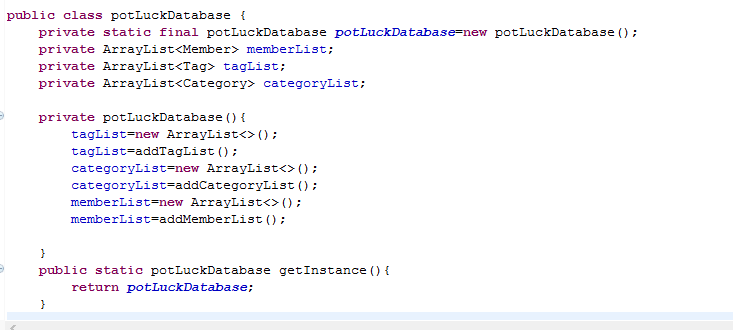
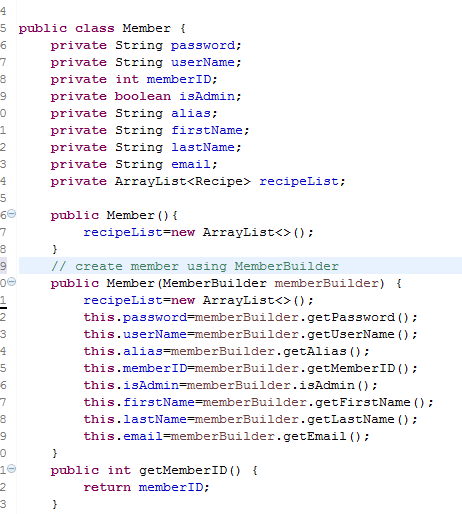
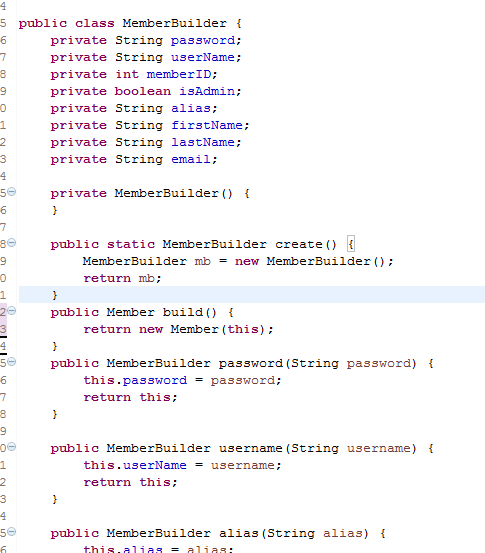
This project develops the recipe storage system. We use four design patterns in this project. They are Builder Design Pattern, Delegate Design Pattern, Singleton Design Pattern and MVC Design Pattern. We also refactor the Member class to the base abstract class, NormalUser and Administrator classes are subclasses of Member class. We create interface RecipeController, then class RecipeControllerImpl which implements this interface to bridge the view layer (RecipeView) and the model (Member). Those four classes implement the MVC Design Pattern and Delegate Design Pattern.

First, we use the Singleton Design Pattern in the PotLuckDatabase class. This class is used to save members, tags, categories. So there should be only one instance of this class in this project to make sure data consistency. Singleton Design Pattern allows only one instance of a given class. We can see that this class keep all data of members, tags, categories in ArrayList until the program is end.

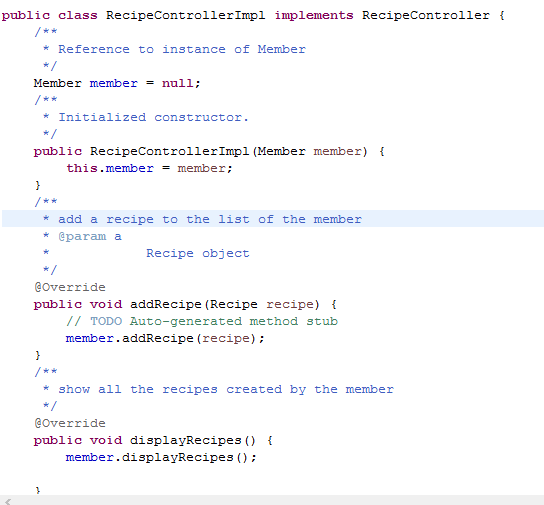


Second, we use the Builder Design Pattern in the MemberBuilder class. This class is used to create complex objects of Member. The class Member has many fields. If write all constructors for every condition, there are telescoping constructors, and the meaning of the parameters in long constructors isn’t self-evident. Using Builder design pattern can separate the construction of Member from its representation, it can create different representation with the same construction process.





Third, we use the Delegate Design Pattern in RecipeControllerImple class. This class is used to complete tasks from RecipeView class, but it delegate its behavior to the Member class. The behaviors can be swapped in and out dynamically while the program is running. In the following code, the constructor can have different Member, administrator or normal user. They have different behaviours.



Fourth, we use MVC design pattern, RecipeView class is the view layer, RecipeControllerImpl class is the control layer, and Member is the model layer.

The singleton design pattern is hard to figure out where to use. However we must use the singleton design pattern to the class which will keep only one copy of the data. Because we can save the data in the Launcher class or other classes, in my group, we save data in another class called PotluckDatabase. It is difficult to decide which class implements the singleton design pattern

In the project, builder design pattern is implemented in MemberBuilder to simply the Member creation. Singleton design pattern is used in the PotluckDatabase class to keep the data consistence. Delegate design pattern increases the flexibility of the program, administrator and normal user do similar behaviour through the RecipeController interface.