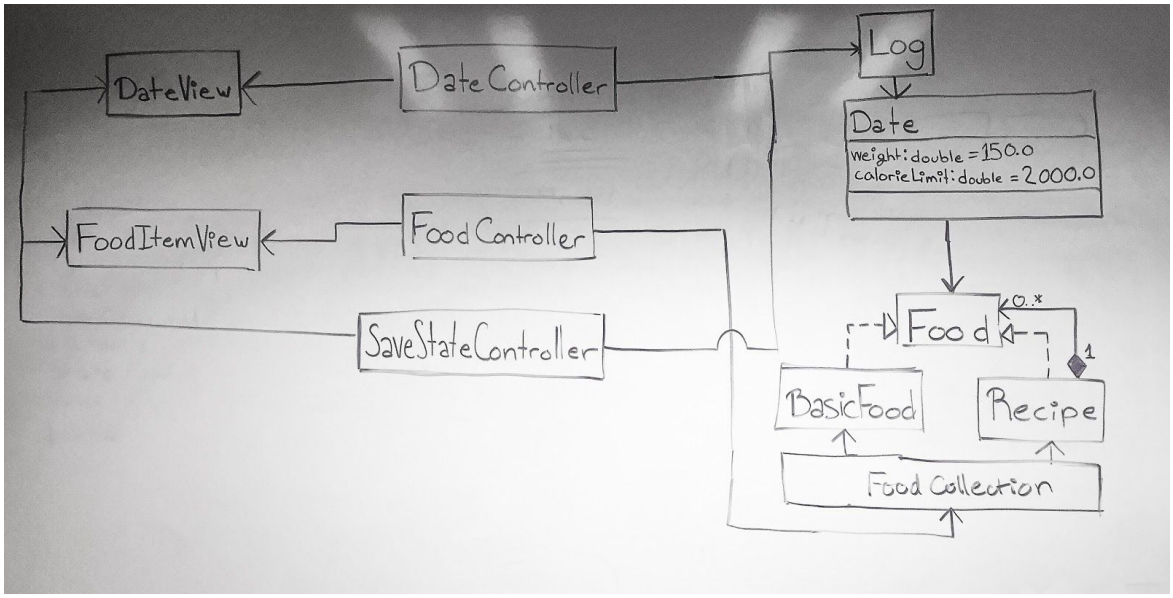
Class Diagram



Models

Log: Contains a list of all Date Objects created

Date: Handles a set weight and calorie limit for a given day. The Date class is responsible for tracking the list of foods a user has consumed on a given day.

Food: Methods and properties which will be used by all Food Objects, both BasicFood and Recipe Objects

BasicFood: Defines Food Objects that the user can use in recipes as well as consume.

Recipe: A collection of Food Objects that can be made up of Basic Foods as well as other Recipes.

FoodCollection: Lists a user's BasicFoods and Recipes that they create.

Controllers

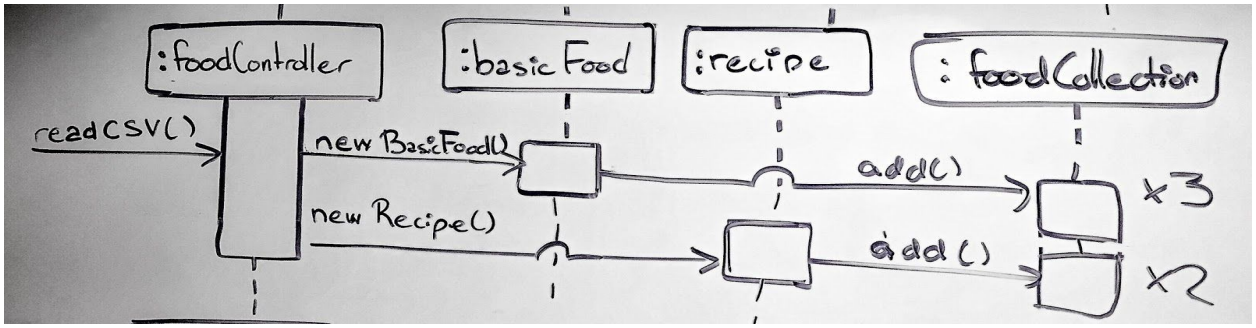
DateController: Gets the Date Objects from the log CSV. Passes that information to the DateView.

FoodController: Gets the BasicFood Objects and Recipe objects that are in the food collection class and passes them to the FoodItemView.

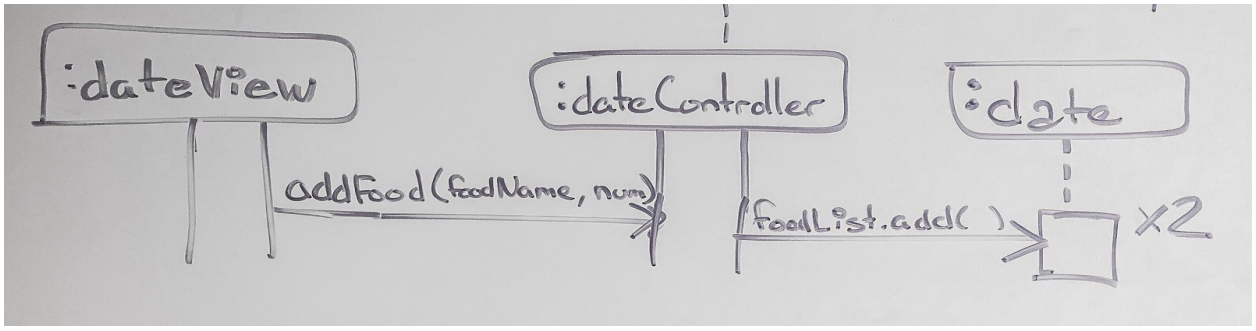
SaveStateController: Gets the information from the Log and FoodCollection lists and saves them out to CSV files upon invocation from any UI View

Sequence Diagrams

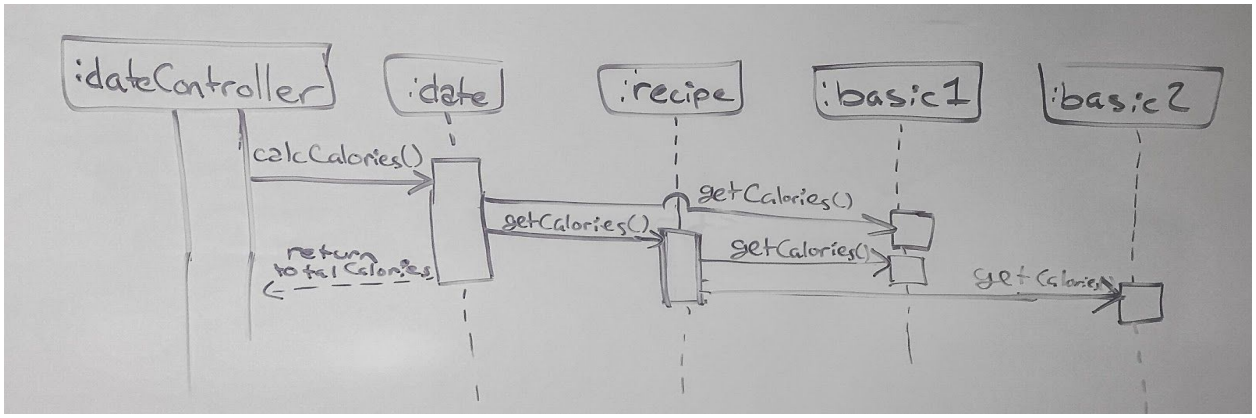
Read in a food database



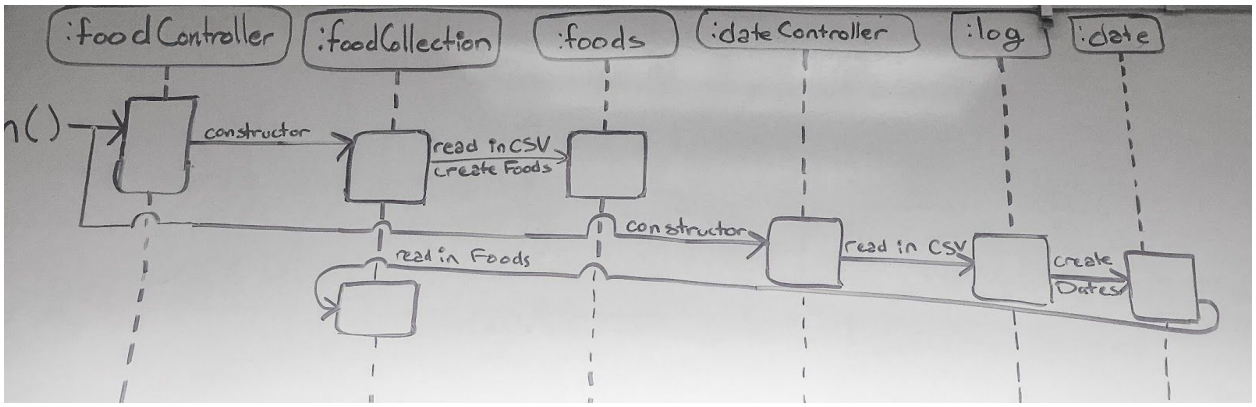
Add a food to a log entry



Compute the total number of calories



Program Start Up



Rational

The organization of our application provides numerous advantages to the flow of data and its organization. For example, we applied the Composite Design Pattern for the food collection-related classes in our model. This allowed us to create two separate classes inherited from an interface that handle BasicFoods and Recipes. Recipes can thus be created from BasicFoods or other already created Recipes. All Foods will share common methods and properties to make manipulation of them all consistent. We also applied the Model View Controller Design Principle in order to separate our application into three distinct areas: one for the user interface, one for the data structure, and one for communication between the two. This will give us different areas to focus on when actually coding the application and maintain separation of concerns.

This design approach our group has ran into some complications. Notable is the instance when the user wishes to create a Recipe. Currently, there is no distinction between when a Recipe is created from BasicFood Objects and other Recipe Objects. Overall our team was able to come up with a somewhat efficient way to organize food data despite issues with nested object types.