Wellness / Release v2.0 Final

Project Design Document

Beep Boop

Justin Newton	<jmn6815@rit.edu>
Aaron Kelly	<axk3897@rit.edu>
Aaron Parker	<ajw2611@rit.edu>
Edward Riley	<emr9018@rit.edu>
Vincent Venutolo	<vxv2326@rit.edu>

Project Summary

The project Wellness is designed to monitor the health status of an individual to provide whether he or she is healthy or not based on what food they ate and when they did recently. As the project is primarily focused on tracking the user's diet to promote or encourage the user to be healthier, this project contains the most basic functions a user needs to manage and to update information in order to produce a result.

Specifically, this program allows the user to add recipes and this program will hold a collection of basic foods. The user can add the name of the food, the number of calories, grams of fat, grams of carbohydrates, and grams of protein in the food.

The user will be able to add the daily log of their food consumption and store it in their data and be able to access it anytime. The user can also add other information such as weight and more. This can be used to determine if the user has accomplished their goal or not.

Design Overview

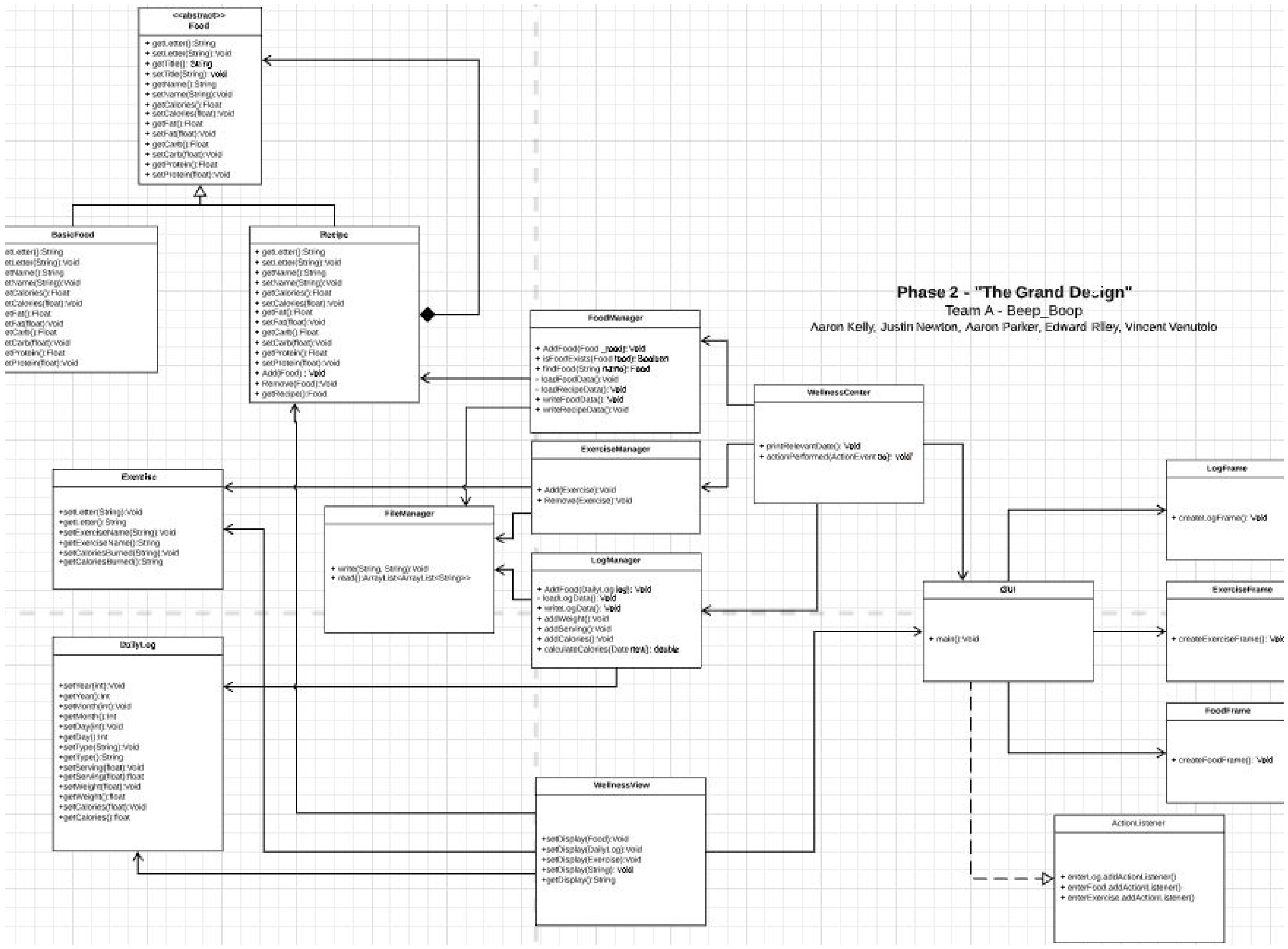
The goal for the project design is to keep change or update the design until it is satisfied the requirements and met the expectations of what it should be. The change in design will be covered here.

The Wellness Project had been undergone the transformation from the beginning with a design sketch include a class and sequence diagram for UML design to the completion. The first draft design was completed and contains the main class, several subclasses rely on it, and their respective data or manager classes. The main class is on Wellness and the other subclasses are User and Recipe classes.

We had been in discussion over whether to meet requirements or considered what a client would be needed. Then the project in which classes are used to set up is changed to have a single class for a response to reading and writing files for both output and input of logs.csv and food.csv. Another class is User and Daily Log that they were configured to function separately so that the User will store only user data and their profile. Daily log class is set for monitoring nutrition status whenever the user eats food for each day. Three other classes which originally had a separate class in which one is responsible for collecting food called FoodCollection is now replaced with Recipe class and is treated as a composite for IFood and BasicFood classes. Thus, FoodCollection was removed and Recipe class is now a sub-main class for the Wellness project for storing and updating data of BasicFood and Recipe.

As the first draft design is completed, it had been taken several changes over time. We applied the patterns to change the class diagram are composite and MVC patterns. The composite pattern was used to treat a simple and complicated object as a simple object when reading for a client. In the project, it applied to several classes for collecting and storing basic foods and recipes: BasicFood, Recipe, and Food. BasicFood is considered a leaf node while Recipe can be either treated as a simple food or recipe contains another simple food. Food is an interface or abstract class, which is a component, used to provide some data for both leaf and composite. For MVC, several majors classes are set to separate based on the following order of MVC. The program started with a model to initiate date for the view to set. And the view will send any actions made by the user to a controller where they are to configure data in model.

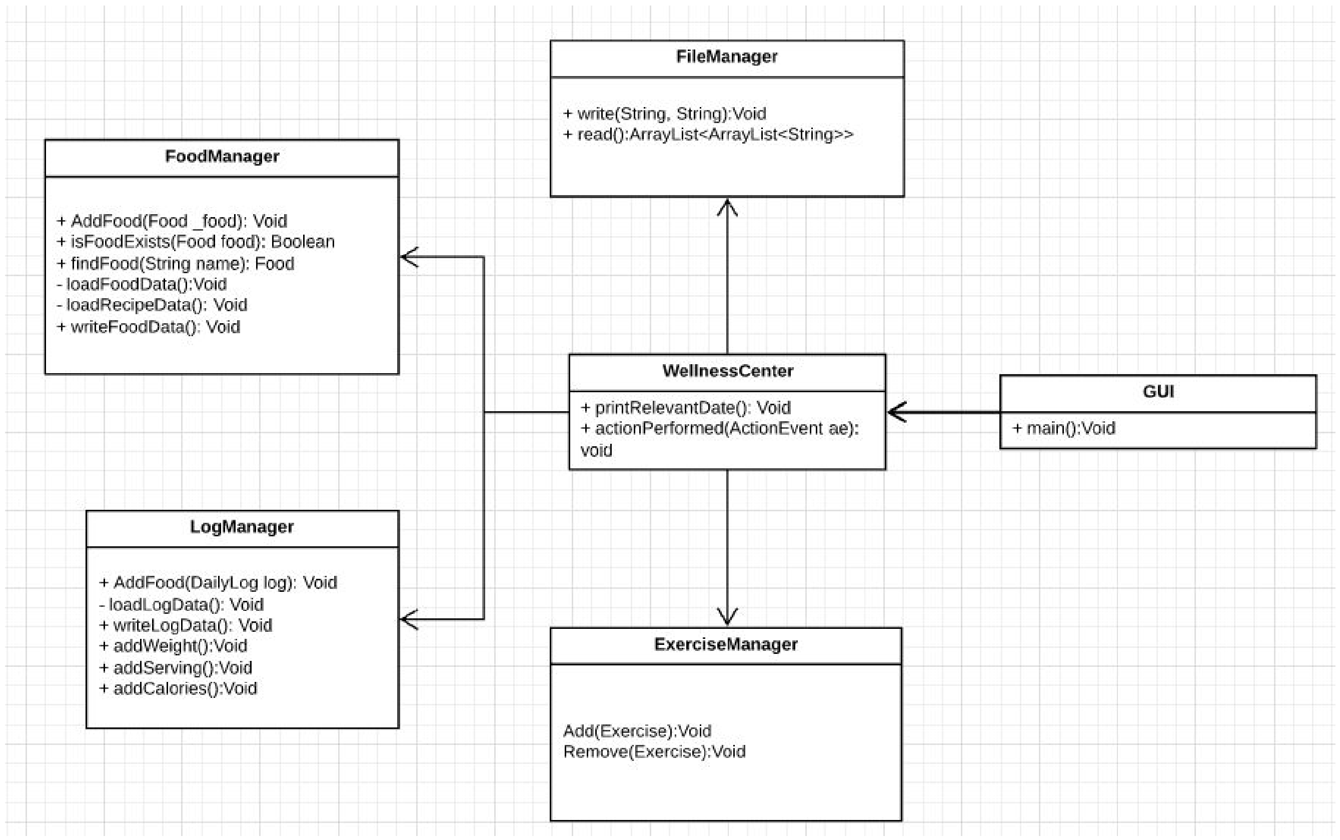
Subsystem Structure


Phase 2 - "The Grand Design"
Team A - Beep_Boop
Aaron Kelly, Justin Newton, Aaron Parker, Edward Riley, Vincent Venutolo
<<abstract>> Food
BasicFood
Recipe
FoodManager
ExerciseManager
FileManager
LogManager
WellnessCenter
Exercise
DailyLog
WellnessView
GUI
LogFrame
ExerciseFrame
FoodFrame
ActionListener
+ printRelevantData(): Void
+ actionPerformed(ActionEvent tle): Void
+ main(): Void
+ createLogFrame(): Void
+ createFoodFrame(): Void
+ write(String, String): Void
+ read(): ArrayList<ArrayList<String>>

This section provides a general overview of all high-level class subsystems that which will be to comprise all together as a whole application. There are a total of 3 high-level classes are the wellnessCenter, Managers for three different data, and GUI main frame.

To start with the main class, Wellness is the center of the Java implementation and is considered the top of the highest level class among classes. It contains many important functions for main functions such as issue a command for information, request specifically which manager to update which data needed to change. This class have the full access to all managers for three different data which are ExerciseManager, FoodManager, and LogManager. Below is a graphical model of the Wellness class and its inheritance to the Wellness:

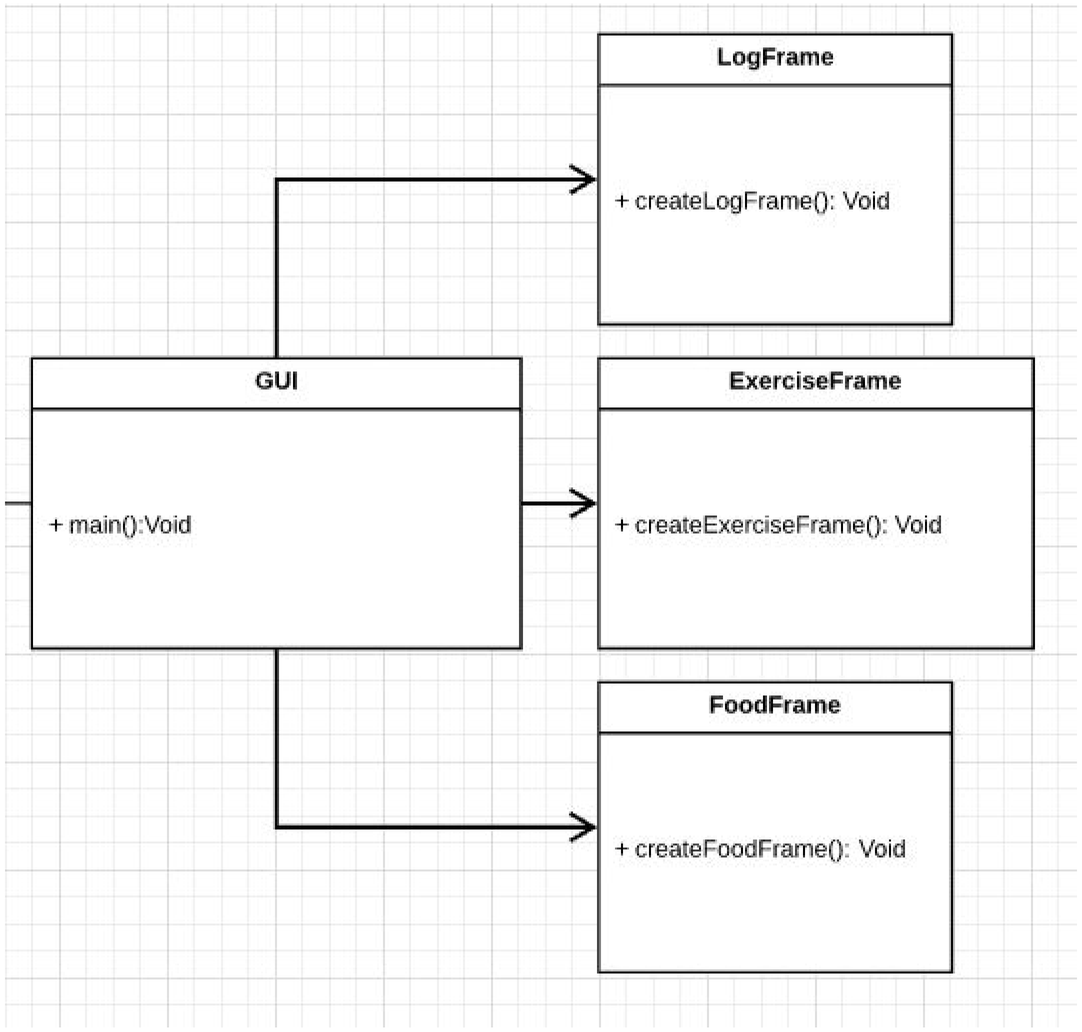
Wellness Figure



Start with the Wellness main class, it is responsible as a command center to exchange information and to update view for the user interface. In order to reduce the degree of dependency on another class or itself for the three different data, the main class had been split into three controllers in which each is responsible for their data. By using pattern above and some of the MVC pattern, this had increased the coupling system in improving for positive consequences. LogManager, FoodManager and ExerciseManager each responsible for configuring data in Log, Food or Recipe, and Exercise respectively.

The third high level subsystem structure is the main user interface, or GUI. The program using the javax.swing.* for the purpose of user interface to provide where user can see what are data now and what changed. In Java implementation, the main user interface contains three different frames to set up based on what type of information user click on. There are LogFrame, ExerciseFrame, and FoodFrame which will ask a very specific data of what each one does requires and nothing else. Having three frames to set up for the main UI is improving for the degree in cohesion when come to use only javax.swing.* for a single look purpose.

User Interface in Java UML



As the picture above shows, it provides a connection to the main station which is Wellness and contains major functions to provide what an individual needs. As the individual request for information to appear, it will command the main station to order any needed functions to process information to bring it up. For the user interface, there will be a new user interface for Graphical User Interface on the next patch to provide a look and responsive feels.

Subsystems

Class BasicFood	
Responsibilities	A model that represents temporary data storage use for the wellness for major functions.
Collaborators (uses)	Food - the interface that is implemented by Basic Food and considers a leaf.

Class Food (abstract)	
Responsibilities	Provide a generic abstract to basic food & recipe. Includes the name, calories, carbs, protein, weight.

Class Recipe	
Responsibilities	A Composite model that can add basic food & recipe forming a composite pattern and represents temporary data storage use for the FoodManager for major functions.
Collaborators (inheritance)	Food - the interface that is implemented by Recipe and considers a composite.

Class WellnessCenter	
Responsibilities	A controller class that will call one of three controller classes that will manipulate the data to one of models. Forming an MVC..

Class WellnessView	
Responsibilities	A view that request the data from one of the models and display the data to UI.

Class DailyLog	
Responsibilities	A model that represents temporary data storage use for the LogManager for major functions.

Class Exercise	
Responsibilities	A model that represents temporary data storage use for the Exercise Manager for major functions.

Class Exercise Manager	
Responsibilities	A class that calls and instantiate ArrayList that accepts Food as a data type that stores the data. Call file manager to write data to file when the user is done. Also, call the file manager to load data from the file.

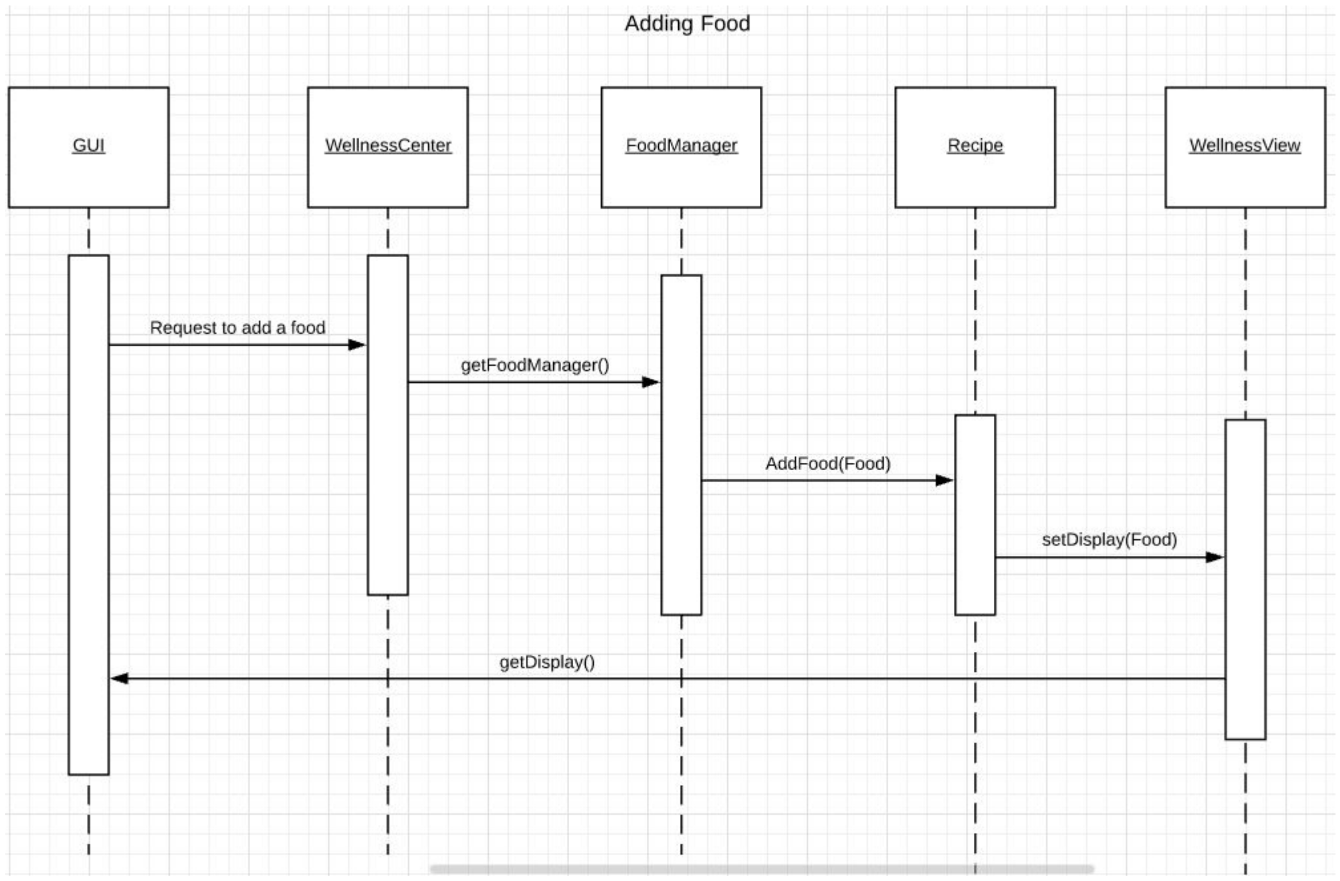
Class Log Manager	
Responsibilities	A class that calls and instantiate ArrayList that accepts DailyLog as a data type that stores the data. Call file manager to write data to file when the user is done. Also, call the file manager to load data from the file.

Class Food Manager	
Responsibilities	A class that calls and instantiate ArrayList that accepts Food as a data type that stores the data. Call file manager to write data to file when the user is done. Also, call the file manager to load data from the file.

Class File Manager	
Responsibilities	Write to file Read from file

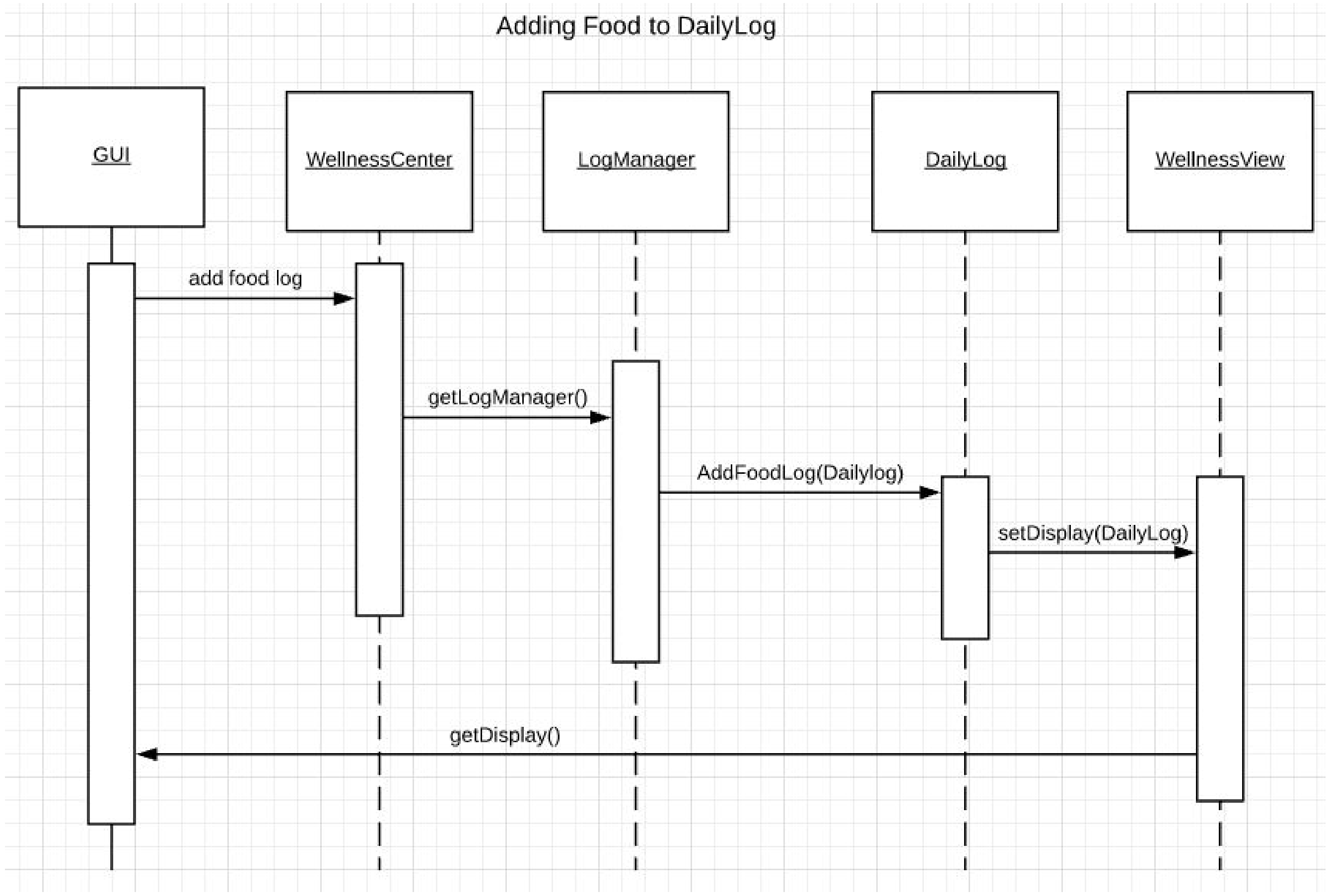
Sequence Diagrams

Part #1: Adding Food



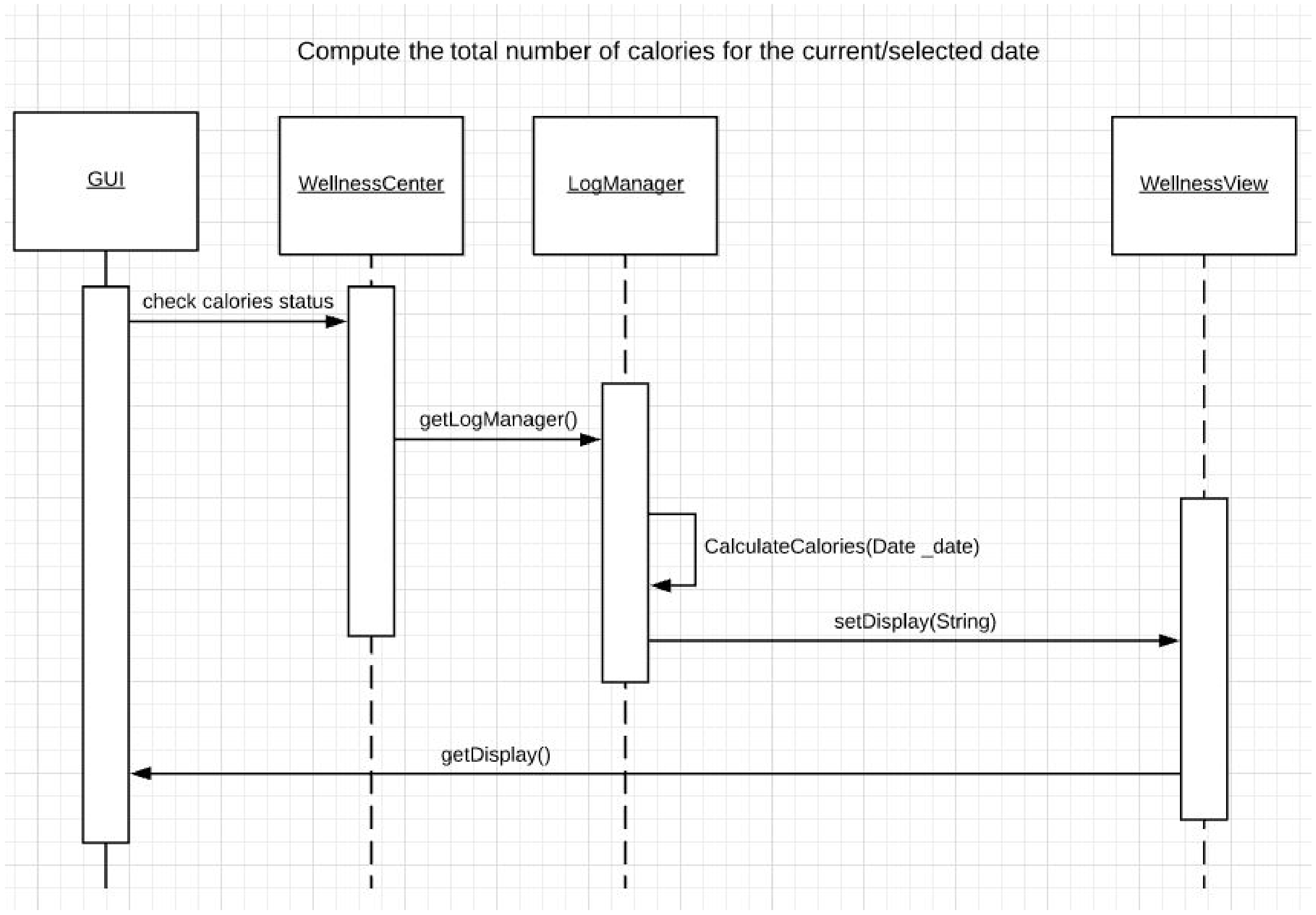
This part of the sequence diagram represents the software the process of user adding a food to the list. When the user request to add a new food, the program will load the food data from the csv file into the program for the user to see the list of food they have. This is done by calling the loadFoodData() method from the FileManager class. After that, the WellnessCenter calls the setDisplay() method from the WellnessView to set up the GUI to be able to display the list of food. Finally, the user will be able to set up the name, number of carbs, and more. Then the program will call AddFood() from recipe then Add() in the BasicFood class.

Part #2: Adding Food to Log



This part of the sequence diagram represents the software in a process of adding a log to the list. When the user wants to add a log, the program will load the log data from the csv file into the program for the user to see the list of log they have. This is done by calling the loadLogData() method from the FileManager class. After that, the WellnessCenter calls the setDisplay() method from the WellnessView to set up the GUI to be able to display the list of foolog. Finally, the user will be able to set up the date, number of carbs, and more. Then the program will call addFoodLog() in the DailyLog class.

Part #3: Compute the total calories for current/selected date



This part of the sequence diagram shows the software processing in calculating the total calories based on the current date of log. Whenever a user requests to see or know how many calories they wanted or needed, it will issue to the WellnessCenter to initialize the object for the LogManager to get calculator function. As the logManager executed the method CalculateCalores(), it will return the result in a double which would be used to set the display to show the result for the WellnessVIew. This was done through the method setDisplay() from WellnessCenter to getDisplay() in WellnessView to update for the view.

Pattern Usage

Composite

Composite Pattern	
Composite	Recipe
Leaf	BasicFood
Component	Food <abstract>

Model-View-Controller

Model-View-Controller Pattern	
Model	Exercise BasicFood Recipe DailyLog
View	WellnessView
Controller	WellnessCenter Food Manager Log Manager Exercise Manager

Observer

Observer Pattern	
Observer	GUI
Observable	WellnessCenter