PROJECT / RELEASE 2

Project Design Document

Team A

Brianna Buttaccio <bab7607@rit.edu>
Caitlyn Daly <cnd9351@rit.edu>
Tiffany Ellis <tae7612@rit.edu>
Roy Tran <rxt7649@rit.edu>
Kevin Reynolds <kmr1188@rit.edu>
Fu Quan Li <fxl2328@rit.edu>

Project Summary

The project's goal is to create an application that allows a user to keep track of personal health information. This information is important to users who wish to begin or maintain a healthy lifestyle. One of the most essential components to a healthy lifestyle is nutrition and keeping track of what you are consuming. Too much of anything is a bad thing and therefore users must be able to ensure they are maintaining a healthy limit on food intake. The ideal use of this application will have users update their information on a daily basis to track their health. Users will able to add foods and recipes to a collection which will store nutritional information including grams of fat, calories, carbohydrates, and proteins per serving. Foods are single items such a banana or an apple that may be consumed by itself or used as an ingredient for a combination of foods. This combination of foods is called a recipe, which have different ingredients that all have their own nutritional information that will be added together to create a total of nutritional information.

Maintaining a steady calorie count can be vital to losing or gaining weight depending on what a user's goal weight. The user will have to enter how much they weigh in case they want to see any changes in weight which can be an indicate health improvement or degradation. Calories are consumed when we eat food and burned whenever we do physical activities, the more intense the activity, the more calories are burned. Regular exercise is known to be essential when maintaining a healthy lifestyle to not only regulate the body but also expend excess calories. Not exercising enough can lead to many health issues such as heart disease, whereas over exercising can cause physical issues such as joint deterioration. The user will be able to see various interactions between calories for exercising. Such as the difference in the calories consumed and calories burned, which will give the user a net calorific value. The exercise will also include how many minutes the exercise was completed in.

The collection of foods and exercises can be referenced by the user when creating and adding information to their daily logs. The daily logs store all the information of the user on that indicated date. This includes foods consumed along with nutritional information, exercises and calories burned per exercise. The daily log is mainly used to show the user all the calorie information. Daily calorie limits can be chosen by the user in order to assist them with accidental overconsumption of calories. This includes the total of calories consumed, total calories burned, net calorific value, calorie goal indicated by the user, and the difference between the net calorie and goal calorie values. The weight of the user is record daily, as weight gain or loss can be an indication of a successful or unsuccessful healthy lifestyle. This is often the reason why people try to start a new healthy lifestyle.

This application will be displayed to the user through a graphical interface. The graphical user interface will be in panels that are separated into clickable tabs. The tabs allow the user easier access to different parts of the application. The user will be able to enter and delete information based on which tab is selected. The Home tab will be a dashboard that displays daily log of the date selected by the user. This includes all the information in the daily log and a bar graph representing total grams of fat indicated by red, total carbohydrates indicated by green, and total proteins indicated by blue. The Log Foods tab have two sub tabs, one to add a daily food

log and one to delete a daily food. The Add Food sub tab will let the user enter how many servings of that food consumed. The Log Exercises tab also has two sub tabs that add or delete the exercise. The user will have to enter how many minutes of that exercise was completed in. Both Log Foods and Log Exercises will be relative to the selected date indicated on the home page. If the user does not enter information for this date, the log will be blank for food and exercises. The Log Weight & Calories tab records the daily weight and calorie limit given from the user. If the user does not enter values, the default weight will be one hundred and fifty pounds and the default calorie limit will be two thousand. The Create Food tab will have sub tabs for creating a food or a recipe. The Create Food sub tab will take in the name and nutritional information from the user. The Create Recipe tab will require the user to include ingredients in the recipe which will added together. The last tab Create Exercise, will take in the name of the exercise and flat rate of Calories burned for one hour for a hundred pounds. Both the Create Food and Create Exercise tabs will allow the user to delete entries as well.

Design Overview

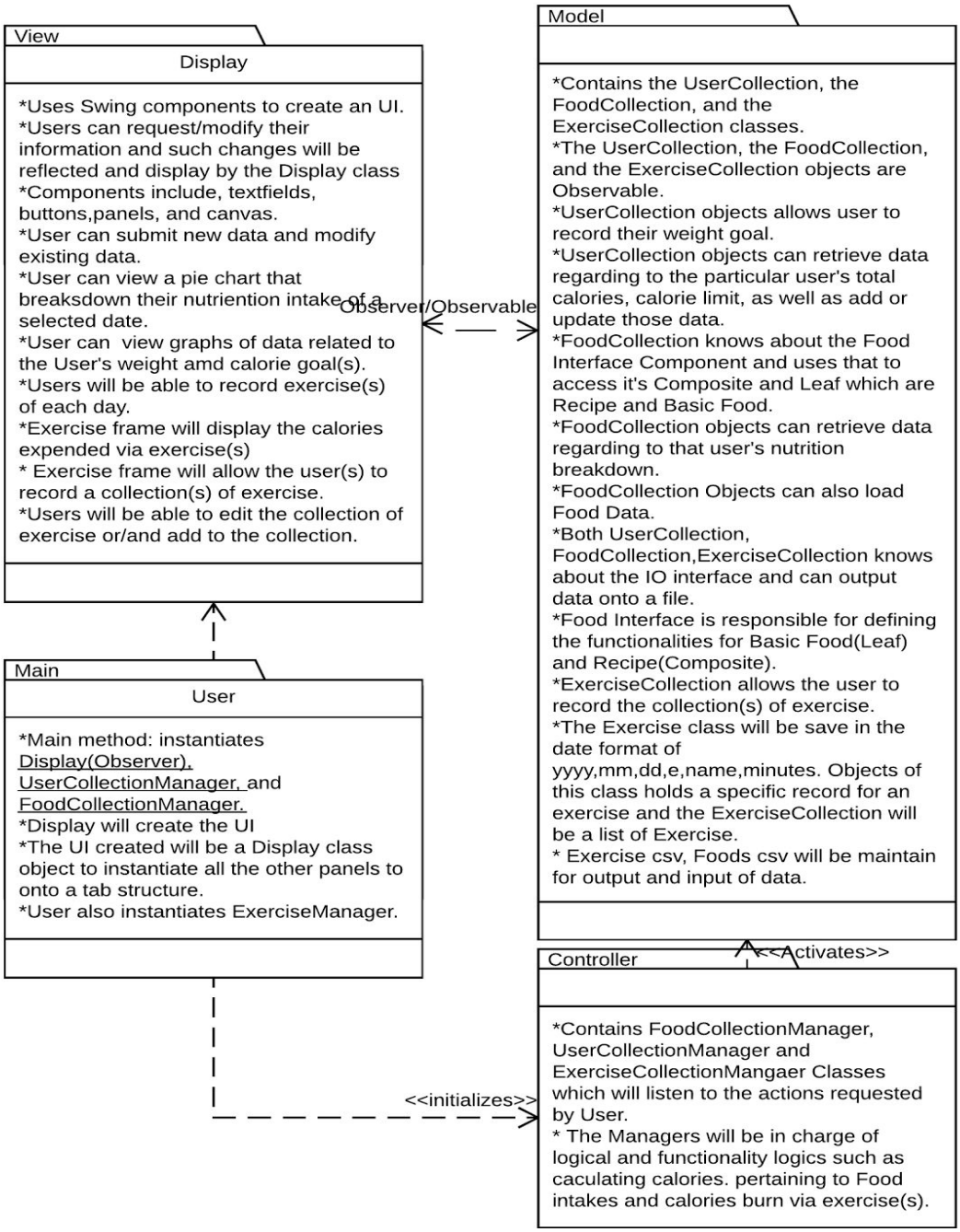
For the design of our project we decided to have the user be the main class. Our project follows the MVC pattern, so the user can see and knows about the view and controller classes which are the Display, LogCollection, ExerciseCollectionManager, and FoodCollectionManager. The user just initializes the classes. The Display will function as our view. It will know about and be able to call on other classes that will act as widgets that can be reused in different parts of the program. These widgets will include a line graph, food consumption list, food and recipe addition and deletion, and a delete button. We wanted to keep all of these widgets separate so that we have good separation of concerns, high cohesion and low coupling. We will also be able to re-use specific widgets on whatever page we need them, and they will change depending on our needs. All of the widgets that include changing data such as food lists and graphs act as observers. All widgets that trigger an action will trigger functions in the controllers.

The LogCollectionManager, ExerciseCollectionManager, and FoodCollectionManager are classes that act as our controllers. They will handle all of the logical functionality that comes with the user's nutritional, daily log data and food and recipe data respectively. The controller will use ActionListeners to run functions when the user clicks on buttons and enter data in the view. Some of the classes that the controller will call upon are the food log, LogCollection, and our io interface. The io interface will allow the food log and LogCollection have access to our csv io which is in charge of reading and writing information to our csv file to store data. The reason that we decided to have an IOInterface and a CSVIO class is so that is the datatype (csv file) ever changes a new class can be written to parse that data specifically without having to change the rest of the files. Originally we had the io class writing and reading to the csv file directly, however by separating the classes we were able to create better abstraction in our program.

LogCollection, FoodCollection, ExerciseCollection, Food, Basic Food, Recipe, CSVIO, IO Interface are our models. Each class is in charge of recording their respective data. The model uses the Observable pattern so that the view can update the display when data changes in the model. All of these files are use the controller to handle the logic and business rules of changing data. The model can see the IO interface and the CSV files so they can store the data

permanently in files. Food is also part of the model. Food represents a basic food or recipe object so that the user can create a food once and reuse it. Food acts as our component in the composite pattern. The component is the Recipe and the basic food is the leaf.

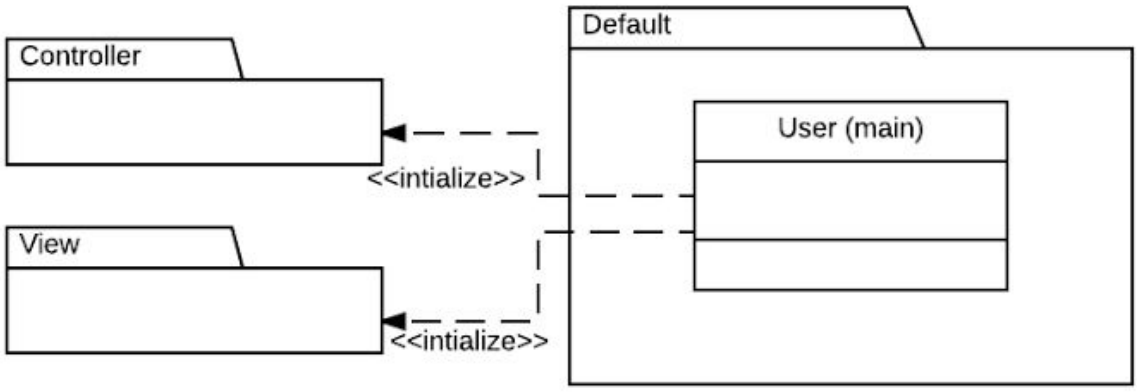
Subsystem Structure



Subsystems

Default

User Main	
Responsibilities	Initializes FoodCollection Manager, LogCollection,ExerciseManager classes in the Controller subsystem. Initializes Display class in the View subsystem.
Collaborators (uses)	controller.FoodCollectionManagerger controller.LogCollectionManager controller.ExerciseManager view.MainDisplayAll



Model

FoodCollection	
Responsibilities	Interfaces with IO interface to read and save Food. As well as read and save Exercise. Observed by the display.
Collaborators (uses)	Model.IOInterface View.MainDisplay

LogCollection	
Responsibilities	Interfaces with IO interface to read and save user food stats. Observed by the display.
Collaborators (uses)	Model.IOInterface View.MainDisplay

Daily Log	
Responsibilities	Logs the information of the users for a specific day.
Collaborators (uses)	Model.Food View.MainDisplay

IO Interface	
Responsibilities	Recieves data to load or save from either User or FoodCollection classes.
Collaborators (uses)	Model.LogCollection Model.FoodCollection Model.CSVIO

CSVIO	
Responsibilities	Saves and loads data from the csv log files.
Collaborators (uses)	Model.IOInterface

Exercise	
Responsibilities	An entry for a specific instance of exercise recorded in a specific date format.
Collaborators	Controller.ExerciseManager

(uses)	

ExerciseCollection	
Responsibilities	A collection/list of Exercise. Maintains and load/save the collection to the Exercise CSV.
Collaborators (uses)	Controller.ExerciseManager Model.Exercise

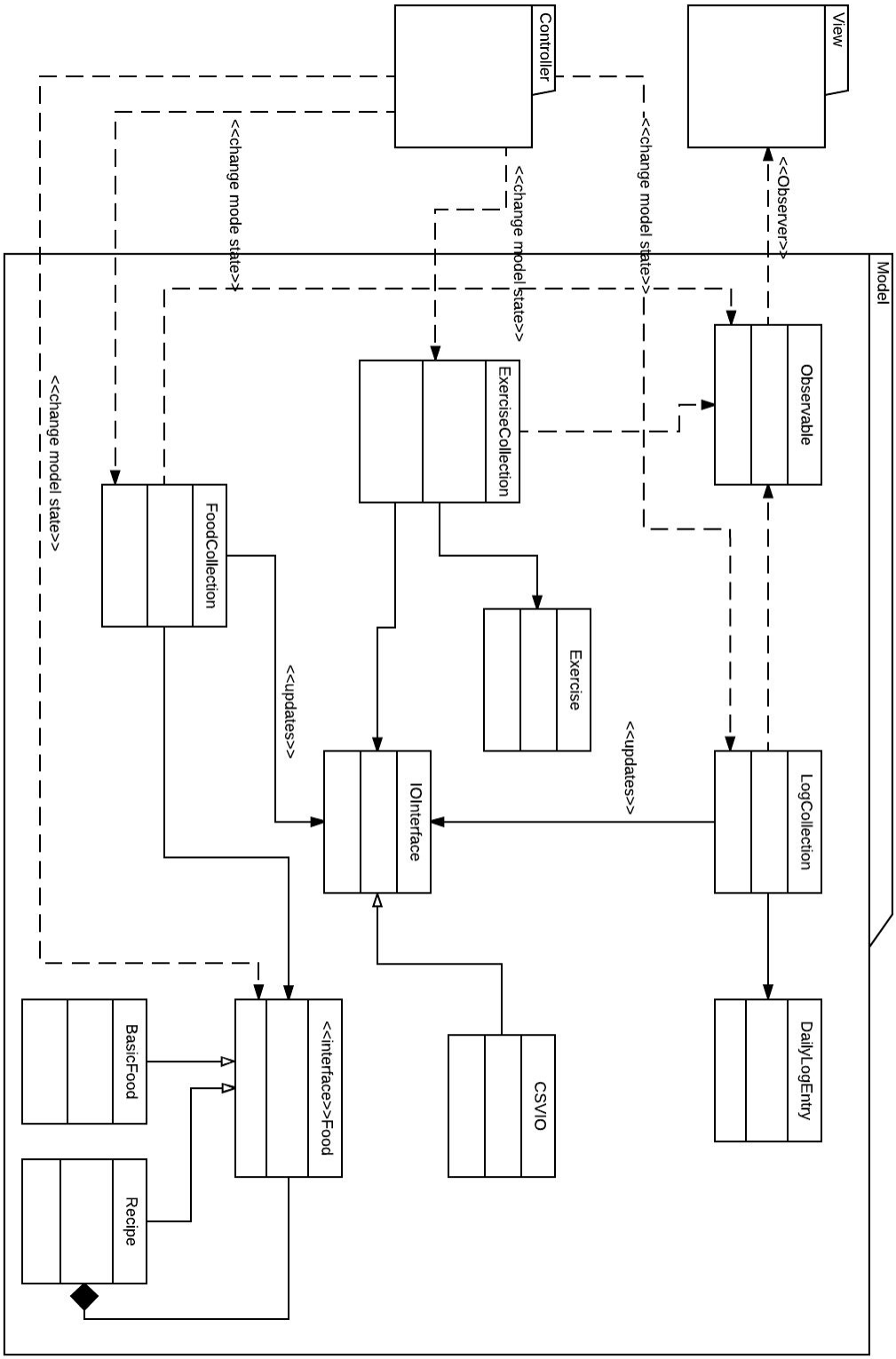
Food	
Responsibilities	Keeps track of the different foods (basic foods and recipes). Recieves data from FoodCollection Manager.
Collaborators (uses)	Controller.FoodLogManager

Basic Food	
Responsibilities	Holds a basic food object
Collaborators (uses)	Model.Food

Recipe	
Responsibilities	Holds a recipe object created from basic food objects
Collaborators (uses)	Model.Food

WeightBarGraphCanvas	
Responsibilities	Responsible for generating the bar graph for the weight graph.
Collaborators (uses)	Model.LogCollection Controller.LogManager

NutritionBarGraphCanvas	
Responsibilities	Responsible for generating the bar graph for the nutrition.
Collaborators (uses)	Model.LogCollection Controller.LogManager


Controller
View
<<change mode state>>
<<change model state>>
<<change model state>>
<<Observer>>
Model
Observable
ExerciseCollection
FoodCollection
Exercise
<<change model state>>
<<updates>>
<<updates>>
IOInterface
LogCollection
BasicFood
<<interface>>Food
CSVIO
DailyLogEntry
Recipe

View

Panel Interface	
Responsibilities	A general interface that will be extended by other panel classes.
Collaborators (uses)	create.JPanel

MainDisplay	
Responsibilities	Create objects from all the other panel classes.
Collaborators (uses)	create.HomePanel create.LogWeightCaloriePanel create.LogFoodPanel create.LogExercisePanel create.CreateExercisePanel create.CreateFoodPanel

HomePanel	
Responsibilities	The landing page panel for all users after the program launches.
Collaborators (uses)	extend.Panel create.JButton, JTextField, JLabel, JPanel

CreateExercisePanel	
Responsibilities	Creates a new Exercise with basic information of the exercise.
Collaborators (uses)	extend.JPanel create.JButton, JTextField, JLabel, JPanel

LogExercisePanel	
Responsibilities	The Exercise panel for users to add and edit their collection of Exercise onto the log.
Collaborators (uses)	extend.JPanel create.JButton, JTextField, JLabel, JPanel

CreateRecipePanel	
Responsibilities	Creates a panel for creating recipes
Collaborators	view.submitRecipeJButton

(uses)	view.addNewFood/CreateLogFoodPanel

Submit Recipe JButton	
Responsibilities	Submits new recipes to the FoodCollection
Collaborators (uses)	view.CreateRecipeTextViewPanel view.CreateLogFoodPanel

CreateFoodPanel	
Responsibilities	Creates a JPanel for adding recipes or basic food
Collaborators (uses)	view.createRecipeTextViewPanel view.createBasicFoodViewPanel view.LogFoodPanel

Submit BasicFood JButton	
Responsibilities	Submits new recipes to the FoodCollection
Collaborators (uses)	view.CreateBasicFoodTextViewPanel view.CreateLogFoodPanel

Foods Consumed Text View Panel	
Responsibilities	Creates a panel to display which foods have been consumed
Collaborators (uses)	view.HomePanel view.LogFoodPanel view.DailyLog

Calorie Intake and Calorie Limit Text View Panel	
Responsibilities	Creates a panel to display calorie intake and limits
Collaborators (uses)	view.HomePanel view.DailyLog

Over/Under Calorie Limit Text View Panel	
Responsibilities	Displays whether or not the user is over or under their calorie limit

Collaborators (uses)	view.HomePanel

LogFoodPanel	
Responsibilities	Holds all of the calorie and food panels
Collaborators (uses)	view.foodsConsumed view.calorieIntake view.overUnderCalorie view.JDatePicker view.CalorieLimit

JDatePicker	
Responsibilities	Allows the user to pick a date
Collaborators (uses)	view.HomePanel

Calorie Limit Text Field Panel	
Responsibilities	Accepts user input for daily calorie limit
Collaborators (uses)	view.HomePanel view.submitCalorieLimit

Submit Calorie Limit JButton	
Responsibilities	Button for setting the user calorie limit
Collaborators (uses)	view.CalorieLimit

WeightGraphCanvas	
Responsibilities	Graph of weight over time
Collaborators (uses)	view.weightFrame

LogWeightandCaloriePanel	
Responsibilities	A JFrame to hold the weight items

Collaborators (uses)	view.weightGraphCanvas view.WeightTextFieldPanel view.Frame

WeightTextFieldPanel	
Responsibilities	Accepts the user's current weight
Collaborators (uses)	view.weightFrame view.submitWeightJButton

SubmitWeightJButton	
Responsibilities	Submits the current weight in the weight text field
Collaborators (uses)	view.WeightTextFieldPanel

Fat/Carbs/ProteinGraphCanvas	
Responsibilities	Graph of the percent fat carbs and protein for the day
Collaborators (uses)	view.LogFoodPanel

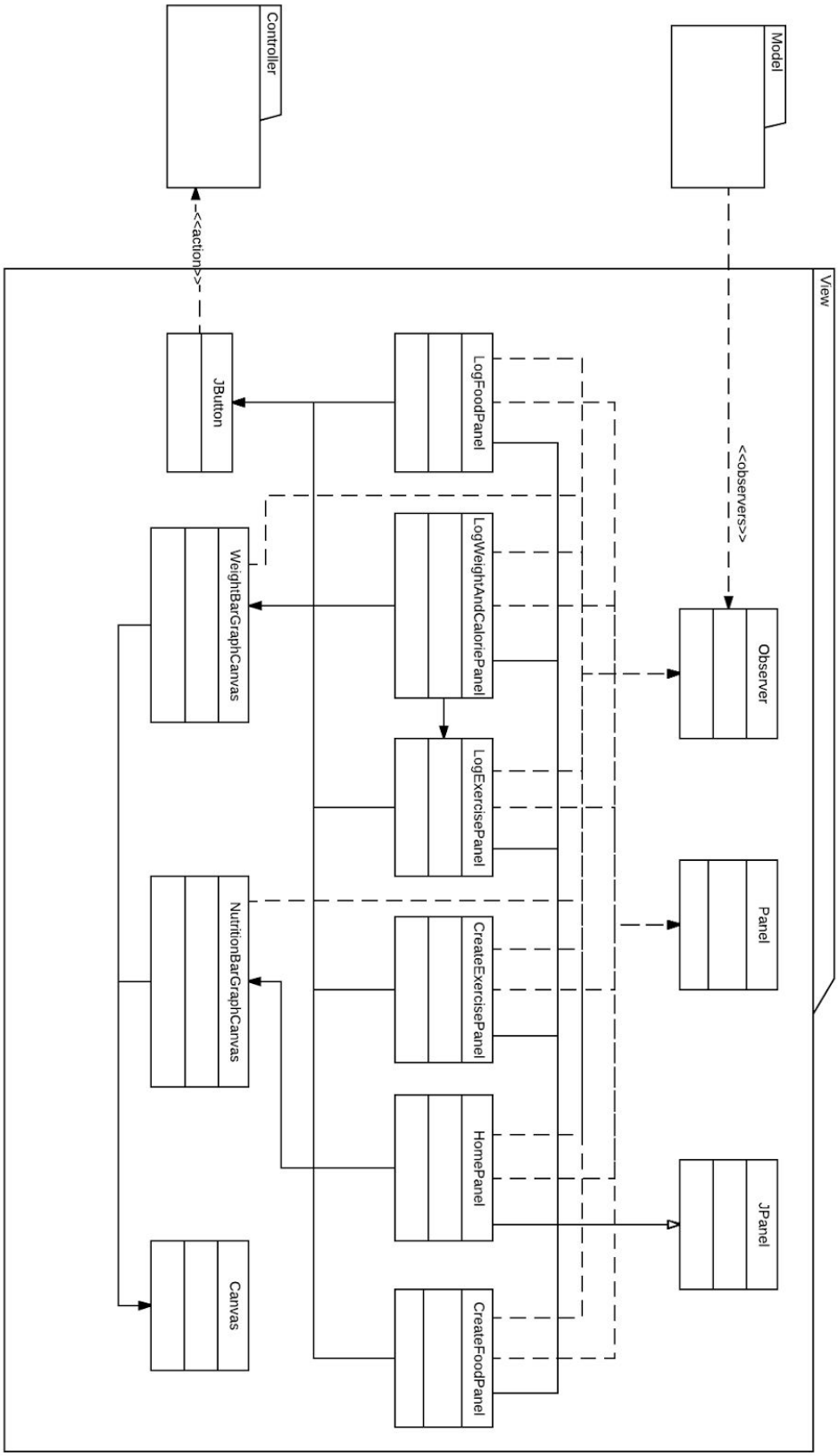
LogFoodPanel	
Responsibilities	A JPanel to add the food items
Collaborators (uses)	view.FatCarbsProteinGraphCanvas view.FoodRecipeTextFieldPanel

Basic Food/Recipe Text Field Panel	
Responsibilities	Accepts a basic food or recipe to be added to LogCollection
Collaborators (uses)	View.LogFoodPanel view.submitFoodJButton

SubmitFoodJButton	
Responsibilities	Submits the current food/recipe in the food/recipe text field
Collaborators (uses)	view.FoodRecipeTextFieldPanel

SubmitExerciseJButton	
Responsibilities	Submits the current exercise in the exercise text field
Collaborators (uses)	view.ExerciseTextFieldPanel

LogExercisePanel	
Responsibilities	JPanel to add exercise to the log
Collaborators (uses)	view.CaloriesExpendedTextField view.MinutesExerciesedTextField

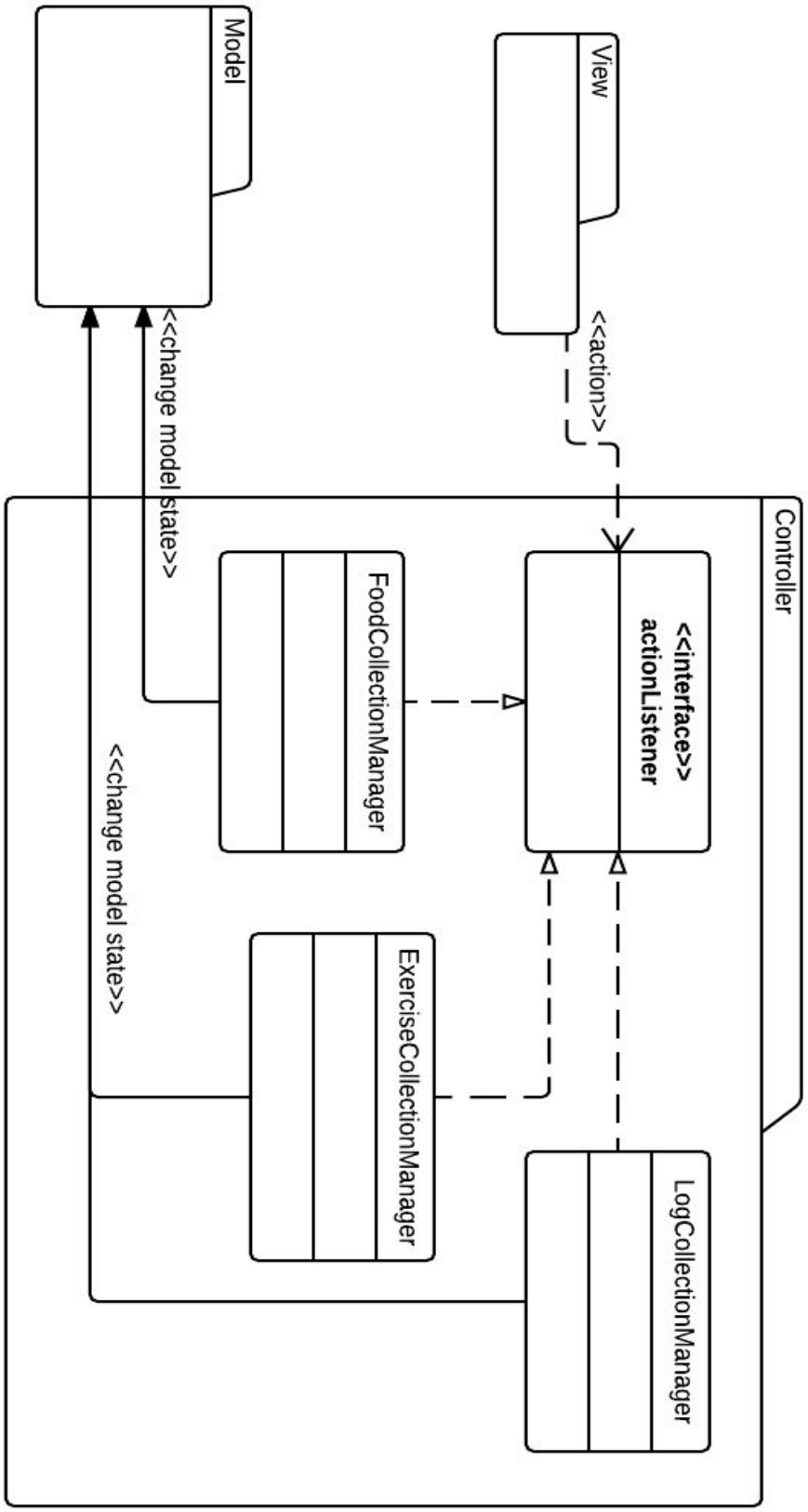

Model
Controller
<<action>>
<<observers>>
View
JButton
LogFoodPanel
LogWeightAndCaloriePanel
WeightBarGraphCanvas
Observer
LogExercisePanel
Panel
NutritionBarGraphCanvas
CreateExercisePanel
HomePanel
JPanel
Canvas
CreateFoodPanel

Controller

LogCollection	
Responsibilities	Initializes and updates LogCollectionclass from user input in the Display class. Calculates the net calories from the ExerciseCollection and the FoodCollection Managers.
Collaborators (uses)	model.LogCollection actionListener

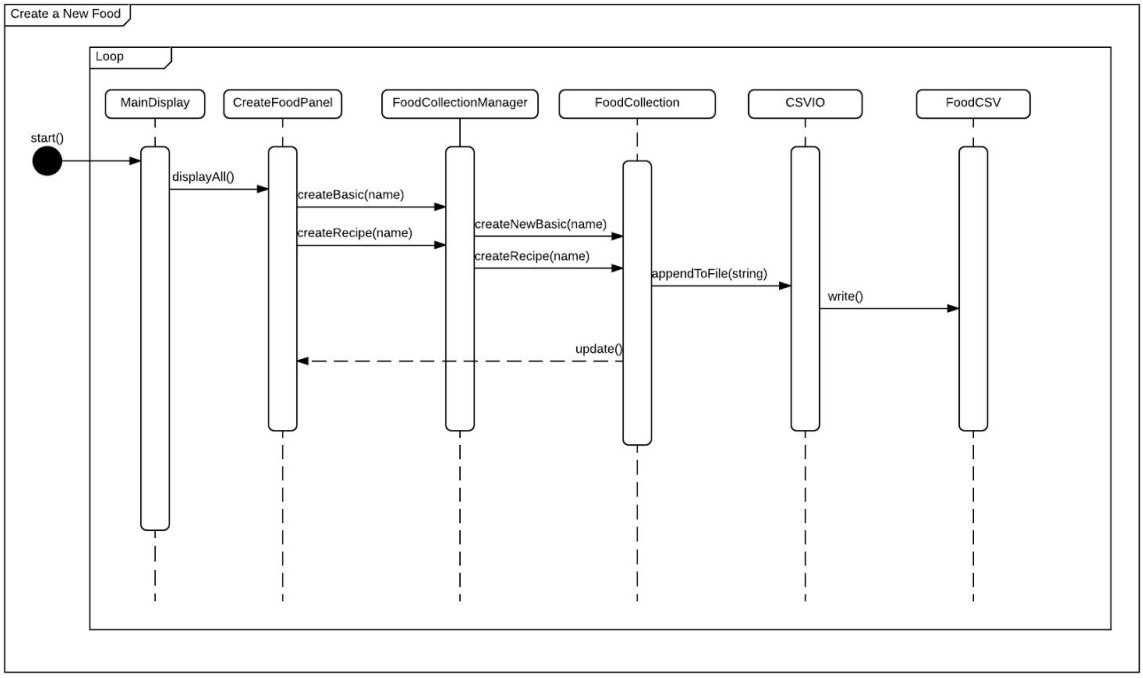
ExerciseCollection Manager	
Responsibilities	Initializes and updates ExerciseCollection class from user input in the Display class. Initializes and creates/updates Exercise components. Calculates all the calories expended from exercises.
Collaborators (uses)	model.ExerciseCollection model.Exercise actionListener

FoodCollection Manager	
Responsibilities	Initializes and updates FoodCollection class from user input in the Display class. Initializes and creates/updates Food components. Calculates all the calories consumed from food.
Collaborators (uses)	model.FoodCollection model.Food actionListener

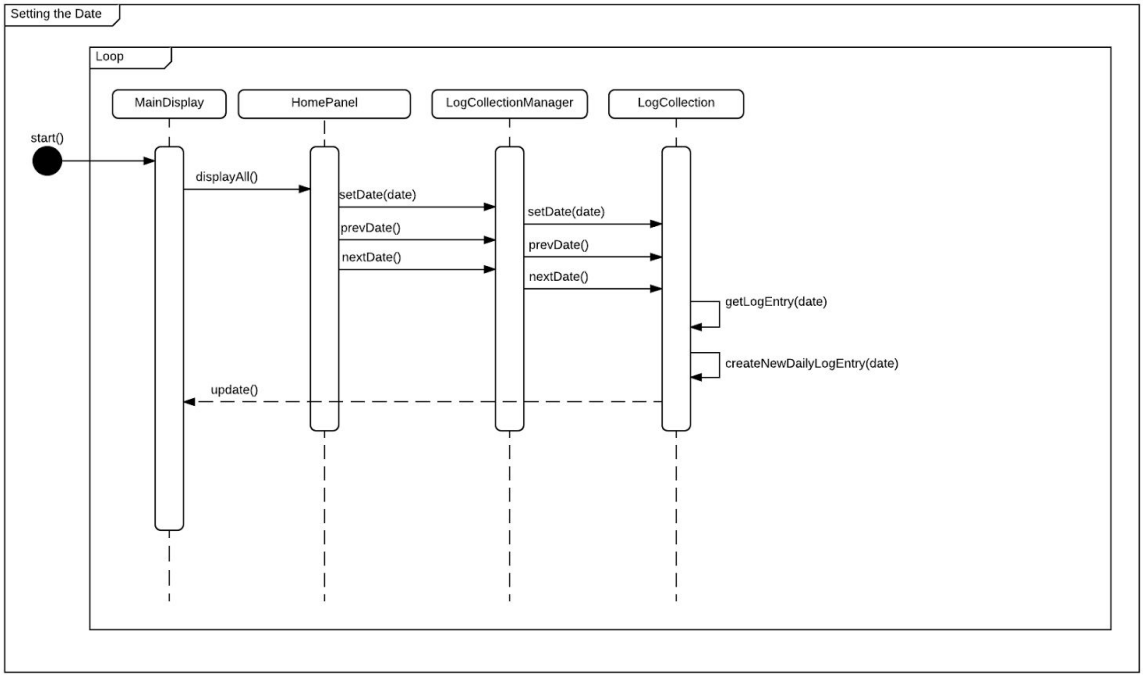

Model
View
<<action>>
<<change model state>>
Controller
<<interface>>
actionListener
FoodCollectionManager
<<change model state>>
ExerciseCollectionManager
LogCollectionManager

Sequence Diagrams

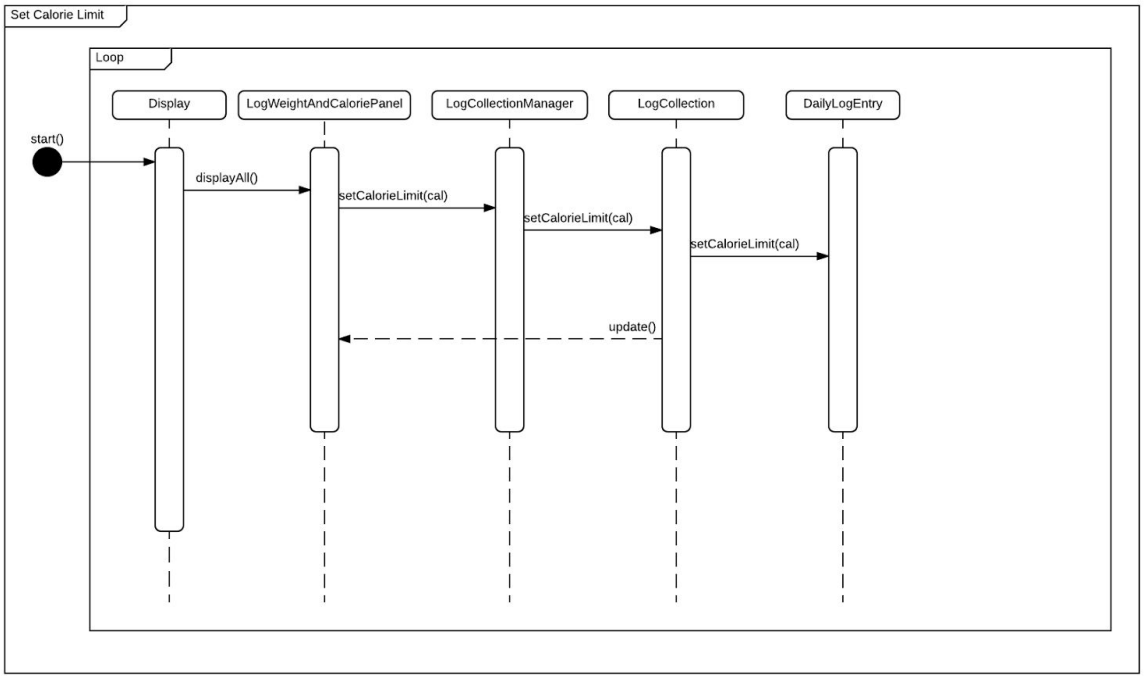
Creating a new basic food or recipe to the FoodCollection



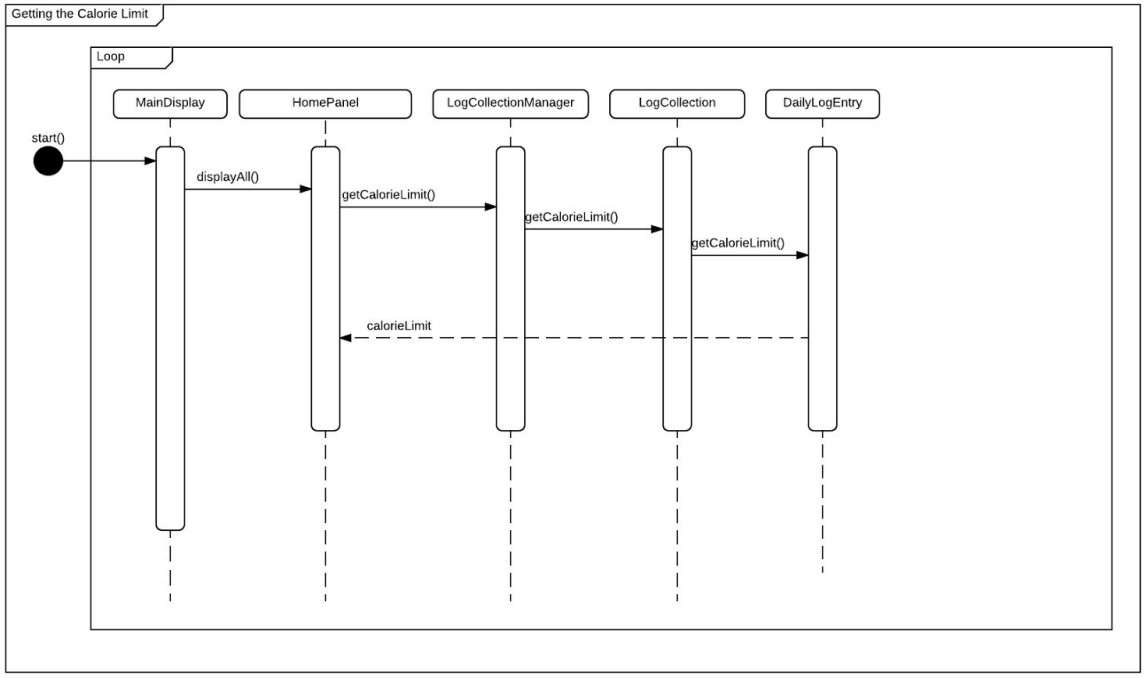
Setting the date



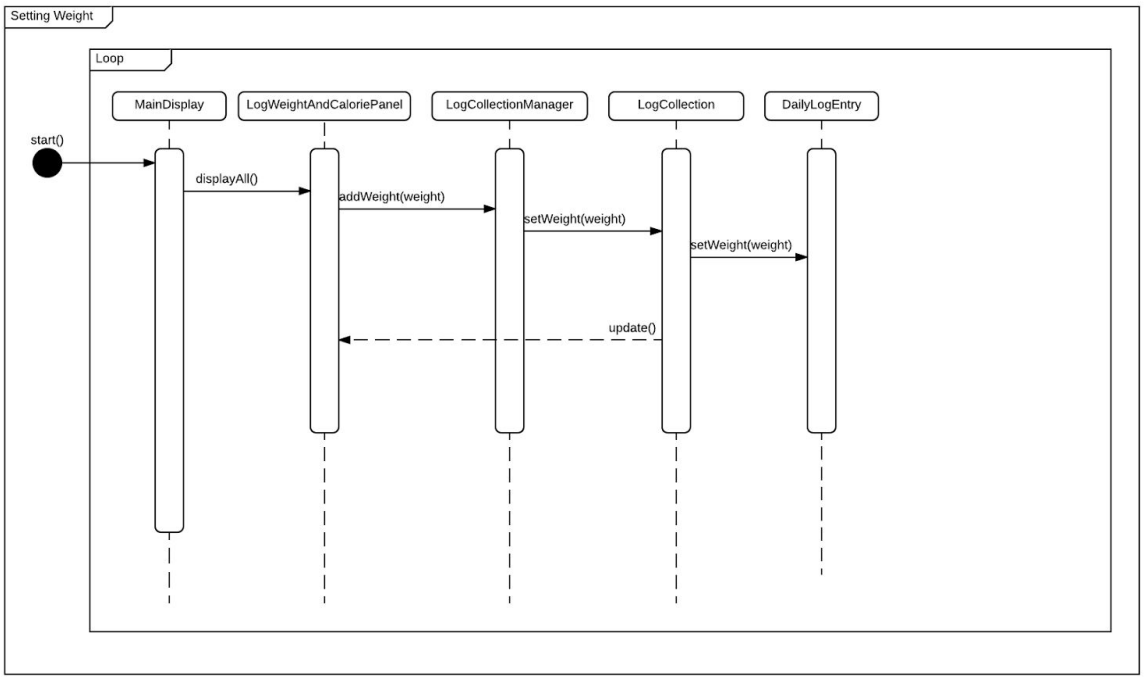
Setting the calorie limit



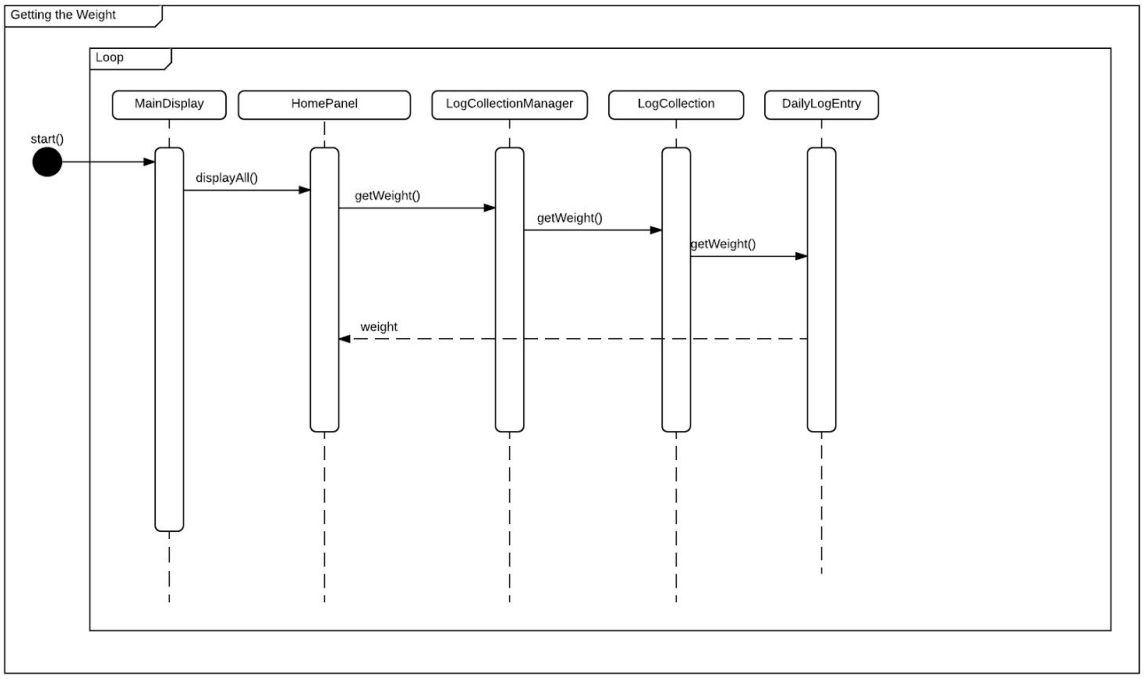
Getting the Calorie Limit



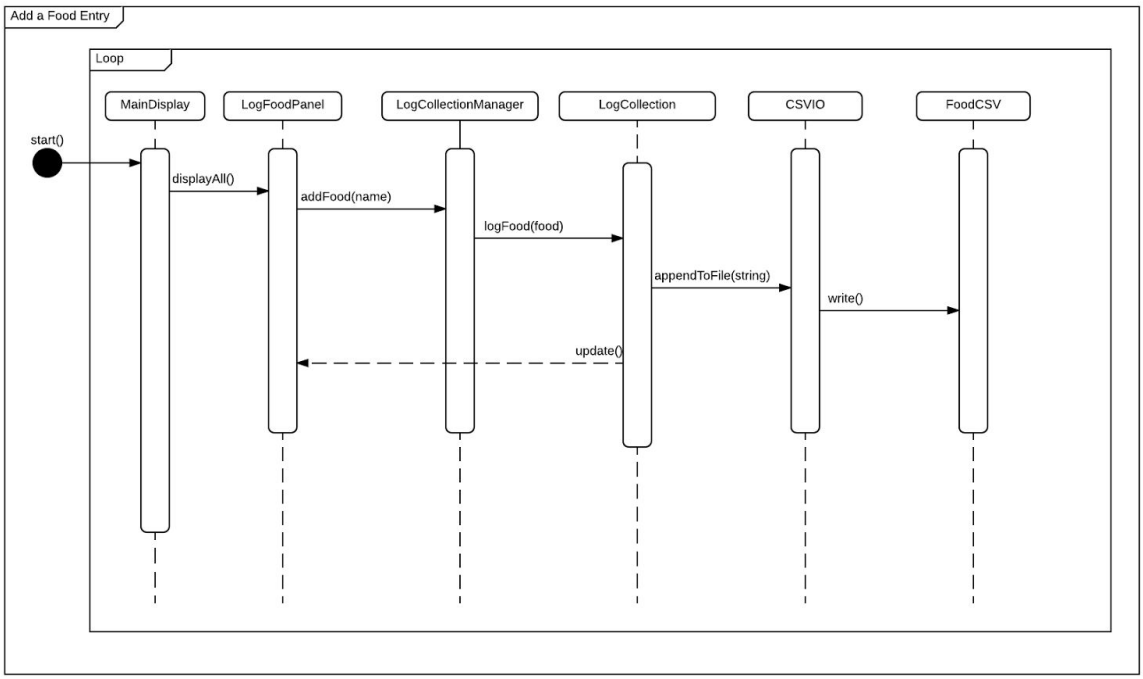
Setting the weight



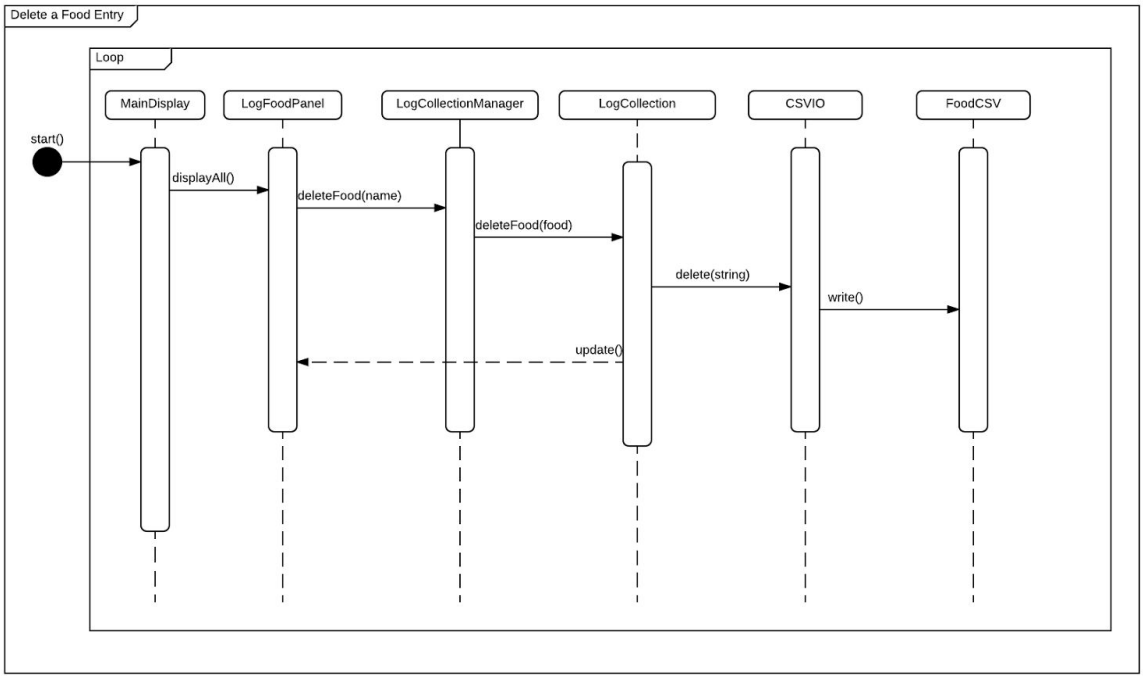
Getting the Weight



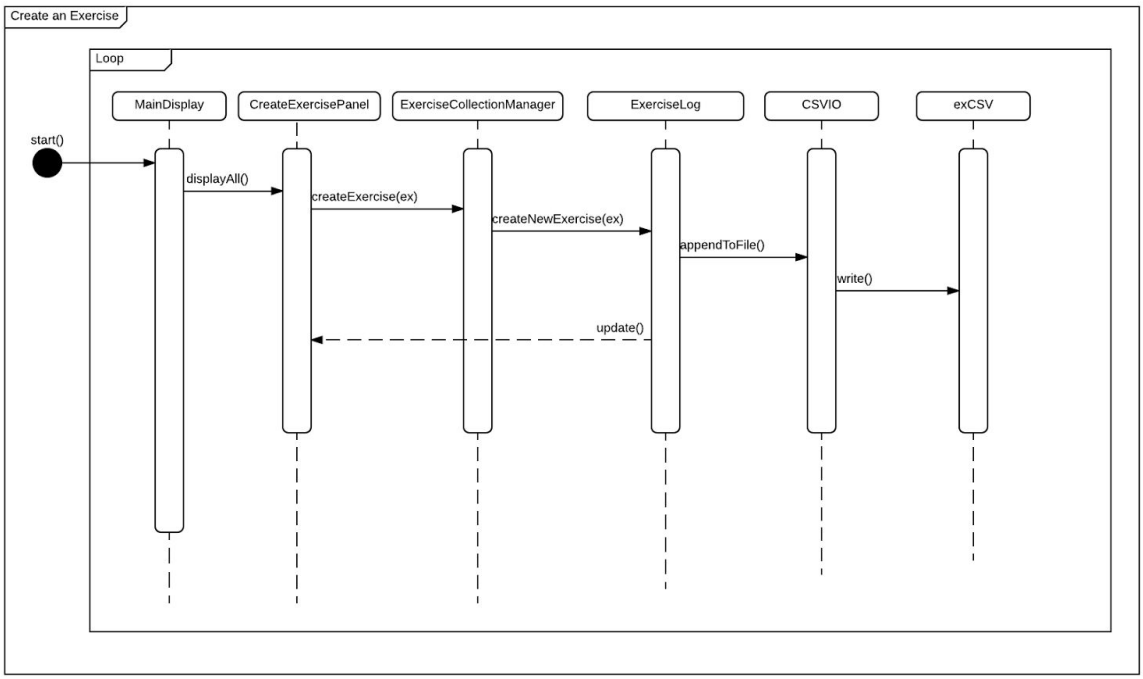
Add food intake to LogCollection



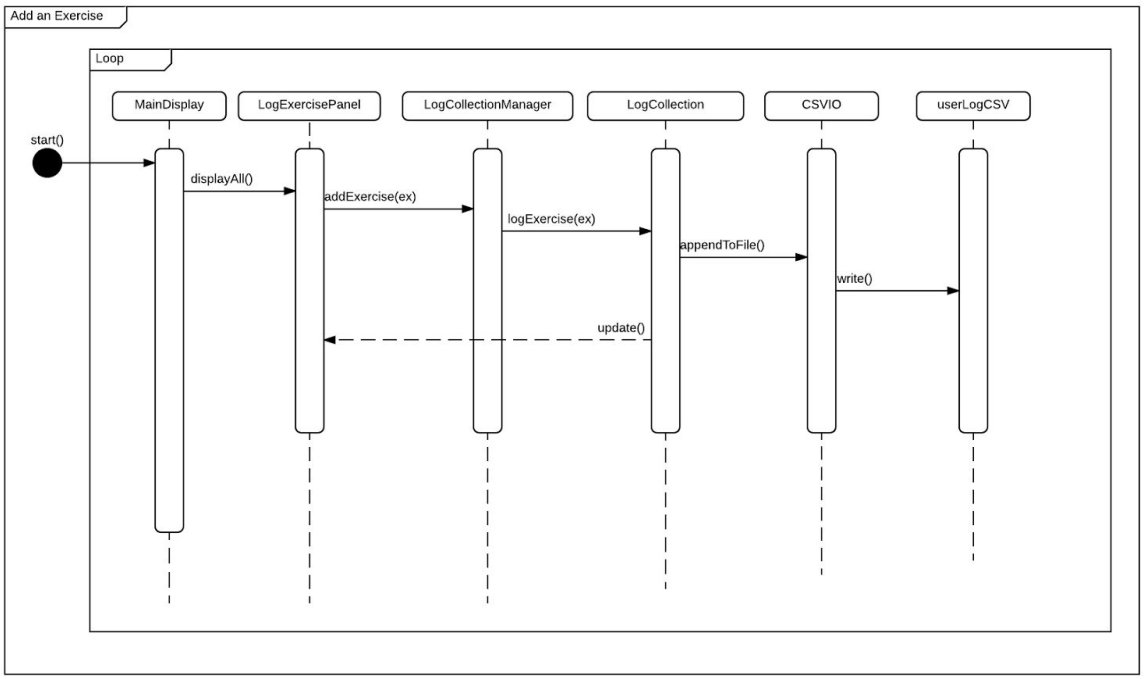
Deleting food entry from the LogCollection



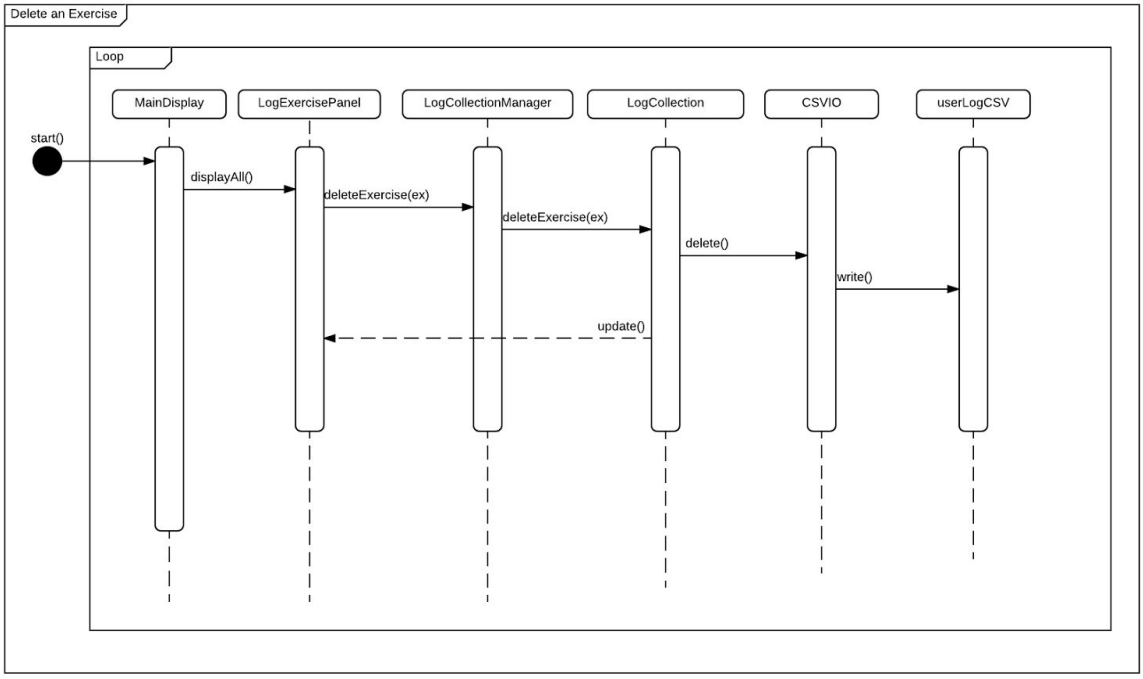
Create a new Exercise



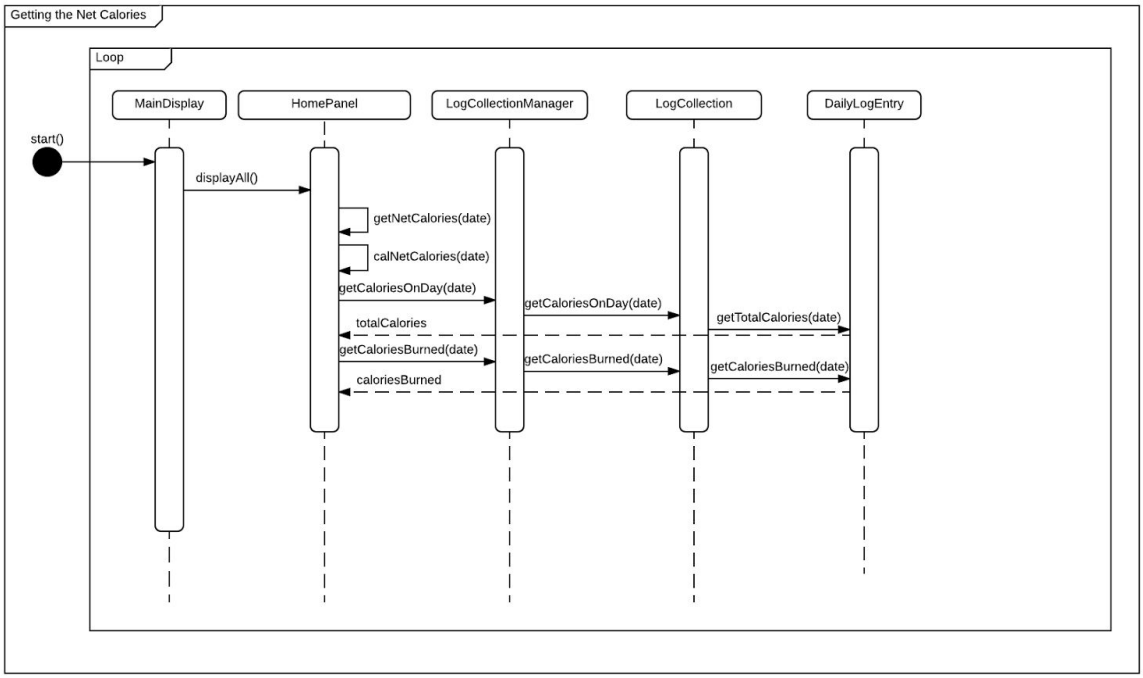
Add Exercise



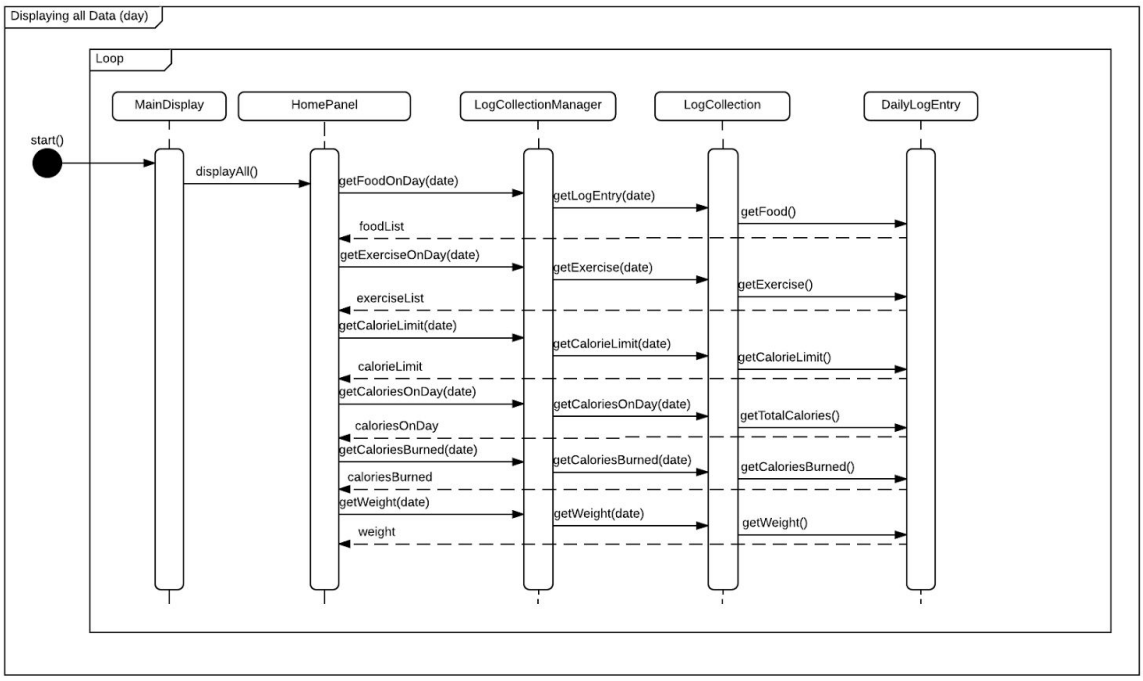
Delete Exercise



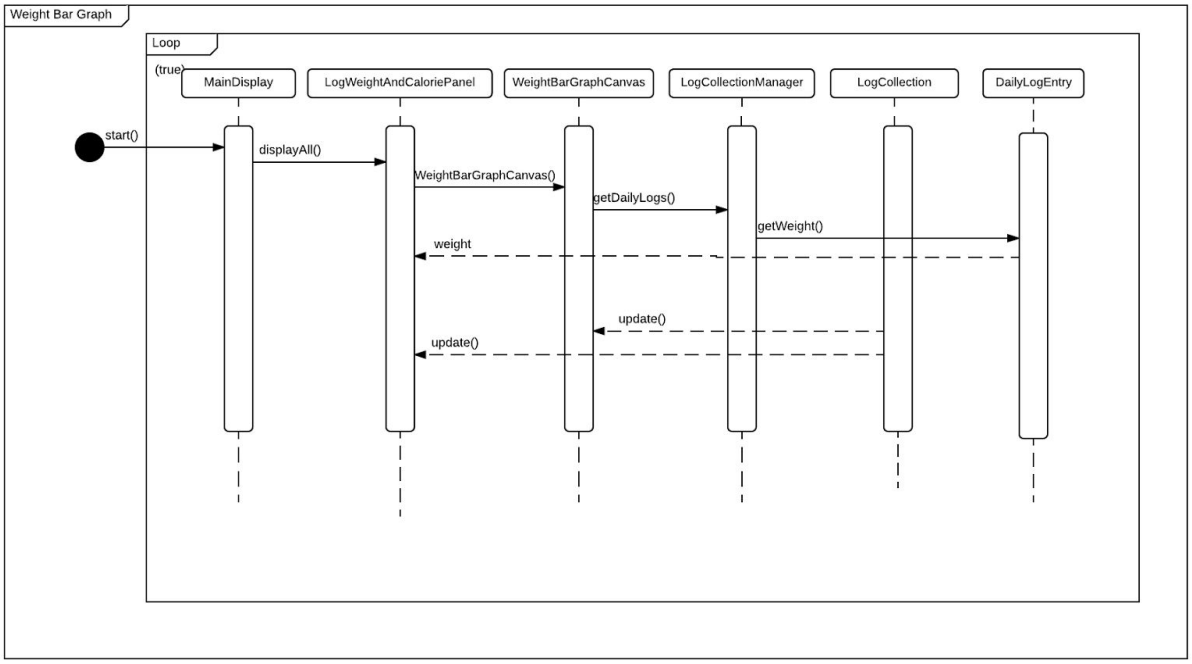
Getting the Total Net Calories



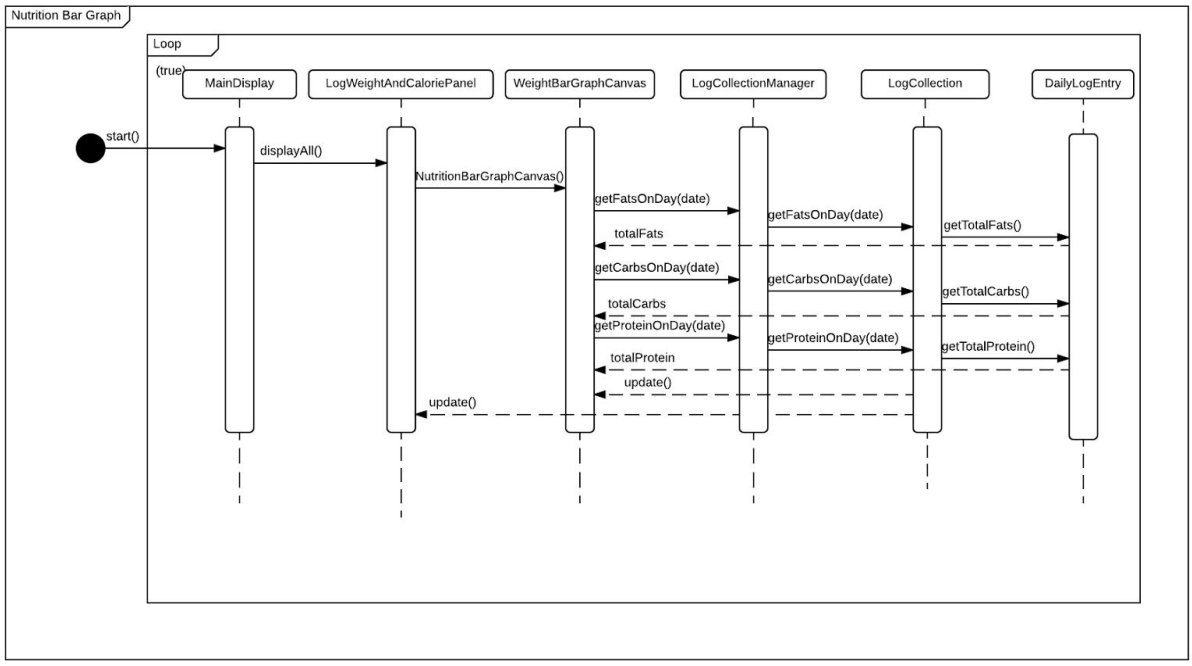
Display All Data Related for the Day



Weight Bar Graph



Nutrition Bar Graph



Pattern Usage

Observer Pattern

Observer Pattern	
Observer(s)	MainDisplay
Observable(s)	LogCollection, FoodCollection, ExerciseCollection

Composite Pattern

Composite Pattern	
Component	Food
Leaf	Basic Food
Composite	Recipe

MVC Pattern

MVC Pattern	
Model	LogCollection, FoodCollection, ExerciseCollection, Exercise, Food, Basic Food, Recipe, CSVIO, IOInterface, DailyLogEntry
View	MainDisplay, CreateExercisePanel, CreateFoodPanel, LogExercisePanel, LogWeightandCaloriePanel, HomePanel, LogFoodPanel, NutritionBarGraphCanvas, WeightBarGraphCanvas
Controller	LogCollection Manager, FoodCollection Manager,ExerciseCollection Manager

Program to Interface

Program to Interface	
Interface	Panel
Classes	HomePanel LogFood LogExercise LogWeightAndCalories Log

Composite Pattern

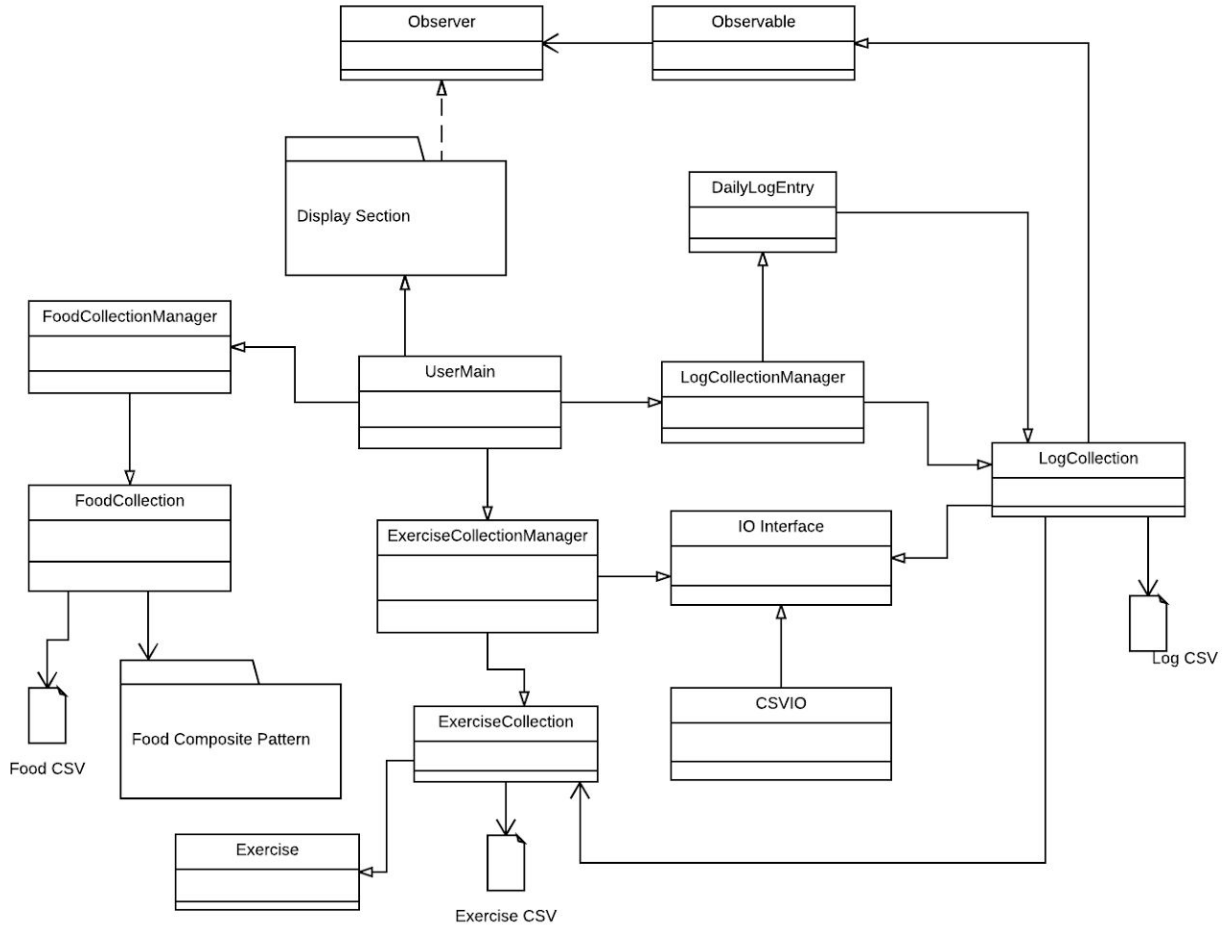
Program to Interface	
Interface	IOInterface
Classes	CSVIO

Dependency Injection

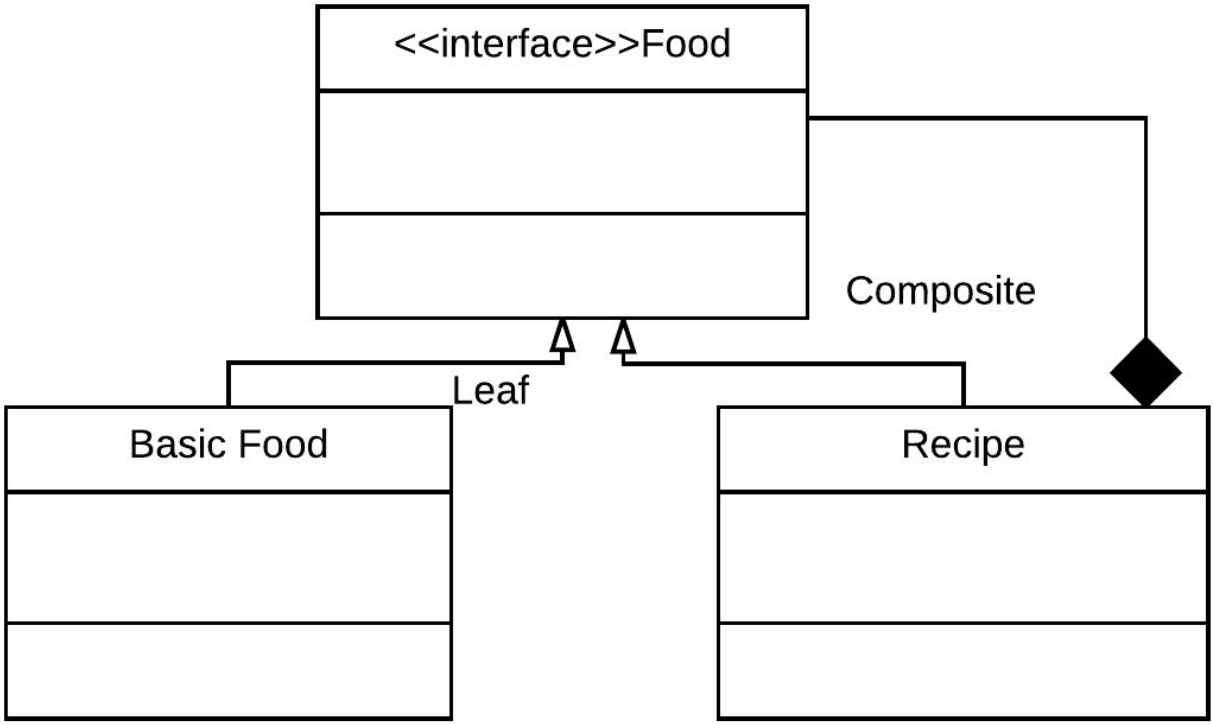
Dependency Injection	
Higher Level	UserMain
Lower Level	

Food Composite Pattern

See subsections below.



Food Composite Pattern



Display Section