Food Query and Meal Analysis

Author : Aditya Kumar Akash, Sapan Gupta
Status: In Review
#CS400 #2018Fall

Overview

This document outlines the design for the Food query and Meal analysis project. The detailed project description and requirements can be found here.

Detailed design

This section outlines the class structure of Food meal planner and analysis in detail.

Class structure

Following are interfaces created for this project :

Interface	**Description**
FoodDataADT	An interface to load and process food item data.
BPTreeADT	A generic B+ Tree interface.
NodeWrapperADT	An interface for classes which could act as wrapper over javafx Node class. The intent is to abstract out classes which can internally contain complex javafx entities, but could be represented externally by a single Node object. This would allow them to be contained inside another javafx containers without exposing any internal details.

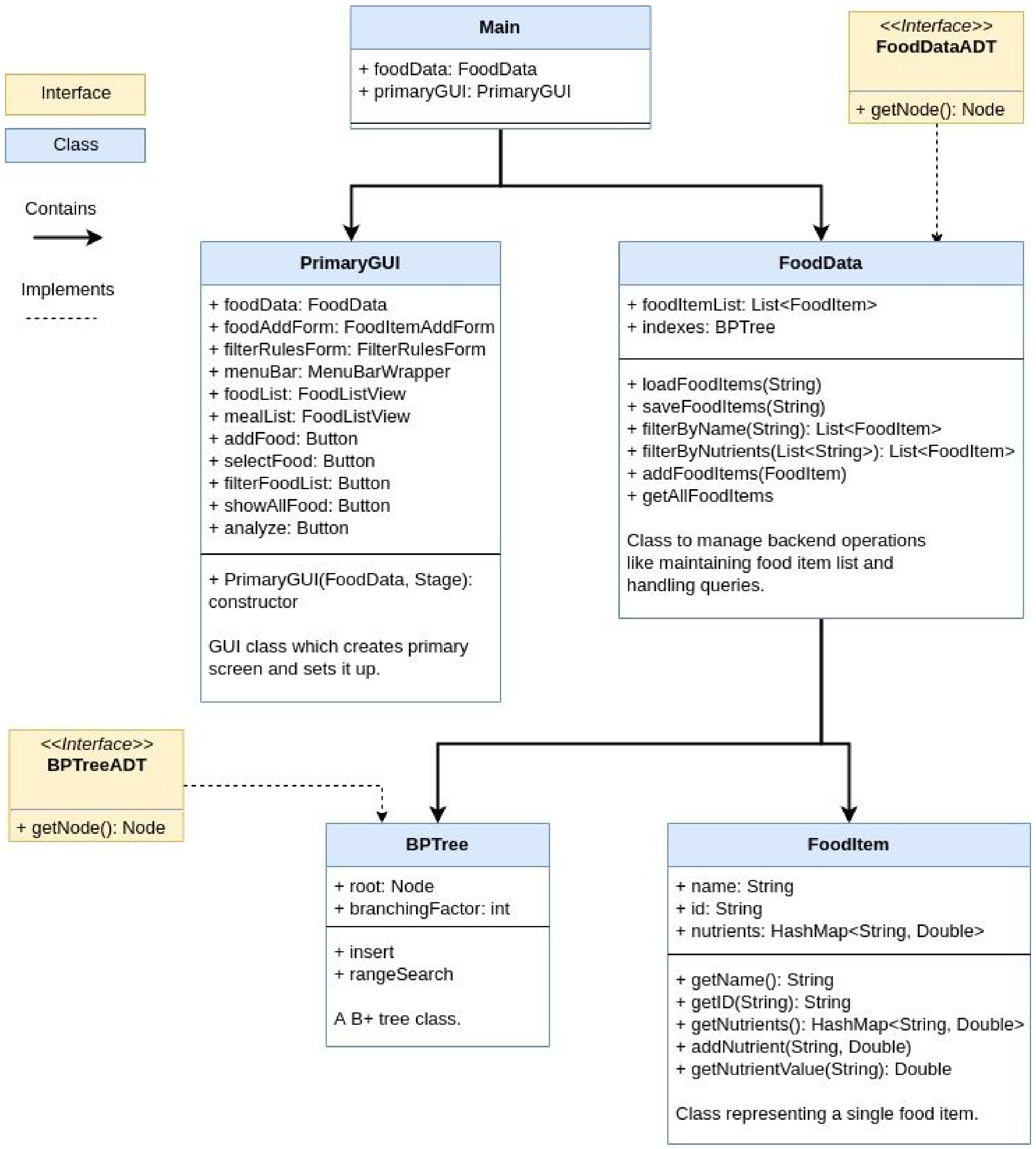
Following are the classes created for accomplishing the project tasks :

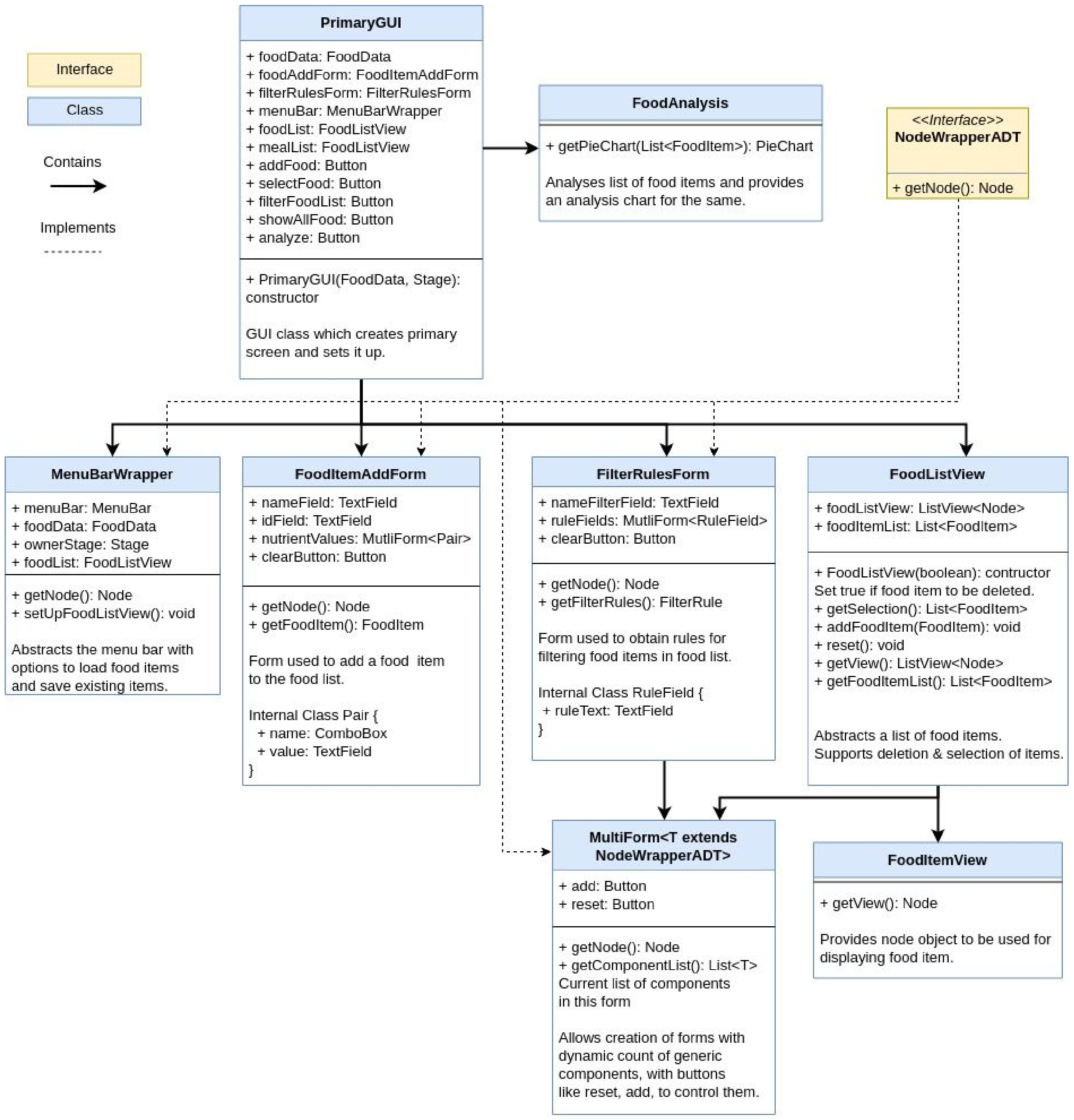
Class Name	**Description**
Main	This is the main application class and is responsible for creation of the backend instance (FoodData) and primary GUI.
FoodItem	This class represents a single food item. It has members for food name, id and the various nutrient values.
BPTree	This class represents a single instance of B+ tree. It provides functionality to add new comparable items and range search over the existing items
FoodData	This class is the main backend class which interacts with different components of UI classes. It abstracts the non UI interaction details, like reading from csv files, writing back food items and filtering of food list.
FoodItemView	This class abstracts javafx Node object which would be used for representing an instance of FoodItem.
FoodListView	This class abstracts a list view for collection of food items. It allows multiple items to be selected and provides API to extract selected items. It provides a scrollable view of the food items. It also provides a control to delete some of the food items, which can be enabled/disabled. Instance of this class is used for created food list and meal list components in the UI.

MultiForm	This class abstracts a UI form structure which has dynamic number of simple components (type T). The components can be dynamically added using an 'Add' button, and reset using 'Reset' button. This structure has uses in cases where multiple inputs needs to be taken from user, but the number of such inputs is not known beforehand. For this project, taking inputs like nutrients in a food and rules to filter food items are relevant cases.
FoodItemAddForm	This class abstracts the form that would be used for addition of new food items to the existing collection.
FilterRulesForm	This class abstracts the form that would be used to obtain rules for filtering the food items which are being currently displayed in food list. There are two kinds of filter 1. Filter based on substring present in the food item names. 2. Filter based on nutrient values present in the food items.
FoodAnalysis	This class returns a chart corresponding to the analysis of food items in the meal list.
MenuBarWrapper	A wrapper class which abstracts the menu bar with options to load food items from a csv file and save the existing food items to as csv file.
PrimaryGUI	This is the main GUI class which creates the primary screen and sets up the scene.

Class Diagrams

The diagrams in this subsection shows how different classes and interfaces are organized, and sheds light on the interaction of the objects.


Interface
Class
Contains
Implements
Main
+ foodData: FoodData
+ primaryGUI: PrimaryGUI
<<Interface>>
FoodDataADT
+ getNode(): Node
PrimaryGUI
+ foodData: FoodData
+ foodAddForm: FoodItemAddForm
+ filterRulesForm: FilterRulesForm
+ menuBar: MenuBarWrapper
+ foodList: FoodListView
+ mealList: FoodListView
+ addFood: Button
+ selectFood: Button
+ filterFoodList: Button
+ showAllFood: Button
+ analyze: Button
+ PrimaryGUI(FoodData, Stage): constructor
GUI class which creates primary screen and sets it up.
FoodData
+ foodItemList: List<FoodItem>
+ indexes: BPTree
+ loadFoodItems(String)
+ saveFoodItems(String)
+ filterByName(String): List<FoodItem>
+ filterByNutrients(List<String>): List<FoodItem>
+ addFoodItems(FoodItem)
+ getAllFoodItems
Class to manage backend operations like maintaining food item list and handling queries.
<<Interface>>
BPTreeADT
+ getNode(): Node
BPTree
+ root: Node
+ branchingFactor: int
+ insert
+ rangeSearch
A B+ tree class.
FoodItem
+ name: String
+ id: String
+ nutrients: HashMap<String, Double>
+ getName(): String
+ getID(String): String
+ getNutrients(): HashMap<String, Double>
+ addNutrient(String, Double)
+ getNutrientValue(String): Double
Class representing a single food item.


Interface
Class
Contains
Implements
PrimaryGUI
+ foodData: FoodData
+ foodAddForm: FoodItemAddForm
+ filterRulesForm: FilterRulesForm
+ menuBar: MenuBarWrapper
+ foodList: FoodListView
+ mealList: FoodListView
+ addFood: Button
+ selectFood: Button
+ filterFoodList: Button
+ showAllFood: Button
+ analyze: Button
+ PrimaryGUI(FoodData, Stage): constructor
GUI class which creates primary screen and sets it up.
FoodAnalysis
+ getPieChart(List<FoodItem>): PieChart
Analyses list of food items and provides an analysis chart for the same.
<<Interface>>
NodeWrapperADT
+ getNode(): Node
MenuBarWrapper
+ menuBar: MenuBar
+ foodData: FoodData
+ ownerStage: Stage
+ foodList: FoodListView
+ getNode(): Node
+ setUpFoodListView(): void
Abstracts the menu bar with options to load food items and save existing items.
FoodItemAddForm
+ nameField: TextField
+ idField: TextField
+ nutrientValues: MutliForm<Pair>
+ clearButton: Button
+ getNode(): Node
+ getFoodItem(): FoodItem
Form used to add a food item to the food list.
Internal Class Pair {
+ name: ComboBox
+ value: TextField
}
FilterRulesForm
+ nameFilterField: TextField
+ ruleFields: MutliForm<RuleField>
+ clearButton: Button
+ getNode(): Node
+ getFilterRules(): FilterRule
Form used to obtain rules for filtering food items in food list.
Internal Class RuleField {
+ ruleText: TextField
}
FoodListView
+ foodListView: ListView<Node>
+ foodItemList: List<FoodItem>
+ FoodListView(boolean): contructor
Set true if food item to be deleted.
+ getSelection(): List<FoodItem>
+ addFoodItem(FoodItem): void
+ reset(): void
+ getView(): ListView<Node>
+ getFoodItemList(): List<FoodItem>
Abstracts a list of food items.
Supports deletion & selection of items.
MultiForm<T extends NodeWrapperADT>
+ add: Button
+ reset: Button
+ getNode(): Node
+ getComponentList(): List<T>
Current list of components in this form
Allows creation of forms with dynamic count of generic components, with buttons like reset, add, to control them.
FoodItemView
+ getView(): Node
Provides node object to be used for displaying food item.

UI actions and class interactions

This section describes how different classes interact for various UI actions. Please refer to the diagrams in the previous sections for clarifications of nomenclature.

Add food

This button is contained in PrimaryGUI class. It adds new food items to the food list.
Interactions :

1. Obtains new FoodItem instance from FoodItemAddForm.
2. Adds the FoodItem instance to FoodData.
3. Adds the FoodItem to the foodList (FoodListView instance).

Select food

This button is also a part of the PrimaryGUI class. It selects food items from food list and adds them to the meal list.
Interactions :

1. Obtains the list of selected food item from foodList (FoodListView instance).
2. Adds the list to mealList (FoodListView instance).

Filter food List

This button is part of the PrimaryGUI class. It applies the filters to the food list and only displays the food items which satisfy the constraints/rule.
Interactions :

1. Obtains the filters (valid substring and nutrient based filter values) from the FilterRulesForm. The correctness of rule format could be done here or inside FilterRulesForm (preferred).
2. Query the FoodData instance with the rules and obtains the filtered list of food items.
3. Reset the foodList (FoodListView instance) and load it with the new list of food items.

Show all food items

This button is part of PrimaryGUI class. It loads all the food items from the FoodData into the food list.
Interactions :

1. Query the FoodData for all the food items and obtains a list.
2. Reset the foodList (FoodListView instance) and loads the new list into the foodList.

Analyze

This button is part of PrimaryGUI class. It displays the analysis of the food items present in the meal list using charts (javafx provides various chart).

Interactions:

1. Obtains list of food items present in the mealList (FoodListView instance).
2. Calls the functions in FoodAnalysis to obtain the chart corresponding to the analysis of the food items.
3. Displays the chart using a dialog box.

Deletion from meal list

Implementation using a single button (easy & preferred)

This button would be part of the PrimaryGUI class. It would delete the selected food items from the meal list.
Interactions :

1. Calls the delete function of mealList (FoodListView instance)
    a. The delete function would fetch the selected food items and clear them from the ListView.

Implementation using close button for each item (non-trivial)

This involves customizing ListCell to display a list item. The list item in this context consists of a node for food item and a button to control its deletion.

Dynamic UI components in MultiForm

As described in previous sections, a MultiForm provides a wrapper to create an abstract form structure which allows dynamic control over the number of simple components (eg. TextBox, or other complex objects). It provides Add and Reset buttons to control the number of components. The API also exposes a function which returns the current list of components present in the form. This allows to extract information from this abstract structure.

Add button

1. Creates a new instance of the simple component of class T
2. Appends it to the underlying list holding the number of components.
3. Adds a new Node to the underlying layout instance (eg. VBox).

Reset button

1. Clears all the existing components from the underlying list
2. Clears the layout instance.
3. Adds a new single component of type T.

MenuBar load file

This is contained in the MenuBarWrapper class. It loads the food items from the provided csv file into the food data.

Interactions :

1. Creates a FileChooser instance to allow the user to select the csv file.
2. Calls the load function from FoodData with the obtained filename.
3. Loads the foodList (FoodListView instance) with the new list of food items.

MenuBar save file

This is contained in the MenuBarWrapper class. It saves the food items from the food data class and saves it to a csv file.
Interactions :

4. Creates a FileChooser instance to allow the user to create a new csv file.
5. Calls save function from FoodData with the obtained filename.