Program 3 Design

Alexander DuPree

November 1, 2018

1 Task and Purpose

Program number three's primary objective is to utilize operator overloads in an object oriented application. The applications purpose is to generate an order of food items specific to dietary needs. The client will be able to look at a menu of food items and search for items based on specific criteria (gluten-free, vegan, low calorie, etc.) and add any desired item to their order. If no food items are found that match their criteria the client will be able to add specific food items to the menu. In the end, the client will have an order of food items that match their specific dietary needs and be able to receive a printout of pertinent nutritional info.

2 Design Considerations

Object oriented programming requires a modular design with an emphasis on encapsulation, inheritence, and polymorphism. Accomplishing this design will require multiple classes each with a single responsibility as their focus. This will help encapsulate and protect data related to a specific task while keeping the program organized and scalable. This assignment aim's to apply those design principles and as such, I will be keeping each class as small as possible, and create only as much production code required to accomplish a specific task while avoiding the excessive use of "getters/setters". This assignment also requires the full suite of operators to be overloaded. Only relevant operators will be overloaded in a way the make sense for the applied class.

3 Classes, Relationships, and Responsibilites

The following is a basic outline of each class to be created for this program and their responsibility and relationship to other classes.

3.1 Food Item

Responsibilities: The primary responsibility of the Food Item abstract base class is to manage the common food data relevant to all items. Each food item, be it an entree, dessert, or beverage share common information like calories, gluten-free, vegan etc. All these common attributes are pushed up into the Food Item class for the base class to handle.

Relationships: The Food Item acts as the abstract base class for all objects in the food hierarchy. Furthermore, the Binary Search Tree used in this assignment will use Food Item pointers and dynamic binding to manage a collection of food items.

Functions: The interface for the Food Item base class will declare virtual methods and operator overloads. The « operator will be overloaded to display data. The » operator overload will read data in from the console. And the equality operators will use calorie counts to determine precedence in sorting.

3.2 Entree

Relationships: The Entree class is derived from the Food Item base class and expands upon the base class methods and attributes.

Responsibilities: The Entree class maintains the same responsibility outlined in the base class. However, the Entree expands upon the methods and attributes of the base class by introducing further nutritional information such as carbohydrate and protein counters.

Functions: The Entree class contains all the same methods as its parent. However, the methods to display/read data are expanded to showcase the attributes specific to the Entree class.

3.3 Dessert

Relationships: The Dessert class is derived from the Food Item base class and expands the base class methods and attributes in much the same manner as the Entree class.

Responsibilities: The Dessert class acts in the same manner as the Food Item base class but expands the responsibilities to include a sugar counter for the item.

Functions: The Dessert class maintains the same methods outlined in the parent class but overrides the display/read methods and operators to include the attributes specific to the Dessert class.

3.4 Beverage

Relationships: The Beverage class is derived from the Dessert class. This is because a beverage contains many of the same information relevant to a dessert and a food item. However a Beverage can also be alcoholic and this is reflected in the Beverages attributes.

Responsibilities: The Beverage maintains the same responsibilities of its parents, but is expanded to manage a flag the determines if the beverage if alcoholic.

Functions: Beverage class expands its parents methods to include the alcoholic flag in the display/read methods.

3.5 Food Menu

Relationships: The Food Menu class is a binary search tree that utilizes dynamic binding to manage a collection of food item objects.

Responsibilities: The primary responsibility of the Food Menu is to manage the collection of Food Items delegated towards it. The Food Menu can perform search, insertion, and removal operations to instantiate Food Item objects. The Food Menu also handles the dynamic allocation and deallocation of food item objects in the system.

Functions: The Food Menu provides methods to insert, remove, and search for food items. The [] operator will be overloaded to allow the user to search for an item. The += operator will allow the client to add food items to the menu, and the -= operator will remove items matching the target name.

3.6 Client Order

Relationships: The Client Order is derived from the Food Menu class and is also a binary search tree of food item pointers.

Responsibilities: The purpose of the Client Order class is to manage the collection of food items the clients has chosen from the food menu. Throughout the application the client will be able to select food items from the menu that satisfy their dietary needs. These food items are added to the client order's tree. At the end of the application the client can receive a printout of the nutritional data that reflects their order.

Functions: The Client Order contains the insertions, removal, and search operations defined

in the parent class. However, the Client Order class also contains a special display method that will print out a detailed description of their order.