CS202 Program #2

Written Design and UML Diagram

1/29/2016

Hyunchan Kim

For this second programming assignment, the main focus is on hierarchy (inheritance) that we did for the first program and dynamic binding. Dynamic binding is basically more flexible since it enables developers to extend the behavior of a system transparently. Therefore, I will be using dynamic binding for the derived classes that are able to provide a different implementation (more functional, more efficient) that should be selected at run time and to build dynamic type hierarchies and to form abstract data types. In other words, dynamic binding is achieved using virtual functions. Base class pointer points to derived class object. And a function is declared virtual in base class, then the matching function is identified at run-time using virtual table entry. In this program, we are going to create a restaurant management program that controls menu items, ingredient information, items in stock, and so on. There will be many functions we need for restaurant managements. If we go more deep in to the design hierarchy, I decided to have 5 main classes, item, menu, entry, appetizer, and drink class. The item class will be on the top base class and the menu class is the next derived class. And then, entry, appetize, and drink classes are the next derived classes from the menu class. For more details, first, the item class will get to have all the data information such as menu, ingredients, quantity, and expiration dates from two external files (menu.txt and stock.txt).

And secondly, the menu class will organize all the menu and ingredient information from the item class. The menu class will be a tree structure that each node will be a linear linked list. The tree will basically be organized by ingredient. In order to design this class with the data structures, there will be two nodes for the tree class and linear linked list class. For each tree node, there should be a linear linked list. They will all be dynamically allocated.

For the next step, there will be three derived classes; entry, appetizer, and drink from the menu class. In terms of the kind of menu, they will be classified into three sections.

In order to avoid to use getter and to make efficient as much as possible, the dynamic binding feature will be helpful that virtual function can minimize work something similar or the same functions from different classes. Especially, in the menu class, there will be common functions such as order, purchase, or remove that might be using in the derived classes (entry, appetize, and drink). There also can be virtual functions from the parent class, item. For the essential functions, I will be using virtual pure functions that you have to implement in the derived classes but for common functions, not mandatory, I will be using virtual functions.

About the functions, I have not specified for the all classes. However, some are the essential functions implemented. In the item class, there should be load function that reads in external files. In the menu class, there will be order, remove, display, and search functions that may be virtual for the derived classes. And since it is composed of data structures, it will require insert, remove, retrieve, display, and remove-all functions. As a requirement, the tree should always be balanced so that it will be a consideration to keep it balanced.

In the entry, appetizer, and drink classes, there will be some common functions from the base class implemented but the actual performance will be different in terms of menu items. As soon as you purchase items, the ingredient items corresponding in the stock will be updated.