CMIS 315 Final Project Grocery Store

Development Approach Plan

Version 1.0

3/7/2015

Contents

[1. Introduction 1](#_Toc413508313)

[2. Overview 1](#_Toc413508314)

[3. Assumptions/Constraints/Risks 1](#_Toc413508315)

[3.1 Assumptions 1](#_Toc413508316)

[3.2 Constraints 1](#_Toc413508317)

[3.3 Risks 1](#_Toc413508318)

[4. Development Approach 2](#_Toc413508319)

[4.1 Development Methodology 2](#_Toc413508320)

[4.2 Lifecycle Management & Transition Approach 2](#_Toc413508321)

[4.3 Methods & Tools 2](#_Toc413508322)

[4.4 Difficulties Encountered 3](#_Toc413508323)

[4.5 Testing 3](#_Toc413508324)

[5. Weaknesses and Improvements 7](#_Toc413508325)

[5.1 Weaknesses 7](#_Toc413508326)

[5.2 Improvements 7](#_Toc413508327)

List of Tables

Table 1: Development Approach Processes ……………………………………………………2

Table 2: Test Data Test Run 1 ………………………………………………………………….3

Table 3: Test Data Test Run 2 ………………………………………………………………….4

List of Figures

[Figure 1: Screen Shot Test Run 1 4](#_Toc413508520)

[Figure 2: Screen Shot Test Run 2 6](#_Toc413508521)

# Introduction

This Development Approach Plan describes the approach for development of the Grocery Store program. This document will outline the development approach, solution design, difficulties encountered, testing strategy, system weaknesses, and possible improvements. This document is written for Professor Omar Zevallos, CMIS 315 instructor.

# Overview

The purpose of this project was to create a program that uses input from the user to create food objects and display data for each item as well as a total price for all items created. The program should allow the user to enter a price for each item, as well as select whether the food item is organic, and has an attribute specific to the particular category of food (ie fruit = seedless, dairy = pasteurized, meat = lean). The program should prompt the user for the total number of food items they wish to add to their cart and then allow the user to select the items from a numeric menu. The use of case statements will prompt the user for object specific information and enter each food object into the vector. A counter will be used to terminate the loop for assigning food objects to the vector. Once all food objects have been entered into the vector the program will display all data for each item as well as the total price for each item in the cart.

# Assumptions/Constraints/Risks

## Assumptions

The Food.h and Food.cpp files are complete and do not need to be modified. The Main.cpp file will need modification for the program to function.

## Constraints

The program must override the virtual print method of the Food class, and must inherit all methods and attributes from this class.

## Risks

None

# Development Approach

## Development Methodology

I used the waterfall approach for software development for this project. I concluded this approach at the coding and testing phase, as the software will not be implemented, and operational support will not be needed. I used this methodology because:

1. The project requirements where clearly defined and unambiguous;
2. The requirements for this project are stable and not subject to change;
3. The project had clear objectives and solutions.

I divided the project into 7 parts:

1. Food.h class header
2. Fruit.h class header
3. Dairy.h class header
4. Meat.h class header
5. Food.cpp class – defines all methods necessary to create abstract food class
6. Fruit.cpp class – defines all methods necessary to create abstract fruit class and create and display concrete classes Apple, Banana, and Grape.
7. Dairy.cpp class – defines all methods necessary to create abstract dairy class and create and display concrete classes Milk, Yogurt, and Cheese.
8. Meat.cpp class – defines all methods necessary to create abstract meat class and create and display concrete classes Beef, Pork, and Chick.
9. Main.cpp – test driver program containing all user prompts and methods to instantiate concrete classes adding them to vector and printing items that user selects.

## Lifecycle Management & Transition Approach

I started with the requirements definition phase, as there was no need for an initial investigation. Some of the requirements were defined in the project description; the rest became apparent during the writing of the initial algorithm. After defining all of the requirements I started with the basic system design by hand writing class diagrams and the basic algorithm. Once I had a clear picture of the system, I began to code the classes. Once each class was defined, I coded the main method. Once I had completed all coding I compiled and ran my program in the debugging mode to begin testing.

## Methods & Tools

Table 1: Development Approach Processes

| Process | Tools & Techniques |
| --- | --- |
| Requirements definition | Paper and Pencil |
| Class diagrams | Paper and Pencil |
| Coding | Microsoft Visual Studio 2013 |
| Testing | Microsoft Visual Studio 2013 |
| Approach Document Creation | Microsoft Word 2010, Microsoft Paint |
| Assignment Submission | Microsoft Word 2010, Notepad |

## Difficulties Encountered

I had a difficult time with the syntax of overloading the base class virtual method. I had to spend some time doing outside research and re-watching the videos before I finally figured out what I was doing wrong. The other difficulty I had was trying to include my own exception handling. I just don’t have a good enough grasp on that yet and had to give up. I deleted out those line of code so that my program works as it should, as long as the user behaves and enters the expected values. The program will exit if the user enters anything unexpected because the exceptions are not handled, except in the case statement.

## Testing

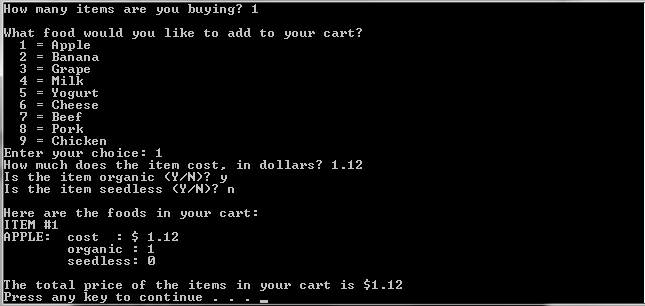
I began testing the program by first compiling the program in Microsoft Visual Studio 2013 and then running in the debugging mode. I chose 1 to enter 1 food object into my cart to check program output.

Table 2: Test Data Test Run 1

| Input | Output |
| --- | --- |
| Start Debugging | How many items are you buying? |
| 1 | What food items would you like to add to your cart?  1 = Apple  2 = Banana  3 = Grape  4 = Milk  5 = Yogurt  6 = Cheese  7 = Beef  8 = Pork  9 = Chicken  Enter your choice: |
| 1 | How much does the item cost, in dollars? |
| 1.12 | Is the item organic (Y/N)? |
| y | Is the item seedless (Y/N)? |
| n | Here are the food items in your cart:  ITEM # 1  APPLE: cost : $1.12  organic : 1  seedless : 0  The total price of the items in your cart is $1.12  press any key to continue… |

The below screen shot shows the output of this run.

Figure 1: Screen Shot Test Run 1

**

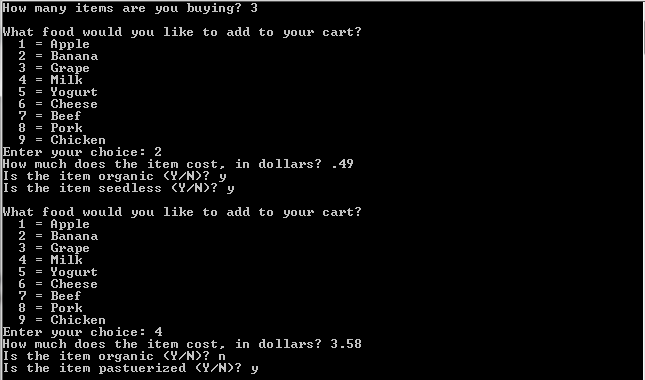
For the second test run I choose one of each members of the Food subclasses. The below table shows the data entered during the second test run with the outputs.

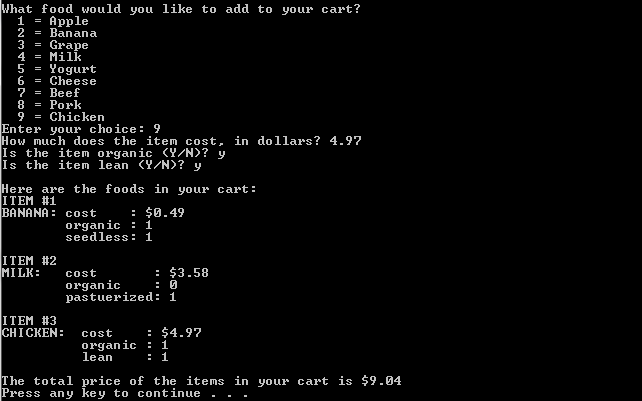
Table 3: Test Data Test Run 2

| Input | Output |
| --- | --- |
| Start Debugging | How many items are you buying? |
| 3 | What food items would you like to add to your cart?  1 = Apple  2 = Banana  3 = Grape  4 = Milk  5 = Yogurt  6 = Cheese  7 = Beef  8 = Pork  9 = Chicken  Enter your choice: |
| 2 | How much does the item cost, in dollars? |
| .49 | Is the item organic (Y/N)? |
| y | Is the item seedless (Y/N)? |
|  | What food items would you like to add to your cart?  1 = Apple  2 = Banana  3 = Grape  4 = Milk  5 = Yogurt  6 = Cheese  7 = Beef  8 = Pork  9 = Chicken  Enter your choice: |
| 4 | How much does the item cost, in dollars? |
| 3.58 | Is the item organic (Y/N)? |
| n | Is the item pasteurized (Y/N)? |
| y | What food items would you like to add to your cart?  1 = Apple  2 = Banana  3 = Grape  4 = Milk  5 = Yogurt  6 = Cheese  7 = Beef  8 = Pork  9 = Chicken  Enter your choice: |
| 9 | How much does the item cost, in dollars? |
| 4.97 | Is the item organic (Y/N)? |
| y | Is the item lean (Y/N)? |
| y | Here are the food items in your cart:  ITEM # 1  BANANA: cost : $0.49  organic : 1  seedless: 1  ITEM # 2  MILK: cost : $3.58  organic : 0  pasteurized: 1  ITEM # 3  CHICKEN: cost : $4.97  organic: 1  lean : 1  The total price of the items in your cart is $0.49  press any key to continue… |

The below screen shot shows the output of this run.

Figure 2: Screen Shot Test Run 2

**

**

# Weaknesses and Improvements

## Weaknesses

The weaknesses that I found in the system are:

1. No welcome message explaining the purpose of program. The user may feel lost simply being asked how many items they would like to add to their cart. Items of what?
2. No exception handling if the user enters a non-integer value at the start of the program. Program execution stops if non-integer value entered.
3. No exception handling if the user enters a non-integer value in the case statement. Program goes into endless loop.
4. No exception handling if the user enters a non-integer value for price. The rest of the program skips and erroneous value is output.
5. No exception handling for values other than Y/N for Boolean values. Anything other than these values entered evaluates to false.

## Improvements

To improve the program I would do the following:

1. Add welcome message explaining the purpose of the program.
2. Add exception handling for non-integer values at the start of the program to display error message prompting user to input integer value.
3. Add exception handling for non-integer values in the case statement program to display error message prompting user to input integer value.
4. Add exception handling for non-integer values for the price for the program to display error message prompting user to input integer value.
5. Add exception handling for values other than Y/N entered for the Boolean values to prompt the user to that the must enter either Y or N.