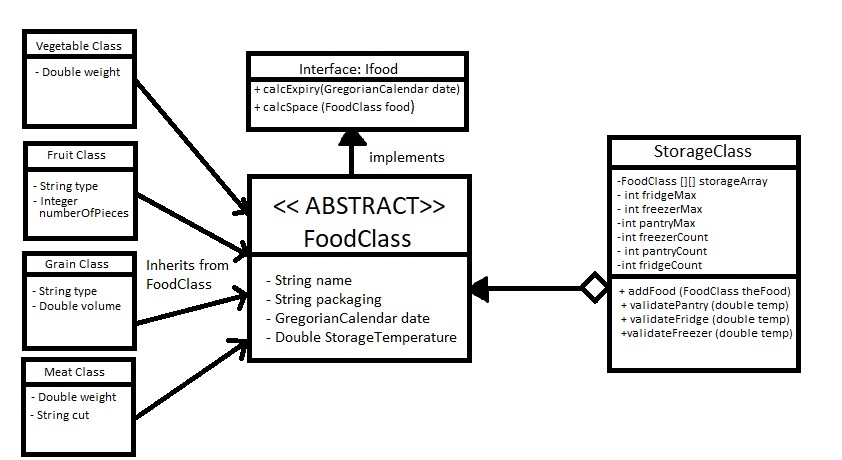
OOPD ASSIGNMENT: PHILOSOPHY OF DESIGN   
BY LARA AHMAD SALEM



For the OOPD Assignment we were assigned to create a whole storage location and food class that would handle storing food objects. I had decided to implement my Food classes using inheritance since there were a lot of similar elements that all the foods shared. For example, I combined the name, packaging, storage Temperature and the Date into an abstract class that would pass on these elements to the rest of the classes that extended from it such as vegetables, fruit, meat and grain. In my design I had to implement the IFood interface so that all my classes could access the calcExpiry method to find out if any of the food objects were past their due date. I have also chosen to implement the Gregorian Calendar class and combined both the use By and Best before date as one class field to make it easier to pass by these objects and store them. My assignment consisted of approximately 11 classes that all worked together invoking methods and passing on different data types to implement a fully structural and working program. I have also incorporated a menu class for the sole purpose of a functional and readable graphical user interface that will allow my users to navigate their way across my program with ease. I had developed abstract methods in my storage class that would enable my users or anyone in this case using the program to add the food objects into the two-dimensional array. Once it has entered the storage class my objects were organized in two dimensional arrays according to their storage temperature. If they did not satisfy any of the temperature ranges they would be disregarded and not populated into the food array. I have also included two types of validation depending on whether the file was read in or added manually. If the food objects are being created while the user is inputting the data separately a submodule would have been called to validate each input and if the input is invalid it would prompt the user to enter it again. This has been successfully achieved by the efficient use of try catches. By using try catches I have successfully managed to prevent users from hopefully crashing the program with invalid inputs. Furthermore, in my program I have established a separate class that does all the validation required depending on the data type. A separate class has been created for File IO so that all file reading and writing will be handled elegantly and efficiently. Additionally, I have implemented two types of toString methods within my program. One which outputs the food object in a sentence that is easy to read and is mainly used in the main menu to display food contents and to remove them. The other type of toString method I used was a special type of format that has comma separated values so that when my file writes out the objects it does so in the same manner as the input file. All together these classes work to maintain and efficiently run my program. This has been achieved with the help of the Oracle api documents and lectures slides that have guided me with the knowledge I needed in order to create this program.