Curtin University
School of Electrical Engineering, Computing and Mathematical Sciences
Object Oriented Program Design (COMP1001)
Semester 1, 2018

Assignment Addendum

This document contains some additional requirements, and some hints/suggestions for the assignment.

Additional Requirements:

The classes for Meat, Grain, Fruit and Vegetables MUST inherit from a class called Food.
The sub classes of Food MUST implement the interface IFood (code given as appendix 1)
The class containing the 2-dimensional array for the Pantry, Fridge and Freezer MAY become a super class, with sub-classes of Pantry, Fridge and Freezer, each containing a (possibly inherited) 1-d array.
You MUST provide a philosophy of design document (more than 1 page, less than two pages), which MUST include a UML diagram.

Hints:

The class containing the 2-dimensional array for the Pantry, Fridge and Freezer (perhaps called Storage) will have, at a minimum, a classfield for the 2-d array, and the number of Food items currently stored in each row. Each row will represent either the Pantry, Fridge or Freezer. This class will also have, at the very minimum, methods to:

- Add food – there may be more than one (method overloading)
- Remove food
- Display contents – will call **`toString`** in Food
- Find expired – will call the **`calcExpired`** in Food

The functionality of these methods is described in the assignment specification.

When an item of Food is removed, the array should be shuffled to fill in the hole.

Your program should also have, at a minimum, a class for the menu, and a class for Input/Output. These classes should not have classfields.

It is OK (and recommended) for main to contain a single call to the appropriate method in the menu class (e.g. `Menu.run()`)

The method calcSpace from IFood will not be <u>used</u> in this assignment – it is provided for possible future development.

Interface code and information follow.

```
public interface IFood
{
   public boolean calcExpiry( Calendar today );
   // imports todays date and exports true if this food item
   // has reached its expiry date, false otherwise.

   public int calcSpace( Food food );
   // imports a Food class object, and exports an integer
   // specifying the volume in litres of storage required.
}

/* Additional information:
   Each subclass of Food has a different calculation for space:
   Meat – weight * 0.86 rounded up
   Grain – volume * 1.0 rounded up
   Fruit – number of pieces * 1.95 rounded up
   Vegetables – weight * 1.025 rounded up
*/
```