IICS

FACULTY OF INFORMATICS

University of Wollongong

CSCI204

July Session 2014

Assignment 1

*(Individual Work – 10% of subject marks)*

This assignment covers the following topics:

* Classes and Objects
* UML Diagrams: Class and Object Diagrams
* Constructors
* Composition and Aggregation

**Remember that:**

**1. All programs should be able to run on the lab’s computers.**

**2. You must put the following information on the header of each text and source file you will be submitting in this assignment:**

**Student’s full name:**

**Student’s ID:**

**Modification Date:**

**Purpose of this file (or program):**

**3. Assignments that are not able to be compiled will result in zero mark given to the assignment.**

**4. You must only use the C++ features that have already been covered in the lectures**

Problem Specification:

In this assignment, you are required to produce a piece of code to simulate the process involved in a Supermarket transaction. Your code should involve three files, Supermarket.cpp, Supermarket.h, and Main.cpp. The code should contain appropriately placed declarations and implementations associated with three classes and a main() function. These classes and functions should be developed according to the specification given below.

You are to write a Supermarket class that, during initialization, reads data from a file, Stock.txt and populate its data. That file has the format shown below:

SJ Supermarket:100:1234.56:

Licorice:500:0.50:50:metre:

Bread buns:1000:2.00:500:6-pack:

Cabbage:500:1.90:300:each:

...

The first line contains the name of the supermarket, the number of different items stocked at that supermarket, and the money the supermarket has on hand initially.

The general format for the other lines is:

Item name:number:unit price:unit weight:unit description:

where number indicates the number of that item, unit price indicates how many dollars a single unit costs, unit weight indicates how much a single unit weighs (in grams), while unit description indicates what consitutes a unit.

Thus the Bread buns line implies that SJ Supermarket has 1000 units of 6-pack buns, each weighting 500

grams, and that the price for each RM2.00.

Note that you will need to read the number of items from this file. It may be different for different runs of the program.

The Supermarket class should have:

1. A name field.
2. A cash field.
3. An array of Item objects in stock. This should be populated during initialisation by the constructor. You may assume that the number of items will not be more than 100.
4. A constructor which reads from the file and populates the data.
5. A shop function which takes a Customer object and interacts in the way later described.
6. A stock report function which displays a complete list of current stock.
7. A financial report function which displays how much cash the supermarket has and how much the current stock is worth (based purely on the pricing and number of each item).
8. An appropriate destructor.

The Item class should have:

1. A name field.
2. A counting field to indicate how many of the items there are.
3. A unit price field.
4. A unit weight field.
5. A unit description field.
6. An appropriate constructor.
7. A function to determine the total value held in that item type.
8. A function to display the data.

The Customer class should have:

1. A name for the customer.
2. A cash field.
3. An array of purchased Item objects (see later) which will be populated during shopping.
4. A constructor to set the data fields.
5. A display function to display the information about the customer and the complete list of purchased Item’s.
6. A display function to indicate the total mass and cost of the purchased items.
7. An appropriate destructor.

Write a driver program that will create a Supermarket object and an array of Customer objects (may be initialized using a separate Customer file that includes several Customer information). The driver program may provide a suitable menu to allow for several options to be performed such as shop, display financial report, display stock report, and Quit. The shop option should include shop that takes a customer’s information, create a customer object, and load it to the Customer array. Every time a purchase is to be made, state the customer’s name and cash available. The program should not allow the customer to buy if the available cash is not enough. The Customer should be allowed to buy as many as they can. The stocked Item in the Supermarket should be decreased as items are purchased and amendments should be made to the cash of the Supermarket and Customer.

Before the program ends, the Supermarket information and the Customer information should be saved to their respective files.

In addition to the code you need to provide a complete class diagram with the relationships between classes shown clearly.

Submission:

You are to submit:

1. The softcopy of the project containing all files (.cpp and .h).
2. A word document containing the UML class diagram with explanation for each class and relationship.

Both softcopy and UML class diagram should be in a zip folder and submitted to Moodle.

The completed Assignment must be submitted latest by **Friday, 19th September 2014, 5 pm**.

Late submissions will be marked with a 25% deduction for each day.

Submissions more than three days late will not be marked, unless an extension has been granted.