**Cheapo Software Solutions (CSS.inc).**

**Vending Machine Opportunity (CS6402: 30%)**

Cheapo Software Solutions is a company that specializes in market ‘opportunities’ in the software sector. They have just spotted one such opportunity. A new vending machine company is looking for software to run their vending machines. They want a system that has all the usual Vending machine functionalities (show all products available, buy products, quit the system), and has a nice GUI feel to it (Java FX).

The vending company works on money-less operation where clients buy credit using an App and can then use that credit to buy bars in the vending machines, make photocopies in the library and buy meals in the café. This app is not part of the specification for this project. Instead the vending machine software just has to load up the ‘client’ file that is updated by that app before the vending machine is started up. The file contains the user’s username, current credit and password (a 6 digit number). The file is a csv file called clients.dat.

The products have a description, price and location. To buy a product the user has to identify themselves (username and password), have enough credit and specify the product location in the vending machine. Remember that when a product is bought, there is one less remaining in the vending machine, and this should be recorded.

There is a separate user of the system – an admin person who also has a user name and password stored persistently in a csv file called admin.dat. They are allowed to add/reload products back into the vending machine and “shut the system down” for the night. This shutdown process stores the current product (products, location, price and amount) information to another csv file called product.dat that is reloaded on product start-up.

The ‘opportunity’ that Cheapo Solutions perceives is that it has found some vending software in a Java textbook. It already has much of the functionality that the vending machine company want. Cheapo Solutions thinks that they will be able to adapt this existing system to their needs and reap the huge reward (€80,000) that the Vending company is willing to offer.

You have just been taken on by Cheapo Solutions (being paid a miserable 30% of your CS6402 module /annum) and they have tasked you with adapting the textbook system to the needs of the vending company. You have 2 weeks to achieve this. Specifically your job is to

1. Incorporate the persistent files client.dat, admin.dat and product.dat into the system, as specified above;
2. Add a graphical user interface (GUI) to the system using JavaFX;
3. Add operator authentication services. This requires changes to the user interface to distinguish between an admin operator (who can restock the machine, do the end-of-day procedure and logout) and a ‘normal’ client who can interact with the machine to view all products, buy a product and logout. Such an authentication service would allow a user to provide their username and their authentication code/password. This would be verified against the code stored in the client.dat/admin.dat text file;
4. The vending machine in the textbook uses American money. This should use Euros for our vending machine company.

A senior Cheapo Technician (Abdul) will be available to help you for 2 hours every week, but you will probably need to work after hours to achieve this target (Hey – it is “Cheapo Software Solutions”!).

**File Formats:**

**Admin.dat** (Example only)

JohnT,125648

**Product.dat** (Example only)

Mars, A1, 1.30, 7

Wispa, A2, 1.30, 10

Twirl, A3, 1.40, 2

Tayto, B1, 1.10, 5

…

Yorkie, C3, 1.40

**Clients.dat** (Example only)

Rebecca, 5.50, 685439,

Mary, 8.90, 777654,

Joe, 0.00, 987456,

…

Serge, 1.40, 975319

**Product Delivery**

You may do this in teams of 2 (or 1 if you so wish, but there will be no decrease in the deliverables specified). If you work in a team of 2, I will expect both team members to be able to explain all the code-base to me and reserve the right to hold interviews in the week following to make sure that each student can explain their/their team’s submitted code.

***Phase 1:***

You will submit a word/pdf document at the end of week 1 outlining your proposed changes to the existing system at a class level: new class responsibilities, removed class responsibilities, new classes. This will be lightly assessed, just to ensure that you are on a more-or-less sensible path and should not be more than a 2-page document – font 12, within the default margins. You should submit this document to SULIS by Friday the 24th of March, 17:00 and this is worth 5% of the 30%.

***Phase 2:***

You should submit your source code, ready to compile/execute, to SULIS by Wednesday the 26th of April, at 17:00. This should include the instructions to build the executable and run it in a readme file.

You should also update your 1-2 pager discussing what you actually did (as opposed to what you originally thought you might do!). I envisage this as a commented document where your original document persists and you use the MS Word/PDF-writer comment functionality to identify and reflect on the things that changed (5% of the 30%).

The remaining 20% will be given for the correct functioning of the system. In terms of this section and the four ticket items listed on page 1, the percentages available are:

* Us being able to run the system 4%
* Persistent data files: 6%
* GUI: 6%
* Different users: 8%
* European Currency: 4%

**Some Design Tips:**

1. Possibly think about the system having layers:
   1. User Interface,
   2. The System (Vending Machine) and associated classes, Product, Item, etc…
   3. Storage (Files).

The connections between these layers could be handled by other classes. For example the communication between the System and Storage could be handled by FileInputService and FileOutputService classes. The FileInputService class could read from files, parse input strings, and create item or coin objects and pass these objects to the system to be stored while the system is running. Likewise for the FileOuputService

1. The system should handle authentication of an operator.
2. You may have an abstract class for a menu and two concrete subclasses for StandardMenu and OperatorMenu or the concrete classes could implement an interface?