**Anil Timilsina**

Group: Anil Timilsina, Brandon Theisen, Cha Yang, Eric Dorphy

**CSCI 430**

**April 3, 14**

**Documentation: Warehouse GUI Final Stage**

Much of the core functionality of this project had already been implemented in the project 1. The transition to the GUI was done in two stages, where in the first stage Finite State Machine (FSM) design was implemented and then in the second stage the GUI was added.

During the first meeting we figured out the classes that we would need to add for the FSM. We coded WareState.Java and WareContext.java as group, where the adjacency matrix was the main thing that needed considerable discussion and planning. After those classes we completed then we divided the four major classes LoginState.java, ManagerState.java, SalesState.java and ClientState.java amongst four of us. I took ManagerState class.

For the FSM stage all I needed to do was separate the manager actions from the old UserInterface.java class and put them together in the ManagerState.java with proper state change calls at appropriate places. Later, a IOHelper class was also created for common methods. For GUI implementation, we completed the login part in the first week and then did the GUI for Manger, Salesclerk and Client individually following similar code implementation.

Throughout this Project I was the owner of the MangerState.java. This class holds all of the manager actions which includes

* add a client, addClient()
* add a manufacturer, addManufacturer()
* delete a manufacturer/supplier, deleteSupplier()
* display all manufacturers, showManufacturers()
* display all clients, showClients()
* go to salesclerk menu, salesMenu()

(also a logout option to go to main screen, logout())

All of these functionalities (methods) are local to the Manager State. They can only be performed by entering the manager state. The methods of this class changed a little bit between FSM implementation and GUI implementation, but the overall process and the theme remained the same.

This class `ManagerState.java` implements one of four major classes of the Warehouse system (states: login, manager, salesclerk, client). This class is dependent on login state, as it can only be entered through login state. But once the system enters the Manager State, it can be entered through the salesclerk state as well if salesclerk state was entered via Manager’s Menu. In this sense, if this entire class was taken out of the program, the program would still work, it would just not have any of the manager’s menus.

**Individual Comments**

This Project gave been an opportunity to gain some experience on something that I have never done before: programming with GUI. Again with the help of the lecture in the class and the library code that was provided the project went pretty smooth. Most of my problems were resolved by either discussing with the group members, or referring to the library code that was given by the professor.

Challenges:

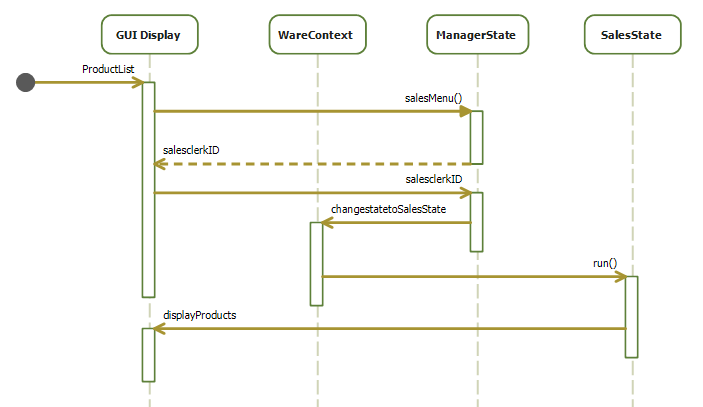
* I am still having some problems designing the sequence diagram with FSM, I do not find as intuitive as sequence diagram that we did earlier
* This most of this concept is very new to me, I need more practice with the Java GUI to increase my code quality

Task Division:

I also think the task was divided evenly among the group members for this Project. A few classes were coded together during the group meetings and other major classes were divided evenly. One of the group members (Eric) had prior experience working with similar projects. So he was more vocal and also resourceful for the project. But most of the tasks were divided evenly in a fair manner and all of the group members did their task on time. Also all of them were present in all of the planned meetings and provided helpful inputs in the decision making process.

**Sequence Diagram**

**Use Case: Show Product List (salesclerk menu) while in Manager State**



Here, changestatetoSalesState has been renamed to salesmenu() method in ManagerState.java since Manager State only goes to Sales State directly, but not the Client State.