Project 1 Final Report

This software implements a system that tracks the actions performed by a warehouse. This warehouse contains a list of products that the warehouse carries, a list of manufacturers that supply products, and a list of clients that order products from the warehouse. Along with storing this information, a user can perform several actions to interact with the warehouse, such as placing orders with the manufacturer and accepting payment from a client.

The software is stored on the csci2 machine under the account cs430118. It is located in the directory /Desktop/Project2 and can be compiled with the java compiler with the command: “javac warehouse.java”. To run the program, enter: “java warehouse”.

The work was equally shared among all members. Tasks were distributed equally during group meetings. It was tough to meet outside of the set meeting time since we all have different time availabilities. To make up for this, we were able to set up and SVN where we could each work on the project on our own time.

The program consists of 16 files, ranging from Client, Manufacturer, and Product classes, to the User Interface and Warehouse implementations.

The user begins in the “S1 Choose a User” state. From here, the user selects which state they would like to migrate to. The choices include “S2 Manager Login”, “S3 Salesman Login”, and “S4 Client Login”.

When the user enters the “S2 Manager Login” state, they are prompted to enter a username and password. Once entered, the system searches through the list of usernames and checks the corresponding password. 3 checks are made: 1) Does the username exist? 2) Does the username have manager privileges? 3) Does the password match? If any of these checks fail, the system displays an error message and prompts the user to try again. While in the login state, the user can opt to cancel, at which point the user returns to the previous state.

The “S3 Salesman Login” and “S4 Client Login” work the same way for Salesman and Client respectively.

Once the Manager successfully logs in, they are presented with a number of manager options, such as “S12 Modify Price”, “S13 Create Link between Manufacturer and Product”, and “S11” Order from Manufacturer”. The user can select those states, at which point they will enter that state. Alternatively, if the user selects to log out, they will return to the previous state. The user can also choose to switch to Salesman or switch to Client, where they will enter the “S3 Salesman Login” state or “S4 Client Login” state.

Once the user enters the “S3 Salesman Login” state, they will login just like the previous “S2 Manager Login” state. Upon successful login they are presented with a number of salesman options, such as “S9 Add Manufacturer”, “S6 Print Product/Quantity”, “S5 Receive Order”, “S7 Add Client”, “S8 Add Product”, and “S10 Manufacturers for a Product”. The user can select those states, at which point they will enter that state. Alternatively, if the user selects to log out, they will return to the previous state. The user can also choose to switch to Salesman or switch to Manager, where they will enter the “S3 Salesman Login” state or “S2 Manager Login” state.

Once the user enters the “S3 Client Login” state, they will login just like the previous “S2 Manager Login” state. Upon successful login they are presented with a number of client options, such as “S16 Check Price”, “S15 Make Order”, and “S14 Account Info”. The user can select those states, at which point they will enter that state. Alternatively, if the user selects to log out, they will return to the previous state. The user can also choose to switch to Salesman or switch to Manager, where they will enter the “S3 Salesman Login” state or “S2 Manager Login” state.

Add Client Sequence Diagram:



Add Client Use Case:

|  |  |
| --- | --- |
| 1. Receive application from a client |  |
| 1. Worker enters 1 to the system |  |
|  | 1. System asks for information on the client: name, address, etc. |
| 1. The worker fills out the requested data fields |  |
|  | 1. The system saves the data entered by the worker and generates a new unique id for that client. |

Add Client Description:

This case was authored by Joseph Donabauer. There were not really any problems encountered during implementation of our add client class, it was very similar to the add member example from the library example so most of the design choices were taken from the example and modified to fit our uses.

Place an order with a Manufacturer Sequence Diagram:



Place an order with a Manufacturer Use Case:

|  |  |
| --- | --- |
| 1. The clerk initiates the system to place an order |  |
|  | 1. The system ask for order details |
| 1. The clerk enters product details which include manufacture name, productId, orderId and quantity. |  |
|  | 1. The system then checks the order and make sure that the order placed is the right one. If the order placed is right one then the system provides again provides the details including the price to the clerk to finalize the order |
| 1. The clerk finalized the order and place the order |  |
|  | 1. The system accept the order and records the transactions |
| 1. Clerk exit the system |  |

Place an order with a Manufacturer Description:

This case was authored by Emilio Rescigno. To start, the place manufacturer order option is selected in the user interface. That in turn collects the manufacturer ID, product ID, and product quantity. With these three pieces of information, a function is called inside the warehouse class with these three parameters. This function then retrieves the corresponding Product and Manufacturer objects based on their IDs, and creates a new ManufacturerOrder object. This new object is then placed in lists inside both the corresponding manufacturer and product.

The most difficult part when implementing this function was ensuring that the lists inside both the Manufacturer and Product classes were updated properly and that data did not get out of sync. Filling manufacturer orders was many degrees more complex.

Receive an order from a Client Sequence Diagram:



Receive an order from a Client Use Case:

|  |  |
| --- | --- |
| 1. Worker receives an order from a client (customer) 2. Worker presses the “place order” button on the system |  |
|  | 1. The system asks for the client ID |
| 1. The worker enters the client ID |  |
|  | 1. The system asks for details of the order (list of products and number of each one) |
| 1. The worker enters the items and quantity of each item |  |
|  | 1. The system checks product availability and determines if the order can be fulfilled. If it can be fulfilled, subtract quantity from inventory and send a packing list to the warehouse floor.   If the request is unable to be fulfilled, subtract remaining balance from inventory, send request to the warehouse floor to be packed, and place remainder of order on backorder.  Display results of the order to the worker. |

Receive an order from a Client Description:

This case was authored by Jordan Musselman. It is very straight forward and there wasn’t many issues in implementing this case. If the user wants to perform this command, they select the option to place an order and then enter all relevant information. The system needs to use the given information to search the warehouse for a particular product and associate it with the client. There are checks to see if the products are in stock and if not, may need to be placed on backorder but this problem was not to difficult to implement.

Receive payment from a Client Sequence Diagram:



Receive payment from a Client Use Case:

|  |  |
| --- | --- |
| 1. The clerk initiates system to accept payment from the customer. |  |
|  | 1. The system ask for details |
| 1. The clerk enters the customer id and order id |  |
|  | 1. The system checks the order and shows the order detail with amount due to the clerk |
| 1. The clerk enters the amount received from the customer and the method of payment |  |
|  | 1. The system deducts the amount paid from the due amount and shows the remaining balance to the clerk. |
| 1. The clerk checks the receipt and finalize the payment |  |
|  | 1. The system records the transactions |
| 1. The clerk exits the system. |  |

Receive payment from a Description:

This case was authored by Kunga Sherpa. The use case and the sequence diagram for receiving payment from a client do not completely resemble with each other. However the core things of both are same. For the use case it is simply an interaction between the system and the users. At each level user enters certain information and the system responds to the request accordingly. I did not face any difficulty in writing the use case. For the sequence diagram every class is labeled in a rectangular box above the vertical line. The vertical line itself represents the class and the arrowhead represents the process. The vertical rectangular box represents the activeness of the class i.e. the class becomes active at the start of the rectangular box and become inactive at the end of rectangular box. Writing the exact process for each class and creating interaction among the different classes were the difficult part while creating the sequence diagram.

Individual Reports

Joseph Donabauer:

Opinion of the project – I feel like this was a good project to tackle, I have worked at a warehouse in the past so I understand that it is a real life application and different parts of the project that are required and nice to have. When the group was able to meet I feel that we worked quite well together, however the times we are available do not work very well so it is difficult to get together. I feel that I have learned a bit about java during this program and more information about objects and how they are used to organize a project more elegantly. The finished product seems to work well, there are some cases where if invalid input is entered strange things may happen but I’m not sure. I feel like I’d be able to work on this type of projects in the future.

Group Dynamic - the days that we planned to meet were: Feb 6, Feb 13, Feb 20, Feb 27, and March 6. I missed the meeting on the 27th due to a family obligation however there were several other informal meetings that I showed up to.

1. I feel that the group worked well whenever we were able to meet together
2. I feel that all members tried to do whatever they could to complete the project and I feel that Emilio R went out of his way to be an all-star on the team.
3. The communion wasn’t the greatest, were all had each other’s phone numbers but should do a better job replying to emails in a timely manner.
4. I think our project more forward just fine however it was helpful to have the due date pushed back a few days.

Kunga Sherpa:

In this project I learned a great deal of stuff about object oriented programming. The text book itself was immensely helpful in understanding the codes and the whole project. Design patterns, serialization, object collections were the ones that I learned most about the project. There were several projects that I have pursued in my previous semester in java. And turning back I felt I could have done that project more effectively and quickly had I implemented the things I learned in this project. Moreover working with a new group was also an extra experience for me. Since most of our team members do not a common schedule to meet, we kept in contact and divided the work load equally.