Metro State University, St. Paul, Minnesota
ICS 372 Section 02 Object-Oriented Design and Implementation (Hybrid)
Group Project 1

Goals

- Gain experience working in groups
- Use OO analysis, design, and implementation techniques

Due Date

11:59 PM on March 17, 2023

Problem

Create, test, and document a system for an application with business processes and other requirements described below.

Business Processses

A small co-op grocery store works by having members join it by paying a certain fee. The business processes are:

1) **Enroll a member**: the system enrolls a member and remembers him/her; it keeps track of the name, address, phone number, date joined, and the amount paid as fee. These values are all entered at the time of member creation. Also, the system creates and maintains a unique id for each member.

2) Remove a member: a member may be removed; the system would need the member's id for this purpose.

3) Retrieve member info: Given a member name, the system displays the member's address, fee paid, and id. If there is more than one member with the same name, print all such members.

4) **Add products:** From time to time, the store decides to carry a new set of products. The products must then be added one by one. Relevant attributes are: product name, id, current price, and a minimum reorder level for the product. Once the quantity in stock reaches the reorder level or below, the product will be automatically reordered. No two products have the same name. So if the store carries coconut milk, there could be products such as "Low-value coconut milk" and "Unsafeway coconut milk."

When the user tries to add a product, the system orders the product. The quantity ordered is twice the minimum reorder level. For example, assume "California Orange Juice" is a new

product that is being entered and that the reorder level is 100 units. The system automatically generates an order for 200 units.

Each item is ordered separately and received as a separate shipment.

5) **Check out a member' cart**: The member comes to the check-out counter with a cart of grocery items. The cashier inputs the product id and quantity for each and every item in the cart. The system computes, computes the total price. All members pay by cash. The system must display the individual items, the number of units, unit price, and price for the item. For example, if the user purchases three gallons of milk and two loaves of bread, the display might be

Milk	3	$4.50	$13.50
Bread	2	$1.50	$3.0
Total			$18.00

The total price must also be displayed.

When a customer is checked out, reorder any product whose level reaches the reorder level (or below the level). Reorder 2 * reorder level. Also display a message saying that the item will be reordered. If an order has already been placed for this product and has not been fulfilled, do not reorder.

For example, suppose the reorder level for Milk is 50 (bottles). Suppose on August 1, the quantity on stock falls to 40 after a certain purchase. You will now create an order for 100 bottles of Milk. Suppose before this order is fulfilled, on August 2, the stock falls from 40 to 35 after another purchase. The system should detect that an order has already been placed and not yet fulfilled, and should not reorder the product.

6) Retrieve product info: Given a product name, the system displays the product's id, price, and stock in hand.

7) **Process shipment**: Whenever there is delivery of items from a supplier, the stocks must be updated. The system needs the product id to locate the product and quantity delivered to the store. Multiple shipments may be processed, one by one. Display the product id, name, and the new stock. Remember that each shipment has exactly one product.

As stated earlier, each item is ordered separately and received as a separate shipment.

8) **Change price**: Changes the price of a product given its id. The system displays the product name and the new price.

9) Print transactions: Given the id of a member and two dates (input in the mm/dd/yyyy format), the system prints information (explained later) about each visit made by the corresponding member between the two dates (including both dates). The system must verify that the member id is valid and that the first date is not after the second date; if these conditions are met, the system displays the following information for each visi:

a) The date on which the user visited the store
b) The total price paid during that visit

A user may visit the store multiple times on a single day. In that case, there will be line items for each visit.

10) List all members. List name, id, and address of each member.

11) List all products. List product name, id, current price, and a minimum reorder level for the product.

12) List all outstanding orders. List for each order that has not been fulfilled, the product name, id, and amount ordered.

13) Save: Saves all data to disk.

14) Retrieve: Retrieves a given file and use it. Only applicable before any other command is issued.

Make sure that wherever there is reasonable common functionality, factor out code fragments from similar classes and move them to/create super classes. Use generics where applicable.

The User Interface and Other Aspects

For the purposes of ensuring uniformity in grading, I ask the following.

1. The interface must be non-GUI, but command driven, just like the library system. I should be able to invoke the business processes by typing in a number as listed under the business processes above: 1 for enrolling a member, 2 for removing a member, etc. Also use 0 for exit and 15 for help.
2. If a use case asks for processing multiple entities, follow that requirement. If a use case asks for processing a single entity (such as adding a member), be sure not to add multiple members in the corresponding design/implementation.
3. When the program starts, it should give an option to look for and load existing data on stable storage. If the user answers in the affirmative, that data should be loaded and used. Do not assume any specific directory structure on my system: use the current directory.
4. In general (that is unless specified elsewhere), the look and feel of the interface should be similar to that of the library system. (Obviously, the functionality is different.) This includes the nature of inputting commands and information, displays, etc.
5. Error messages must be as specific as possible. For example, if an order request is invalid, display the precise reason: bad component id, bad supplier id, quantity <= 0.

When the user interface starts, if the user does NOT wish to use a saved file, the program should give an option to programmatically set up a test bed, by prompting as follows.

Do you want to generate a test bed and invoke the functionality using asserts?

If the user answers **y** or **yes**:
1) The program first creates a number of products (at least 20) and members (at least 5).
2) After completing Step 1, the program invokes the façade methods for business processes numbered 1, 2, 4, 5, 7, 8. Using assert statements, it checks the returned information and ensures that they are correct. This step should be a thorough test of the corresponding business processes. This test bed should be available for further interactive testing. (See Step 3.)
3) Finally, the program presents the command line interface, so the user can enter more test data using the interactive approach.

Things to be Submitted

You need to submit two files: one for analysis and design, and a second one for implementation.

Analysis and Design

Submit a single PDF document that contains all of the following.

1. A use case for business processes 1, 4, 5, 7, and 8. There is no need to submit use cases for the others.
2. The conceptual class diagram.
3. Sequence diagrams for use cases corresponding to business processes 1, 4, 5, 7, and 8.
4. The physical class diagram.

The use cases must be typed.

You can talk to me to get any clarifications you need, especially with respect to the requirements.

The sequence and class diagrams must be drawn using an appropriate software tool. I will ignore all parts of the document that are hand-written or hand-drawn.

I will also ignore any document that is in any other format (Word, GIF, JPEG, etc.) If you submit multiple documents, I will choose one of the PDF files and ignore everything else. Be sure to ensure that the information is all in the portrait mode wherever possible. Do not have any part of the information cut off or unviewable in any way.

Draw the diagrams clearly. The following are common mistakes and improprieties I have observed over the years.

1. Placing the classes awkwardly: Class A extends class B, but A is not placed below B.
2. Not using proper lines for class extension, composition, etc. (I suggest that you

use Visio, because it labels the icons for interface implementation, class extension, composition, etc.; that gives you better clues as to which stencil to use for what.)
3. Not identifying the relationships properly.
4. Not drawing the lines (for method calls) in the sequence diagrams horizontally.
5. Not using the proper notations (+, -, #) for public, private, and protected members.

Such mistakes will result in loss of credit.

Implementation

Submit the entire application as an Eclipse project using Java modules. The communication between the façade and the user interface must use techniques we employed for the library system. The implementation must also employ safe coding practices we discussed in class. All code (classes, interfaces, constructors, and methods) must be properly formatted and documented. You are welcome to use the library code provided on D2L and adapt the functionality. The documentation, naming conventions, and code formatting must follow the coding conventions we have discussed before.

Document and lay out your code properly as specified under the CodingStandards.pdf document. Refer to the library code for more examples.

Remember that I have to read a lot of code and if your code does not display properly on y machine, you will not get full credit.

I will provide **no** debugging support. You must resolve such issues within the group.

Responsibilities

Every student must write at least one use-case and draw at least one sequence diagram. Label each sequence diagram and use case with the name of the person who did the work. If a student does not draw any use cases or draw any sequence diagrams (as per the labels on these entries in the submitted document), that student will lose credit for those parts of the rubric.

Grading

Your assignment will be graded as written in this section.

1. The use cases reflect the business processes.
2. The sequence diagrams implement the use cases.
3. The class diagrams must be based on the information gathered from the sequence diagrams.
4. The Java code is based on the design.

Analysis (27 points)

The grade will be based on the use cases and the conceptual class diagram.

Component	Number of Items	Points per Item	Total
Use cases	5	4	20
Miscellaneous	1	3	3
Conceptual class diagram	1	4	4

To get full credit for a use case:

1. It should reflect the business process completely.
2. It should be written properly.

The miscellaneous component is to take care of the situation when several use cases have some (possibly different) minor errors (usually the way it is written as opposed to being faithful to the business process), that were not individually penalized because that would be unfair. For example, a single misspelling in one use case will not result in any loss of credit: I think that would be too harsh. But more than one instance of such errors or several single instances of different minor errors are penalized through this component.
Part of the credit for use cases is for presenting it properly. Unless the use case reflects the business process almost completely, there will be no credit for presentation.
Be sure to properly label the conceptual class diagram. You must have all conceptual classes in the diagram, the relationships must be properly labeled and the correct notations should be used.
Remember that conceptual class diagrams are not the same as physical class diagrams. This has been a confusion with a few groups in past semesters.

Each use case must be labeled with a proper name reflecting its functionality and the name of the student who wrote it. If a student does not have his/her name attached to at least one use case, that student will lose 7 points.

Design (36 points)

The grade will be based on the sequence and class diagrams.

Component	Number of Items	Points	
		Per item	Total
Sequence	5	5	25
Design quality	1	6	5
Class diagram	1	6	6

The sequence and class diagrams will be graded based on

- Quality of design
- Correctness with respect to the requirements

- Quality of the presentation

Each sequence diagram should correctly implement the functionality of the respective use case and should be properly drawn. If a sequence diagram is meaningless (w.r.t the use case), no credit will be given for it, even if it is well drawn.
Design quality will depend on how well you implemented the use cases and how well the classes are properly structured.

Each sequence diagram must be labeled with a proper name reflecting its functionality and the name of the student who drew it. If a student does not have his/her name attached to at least one sequence diagram, that student will lose 5 points.

Implementation (37 points)

This will be based on accuracy, program structure, coding and documentation, etc.

Criterion	**Point**
Class and method documentation	6
Correctness (My testing)	15
Secure coding	4
Automated testing	6
Coding conventions and structure	6

Grading of correctness and automated testing will be based on the following.
1. Are all use cases realized?
2. Is the user interface as specified?
3. Are classes designed as per design?
4. Are results displayed properly?
5. Does the implementation adhere to the principles of cohesion and coupling?
6. Is the user interface similar in feel to the library system?
7. Is there an automated testing facility? Does it test all the specified functions? Does the program pass those tests?

If you don't submit an Eclipse Java project as a zip file that opens correctly or is not executable directly, you will lose 10 percent of the credit for implementation.

Grading Issues Related to Group Work

Usually, all members of a group get the same grade. But I have had multiple groups disputes, which have often made grading a somewhat difficult process. I am forced to reserve the right to give students within a group different grades. This will not be done capriciously; I will inform the student involved before I give him/her a different grade.

You must submit a self- and peer-evaluation form by March 18. You must answer all questions. While there is no credit added to your score, non or incomplete submission

will result in a loss of grade (individually for the student and up to 5 points) for the project.