

Metropolitan State University, St. Paul, MN
ICS 372 Object-Oriented Design and Implementation
Group Project 2

1 Due Dates and Goals

1.1 Due:

11:59 PM on April 11, 2019

1.2 Goals:

1. Perform use-case analysis techniques to discover and specify the conceptual classes.
2. Use design principles to translate conceptual class design into an appropriate set of abstract and concrete classes and interfaces
3. Efficiently develop systems using design patterns including Facade and Singleton
4. Use principles of the agile methodology by following the Unified Process
5. Use the Unified Modeling Language to document work
6. Implement a design utilizing structures such as classes and interfaces,
7. Work in small groups
8. To employ Java coding standards.

2 The Problem

You have to make modifications to the problem in Group Project 1.

The major changes are as follows.

1. The system now supports contributions through bank accounts as well. Bank accounts behave exactly the same way as credit cards as far as the information they store. The organization must still distinguish between them because the way they are charged are different.
2. A single use case still adds credit cards and bank accounts. (See below.)
3. The transactions for a donor are preserved even after the donor is removed from the system.
4. For each payment method (credit cards and bank accounts), the organization keeps track of the number of donations and the total amount charged.
5. The organization incurs expenses. It keeps track of all expenses. Each expense has a description, amount, and the date the expense was incurred.

6. The organization keeps track of the amount of money it has. This is the total amount received through donations less the total expenses incurred so far.
7. The system is set up to handle new functionality on the payment methods. One such functionality that you must implement using this feature is the ability to list all payment methods (that is, credit cards and bank accounts) for which more than a certain amount has been collected so far. (See below.)

3 The New or Changed Business Processes

The business processes are:

1. (Modified) Add a payment method: The actor gets the option to add a bank account or a credit card. After choosing the payment mechanism type, the information to be supplied by the actor are the donor id (the corresponding donor is the credit card or bank account holder), the credit card number or bank account number , and the amount to be donated each time a donation occurs.
2. (Modified) List all transactions: the system displays the credit card or bank account number, amount, and the date (month, day, and year) for every transaction ever carried out.
3. (Modified) List a specific donor. The actor will provide the donor id. The system must display the donor name, phone and credit card or bank account number and amount of all payment methods associated with the donor. No transactions should be output.
4. (Modified) Remove a specific donor: The actor will provide the donor id. If the donor with this donor id exists, all information associated with the donor **excluding** the transactions for that user are removed.
5. (Modified) Remove a credit card: The actor will provide the donor id and the credit card number. If the entries are valid, the information for that credit card (account number, amount, etc) will be removed. The transactions associated with the card are **not** removed.
6. (New) Remove a bank account: The actor will provide the donor id and the bank account number. If the entries are valid, the information for that bank account (account number, amount, etc) will be removed. The transactions associated with the account are **not** removed.
7. (New) Add one or more expenses. The actor supplies an expense type (a string) and the amount for each expense.
8. (New) List organization info: List the total amount ever donated, the total expenses, and the current balance.
9. (New) List payment method info: The actor supplies a threshold amount and the system displays for each bank account and credit card, the number of transactions and the amount received through it, provided the amount received is more than the threshold amount.
10. (New) List all expenses. Display all attributes.

3.1 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 (and other stuff) by typing in a number as specified below.

Command	Process	Result; Not all prompts and interactions shown
0	Exit the application	A prompt to save (if and only if the data is “dirty”)
1	Add a donor	The information (donor data) added
2	Add a payment method	The information (card data) added
3	Process donations	Total amount received as donation
4	List Transactions	One transaction per line (see business processes)
5	List Donors	One donor per line (see business processes)
6	List Donor	Donor information on one line (see business processes)
7	Remove donor	Success or failure info
8	Remove credit card	Success or failure info
9	Remove bank account	Success or failure info
10	Add expenses	Success or failure info
11	Organization info	Total amount donated, spent, and balance
12	List payment method info	info for each payment method
13	List all expenses	info for each expense
14	Save	Success or failure info
15	Help	A helpful list of commands (See the Library system behavior)

2. 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.
3. In general (that is unless specified elsewhere), the 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, donor id, etc.

3.2 Design and Implementation Requirements

Your design and implementation must adhere to the following.

1. If two classes have similar functionality, their common code should be extracted to a superclass and the subclasses must be configured appropriately to maximize reuse.

2. Generics and other refactoring approaches we saw with the library system must be appropriately employed to manage collections.
3. The Visitor pattern must be used to implement the use case for listing payments that are above a given threshold.
4. If needed, move methods and extract methods.
5. If appropriate, use polymorphism to replace conditionals.
6. Use the singleton pattern where appropriate.
7. Use proper access specifiers.
8. Employ the `equals` method correctly.

4 Deliverables

You need to submit three files: one for analysis, one for design, and a third one for implementation.

4.1 Analysis

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

1. A use case for each of the five business processes listed as New. There is no need to submit use cases for the ones marked as modified, and save, exit, and help.
2. The conceptual class diagram for the entire system (not just the new stuff).

The use cases must be typed and the class diagram drawn using an appropriate software tool. I will ignore all parts of the document that are not typed. 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.

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

4.2 Design

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

1. A sequence diagram for use cases corresponding to the business processes numbered 6, 7, 8, 9, and 10, under Section 3.
2. The physical class diagram.

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

4.3 Implementation

Submit the entire application as an Eclipse project. All code 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 and documented before. Refer to the library code for more examples.

5 Grading

Your assignment will be graded as written in this section and given in more detail under the rubrics attached to the submission folder.

While doing the project ensure that

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

5.1 Analysis (26 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	3	15
Overall	1	3	3
Conceptual class diagram	1	8	8

To get full credit for a use case:

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

Take a look at the rubrics for details.

The miscellaneous component is to take care of the situation when several use cases have some (possibly different) minor error (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 repeated instance of such 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, 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.

5.2 Design (36 points)

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

Component	Number of Items	Points	
		per Item	Total
Sequence diagrams	5	2	10
Design quality	1	14	14
Class diagram	1	12	12

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. As long as the design is understandable, I will not consider the presentation (how well a sequence diagram is drawn) itself into account.

Look at the rubrics for more details on grading.

5.3 Implementation (38 points)

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

Criterion	Points
Documentation	6
Correctness	26
Coding conventions and structure	6

Correctness 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. Is the interface similar in feel to the library system? (For example, is the donor id relatively simple to key in?)

Be sure to take a look at the rubrics for this part of the project. I will publish this around March 28.

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

6 Agile Development

I suggest that you analyze, design, and implement the code using the agile development approach. Perhaps you can write one use case: to add donors and credit cards, draw the sequence diagrams quickly (perhaps hand drawn) and implement the code (maybe working in a pair) and test that out. A (quick) creation of the user interface and the facade will be immensely useful for getting off to a great start.

One team member can play the role of the end-user.

After the above two are completed, you can add a few more use cases and eventually complete the process in three or four iterations.

The wikipedia article on agile development has some interesting and useful sections.