..

Metro State University,

St. Paul, Minnesota ICS 372-02

Object-Oriented Design and Implementation

• Gain experience working in groups

• Implementing your design of Project Part1

Due Date 11:59 PM on November 19th, 2023

The User Interface and Other Aspects

For the purpose 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.

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.

Submission:

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 my machine, you will not get full credit.

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

Grading:

The Java code is based on the design. The grading will be based on accuracy, program structure, coding and documentation, etc.

See details below:

|  |  |
| --- | --- |
| Criterion | Points |
| Classes and methods documentation | 20 |
| Correctness/Testing | 60 |
| Secure coding | 10 |
| Coding conventions and structure | 10 |

Grading of correctness /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?