CCPROG3 MCO1 Program Specifications

Program Overview – Enhanced Trading Card Inventory System

Note: *MCO1 and MCO2 is to be done by pair, and by the same pair of students barring any reports of freeloading or decision by the pair to split. In the event of a split for any reason, the students who used to be in the pair would now need to work solo; they may not join an existing group or pair up with a new partner.*

Your task for MCO2 is to enhance the Trading Card Inventory System (TCIS). After developing your original TCIS, it was noted that it didn't have any functionality to sell cards. Furthermore, there seems to be specialized binder types and deck types that, if curated properly, can be worth more than regular binders and decks. Your task is to extend (not modify) your existing TCIS to accommodate these new functionalities. Functionality of the original TCIS must remain, unless the functionality listed below otherwise specifies.

Furthermore, your enhanced TCIS must have a working and functional graphical user interface (GUI), preferably using Java Swing.

Listed below are the additional functionalities of the enhanced TCIS:

1. **Sell Cards**
   This feature allows a collector to sell cards. A card may only be sold for its **real value** (i.e. base value increased by a multiplier based on its variant if applicable). Once sold, a card is removed from the player's collection. Note that cards can only be sold from the collection; cards in binders and decks cannot be sold (the binder or the deck can be sold however, see the appropriate sections below). The money the collector earns from the sale must be tracked by the TCIS; it will go to the collector's total money.

2. **Binder Types**
   Binders now will have types; they can be either of the following:
   a. **Non-curated Binder** – a basic binder. Non curated binders don't have any restrictions, and can have any set of cards in them. Non-curated binders **cannot be sold**.
   b. **Sellable Binder** – this kind of binder is like the non-curated binder, but there is an expectation for these binders to be sold. Selling a sellable binder may be subject to handling tax, depending on the kind of sellable binder. These kinds are:
      i. **Pauper Binder** – a pauper binder can only contain common and uncommon cards. When sold, these binders will incur no handling tax. It's price will simply be the total cost of the cards inside the binder.
      ii. **Rares Binder** – a rares binder can only contain rare and legendary cards of any variant. As rare and legendary cards are typically more expensive, when sold, the value of a rares binder will be the total value of the cards in the binder (real value, see **Sell Cards**) with an additional 10% handling fee (this is 10% of the total real value of the cards in the binder). The handling fee goes to the total money of the collector (see **Sell Cards**).
      iii. **Luxury Binder** – a luxury binder exclusively contains cards whose variant is not normal (i.e. extended-art, full-art and alt-art only). By default, the value of a luxury-binder is the total real value of the cards in the binder. However, collectors can set the total price of a luxury binder themselves. The price set must not be lower than the total real value of the cards inside the binder, however. If the collector tries to set the value to below the total real value of the cards in the binder, they will be notified and the value would not be set. Note that when the actual sale happens, a 10% handling fee is added to the cost of the binder, similar to a rares binder. This 10% is based on the value set by the collector. The handling fee goes to the total money of the collector (see **Sell Cards**).
   c. **Collector Binder** – like a basic binder, but can only contain rare and legendary cards whose variant is not normal (i.e. extended-art, full-art and alt-art only). As they are like basic binders, they cannot be sold.

3. **Manage Binder**
   Given the new binder types, some enhancements have been added to the Manage Binder functionality:
   a. **Create a new Binder** – works like before, but now the user must indicate the type of binder.
   b. **Delete a Binder** – works like before; no change.
   c. **Add/Remove Card to/from the Binder** – works like before, but if the user attempts to add an illegal card (i.e. a card that does not meet the requirements of a binder), the user is notified and the card is not added to the binder.
   d. **Trade Card** – trades *can only be performed* on **non-curated and collector binders**. Trades cannot be performed in other kinds of binders.
   e. **View Binder** – works like before; no change.
   f. **Sell Binder** – this option must only be available for binders that can be sold (i.e. you cannot sell a non-sellable binder). When sold, the binder is removed from the collection, along with all the cards inside of them, and an amount of money is added to the total money of the collector equal to the cost of the binder that was sold (see **Sellable Binder**).

4. **Deck Types**
   Like binders, decks now have types. There really is only two kinds of decks:
   a. **Normal Deck** – The usual deck. This has all the functionality of decks in MCO1.
   b. **Sellable Deck** – This deck is meant to be sold. When sold, the user gains an amount of money equal to the real value of the cards in the deck (see **Sell Cards**).

5. **Manage Decks**
   Given the new deck types, some enhancements have been added to the Manage Deck functionality:
   a. **Create a new Deck** – works like before, but now the user must indicate the type of deck.
   b. **Delete a Deck** – works like before; no change.
   c. **Add/Remove Card to/from the Deck** – works like before; no change.
   d. **View Deck** – works like before; no change.
   e. **Sell Deck** – this option must only be available for decks that can be sold (i.e. you cannot sell a non-sellable deck). When sold, the deck is removed from the collection, along with all the cards inside of them, and an amount of money is added to the total money of the collector equal to the cost of the deck that was sold (see **Sellable Deck**).

Other Requirements

1. The design and implementation of the solution should…
   - Conforms to the specifications described in MCO1, extended with the functionality above.
   - Exhibit proper object-oriented concepts, like encapsulation, abstraction, inheritance and polymorphism, and as such must use super classes, abstract classes and interfaces as and when necessary.
2. Interaction with the user (i.e. input and output) should be through a **graphical user interface (GUI)**.
3. Javadoc should be prepared for the application.

General Instructions

Deliverables

The deliverables for both MCOs include:

1. Signed declaration of original work (declaration of sources and citations may also be placed here)
   - See Appendix A for an example
2. Softcopy of the class diagram following UML notations
   - Kindly ensure that the diagram is easy to read and well structured
3. Javadoc-generated documentation for proponent-defined classes with pertinent information
4. The individual Java files of the system.

Submission

All deliverables for the MCO are to be submitted via **Canvas**. Submissions made in other venues will not be accepted. Please also make sure to take note of the deadlines specified on Canvas. No late submissions will be accepted.

Grading

For grading of the MCO, please refer to the MCO rubrics indicated in the syllabus or the appendix.

Collaboration and Academic Honesty
This project is meant to be worked on as a pair (i.e. max of 2 members in a group). In exceptional cases, a student may be allowed by their instructor to work on the project alone; however, permission should be sought as collaboration is a key component of the learning experience. Under no circumstance will a group be allowed to work on the MCO with more than 2 members.

A student cannot discuss or ask about design or implementation with other persons, with the exception of the teacher and their groupmate. Copying other people's work and/or working in collaboration with other teams are not allowed and are punishable by a grade of 0.0 for the entire CCPROG3 course and a case may be filed with the Discipline Office. In short, do not risk it; the consequences are not worth the reward[1]. Comply with the policies on collaboration and AI usage as discussed in the course syllabus.

Documentation and Coding Standards
Do not forget to include internal documentation (comments) in your code. At the very least, there should be an introductory comment and a comment before every class and every method. This will be used later to generate the required External Documentation for your Machine Project. You may use an IDE or the appropriate command-based instructions to create the documentation, but it must be PROPERLY constructed.

Please note that we're not expecting you to add comments for each and every line of code. A well-documented program also implies that coding standards are adhered to in such a way that they aid in the documentation of the code. Comments should be considered for more complex logic.

Bonus Points
To encourage that usage of version control, please note that a small portion of the bonus points for MCO2 will be the usage of version control. Please consider using version control as early as MCO1 to help with collaborating within the group.

Resources and Citations
All sources should have proper citations. Citations should be written using the APA format. Examples of APA-formatted citations can be seen in the References section of the syllabus. You're encouraged to use the declaration of original work document as the document to place the citations.

Demonstration Delivery
The mode of delivery of the demo for MCO1 is left to the discretion of your instructor. All members are expected to be present in the video demonstration and should have relatively equal parts in terms of the discussion. Any student who is not present during the demo will receive a zero for the phase.

During the MP demo, it is expected that the program can be compiled successfully and will run. If the program does not run, the grade for that phase is 0. However, a running program with complete features may not necessarily get full credit, as implementation (i.e., code) will still be checked.

Other Notes
You are also required to create and use methods and classes whenever possible. Make sure to use Object-Based (for MCO1) and Object-Oriented (for MCO2) Programming concepts properly. No brute force solution. When in doubt, consult with your instructor.

Statements and methods not taught in class can be used in the implementation. However, these are left for the student to learn on his or her own

[1] What is a measly passing grade compared to a life-long burden on your conscience?

Appendix A. Template for Declaration of Original Work

Declaration of Original Work

We/I, [Your Name(s)] of section [section], declare that the code, resources, and documents that we submitted for the [1st/2nd] phase of the major course output (MCO) for CCPROG3 are our own work and effort. We take full responsibility for the submission and understand the repercussions of committing academic dishonesty, as stated in the DLSU Student Handbook. We affirm that we have not used any unauthorized assistance or unfair means in completing this project.

[*In case your project uses resources, like images, that were not created by your group.*] We acknowledge the following external sources or references used in the development of this project:

1. Author. Year. Title. Publisher. Link.
2. Author. Year. Title. Publisher. Link.
3. Author. Year. Title. Publisher. Link.

By signing this declaration, we affirm the authenticity and originality of our work.

Signature and date	*Signature and date*
Student 1 Name ID number	Student 2 Name ID number

[*Note to students: Do not submit documents where your signatures are easily accessible. Ideally, submit a flattened PDF to add a layer of security for your digital signature*