GD/CCPROG3 Machine Project Specifications
1st Term, AY 2025-2026

I. Introduction

The Convenience Store Simulation aims to model the operations of a small retail store using object-oriented programming principles in Java.

II. Convenience Store Application

The system should allow the user to manage a catalog of products available in the store, simulate customer interactions, handle purchases, and track sales. It should be capable of maintaining an inventory of various **products**, each with attributes such as **name, price, quantity in stock, and category** (e.g., food, beverage, toiletries). Features such as **expiration date, brand, or variant (e.g. soda zero, soda light, soda regular)** may be included to products when applicable.

III. Details of important object attributes and behaviors

The convenience store employee should be able to add new products to the inventory, restock existing items, and update product information. Inventory tracking should be dynamic, automatically reducing item quantities after purchases and flagging low-stock items for restocking. For additional features, the system may support perishable goods with expiration tracking.

Customer simulation should involve generating or allowing user input for a customer who selects products to purchase. The system should support adding multiple items to a shopping list or cart, viewing the running total, and proceeding to checkout. During checkout, the system should calculate the total cost, including tax (VAT), accept payment input, and compute change. You must also include a membership card that a customer may or may not have. Swiping such cards during checkout transactions accumulate points that may later be used as discount amounts deducted from the total.

Example: For every P50 in the total awards 1 point. 1 point is equivalent to 1 peso.

Additional discount logic (e.g., senior discount, membership points) must also be implemented. The system must be able to generate a receipt at the end of a transaction, displaying itemized purchases, total cost, amount received, change due, and a timestamp. Receipts are to be displayed via console (for customer's copy) and saved to a text file for record-keeping.

IV. Graphical User Interface

The graphical user interface must be user-friendly, requiring the least amount of interactions to carry out an operation. At minimum, the program should present a clear, menu-driven interaction model, allowing users to perform core tasks without editing code. File handling must be incorporated for saving and loading inventory and customer data (if applicable). The products must be laid out for easier navigation for the customer when selecting products to buy. Payment transactions must be carried out similar to how it is performed in real-life.

Students are expected to apply object-oriented design principles throughout the development of the simulation. Proper use of encapsulation, inheritance, polymorphism, and abstraction should be evident in the structure of the system. The design should promote modularity, clarity, and reusability. Creativity and thoughtful extensions beyond the base requirements (such as customer profiles, employee login) are encouraged.

V. Minimum Requirements

5 product categories:

- Food (Snack and ready-to-eat)
- Beverages (Hot,Cold, Alcoholic)
- Toiletries (Soap, Shampoo, Beauty products)
- Cleaning Products (Detergent, Tissue, Hand sanitizers)
- Medications (over-the-counter medicines)

At least 5 types of items per category (e.g. under food category - sandwich, pastries, fried chicken, etc.)
Initial number of items per type must be 10
Items must be arranged in shelves.

Functionalities: Product selection to be purchased, payment transactions, Inventory maintenance

VI. Milestones (see section VIII for Deliverables)

a. MCO1 – due 9:00 pm, October 24, 2025 (F)

1. UML Class Diagrams for the whole application (users, products, and other classes that must be identified from the narrative)
2. Implementation of the class
3. Driver to test the basic application. No GUI display is expected. However, displays must be made in the Console.
   For example: Selection of products and payment transactions via command line interface.

b. MCO2 – due 12:00 nn, November 24, 2026 (M)

1. UML Class Diagrams (Object-Oriented)
2. GUI with Mouse-Controlled Inputs.
3. Complete implementation of the application

4. Using the Model-View-Control (MVC) design pattern.

VII. Deliverables

The deliverables for both MCOs include:

1. The design and implementation of the solution should:
   - Conform to the specifications described above
   - Exhibit proper object-based / object-oriented concepts, like encapsulation and information-hiding, etc.
   - **NOT** be derived or influenced from any use of Generative AI tools or applications
2. Signed declaration of original work (declaration of sources and citations may also be placed here)
   - See **Appendix A** for an example
3. Softcopy of the class diagram following UML notations (in pdf or png)
   - Kindly ensure that the diagram is easy to read and well structured
4. Javadoc-generated documentation for proponent-defined classes with pertinent information
5. Zip file containing the source code with proper internal documentation
   - The program must be written in Java
   - Include external libraries that were used (e.g., JavaFX)
6. Test script following the format indicated in **Appendix B**
   - In general, there should be at least 3 categories (as indicated in the description) of test cases per method (except for setters and getters).
   - There is no need to test user-defined methods which are ONLY for screen design (i.e., no computations/processing; just print/println).
7. For MCO1 only: A video demonstration of your program
   - While groups have the freedom to conduct their demonstration, a demo script will be provided closer to the due date to help with showing the expected functionalities.
   - The demonstration should also quickly explain key aspects of the program's design found in the group's class diagram
   - Please keep the demo as concise as possible and refrain from adding unnecessary information
8. Groups should back-up their projects. A softcopy of the final unmodified files (for each phase) should be sent to their own emails, apart from regular submissions of progress in AnimoSpace and/or Git.

VIII. Submission

All deliverables for the MCO are to be submitted via AnimoSpace. Submissions made in other venues will not be accepted. Please also make sure to take note of the deadlines specified on AnimoSpace. Late submissions will not be accepted.

IX. Grading

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

X. Collaboration and Academic Honesty

This project is meant to be developed as a pair for both phases (i.e., MCO1 and MCO2) barring any reports of freeloading or decision by the pair to split. 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. Questions about the project specifications should be raised in the Discussion page in AnimoSpace. 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 subject and a case may be filed with the Discipline Office. In short, do not risk it; the consequences are not worth the reward. Comply with the policies on collaboration and AI usage as discussed in the syllabus.

XI. 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 project via javadoc. You may use an IDE or the appropriate command-based instructions to create the documentation, but it must be PROPERLY constructed.

Please note that you're not expected to provide 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.

XII. Bonus Points

Bonus points will only be awarded for MCO2. The above description of the program is the basic requirement. Any additional feature will be left to the creativity of the group. Bonus points would be awarded depending on the additional implemented features. These additional features could include:

- 3D Walk-through user experience in the convenience store
- Additional features for the convenience store management such as daily income reports

Depending on the scale of the new feature, additional points will be awarded to the group. **The additional features must be implemented under the object-oriented paradigm**. However, **make sure that all the minimum requirements are completely and correctly met first; if this is not the case then no additional points will be credited despite the additional features.** To encourage the 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 group collaboration.

XIII. 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 the original work document as the document to place the citations.

Further, this is to emphasize that you DO NOT need to create your own product images for the project. You can just use what's available on the Internet and just include these into your project, just make sure to cite your sources.

XIV. Demo

Demo for MCO1 is via a video submission. 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.

In MCO2, the demo is live and will include an individual demo problem. The schedule for the demo will generally be during class time, but there may be other schedules opened by the faculty in case there is not enough time to accommodate everyone. Sequence (of who goes first in the class to do the demo) is determined by the faculty. Thus, do not be absent or late during announced demo days. A student or a group who is not present during the demo or who cannot answer questions regarding the design and implementation of the submitted project convincingly will incur a grade of 0 for that project phase.

During the demo, it is expected that the program can be compiled successfully in the command prompt 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.

XV. 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.

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.

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:*

1. *Do not submit documents where your signatures are easily accessible. Ideally, submit a flattened PDF to add a layer of security for your digital signatures*
2. *You may use the eSignature feature of Google Docs. See this template.*]

Appendix B. Example of Test Script Format

Class: MyClass						
Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
isPositive	**1**	**Determines that a positive whole number is positive**	**74**	**true**	**true**	**P**
	2	**Determines that a positive floating point number is positive**	**6.112**	**true**	**true**	**P**
	3	**Determines that a negative whole number is not positive**	**-871**	**false**	**false**	**P**
	4	**Determines that a negative floating point number is not positive**	**-0.0067**	**false**	**false**	**P**
	5	**Determines that 0 is not positive**	**0**	**false**	**false**	**P**