Department of Software Technology

Object Oriented Design and Programming (**CCPROG3**)
Major Course Output (MCO)
3rd Trimester, SY 2024 – 2025
Revision 1.2

JAVAJEEPS

A Coffee Truck Business Simulation

Introduction [1]

Starting a coffee business with a truck offers several advantages over a traditional brick-and-mortar store (i.e Starbucks). For one, it provides greater flexibility and lower overhead costs. Additionally, a mobile coffee business can be launched relatively quickly, allowing you to get up and running faster and start serving customers sooner.

For those who may be unfamiliar, a coffee truck is essentially a mobile coffee shop on wheels. It's a vehicle equipped with the necessary equipment to brew and serve high-quality coffee and other beverages. You may have seen similar concepts in action, such as the popular 'Jolli-jeeps' of the 80's and 90's in Makati, which are customised jeepneys that serve as mobile food vendors. Inspired by this iconic Filipino concept, our mobile coffee business could be aptly named '**JavaJeeps**' - a playful nod to the Jolli-jeeps, but with a coffee (and OOP) twist. Just like Jolli-jeeps, JavaJeeps would bring customers a unique and exciting experience, serving up great coffee on the go and making it an excellent option for events, festivals, and high-traffic areas."

As a tribute to this innovative business concept, we aim to replicate its success by designing and building a mobile coffee vendor. This will ultimately lead to a **simulated 'Coffee Truck Business'** - a project that showcases our version of a coffee truck. The detailed specifications are as follows:

1. The Main User Interface

In this simulation application, the user is first presented with the following options until the user chooses to **Exit** the application.

---

[1] Willy Jeep Coffee truck - https://preview.redd.it/found-this-little-willy-coffee-stand-in-colombia-v0-xc6405t5zy9a1.jpg?width=1080&crop=smart&auto=webp&s=a2ad1c3f55f51a1e04a959ea4ddf8f5366894226

1.1. Create a *Coffee Truck.*

The user can choose to create either a Regular Coffee Truck (*JavaJeep*) or a Special Coffee Truck (*JavaJeep+*).
Coffee trucks are strategically positioned at different locations across the city. Each truck is assigned to a distinct location[2], ensuring that no two trucks occupy the same spot.

These specially constructed vehicles are designed to provide customers with a range of coffee drinks and are equipped with all the necessary provisions, including equipment, ingredients, and other essential supplies.

Sections 2 & 3 below provide more details about the different types of coffee trucks.

Part of the creation process is the initial loadout or truck provisioning.
The initial loadout process involves assigning items (ingredients or cups) to specific storage bins on the truck, along with their quantities, before deployment. This allows users to customise the truck's configuration for its location. It also involves setting the prices for each product item, including add-on items for Special Trucks.

1.2. Simulate *Coffee Truck* features.

The user can simulate the following features of a specific truck:

- Sale and preparation of the coffee drink

  This feature should include the following:

  - A user interface to effect the sale of a coffee drink according to the parameters discussed in this document.
  - A step-by-step description of the preparation process.
  - Show the breakdown of quantities used (i.e., the number of grams of coffee used)
  - Show the total cost/price of the drink.

- View truck information

  This feature allows the user to query various information about a specific truck. These include:

  - Coffee Truck Type and Location.
  - The contents of the storage bins, the type of items, and the quantity.
  - The menu (available products) with prices. And for Special trucks, add-ons and extra shots are included.
  - A list of sales transactions– indicating the type of drink, the amount of ingredients used, and the total cost.

[2] For simplicity, we will limit the location to as a **String** value

- Restocking of Storage Bins and Maintenance

  The restocking feature lets the user replenish the items in the storage bin. In addition, the user can also:

  - Replace the contents of a storage bin with a different kind of item.
  - Empty a storage bin.

  The maintenance feature lets the user update specific truck information. This includes:

  - The truck location
  - Product price list

1.3. Dashboard Feature.

The dashboard feature gives the user a bird's eye view of the status of the Coffee Truck business. At a minimum, the user should be able to see the following information at a glance:

- The number of coffee trucks deployed. Broken down into truck types
- The aggregate amount of inventory (ingredients and cups) stored in the trucks. Broken down by the kind of inventory.
- The combined transaction summary of all trucks. Indicating the type of sale and the aggregate sales total.

1.4. **Exit**

Terminate the application properly. No data will be retained upon exit.

2. Regular Coffee Truck

2.1. Storage Bins

A coffee truck features built-in **Storage Bins** for coffee drink ingredients, including coffee beans, milk, and water.

- A standard truck has eight (8) storage bins to store ingredients and coffee cups.
- Each bin is numbered accordingly for proper identification.
- You can store any item or ingredient in a storage bin, but it must be a single type - you can only store one (1) type of item at a time.
- The table below lists the bin capacity based on the item.

Item	Storage Bin Capacity
Small Cup	80 pcs
Medium Cup	64 pcs
Large Cup	40 pcs

Coffee Beans	1008 grams
Milk	640 fluid ounces (fl. oz.)
Water	640 fluid ounces (fl. oz.)

2.2. The Coffee Drinks

Detailed below are the parameters for preparing our coffee drinks.

- <u>All coffee drinks are available in three (3) sizes</u>
  - Small cup - 8 fluid ounces (fl. oz.[3])
  - Medium cup - 12 fluid ounces (fl. oz.)
  - Large cup - 16 fluid ounces (fl. oz.)

- <u>We craft every coffee drink using just three (3) essential ingredients.</u>
  - Coffee Bean - measured in grams (g).
  - Milk - measured in fluid ounces (fl. oz).
  - Water - measured in fluid ounces (fl. oz).
    Note: For our purposes, we assume that **<u>one (1) fl. oz. = 28.34952 grams</u>**

- <u>Coffee Drink Preparation</u>
  - We use <u>Espresso</u> brew as the base ingredient for all our coffee drinks. Where a single shot of espresso brew is **one (1) fluid ounce,** and the <u>Standard brew</u> ratio is **1:18** (1 part coffee[4] to 18 parts water)

  - Café Americano (proportion to cup size).
    - One (1) part <u>espresso</u>, two (2) parts water

  - Latte (proportion to cup size)
    - One-fifth (⅕) part <u>espresso</u> and four-fifths (⅘) part Milk.

  - Cappuccino
    - One-third (⅓) part <u>espresso</u> and two-thirds (⅔) part Milk.

Consider a customer ordering a Medium Cup of Cappuccino. As the barista prepares the order, the simulation will display a step-by-step process, such as:

```
>>> Preparing Medium Cup…
>>> Brewing Standard espresso - 1.49 grams of coffee…
>>> Adding Milk…
>>> Cappuccino Done!
```

[3] For our purposes, we liberaly use units of measure and may not strictly adhere to established standards.
[4] Note that the base unit of measure for coffee is in grams.

Important Note: When ingredients and materials are used to prepare a drink, they are automatically deducted from their respective storage bins. This ensures that the storage bin quantities always accurately reflect the current inventory levels in real time.

3. Special Coffee Truck

Apart from the features of a regular coffee truck described above, a special coffee truck has the following additional features:

3.1. Extra Storage Bins

Our special coffee truck comes with two (2) extra storage bins. These bins are reserved specifically for storing syrup add-ons to enhance our coffee drinks. Examples of these syrups include hazelnut, chocolate, almond, and sweetener (sucrose). Each storage bin has a maximum capacity of 640 fluid ounces.

When it comes to storing add-on ingredients, you can only store a single type in a storage bin—for instance, you can store only vanilla syrup or only hazelnut syrup in one bin. You determine the kind of add-on ingredient you want to store when you "loadout" the truck during the creation process.

3.2. The Custom Coffee Drink

A Special truck gives customers the freedom to customise their drinks.

We offer three (4) espresso brewing options: the Standard brew, a Strong brew, a Light brew, and a custom brew. The preparation ratio is outlined below.

- Standard brew ratio is **1:18** (See Regular Truck features in Section 2.2)
- Strong brew ratio is **1:15** (1 part coffee to 15 parts water)
- Light brew ratio is **1:20** (1 part coffee to 20 parts water)
- Custom brew ratio **1:?** (The customer determines the water part.)

Lastly, customers can choose to supplement their drink with one or more syrup add-ons available on the truck.

Consider a customer ordering a Medium Cup Cappuccino, a strong brew with hazelnut syrup, and an extra shot of strong-brew espresso. As the barista prepares the order, the simulation will display a step-by-step process, such as:

```
>>> Preparing Medium Cup…
>>> Brewing Strong espresso - 1.89 grams of coffee
>>> Adding Milk…
>>> Adding Hazelnut Syrup
>>> Adding an extra shot of Strong brew espresso -1.89 grams of coffee.
>>> Custom Cappuccino Done!
```

4. Milestones

4.1. MCO1 – due on June 28, 9 PM

4.1.1. UML Class Diagram for regular truck specifications.

4.1.2. Java implementation of the Regular truck specifications.

4.1.3. Text-based implementation (not GUI).

4.2. MCO2 – due on July 28, 7:30 AM

4.2.1. Complete UML Diagram for the regular and special truck specifications.

4.2.2. Complete Java implementation, following the Model-View-Controller (MVC) design pattern.

4.2.3. GUI with Mouse-controlled inputs.

4.3. Deliverables

The deliverables for both MCOs include:

4.3.1. The design and implementation of the solution should…

- Conforms 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

4.3.2. To allow for an easier time to validate the program, the usage of libraries outside of what is available in the Java 21 API is not allowed.

4.3.3. Signed declaration of original work (declaration of sources and citations may also be placed here)

- See Appendix A for an example

4.3.4. Softcopy of the class diagram following UML notations (in pdf or png)

- Kindly ensure that the diagram is easy to read and well-structured

4.3.5. Javadoc-generated documentation for proponent-defined classes with pertinent information.

4.3.6. A zip file containing the source code with proper internal documentation.

4.3.7. 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).
-

4.3.8. **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

5. 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 The student/s should make pertinent back-ups of his/her/their own project. A softcopy of the final unmodified files (for each phase) should be sent to the student/s own email address/es, apart from regular submissions of progress in AnimoSpace and/or Git.on AnimoSpace. No late submissions will be accepted.

6. Grading

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

7. 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. [Questions about the MP specs should be raised in the Discussion page in AnimoSpace.] Copying other people's work and/or working in collaboration with other teams is not allowed and is 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. Comply with the policies on collaboration and AI usage as discussed in the course syllabus.

8. 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 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 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.

9. Bonus Points

No bonus points will be awarded for MCO1. 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 student. Bonus points will be awarded depending on the additional features implemented. These additional features could include new specialised coffee drinks or trucks, saving and loading simulation data into files, etc. Depending on the scale of the new feature, additional points will be awarded to the team. However, ensure that all the minimum requirements are completely and correctly met first; if this is not the case, 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 for the usage of version control. Please consider using version control as early as MCO1 to help collaborate within the group.

10. Resource Citation

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.

Further, this is to emphasize that you DO NOT need to create your own sprites (background pictures, images, etc.) for the application. You can just use what's available on the Internet and just include them into your project, just make sure to cite your sources.

11. 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, demo is live and will include an individual demo problem. 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 MP 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.

12. 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: 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*]

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