CCPROG3 Machine Project Specifications
Prepared by: Tuazon, John Byron

Note: MCO1 and MCO2 are 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.

Project Overview
You have been appointed by the Pokémon League to develop an **Enhanced Pokédex**—a comprehensive application that extends the traditional Pokédex to include global monitoring of Pokémon trainers. This system will also support the management of Pokémon moves and the documentation of items held by each trainer.

The application must be developed using the **Java programming language**, applying the core principles of **object-oriented programming (OOP)**. Its primary goal is to provide a centralized interface for listing trainers, their active Pokémon lineup, stored Pokémon, and inventory items.

Core Functionalities

1. Pokémon Management
This module allows users to manage all Pokémon-related data.

a. Add New Pokémon
Add a new Pokémon to the database with the following:

- **Attributes:**
    - Pokédex Number
    - Name
    - Type 1
    - Type 2 (Optional)
    - Base Level
    - Evolves From (Pokédex Number)
    - Evolves To (Pokédex Number)
    - Evolution Level
    - Base Stats:
        - HP
        - Attack
        - Defense
        - Speed
    - Move Set
    - Held Item
- **Methods:**
    - cry() - when this method is called, the Pokemon’s cry (sound it makes) will play.

b. View All Pokémon
Display all Pokémon in the database with their basic attributes.

c. Search Pokémon
Search for Pokémon by name, type, or other attributes.
Source: https://pokemondb.net/pokedex/all

---

2. Moves Management
Manage the database of Pokémon moves.

a. Add Move
Add a new move with the following attributes:

- **Name**
- **Description**
- **Classification**:
    - HM (Hidden Machine)
    - TM (Technical Machine)
- **Type 1**
- **Type 2** (Optional)

b. View Moves
Display all available moves with key attributes.

c. Search Moves
Search for specific moves using keywords or filters.
Sources:
https://pokemondb.net/move/all
https://bulbapedia.bulbagarden.net/wiki/List_of_moves

---

3. Item Management
Manage items used in the Pokémon world with the following attributes:

- Name
- Category
- Description
- Effects
- Buying Price
- Selling Price

List of Available Items

Item Name	Category	Short Description	Effect	Buying Price	Selling Price

HP Up	Vitamin	A nutritious drink for Pokémon.	+10 HP EVs	₽10,000	₽5,000
Protein	Vitamin	A nutritious drink for Pokémon.	+10 Attack EVs	₽10,000	₽5,000
Iron	Vitamin	A nutritious drink for Pokémon.	+10 Defense EVs	₽10,000	₽5,000
Carbos	Vitamin	A nutritious drink for Pokémon.	+10 Speed EVs	₽10,000	₽5,000
Rare Candy	Leveling Item	A candy that is packed with energy.	Increases level by 1 (stat gain depends on Pokémon's base stats and EVs)	Not sold	₽2,400
Health Feather	Feather	A feather that slightly increases HP.	+1 HP EV	₽300	₽150

Muscle Feather	Feather	A feather that slightly increases Attack.	+1 Attack EV	₽300	₽150
Resist Feather	Feather	A feather that slightly increases Defense.	+1 Defense EV	₽300	₽150
Swift Feather	Feather	A feather that slightly increases Speed.	+1 Speed EV	₽300	₽150
Zinc	Vitamin	A nutritious drink for Pokémon.	+10 Special Defense EVs	₽10,000	₽5,000

Evolution Stones

Item Name	Category	Short Description	Effect	Buying Price	Selling Price
Fire Stone	Evolution Stone	A stone that radiates heat.	Evolves Pokémon like Vulpix, Growlithe, Eevee (into Flareon), etc.	₽3,000–₽5,000	₽1,500

Water Stone	Evolution Stone	A stone with a blue, watery appearance.	Evolves Pokémon like Poliwhirl, Shellder, Eevee (into Vaporeon), etc.	₽3,000–₽5,000	₽1,500
Thunder Stone	Evolution Stone	A stone that sparkles with electricity.	Evolves Pokémon like Pikachu, Eevee (into Jolteon), Eelektrik, etc.	₽3,000–₽5,000	₽1,500
Leaf Stone	Evolution Stone	A stone with a leaf pattern.	Evolves Pokémon like Gloom, Weepinbell, Exeggcute, etc.	₽3,000–₽5,000	₽1,500
Moon Stone	Evolution Stone	A stone that glows faintly in the moonlight.	Evolves Pokémon like Nidorina, Clefairy, Jigglypuff, etc.	Not sold	₽1,500

Sun Stone	Evolution Stone	A stone that glows like the sun.	Evolves Pokémon like Gloom (into Bellossom), Sunkern, Cottonee, etc.	₽3,000–₽5,000	₽1,500
Shiny Stone	Evolution Stone	A stone that sparkles brightly.	Evolves Pokémon like Togetic, Roselia, Minccino, etc.	₽3,000–₽5,000	₽1,500
Dusk Stone	Evolution Stone	A dark stone that is ominous in appearance.	Evolves Pokémon like Murkrow, Misdreavus, Doublade, etc.	₽3,000–₽5,000	₽1,500
Dawn Stone	Evolution Stone	A stone that sparkles like the morning sky.	Evolves male Kirlia into Gallade, female Snorunt into Froslass.	₽3,000–₽5,000	₽1,500

Ice Stone	Evolution Stone	A stone that is cold to the touch.	Evolves Pokémon like Alolan Vulpix, Galarian Darumaka, Eevee (into Glaceon)	₱3,000–₱5,000	₱1,500

a. View Items
Display all items in the database.

b. Search Items
Search for specific items using keywords.

The following items are available in the Pokedex:

Sources:
https://pokemondb.net/item/all
https://bulbapedia.bulbagarden.net/wiki/List_of_items

---

4. Trainer Management
Manage Pokémon trainer profiles and their associated data.

a. Add Trainer
Create a new trainer profile with the following:

- **Attributes:**
  - Trainer ID
  - Name
  - Birthdate
  - Sex
  - Hometown
  - Description
  - Money
- **Methods:**
  - Buy Item
  - Use Item
  - Add Pokémon to Lineup
  - Switch Pokémon from Storage
  - Release Pokémon
  - Teach Moves

b. View Trainers

Display all trainers and their active Pokémon lineup, and in storage.

c. Search Trainers
Search for trainers using keywords.

Restrictions and Requirements

1. **Unique Pokédex Numbers:** No duplicate Pokédex numbers.
2. **Unique Pokémon Names:** No duplicate names. Indicate the region of a Pokémon if needed.
3. **Default Moves:** New Pokémon must start with the moves “Tackle” and “Defend.”
4. **Initial Trainer Funds:** New trainers begin with ₱1,000,000.00 (PkD) for item purchases.
5. **Items List:** You can add more items based on the given link; however, choose items that have direct effect to the stats of the Pokemon when used.
6. **Item Transactions:**
    - Buying an item deducts its full price from the trainer’s money.
    - Selling an item returns 50% of its price.
7. **Item Usage:** Using an item affects Pokémon based on its defined effects.
8. **Held Items:** Certain items (e.g., accessories) can be held by Pokémon. Only one item can be held by a Pokemon at any given time. Changing the held item will discard the current item.
9. **Item Limits:** Trainers may carry up to 10 unique items, with a maximum of 50 items in his/her bag. Exceeding the 50- limit requires discarding existing items.
10. **Pokémon Ownership:** Each trainer maintains a unique instance of a Pokémon with base stats upon assignment.
11. **Lineup Limit:** Trainers may have up to 6 active Pokémon. Adding more requires moving one to storage or releasing it.
12. **Move Set Limit:** A Pokémon may learn up to 4 moves. Teaching a new move requires forgetting one, unless the move is an HM.
13. **Move Compatibility:** A move must share at least one type with the Pokémon learning it.
14. **HM Restriction:** HM moves cannot be forgotten.

Special Items

1. **Rare Candy**
    - Instantly levels up a Pokémon.
    - Each level-up increases base stats by 10%.
    - If enough Rare Candies are used to reach the evolution level, the Pokémon evolves, retaining all stats, moves, and held items.
        1. For a reference in the evolution based on levelling, refer to this list: https://pokemondb.net/evolution/level
2. **Evolution Stones**
    - Instantly evolves a compatible Pokémon.
    - Resulting stats will be the higher value between the current stats and the evolved form’s base stats.
        1. For the sake of simplicity, choose only five Pokemons that can evolve using an evolution stone. Choose from this list: https://pokemondb.net/evolution/stone

Application Menus

Implement the project by providing an interactive menu where the user can select and run the different functionalities. For MCO1, the menu can be activated by entering the option number via the console while a GUI menu is required for MCO2 with mouse-controlled inputs.

Project Milestones

Milestone 1 (MCO1)
Due Date: June 28, 9:00 PM
Deliverables:

1. **UML Class Diagrams**
    - Design class diagrams for the following core components:
        - Pokémon
        - Items
        - Moves
2. **Feature Implementation (Text-Based Interface)**
    - **Pokédex Module**
        - Add new Pokémon
        - Display all Pokémon
        - Search for Pokémon
    - **Moves Module**
        - Add new moves
        - Display all moves
        - Search for moves
    - **Items Module**
        - Display all items
        - Search for items
3. **Interface Requirements**
    - A **text-based menu system** should be implemented.
    - User inputs (e.g., selecting a menu option by number) should trigger corresponding methods.
    - Confirmation messages must be displayed to indicate successful execution of actions.

Milestone 2 (MCO2)
Due Date: July 28, 7:30 AM
Deliverables:

1. **Updated UML Class Diagrams**
    - Refined object-oriented class diagrams reflecting the final implementation.
2. **Graphical User Interface (GUI)**
    - Implement a fully functional GUI with **mouse-controlled inputs** for all features.
3. **Complete Application Integration**
    - Finalize the application using the **Model-View-Controller (MVC)** design pattern.
    - Ensure all modules (Pokémon, Moves, Items, Trainers) are fully integrated and functional within the GUI.

Deliverables
The deliverables for both MCOs include:

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 from or influenced by any use of Generative AI tools or applications
2. To allow for an easier time to validate the program, usage of libraries outside of what is available in the Java 21 API is not allowed.
3. Signed declaration of original work (declaration of sources and citations may also be placed here)
    - See Appendix A for an example
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
5. Javadoc-generated documentation for proponent-defined classes with pertinent information
6. Zip file containing the source code with proper internal documentation
    - The program must be written in Java
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 that are ONLY for screen design (i.e., no computations/processing; just print/println).
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
9. The student/ student/s should make pertinent back-ups of//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.

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. No late submissions will be accepted.

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

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 circumstances 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 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. 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 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.

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 would be awarded depending on the additional implemented features. These additional features could include having evolution animations, visual team displays, or item usage animations. Depending on the scale of the new feature, additional points will be awarded to the team. 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 for 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 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 sprites (background pictures, plant/zombie pictures, etc.) for the game. You can just use what's available on the Internet and just include them in your project, just make sure to cite your sources.

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. The 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 the 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.

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 C. 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**