De La Salle University
College of Computer Studies
Department of Software Technology

Fatal Fantasy: Tactics

CCPROG3 MCO2 Specifications (AY 2024-2025, Term 3)

Release Date: June 20, 2025

Due Date: July 28, 2025 (M) 0730

Prepared by: Kristine Kalaw

1 Project Introduction & Overview

You must create a Java program based on the project specifications detailed below. This project is a culminating program activity that showcases the programming concepts learned in class. Through the project, students should be able to demonstrate the learning outcomes indicated in the syllabus. This project is split into two phases: MCO1 and MCO2.

This project is designed to be completed in pairs (i.e., with a maximum of two members per group). In exceptional cases, the instructor may permit a student to work individually; however, this requires prior approval, as collaboration is an essential part of the learning experience. Under no circumstances will groups of more than two members be allowed.

Whether working individually or as a pair, group composition must remain consistent throughout the entire project. Partner changes between Phase 1 and Phase 2 are not permitted. If a pair separates for any reason, both students must continue the project individually; they may not join another group or form a new pairing.

The project is designed to progressively showcase Object-Oriented Programming Principles. This will be done in two main phases: MCO1 will focus on application design through a UML class diagram and a text-based game simulation via the Command Line Interface (CLI). MCO2 will then culminate in a full GUI-based application with extended mechanisms built upon the foundation of MCO1.

Fatal Fantasy: Tactics is a turn-based fighting game where two players build custom characters by selecting from a variety of character options—resulting in unique characters with distinct playstyles. Players engage in tactical combat by strategically managing their Energy Points (EP) to execute actions and deplete their opponent's Health Points (HP).

Key features of the game include:

- **Character Management** – Players can view, add, edit, and delete their customized characters before entering battle.
- **Character Creation** – Players create unique characters by selecting from various character options.
- **Turn-Based Combat Mechanics** – Battles are driven by strategic decisions, with players managing Energy Points (EP) to execute actions and reduce their opponent's Health Points (HP).
- **Combat Log System** – A detailed combat log tracks all game events, providing transparency and insight into each move made during the match.

2 MCO2: Extended Game Functionality + GUI

2.1 Program Functionalities

You are tasked with extending your output in MCO1 with the following features:

2.1.1 On Character Creation

- **Race Selection**
    - An additional race selection step is added at the beginning of character creation (i.e., race selection, then class selection, then ability selection, and so on).
    - Enhances the character's base properties, with each race offering different boosts such as increased Health Points (HP), Energy Points (EP), or access to an additional ability slot. Refer to Appendix E for the list of races and their corresponding bonuses.

2.1.2 On Character Management

- **Magic Item Inventory**
  - If a character is awarded a magic item, it is automatically added to the character's inventory. There's no limit to how many magic items a character can hold in their inventory (aside from the practical cap set by the maximum integer value).

- **Equipped Magic Item**
  - A character can only equip 1 magic item from their inventory. Equipping/unequipping items happens on the edit character feature.

2.1.3 On the Core Game

- **Using Magic Items**
  - Instead of using class abilities or the base moves Defend or Recharge, characters can opt to use this new move:
    - **Use Magic Item** (no EP cost): The character activates a single-use magic item
  - Note that only single-use magic items can be activated with the **Use Magic Item** move, as passive-effect magic items are automatically applied and do not require manual activation. Passive-effect items provide a continuous bonus as long as they are equipped. Their effects automatically trigger at specific points (e.g., the start of the turn).
    - Any HP or EP restored by a magic item is capped at the character's maximum HP or EP, respectively.

- **Winning Magic Items**
  - Characters are awarded a randomly selected magic item for every third win, regardless of whether the wins are consecutive.

A magic item can either be single-use or have passive effects. Single-use items are automatically removed from the player's inventory after they are used. Refer to Appendix F for the list of magic items, their effects, and their drop rate (i.e., the likelihood of the item being obtained).

2.1.4 Graphical User Interface

Developing a graphical user interface (GUI) is one way to offer an application that is easier for users to understand and navigate. This includes providing window frames that can contain buttons, text input fields, and other onscreen components for each operation of your application. You may utilize any GUI components readily available in Java. All graphical components in Java belong to a common inheritance hierarchy, so they share a common set of methods that can be used to get and set properties such as background color, size, and font.

In case you decide to design a more artistic GUI requiring more complex artwork, remember that it is not required to be created from scratch. You may retrieve or use already existing digital artworks as long as you cite them accordingly.

Kindly take note that some IDEs provide drag-and-drop functionality for creating GUIs. While you're allowed to explore these tools, note that these tools are known for creating spaghetti code – making it harder to understand and manipulate the underlying code. Also, some IDEs (e.g., Netbeans) provide GUI builders that utilize their own GUI APIs. You're asked to refrain from using such arrangements mainly because of the requirement to stick to what is available in the Java API.

Note: Your program must be implemented with GUI elements. Failure to do so will result in the program not being checked. The GUI component in the rubric is with respect to usability – not generally the implementation of GUI.

2.1.5 Model-View-Controller Architecture

The Model-View-Controller or MVC is an architectural design pattern that allows the application to be divided based on defined roles. To promote a more organized and easily maintainable source code, MVC architecture should be followed.

2.2 Bonus Points

Students can earn **up to 10 bonus points (exceeding the maximum score)** by implementing additional features that will enrich the quality of the project. Bonus points will only be given for meaningful and non-trivial additions to the base project (i.e., simply using different font colors is considered trivial, and thus will not incur bonus points). Ideas for bonus features include, but are not limited to:

- Hall of Fame
    - View interesting game statistics such as top players or top characters.
- Player vs. Bot
    - Current project specifications dictate that 2 human players are needed to play the game. A “bot” can be programmed to serve as the opponent for the human player.
- Generate Character
    - On character creation, provide a facility for automatically generating a character build, which the user can approve or modify a little bit.
- Status Effects
    - Introduce new class abilities that deal status effects in addition to/instead of damage. For example, “stunned” can mean that the opponent skips a turn, effectively forcing them to do nothing for the round; “poisoned” can mean that the target can keep on incurring damage every round. There can also be positive status effects such as being able to resist damage.
- Leveling & Experience Points
    - For every battle, characters can still gain experience points (winning a battle makes them earn more). After a certain threshold, characters can level up. Alternatively, milestone leveling can be considered. For example, a character must win 5 battles before they level up, and then 10 battles for the next level up, and so on. Leveling up could mean access to more class ability slots.
- More Classes and Races
    - Introduce more classes and races with new interesting abilities and mechanics. A reskin of what’s currently available will not be considered as a bonus feature.
- More Magic Items
    - Introduce magic items with more interesting effects such as those whose passive effects trigger at the end of the turn, those that can trigger healing when the player drops to 0 HP and is then considered as used (i.e., automatic single-use), those that introduce resistances to certain damage types, those that provide boosts to attacks, etc. A reskin of what’s currently available will not be considered as a bonus feature.
- Magic Item Trading
    - Characters may trade magic items with other characters, even if the character belongs to a different player. Ideally, some rule must be in place such as “you can only trade items of the same rarity" so that the trade is fair (e.g., common items can only be traded with common items).

To encourage the usage of version control (such as Git), **up to 4 bonus points may be awarded**. While the usage of version control will not be taught in this course, you are encouraged to organize and store your code via a Git repository, like the services offered by GitHub. Utilizing some form of version control will make it easier for the members of the group to collaborate with each other and the commit history also helps in providing some form of accountability. Awarded points may vary based on how the group was able to leverage the usage of version control (e.g. small and often commits, descriptive commit comments, contributions from all members, branching).

Bonus points will only be given to projects that have met all the minimum requirements. The amount of bonus points is up to the discretion of the instructor. Ensure that additional features do not conflict with the minimum requirements stated in the specifications; otherwise, these may result in some of those requirements not being met. When in doubt, please consult your instructor. **Lastly, bonus points are capped at 10 points.** For example, if a group was awarded 10 points for the implementation of additional features and 4 points for effective usage of version control, only 10 points would be awarded.

2.3 Implementation Requirements

The project must abide by the following implementation guidelines:

- The program should be **fully implemented in the Java programming language**.
- Observe the appropriate use of programming constructs (e.g., conditional statements, loops, data structures, classes).
    - **Do not use brute force.**
    - Codes must be modular (i.e., split into several logical methods/classes). Codes that are not modularized properly will not be accepted, even if the program works properly.
    - Maximize (but not overuse) object-oriented concepts, like inheritance, polymorphism, and method overloading/overriding.
- You can use these in the future when you have much more experience and wisdom in programming. While you are the **in PROG series you are NOT ALLOWED to do the following:**
    - ✘ to declare and use global variables (i.e., variables declared outside any classes)
    - ✘ to use `exit`, `goto` (i.e., to jump from code segments to code segments), `break` (except in `switch` statements), or `continue` statements
    - ✘ to use `return` statements to prematurely terminate a loop, function, or program
    - ✘ to use `return` statements in `void` functions
    - ✘ to call the `main()` function to repeat the process instead of using loops
- You may only use what is available in the Java API
- You may use topics outside the scope of CCPROG3, but this will be **self-study**.
- Adhere to coding conventions and **must include proper documentation** as indicated in Section 3.1.
- Debug and test your program exhaustively
    - The submitted program is expected to compile successfully and 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.
    - As part of the requirements, you are required to create a test plan and execute your program against it. See Section 3.2 for more details.

You're expected to use your output for MCO1 as a basis for your MCO2; however, depending on how you approached, MCO1, you might be forced to overhaul portions of your design and/or implementation. This is intended to get you to think about how design might accommodate new features in an existing system. While there is no MCO3 (i.e. no further extensions), everyone is encouraged to approach MCO2 by looking at how object-oriented concepts could make the system more flexible to change.

3 Documentation

3.1 Internal Code Documentation

You are **required** to include internal documentation in your source 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. It should follow the Javadoc style guide.

3.2 Test Plan

As part of the requirements of this project, you are also required to submit a test plan document. This document shows the tests done to ensure the correctness of the program.

- It should be presented in a table format. Refer to Appendix B for an example.
    - The test plan template can be accessed in this link. Refer to Appendix C for the guide on how to export the template as a PDF.
- There should be **at least three test cases per function** (as indicated in the Test Description).
    - There is no need to test functions that are intended for interface display/design.
- Test descriptions are supposed to be unique and should indicate classes/groups of test cases on what is being tested.

4 Deliverables

4.1 Submission & Demo

- **The deadline for MCO2 is July 28, 2025 (M) 0730 via AnimoSpace.** After the indicated time, the submission page will be locked and thus considered no submission (equivalent to 0 in this phase).
- The schedule for the demo will be announced at a later date.
- During the demo, you are expected to **appear on time**, be able to **answer questions about the implementation of the project**, and/or be able to **revise the program based on a given demo problem**. Note that more than just delivering a working program, the most important outcome of this project is for you to learn and have a good understanding of how your program works. Therefore, failure to meet any of these expectations could result in a grade of 0.
- 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 the project is automatically 0**. However, a running program with complete features may not necessarily get full credit, as implementation (i.e., code) and other submissions (e.g., non-violation of restrictions evident in code, test script, and internal documentation) will still be checked.

4.2 Deliverables Checklist

The following are the deliverables for both MCOs:

- ☐ Upload via AnimoSpace submission a zip file containing the following:
  - ☐ A signed declaration of original work (refer to Appendix A)
  - ☐ Test plan PDF (refer to Appendix B for a sample)
  - ☐ Class diagram (exported as PDF and PNG)
  - ☐ Javadoc-generated documentation for proponent-defined classes with pertinent information
  - ☐ Source code
  - ☐ [MCO1 only] Video demonstration of your program
- ☐ Email this zip file as an attachment to YOUR own DLSU email address on or before the deadline (the video demo may be excluded from the ZIP file due to file size limitations)

Below is a sample of how the zip file content can be organized, wherein the **`docs/`** folder is the Javadoc-generated documentation and **`src/`** is the root folder of the source code.

```
fatal_fantasy/
|-- docs/
|-- src/
    |-- Driver.java
    |-- <any other files and folders related to the source code>
|-- declaration-of-original-work.pdf
|-- class-diagram.pdf
|-- class-diagram.png
|-- test-plan.pdf
```

5 Collaboration and Academic Honesty

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. Comply with the policies on collaboration and AI usage as discussed in the course syllabus.

6 Grading

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

Acknowledgments / Disclosure

The following Generative AI tools were used to assist in the making of this document.

Google Gemini

- Gemini was used for generating the list of class options and abilities (Appendix D). The instructor reviewed and adjusted some of the AI-generated content.
- Gemini was used for generating the list of race options and bonuses (Appendix E). The instructor reviewed and adjusted some of the AI-generated content.
- Gemini was used for generating the list of magic items and effects (Appendix F). The instructor reviewed and adjusted some of the AI-generated content.

ChatGPT

- ChatGPT was used mainly for streamlining and/or sentence construction. The instructor validated the AI-generated output and modified it as needed.
- ChatGPT was used to generate magic item name suggestions (Appendix F). The instructor reviewed the AI-generated content and selected some for use.

Appendix A: Declaration of Original Work Template

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: Test Plan Format Sample

MyClass						
Method	**#**	**Test Description**	**Test Input**	**Expected Output**	**Actual Output**	**P/F**
getAverage	1	arr contains only non-negative numbers	arr = {1,2,3,4,5,0} arrSize = 6	2.5		
	2	arr contains only negative numbers	arr = {-1,-2,-3,-4,-5} arrSize = 5	0.0		
	3	arr contains a mix of positive numbers, negative numbers, and zeros	arr = {-1,0,5,-4,7,1} arrSize = 6	3.25		
	4	arr contains only zeros	arr = {0, 0, 0} arrSize = 3	0.0		
anotherMethod	1	...	...	...		
	2	...	...	...		
...	...	...	...	...		

Given the sample method above, the following are 4 distinct classes of tests:

1. Testing with arr containing only non-negative numbers
    - Non-negative numbers (positive numbers and zeros) are the only values that can be part of the average computation
2. Testing with arr containing only negative numbers
3. Testing with arr containing a mix of positive numbers, negative numbers, and zeros.
4. Testing with arr containing only zeros
    - Zero is a non-negative number

The following test descriptions are **incorrectly** formed:

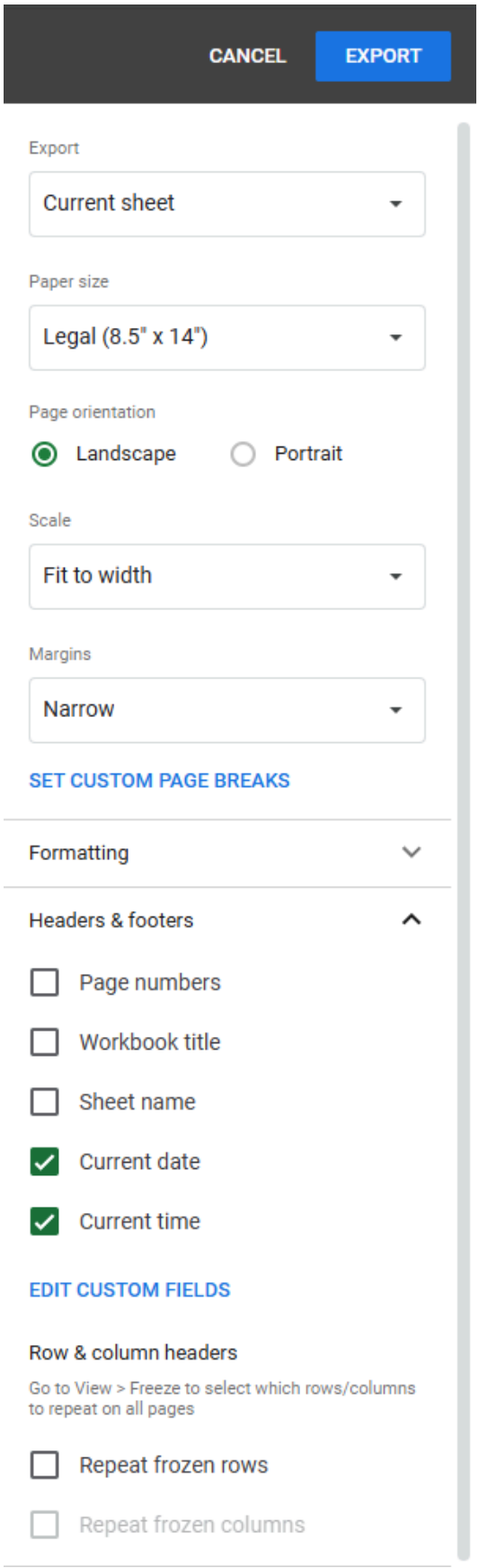
- **Too specific:** Testing with arr = {1, 2, 3, 4, 5}
- **Too general:** Testing if the function can correctly compute the average of the non-negative numbers
- **Not necessary (because already defined in the pre-condition):** Testing with a negative arrSize

The test plan template can be accessed in this link. Refer to Appendix C for the guide on how to export the template as a PDF.

Appendix C: Export Test Plan Guide

1. In the **TestPlan** sheet:

   File > Download > PDF (.pdf)
2. Adjust the settings to:
   - Export: Current Sheet
   - Paper size: Legal (8.5 x 14)
   - Page orientation: Landscape
   - Scale: Fit to width
   - Margins: Narrow
   - Formatting dropdown
     - Leave as is
   - Headers& footers dropdown
     - Check: Current date
     - Check: Current time
     - Uncheck: Repeat frozen rows
3. Click Export.



Appendix D: Class Options and Abilities

Mage

Mages command arcane energies, specializing in powerful spells and mystical manipulation

Ability	Description and Effect	EP Cost
Arcane Bolt	Launch a basic magical projectile that deals 20 arcane damage to the target.	5
Arcane Blast	Unleash a burst of fiery energy, dealing 65 arcane damage to the target.	30
Mana Channel	Draw upon ambient magical energy to restore your own. Restores 15 EP.	0
Lesser Heal	Weave a minor healing spell to mend your wounds. Restores 40 HP.	15
Arcane Shield	Conjure a protective barrier of mystical energy around yourself. You do not take any damage for the round.	12

Rogue

Rogues are agile and tricky, relying on precision and debilitating opponents

Ability	Description and Effect	EP Cost
Shiv	A quick, precise stab that deals 20 physical damage.	5
Backstab	Strike a vital point and deal 35 points of physical damage.	15
Focus	Take a moment to concentrate, restoring your mental energy. Restores 10 EP.	0
Smoke Bomb	Throw a smoke bomb, making you harder to hit. You have a 50% chance of evading any incoming attacks in the current round.	15
Sneak Attack	You rely on your agility to evade your opponent, taking no damage from any of their attacks, while you deal 45 physical damage to them.	25

Warrior

Warriors are tough, focusing on direct combat and robust defense

Ability	Description and Effect	EP Cost
Cleave	A sweeping strike that deals 20 physical damage.	5
Shield Bash	Slam your shield into the opponent, dealing 35 physical damage.	15
Ironclad Defense	Brace yourself, effectively taking no damage for the current round.	15
Bloodlust	Tap into your inner fury, restoring a small amount of health. Restores 30 HP.	12
Rallying Cry	Let out a powerful shout, inspiring yourself and recovering 12 EP	0

Appendix E: Race Options and Bonuses

Race	Description	Bonus
Human	Adaptable and resilient, humans possess a balanced set of attributes.	+15 to max HP +5 to max EP
Dwarf	Stocky and tough, dwarves are known for their incredible endurance and steadfastness.	+30 to max HP
Elf	Graceful and naturally attuned to arcane energies, elves excel in precise actions and magical prowess.	+15 to max EP
Gnome	Clever and resourceful, gnomes have a knack for finding hidden opportunities or leveraging unusual tricks.	+1 additional ability slot (the choice can be from ANY class)

Appendix F: Magic Items and Effects

Common Items (60% drop chance)			
Item Name	**Activation Type**	**Description**	**Effect**
Potion of Minor Healing	Single-Use	A basic potion that restores a small amount of health.	Heals the user for 40 HP
Scroll of Minor Energy	Single-Use	A simple scroll inscribed with runes that replenish a small amount of energy.	Restores 20 EP to the user
Defender's Aegis	Single-Use	A small, temporary barrier that absorbs damage.	Negates all incoming damage

Uncommon Items (35% drop chance)			
Item Name	**Activation Type**	**Description**	**Effect**
Amulet of Vitality	Passive	An enchanted amulet that subtly strengthens your life force.	Increases max HP by 20
Ring of Focus	Passive	A plain ring that helps you concentrate, boosting your energy regeneration.	Additional +2 EP at the start of each turn

Rare Items (5% drop chance)			
Item Name	**Activation Type**	**Description**	**Effect**
Orb of Resilience	Passive	A small, smooth orb that provides a constant minor protective aura.	Heal +5 HP at the start of each turn
Ancient Tome of Power	Passive	A worn book filled with forgotten wisdom that grants a small, continuous surge of power.	Additional +5 EP at the start of each turn