CCPROG3
Machine Project
AY 2024 – 2025 T3

Release date:	May 30, 2025
Phase 1 due date:	June 28, 2025 900pm
Phase 2 due date:	July 28, 2025 730am

College of Computer Studies

Warrior

Motivation

Games have always been a way for people to distract themselves from the worries of everyday life and a way to pass the time. They play and get a sense of enjoyment as they play. They also see a game as a challenge to beat. If the game is too easy, they drop it after a very short while. Game developers include a difficulty setting that the players can set if they wish for more of a challenge in the same game or if they just want to feel all powerful and beat things without much effort. It adds to the fun and gives the player a sense of accomplishment when they beat the game with a harder setting.

Game Components

The main objective of the game is to play the role of a warrior with the goal of defeating a selected foe. The player will create a character and choose the armor and weapon for her/his character. The player will also select different environments for the battle that can make the match easier or harder depending on the choice. This will give the player an option should she/he want to take on a challenge.

In general, your game should cover the following features:

1. Create a character and select armor and weapon to determine stats.
2. At least 3 kinds of armor to choose from.
3. At least 3 kinds of weapons to choose from.
4. At least 3 types of environments for the battle to take place.
5. At least 3 kinds of opponents that the player will select to battle with.

The game generally contains the following components: (Kindly take note that you are not limited to these components and are free to add more components as you see fit; however, the components below should all be present in your project):

- **Warrior (Player Character)** – The character the player controls. Will have the following stats without armor and weapon.

Hit Points (HP)	Attack (Atk)	Defense (Def)	Speed (Spd)
100	1	1	50

Hit Points – The players life. When it reaches zero(0), the character is knocked out (KO'd) and the game is over.

Attack – The amount of damage the player will deal to the opponent.

Defense – Protection from the attack of the opponent.
Eg. If the monster's attack is 10 and your defense is 1, you only take 9 damage.

Speed – Determines if you will be the first to strike. If your speed is higher than that of the opponent's, you strike first.

- **Armor** – Protection against the attack of the opponent. There will be 3 kinds of armor to choose from which will affect *Defense* and *Speed*.

Armor Type	Defense	Speed Penalty
Light	+20	-5
Medium	+30	-15
Heavy	+40	-25

- **Weapon** – The weapon you will use for attacking your opponent. There will be 3 kinds of weapons to choose from which will affect *Attack* and *Speed*.

Weapon Type	Attack	Speed Penalty	Weapon Ability (Phase 2)
Dagger	+20	0	When defending, every other defend will become a 100% evade
Sword	+30	-10	When attacking, gain an additional +10 attack
Battle Axe	+40	-20	When charging, gain 5 speed and 5 attack in the next turn

- **Opponents** – The player will choose an opponent to face in combat.

Opponent	Hit Points (HP)	Attack (Atk)	Defense (Def)	Speed (Spd)	Faux-AI (Phase 2)
Thief	150	20	20	40	Attack Continuously
Viking	250	30	30	30	Attack, Defend, Attack
Minotaur	350	40	40	20	Attack, Charge, Attack

Faux-AI – The Opponent's attack pattern.

REQUIRED METHOD(For Phase 2)!

- **Think()** – This method will determine what the opponent will do next.

- **Environments** – The place where the battle will be held. Depending on which environment you choose, either your player will benefit or the opponent will benefit. This will add a layer of challenge for the player if he/she wishes for something harder.

Environment	Player Penalty	Opponent Penalty
Arena	No penalty	No Penalty
Swamp	Player takes 1 damage every turn.	Opponent gains 1 atk every turn.
Colosseum	Player gains 1 atk every turn.	Opponent loses 1 def each turn.

Playing the game

You start by setting up your character. You select the armor and weapons then select an opponent. The battle will be turn-based and during the start of each turn, the player will select a command.

The player will have the following commands:

Attack – The player will strike the opponent.

Defend – The player will block the attack of the opponent. You will take only half damage of the opponent's attack. This command ignores the speed attribute and will always activate first. This applies for both the player and opponent.

Example: If your speed is lower than your opponents, this means if he/she attacks you, their action will happen first. But if you choose Defend for that turn, your action will happen first allowing you to block the attack.

Charge – You skip a turn/don't attack but the next turn your damage is tripled. You can only charge once every other turn since the effects of charge are active on the turn after using it. In short, you cannot stack charging.

The *Speed* attribute will determine whose action will execute first. The goal is to beat the opponent before your hit points reach zero(0).

MP REQUIREMENTS:

For Phase 1, you are required to implement the following functionalities:

- Character creation
- Armor selection
- Weapon selection
- Opponent selection
- Environment selection
- Turn-based battle system

For Phase 2, additional features are required to be implemented as specified below:

- **2 Parent Classes each with 3 Child Classes.**
- Faux-AI Implementation (see Opponents table above)
- Weapon Ability Implementation (see Weapon table above)
- GUI-based, full implementation of all game elements. Game should properly start, iterate and end.
- Complete implementation of the game following Model-View-Controller(MVC) framework.
- Design and Implementation should exhibit proper object-oriented concepts, like inheritance, polymorphism and method overloading/overriding.
- Errors should be checked and properly handled

Requirements for both phases:

1. The following deliverables for each phase are to be uploaded on AnimoSpace:
    - Declaration of original work (Declaration of sources and citations may also be placed here)
    - **FOR PHASE 1 ONLY:** Video recording of you demonstrating the program which shows all the functionalities expected in Phase 1.
    - Class diagram following UML notations. The class diagram should exhibit the appropriate object-oriented programming principles.
    - Pdf of the test script following the format indicated below.
    - javadoc-generated documentation for proponent-defined classes with pertinent information;
    - source code with proper internal documentation;
    - class diagram; and
    - test script document

2. 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. 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 met first; if this is not the case then no additional points will be credited despite the additional features.

   We would just like to emphasize that you DO NOT need to create your own sprites (background pictures, item/product pictures, etc.) for the project. You can just use what's available on the Internet. If you wish to do so, we would also like to remind you to please cite where you got your sprites.
3. For the minimum requirements of this MP, all the requirements written in this document should be present and working.

4. 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 command prompt command javadoc to create this documentation, but it must be PROPERLY constructed.

5. 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.

6. You are required to create and use methods and classes whenever possible. Make sure to use Object-Based and Object Oriented Programing concepts properly. No brute force solution.

7. This project must be done in groups of 2. Going solo will only be allowed if permitted.
8. You cannot discuss or ask about design or implementation with other groups, with the exception of the teacher and his/her 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. (Also, what is a measly passing grade compared to a life-long burden on your conscience?)

9. Submission of the project is on the indicated submission deadlines. Late submission will not be accepted and will therefore result to 0 for that phase.

10. Incomplete submissions will incur deductions.

11. During the demo, all members of the group should be present. The group should use the machine in the lab for the demo and should know how to generate the bytecode file and to run the said file in the command prompt. Apart from question-answer, it is possible that a demo problem be given to the group as part of the demo.

12. 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.

13. 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.

14. All sources should have proper citations. Citations should be written using the APA format. Examples of APAformatted citations can be seen in the References section of the syllabus.

15. As part of the requirement, the proponent/s should make pertinent back-ups of his/her/their own project.

16. In the event that only 1 person in the group is doing all the work, notify me as early as possible. If it cannot be resolved then the group will be disbanded and members will have to work individually. No notification would mean that you have accepted your groupmate's behavior and have therefore surrendered any motion to complain about said groupmate.

SUBMISSION CHECKLIST FOR EACH PHASE

- Files:
  - The Java source files
  - The output of Javadoc (compressed into a zip file)
  - A copy of the UML Class Diagram
  - Test script
  - Declaration of original work (Declaration of sources and citations may also be placed here)
  - UML Class Diagram
  - Test Script for each method in each class that you created, in the following table format (with sample entries).

    Note: 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).

Sample is shown below.

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

- Upload your project to AnimoSpace on or before the deadline.
- Email the Java source files as attachments to YOUR own email for backup purposes.

*For clarifications or questions about this MP specs, please direct your inquiries to me via Google Chat.