CCPROG3 Machine Project Specifications
3rd Term, AY 2024-2025

Note: *MCO1 and MCO2 is 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.*

I. Introduction

Create a Java desktop game application derived from the original game called "Plants vs. Zombies". In this game, the objective of the player is to use his limited supply of plants or greens and seeds to prevent a mob of zombies from invading his home.

The environment is a board with 5 or 6 lanes, where each lane has 9 to 10 tiles. Each game lasts for 3 minutes. Once the game starts, sun drops at a constant rate. The player can start collecting sun. Once there is enough sun collected, the player can start putting available plants on the tiles. After a few seconds, zombies start to move from one end towards the player's home that is at the other end of the lane. The plants on the lane can either attack or defend against the zombies on that lane. At the last few seconds of the game, a wave of zombies storm to the player's home. At level 1, the number of zombies during the wave is 5. This increases by 2 on each level afterwards.

Game ends when either one of the following conditions is satisfied:

- When one zombie reaches the player's home, Zombies win; or
- When game time ends, Plants win.

II. Zombies

There are a number of types of zombies that differ in attributes: speed, damage, and health. The values for these attributes depends on the type of zombie generated.

Zombies can wear armor (e.g. flag, cone, pole vault, bucket) that may increase their tolerance, or change their movement. Some armors slow down the zombies, and some decreases damage: flag slows down the zombie by 1; the cone decreases speed by 2, decreases damage by 2; and bucket decreases speed by 3, decreases damage by 5.

Some zombies walk, some skip.

	Zombie	Normal zombie Speed = 4, Damage = 10, Health = 70
	Flag Zombie	Moves slightly faster and signals a huge wave incoming

	Conehead Zombie	Uses a traffic cone to protect itself
	Pole Vaulting Zombie	Single jump, jumps over the first plant it encounters with a pole
	Buckethead Zombie	Has a bucket that is extremely resistant to damage

III. Plants

There are a number of types of plants that differ in attributes: sun cost, regenerate rate, damage, health, range, direct damage, and speed.

Sunflower		Gives you additional sun
Peashooter		Shoots peas at attacking zombies
Cherry Bomb		Blows up all zombies in an area
Wall-nut		Blocks all zombies and protects your other plants
Shovel		Lets you dig up a plant to make room for other plant
Potato mine		Explodes on contact but takes time to arm itself
Snow Pea		Shoots frozen peas that damage and slow the enemy

IV. Attributes

Zombie Attributes

Speed	How fast the zombie moves
Damage	How much damage it deals to the plant it is attacking
Health	How much damage it can sustain

Plant Attributes

Cost	How much sun needed to use this
Regenerate Rate	How much time it needs to regenerate
Damage	How much damage it deals the zombie
Health	How much damage it can sustain
Range	How far its attack reaches
Direct damage	How much damage it costs when zombie is at a closer range
Speed	How fast the next attack will be

V. Zombie and Plant Generation

Zombies are generated based on the following time intervals:

Period	Interval
30 to 80	Every 10 seconds
81 to 140	Every 5 seconds
141 to 170	Every 3 seconds
171 to 180	Wave of zombies

There is an equal probability for the Zombie to appear in each lane.

Plants are generated depending on their regenerate rates. Plants can be used as long as enough sun is collected by the player.

VI. Minimum Requirements

a. Graphical User Interface

You are to provide a windows application with a mouse interface.

b. Game Phase

Plants and Zombies

During the game, every plant that is available should be clearly indicated.

You are required to implement the following:

- Zombie
- Flag Zombie
- Conehead Zombie
- Sunflower
- Peashooter
- Cherry bomb

For values of the attributes that are not defined explicitly in this specs, you can refer to the references and use a scaled version of these values. For reference on other zombies and plants, you may refer to:
http://plantsvszombies.wikia.com/wiki/Gallery_of_zombies
and http://plantsvszombies.wikia.com/wiki/Plants_(PvZ)

Levels

You are to implement at least three different levels, i.e. three different maps. You may add the number and types of zombies, and the available plants. But all the listed zombies and plants should be available in the final project.

VII. Milestones (see section VIII for Deliverables)

a. MCO1 – due June 28, 9PM

1. UML Class Diagrams for Zombie, Sunflower, Pea Shooter
2. Implementation of the class Zombie, Sunflower, Pea Shooter
3. Driver to test the basic game (i.e., 1 map). That is, it should cover generation of zombie objects, sunflower objects, peashooter objects at the proper intervals and at the proper tile. No GUI display is expected. However, displays must be made in the Console). For example: "Zombie appeared in Row 1, Column 8 with the following initialized values in the attributes…". Allow user inputs to be given via keyboard (where pea shooters are going to be placed, allow collection of sun). For example, "Have enough sun, do you want to add Sunflower or Peashooter to the board?"; "Enter row # then column # where pea shooter will be placed:" Lastly, show per [simulated] time interval, what is happening. For example: "Time: 00:26…. Sun Generated, Zombie previously in Row 1 Column 8 now moved to Row 1 Column 7".

b. MCO2 – due July 28, 7:30AM

1. UML Class Diagrams (Object-Oriented)
2. GUI with Mouse-Controlled Inputs.
3. Complete implementation of the game, following Model-View-Control (MVC).

VIII. 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 or influenced from 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 which 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/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.

IX. 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.

X. Grading

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

XI. 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 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.

XII. 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.

XIII. 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 new specialized zombies or plants, saving and loading current status of player in files, etc. 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 the usage of version control. Please consider using version control as early as MCO1 to help with collaborating within the group.

XIV. 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 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 into your project, just make sure to cite your sources.

XV. 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.

XVI. 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**