PLANTS VS. ZOMBIES

FINAL PROJECT OOP

The "Plant vs Zombie" project is a simplified version of the popular tower defense game where players place plants to defend against waves of zombies. This project leverages Object-Oriented Programming (OOP) principles to create a modular, maintainable, and extensible codebase. The game is designed with several key classes representing different elements of the game, such as plants, zombies, projectiles, and the game board.

GAME DEVELOPERS

Do Minh Duy

Luong An Khang

Phan Minh An

Phung Huy Quang

ADVISOR

Prof. Tran Thanh Tung

Contents

PART 1:

INTRODUCTION

1. About Plant vs Zombies:

Plants vs. Zombies (abbreviated as PvZ or PvZ1) is a tower defense video game developed and originally published by PopCap Games and it is the first installment in the Plants vs. Zombies series. The game involves homeowner who uses a variety of different plants to prevent waves of zombies from entering his house and "eating his brain".

2. About the game project:

In our game, we implemented a simplified version of the "Plants vs. Zombie" game by applying OOP principles such as encapsulation, inheritance, and polymorphism to ensure the code is modular and maintainable. Besides, we also create an engaging and interactive game experience.

We added some features in the game:

- Skip the starting levels for faster accessibility.
- Remove the lawn mowers (harder gameplay).
- Add new zombie: “Brickhead”.

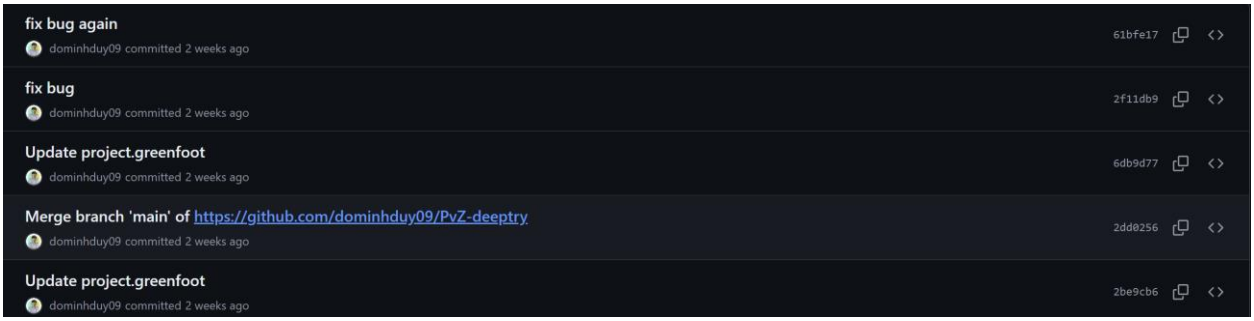
3. Preferences:

- Images: https://www.google.com/imghp
- Art and music by:
    - Popcap Games: https://www.ea.com/ea-studios/popcap/plants-vs-zombies
    - Nintendo: https://www.nintendo.com
- Game codes:
    - https://github.com/arminkz/PlantsVsZombies
    - https://github.com/TheExploration/Plants-Vs-Zombies
- Tutorials:
    - GitHub commit: https://www.youtube.com/watch?v=lYiE5lBS13E
    - Greenfoot: https://youtube.com/playlist?list=PLmwzeqwf733_hpzeDcqW6VUSZciqwkh2S
    - PvZ Clone: https://www.youtube.com/watch?v=pUwLbVlrGtY https://www.youtube.com/watch?v=dsreCeC-QMs
    - PvZ java:https://www.youtube.com/watch?v=ch5Mgp_KjZg

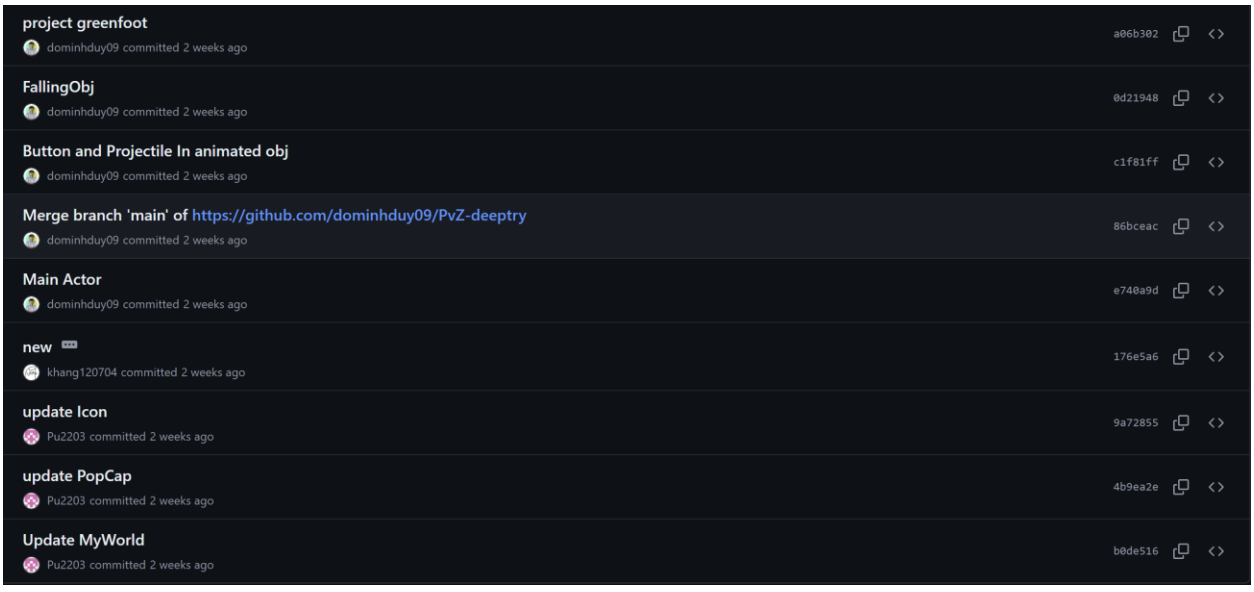
4. Developers:

Name + GitHub	ID	Contribution
Do Minh Duy *dominhduy09*	ITITSB22029	- Write report. - Draw UMLs. - Zombie class. - Create ReadmeTXT. - MainActor.
Phung Huy Quang *Pu2203*	ITITWE22145	- Write report. - Draw UMLs. - Create World class.
Phan Minh An *Ze1Z1*	ITITSB22028	- Write report. - Create Zombie class. - Create IdleZombie class. - Create Plant class. - Make Slides.
Luong An Khang *khang120704*	ITITWE22140	- Write report. - Create SeedPacket. - Create TransparentObject.

- Commit history:


fix bug again
dominhduy09 committed 2 weeks ago
61bfe17
fix bug
dominhduy09 committed 2 weeks ago
2f11db9
Update project.greenfoot
dominhduy09 committed 2 weeks ago
6db9d77
Merge branch 'main' of https://github.com/dominhduy09/PvZ-deeptry
dominhduy09 committed 2 weeks ago
2dd0256
Update project.greenfoot
dominhduy09 committed 2 weeks ago
2be9cb6


update PopCap, MyWorld, Icon
Pu2203 committed 2 weeks ago
Verified
7fcce83
Merge branch 'main' of https://github.com/dominhduy09/PvZ-deeptry
Pu2203 committed 2 weeks ago
3c35a9c
Merge branch 'main' of https://github.com/dominhduy09/PvZ-deeptry
khang120704 committed 2 weeks ago
8332d85
AnKhang commit
khang120704 committed 2 weeks ago
e5ea760
Merge branch 'main' of https://github.com/dominhduy09/PvZ-deeptry
dominhduy09 committed 2 weeks ago
699705b
Merge branch 'main' of https://github.com/dominhduy09/PvZ-deeptry
Ze1Z1 committed 2 weeks ago
7b6a0cf
MInh An commit Zombie Idle Zombie Plant
Ze1Z1 committed 2 weeks ago
00fd039


project greenfoot
dominhduy09 committed 2 weeks ago
a06b302
FallingObj
dominhduy09 committed 2 weeks ago
0d21948
Button and Projectile In animated obj
dominhduy09 committed 2 weeks ago
c1f81ff
Merge branch 'main' of https://github.com/dominhduy09/PvZ-deeptry
dominhduy09 committed 2 weeks ago
86bceac
Main Actor
dominhduy09 committed 2 weeks ago
e740a9d
new
khang120704 committed 2 weeks ago
176e5a6
update Icon
Pu2203 committed 2 weeks ago
9a72855
update PopCap
Pu2203 committed 2 weeks ago
4b9ea2e
Update MyWorld
Pu2203 committed 2 weeks ago
b0de516

PART 2:

SOFTWARE REQUIREMENTS

1. Tools and platforms:

We used all the tools and platforms below:

a. Canva:
   - UML making.
   - Creating Backgrounds.

b. Visual Studio Code:
   - Creating game elements.

c. GitHub Desktop:
   - Commiting work.

d. Greenfoot:
   - Execute the game.

e. Photoshop:
   - Editing images.

2. About Greenfoot:

Greenfoot is an educational software designed to teach programming concepts through the creation of interactive graphical applications, such as simulations and games. Developed by the King's College London, it leverages the Java programming language in an engaging and user-friendly environment. Greenfoot provides a visual and interactive platform where users can write code to control actors in a 2D world, making it an ideal tool for beginners and educators to introduce Object-oriented Programming. Its intuitive interface and extensive documentation support a hands-on learning approach, allowing users to see the immediate effects of their code, thereby reinforcing programming concepts and logic in a practical, enjoyable manner.

PART 3:

DESIGN & IMPLEMENTATION

1. Plant class:

The "Plants vs. Zombies" game centers around players using various types of plants to fend off waves of zombies. We show off the design, implementation, and functionality of the plants, as well as the class attributes, methods, and inheritance structure used to create diverse plant behaviors.

a. Shooting mechanism:

The shooting mechanism in the game is a vital feature, especially for offensive plants like the Peashooter. This report outlines the design, implementation, and functionality of the shooting mechanism, detailing how plants detect zombies, shoot projectiles, and cause damage.

In our game, we use Only “pea” as the only projectile. And the projectiles are built by:

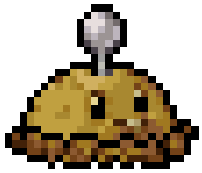
```
for (Zombie i : MyWorld.level.zombieRow.get(yPos)) {
    if (Math.abs(i.getX() - getX()) < 30) {
        if (!foundTarget) {
            hitZombie = i;
            foundTarget = true;
        }
        if (!hit) {


            hitZombie.hit(damage);
            hit = true;

        } else if (hitZombie.getWorld() != null && getX() < hitZombie.getX()) {
            move(speed);
        }
    }
```

b. Specific Plant types:

In PvZ, we includes 4 variaties of plants: Shooting (Peashooters), Producing (Sunflowers), Defending (Potato), and Exploding (PotatoMine).

- Peashooters: hp = 60, dmg = 10.
- Sunflowers: hp = 60, time to produce a sun = 20s (much longer than usual)
- Wallnut: hp = 730 (increased).
- PotatoMine: hp (ungrown) = 60, dmg = die.



2. Zombie class:

The Zombie class was designed using Object-Oriented Programming (OOP) principles to ensure modularity and reusability. The main class, Zombie, serves as a base class from which specific zombie types inherit.

The hp of zombie is count by:

```
public void takeDmg(int dmg) {
    hp -= dmg;
    if (hp <= 0) {
        for (ArrayList<Zombie> i : MyWorld.level.zombieRow) {
            if (i.contains(this)) {
                i.remove(this);
                break;
            }
        }
        getWorld().removeObject(this);
        return;
    }
}
```

We changed the hp values of zombies to be tankier.

a. Eating mechanism:

The eating mechanism is a critical feature that determines how zombies interact with and damage plants. This include the design, implementation, and functionality of the eating mechanism, detailing how zombies move towards and destroy plants.

```
public boolean isEating() {
    var row = MyWorld.board.Board[getYPos()];
    for (int i = 0; i < MyWorld.board.Board[0].length; i++) {
        if (row[i] != null) {

            if (Math.abs(row[i].getX() - getX()+5) < 35) {
                if (row[i] instanceof PotatoMine) {
                    if (((PotatoMine)row[i]).armed == true) {
                        eating = false;
                        return false;
                    }
                }

                eating = true;
                target = row[i];
                return eating;
            }

        }
    }

    eating = false;
    return eating;

}
```

We set the grown potato mine to be inedible.

b. Specific Zombie Types:

To create diverse zombie behaviors, specific zombie types were derived from the base Zombie class. Each type has unique attributes and methods.

For the zombies with an object on their head, we set the health for them and their specific animations, such as

```java
public class Cone extends FallingObject
{
    /**
     * Act - do whatever the Cone wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public Cone() {
        super(-2, 0.2, 0.9, Random.Int(1, 5), 620L);

    }
}
```

This technique also be used with arms and heads.

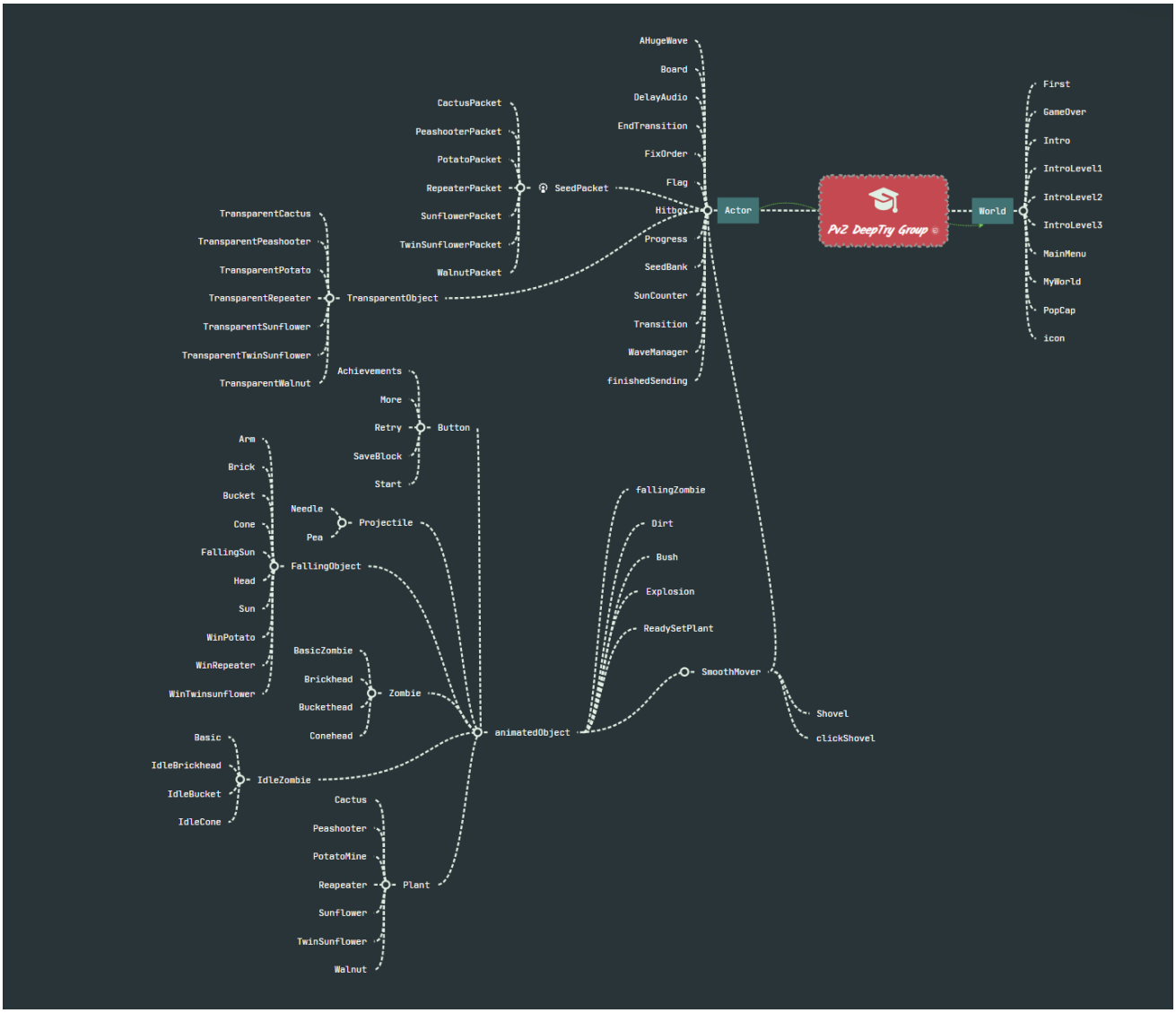
For the cone:

```java
public void update() {
    if (hp > 232) {
        if (!isEating()) {
            animate(coneheadwalk, duration:350, loop:true);
            move(-walkSpeed);
        } else {
            animate(coneheadeat, duration:200, loop:true);
            playEating();
        }
    } else if (hp > 166) {
        if (!isEating()) {
            animate(coneheadwalkd, duration:350, loop:true);
            move(-walkSpeed);
        } else {
            animate(coneheadeatd, duration:200, loop:true);
            playEating();
        }
    } else if (hp > 100) {
        if (!isEating()) {
            animate(coneheadwalkdd, duration:350, loop:true);
            move(-walkSpeed);
        } else {
            animate(coneheadeatdd, duration:200, loop:true);
            playEating();
        }
    } else {
        if (cone) {
            cone = false;
            MyWorld.addObject(new Cone(), getX(), getY()-25);
        }
```

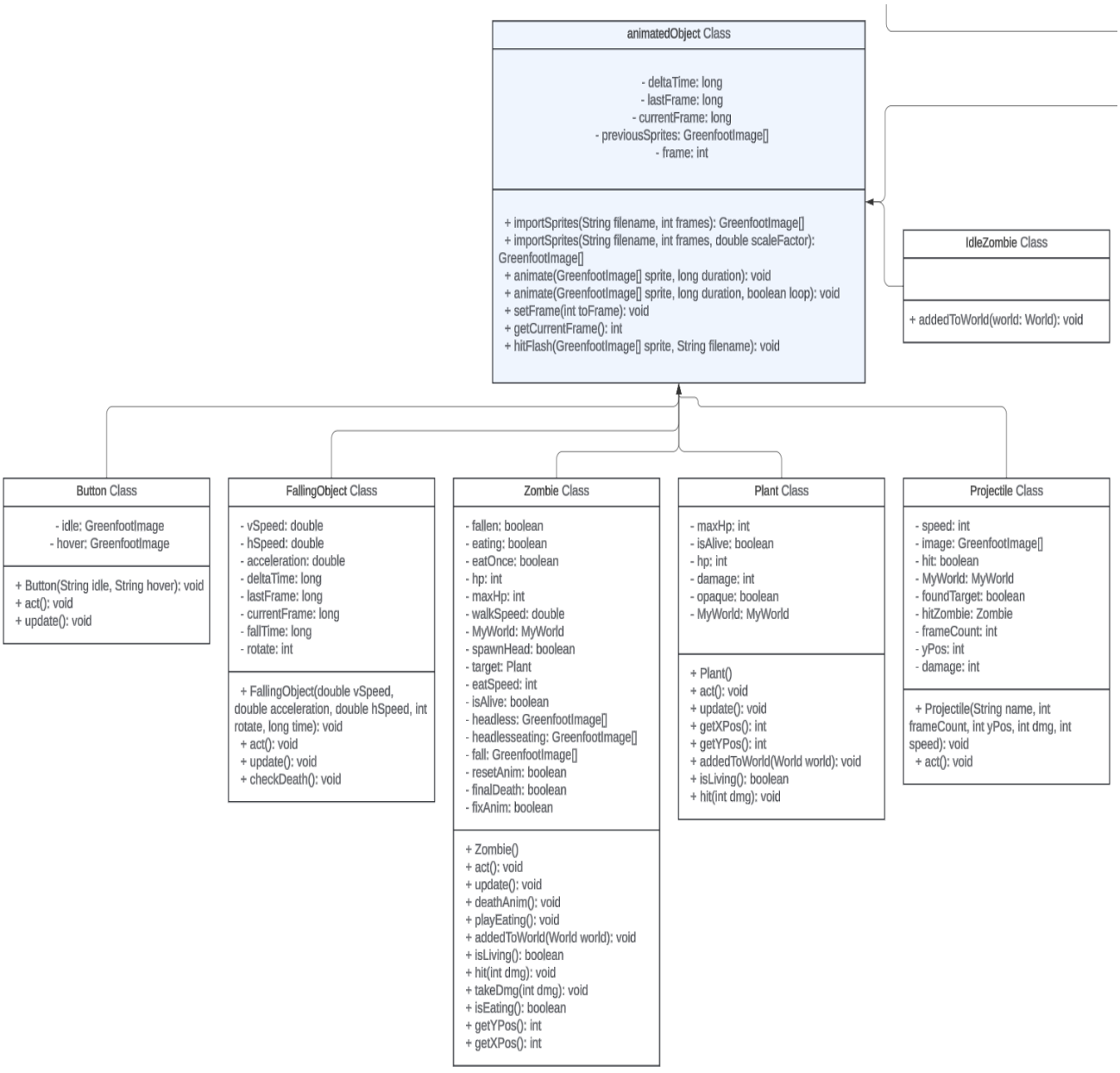
The armored zombies interact with other game components in the same way as standard zombies but with increased health. Their higher durability makes them more challenging to defeat, but with the same technique.

3. UMLs:

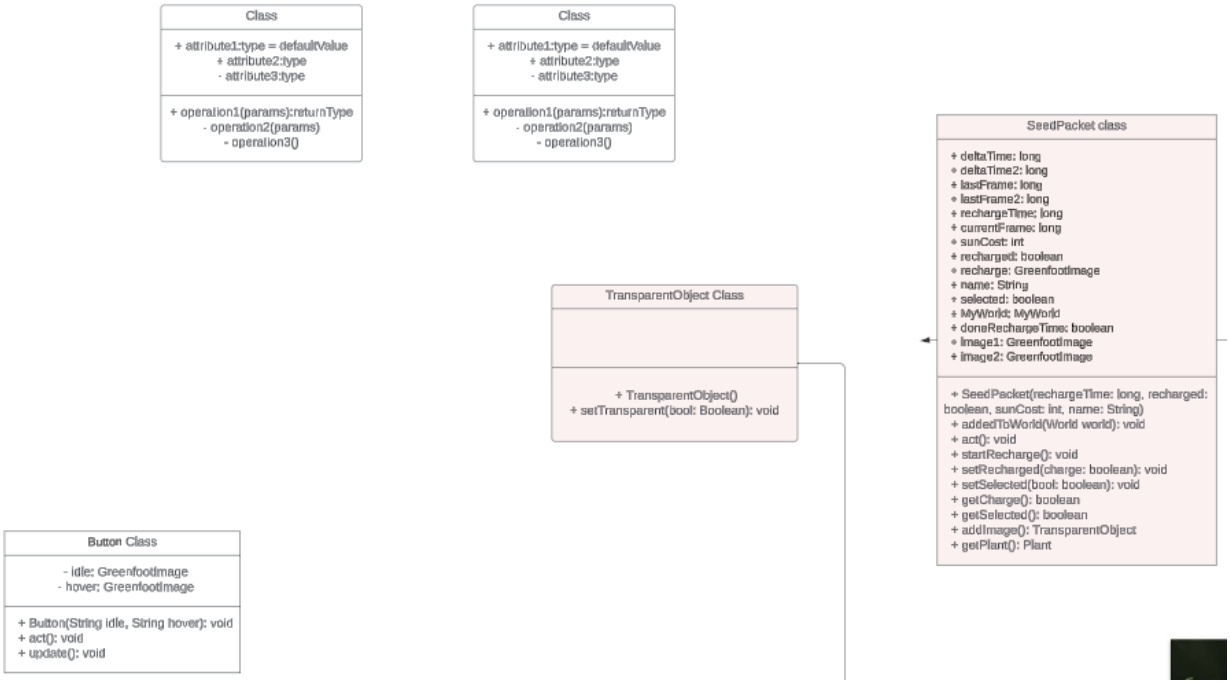
a. UML Class Diagram:



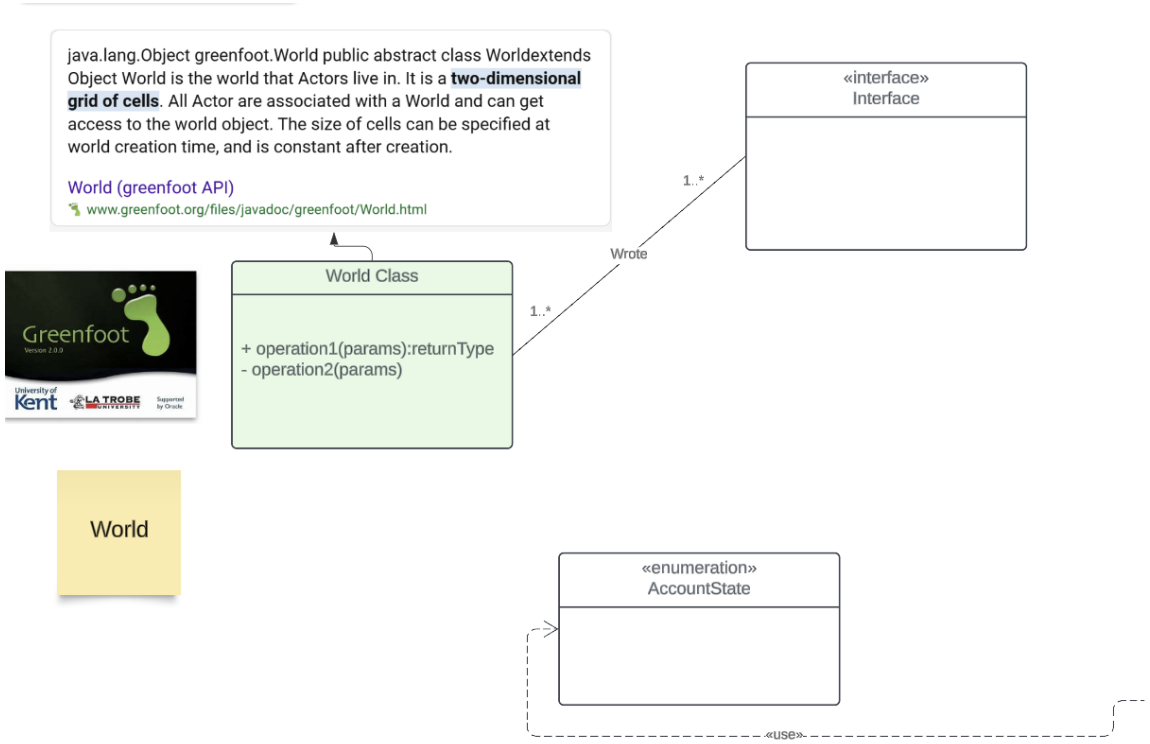
b. UML of animateObject class:



c. UML of SeedPacket class:



d. UML of World class:



PART 4:

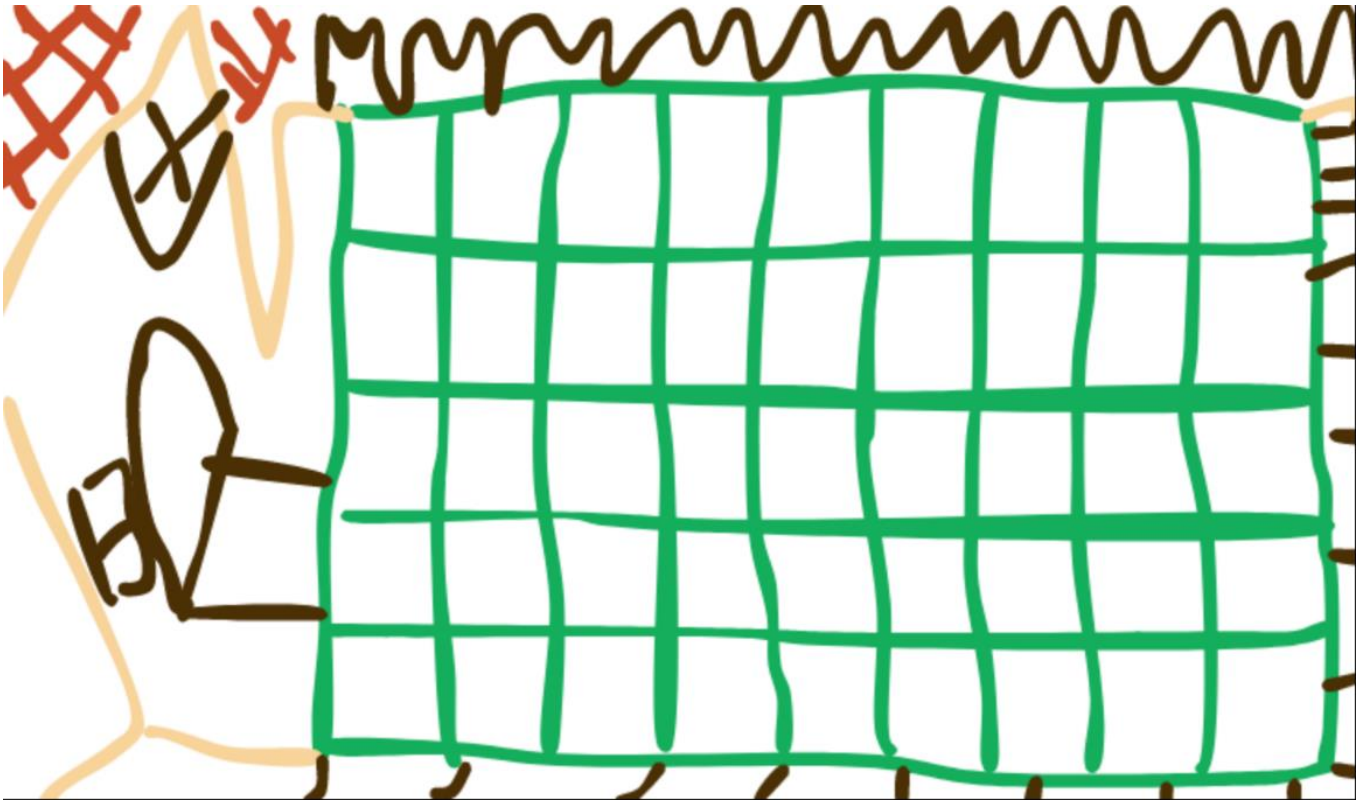
FINAL RELEASE

1. Main menu:

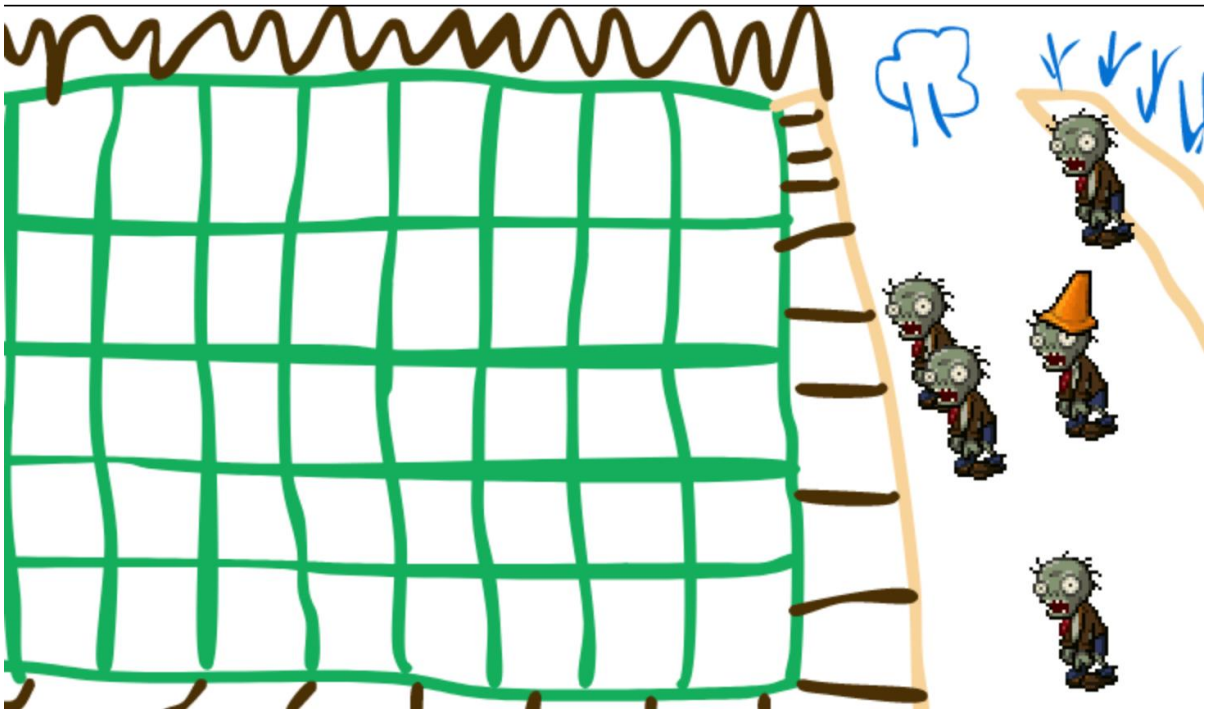
- First start at the menu
- Click on “START ADVENTURE” to play the game

2. Gameplay:

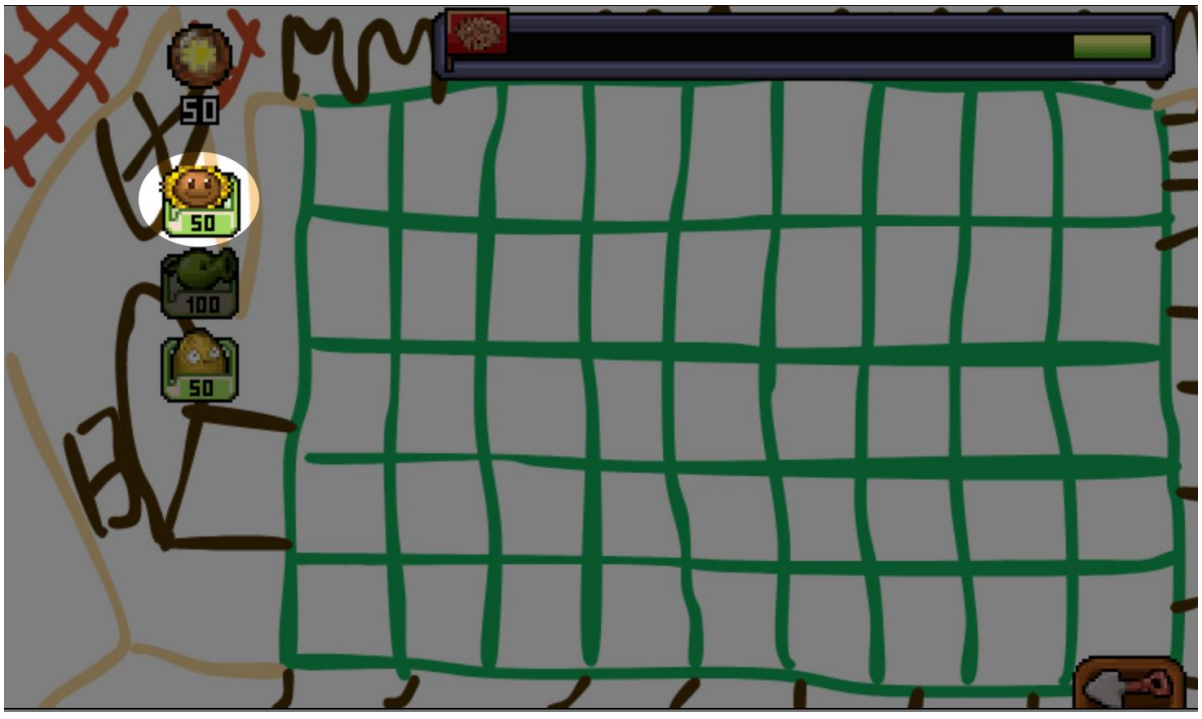
- You will see the big, long frontyard of your house.



- And the game will show you the types of zombies appear in that level.



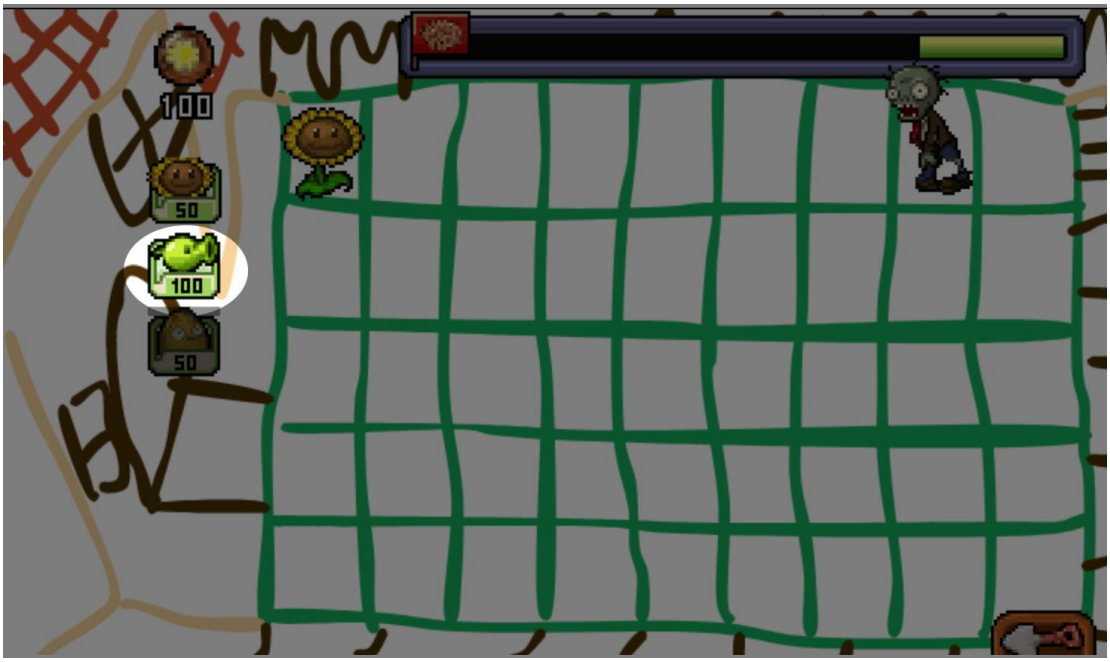
- First select the SunFlower



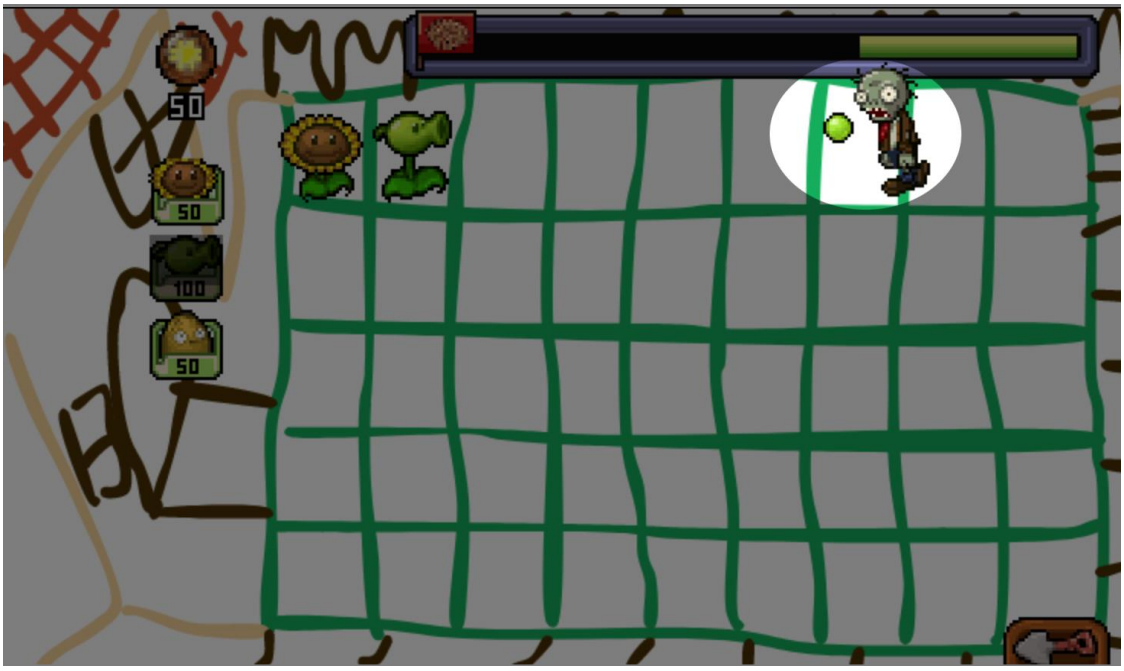
- Then put it in the top left square.

- Collect sun from flower and The Sun.

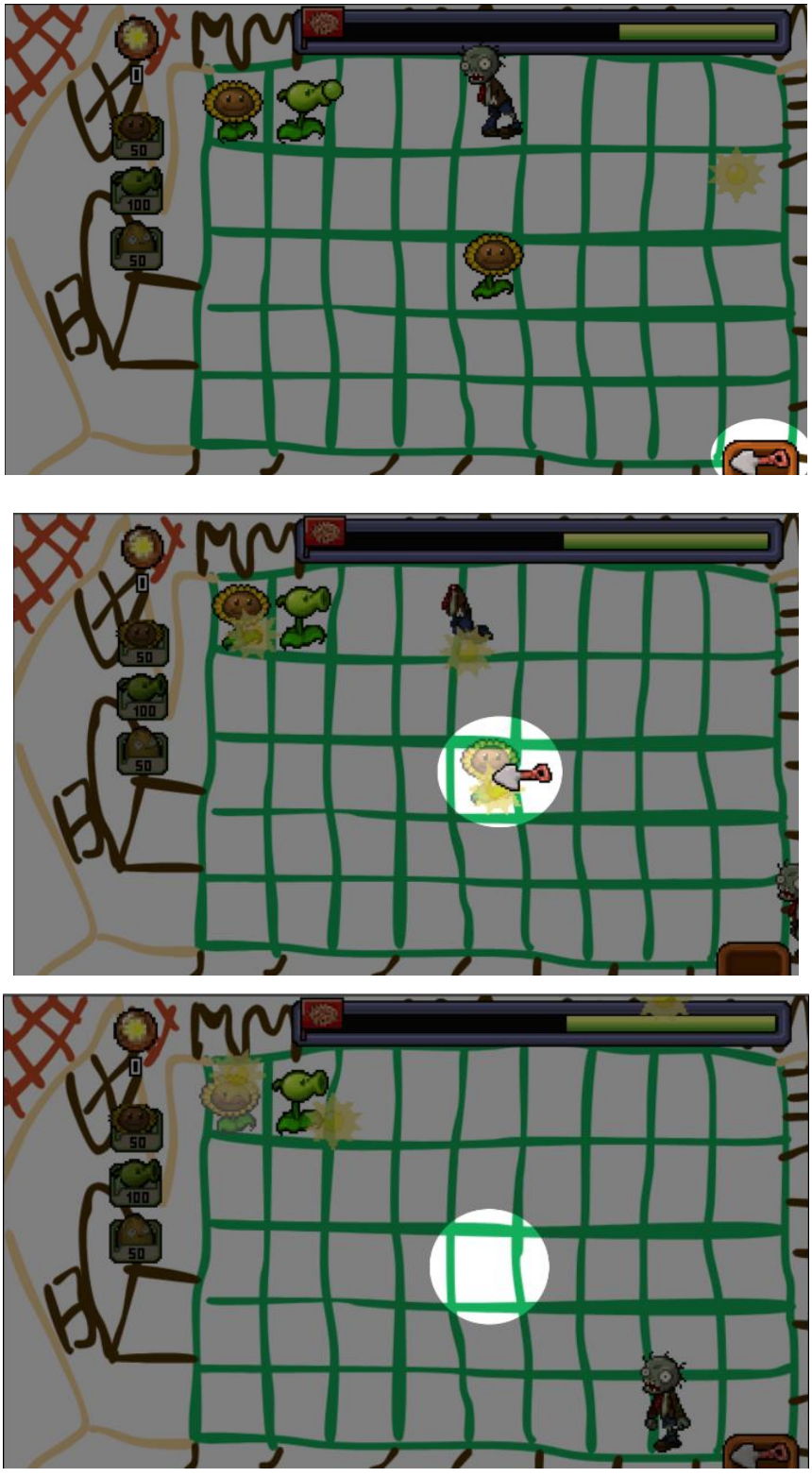
- When you have 100 sun, you can buy PeaShooter to atack the zombies.



- The pea deals damage to enemy from far distance.



- Use the shovel to remove a plant.



- When you lose, a screen will appear to say that…

And you can choose to try again.

PART 5:

EXPERIENCE

Working on the "Plant vs. Zombie" project taught us that a game is much more than just software.

Over two months, we realized that for a game to be enjoyable, it needs rich content, engaging gameplay, and appealing visuals—just like a web server needs quality content to be useful.

While developing the game, we had to consider all aspects: the environment, storyline, gameplay mechanics, artwork, and animations. This project allowed us to apply classroom lessons practically, deal with numerous bugs, and encourage self-directed learning. It reinforced that a computer science major requires continuous self-study, as the IT world is vast and ever-evolving. Our team is committed to completing this game and publishing it as our first project, aiming to develop more applications that enhance the user experience.

And we would like to express our sincere gratitude to our tutor, Mr. Tung, for guiding us through the Object-Oriented Programming (OOP) project. Your support, insightful feedback, and dedication are invaluable. This project has significantly enhanced us in understanding and applicating OOP principles. Thank you for providing an exciting environment of exploration and learning.