**REPORT PROJECT OOP: PLANTS VS ZOMBIES**

**CLASS: TUESDAY MORNING**

**LAB: THURSDAY AFTERNOON**

GROUP MEMBERS:

1. Nguyễn Minh Nghiệp

2. Nguyễn Quốc Việt

3. Trần Đoàn Quốc Đạt

4. Phan Nguyễn Xuân Phúc



**INTERNATIONAL UNIVERSITY**

**PLANTS VS ZOMBIES**

**Object-Oriented Programming**

07/01/2021



WELCOME TO OUR PROJECT

-----------------------------------------------------------------------------------------------

I. Introduction:

Nowadays, technology is growing rapidly and the Software Technology industry also develops. Therefore, the classic Procedural-Oriented Programming method is more and more become out of date and cannot be responded completely for all requirements. That leads the invention of a new method follow the principles Alan-kay named “Object-Oriented Programming” to overcome those cases.

This project was created based on Object-Oriented Programming method by Java language. Therefore, this has solved some problems that was occur while building by common Procedural-Oriented Programming.

* The code is clear, concise and easy to understand.
* The project is a unified logic system by many related-classes combined together.
* Each class has many Methods which take on different behaviors of own class.
* The ability of reusing the resources.

The purpose of this project is to design a basic game that was build on foundation of object-oriented. Our group decided to modify a game named “Plants vs Zombies” follow the principle of the method. This game will show clearly the object-oriented property, the combination of classes, between class and object. Plants vs Zombies require player to use the plant which can shoot out the seeds to kill the zombie and protect the house.

II. Property of Plants vs Zombies:

1. Goal:

In this game, players place different types of plants and fungi, each with their own unique offensive or defensive capabilities in a house in order to stop the zombies. The playing field has 5 to 6 horizontal lanes, and with rare exceptions, a zombie will only move towards the player’s house along one lane.  Most plants can only attack or defend against zombies in the lane they are in.

1. Rule:

The rule is very simple: player can go to the next level when they reach the point by killing the zombie (each zombie is 10 point). In the other hand, if you let the zombie go to home, you will lose and start again.

III. Literature:

1. Class:

In Object-Oriented Programming, a class is a blueprint for creating the objects (a particular data structure), providing initial values for state (member variable or attributes), and implementations of behavior ( member functions or methods ).

The user – defined objects are created using class keyword. The class is a blueprint that defines a nature of a future object. An instance is a specific object created from the particular class. Classes are used to create and manage new objects and support inheritance – a key ingredient in object – oriented programming and a mechanism of reusing code.

Class Declaration:

|  |
| --- |
| Public Class <Class name> {  <list of instance variable>  <list of method>  } |

1. Object:

An object is nothing but a self – contained component which consists of methods and properties to make a particular type of data useful. Object determines the behavior of the class. When you send a message to an object, you are asking the object to invoke or execute one of its method.

From a programming point of view, an object can be a data structure, a variable or a function. It has a memory location allocated. The object is designed as class hierarchies.

Object Declaration:

|  |
| --- |
| ClassName ReferenceVariable = new ClassName() |

1. Other Function:
2. Create game looper:

|  |
| --- |
| VariableName = new Timer (delay, ActionEvent e) -> {  <list of functions>  } |

1. Load Image

|  |
| --- |
| VariableName = new ImageIcon(this.getClass().getResource(Path)).getImage(); |

1. Draw Image

|  |
| --- |
| Boolean Graphics.drawImage(Image img, int x, int y, ImageObserver observer); |

1. Mouse

|  |
| --- |
| public class <NameClass> extends JPanel implements MouseListener {  Public void jPanel1MouseClicked(java.awt.event.MouseEven evt){  GameWindow.begin();  } |

IV. Methodology:

This project is combined by many classes. There are nine main classes, those are Zombie, Pea, Plant, PlantCard, Menu, LevelData, GameWindow, GamePanel and Collider.

**First**, we create those main class. In class Menu, we use JPanel to create the menu screen, and also use MouseEven for clicking.

**Second**, in class GamePanel, we add all of plant’s images and zombie’s images. Moreover, lane of zombies and peas are also created, and the same for the looper for peas and zombies. In the advance function we get zombies and get peas. Last thing we do in this class is that we draw the image and set the condition for the progress.

**Third**, super class Zombie was created. In this class, we set health for the zombie, speed and the initial position. In the advance function, we make the zombie move, the slow effect and so on. The last is the function getZombie. Follow that, we create 3 sub-classes, these are NormalZombie, ConeHeadZomebie (Change the health form 1000 to 1800) and BucketHeadZombie(Increase health to 2600).

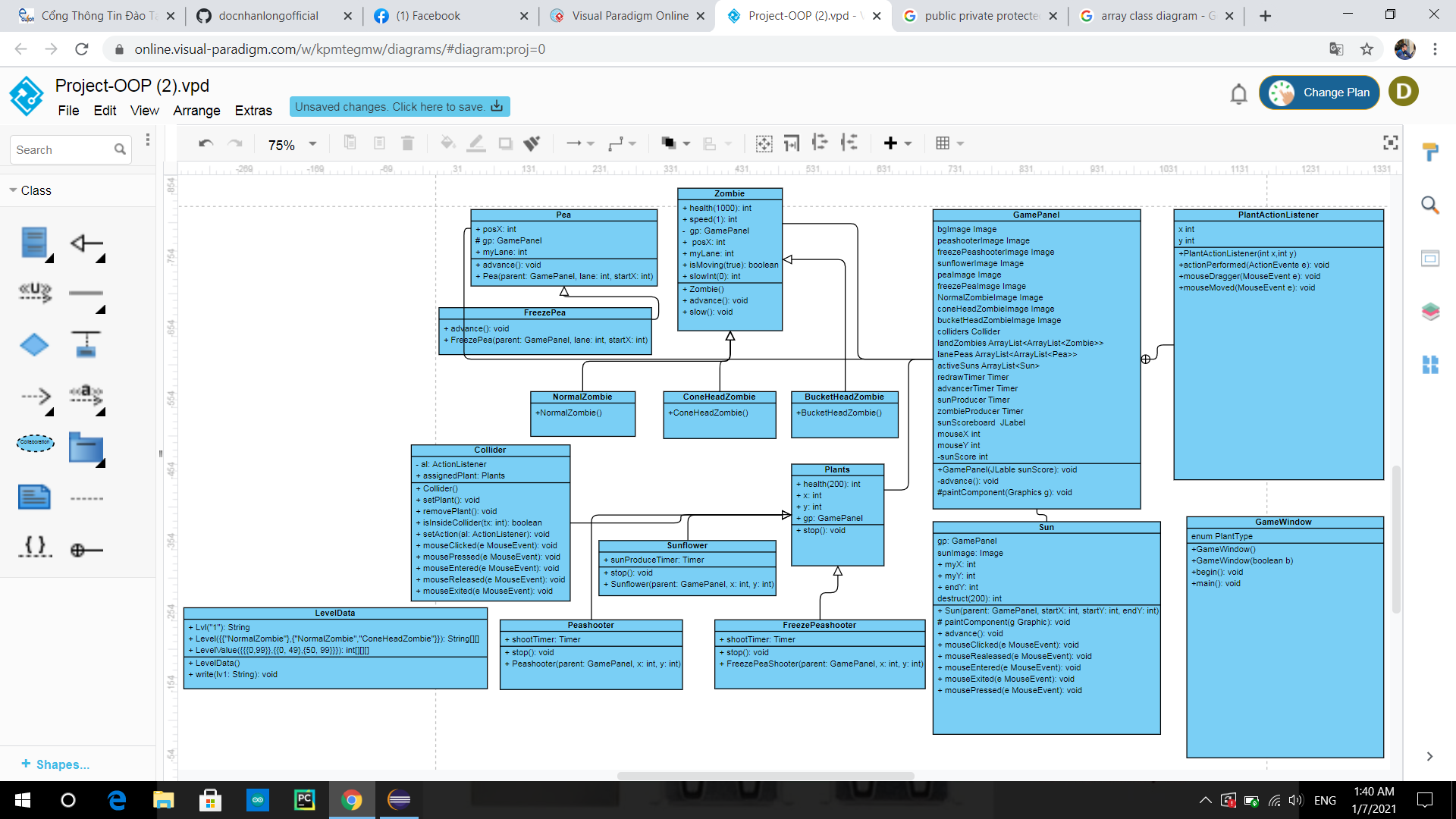
**Fourth**, super class Pea and Plant are created. In Pea class, we set the position, make the bullet move and the effect when they hit the zombies (for example: decrease the health of zombie and plus 10 points when kill zombies). In Plant class, we just set the position of it. The sub-classes for Plant are Peashooter and FreezePeashooter, in those class, we make we set the timer for each pea to shoot the bullet. In sub-classes FreezePea of Pea, we add the slow effect.

**Fifth**, LevelData class is the class which set the level and store as well as reset the data when we win or lose. We import java.io.\*; in order to handle file including read files & write file. The reason we choose this because we want to store the data of our game project. The data will be stored in Level.vbhv. In Level 1, there are 2 kinds of zombies: Normal Zombies & Conehead Zombies. In Level 2, there are 3 kinds of zombies: Normal Zombies, Conehead Zombies & Buckethead Zombies.

**Sixth**, the Collider class and PlantCard class were created. In the Collider class, we set the action for set plant and remove plant and some action. In PlantCard class, we set the image of plant’s card and the action when we click on the plant’s card. If the zombies touch plants, the plants will lose heal & disappear.

**Finally**, the class GameWindow was created, in this we add plant’s cards image. set location for each of plants and create main function in there.

VI. Class Diagram:



VII. Result:

Based on principle of OOP Method, our project has been completely with basic rules and properties of Plants vs Zombies game. The combination of classes and objects in system is relative logically. We also do successfully the display to connect the user with program, it can be control by other input devices like: mouse, keyboard. Moreover, there are some new features out of the basic rule is the sound while playing, alongside with Graphic User Interface to make the user interface is more attractive.

VIII. Limited:

Beside the success of build the game with basic rules, our project still has many cases that cannot be solved:

- Do not have Save Game, Save Score, input the Name for the user.

- This game contains 3 kinds of plants: Sun flower, Pea shooter, Freeze Pea Shooter & 3 kinds of Zombies: Normal Zombie, Cone Head Zombies & Bucket Head Zombies.

- There are only 2 level. We have not yet built more upgrade levels.

IX. Conclusion:

Plants vs Zombies Game that was built by object – oriented method is easier and logically than traditional method. This shows clearly polymorphism, inheritance, encapsulation, data abstraction of OOP and the relationship between Plant, Pea, Zombie alongside with their children classes. Besides that, learning more knowledge out of the limited of this course is one of the important things to do while performing this project.