OOP PROJECT REPORT
Plant Vs Zombie Game

Submitted by:
No Hope Team

Lecturer:
Tran Thanh Tung
Pham Quoc Son Lam

Members:
Đặng Quang Vinh (ITITIU19247)
Trương Hiếu Nghĩa (ITITIU19164)
Lê Văn Quang (ITITIU19195)

Contribution:

Đặng Quang Vinh: To be in charge of all plant class, shot class, level class, synthesize the code ,test run and write report.

Lê Văn Quang: To be in charge of all Zombie class, make powerpoint slide.

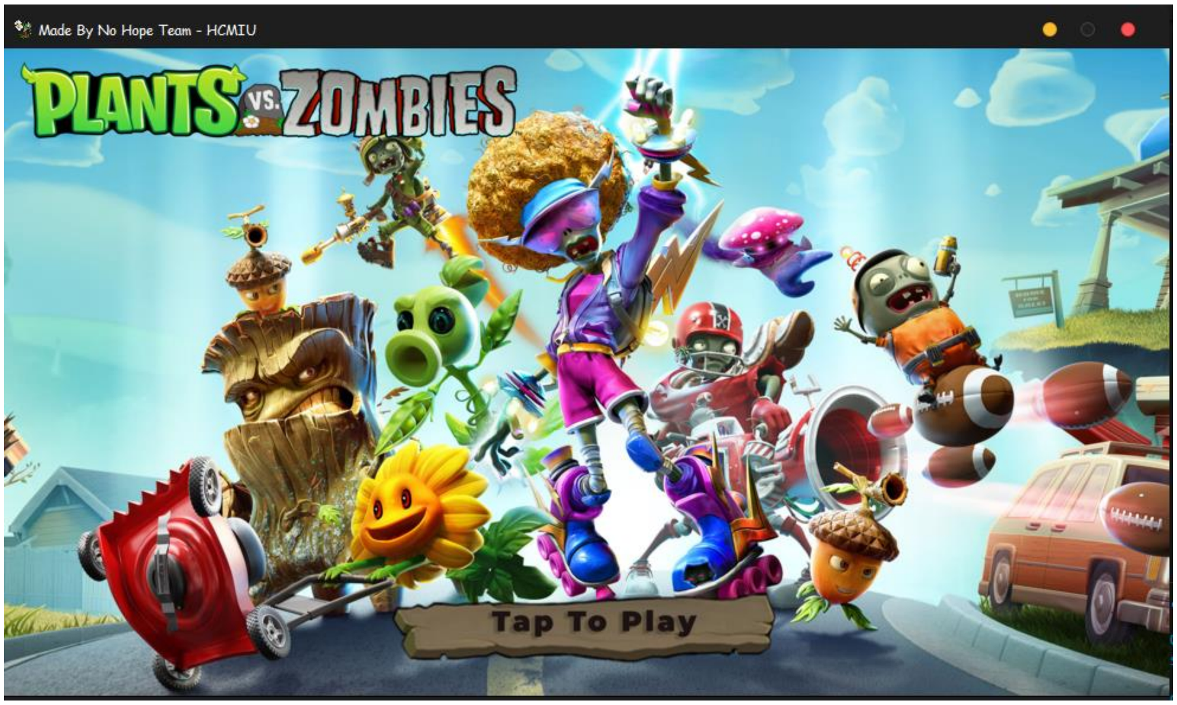
Trương Hiếu Nghĩa: To be in charge of all gui class, gameState class.

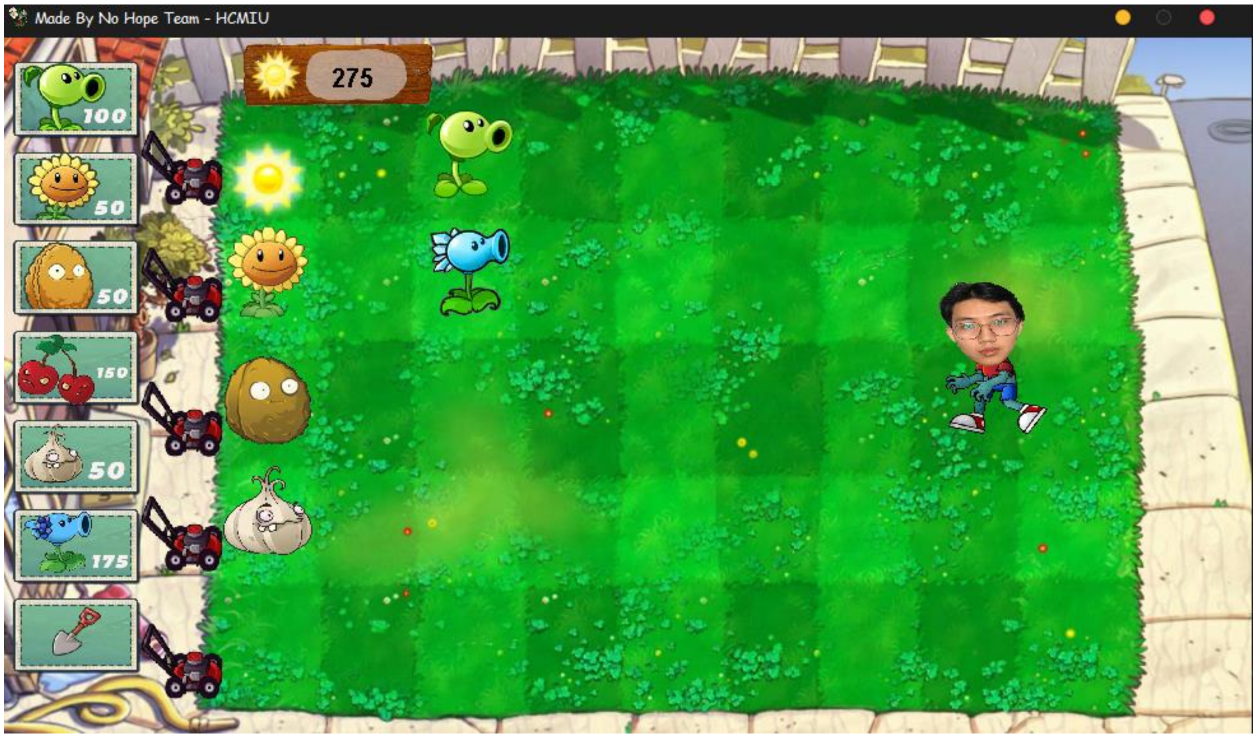
REPORT

I/ INTRODUCTION

Plants vs Zombies is a tower-defense and strategy game which involves an army of zombies trying to enter the house (lawn) and eat your brains. The only way you can stop them is by using your arsenal of plants that will zap the zombies at your command. The plants - Pea-shooters, Walnuts, IceShootPlant, SunMakerPlant,… - are all set to destroy the zombies who dare to enter your backyard.

Some images when game start:





II/ GAME RULE AND FEATURES

1/ Game rule

The playing field is divided into 5 horizontal lanes, a zombie will only move towards the player's house along one lane. Planting costs "sun" (sun acts as the currency for the game, required for buying plants), which can be gathered for free (albeit slowly) and also by planting the sunmaker plant which generates a sun at regular intervals. Plants can only attack or defend against zombies in the lane they are planted in. If zombie eats all plant in lane, the law mower will run and clean all zombie in this lane but law mower will disappear after that and zombie can go to your house (game lose) if zombie eat all plant in the lane.

2/Game feature:

In our implementation we have provided 6 plants ,1 shovel and 1 bar to contain plants to chose:

SimpleShotPlant - SimpleShotPlant is the first and the primary attacking plant the game. It is a peashooting plant that shoots peas on seeing a zombie. It costs 100 suns.

IceShootPlant - IceShootPlant is the second and the primary attacking plant the game. It is a peashooting plant that shoots peas on seeing a zombie and make zombie walk slowly. It costs 175 suns.

SunMakerPlant - SunMakerPlant is an essential sun-producing plant which is necessary for producing suns - the currency of the game. It costs 50 suns. Each sun produced from sunflower adds 25 units to your sun counter.

Walnut - Walnut is the defensive plant that acts as a shield for the player's plants. It takes a long time for zombies to eat it, providing an effective cover for the plants located behind it. It is mainly used to stall zombies to waste their time, letting other plants attack them.
It costs 50 suns.

Cherry Plant – Cherry Plant is the explosive tree because it will explode if one zombie walk through this plant. It costs 150 suns.

GradelPlant – GradelPlant is the strange plant, it make zombie get drunk and make zombie walk in the opposite direction. It costs 150 suns.

Shovel – Shovel will remove the plant you chose in the lane.

PlantInBar – the bar to contain plants to chose.

In this mini version we implemented only 3 zombies:

Quang Zombie – kind of normal zombies with Quang's face put in zombie body. In fact, it takes seven shots for a pea plant to kill them. Having no special defensive equipment or travel abilities, these zombies are susceptible to any type of attack.

Vinh Zombie – kind of normal zombies with Vinh's face put in zombie body. In fact, it takes seven shots for a pea plant to kill them. Having no special defensive equipment or travel abilities, these zombies are susceptible to any type of attack.

Nghĩa Zombie – kind of normal zombies with Nghia's face put in zombie body. In fact, it takes seven shots for a pea plant to kill them. Having no special defensive equipment or travel abilities, these zombies are susceptible to any type of attack.

Github Link : https://github.com/vinhkute24/vinhkute24-Plant-Vs-Zombie-

III/ THE DESIGN: CLASSES IN GAME

1. Class Hierarchy

- JFrame
  - JFrame.GameFrame
- Sound
- GameLoop
- GameState
- Plant
  - Plant.CherryPlanet
  - Plant.GradelPlant
  - Plant.IceShootPlant
  - Plant.Shovel
  - Plant.SimpleShootPlant
  - Plant.SunMakerPlant
  - Plant.WalnutPlant
  - Plant.PlantInPlantBar
- Zombie

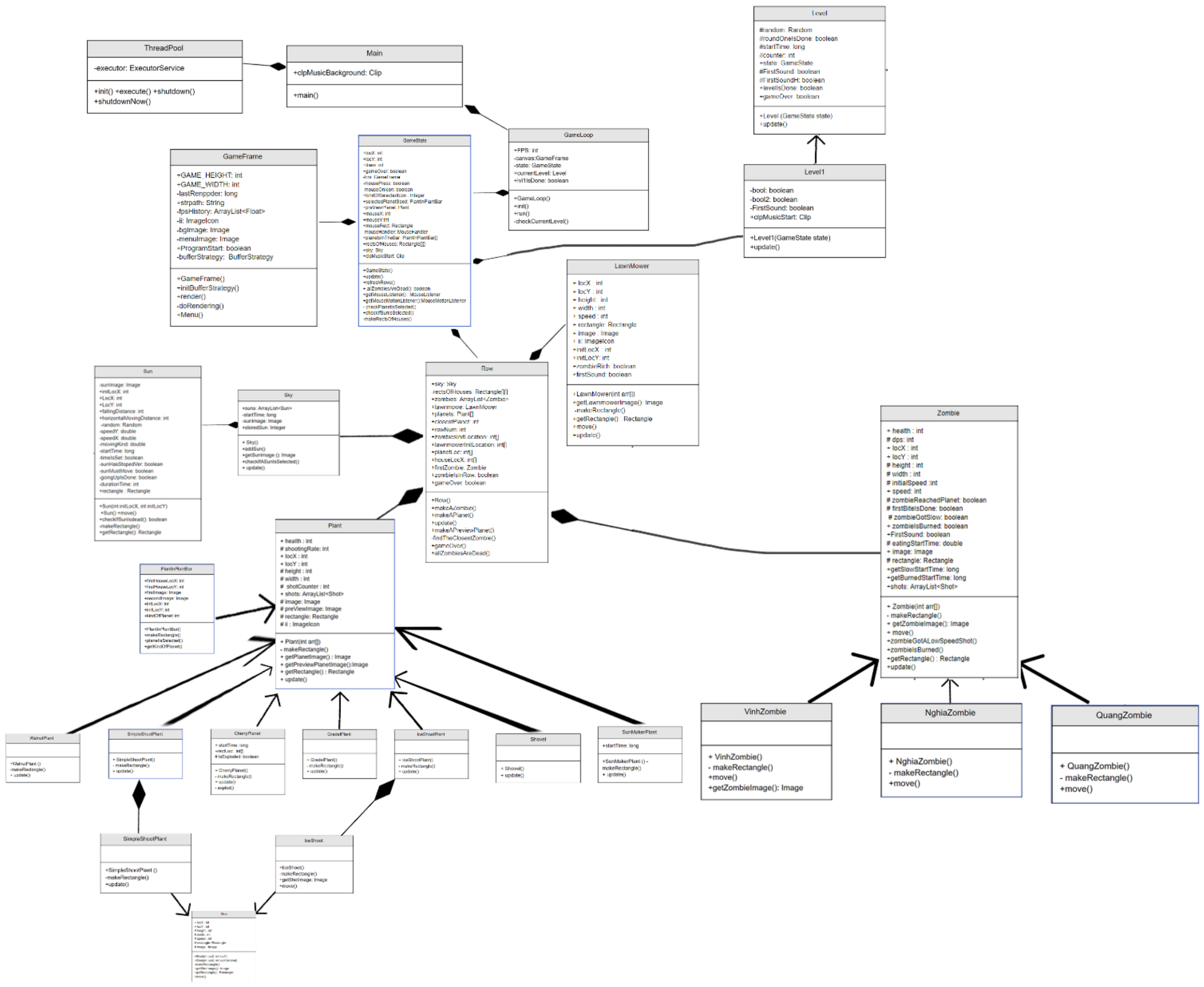
- Zombie.QuangZombie
- Zombie.NghiaZombie
- Zombie.VinhZombie

ThreadPool
Row
Shot
- Shot.SimpleShot
- Shot.IceShot

Level
- Level.Level1

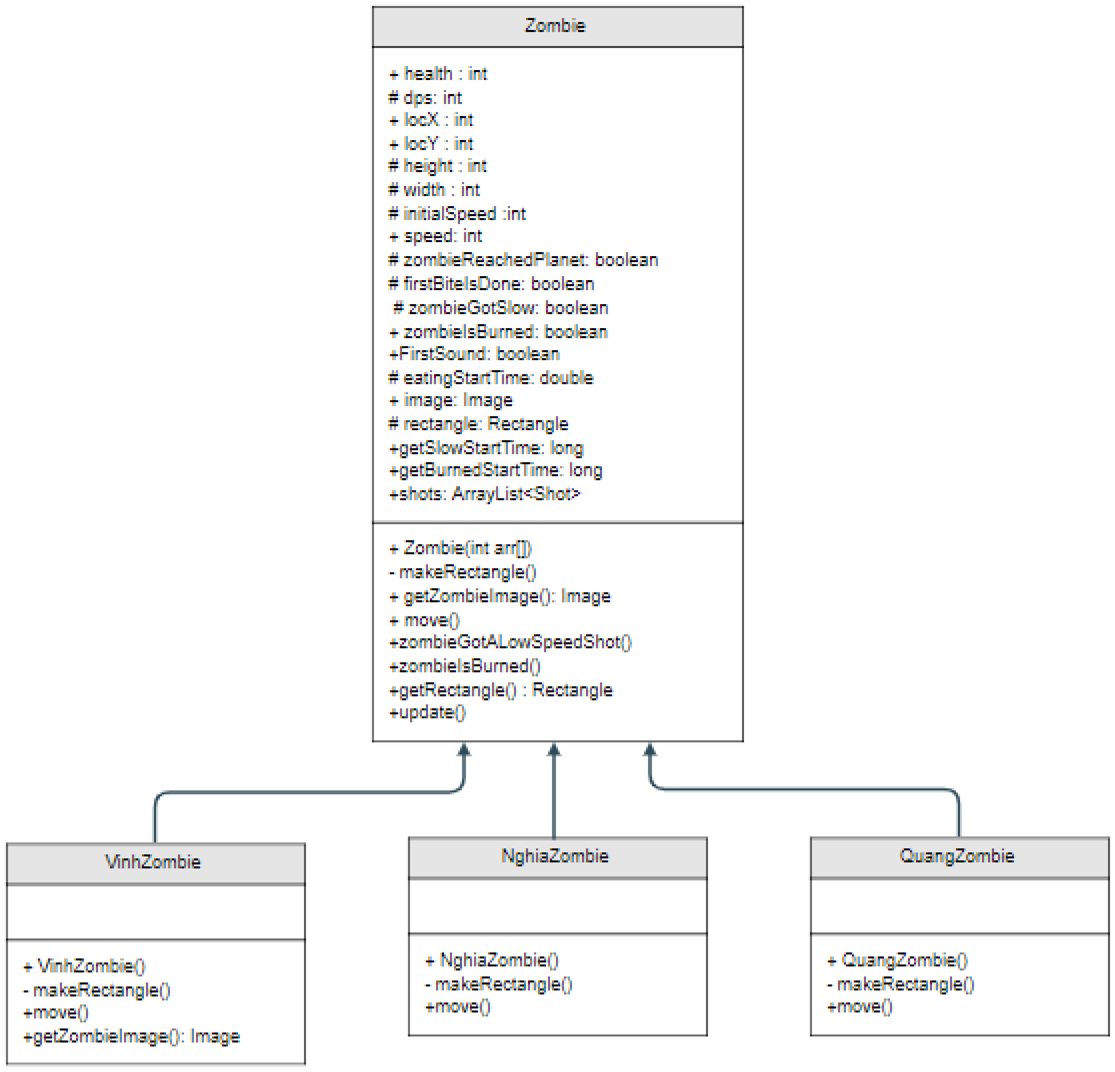
LawnMower
Main
Sky
Sun

2. UML Diagram of Plant Vs Zombie Game



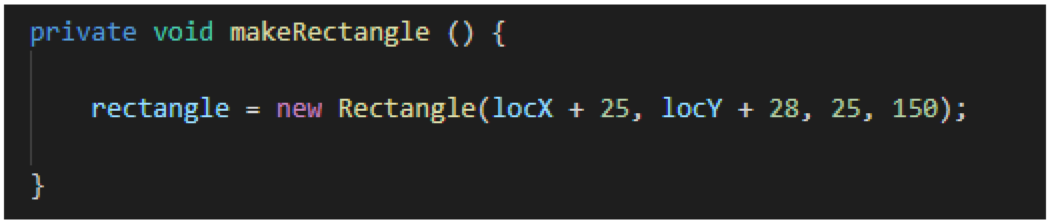
IV/ MAIN CLASS – FUNCTION – EXPLAIN:

1. Zombie Class



Zombie Class is created to desgin others zombies easilly. Class VinhZombie, QuangZombie, NghiaZombie are inheritant from the zombie Class about attributes and fucntions. In Zombie class we create zombie by draw a rectangle represent for 1 zombie

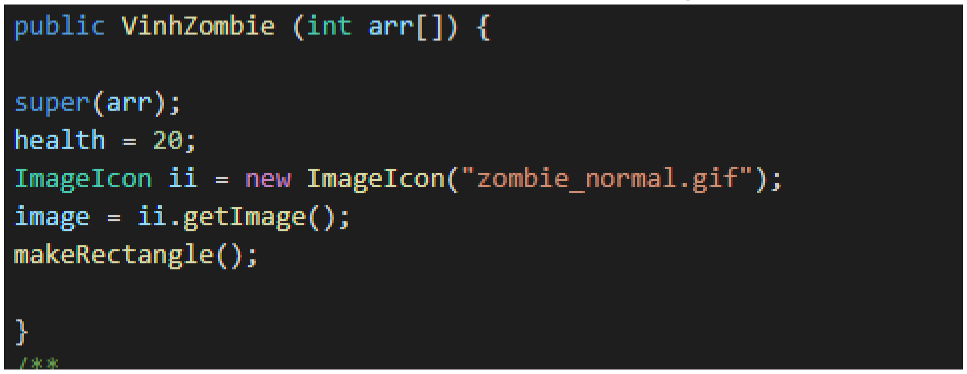
//Function to draw a rectangle arround the zombie



```
private void makeRectangle () {

    rectangle = new Rectangle(locX + 25, locY + 28, 25, 150);

}
```

Take the zombie image to user can see the zombie and play easily, zombie gif file to make zombie move smoothly.

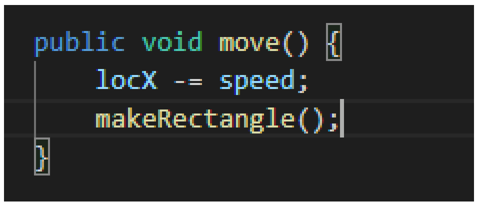
//Zombie gif file and rectangle are created in zombie constructor. (picture below is VinhZombie's constructor)



```
public VinhZombie (int arr[]) {

super(arr);
health = 20;
ImageIcon ii = new ImageIcon("zombie_normal.gif");
image = ii.getImage();
makeRectangle();

}
/**
```

Then, the zombie will move by move by subtracting coordinates x with speed is initalized with each zombie.

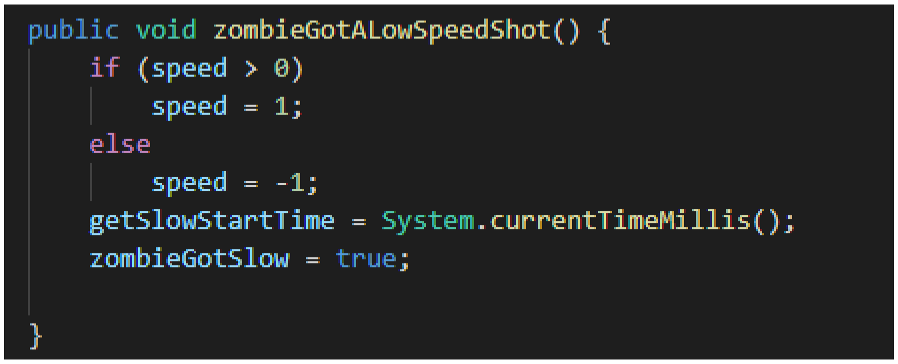
//Function make zombie move



```
public void move() {
    locX -= speed;
    makeRectangle();
}
```

Beside that, the zombie also get slow down when iceShotPlant shot.

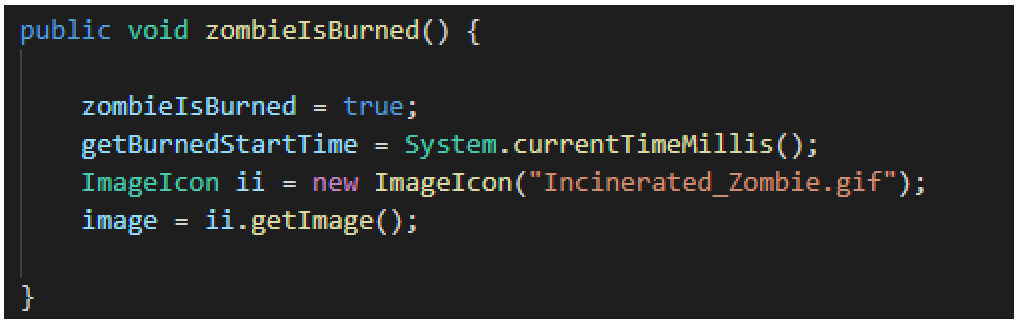
//Function to make zombie slow down



```
public void zombieGotALowSpeedShot() {
    if (speed > 0)
        speed = 1;
    else
        speed = -1;
    getSlowStartTime = System.currentTimeMillis();
    zombieGotSlow = true;

}
```

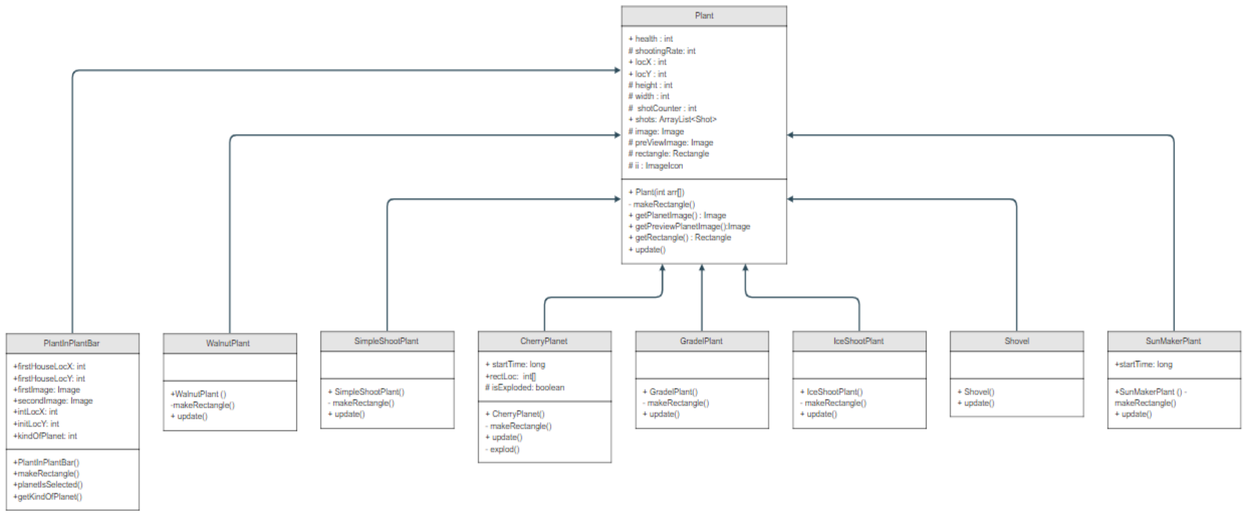
Or, being burned when walk through the cherry plant.

//Function to make zombie burned (take image zombie is burned)



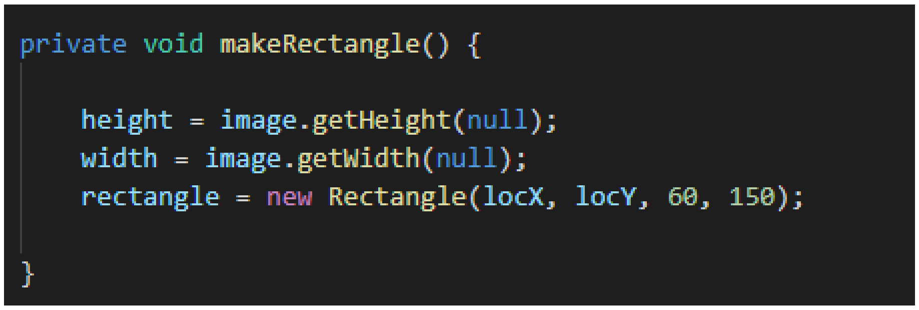
```
public void zombieIsBurned() {

    zombieIsBurned = true;
    getBurnedStartTime = System.currentTimeMillis();
    ImageIcon ii = new ImageIcon("Incinerated_Zombie.gif");
    image = ii.getImage();

}
```

2/ Plant class



Plant Class is created to desgin others plant easilly. Class WalnutPlant, SimpleShootPlant, IceShootPlant,... are inheritant from the Plant Class about attributes and fucntions. In Plant class we create plant by draw a rectangle represent for 1 kind of plant

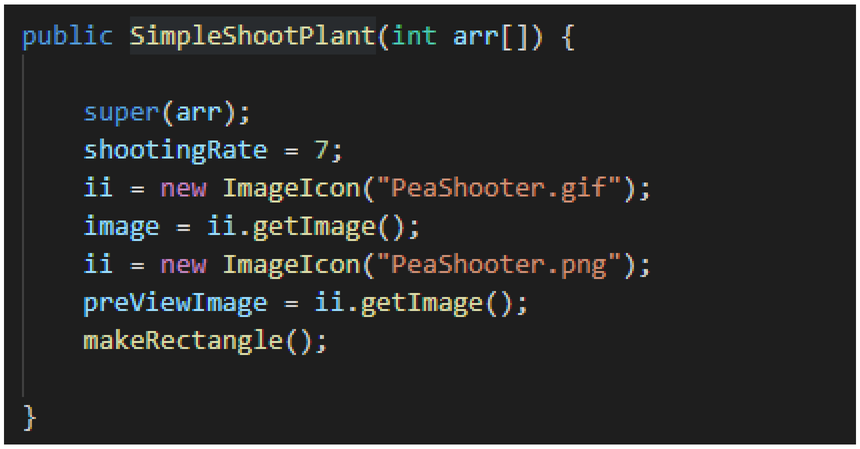
//Function to draw a rectangle arround the plant



```
private void makeRectangle() {

    height = image.getHeight(null);
    width = image.getWidth(null);
    rectangle = new Rectangle(locX, locY, 60, 150);

}
```

Take the plant image to user can see the plant and play easily, plant gif file to make plant move smoothly.

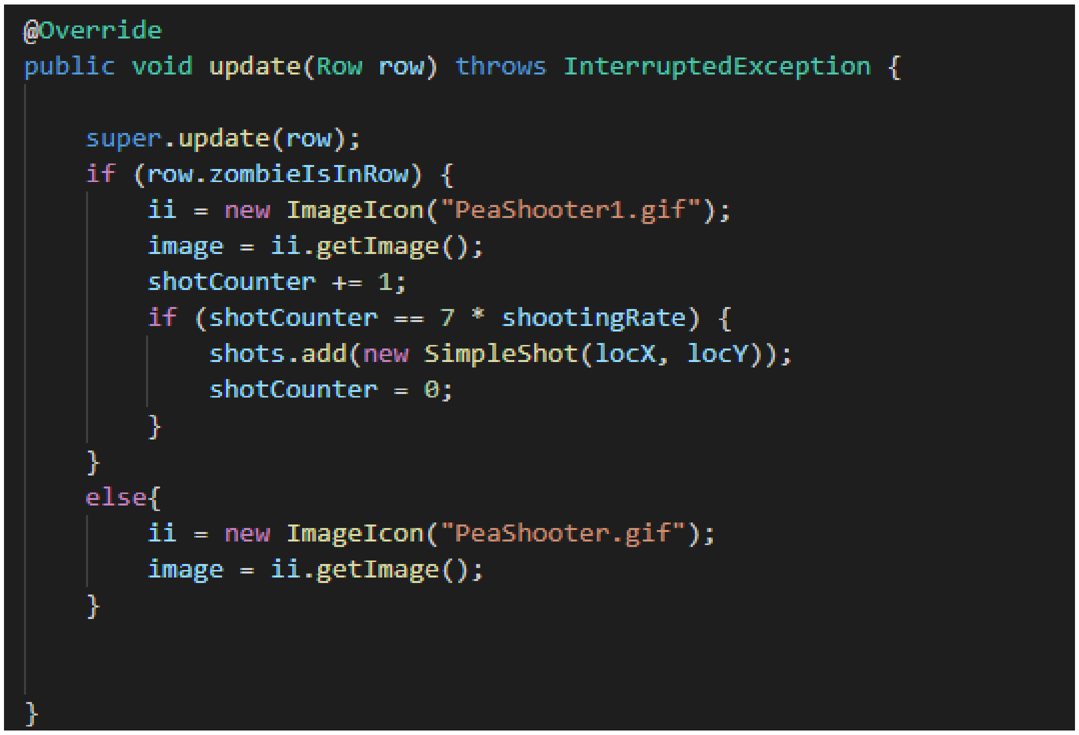
//Plant gif file and rectangle are created in plant constructor. (picture below is SimpleShootPlant's constructor)



```java
public SimpleShootPlant(int arr[]) {

    super(arr);
    shootingRate = 7;
    ii = new ImageIcon("PeaShooter.gif");
    image = ii.getImage();
    ii = new ImageIcon("PeaShooter.png");
    preViewImage = ii.getImage();
    makeRectangle();

}
```

With update void function of SimpleShootPlant class we create the activity prepare to shoot and shoot by checking zombie walk through to the row or not (zombieIsInRow function). Beside that, this plant will attack the zombie and make zombie lose health then die if health = 0.

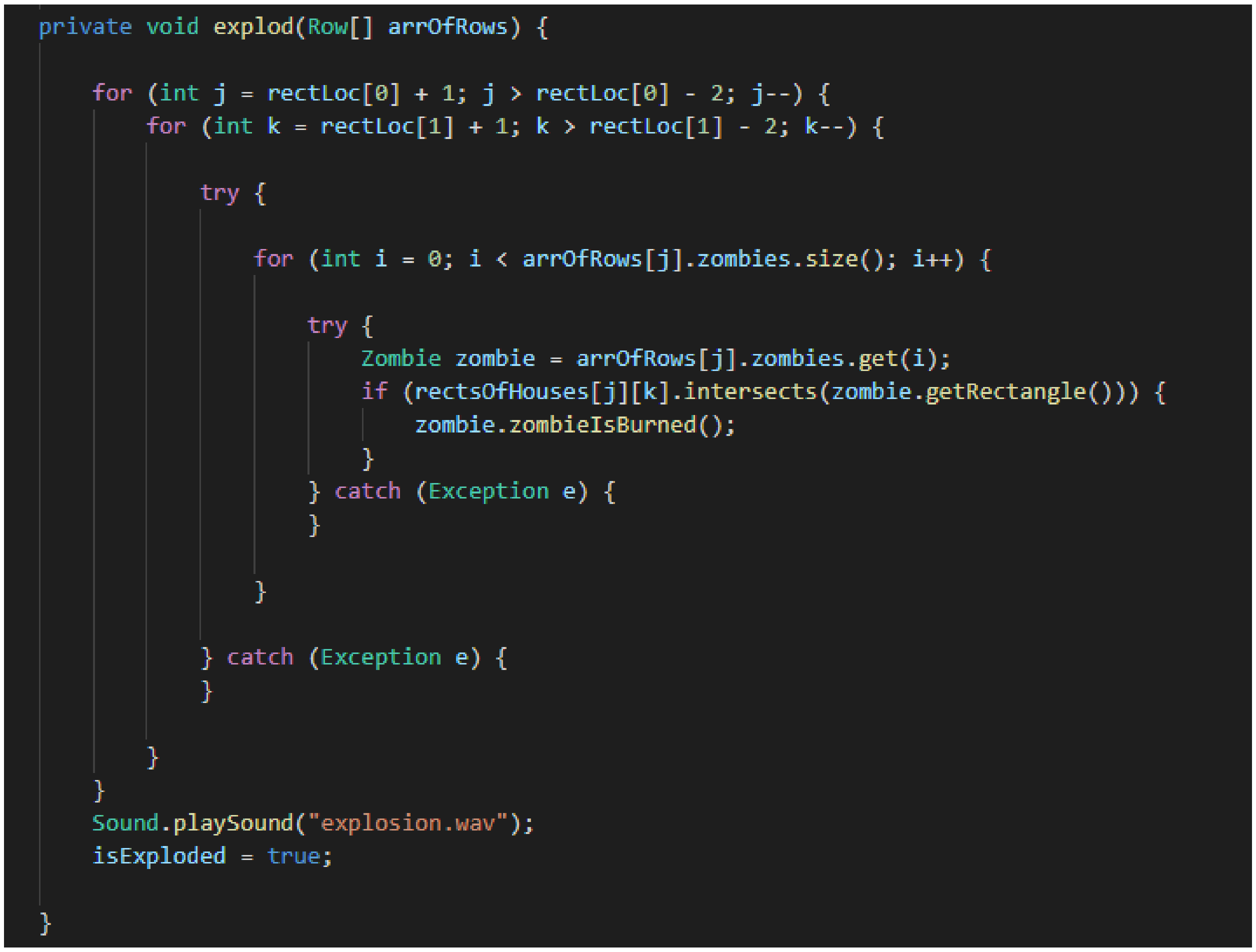
//update void function in SimpleShootPlant



```java
@Override
public void update(Row row) throws InterruptedException {

    super.update(row);
    if (row.zombieIsInRow) {
        ii = new ImageIcon("PeaShooter1.gif");
        image = ii.getImage();
        shotCounter += 1;
        if (shotCounter == 7 * shootingRate) {
            shots.add(new SimpleShot(locX, locY));
            shotCounter = 0;
        }
    }
    else{
        ii = new ImageIcon("PeaShooter.gif");
        image = ii.getImage();
    }


}
```

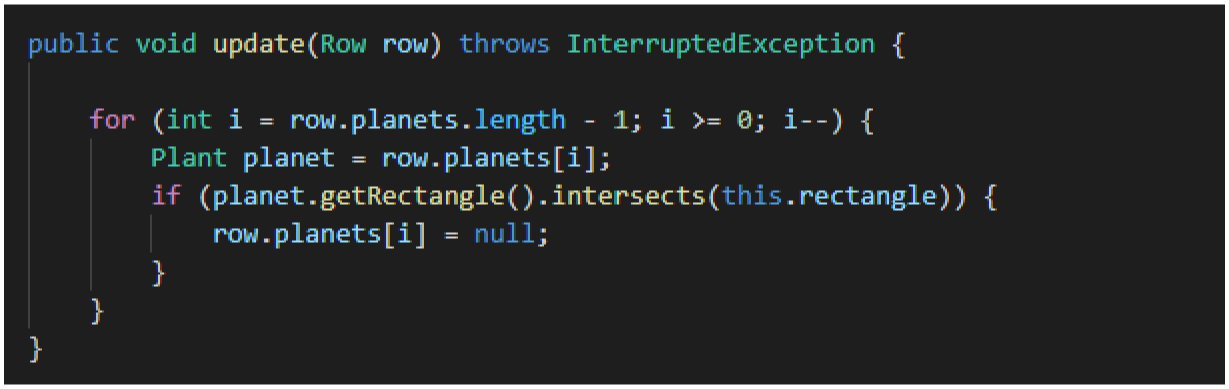
Each plant has different features.Like Cherry Plant, the plant will explore and burn 9 zombies arround the lane.

//Function to burn Zombies (Cherry Plant class)



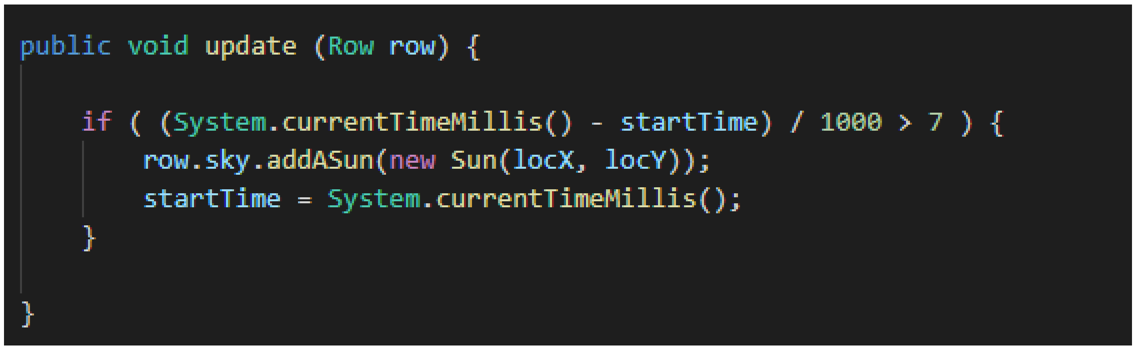
```java
private void explod(Row[] arrOfRows) {

    for (int j = rectLoc[0] + 1; j > rectLoc[0] - 2; j--) {
        for (int k = rectLoc[1] + 1; k > rectLoc[1] - 2; k--) {

            try {

                for (int i = 0; i < arrOfRows[j].zombies.size(); i++) {

                    try {
                        Zombie zombie = arrOfRows[j].zombies.get(i);
                        if (rectsOfHouses[j][k].intersects(zombie.getRectangle())) {
                            zombie.zombieIsBurned();
                        }
                    } catch (Exception e) {
                    }

                }

            } catch (Exception e) {
            }

        }
    }
    Sound.playSound("explosion.wav");
    isExploded = true;

}
```

In Shovel class, the update void function is about remove the plant in the lane when you chose it.



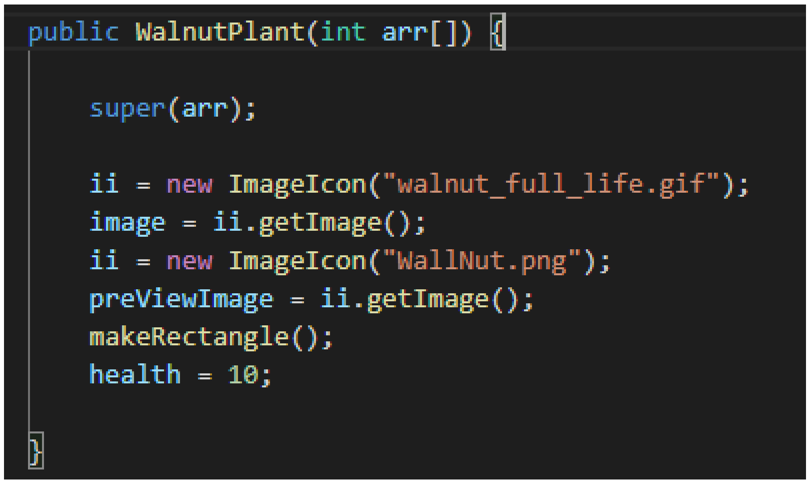
```java
public void update(Row row) throws InterruptedException {

    for (int i = row.planets.length - 1; i >= 0; i--) {
        Plant planet = row.planets[i];
        if (planet.getRectangle().intersects(this.rectangle)) {
            row.planets[i] = null;
        }
    }
}
```

In SunMakerPlant class, the update void function is about add new sun in the lane each 7 minutes, if you chose this your store sun will increace 25 suns to buy another plant.



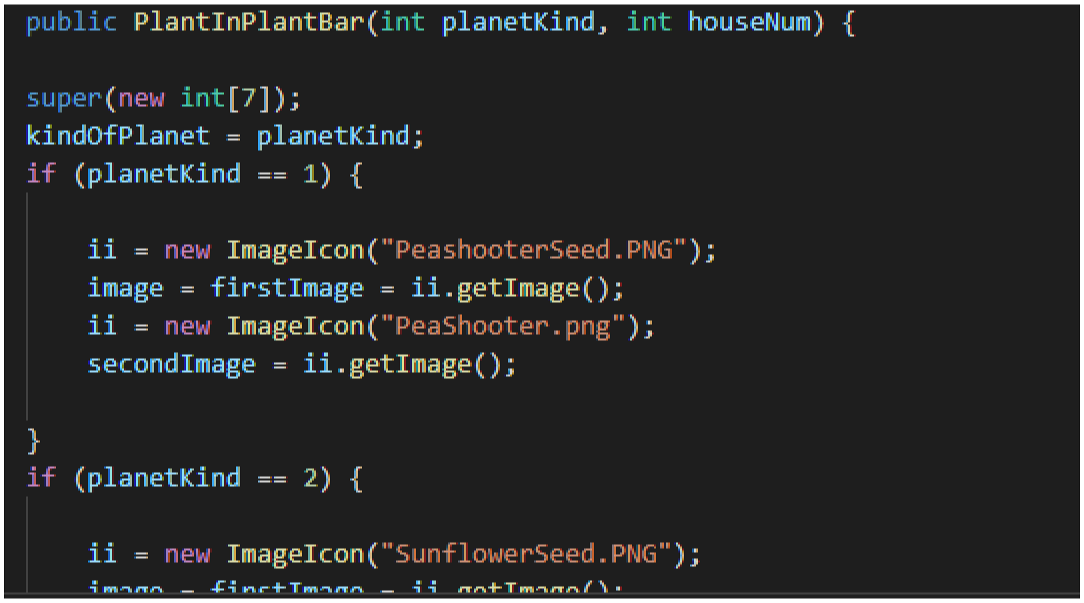
```
public void update (Row row) {

    if ( (System.currentTimeMillis() - startTime) / 1000 > 7 ) {
        row.sky.addASun(new Sun(locX, locY));
        startTime = System.currentTimeMillis();
    }

}
```

WalnutPlant class – the WalnutPlant does not have any feature special except this plant have a huge health, to take longer time when zombie eat.



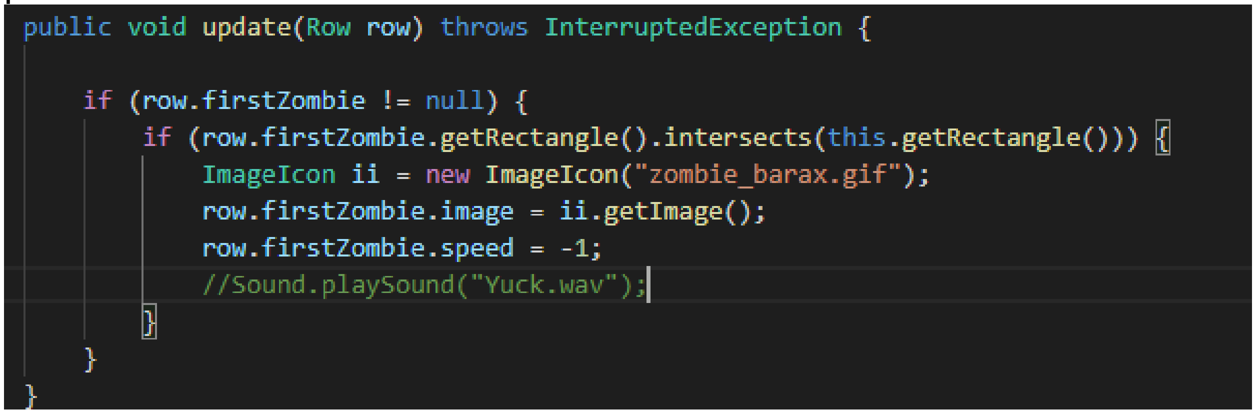
```
public WalnutPlant(int arr[]) {

    super(arr);

    ii = new ImageIcon("walnut_full_life.gif");
    image = ii.getImage();
    ii = new ImageIcon("WallNut.png");
    preViewImage = ii.getImage();
    makeRectangle();
    health = 10;

}
```

Class PlantInPlantBar - is the class that store all plant you can chose to plant. We create one-dimensional to store of all kind of plant and get plant image in the Plant bar GUI in game.



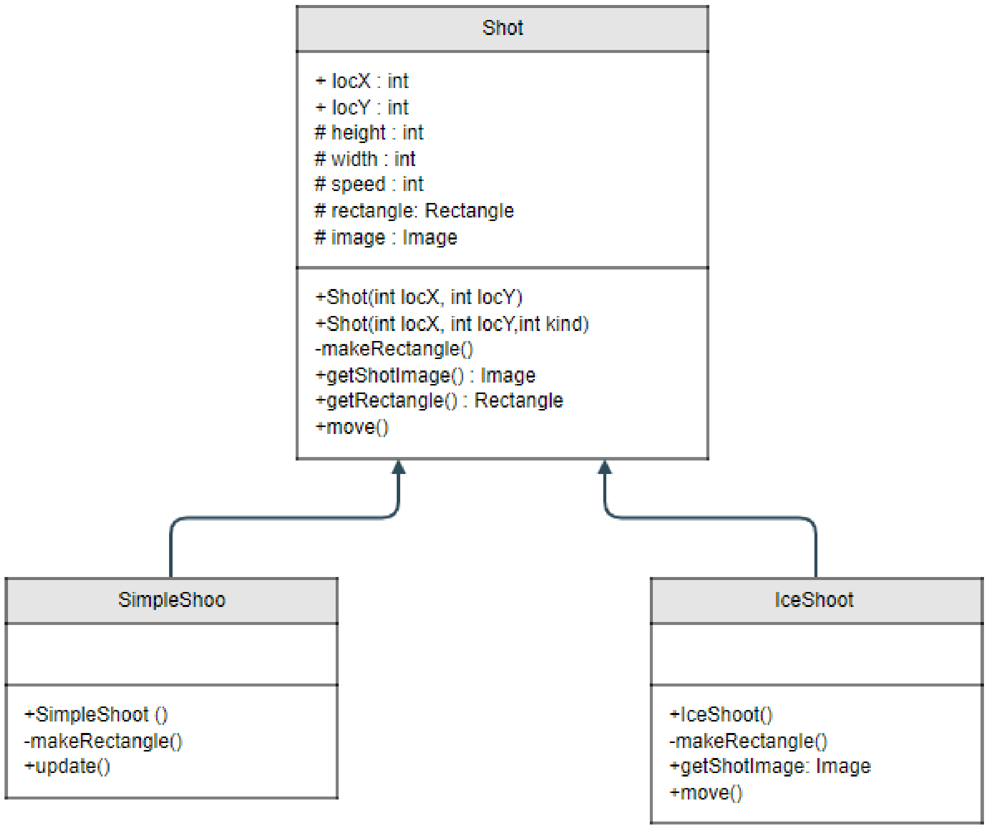
```java
public PlantInPlantBar(int planetKind, int houseNum) {

super(new int[7]);
kindOfPlanet = planetKind;
if (planetKind == 1) {

    ii = new ImageIcon("PeashooterSeed.PNG");
    image = firstImage = ii.getImage();
    ii = new ImageIcon("PeaShooter.png");
    secondImage = ii.getImage();

}
if (planetKind == 2) {

    ii = new ImageIcon("SunflowerSeed.PNG");
    image = firstImage = ii.getImage();
```

With update() void function in GradelPlant class, we will make zombie turn the head and walk in the opposite way when zombie walk through the plant.

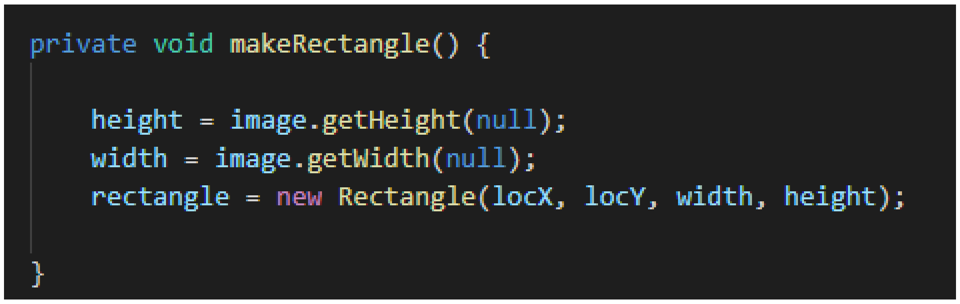


```java
public void update(Row row) throws InterruptedException {

    if (row.firstZombie != null) {
        if (row.firstZombie.getRectangle().intersects(this.getRectangle())) {
            ImageIcon ii = new ImageIcon("zombie_barax.gif");
            row.firstZombie.image = ii.getImage();
            row.firstZombie.speed = -1;
            //Sound.playSound("Yuck.wav");
        }
    }
}
```

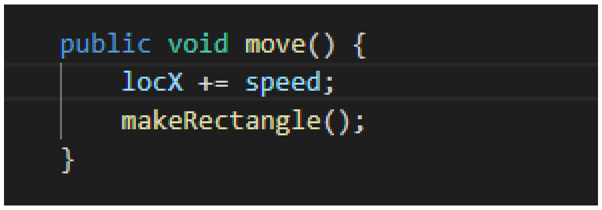
3. Shot class



In shot class we create the shoot and collision handling between the bullet and zombie. First we draw a rectangle then get bullet image base on the rectangle.The SimpleShootPlant class and IceShoot will inheritant the Shoot class about attributes and fucntions. Then The SimpleShoot will be initialized in class SimpleShootPlant and IceShoot will be initialized in class IceShootPlant


```java
private void makeRectangle() {

    height = image.getHeight(null);
    width = image.getWidth(null);
    rectangle = new Rectangle(locX, locY, width, height);

}
```

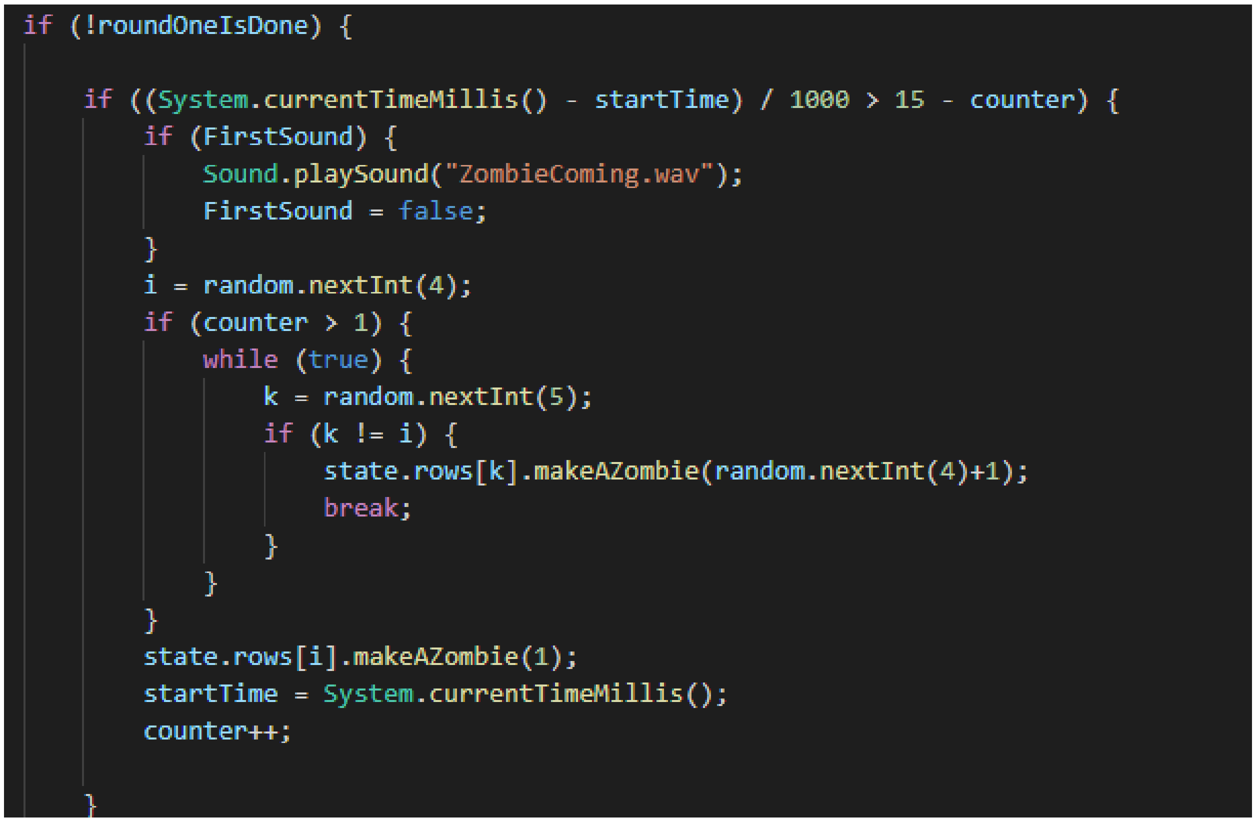
Then we make the bullet move by plus location x.


```java
public void move() {
    locX += speed;
    makeRectangle();
}
```

4. Level class

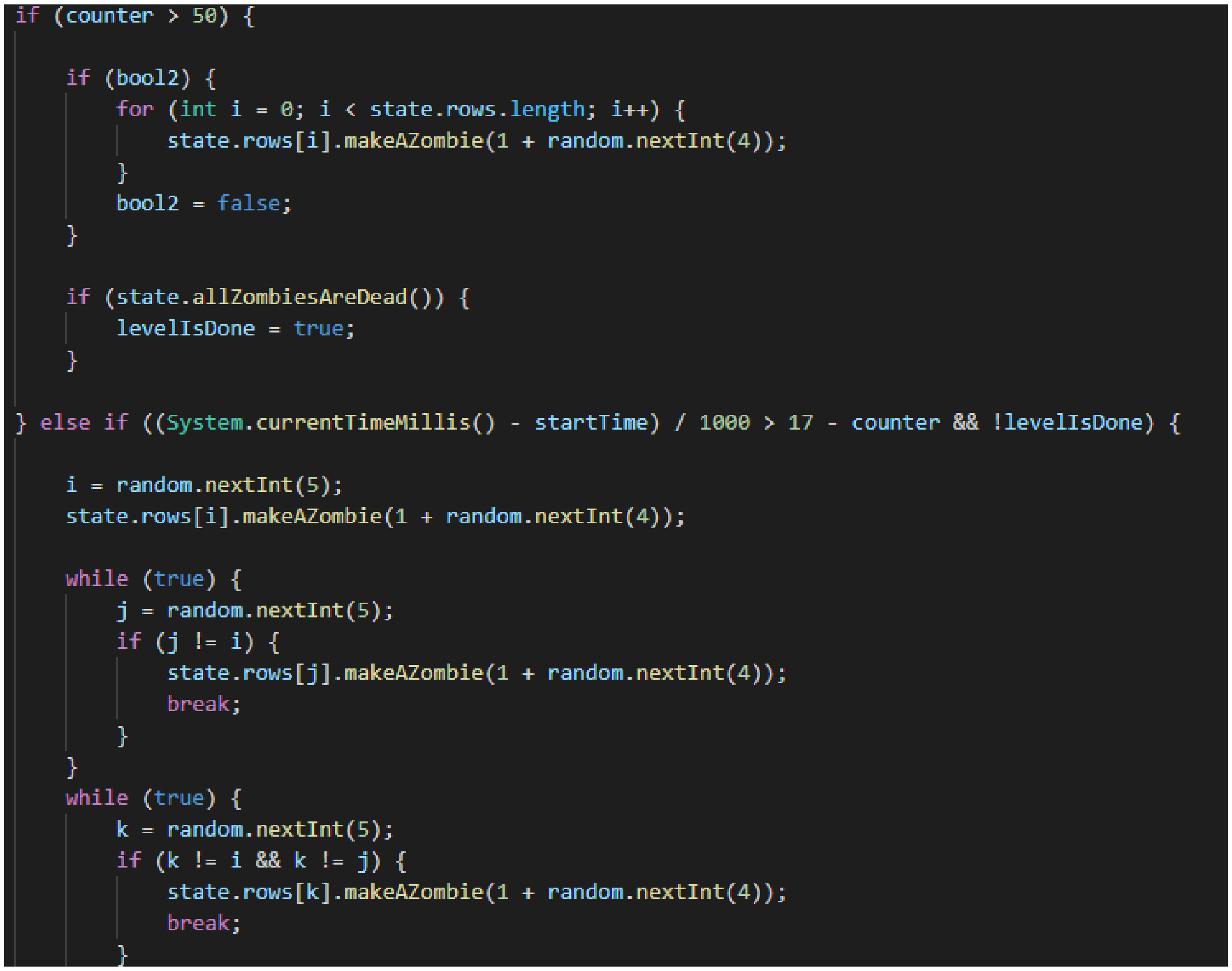
In this class we will level of the game. Currently we only have level 1. In level 1, we have two state: state 1 which have a few zombie appear and state 2 is the big wave which have a lot of zombies.

//State 1 algorithm



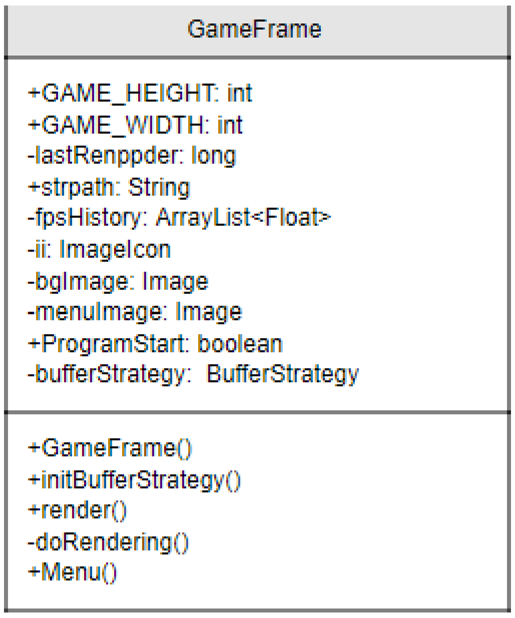
```
if (!roundOneIsDone) {

    if ((System.currentTimeMillis() - startTime) / 1000 > 15 - counter) {
        if (FirstSound) {
            Sound.playSound("ZombieComing.wav");
            FirstSound = false;
        }
        i = random.nextInt(4);
        if (counter > 1) {
            while (true) {
                k = random.nextInt(5);
                if (k != i) {
                    state.rows[k].makeAZombie(random.nextInt(4)+1);
                    break;
                }
            }
        }
        state.rows[i].makeAZombie(1);
        startTime = System.currentTimeMillis();
        counter++;

    }
```

//state 2 algorithm



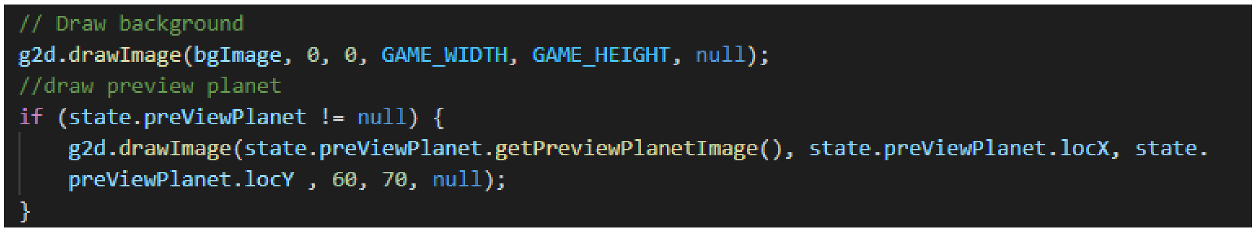
```java
if (counter > 50) {

    if (bool2) {
        for (int i = 0; i < state.rows.length; i++) {
            state.rows[i].makeAZombie(1 + random.nextInt(4));
        }
        bool2 = false;
    }

    if (state.allZombiesAreDead()) {
        levelIsDone = true;
    }

} else if ((System.currentTimeMillis() - startTime) / 1000 > 17 - counter && !levelIsDone) {

    i = random.nextInt(5);
    state.rows[i].makeAZombie(1 + random.nextInt(4));

    while (true) {
        j = random.nextInt(5);
        if (j != i) {
            state.rows[j].makeAZombie(1 + random.nextInt(4));
            break;
        }
    }
    while (true) {
        k = random.nextInt(5);
        if (k != i && k != j) {
            state.rows[k].makeAZombie(1 + random.nextInt(4));
            break;
        }
```

5. GUI – GameFrame class



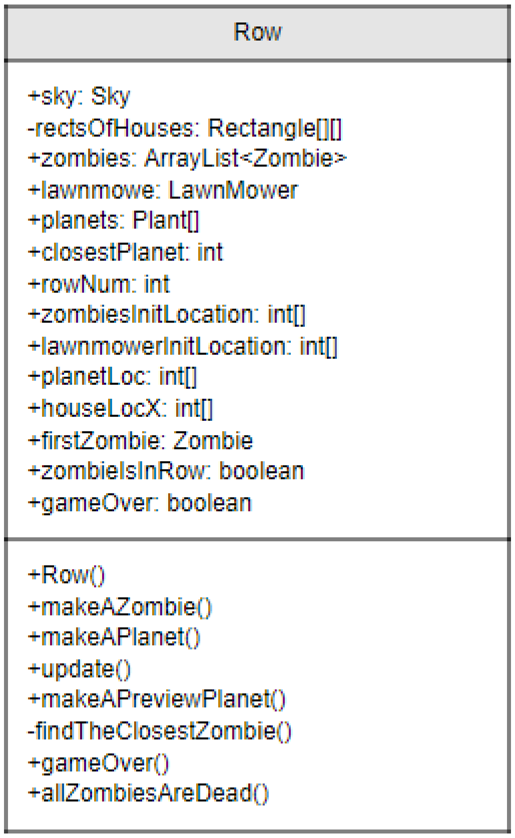
This class, will extend Jframe to draw all the image of the game when game start. \

//Example the function to draw preview planet and background



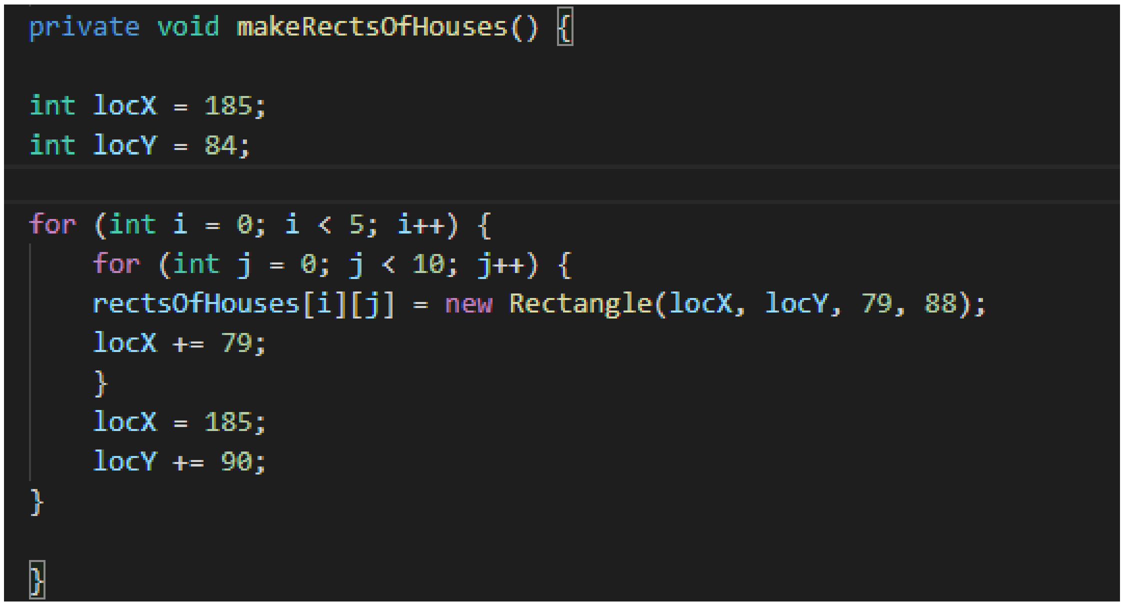
```
// Draw background
g2d.drawImage(bgImage, 0, 0, GAME_WIDTH, GAME_HEIGHT, null);
//draw preview planet
if (state.preViewPlanet != null) {
    g2d.drawImage(state.preViewPlanet.getPreviewPlanetImage(), state.preViewPlanet.locX, state.
    preViewPlanet.locY , 60, 70, null);
}
```

6. Row class



```
+sky: Sky
-rectsOfHouses: Rectangle[][]
+zombies: ArrayList<Zombie>
+lawnmowe: LawnMower
+planets: Plant[]
+closestPlanet: int
+rowNum: int
+zombiesInitLocation: int[]
+lawnmowerInitLocation: int[]
+planetLoc: int[]
+houseLocX: int[]
+firstZombie: Zombie
+zombieIsInRow: boolean
+gameOver: boolean

+Row()
+makeAZombie()
+makeAPlanet()
+update()
+makeAPreviewPlanet()
-findTheClosestZombie()
+gameOver()
+allZombiesAreDead()
```

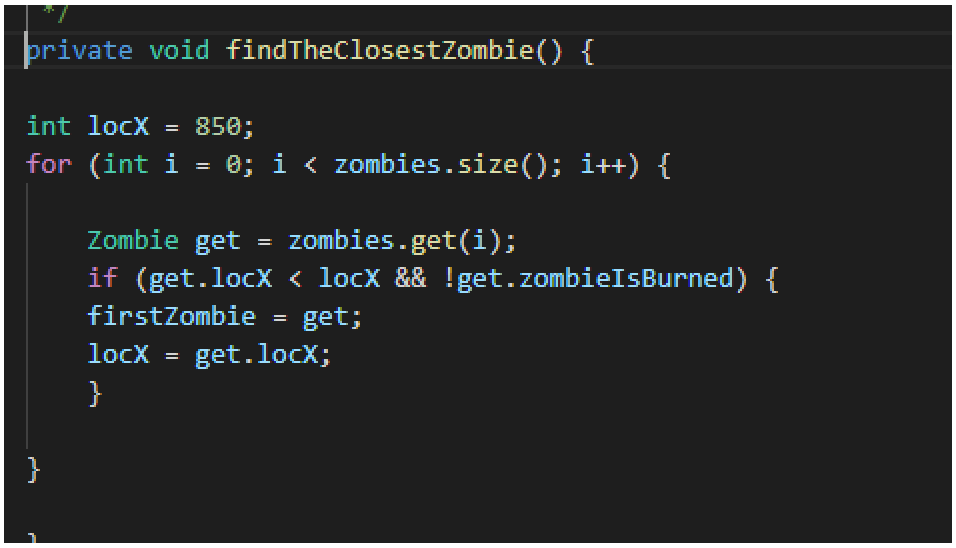
This is the class which handles zombies moving,planets shooting and death of zombies or plants, make a rectangle of the house.
With draw reactangle of the house represent of each house’s square, We will create a game state with two-dimensional array with 5 columns and 10 rows then we draw rectangle representing each square in the yard



```
private void makeRectsOfHouses() {

int locX = 185;
int locY = 84;

for (int i = 0; i < 5; i++) {
    for (int j = 0; j < 10; j++) {
    rectsOfHouses[i][j] = new Rectangle(locX, locY, 79, 88);
    locX += 79;
    }
    locX = 185;
    locY += 90;
}

}
```

.

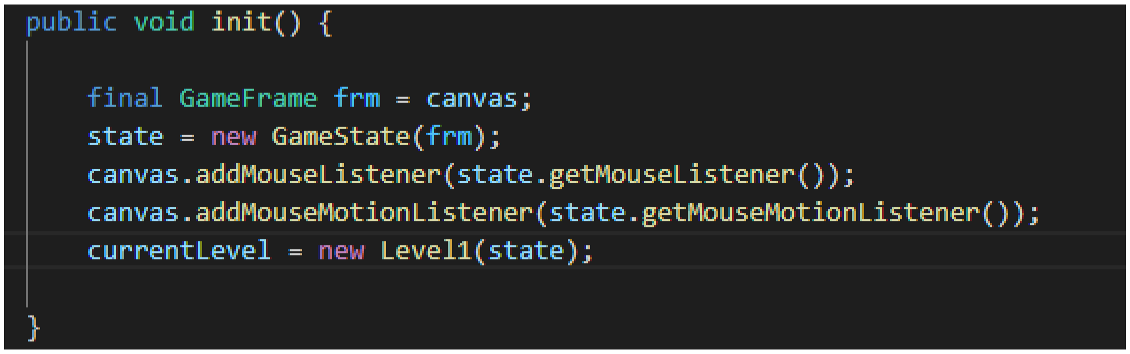
We also have the algorithm to find the closest zombie and make the plant attack



```
 */
private void findTheClosestZombie() {

int locX = 850;
for (int i = 0; i < zombies.size(); i++) {

    Zombie get = zombies.get(i);
    if (get.locX < locX && !get.zombieIsBurned) {
    firstZombie = get;
    locX = get.locX;
    }

}

}
```

V. ALGORITHM OF PLANT VS ZOMBIE GAME:

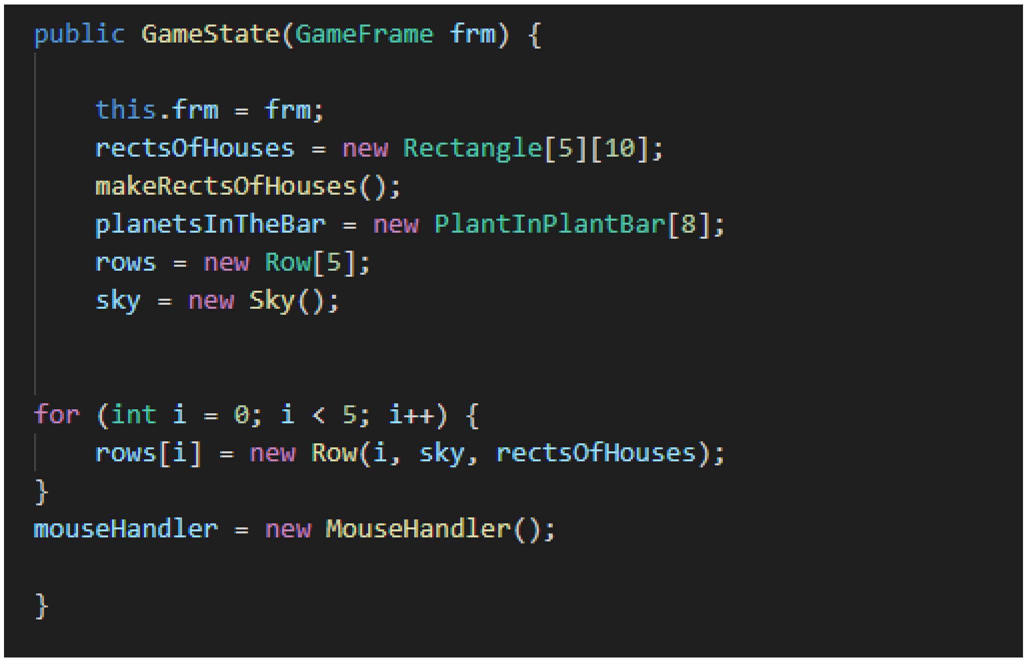
When game start the GameLoop class will initialized with init() void function, this function will draw GUI from gameFrame class, element of the game from the GameState class, mouse handle and start game with level 1



```java
public void init() {

    final GameFrame frm = canvas;
    state = new GameState(frm);
    canvas.addMouseListener(state.getMouseListener());
    canvas.addMouseMotionListener(state.getMouseMotionListener());
    currentLevel = new Level1(state);

}
```

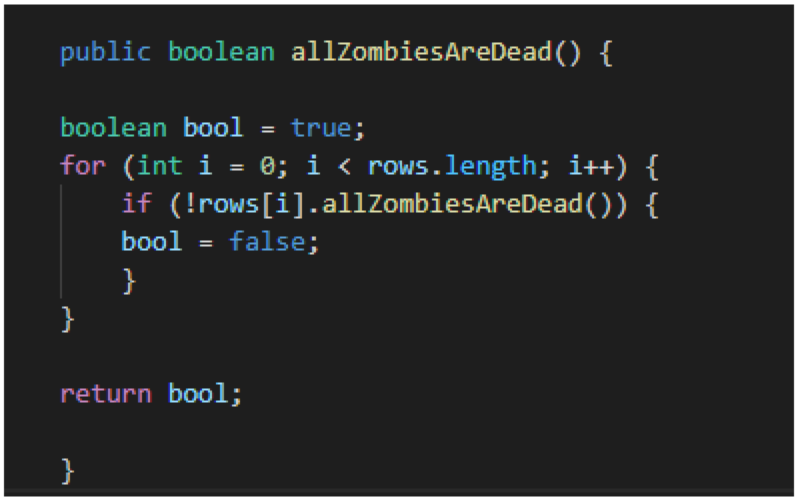
.

Then in gameState class we create square of the lane which is array 2-dimesion, initialize the PlantInPlantBar,..



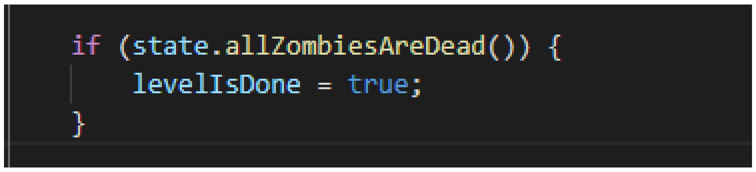
```java
public GameState(GameFrame frm) {

    this.frm = frm;
    rectsOfHouses = new Rectangle[5][10];
    makeRectsOfHouses();
    planetsInTheBar = new PlantInPlantBar[8];
    rows = new Row[5];
    sky = new Sky();


for (int i = 0; i < 5; i++) {
    rows[i] = new Row(i, sky, rectsOfHouses);
}
mouseHandler = new MouseHandler();

}
```

Then we play game with level 1, and the allZombiesAreDead() function will check all the zombie in the row. If not then we will win the game.



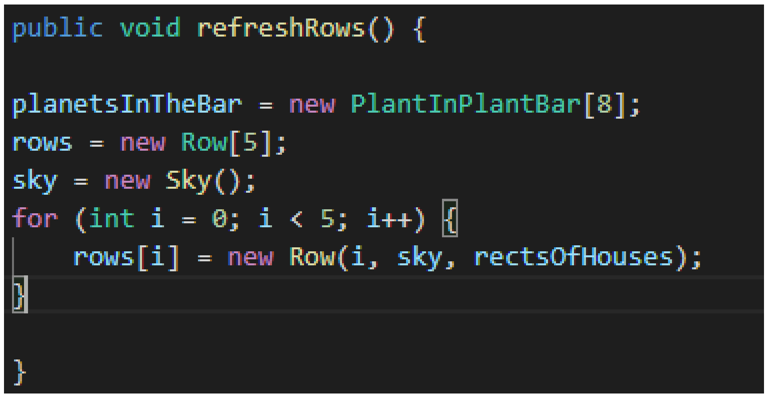
```java
public boolean allZombiesAreDead() {

boolean bool = true;
for (int i = 0; i < rows.length; i++) {
    if (!rows[i].allZombiesAreDead()) {
    bool = false;
    }
}

return bool;

}
```

//if-else to check the zombie in the row with allZombiesAreDead function



```java
if (state.allZombiesAreDead()) {
    levelIsDone = true;
}
```

But if you let the zombie go to your house, the state will refresh and you can play game again.



```java
public void refreshRows() {

planetsInTheBar = new PlantInPlantBar[8];
rows = new Row[5];
sky = new Sky();
for (int i = 0; i < 5; i++) {
    rows[i] = new Row(i, sky, rectsOfHouses);
}

}
```

VI/Conclusion

Plant vs Zombie game that was build by object – oriented method is more easier and locially than traditional – method. This shows cleary polymorphism, inheritance, encapsulation, data abstraction of OOP and the realationship between classes. Besides that, learning more knownledge out of the limited of this course is one of the important things to do while performing this project.