PROJECT OOP

GROUP MEMBERs

TRẦN ĐĂNG NHẤT
ITITIU22115

ĐỖ PHÚ THÀNH
ITITWE22129

LÊ ĐOÀN CƯỜNG THỊNH
ITITIU22151

ĐẶNG NGUYỄN NHẬT VY
ITITDK22102

PHAN VĂN TÀI ANH
ITITIU22011

TABLE OF CONTENTS

1

INTRODUCTION

Original version

iOS

Project version

PLANTS VS. ZOMBIES™

- Easier to understand, play, and represent the mechanism
- To serve the aim of education

PROJECT VERSION

- Auto spawn Sun and the sun point counter
- The buying mechanics using sun point
- The game is won when the 20th zombie is defeated

- Automatic spawn zombies
- The game is lost when a zombie reaches our house

PLANTS VS. ZOMBIES™

Project Timeline

Week 1

- Requirement Analysis and Task Assignment
- Develop Core Components
- Review and Debugging

Week 2

- Complete Feature Development
- Document Code and Report
- Slide Design

Week 3

- Integrate Components
- Prepare Report
- Polish Slides

Week 4

- Final Testing
- Final Review
- Presentation Rehearsal

2

Summary

Project Goals

01

To aimed to develop a simplified version

02

To focus on core game mechanics, efficient coding practices, and educational enhancements

Development Approach

Programming Techniques

- Object-Oriented Programming
- Event-Driven Programming

Tools and Resources

- Development Tools


- Media Resources



Game Components



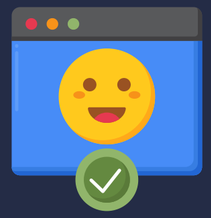





Achievements

Developed a fully functional game with modular and maintainable code

Created a user-friendly interface and engaging gameplay



Documented the development process comprehensively for educational purposes

3

DESIGN AND TECHNIQUE

3, System Design

01

Define conponents in the game:

- Plants
- Zombies
- Projectile, buying
- Game Window

02

Buiding The UML contains:

- class diagrams
- links between class diagrams
- UI Design

03

Building packages which contain classes present the mechanism of components

04

Checking and fixing bugs

3, System Design: Ideas

The group can be create:

- Plants: Bought by using sun, ability is unique:

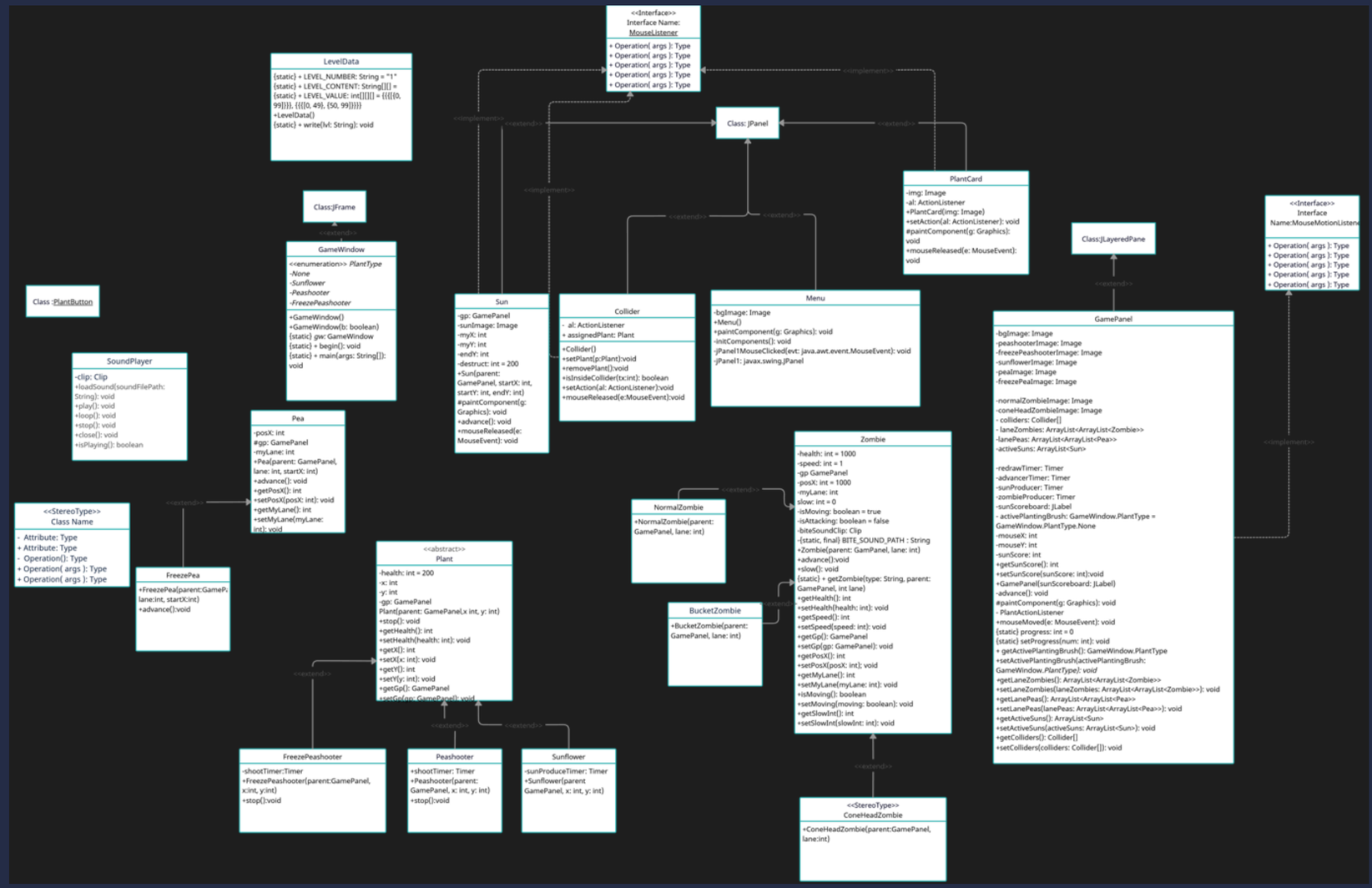
-- Sunflower: Preoduce flower for each 5 seconds.
-- Peashooter: shoot projectile (Pea) for each 4 seconds.
-- Freeze Peashooter: shoot projectile (Freeze Pea) with slow effect for each 4 seconds.
==> The general class Plant (HP, position, gp) with the extends class for each type of plant.

- Zombie: Random, different in the HP and appearance

-- Normal: HP = 1000
-- ConeHead: HP = 3000
==> The general class Zombie (HP, position, attack point, gp) with the extends class for each type of zombie

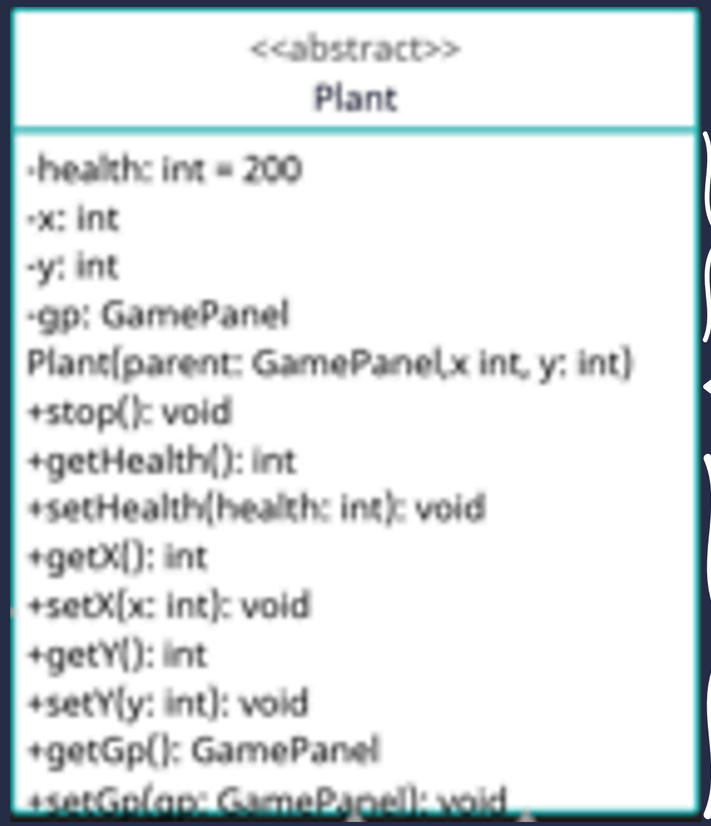
- Menu: Contains Start button
- Window: Contains background, and provide environment for objects

3, System Design: UML



3, System Design: Working

- Plants



- Attributes: Using in the methods as properties of the Plants [Health, Position, Gamepanel]
- Constructor: Initialize the value of attributes
- Encapsulators: Allow to access and modify the attributes

3, System Design: Working

- Plants: Attribute & Constructor

```
private int health = 200;
private int x;
private int y;
private GamePanel gp;
```

- Health: initialized to 200, presumably representing the plant's health
- x, y: coordinates of the plant in the game panel
- gp: represents the context or container in which the plant exists.

```
public Plant(GamePanel var1, int var2, int var3) {
    this.x = var2;
    this.y = var3;
    this.gp = var1;
}
```

- Assign to var2 & var3
- Assign to var1

3, System Design: Working

- Plants: Encapsulator

```
public int getHealth() {
    return this.health;
}

public void setHealth(int var1) {
    this.health = var1;
}

public int getX() {
    return this.x;
}

public void setX(int var1) {
    this.x = var1;
}

public int getY() {
    return this.y;
}

public void setY(int var1) {
    this.y = var1;
}

public GamePanel getGp() {
    return this.gp;
}

public void setGp(GamePanel var1) {
    this.gp = var1;
}
```

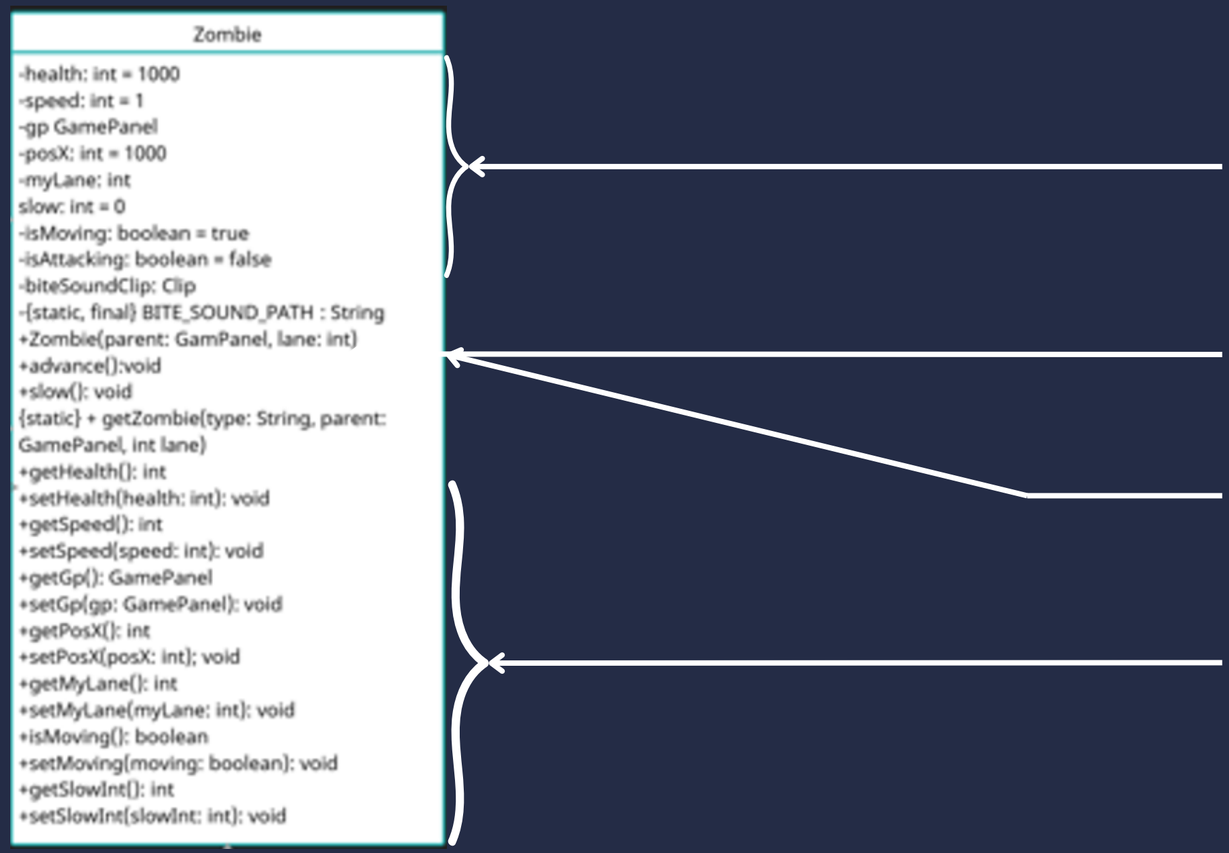
- Access & Modify Health
- Access & Modify X-coo
- Access & Modify Y-coo
- Access & Modify gp

- Private attribute

```
private int health = 200;
private int x;
private int y;
private GamePanel gp;
```

3, System Design: Working

- Zombies



- Attributes: Using in the methods as properties of Zombie [Health, X position, current lane, Gamepanel]
- Constructor: Initialize the value of attributes
- Methods advanced(): present the mechanism of zombie: moving, attacking, slowed effect
- Encapsulators: Allow to access and modify the attributes

3, System Design: Working

- Zombies: Attribute & Constructor

- Speed of zombie :posX -= speed when Z is moving). It's set by 1
- Health of zombie set by 1000
- isMoving: moving state (move = true, stop = false). It's set by true

```
private int health = 1000;
private int speed = 1;
private GamePanel gp;
private int posX = 1000;
private int myLane;
private boolean isMoving = true;
int slowInt = 0;
```

- gp: represents the context or container in which the zombie exists.
- posX: x coordinate
- myLane: y coordinate, presents the current lane

```
public Zombie(GamePanel var1, int var2) {
    this.gp = var1;
    this.myLane = var2;
}
```

3, System Design: Working

- Zombies: Moving & Collision checking & stop

```
if (this.isMoving) {
    boolean var1 = false;
    Collider var2 = null;

    for(int var3 = this.myLane * 9; var3 < (this.myLane + 1) * 9; ++var3) {
        if (this.gp.getColliders()[var3].assignedPlant != null && this.gp.getColliders()[var3].isInsideCollider(this.posX)) {
            var1 = true;
            var2 = this.gp.getColliders()[var3];
        }
    }
```

- boolean var1 act as a flag to detect the plants
- Create 9 cells of collider each lane (the Window has size 5 lanes x9)
- Check: Plant exist in the Collider
- Check: Zombie X position is inside the Collider
- Action: Zombie interact with plant

- Zombie reach player's house (posX =0)
- => Stop moving
- => show Notification
- => Remove old Window
- => Start new Window

```
if (this.posX < 0) {
    this.isMoving = false;
    JOptionPane.showMessageDialog(this.gp, "ZOMBIES ATE YOUR BRAIN !\nStarting the level again");
    GameWindow.gw.dispose();
    GameWindow.gw = new GameWindow();
}
```

3, System Design: Working

- Zombies: Advanced method

```
if (!var1) {
   if (this.slowInt > 0) {
      if (this.slowInt % 2 == 0) {
         --this.posX;
      }

      --this.slowInt;
   } else {
      --this.posX;
   }
} else {
   var2.assignedPlant.setHealth(var2.assignedPlant.getHealth() - 10);
   if (var2.assignedPlant.getHealth() < 0) {
      var2.removePlant();
   }
}
```

- CASE1: No Plants in interaction range + Attacted by slow projectile:
- => zombie move from right to left (--posX)
- => Every 2 frame slowInt decrease by 1 frame

- CASE2: No plant in interaction range + no slowed
- => zombie move from right to left (--posX)

- CASE 3: Plants in interaction range + attacking plant:
- => PlantHealth = PlantHealth - 10

- CASE 4: Plant health become 0:
- => remove plant from game panel.

```
public static Zombie getZombie(String var0, GamePanel var1, int var2) {
   Object var3 = new Zombie(var1, var2);
   switch (var0) {
      case "NormalZombie":
         var3 = new NormalZombie(var1, var2);
         break;
      case "ConeHeadZombie":
         var3 = new ConeHeadZombie(var1, var2);
   }

   return (Zombie)var3;
}
```

- Spawn Zombie: Call the extends classes

3, System Design: Working

- Zombies: Encapsulators allow to access and modify the data of attributes

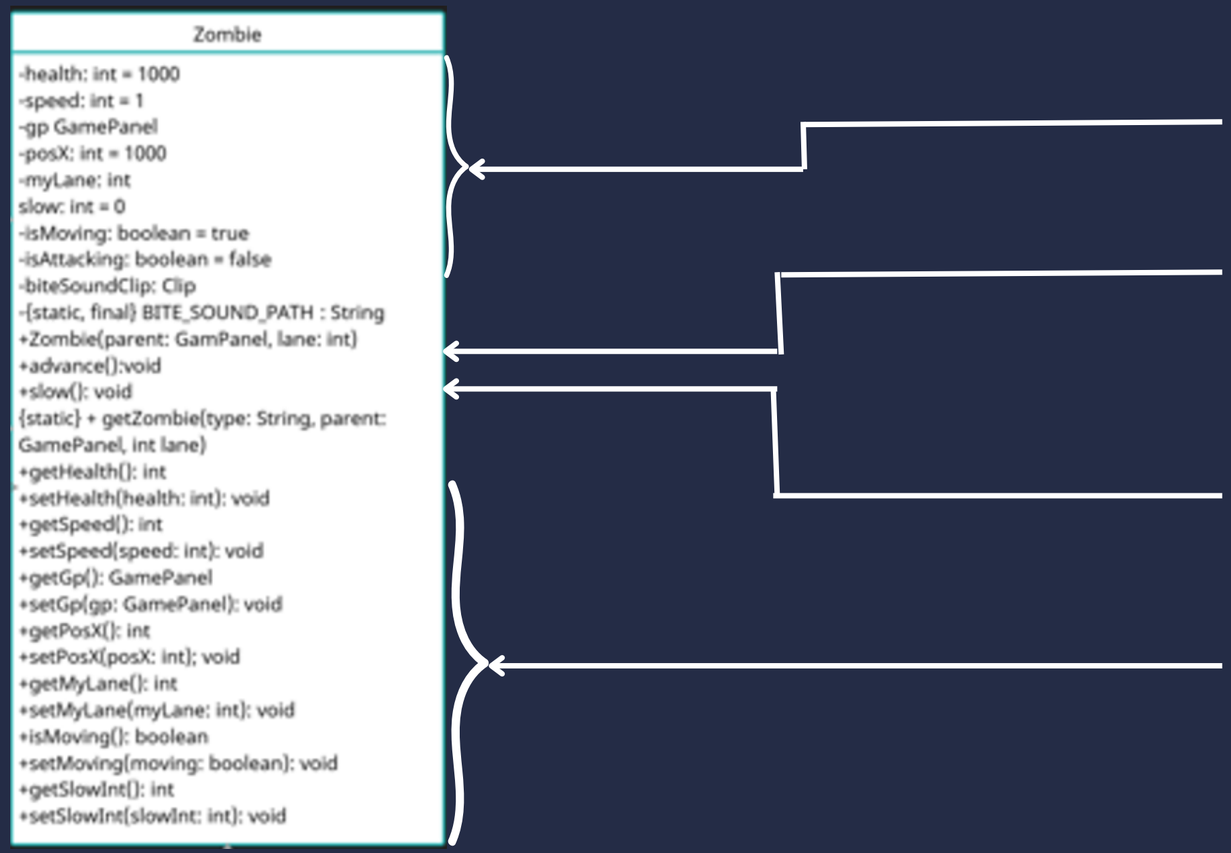
```java
public int getHealth() {
    return this.health;
}

public void setHealth(int var1) {
    this.health = var1;
}

public int getSpeed() {
    return this.speed;
}

public void setSpeed(int var1) {
    this.speed = var1;
}

public GamePanel getGp() {
    return this.gp;
}

public void setGp(GamePanel var1) {
    this.gp = var1;
}

public int getPosX() {
    return this.posX;
}

public void setPosX(int var1) {
    this.posX = var1;
}
```

```java
private int health = 1000;
private int speed = 1;
private GamePanel gp;
private int posX = 1000;
private int myLane;
private boolean isMoving = true;
int slowInt = 0;
```

```java
public int getMyLane() {
    return this.myLane;
}

public void setMyLane(int var1) {
    this.myLane = var1;
}

public boolean isMoving() {
    return this.isMoving;
}

public void setMoving(boolean var1) {
    this.isMoving = var1;
}

public int getSlowInt() {
    return this.slowInt;
}

public void setSlowInt(int var1) {
    this.slowInt = var1;
}
```

3, System Design: Working

- Zombies



- Attributes: Using in the methods as properties of Zombie [Health, X position, current lane, Gamepanel]
- Constructor: Initialize the value of attributes
- Methods: present the mechanism of zombie: moving, attacking, slowed effect
- Encapsulators: Allow to access and modify the attributes

3, System Design: Working

- Zombies’s extends classes

- Name: NormalZombie must be the same to “NormalZombie” in getZombie() method to be call in this spawn class.
- Constructor:
- => inherite the lane and gp from Zombie class.
- => Set the damage by 10 and health by 200. => unique properties.

```
public class NormalZombie extends Zombie {

    public NormalZombie(GamePanel parent, int lane) {
        super(parent, lane);
        setDmg(10);
        setHealth(200);
    }
}
```

- Name: ConeHeadZombie must be the same to “ConeHeadZombie” in getZombie() method to be call in this spawn class.
- Constructor:
- => inherite the lane and gp from Zombie class.
- => Set the damage by 15 and health by 370. => unique properties.

```
public class ConeHeadZombie extends Zombie {

    public ConeHeadZombie(GamePanel parent, int lane) {
        super(parent, lane);
        setDmg(15);
        setHealth(370);
    }
}
```

3, System Design: Working

Sun

Pea

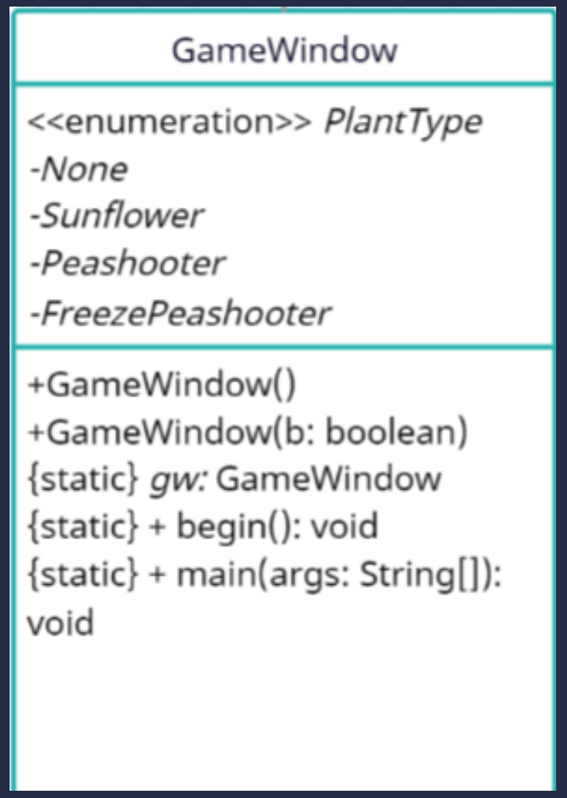
Freeze Pea

3, System Design: Working

Cell grip

3, System Design: Working

- Window



- Attributes: (Enumeration) listing the types of plants
- Constructors: Initializing game window with main interface: PlantCard, GamePanel, sound
- Methods: Creating the new game window, staring the program

3, System Design: Working

- Window: Attributes

```
public class GameWindow extends JFrame {

    enum PlantType {
        None,
        Sunflower,
        Peashooter,
        FreezePeashooter
    }

    static SoundPlayer soundPlayerMenu;
    static SoundPlayer soundPlayerGame;
// Static sound player for game music

    static GameWindow gw;
```

- Listing the plants in the game
- Managing the sound
- Representing for main game window

3, System Design: Working

- Window: Constructors

```
public GameWindow() {
    setSize(width:1012, height:785);
    setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    setLayout(manager:null);

    JLabel sun = new JLabel(text:"SUN");
    sun.setLocation(x:37, y:80);
    sun.setSize(width:60, height:20);

    GamePanel gp = new GamePanel(sun);
    gp.setLocation(x:0, y:0);
    getLayeredPane().add(gp, new Integer(value:0));

    setResizable(resizable:false);
    setVisible(b:true);
```

- Main interface configuration: setting the size of the window (1012 x 785), closing the window, using a free layout
- Displaying “SUN”, setting location (37, 80), setting size (60 x 20)
- Displaying main area
- Not allow resizing the window, showing the window

3, System Design: Working

- Window: Constructors

- Stopping the sound

```
// Stop menu music if playing
if (soundPlayerMenu != null) {
    soundPlayerMenu.stop();
}

// Initialize and play game music
soundPlayerGame = new SoundPlayer();
soundPlayerGame.loadSound(soundFilePath:"sound/main-sound.wav");
soundPlayerGame.loop(); // Loop the music
```

- Playing and replaying the sound

➡ Control music when switching from menu to game

3, System Design: Working

- Window: Constructors

```
PlantCard sunflower = new PlantCard(new
ImageIcon(this.getClass().getResource("images/cards/card_sunflower.png")).getImage()
;
      sunflower.setLocation(110, 8);
      sunflower.setAction((ActionEvent e) -> {
         gp.setActivePlantingBrush(PlantType.Sunflower);
      });
      getLayeredPane().add(sunflower, new Integer(3));

      PlantCard peashooter = new PlantCard(new
ImageIcon(this.getClass().getResource("images/cards/card_peashooter.png")).getImage
());
      peashooter.setLocation(175, 8);
      peashooter.setAction((ActionEvent e) -> {
         gp.setActivePlantingBrush(PlantType.Peashooter);
      });
      getLayeredPane().add(peashooter, new Integer(3));

      PlantCard freezepeashooter = new PlantCard(new
ImageIcon(this.getClass().getResource("images/cards/card_freezepeashooter.png")).getI
mage());
      freezepeashooter.setLocation(240, 8);
      freezepeashooter.setAction((ActionEvent e) -> {
         gp.setActivePlantingBrush(PlantType.FreezePeashooter);
      });
      getLayeredPane().add(freezepeashooter, new Integer(3));

      getLayeredPane().add(sun, new Integer(2));
      setResizable(false);
      setVisible(true);
   }
```

- representing the sunflower card in the interface
- downloading the image of the card
- representing the peashooter card in the interface
- setting action of the card
- representing the freezepeashooter card in the interface

➡ Adding plant cards

3, System Design: Working

- Window: Constructors

```
public GameWindow(boolean isMenu) {
    if (isMenu) {
        Menu menu = new Menu();
        menu.setLocation(x:0, y:0);
        setSize(width:1012, height:785);
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        getLayeredPane().add(menu, new Integer(value:0));
        menu.repaint();
        setResizable(resizable:false);
        setVisible(b:true);
    }
}
```

- creating an window in menu mode, and having components of menu

- Setting the size of window (1012 x 785), closing the window

➡ Initializing a game interface

3, System Design: Working

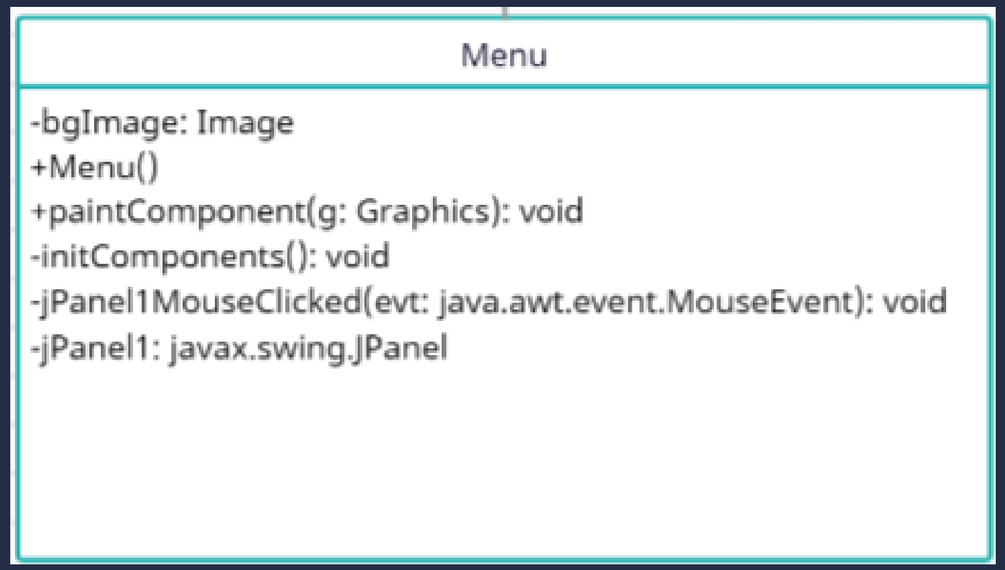
- Window: Constructors

```
    public static void begin() {
        gw.dispose();
        gw = new GameWindow();
    }

    Run | Debug
    public static void main(String[] args) {
        // Initialize menu music
        soundPlayerMenu = new SoundPlayer();
        soundPlayerMenu.loadSound(soundFilePath:"sound/background-music.wav");
        soundPlayerMenu.loop(); // Loop the music

        // Launch the menu
        gw = new GameWindow(isMenu:true);
    }
}
```

- closing the current interface and creating the game window to start the game
- Creating and playing music, displaying the interface

3, System Design: Working

- Menu



- Attributes: Storing the background image
- Constructors: Initializing menu object
- Methods: Drawing the graphic components, initializing GUI components, handling click-mouse events

3, System Design: Working

- Menu: Attributes

```
public class Menu extends JPanel {
    private Image bgImage;
    private SoundPlayer soundPlayer;
    private JPanel jPanel1;
```

- Storing background image of Menu
- Controlling the sound: playing, downloading, stopping and replaying
- Serving for interaction: click-mouse event

3, System Design: Working

- Menu: Constructors

```
public Menu() {
    this.initComponents();
    this.setSize(1012, 785);
    this.bgImage = (new ImageIcon(this.getClass().getResource("images/menu.jpg"))).getImage();
    this.soundPlayer = new SoundPlayer();
    this.soundPlayer.loadSound("sound/background-music.wav");
    this.soundPlayer.loop();
}
```

- Configuring the interface
- Setting up the size: 1012 x 785
- Downloading the image
- Downloading, playing, and replaying the sound

3, System Design: Working

- Menu: Methods

```
protected void paintComponent(Graphics var1) {
    super.paintComponent(var1);
    var1.drawImage(this.bgImage, 0, 0, (ImageObserver)null);
}
```

- Drawing the background image of menu

```
private void jPanel1MouseClicked(MouseEvent var1) {
    this.soundPlayer.stop();
    GameWindow.begin();
}
```

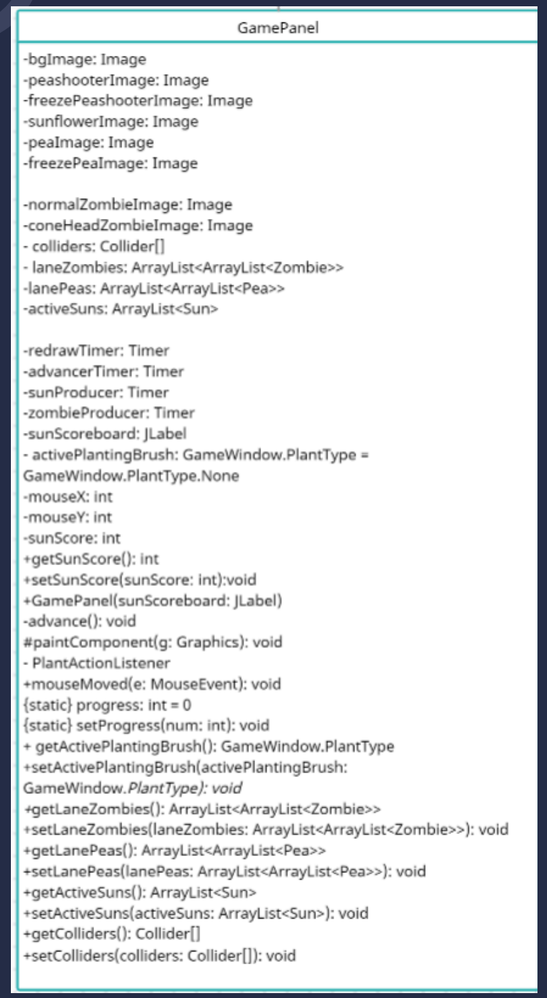
- soundPlayer.stop(): stopping the sound
- GameWindow.begin(): starting the game

```
private void initComponents() {
    this.jPanel1 = new JPanel();
    this.setPreferredSize(new Dimension(1012, 785));
    this.jPanel1.setOpaque(false);
    this.jPanel1.addMouseListener(new Menu$1(this));
    GroupLayout var1 = new GroupLayout(this.jPanel1);
    this.jPanel1.setLayout(var1);
    var1.setHorizontalGroup(var1.createParallelGroup(Alignment.LEADING).addGap(0, 387, 32767));
    var1.setVerticalGroup(var1.createParallelGroup(Alignment.LEADING).addGap(0, 116, 32767));
    GroupLayout var2 = new GroupLayout(this);
    this.setLayout(var2);
    var2.setHorizontalGroup(var2.createParallelGroup(Alignment.LEADING).addGroup(Alignment.TRAILING,
var2.createSequentialGroup().addContainerGap(523, 32767).addComponent(this.jPanel1, -2, -1, -2).addGap(102,
102, 102)));

var2.setVerticalGroup(var2.createParallelGroup(Alignment.LEADING).addGroup(var2.createSequentialGroup().a
ddGap(122, 122, 122).addComponent(this.jPanel1, -2, -1, -2).addContainerGap(547, 32767)));
}
```

- Configure the main interface and jPanel1
- Using “GroupLayout” to arrange components
- Adding a click-mouse event

3, System Design: Working

- Panel



- Attributes: Image, logic & data, timers, displaying score, the type of plant chosen for planting, coordinate of mouse
- Constructors: displaying score
- Methods: taking & updating score, updating logic, drawing, handling mouse events, updating & following progress
- Encapsulators: getting & updating lists of plants, zombies, peas

3, System Design: Working

- Panel: Attributes

```
public class GamePanel extends JLayeredPane implements MouseMotionListener {
    private Image bgImage;
    private Image peashooterImage;
    private Image freezePeashooterImage;
    private Image sunflowerImage;
    private Image peaImage;
    private Image freezePeaImage;
    private Image normalZombieImage;
    private Image coneHeadZombieImage;
    private Image BucketheadZombieImage;
    private Image normalZombieEatImage;
    private Image coneHeadZombieEatImage;
    private Image BucketheadZombieEatImage;
    private Collider[] colliders;
    private ArrayList<ArrayList<Zombie>> laneZombies;
    private ArrayList<ArrayList<Pea>> lanePeas;
    private ArrayList<Sun> activeSuns;
    private Timer redrawTimer;
    private Timer advancerTimer;
    private Timer sunProducer;
    private Timer zombieProducer;
    private JLabel sunScoreboard;
    private GameWindow$PlantType activePlantingBrush;
    private int mouseX;
    private int mouseY;
    private int sunScore;
    private static final String BITE_SOUND_PATH = "sound/bite.wav";
    static int progress = 0;
```

- Image
- Interaction area
- Lists of zombies, plants, peas
- Timer of actions
- Displaying sun score
- Current plants planted
- Mouse’s position
- Sun score
- Link to “Bite” sound
- Following progress

3, System Design: Working

- Panel: Constructors

```
public GamePanel(JLabel var1) {
    this.activePlantingBrush = PlantType.None;
    this.setSize(1000, 752);
    this.setLayout((LayoutManager)null);
    this.addMouseMotionListener(this);
    this.sunScoreboard = var1;
    this.setSunScore(50);
```

- Displaying screen size (1000 x 725)
- References to sunScoreboard
- Initial sun score
- Downloading image

```
this.bgImage = (new ImageIcon(this.getClass().getResource("images/mainBG.png"))).getImage();
this.peashooterImage = (new ImageIcon(this.getClass().getResource("images/plants/peashooter.gif"))).getImage();
this.freezePeashooterImage = (new ImageIcon(this.getClass().getResource("images/plants/freezepeashooter.gif"))).getImage();
this.sunflowerImage = (new ImageIcon(this.getClass().getResource("images/plants/sunflower.gif"))).getImage();
this.peaImage = (new ImageIcon(this.getClass().getResource("images/pea.png"))).getImage();
this.freezePeaImage = (new ImageIcon(this.getClass().getResource("images/freezepea.png"))).getImage();
this.normalZombieImage = (new ImageIcon(this.getClass().getResource("images/zombies/zombievip.gif"))).getImage();
this.normalZombieEatImage = (new ImageIcon(this.getClass().getResource("images/zombies/ZombieVipEat.gif"))).getImage();
this.BucketheadZombieImage = (new ImageIcon(this.getClass().getResource("images/zombies/BucketheadZombie.gif"))).getImage();
this.BucketheadZombieEatImage = (new ImageIcon(this.getClass().getResource("images/zombies/BucketheadZombieAttackp.gif"))).getImage();
this.coneHeadZombieImage = (new ImageIcon(this.getClass().getResource("images/zombies/ConeheadZombie.gif"))).getImage();
this.coneHeadZombieEatImage = (new ImageIcon(this.getClass().getResource("images/zombies/ConeheadZombieAttack.gif"))).getImage();
```

3, System Design: Working

- Panel: Constructors

```
this.laneZombies = new ArrayList();
this.laneZombies.add(new ArrayList());
this.laneZombies.add(new ArrayList());
this.laneZombies.add(new ArrayList());
this.laneZombies.add(new ArrayList());
this.laneZombies.add(new ArrayList());
this.lanePeas = new ArrayList();
this.lanePeas.add(new ArrayList());
this.lanePeas.add(new ArrayList());
this.lanePeas.add(new ArrayList());
this.lanePeas.add(new ArrayList());
this.lanePeas.add(new ArrayList());
this.activeSuns = new ArrayList();
```

- Creating lists of zombies, peas, suns

- Representing cells (5 rows x 9 columns =45 cells

```
this.colliders = new Collider[45];

for(int var2 = 0; var2 < 45; ++var2) {
  Collider var3 = new Collider();
  var3.setLocation(44 + var2 % 9 * 100, 109 + var2 / 9 * 120);
  var3.setAction(new GamePanel$PlantActionListener(this, var2 % 9, var2 / 9));
  this.colliders[var2] = var3;
  this.add(var3, new Integer(0));
}
```

- Formula: x = 44 + (column index × 100) & y = 109 + (row index × 120)

3, System Design: Working

- Panel: Constructors

```
this.redrawTimer = new Timer(16, (var1x) -> {
         this.repaint();
      });
      this.redrawTimer.start();
      this.advancerTimer = new Timer(60, (var1x) -> {
         this.advance();
      });
      this.advancerTimer.start();
      this.sunProducer = new Timer(4000, (var1x) -> {
         Random var2 = new Random();
         Sun var3 = new Sun(this, var2.nextInt(800) + 100, 0,
var2.nextInt(300) + 200);
         this.activeSuns.add(var3);
         this.add(var3, new Integer(1));
      });
      this.sunProducer.start();
      this.zombieProducer = new Timer(7000, (var1x) -> {
         Random var2 = new Random();
         LevelData var3 = new LevelData();
         String[] var4 =
LevelData.LEVEL_CONTENT[Integer.parseInt(LevelData.LEVEL_NUMBER) - 1];
         int[][] var5 =
LevelData.LEVEL_VALUE[Integer.parseInt(LevelData.LEVEL_NUMBER) - 1];
         int var6 = var2.nextInt(5);
         int var7 = var2.nextInt(100);
         Zombie var8 = null;

         for(int var9 = 0; var9 < var5.length; ++var9) {
            if (var7 >= var5[var9][0] && var7 <= var5[var9][1]) {
               var8 = Zombie.getZombie(var4[var9], this, var6);
            }
         }
         ((ArrayList)this.laneZombies.get(var6)).add(var8);
      });
      this.zombieProducer.start();
   }
```

- Refreshing the screen
- Updating the game state
- The time that producing “sun”
- The time that producing zombies

3, System Design: Working

- Panel: Methods

```
private static final String BITE_SOUND_PATH = "sound/bite.wav";
public void playBiteSound() {
    try {
        File var1 = new File("sound/bite.wav");
        if (!var1.exists()) {
            System.err.println("Bite sound file not found: sound/bite.wav");
            return;
        }

        AudioInputStream var2 = AudioSystem.getAudioInputStream(var1);
        Clip var3 = AudioSystem.getClip();
        var3.open(var2);
        var3.start();
    } catch (Exception var4) {
        System.err.println("Error playing bite sound: " + var4.getMessage());
    }
}
```

- Readability and standardize file to save the path
- Checking the audio file & play sound
- References
- Checking
- Reading audio file
- Playing
- Displaying the error

3, System Design: Working

- Panel: Methods

```
private void advance() {
   int var1;
   for(var1 = 0; var1 < 5; ++var1) {
      Iterator var2 = ((ArrayList)this.laneZombies.get(var1)).iterator();

      while(var2.hasNext()) {
         Zombie var3 = (Zombie)var2.next();
         var3.advance();
      }

      for(int var4 = 0; var4 < ((ArrayList)this.lanePeas.get(var1)).size(); ++var4) {
         Pea var5 = (Pea)((ArrayList)this.lanePeas.get(var1)).get(var4);
         var5.advance();
      }
   }

   for(var1 = 0; var1 < this.activeSuns.size(); ++var1) {
      ((Sun)this.activeSuns.get(var1)).advance();
   }
}
```

- Zombies move to the left
- Movement of peas
- Updating the sun's falling state

- Drawing the entire interface for the component & element

```
protected void paintComponent(Graphics var1) {
   super.paintComponent(var1);
   var1.drawImage(this.bgImage, 0, 0, (ImageObserver)null);
```

- Cleaning
- Displaying background

3, System Design: Working

- Panel: Methods

➡ Drawing game plants

```
int var2;
for(var2 = 0; var2 < 45; ++var2) {
   Collider var3 = this.colliders[var2];
   if (var3.assignedPlant != null) {
      Plant var4 = var3.assignedPlant;
      if (var4 instanceof Peashooter) {
         var1.drawImage(this.peashooterImage, 60 + var2 % 9 * 100, 129 + var2 / 9 * 120, (ImageObserver)null);
      }

      if (var4 instanceof FreezePeashooter) {
         var1.drawImage(this.freezePeashooterImage, 60 + var2 % 9 * 100, 129 + var2 / 9 * 120, (ImageObserver)null);
      }

      if (var4 instanceof Sunflower) {
         var1.drawImage(this.sunflowerImage, 60 + var2 % 9 * 100, 129 + var2 / 9 * 120, (ImageObserver)null);
      }
   }
}
```

- Containing Collider object
- Column (located x)
- Rows (located y)
- Types of plants

3, System Design: Working

- Panel: Methods

➡ Drawing types of zombies

```
for(var2 = 0; var2 < 5; ++var2) {
   Iterator var5 = ((ArrayList)this.laneZombies.get(var2)).iterator();

   while(var5.hasNext()) {
      Zombie var7 = (Zombie)var5.next();
      if (var7 instanceof NormalZombie) {
         if (!var7.isAttacking()) {
            var1.drawImage(this.normalZombieImage, var7.getPosX(), 109 + var2 * 120, (ImageObserver)null);
         } else {
            var1.drawImage(this.normalZombieEatImage, var7.getPosX(), 109 + var2 * 120, (ImageObserver)null);
         }
      } else if (var7 instanceof ConeHeadZombie) {
         if (!var7.isAttacking()) {
            var1.drawImage(this.coneHeadZombieImage, var7.getPosX(), 109 + var2 * 120, (ImageObserver)null);
         } else {
            var1.drawImage(this.coneHeadZombieEatImage, var7.getPosX(), 109 + var2 * 120, (ImageObserver)null);
         }
      } else if (var7 instanceof BucketheadZombie) {
         if (!var7.isAttacking()) {
            var1.drawImage(this.BucketheadZombieImage, var7.getPosX(), 109 + var2 * 120, (ImageObserver)null);
         } else {
            var1.drawImage(this.BucketheadZombieEatImage, var7.getPosX(), 109 + var2 * 120, (ImageObserver)null);
         }
      }
   }
```

- 5 elements (list of zombies in lane)
- not attack, moving
- attack, eating
- Y: determine plants position
- X: current position of zombies
- Coordinates
- Types of zombies

3, System Design: Working

- Panel: Methods

➡ Drawing types of peas

```
for(int var6 = 0; var6 < ((ArrayList)this.lanePeas.get(var2)).size(); ++var6) {
    Pea var8 = (Pea)((ArrayList)this.lanePeas.get(var2)).get(var6);
    if (var8 instanceof FreezePea) {
        var1.drawImage(this.freezePeaImage, var8.getPosX(), 130 + var2 * 120, (ImageObserver)null);
    } else {
        var1.drawImage(this.peaImage, var8.getPosX(), 130 + var2 * 120, (ImageObserver)null);
    }
}
}
```

- List contains peas
- Types of peas
- Horizontal position (x)
- y: determine the lane

- Type of plants that selected

```
if (!"".equals(this.activePlantingBrush) && this.activePlantingBrush == PlantType.Sunflower) {
    var1.drawImage(this.sunflowerImage, this.mouseX, this.mouseY, (ImageObserver)null);
}
```

- displaying the corresponding image at the mouse position

3, System Design: Working

- Panel: Methods

➡ Managing the progress: Changing level or ending game

```
public static void setProgress(int var0) {
    progress += var0;
    System.out.println(progress);
    if (progress >= 200) {
        if ("1".equals(LevelData.LEVEL_NUMBER)) {
            JOptionPane.showMessageDialog((Component)null, "LEVEL_CONTENT Completed !!!\nStarting next LEVEL_CONTENT");
            GameWindow.gw.dispose();
            LevelData.write("2");
            GameWindow.gw = new GameWindow();
        } else {
            JOptionPane.showMessageDialog((Component)null, "LEVEL_CONTENT Completed !!!\nMore Levels will come soon !!!\nResetting data");
            LevelData.write("1");
            System.exit(0);
        }

        progress = 0;
    }

}
```

- Increasing the progress
- Current level: 1
- Displaying notification
- Closing window
- Writing new level
- Initializing the new game window
- Displaying notification
- Exiting

3, System Design: Working

- Panel: Encapsulators

➡ Providing access to important data structures

```
public GameWindow$PlantType getActivePlantingBrush() {
   return this.activePlantingBrush;
}

public void setActivePlantingBrush(GameWindow$PlantType var1) {
   this.activePlantingBrush = var1;
}

public ArrayList<ArrayList<Zombie>> getLaneZombies() {
   return this.laneZombies;
}

public void setLaneZombies(ArrayList<ArrayList<Zombie>> var1) {
   this.laneZombies = var1;
}

public ArrayList<ArrayList<Pea>> getLanePeas() {
   return this.lanePeas;
}

public void setLanePeas(ArrayList<ArrayList<Pea>> var1) {
   this.lanePeas = var1;
}

public ArrayList<Sun> getActiveSuns() {
   return this.activeSuns;
}

public void setActiveSuns(ArrayList<Sun> var1) {
   this.activeSuns = var1;
}

public Collider[] getColliders() {
   return this.colliders;
}

public void setColliders(Collider[] var1) {
   this.colliders = var1;
}
}
```

```
private Image bgImage;
private Image peashooterImage;
private Image freezePeashooterImage;
private Image sunflowerImage;
private Image peaImage;
private Image freezePeaImage;
private Image normalZombieImage;
private Image coneHeadZombieImage;
private Image BucketheadZombieImage;
private Image normalZombieEatImage;
private Image coneHeadZombieEatImage;
private Image BucketheadZombieEatImage;
private Collider[] colliders;
private ArrayList<ArrayList<Zombie>> laneZombies;
private ArrayList<ArrayList<Pea>> lanePeas;
private ArrayList<Sun> activeSuns;
private Timer redrawTimer;
private Timer advancerTimer;
private Timer sunProducer;
private Timer zombieProducer;
private JLabel sunScoreboard;
private GameWindow$PlantType activePlantingBrush;
private int mouseX;
private int mouseY;
private int sunScore;
private static final String BITE_SOUND_PATH = "sound/bite.wav";
static int progress = 0;
```

```
public void mouseDragged(MouseEvent var1) {
}

public void mouseMoved(MouseEvent var1) {
   this.mouseX = var1.getX();
   this.mouseY = var1.getY();
}
```

```
public int getSunScore() {
   return this.sunScore;
}

public void setSunScore(int var1) {
   this.sunScore = var1;
   this.sunScoreboard.setText(String.valueOf(var1));
}
```

CONCLUSION

creativity

inspiration

accessibility

REFERENCES

- PopCap Games. (2009). Plants vs Zombies
- How to play PvZ
- The success of PvZ
- https://www.youtube.com/@BroCodez/
- https://github.com/leaser019/
- https://www.ea.com/ea-studios/popcap/plants-vs-zombies

Q&A AND DEMO

Thank You