



มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

รายงาน

Hunt Survival

Java OOP

จัดทำโดย

นายกฤตภาส ดวงครุฑ 6504062630014

เสนอ

ผู้ช่วยศาสตราจารย์ สกิต ประสมพันธ์

วิชา Object-Oriented-Programming

ภาคเรียนที่ 1 ปีการศึกษา 2568

ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ คณะวิทยาศาสตร์ประยุกต์

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

บทนำ

1.1 ชื่อ Project

2D Battle Survival Game

1.2 ที่มาและความสำคัญของโครงการ

ในปัจจุบันการเรียนรู้เกี่ยวกับการเขียนโปรแกรมเชิงวัตถุ (Object-Oriented Programming: OOP) ได้รับความนิยม เนื่องจากช่วยให้สามารถพัฒนาโปรแกรมที่ซับซ้อนได้ง่ายขึ้น โครงการนี้จึงมีจุดมุ่งหมายเพื่อสร้างเกม 2 มิติที่นำหลักการ OOP มาประยุกต์ใช้ เช่น

Encapsulation, Inheritance, Polymorphism และ Composition เป็นต้น

1.3 ประเภทของโครงการ

โครงการประเภท Game Application (Java Desktop Application)
โดยใช้ภาษา Java และ Java Swing สำหรับสร้างกราฟิกและการโต้ตอบกับผู้ใช้

1.4 ประโยชน์ของโครงการ

- ฝึกการเขียนโปรแกรมเชิงวัตถุ (OOP) อย่างเป็นระบบ
- เรียนรู้หลักการทำงานของ Game Loop และ Thread
- เข้าใจหลักการตรวจสอบการชน (Collision Detection)
- สามารถต่อยอดเกมหรือโปรแกรมอื่นได้ในอนาคต

1.5 ขอบเขตของโครงการ

- พัฒนาเกม 2 มิติด้วย Java Swing
- ผู้เล่นสามารถเคลื่อนที่ ยิงกระสุน เก็บค่าประสบการณ์ (EXP)
- ศัตรูเกิดเพิ่มตามเวลา (Spawn System)
- มีระบบ HP Bar, EXP Bar, Score, Timer และ Game Over
- ใช้ภาพ PNG สำหรับตัวละครและพื้นหลัง

บทที่ 2 การพัฒนาโปรแกรม

2.1 เนื้อเรื่องย่อหรือวิธีการเล่น

ผู้เล่นจะควบคุมตัวละครหลักเพื่อเอาชีวิตรอดจากศัตรูที่เกิดขึ้นอย่างต่อเนื่อง สามารถเคลื่อนที่ด้วยปุ่มลูกศรและกดยิงด้วยปุ่ม Spacebar เมื่อศัตรูถูกยิงจะได้รับ EXP และคะแนน เมื่อ HP หมดเกมจะจบลง

2.2 ไอเดียและแรงบันดาลใจ

เกมนี้ได้แรงบันดาลใจจากเกมแนวเอาชีวิตรอด เช่น “Vampire Survivors” ซึ่งผู้เล่นต้องอยู่ให้นานที่สุดและอัปเลเวลต่อเนื่อง

2.3 รูปแบบโปรแกรม

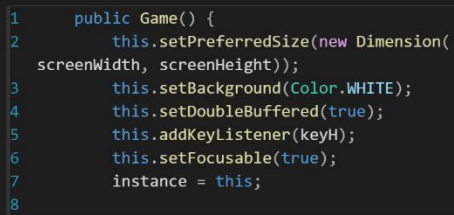
เป็นโปรแกรมประเภท Desktop Application (Java JFrame + JPanel)

โดยใช้คลาส Game เป็นศูนย์กลางในการจัดการทุกส่วนของเกม

2.4 การใช้หลัก OOP ในโปรแกรม

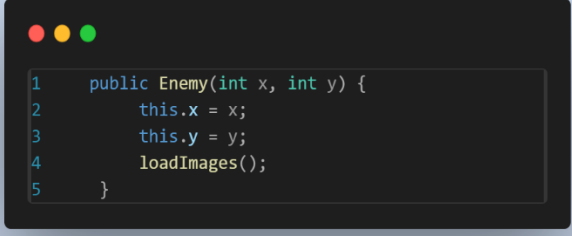
1. Constructor

ทุกคลาสมี Constructor ใช้สร้างค่าเริ่มต้น เช่น

A screenshot of a code editor window with a dark background and light blue text. The code is for a Java class named Game. It shows a constructor public Game() with eight lines of code: setting preferred size, background color to WHITE, double buffering to true, adding a key listener, setting focusable to true, and assigning instance to this. The code is numbered 1 through 8 on the left.

```
1 public Game() {  
2     this.setPreferredSize(new Dimension(  
3         screenWidth, screenHeight));  
4     this.setBackground(Color.WHITE);  
5     this.setDoubleBuffered(true);  
6     this.addKeyListener(keyH);  
7     this.setFocusable(true);  
8     instance = this;  
}
```

Game.java

A screenshot of a code editor window with a dark background and light blue text. The code is for a Java class named Enemy. It shows a constructor public Enemy(int x, int y) with five lines of code: assigning x and y to this.x and this.y, calling loadImages(), and closing the constructor block. The code is numbered 1 through 5 on the left.

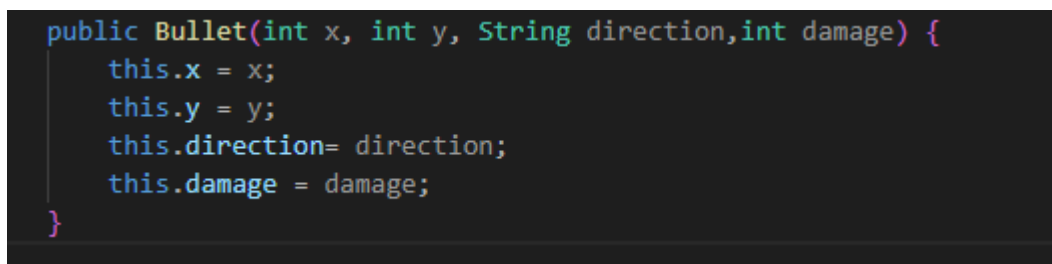
```
1 public Enemy(int x, int y) {  
2     this.x = x;  
3     this.y = y;  
4     loadImages();  
5 }
```

Enemy.java



```
1 public Player(KeyHandle keyH) {
2     this.keyH = keyH;
3     this.x = 400;
4     this.y = 300;
5     getPlayerImage();
6 }
7
```

Player.java

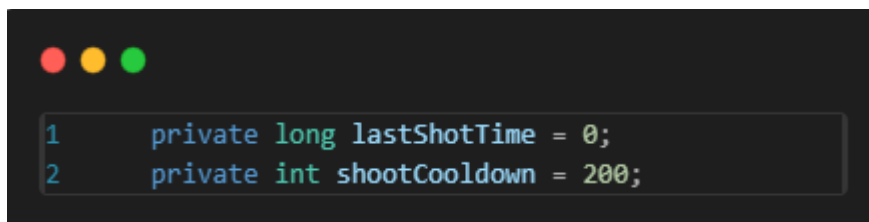


```
public Bullet(int x, int y, String direction,int damage) {
    this.x = x;
    this.y = y;
    this.direction= direction;
    this.damage = damage;
}
```

Bullet.java

2. Encapsulation (การห่อหุ้มข้อมูล)

ใช้ private หรือ protected เพื่อป้องกันการเข้าถึงตัวแปรโดยตรง เช่น




```
1 private long lastShotTime = 0;
2 private int shootCooldown = 200;
```

Bullet.java

3. Composition

ภายใน Game มีการสร้างออบเจกต์ของ Player, Enemy, Bullet, TileManager



```
1 KeyHandle keyH = new KeyHandle();
2 Thread gameThread;
3 Player player = new Player(keyH);
4 ArrayList<Enemy> enemies = new ArrayList
  <>();
```

Game.java

```
public ArrayList<Bullet> bullets = new ArrayList<>();
```

Player.java

4. Polymorphism

มีการใช้ Polymorphism ผ่านการ Override เมธอด

```

@Override
public void keyPressed(KeyEvent e){
    int code = e.getKeyCode();
    if(code == KeyEvent.VK_W){
        upPressed = true;
    }
    if(code == KeyEvent.VK_S){
        downPressed = true;
    }
    if(code == KeyEvent.VK_A){
        leftPressed = true;
    }
    if(code == KeyEvent.VK_D){
        rightPressed = true;
    }
    if (code == KeyEvent.VK_SPACE) {
        spacePressed = true;
    }
    if (code == KeyEvent.VK_R) {
        rPressed = true;
    }
}

```

KeyHandle.java

```

@Override
public void run() {
    double drawInterval = 1000000000 / FPS;
    double nextDrawTime = System.nanoTime() + drawInterval;

    while (gameThread != null) {
        update();
        repaint();

        try {
            double remainingTime = nextDrawTime - System.nanoTime();
            remainingTime = remainingTime / 1000000;

            if (remainingTime < 0) {
                remainingTime = 0;
            }

            Thread.sleep((long) remainingTime);
            nextDrawTime += drawInterval;
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

```

Game.java

```

@Override
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2 = (Graphics2D) g;
    tileM.draw(g2);
    // Player
    player.draw(g2);

    // Bullets
    for (Bullet b : player.bullets) {
        b.draw(g2);
    }

    // Enemies
    for (Enemy e : enemies) {
        e.draw(g2);
    }

    // HUD
    drawHUD(g2);

    // Game Over
    if (gameOver) {
        g2.setColor(new Color(r:0, g:0, b:0, a:150));
        g2.fillRect(x:0, y:0, screenWidth, screenHeight);
        g2.setColor(Color.WHITE);
        g2.setFont(new Font(name:"Arial", Font.BOLD, size:48));
        g2.drawString(str:"GAME OVER", screenWidth / 2 - 150, screenHeight / 2);
    }
}

```

Game.java

5. Abstract

มีคลาสแม่ Entity เป็น Abstract Class ที่ใช้สืบทอดโดย Player และ Enemy

```

import java.awt.image.*;

public abstract class Entity {
    public BufferedImage up1,up2,down1,down2,left1,left2,right1,right2,downe1;
    public String direction;
    public int spriteCounter = 0;
    public int spriteNum = 1;
}

```

Entity.java


```

public class Player extends Entity {
    public int x, y;
    public int speed = 3;
    public int hp = 100;
    public String direction = "down";
    public int exp = 0;
    public int level = 1;
    public int expToNextLevel = 50;
    private long lastShotTime = 0;
    public int shootCooldown = 200; // 200 ms (0.2 วินาที)
    int attackPower = 10;
    int maxHp = 100;

    KeyHandle keyH;

    // ภาพแต่ละทิศ
    BufferedImage up1, up2, down1, down2, left1, left2, right1, right2;

    public ArrayList<Bullet> bullets = new ArrayList<>();

```

Player.java

```

public class Enemy extends Entity {
    public int x, y;
    public int speed = 1;
    public int hp = 20;
    public boolean alive = true;
    public int expReward = 20;
    BufferedImage downe1;

    public Enemy(int x, int y) {
        this.x = x;
        this.y = y;
        loadImages();
    }

```

Enemy.java

6. Inheritance

คลาส Player และ Enemy สืบทอดจาก Entity เพื่อใช้คุณสมบัติร่วม เช่นตำแหน่งและการเคลื่อนที่

```
1 public class Player extends Entity {
2     public int x, y;
3     public int speed = 3;
4     public int hp = 100;
5     public String direction = "down";
6     public int exp = 0;
7     public int level = 1;
8     public int expToNextLevel = 50;
9     private long lastShotTime = 0;
10    public int shootCooldown = 200;
11    // 200 ms (0.2 วินาที)
12    int attackPower = 10;
13    int maxHp = 100;
```

Player.java

```
1 public class Game extends JPanel implements
  Runnable {
2     // =====
3     // 🖱️ SET SCREEN
4     // =====
5     final int originalTileSize = 16;
6     final int scale = 3;
7     public static Game instance;
8     final int tileSize = originalTileSize *
  scale;
9     final int maxScreenCol = 16;
```

Game.java

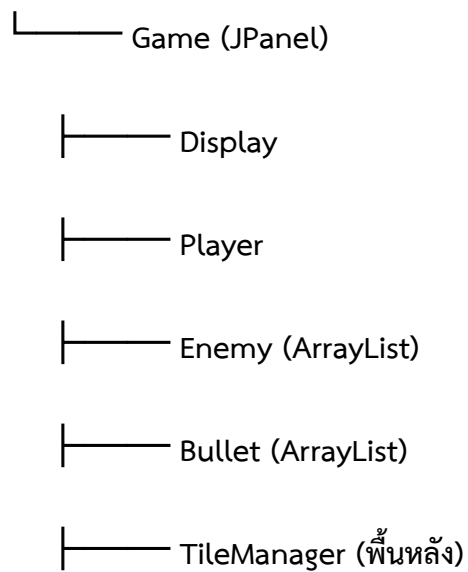
2.5 GUI (Graphic User Interface)

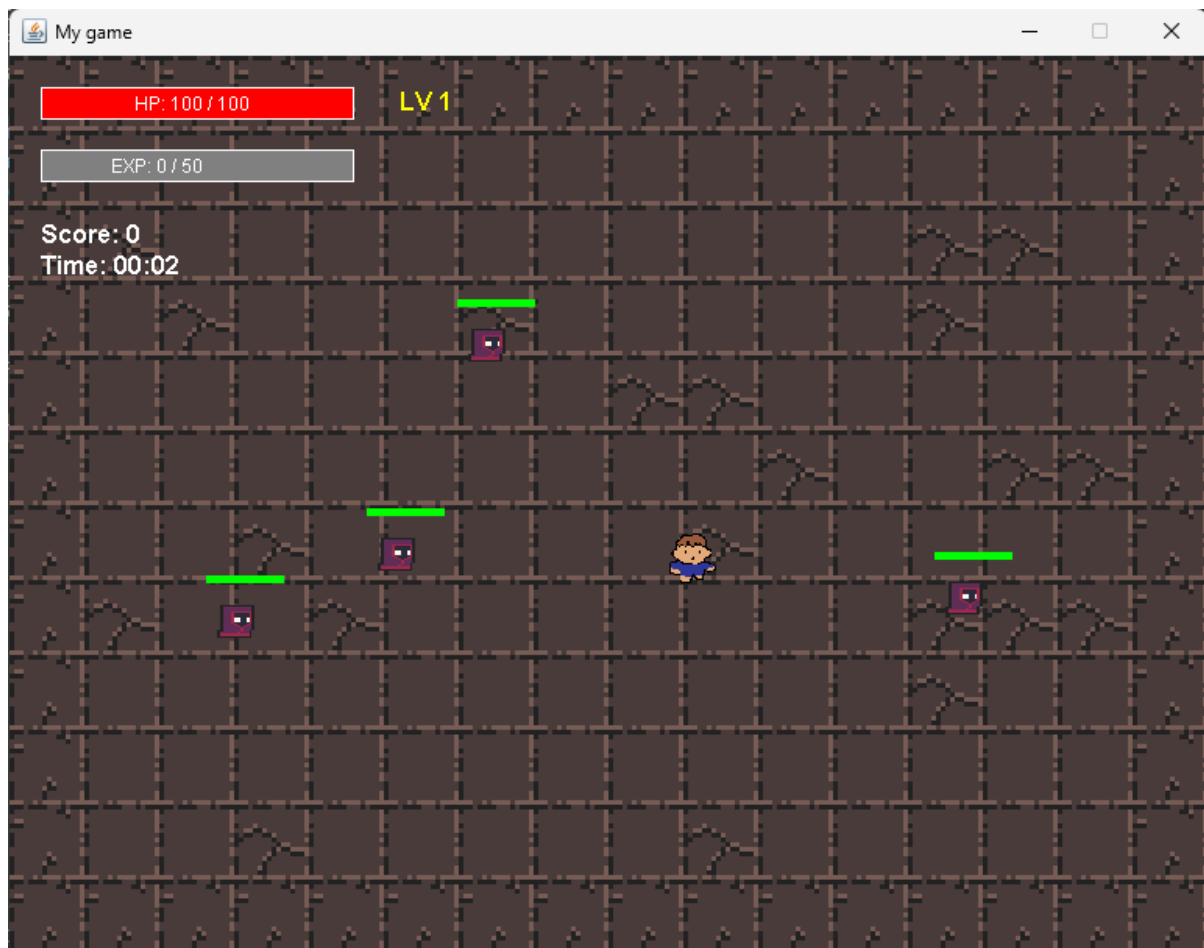
ประกอบด้วยองค์ประกอบหลัก:

- **JFrame**: หน้าต่างหลักของเกม
- **JPanel**: พื้นที่วาดเกม (คลาส Game)
- **Graphics2D**: ใช้วาด Player, Enemy, กระสุน, HUD

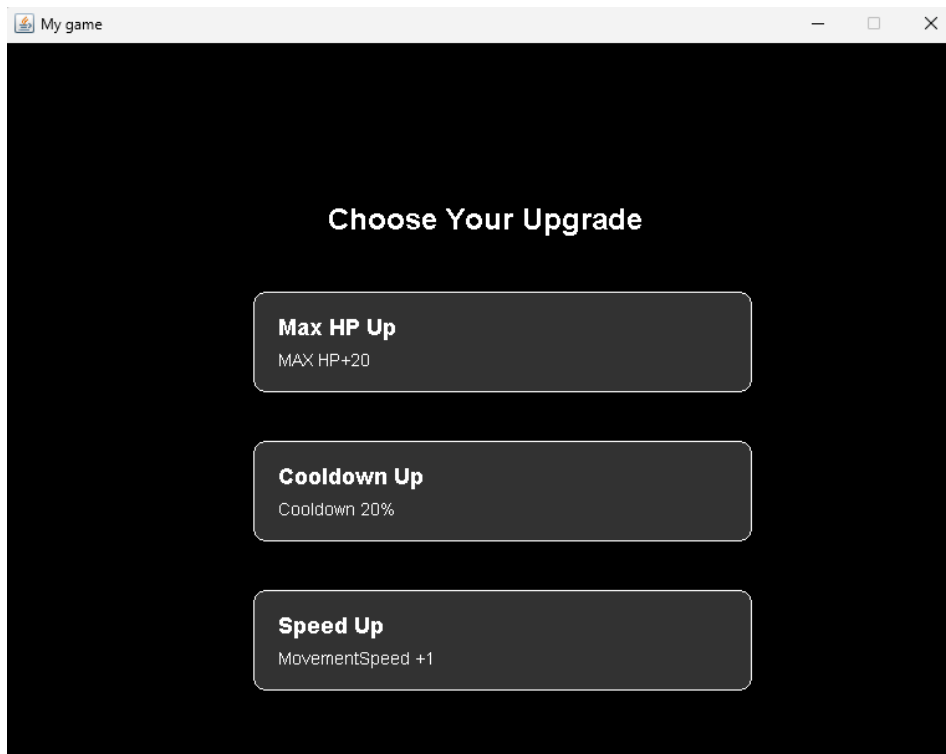
โครงสร้าง GUI

JFrame (หน้าจอหลัก)





Gameplay



UpgradeUI



Game over

2.6 Event Handling

ใช้คลาส KeyHandle สำหรับตรวจจับการกดแป้นพิมพ์ เช่น

- $\uparrow \downarrow \leftarrow \rightarrow$ เคลื่อนที่
- Spacebar \rightarrow ยิงกระสุน
- R \rightarrow เริ่มเกมใหม่หลัง Game Over

ตัวอย่างโค้ด:

```

1  import java.awt.event.*;
2  public class KeyHandle implements
KeyListener{
3      public boolean upPressed,downPressed,
leftPressed,rightPressed,spacePressed,
rPressed;
4      @Override
5      public void keyTyped(KeyEvent e){
6
7      }
8      @Override
9      public void keyPressed(KeyEvent e){
10         int code = e.getKeyCode();
11         if(code == KeyEvent.VK_W){
12             upPressed = true;
13         }
14         if(code == KeyEvent.VK_S){
15             downPressed = true;
16         }
17         if(code == KeyEvent.VK_A){
18             leftPressed = true;
19         }
20         if(code == KeyEvent.VK_D){
21             rightPressed =true;
22         }
23         if (code == KeyEvent.VK_SPACE) {
24             spacePressed = true;
25         }
26         if (code == KeyEvent.VK_R) {
27             rPressed = true;
28         }
29     }
30     @Override
31     public void keyReleased(KeyEvent e){
32         int code = e.getKeyCode();
33         if(code == KeyEvent.VK_W){
34             upPressed = false;
35         }
36         if(code == KeyEvent.VK_S){
37             downPressed = false;
38         }
39         if(code == KeyEvent.VK_A){
40             leftPressed = false;
41         }
42         if(code == KeyEvent.VK_D){
43             rightPressed =false;
44         }
45         if (code == KeyEvent.VK_SPACE) {
46             spacePressed = false;
47         }
48         if (code == KeyEvent.VK_R) rPressed =
false;
49     }
50 }
51

```

KeyHandle.java

บทที่ 3 สรุป

3.1 ปัญหาที่พบระหว่างการพัฒนา

- การจัดการกระสุนให้ไม่ยิงรัวเกินไป ต้องใช้ตัวแปร fireDelay ควบคุมเวลา
- การตรวจชนระหว่างวัตถุหลายชนิดต้องใช้ Rectangle.intersects()
- เมื่อศัตรูเกิดถี่เกินไปต้องใช้ระบบ Timer และการสุ่มตำแหน่ง

3.2 จุดเด่นของโปรแกรม

- มีระบบ Game Loop, Enemy Spawn, EXP / Level Up, HP Bar, Score / Time HUD
- สามารถต่อยอดเป็นเกม RPG

3.4 อัลกอริทึมสำคัญ

- Collision Detection Algorithm

```
1      for (Enemy e : enemies) {
2          if (e.alive && b.active) {
3              Rectangle bulletRect = new Rectangle(b.x, b.y, 10, 10);
4              if (bulletRect.intersects(e.getBounds())) {
5                  e.takeDamage(b.damage);
6                  if (!e.alive) player.gainExp(e.expReward); score += 10; //Score
7                  b.active = false;
8              }
9          }
10     }
```

- Enemy Spawn Algorithm

```
1      if (spawnTimer >= spawnInterval) {
2          spawnEnemy();
3          spawnTimer = 0;
4      }
```