โครงงาน

Mini project

จัดทำโดย

6504062630171 นายพสิษฐ์ มัสสะอาด

เสนอ

ผู้ช่วยศาสตราจารย์สถิต ประสมพันธ์

วิชา Object Oriented Programming

ภาคเรียนที่ 1/2565

ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ คณะวิทยาศาสตร์ประยุกต์

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

บทนำ

ที่มาและความสำคัญ

: เพื่อฝึกฝนการเขียนโปรแกรมรูปแบบ Object-Orient ให้ได้อย่างมีประสิทธิภาพ โดยใช้สร้างในรูปแบบเกม เพื่อประยุกต์สิ่งที่ได้ศึกษามา เพื่อเพิ่มความแม่นยำ และความชำนาญในศาสตร์การเขียนโปรแกรมลักษณะนี้ยิ่งขึ้น

ประเภทโครงงาน

: เกม

ประโยชน์

- **ฝึกฝนความสามารถในการหาความรู้ และประยุกต์ใช้**
- **ฝึกฝนการใช้งานเครื่องมือเพื่อเพิ่มประสิทธิภาพการทำงานต่าง ๆ**

ตารางแผนการทำงาน

ลำดับ	งานที่ทำ	1-15 พย.	16-30 พย.
1	ศึกษาและวางแผนการทำเกม	X	
2	วางโครงสร้างและคลาสต่างๆ	X	
3	อิมพลีเม้นท์แผนที่วางไว้	X	X
4	จัดทำเอกสาร		X
5	ตรวจสอบจุดผิดพลาด		X

ขอบเขตการทำงาน

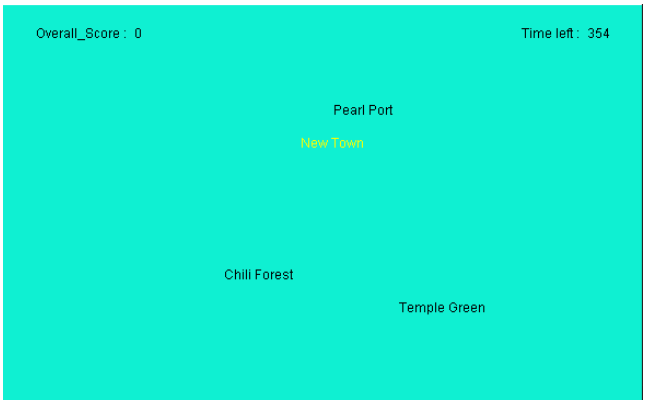
- **รายละเอียดของเกม**

: ผู้เล่นคือนักผจญภัยที่เริ่มต่อสู้กับมอนสเตอร์ระหว่างทางไปเมืองต่างๆ ไปเรื่อยๆจนกว่าจะหมดอายุขัย

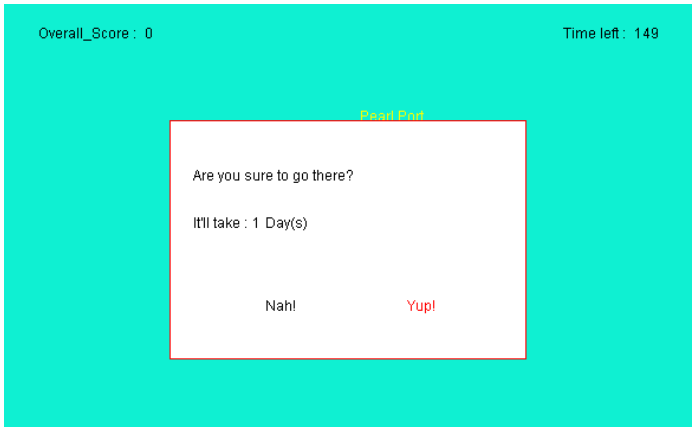
- **วิธีการเล่น**

: ปุ่มชี้ทิศทาง ซ้าย และ ขวา ในการเลื่อนไปมาระหว่างตัวเลือก ปุ่ม ขึ้น เพื่อเลือก เข้าแถบอัพสเตตัสด้วยการกด E ผู้เล่นต้องอัพสเตตัสของตัวเอง เพื่อที่จะมีค่าต่างๆมากพอที่จะสู้กับมอนสเตอร์ชุดต่อๆไป

- **StoryBoard**



: หน้าตัวเลือกพื้นที่ที่จะไป ซึ่งแต่ละที่ก็จะมีจำนวนมอนสเตอร์ไม่เท่ากัน โดยที่ Player ไม่สามารถวนอยู่แค่เมืองเดียวได้



: หน้ายืนยันการเดินทาง โดยจะมีจำนวนมอนสเตอร์ บอกอยู่ที่ Days

: หน้าต่อสู้ โดยที่ทางซ้าย คือผู้เล่น และทางขวา คือ มอนสเตอร์

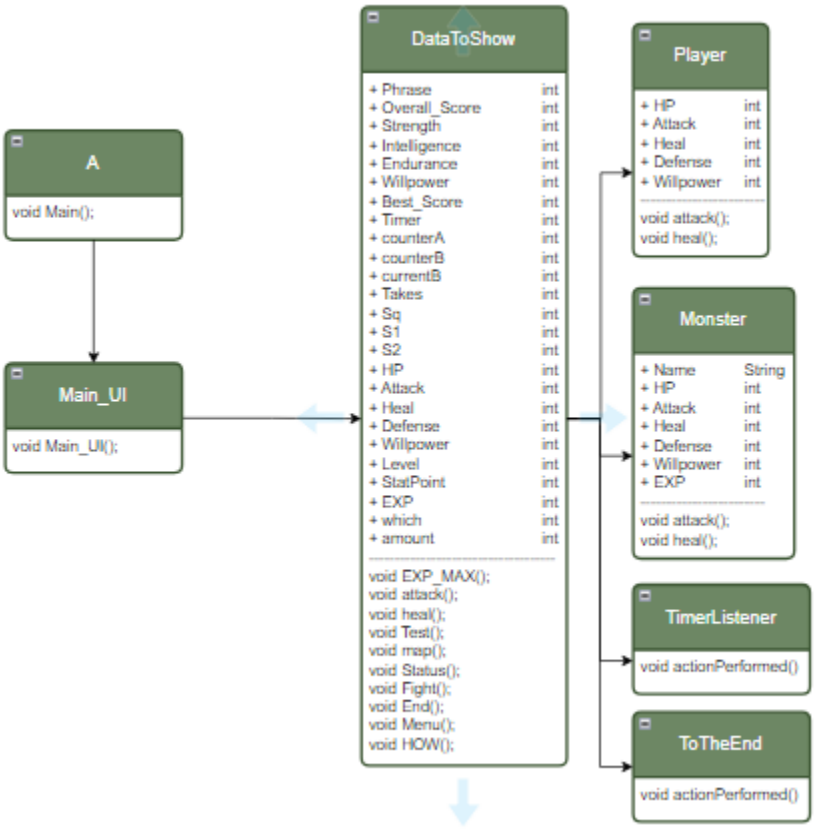
: เมื่อแพ้ ก็จะถูกส่งมายังหน้านี้ เมื่อชนะจะถูกส่งกลับไปหน้าแมพเหมือนเดิม เพื่อเล่นต่อไป

ส่วนการพัฒนา

- รูปแบบการพัฒนา

1. สร้างระบบ **Panel** เพื่อการแสดงผล.
2. สร้างหน้าต่าง ๆรองรับฟังก์ชั่นต่าง ๆที่ต้องการให้มี
3. สร้างสเตตัสต่าง ๆ ฟังก์ชั่นที่เกี่ยวข้องกับระบบเกม และไลบรารี่ศัตรู
4. ปรับระบบการต่อสู้ให้ผู้เล่นมีทางสู้ได้ แม้ในระดับที่สูงขึ้น

- **Class Diagram**



- **คำอธิบายส่วนโปรแกรม**

```
public DataToShow() {
    Timer timer = new Timer(delay:10,new TimerListener());
    timer.start();
    setFocusable(focusable:true);
    addKeyListener(new KeyAdapter() {
        @Override
        public void keyPressed(KeyEvent e) {···
    });
}
```

ตัวอย่าง Constructor ของ DataToShow ที่จะกำหนดสิ่งที่เกิดขึ้นภายในโปรแกรมระหว่างทำงาน และติดตั้งการรับคีย์จากผู้ใช้

```
class DataToShow extends JPanel {
    private int Phrase = -1, P = 0, Overall_Score = 0, Strength = 1, Intelligence = 1, Endurance = 1, Willpower = 1, Best_Score = 0;
    private int Timer = 360;
    private int counterA = 50; /* For status loop */
    private int counterB = 40; /* For map loop */
    private int currentB = 0; /* Indicate which city player in */
    private int Takes = 0;
    private int Sq = 0, S1 = 0, S2 = 0;
    private int HP = 150, Attack = 70, Defense = 10, Level = 1, StatPoint = Level * 20, Heal = 20, EXP = 0;
    private int which, amount;
    Player A = new Player(HP,Attack,Defense,Heal,Willpower);
    Monster B = new Monster(any:0, amount:1);
    Random C = new Random();
```

ตัวอย่างการใช้งาน Encapsulation กับตัวแปร คือ การกำหนดไม่ให้เกิดการเปลี่ยนตัวแปรในคลาสนี้จาก Class อื่นโดยตรง

```
Player(int HP, int Attack, int Defense, int Heal, int Willpower) {
    this.HP = HP; this.Attack = Attack; this.Heal = Heal; this.Willpower = Willpower; this.Defend = Defense;
}
```

ตัวอย่างการใช้งาน Composition ในการรับค่าจาก Constructor เพื่อแปลงค่าในเอทิตี้ตัวเองเพื่อใช้งานต่อๆไป

```
class TotheEnd implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        Timer = Timer - 1;
        if( Timer == 0 ) {
            End();
        }
    }
}
```

ตัวอย่างการใช้งาน Polymorphism ในการ Override ฟังก์ชั่นเปล่า จาก ActionListener ลงในชื่อ actionPerformed() ซึ่งเป็น Method จาก ActionListener

```java
public void Fight() {

    this.Phrase = 3;

    int a1 = 5, a2 = 1, b1 = 2, b2 = 1; /* 2 = Start point, 1 = Border */
    Random rand = new Random();

    if( Level <= 12 ) {
        if( Level <= 8 ) {
            if( Level <= 4 ) {
                if( Level <= 3 ) {
                    a1 = 5; a2 = 1; b1 = 2; b2 = 1;
                }
            } else {
                a1 = 14; a2 = 1; b1 = 5; b2 = 1;
            }
        } else {
            a1 = 14; a2 = 5; b1 = 20; b2 = 5;
        }
    } else {
        a1 = 19; a2 = 5; b1 = 51; b2 = 20;
    }

    which = rand.nextInt(a2, a1+1);
    amount = rand.nextInt(b2, b1+1);

    Player Some = new Player(HP, Attack, Defense, Heal, Willpower);
    Monster Any = new Monster( which, amount );
    B = Any;
    A = Some;

}
```

หนึ่งในอัลกอริทึมสำคัญของเกมนี้ คือ การกำหนดระดับความยาก โดยอิงจาก Level ของผู้เล่น โดยตัวฟังก์ชั่นจะกำหนดศัตรูที่จะเจอ และจำนวน Magnifier เพื่อความยากที่มากขึ้น

- **ส่วน GUI ภายในเกม**

มีการใช้งานดังนี้ :

```java
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
```

EventHandling :

ใช้งานปุ่มลูกศร ซ้าย-ขวา ในการเลือกระหว่างตัวเลือก

ใช้งานลูกศร บน เพื่อเลือกตัวเลือกนั้นๆ

ใช้งานปุ่ม E เพื่อเข้าหน้าอัพเดทสเตตัส

ใช้งาน G ในหน้าหน้าอัพเดทสเตตัสเพื่อจะโดนส่งไปที่ไว้วันกลับ

สรุป

ปัญหาที่พบระหว่างพัฒนา

- การเบิร์นเอ้าท์ระหว่างพัฒนา

- OverEngineering และ Over-expectation

- การเชื่อมไปมาระหว่าง EventHandler และ DrawComponent

- การควบคุมฟังก์ชั่นต่างๆให้ทำงานภายใต้ความสมดุล

จุดเด่นที่สุดของโปรแกรม

- มีจุดหลายจุดที่สามารถพัฒนาต่อยอด ที่สามารถทำให้เกมนี้มีมิติมากขึ้นกว่านี้ได้ หากมีเวลามากพอ

- Replay value ที่ไม่แย่ สามารถเล่นได้ 2 – 3 ครั้งก่อนที่จะเบื่อ

- โค้ดอ่านยาก จากการรวมทุกอย่างลงไฟล์เดียว ทำให้จะต้องใช้เวลาแกะนานขึ้น

คำขอร้องแด่สำหรับผู้สอน

: คงจะดีกว่านี้ ถ้างานนี้ สเกลคะแนนที่ได้เยอะกว่านี้ และได้เวลาในการทำมากกว่า 1 เดือนครับ