Game Project

The Victory gate

จัดทำโดย

6504062663192 ภัทรพล ปิ่นทอง

เสนอ

ผู้ช่วยศาสตราจารย์สถิตย์ ประสมพันธ์

วิชา Object Oriented Programming

ภาคเรียนที่ 1/2566

ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ คณะวิทยาศาสตร์ประยุกต์

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

เกี่ยวกับโครงงาน

บทที่ 1: บทนำ

ที่มาและความสำคัญของโปรเจ็ค ประเภทของโครงการ ประโยชน์

บทที่ 2: ส่วนการพัฒนา

เนื้อเรื่องย่อหรือวิธิการเล่น (กรณีเกมส์)

Class Diagram

รูปแบบการพัฒนา Application / Applet

อธิบายส่วนของโปรแกรมที่มี

- Constructor
- Encapsulation
- Composition
- Polymorphism
- Abstract
- Inheritance

หน้าจอ GUI

บทที่ 3: สรุป

ปัญหาที่พบระหว่างการพัฒนา

จุดเด่นของโปรแกรมที่ไม่เหมือนใคร

บทที่ 1 บทนำ:

ที่มาและความสำคัญของโปรเจ็ค

โครงงานนี้จัดขึ้นเพื่อวัดผลความสามารถในการเรียนวิชา Object Oriented Programming โดย การนำเรื่องที่เรียนมาสร้างเป็นชิ้นงานในรูปแบบเกม ผู้จัดทำได้สร้างเกมนี้ขึ้นมา

ประเภทของโครงการ

โปรแกรมเกมแอปพลิเคชั่น

ประโยชน์

1 เล่นเพื่อความสนุกสนาน

2 ฝึกความอดทน

3 ฝึกประสาทสัมผัส

ขอบเขตของโครงการ

1.ตัวละครในเกมเคลื่อนย้ายในแนวแกน X และ Y

2.เคลื่อนไหวตัวละครด้วยการใช้ ปุ่ม a,d หรือ ลูกศรซ้าย,ขวาเพื่อบังคับทิศทาง และ w,spacebar,ลูกศรขึ้น เพื่อกระโดด

3. เกมมี1 level

4 เมื่อตัวละครชน F พลังชีวิตจะลด

5.เมื่อตัวละครกระโดดข้าม F คะแนนจะเพิ่มขึ้น

6. เมื่อค่าพลังชีวิต = 0 เกมจะจบ หรือเวลาหมด(60 วินาที) จะจบเกม

บทที่ 2 : ส่วนการพัฒนา

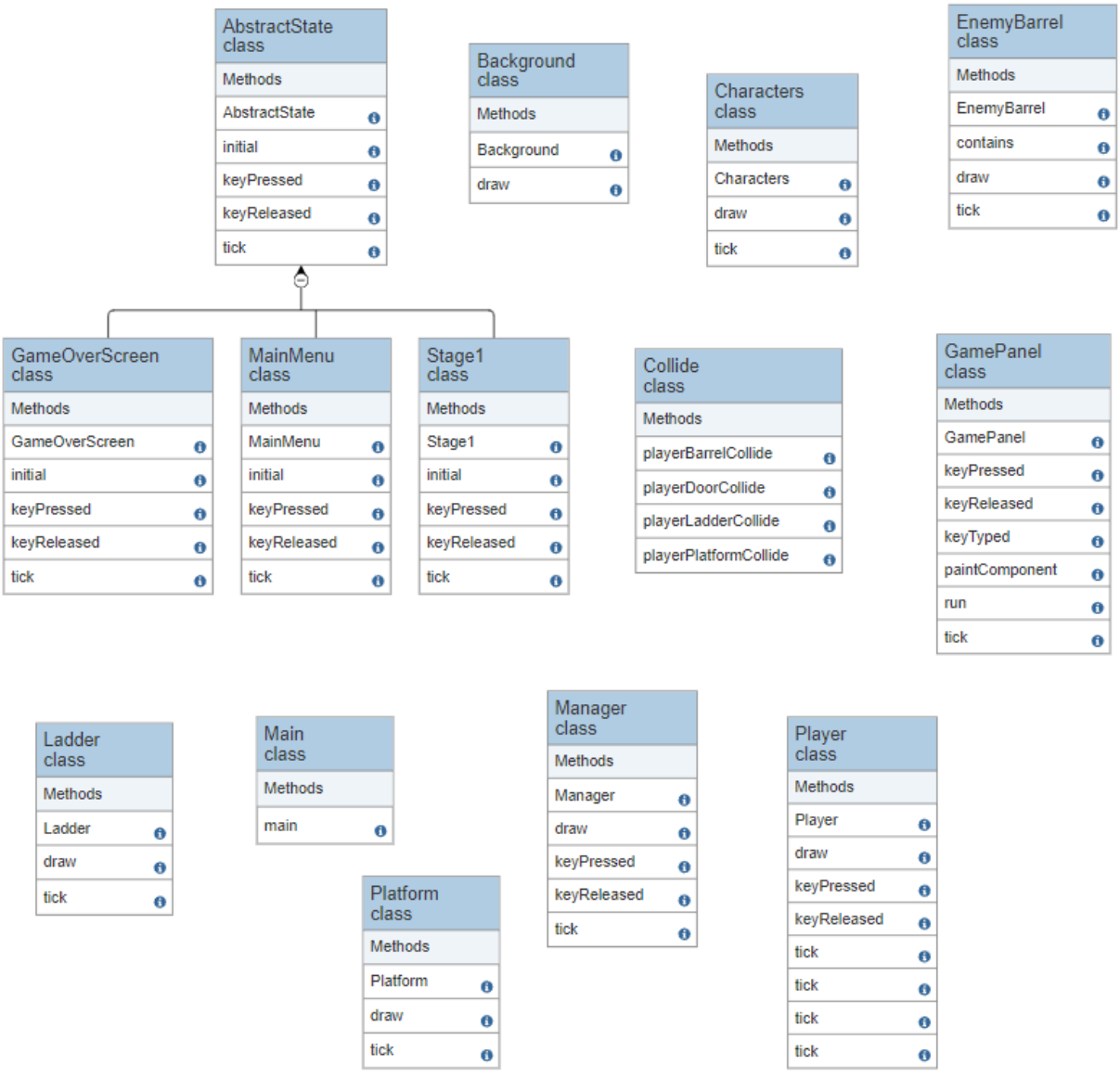
เนื้อเรื่องย่อ

เกมประตูแห่งชัยชนะ โดยจะมีเด็กหนุ่มคนหนึ่งที่ต้องการหาทางออกสู่โลกภายนอกแต่เขาไม่สามารถออกไปได้ เพราะมีตัวขัดขวางที่ชื่อ F มาขัดขวางไม่ให้ออกไปง่ายๆ
ผู้เล่นจะต้องก้าวข้าม F เพื่อที่จะผ่านออกไปเปิดประตูสู่โลกภายนอก

วิธีเล่น

ใช้ a,d หรือ ลูกศรซ้าย,ขวาเพื่อบังคับทิศทาง และ w,spacebar,ลูกศรขึ้น เพื่อกระโดด ข้าม F เมื่อเริ่มเกม จะมีค่าพลังชีวิต = 3 เมื่อเดินชน F จะลดเลือดทีละ 1 เมื่อค่าพลังชีวิต = 0 เกมจะจบ เกมจะมีการจับเวลา 60วินาที เมื่อเวลาหมด จะจบเกม

Class Diagram



ส่วนของโปรแกรมที่มี

Constructor

```
public AbstractState(Manager gamestates) {//constructor
    this.gamestates = gamestates;
    initial();
}
```

มี constructor ที่รับพารามิเตอร์ชนิด Manager. เรียกเมทอด initial().

```
public Background(int x2, int y2) { // constructor
setBounds(x2, y2, 1200, 1200);
}
```

มี constructor ที่รับพารามิเตอร์ชนิด int x2 และ int y2. เรียกเมทอด setBounds(x2, y2, 1200, 1200) เพื่อกำหนดขอบเขตของ Background ให้มีขนาด 1200x1200.

```
public Characters(int x4, int y4) {
    setBounds(x4, y4, 20, 20); // size of rectangles
}
```

มี constructor ที่รับพารามิเตอร์ชนิด int x4 และ int y4. เรียกเมทอด setBounds(x4, y4, 20, 20) เพื่อกำหนดขอบเขตของ Characters ให้มีขนาด 20x20.

```
public EnemyBarrel(int w, int h) {
    x = 240;
    y = 120;
    this.w = w;
    this.h = h;
}
```

มี constructor ที่รับพารามิเตอร์ชนิด int w และ int h.

Encapsulation

```
public GameOverScreen(Manager gamestates) {
    super(gamestates);
}
```

เรียก constructor ของคลาส AbstractState และส่ง gamestates ไป.

```
public GamePanel() {
    setPreferredSize(new Dimension(1200, 1200));
    addKeyListener(this);
    setFocusable(true);
    start();    }        //calls start
```

มี constructor ที่ไม่รับพารามิเตอร์. กำหนดขนาดของ GamePanel ด้วย setPreferredSize(new Dimension(1200, 1200)). เพิ่ม KeyListener และตั้งค่าให้ GamePanel เป็น focusable.

```
public Ladder(int x3, int y3) {
    setBounds(x3, y3, 60, 110); // size of ladder
}
```

มี constructor ที่รับพารามิเตอร์ชนิด int x3 และ int y3. เรียกเมทอด setBounds(x3, y3, 60, 110) เพื่อกำหนดขอบเขตของ Ladder ให้มีขนาด 60x110.

```
public MainMenu(Manager gamestates) {
    super(gamestates);
}
```

เรียก constructor ของคลาส AbstractState และส่ง gamestates ไป.

Composition

```
public Manager() { // constructor
    stages = new Stack<AbstractState>();//creates a stack of game states
    stages.push(new MainMenu(this));//adds start menu
}
```

มี constructor ที่รับพารามิเตอร์ชนิด Manager gamestates. เรียก constructor ของคลาส AbstractState และส่ง gamestates ไป.

```
public Platform(int x, int y) {
    setBounds(x, y, 665, 40); // size of platform
}
```

มี constructor ที่รับพารามิเตอร์ int x และ int y ซึ่งใช้สำหรับกำหนดตำแหน่งและขนาดของแพลตฟอร์ม. เรียก setBounds(x, y, 665, 40) เพื่อกำหนดขนาดและตำแหน่งของ Rectangle.

```
public Player(int w, int h) {
    x = 100; // where horizontally the player character will start
    y = 700; // where vertically the player character will start
    this.w = w;
    this.h = h;
}
```

มี constructor รับพารามิเตอร์เพื่อกำหนดตำแหน่งและขนาดเริ่มต้นของผู้เล่น.

```
public Stage1(Manager gamestates) {
    super(gamestates);
    GamePanel.TickCounter = 0;
    if (win ==1) {
        gamestates.stages.push(new GameOverScreen(gamestates));
    }
}
```

เรียก constructor ของ superclass (AbstractState) เพื่อทำการ initialize

Encapsulation

```
public Manager gamestates;
```

มีตัวแปร gamestates ที่ถูกกำหนด modifier เป็น private.

```
public class Background extends Rectangle
```

มีการใช้ inheritance จาก Rectangle ซึ่ง Rectangle เป็น class ที่มีตัวแปรและเมทอดที่ถูกกำหนดเป็น protected.

```
public class Characters extends Rectangle
```

มีการใช้ inheritance จาก Rectangle ซึ่ง Rectangle เป็น class ที่มีตัวแปรและเมทอดที่ถูกกำหนดเป็น protected.

```
public Stack<AbstractState> stages;
```

stages ถูกประกาศเป็น public เพื่อให้สามารถเข้าถึงได้จากภายนอกคลาส

```
public void keyPressed(int P) {
    stages.peek().keyPressed(P);
}
public void keyReleased(int R) {
    stages.peek().keyPressed(R);
}
public void tick() {
    stages.peek().tick();
}
public void draw(Graphics g) {
    stages.peek().draw(g);
}
```

เมธอดที่อยู่ภายในคลาส Manager ที่ใช้สำหรับการจัดการกับการกดปุ่มและการอัปเดตและวาดรูปของเกม

```
protected Characters chara;
```

```
String[] choices = { "Start", " ", "No" };
protected int select = 0;
```

chara กำหนดเป็น protected เพื่อจำกัดการเข้าถึงโดยตรงจากภายนอกคลาส

choices และ select กำหนดเป็น protected ซึ่งหมายถึงสามารถเข้าถึงได้จากsubclass

```
protected long tTime = 1000000 / 60;//60fps
protected Thread t;
protected boolean running = false;
protected static int TickCounter;
protected Manager gamestates;
```

มีการใช้ protected ในการประกาศตัวแปรและเมทอด, ทำให้สามารถเข้าถึงได้จากคลาสที่สืบทอดมา (protected สามารถเข้าถึงได้จากคลาสที่อยู่ใน package เดียวกัน).

```java
public class EnemyBarrel {
    protected int platformlevel = 0;
    protected boolean left = false;
    protected boolean right = false;
    protected boolean barrelfall;
    protected double x, y;
    protected static int w;
    protected static int h;
    protected double downvelocity = 6;
    protected double currentfallvel = 0.2;
    Rectangle r1 = new Rectangle((int) x, (int) y, w, h);
```

มีการใช้ protected ในการประกาศตัวแปรและเมทอดในคลาส EnemyBarrel, ทำให้สามารถเข้าถึงได้จากคลาสที่สืบทอดมา.

```java
public void keyPressed(int k) { //make flow more smoothly
    if (k == KeyEvent.VK_LEFT || k ==KeyEvent.VK_A) {···

    if (k == KeyEvent.VK_RIGHT || k ==KeyEvent.VK_D) {···

    if (k == KeyEvent.VK_UP|| k ==KeyEvent.VK_SPACE|| k ==KeyEvent.VK_W) {···
}
```

มี methods สำหรับ handling key presses และ key releases

Composition

```
protected Manager gamestates;
```

```
public void run() { // method that will run the gameloop
    gamestates = new Manager();
    long begin, wait;
    while (running = true) { // when the game is running, this ploc
        begin = System.nanoTime(); //nano 10^-9
        TickCounter = TickCounter + 1;
        tick();
        repaint(); // responsible for the graphics in the gameloop
        wait = tTime - begin / 1000000;//puts into milliseconds
        if (wait <= 0) {
            wait = 5;// was 5
        }
        try {
            Thread.sleep(wait);
        } catch (Exception excep) {
            System.out.println("Exception Caught");
        } } }
```

```
public void tick() { //updates the logic
    gamestates.tick(); }
```

สร้างและใช้งาน Manager ในเมทอด run() และ tick().

```
> public class Ladder extends Rectangle {···
```

ใช้ Rectangle ในการสืบทอดและให้ Ladder ได้รับคุณสมบัติและเมทอดของ Rectangle.

```
public class Manager {
```

```
public void tick() {
    stages.peek().tick();
}
public void draw(Graphics g) {
    stages.peek().draw(g);
}
```

Manager มี composition กับ GamePanel

Polymorphism

```
public abstract void initial();
public abstract void tick();
protected abstract void draw(Graphics g);
public abstract void keyPressed(int k);
public abstract void keyReleased(int k);
```

ใช้ Polymorphism ในรูปแบบของ abstract methods ที่สามารถถูก implement ใน subclass ได้.

```
    public void draw(Graphics g) {
        g.setColor(Color.black); //sets the background
        g.fillRect((int) x, (int) y, width, height); //
    }
}
```

มีเมทอด draw ที่สามารถถูก override ใน subclass ได้.

```
public void draw(Graphics g) {
```

มีเมทอด draw ที่สามารถถูก override ใน subclass ได้.

```
public void keyPressed(int k) { ···

public void keyReleased(int k) {    }
```

```
public void draw(Graphics g) { ···

public void tick(Platform[] platforms) {···


public void tick(Characters[] charas) {···

public void tick(Ladder[] ladders) {···
```

ใช้เมทอด keyPressed, draw, และ tick ที่มีพารามิเตอร์แตกต่างกัน เพื่อให้สามารถทำงานกับตัวแปรชนิดต่าง ๆ.

Abstract

```
public abstract class AbstractState
```

ประกาศคลาส AbstractState เป็น abstract class

```
public void keyPressed(int kp) {
    if (kp == KeyEvent.VK_DOWN) { //if down arrow key pressed
        select = select + 1; //goes through the options
        if (select >= 3) {
            select = 2;
        }
    } else if (kp == KeyEvent.VK_UP) { //if up arrow key pressed
        select = select - 1; //goes through the options
        if (2 > select) {
            select = 0;
        }
    }
    if (kp == KeyEvent.VK_ENTER) {
        if (select == 0) { //restart
            gamestates.stages.push(new Stage1(gamestates));
            Stage1.lives = 3;
            Stage1.win = 0;
            Stage1.score = 0;
        } else if (select == 2) { // exit choice on menu
            System.exit(0);
        }
    }
}
```

มีการใช้ Abstract GameOverScreen สืบทอดมาจาก AbstractState และได้มีการimplement เมทอด initial(), keyPressed(), keyReleased(), และ draw(Graphics g).

Inheritance

```
public class Background extends Rectangle
```

Background สืบทอดจากคลาส Rectangle

```
public class Characters extends Rectangle {
```

Characters สืบทอดจากคลาส จาก Rectangle.

```
public class GameOverScreen extends AbstractState {
```

GameOverScreen สืบทอดมาจาก AbstractState.

```
public class GamePanel extends JPanel implements KeyListener, Runnable {
```

GamePanel สืบทอดมาจาก JPanel

```
public class MainMenu extends AbstractState
```

MainMenu สืบทอดมาจาก AbstractState

```
public class Platform extends Rectangle
```

Platform สืบทอดมาจาก Rectangle

```
public class Stage1 extends AbstractState {
```

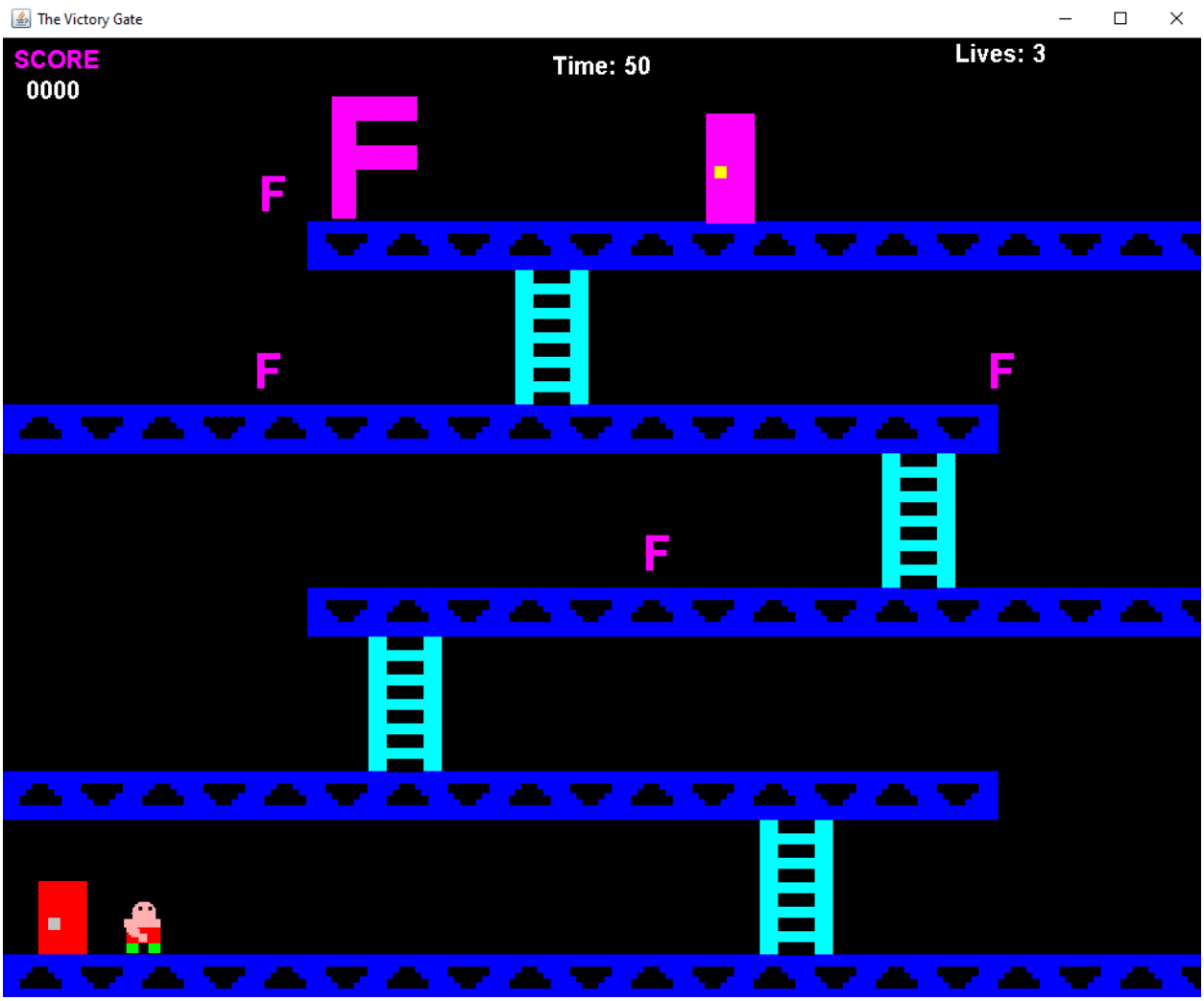
Stage1 สืบทอดมาจาก AbstractState

GUI

หน้าต่างเริ่มเกม จะสามารถเลือกกด Start หรือ No

ถ้ากด Start จะเข้าสู่เกม ถ้ากด No จะปิดโปรแกรมใช้

JFrame สร้างหน้าต่างหลักของเกม



เมื่อเริ่มเกมจะมีศัตรูมาจากFตัวใหญ่ เป็นFตัวเล็กเดินตามทางมาเรื่อยๆ ผู้เล่นจะต้องเดินไต่บันไดขึ้นไปถึงประตูข้างบนจึงจะสามารถชนะได้ เกมจะจับเวล60 วินาที ถ้าหากเดินไปชนกับFตัวเล็ก พลังชีวิต(lives) จะลดลงครั้งละ 1 แต้ม เมื่อพลังชีวิตหมดหรือเวลาหมดจะเกมโอเวอร์

เมื่อพลังชีวิตหมดหรือเวลาหมดจะแสดงหน้าต่างGAME OVER และคะแนนทั้งหมดแล้วจะตัวเลือก Restart กับ Exit Game

เมื่อกด Restart จะกลับไปเริ่มเกมใหม่ เมื่อกด Exit Game จะปิดโปรแกรม

บทที่ 3 : สรุป

ปัญหาที่พบระหว่างการพัฒนา

เนื่องจากแบ่งเวลาให้แต่ละรายวิชาได้ไม่ดีและความรู้ไม่พอทำให้การทำโปรเจคไม่เป็นไปตามแผนที่วางไว้

จุดเด่นของโปรแกรมที่ไม่เหมือนใคร

โปรแกรมใช้การวาดตัวละครทั้งหมดไม่ได้มีการนำภาพมาใส่ทำให้เป็นเอกลักษณ์ของเกมนี้

คำแนะนำสำหรับผู้สอนที่อยากให้อธิบาย หรือที่เรียนแล้วไม่เข้าใจ หรืออยากให้เพิ่มสำหรับน้องๆรุ่นต่อไป

อยากให้อาจารย์ค่อยๆฝึกทำแต่ละงานแบบค่อยเป็นค่อยไป และอยากให้การบ้านน้อยลงกว่านี้เพราะทุกวิชาการบ้านเยอะมากทำให้จัดสรรเวลาในการทำแต่ละวิชายาก