期末報告

主題：實作貪吃蛇遊戲

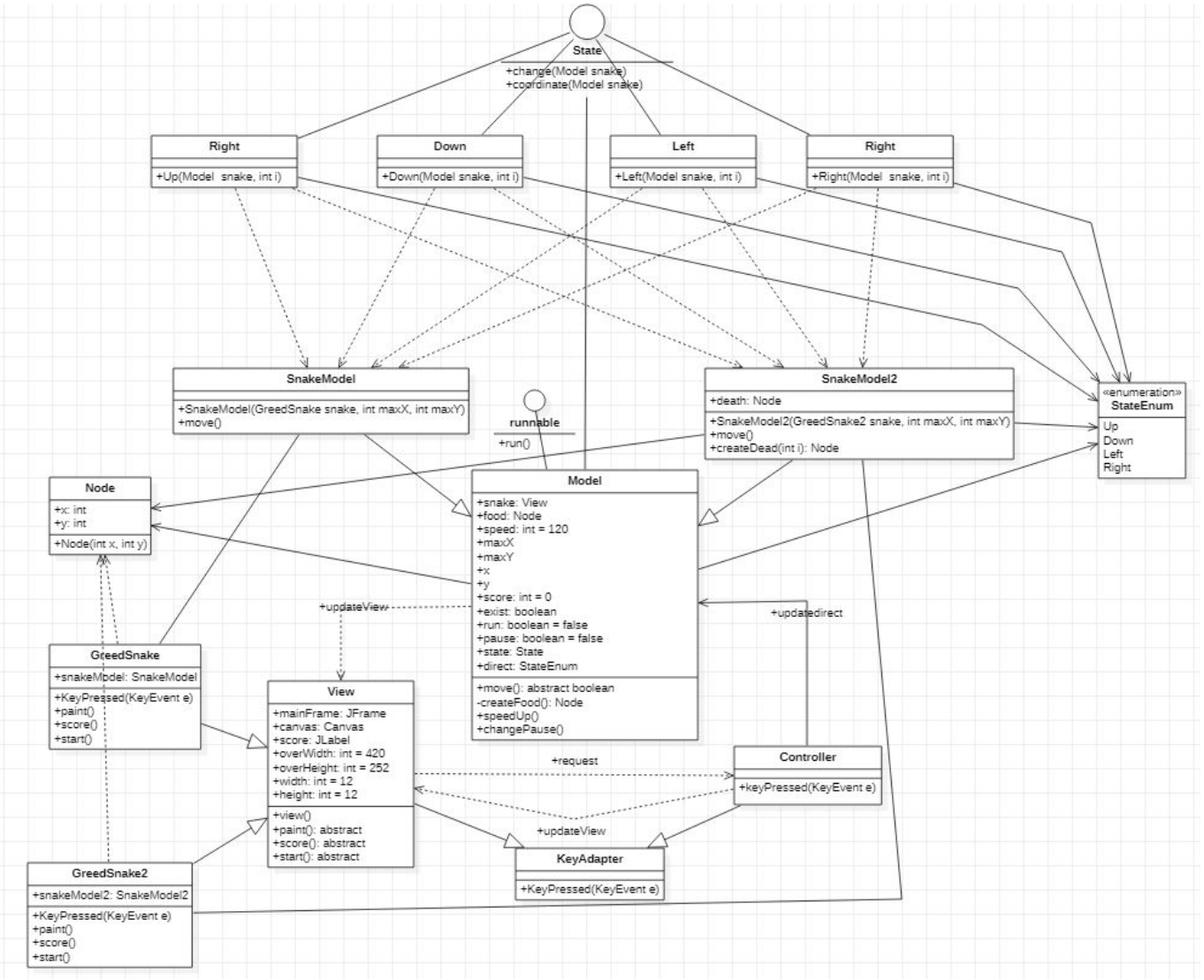
組員：D0713052楊翔竣
D0745560張誌元
D0745526唐家謙
D0745395吳東穎

1.題目說明：

設計一個貪吃蛇小遊戲以MVC進行架構，View和Controller去繼承AWT的KeyAdapter，使Controller可以使用keyevent去控制Model狀態，也讓View可以順利初始化遊戲界面，Model繼承Runnable，利用run程序可以給View即時的反饋達成一個動態介面。

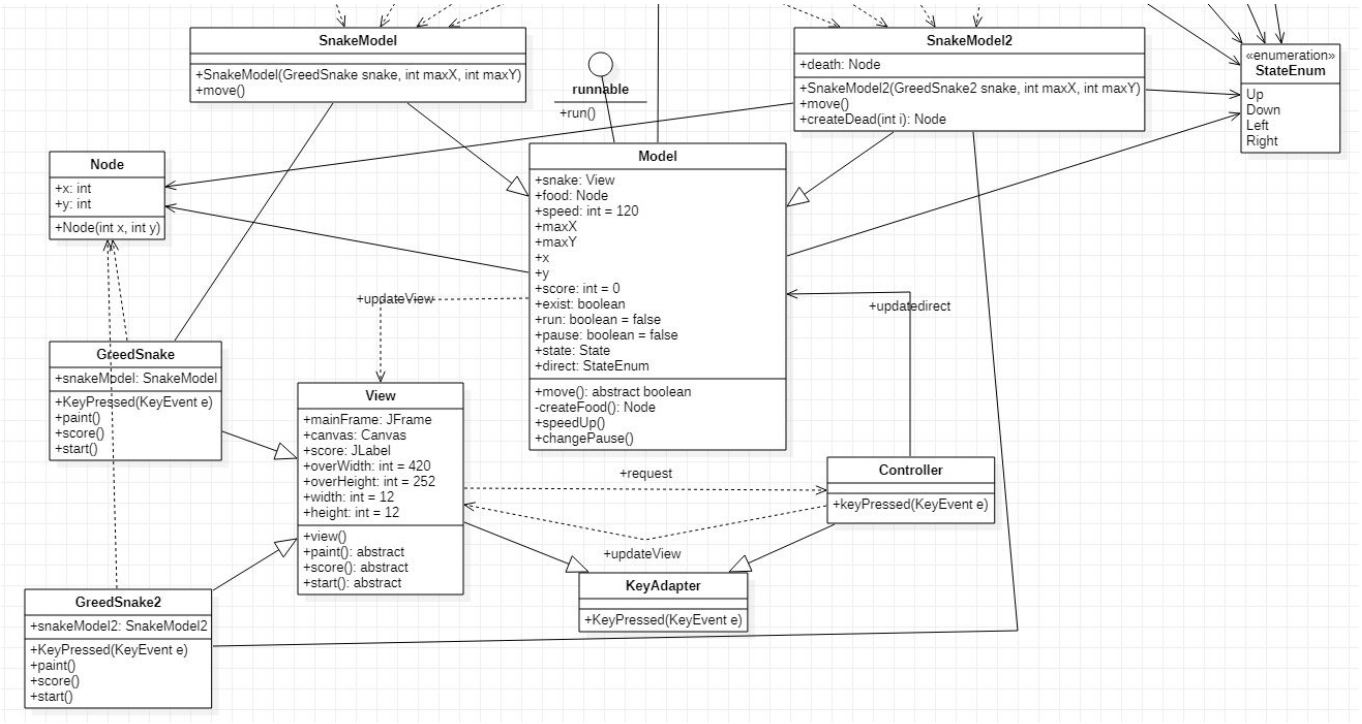
我們實作兩個Applications，一個是一般的GreedSnake，另一個則是有毒藥且無視範圍的GreedSnake2，這兩個Application主要由4個class去實作，分別是GreedSnake、SnakeModel、GreedSnake2、SnakeModel2，其中SnakeModel、SnakeModel2繼承Model，GreedSnake、GreedSnake2繼承View。

2.系統設計：


State
+change(Model snake)
+coordinate(Model snake)
Right
+Up(Model snake, int i)
Down
+Down(Model snake, int i)
Left
+Left(Model snake, int i)
Right
+Right(Model snake, int i)
SnakeModel
+SnakeModel(GreedSnake snake, int maxX, int maxY)
+move()
runnable
+run()
SnakeModel2
+death: Node
+SnakeModel2(GreedSnake2 snake, int maxX, int maxY)
+move()
+createDead(int i): Node
«enumeration»
StateEnum
Up
Down
Left
Right
Node
+x: int
+y: int
+Node(int x, int y)
Model
+snake: View
+food: Node
+speed: int = 120
+maxX
+maxY
+x
+y
+score: int = 0
+exist: boolean
+run: boolean = false
+pause: boolean = false
+state: State
+direct: StateEnum
+move(): abstract boolean
-createFood(): Node
+speedUp()
+changePause()
+updateView
+updatedirect
GreedSnake
+snakeModel: SnakeModel
+KeyPressed(KeyEvent e)
+paint()
+score()
+start()
View
+mainFrame: JFrame
+canvas: Canvas
+score: JLabel
+overWidth: int = 420
+overHeight: int = 252
+width: int = 12
+height: int = 12
+view()
+paint(): abstract
+score(): abstract
+start(): abstract
+request
Controller
+keyPressed(KeyEvent e)
+updateView
KeyAdapter
+KeyPressed(KeyEvent e)
GreedSnake2
+snakeModel2: SnakeModel2
+KeyPressed(KeyEvent e)
+paint()
+score()
+start()

3.系統設計說明：

1.框架功能與應用程式功能：

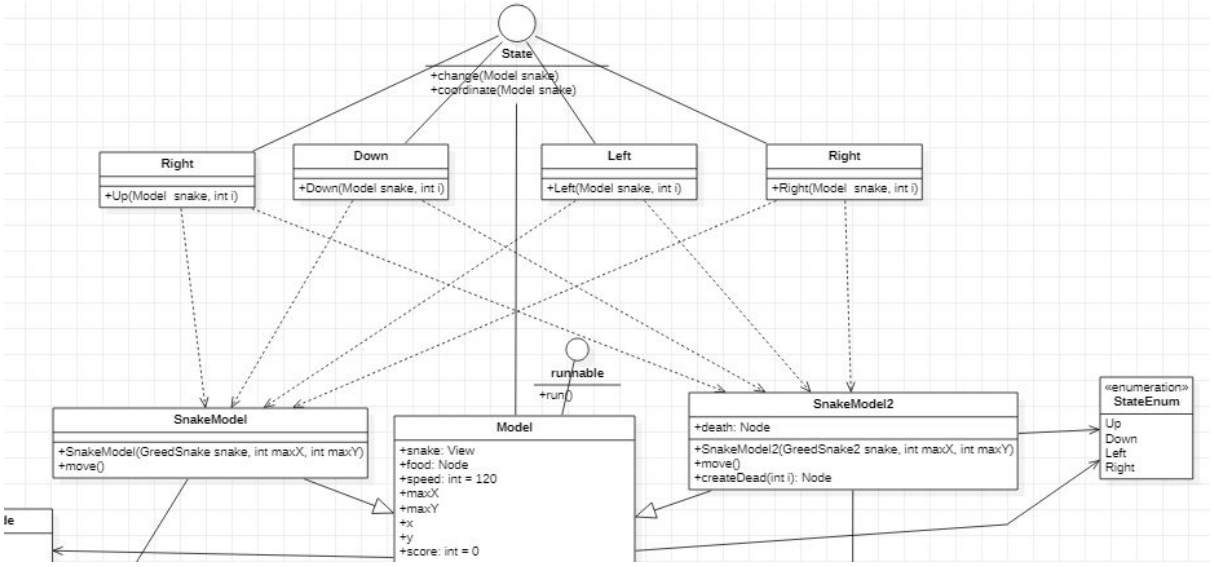


在View中使用Swing製造GUI介面，建立好遊戲初始介面，接著使用Theard執行Model.run，run()會使用到move()去判斷程式運動狀況，再更新View並paint()出來，Model的direct會被Controller所改變，藉此達成一個動態程式。

第一個Application都是把Framework的架構實做出來，作成最一般的GreedSnake，以上下左右鍵控制，space可暫停遊戲，enter可重新遊戲。

第二個Application我們新增了毒藥屬性和生成毒藥的function()，與第一個Application主要差別在於SnakeModel2.move()和GreedSnake.paint()。

2.如何應用設計原理與設計樣式：



建立一個State的interface，內含change()(可改變direct)、coordinate()(改變Model座標)，建立4個class "Up、Down、Left、Right"去實作State interface。

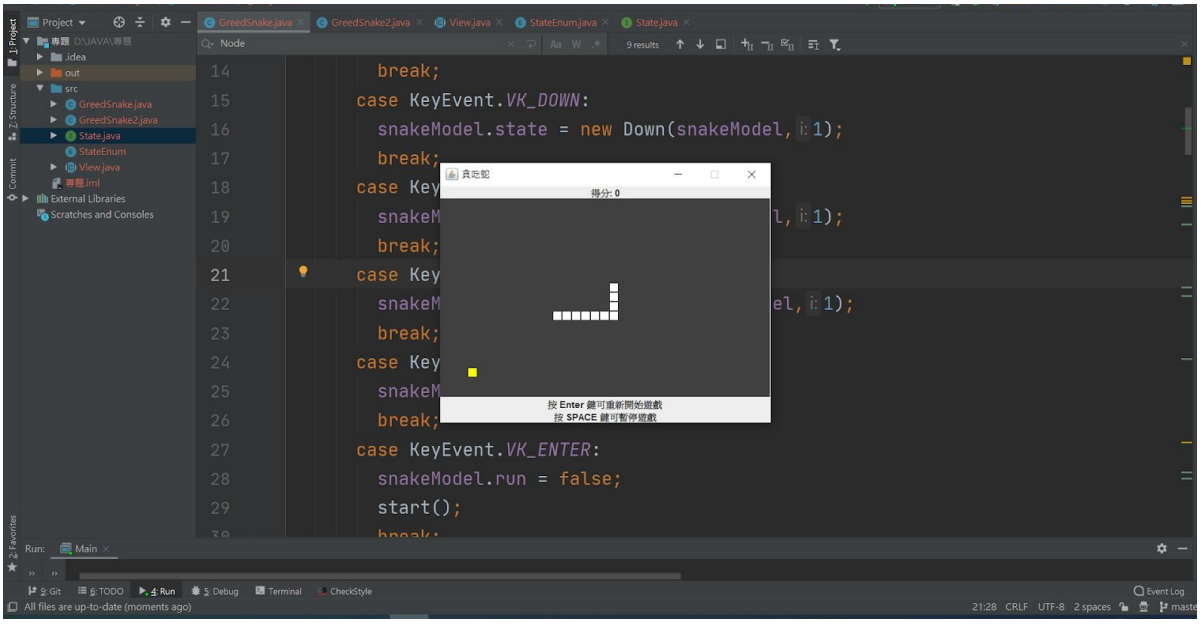
我們將Model的state拉出來另外控制，當玩家使用到keyPressed(也就是上下左右鍵)，會透過Up、Down、Left、Right其中之一去改變Model的direct和蛇的座標(即x、y)。

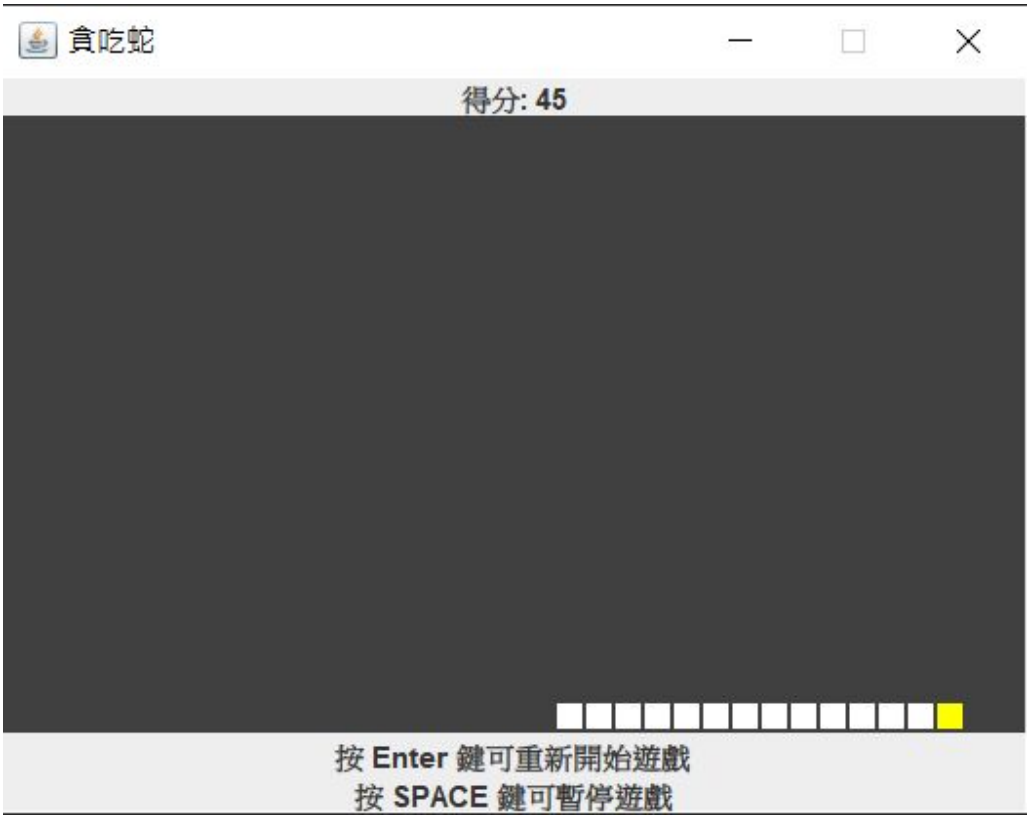
4.程式狀態說明：

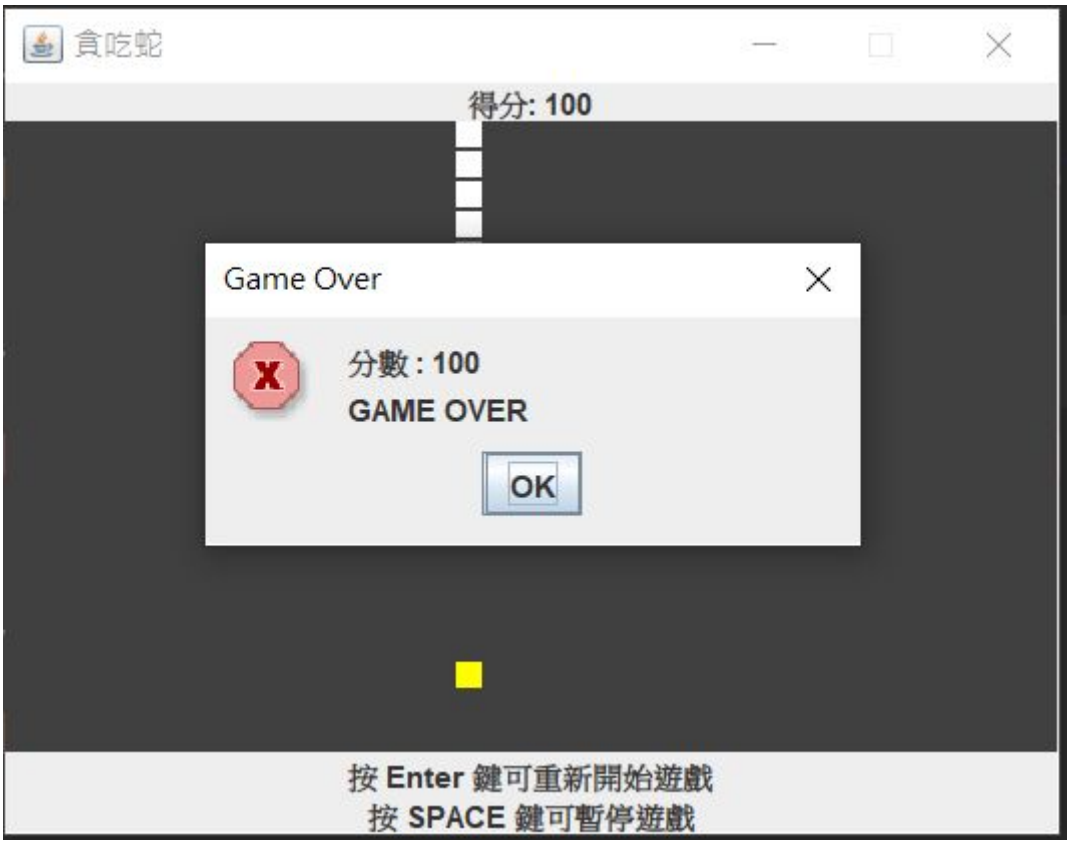
1.測試與執行畫面說明：

Application 1.

以上下左右鍵控制snake移動，Enter可重新遊戲，Space可暫停遊戲。

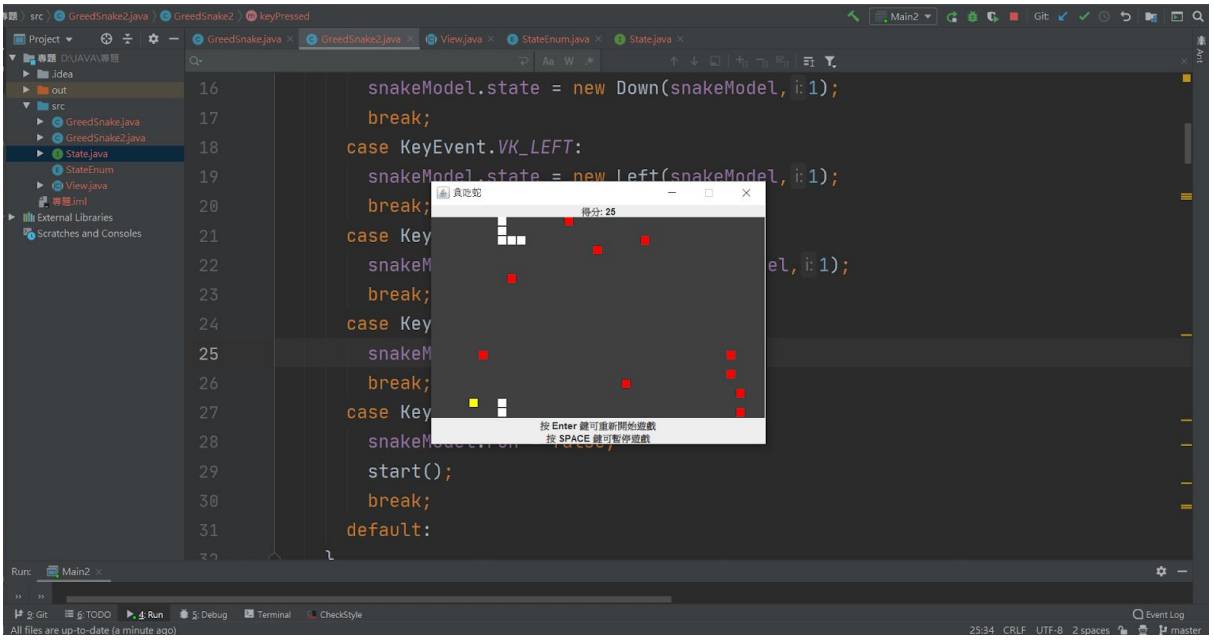


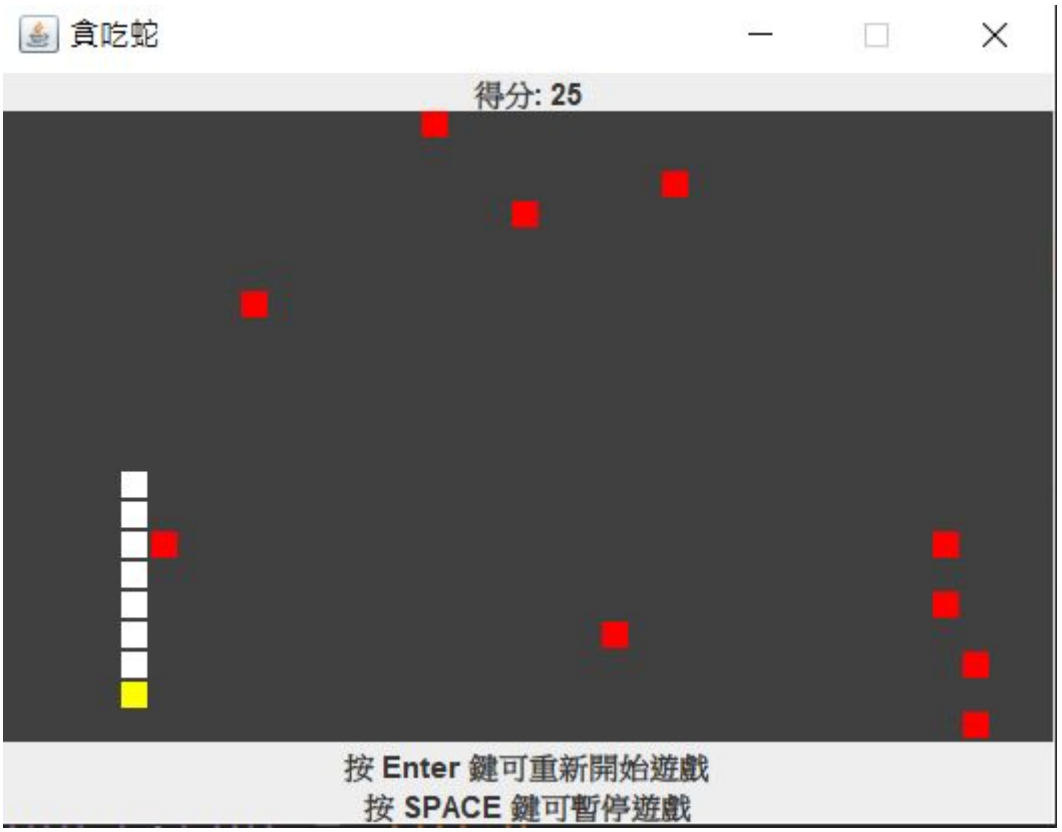




Application 2:

一樣以上下左右鍵做改變，只是規則改變，吃到紅色毒藥或碰掉自己身體才算死亡，若超出範圍會從另一端顯示並繼續進行遊戲




貪吃蛇
得分: 25
按 Enter 鍵可重新開始遊戲
按 SPACE 鍵可暫停遊戲


貪吃蛇
得分: 70
Game Over
分數 : 70
GAME OVER
OK
按 Enter 鍵可重新開始遊戲
按 SPACE 鍵可暫停遊戲

2.完成度說明：

應該要製作開始介面給玩家自行選定遊玩哪一種模式，這是我們沒有做到的，且snake在吃到食物時會有些許延遲會影響操作。最嚴重的是，我們在實作時，View和Controller做到最後變成合在一起，沒有完全遵守好MVC架構流程。

3.程式碼長度說明：

Framework + Application1 + Application2 + State +StateEnum
= 129 + 143 + 185 + 96 + 3 = 556

5.心得：

楊翔竣：其實一開始我不是很懂老師想要我們做甚麼，是慢慢找資料，慢慢去實作才越來越有感覺，實作中也會遇到各式各樣的問題，像是該用甚麼API、該如何設計符合我們的框架等，這樣的過程我覺得是很有意義的，沒有實作過就不會真正的去理解它。今天聽了別組的報告，使我對framework又有更深的理解，這是我們以後去公司基本上都會使用到的東西，我覺得可以在學校中學習這種未來會廣泛運用到的課程，真的會對學生很有幫助。

6.原始碼：

Framework:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.util.LinkedList;
import java.util.Random;

public abstract class View extends KeyAdapter {
 //遊戲介面
 JFrame mainFrame;
```

```
 Canvas canvas;
 JLabel score;
 //活動範圍
 public static final int overWidth = 420;
 public static final int overHeight = 252;
 //節點大小
 public static final int width = 12;
 public static final int height = 12;
 //Applications實作
 public abstract void keyPressed(KeyEvent e);
 public abstract void paint();
 public abstract void score();
 public abstract void start();

 public View() {
   mainFrame = new JFrame("貪吃蛇");
   mainFrame.setLocation(550,250);
   score = new JLabel("得分:", JLabel.CENTER);
   mainFrame.add(score, BorderLayout.NORTH);
   canvas = new Canvas();
   Container contain = mainFrame.getContentPane();
   canvas.setSize(overWidth+1,overHeight+1);
   canvas.addKeyListener(this);
   contain.add(canvas, BorderLayout.CENTER);

   JPanel panel = new JPanel();  //規劃幫助資訊版面
   panel.setLayout(new BorderLayout());
   JLabel help;
   help = new JLabel("按 Enter 鍵可重新開始遊戲", JLabel.CENTER);
   panel.add(help, BorderLayout.CENTER);
   help = new JLabel("按 SPACE 鍵可暫停遊戲", JLabel.CENTER);
   panel.add(help, BorderLayout.SOUTH);
   mainFrame.add(panel, BorderLayout.SOUTH);

   mainFrame.addKeyListener(this); //鍵盤處理
   mainFrame.pack();  //介面尺寸
   mainFrame.setResizable(false);  //設定視窗大小不能變化
   mainFrame.setVisible(true); //視窗可見
   mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //退出
應用程式
 }
}
```

```
abstract class Model implements Runnable {
 View snake;
 Node food;
 LinkedList link = new LinkedList(); //snake body

 int speed = 120;
 //最大範圍
 int maxX;
 int maxY;
 int x = 0;
 int y = 0;
 int score = 0;
 boolean[][] exist;  //判斷該位置是否有節點
 //遊戲狀態
 boolean run = false;
 boolean pause = false;
 State state;
 StateEnum direct = StateEnum.Up;
 //Applications實作
 public abstract boolean move();

 //run():貪吃蛇運動執行緒
 public void run() {
   run = true;
   while(run) {
     try {
       //設定速度
       Thread.sleep(speed);
     }
     catch(Exception e) {
       break;
     }

     if(!pause) {
       if(move())  //暫停
         snake.paint();
       else { //遊戲結束
         JOptionPane.showMessageDialog(null,
             "分數 : " + score + "\nGAME OVER",
             "Game Over", JOptionPane.ERROR_MESSAGE);
         break;
```

```
        }
      }
    }
    run = false;
  }

  //createFood():生成食物
  Node createFood() {
    int x = 0;
    int y = 0;
    do {
      Random rand = new Random();
      x = rand.nextInt(maxX);
      y = rand.nextInt(maxY);
    } while(exist[x][y]);
    return new Node(x,y);
  }

  public void speedUp() {
    speed -= 5;
  }

  public void changePause() {
    pause = !pause;
  }
}

//生成圖片基本元素
class Node {
  int x;
  int y;
  public Node(int x,int y) {
    this.x = x;
    this.y = y;
  }
}
```

Application1:

```
import java.util.*;
import java.awt.*;
import java.awt.event.*;

public class GreedSnake extends View {
 SnakeModel snakeModel = null;

 //keyPressed():按鍵
 public void keyPressed(KeyEvent e) {
   int key = e.getKeyCode();
   switch(key) {
     case KeyEvent.VK_UP:
       snakeModel.state = new Up(snakeModel,1);
       break;
     case KeyEvent.VK_DOWN:
       snakeModel.state = new Down(snakeModel,1);
       break;
     case KeyEvent.VK_LEFT:
       snakeModel.state = new Left(snakeModel,1);
       break;
     case KeyEvent.VK_RIGHT:
       snakeModel.state = new Right(snakeModel,1);
       break;
     case KeyEvent.VK_SPACE:
       snakeModel.changePause();
       break;
     case KeyEvent.VK_ENTER:
       snakeModel.run = false;
       start();
       break;
     default:
   }
 }

 //paint():繪製遊戲
 public void paint() {
   Graphics g = canvas.getGraphics();
   //背景
   g.setColor(Color.DARK_GRAY);
   g.fillRect(0,0,overWidth,overHeight);
   //snake
```

```
    g.setColor(Color.WHITE);
    LinkedList list = snakeModel.link;
    Iterator it = list.iterator();
    while(it.hasNext()) {
      Node n = (Node)it.next();
      g.fillRect(n.x * width,n.y * height,
          width - 1,height - 1);
    }
    //food
    g.setColor(Color.yellow);
    Node n = snakeModel.food;
    g.fillRect(n.x*width,n.y*height,width-1,height-1);
    score();
  }

  //score():改變計分牌
  public void score() {
    String s = "得分: " + snakeModel.score;
    score.setText(s);
  }

  //start():遊戲開始,放置貪吃蛇
  public void start() {
    if(snakeModel == null || !snakeModel.run) {
      snakeModel = new SnakeModel(this,overWidth/width,
          overHeight/height);
      //狀態已running呈現
      (new Thread(snakeModel)).start();
    }
  }
}

class SnakeModel extends Model {
  //SnakeModel():初始化遊戲
  public SnakeModel(GreedSnake snake, int maxX, int maxY) {
    this.snake = snake;
    this.maxX = maxX;
    this.maxY = maxY;
    exist = new boolean[maxX][];
    for(int i = 0; i < maxX; i++) {
      exist[i] = new boolean[maxY];
      Arrays.fill(exist[i],false);  //沒有蛇和食物的節點設false
```

```
  }
  int length = maxX > 20 ? 10 : maxX / 2;
  for(int i = 0; i < length; i++) {
    x = maxX / 2 + i;
    y = maxY / 2;
    link.addLast(new Node(x,y));
    exist[x][y] = true; //蛇身位置設為true
  }
  food = createFood();
  exist[food.x][food.y] = true; //食物位置設為true
}

//move():碰撞事件
public boolean move() {
  switch (direct) {
    case Up:
      state = new Up(this, 2);
      break;
    case Down:
      state = new Down(this, 2);
      break;
    case Left:
      state = new Left(this, 2);
      break;
    case Right:
      state = new Right(this, 2);
      break;
    default:
  }

  if ((0 <= x && x < maxX) && (0 <= y && y < maxY)) {
    if (exist[x][y]) { //吃到食物或撞到身體
      if (x == food.x && y == food.y) {
        link.addFirst(food);  //在首位加節點
        score += 125 - speed + 5;
        speedUp();
        //重新生成食物
        food = createFood();
        exist[food.x][food.y] = true;
        return true;
      } else return false;
    } else {
```

```
        //加頭去尾，運動狀態
        link.addFirst(new Node(x, y));
        exist[x][y] = true;
        Node n = (Node) link.removeLast();
        exist[n.x][n.y] = false;
        return true;
      }
    }
    return false; //撞到牆
  }
}

class Main {
  public static void main(String[] args) {
    GreedSnake greedSnake = new GreedSnake();
    greedSnake.start();
  }
}
```

Application2:

```
import java.util.*;
import java.awt.*;
import java.awt.event.*;

public class GreedSnake2 extends View {
  SnakeModel2 snakeModel = null;

  //keyPressed():按鍵
  public void keyPressed(KeyEvent e) {
    int key = e.getKeyCode();
    switch(key) {
      case KeyEvent.VK_UP:
        snakeModel.state = new Up(snakeModel,1);
        break;
      case KeyEvent.VK_DOWN:
        snakeModel.state = new Down(snakeModel,1);
        break;
      case KeyEvent.VK_LEFT:
        snakeModel.state = new Left(snakeModel,1);
```

```
      break;
    case KeyEvent.VK_RIGHT:
      snakeModel.state = new Right(snakeModel,1);
      break;
    case KeyEvent.VK_SPACE:
      snakeModel.changePause();
      break;
    case KeyEvent.VK_ENTER:
      snakeModel.run = false;
      start();
      break;
    default:
  }
}

//paint():繪製遊戲
public void paint() {
  Node n;
  Graphics g = canvas.getGraphics();
  //背景
  g.setColor(Color.DARK_GRAY);
  g.fillRect(0,0,overWidth,overHeight);
  //snake
  g.setColor(Color.WHITE);
  LinkedList link = snakeModel.link;
  Iterator it = link.iterator();
  while(it.hasNext()) {
    n = (Node)it.next();
    g.fillRect(n.x * width,n.y * height,
        width - 1,height - 1);
  }

  //毒藥
  g.setColor(Color.red);
  for(int i =0; i < 10; i++) {
    n = snakeModel.dead[i];
    g.fillRect(n.x*width,n.y*height,width-1,height-1);
  }

  //food
  g.setColor(Color.yellow);
  n = snakeModel.food;
```

```
    g.fillRect(n.x*width,n.y*height,width-1,height-1);
    score();
  }

  //score():改變計分牌
  public void score() {
    String s = "得分: " + snakeModel.score;
    score.setText(s);
  }

  //start():遊戲開始,放置貪吃蛇
  public void start() {
    if(snakeModel == null || !snakeModel.run) {
      snakeModel = new SnakeModel2(this,overWidth/width,
          overHeight/height);
      //狀態已running呈現
      (new Thread(snakeModel)).start();
    }
  }
}

class SnakeModel2 extends Model {
  Node[] dead = new Node[10];

  //SnakeModel2():初始化遊戲
  public SnakeModel2(GreedSnake2 snake, int maxX, int maxY) {
    this.snake = snake;
    this.maxX = maxX;
    this.maxY = maxY;
    exist = new boolean[maxX][];
    for(int i = 0; i < maxX; i++) {
      exist[i] = new boolean[maxY];
      Arrays.fill(exist[i],false);  //沒有蛇和食物的節點設false
    }

    int length = maxX > 20 ? 5 : maxX / 2;
    for(int i = 0; i < length; i++) {
      x = maxX / 2 + i;
      y = maxY / 2;
      link.addLast(new Node(x,y));
      exist[x][y] = true; //蛇身位置設為true
    }
```

```
  for(int i = 0; i < 10; i++) {
    dead[i] = createDead();
    exist[dead[i].x][dead[i].y] = true;
  }

  food = createFood();
  exist[food.x][food.y] = true; //食物位置設為true
}

//move():碰撞事件
public boolean move() {
  switch (direct) {
    case Up:
      state = new Up(this, 2);
      break;
    case Down:
      state = new Down(this, 2);
      break;
    case Left:
      state = new Left(this, 2);
      break;
    case Right:
      state = new Right(this, 2);
      break;
    default:
  }

  if ((0 <= x && x < maxX) && (0 <= y && y < maxY)) {
    if (exist[x][y]) { //吃到食物或撞到身體
      if (x == food.x && y == food.y) {
        link.addFirst(food);  //在首位加節點
        score += 125 - speed + 5;
        speedUp();
        //重新生成食物
        food = createFood();
        exist[food.x][food.y] = true;
        return true;
      } else return false;
    } else {
      //加頭去尾，運動狀態
      link.addFirst(new Node(x, y));
```

```
        exist[x][y] = true;
        Node n = (Node) link.removeLast();
        exist[n.x][n.y] = false;
        return true;
      }
    } else {  //超出範圍
      if (x == maxX && direct == StateEnum.Right)
        x = 0;
      if (x == -1 && direct == StateEnum.Left)
        x = 34;
      if (y == maxY && direct == StateEnum.Down)
        y = 0;
      if (y == -1 && direct == StateEnum.Up)
        y = 20;
      link.addFirst(new Node(x, y));
      exist[x][y] = true;
      Node n = (Node) link.removeLast();
      exist[n.x][n.y] = false;
      return true;
    }
  }

  protected Node createDead() {
    int x = 0;
    int y = 0;
    do {
      Random rand = new Random();
      x = rand.nextInt(maxX);
      y = rand.nextInt(maxY);
    } while(exist[x][y]);
    return new Node(x,y);
  }
}

class Main2 {
  public static void main(String[] args) {
    GreedSnake2 greedSnake = new GreedSnake2();
    greedSnake.start();
  }
}
```

State:

```
interface State {
 void change(Model snake); //避免衝突
 void coordinate(Model snake); //改變snake x,y
}

class Up implements State {
 public Up(Model snake, int i) {
   if(i == 1)
     change(snake);
   else if(i == 2)
     coordinate(snake);
 }

 public void change(Model snake) {
   if(snake.direct != StateEnum.Down) {
     snake.direct = StateEnum.Up;
   }
 }

 public void coordinate(Model snake) {
   Node n = (Node)snake.link.getFirst();
   int y = n.y;
   y--;
   snake.x = n.x;
   snake.y = y;
 }
}

class Down implements State {
 public Down(Model snake, int i) {
   if(i == 1)
     change(snake);
   else if(i == 2)
     coordinate(snake);
 }

 public void change(Model snake) {
   if(snake.direct != StateEnum.Up) {
     snake.direct = StateEnum.Down;
   }
 }
```

```
 public void coordinate(Model snake) {
   Node n = (Node)snake.link.getFirst();
   int y = n.y;
   y++;
   snake.x = n.x;
   snake.y = y;
 }
}

class Left implements State {
 public Left(Model snake, int i) {
   if(i == 1)
     change(snake);
   else if(i == 2)
     coordinate(snake);
 }

 public void change(Model snake) {
   if(snake.direct != StateEnum.Right) {
     snake.direct = StateEnum.Left;
   }
 }

 public void coordinate(Model snake) {
   Node n = (Node)snake.link.getFirst();
   int x = n.x;
   x--;
   snake.x = x;
   snake.y = n.y;
 }
}

class Right implements State {
 public Right(Model snake, int i) {
   if(i == 1)
     change(snake);
   else if(i == 2)
     coordinate(snake);
 }

 public void change(Model snake) {
```

```
    if(snake.direct != StateEnum.Left) {
      snake.direct = StateEnum.Right;
    }
  }

  public void coordinate(Model snake) {
    Node n = (Node)snake.link.getFirst();
    int x = n.x;
    x++;
    snake.x = x;
    snake.y = n.y;
  }
}
```

StateEnum:

```
public enum StateEnum {
  Up,Down,Left,Right
}
```