

# 实验报告

实验名称	实验 6 综合实践				实验时间	第 16 周	
实验环境							
Eclipse/Jcreator Pro/JDK。							
实验目的和内容要求							
<div>一、实验目的</div> <div>1. 掌握 JAVA 语言的综合使用；</div> <div>2. 能够使用 JAVA 语言实现简单项目。</div> <div>二、实验要求</div> <div>1. 从实验内容中选择一道完成。</div> <div>2. 提供所完成的题目的主要实验代码和运行结果的界面截图。</div> <div>三、实验内容</div> <div>1. 警察抓小偷</div> <div>公元 2222 年，赛博飞警发现了蒙面怪盗老 K，现在需要你为飞警提供燃料，将怪盗老 K 绳之以法，你能顺利执行任务吗？</div> <div>主要规则如下：</div> <div>(1) 游戏空间为封闭的环状道路，屏幕显示随机生成的字符串；</div> <div>(2) 小偷以一定的速度自动向前移动，而警察需要通过玩家键入与字符串相匹配的字符向前追赶；</div> <div>(3) 完成一段字符串，则自动生成新的字符串；</div> <div>(4) 胜利条件：警察抓到小偷（移动到小偷所在坐标）；</div> <div>(5) 失败条件：在规定的时间内未抓到小偷。</div> <div>2. 贪吃蛇大作战</div> <div>“太贪吃”和“大迷糊”是两条贪吃蛇。饥饿的他们来到一块苹果园，你能帮他们找到苹果吗？</div> <div>主要规则如下：</div> <div>(1) 游戏会在界面中不同的位置随机生成苹果，但同一时刻界面中只会存在一个苹果；</div> <div>(2) 玩家可以通过“上下左右”方向键控制“太贪吃”的移动方向，寻找苹果，每吃一个苹果就能获得一定的积分，同时“太贪吃”的身体会随着吃到苹果数量的增加而变长。</div> <div>(3) “大迷糊”不能被玩家控制，在场景里会随机移动，“太贪吃”在寻找苹果的过程中需要避开“大迷糊”。</div> <div>(4) 失败条件：蛇头碰到蛇身、蛇头碰到边界、“太贪吃”碰到“大迷糊”。</div> <div>3. 沙漏棋</div> <div>PiPi 设计了一种新的棋类游戏，其棋盘与棋子摆放如图 1 所示。因其独特的沙漏状棋盘，PiPi 将这种棋命名为“沙漏棋”。你能帮 PiPi 实现他的“沙漏棋”吗？</div> <div>主要规则如下：</div>							

图 1 沙漏棋

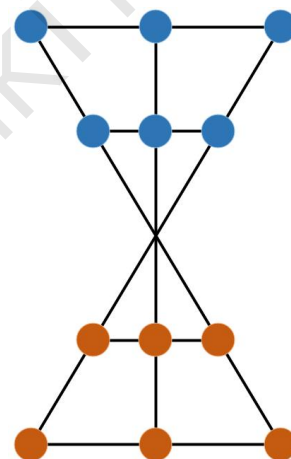


图 1 沙漏棋

- (1) 正常走棋只能向前或水平方向移动一步，不能后退；
- (2) 当同一直线的相邻位置有对方棋子，且该对方棋子背后有空位时可以从对方棋子上跳过，并吃掉被跳过的棋子；
- (3) 吃棋时可以后退吃棋，支持连跳连吃；
- (4) 胜利条件：将对方棋子全部吃光；

#### 四、实验思考

1. 简述在实验过程中遇到的问题与解决方法。
2. 简述实验过程中的发现与收获，未解决或需进一步解决的问题。

#### 实验过程、结果分析与总结

## 我选题为 《2. 贪吃蛇大作战》

### 1. 程序结构

程序主要部分为：

接口 SnakeDirection 把蛇的运动方向转为数字，方便后续使用

类 StartGame 用于开始游戏

类 SnakeFrame 继承了 JFrame 来绘制图形化界面。这个类里面包含了它的构造函数、建立界面、设置监听、setBackgroundimage 函数设置图片图标、一个继承了 JPanel 的内部类 snakepanel 来计算得分和最高分、画蛇头和食物、绘制新线程、定时刷新界面、两条蛇的移动、判断撞到东西等

### 2. 设计思路

(1) NPC 蛇：我运用了随机数的方法，把蛇的运动方向转化为数字，便可使 NPC 蛇在画面中到处移动

(2) 主角蛇：监听来控制蛇的移动，坐标与事物的坐标相同时身体增长，坐标超出画框范围，以及坐标等于 NPC 蛇的坐标时时游戏结束

### 3. 程序源码

StartGame.java

```
package com.snake.ui;

public class StartGame {

    public static void main(String[] args) {

        new SnakeFrame();

    }

}
```

## SnakeFrame.java

```
package com.snake.ui;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.LinkedList;
import java.util.Random;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.Timer;

import com.snake.utils.PropertiesUtils;

/**
 * @author
 *
 */
public class SnakeFrame extends JFrame{

    private static final long serialVersionUID = 8866826595307493727L;
    //宽度
    private static final int WIDTH = 800; //
    //高度
    private static final int HEIGHT = 600;
    //通用综合长度
    private static final int CELL = 20;
    //蛇头
    private JLabel snakeHeader;
    //大迷糊
    private JLabel bigSnakeHeader;
    //食物
    private JLabel fruit;
    //随机数
    private Random random = new Random();
```

```

//键盘按键
private int dir = 1;

private int big = 1;
//蛇身
private LinkedList<JLabel> bodies = new LinkedList<JLabel>();
//大迷糊
private LinkedList<JLabel> bigBodies = new LinkedList<JLabel>();
//得分图片
private String[] fruits = {"apple.png"};
//蛇身图片
private String[] snakeBody = {"green.png", "red.png", "yellow.png", "purple.png"};
//大迷糊
private String[] bigSnakeBody = {"red.png"};
//历史 得分
private JLabel highestLabel;
private JLabel currentLabel;
//当前得分
private int highestScore;
private int currentScore;
//历史得分配置
private PropertiesUtils prop = PropertiesUtils.getInstance();
//蛇动作刷新定时器
private Timer timer;

private boolean status = true;

public SnakeFrame() {
    //设置窗体属性
    ImageIcon icon = new ImageIcon("../src/com/snake/images/snake.jpg");
    this.setIconImage(icon.getImage());
    this.setTitle("贪吃蛇");
    this.setSize(WIDTH+4, HEIGHT+34);
    this.setLocationRelativeTo(null);
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    this.setResizable(false);
    this.setLayout(null);

    SnakePanel snakePanel = new SnakePanel();
    snakePanel.setBounds(0, 0, WIDTH, HEIGHT);
    this.add(snakePanel);

    //监听键盘输入
    this.addKeyListener(new KeyAdapter() {

```

```

@Override
public void keyPressed(KeyEvent e) {

    int keyCode = e.getKeyCode();

    //监听对应的按键 更改移动方向和图片
    switch (keyCode) {
        //左键
        case KeyEvent.VK_LEFT:
            if(dir != SnakeDirection.RIGHT){
                dir = SnakeDirection.LEFT;
                setBackgroundImage(snakeHeader, "header_l.png");
            }
            break;
        //右键
        case KeyEvent.VK_RIGHT:
            if(dir != SnakeDirection.LEFT){
                dir = SnakeDirection.RIGHT;
                setBackgroundImage(snakeHeader, "header_r.png");
            }
            break;
        //上键
        case KeyEvent.VK_UP:
            if(dir != SnakeDirection.BOTTOM){
                dir = SnakeDirection.TOP;
                setBackgroundImage(snakeHeader, "header_t.png");
            }
            break;
        //下键
        case KeyEvent.VK_DOWN:
            if(dir != SnakeDirection.TOP){
                dir = SnakeDirection.BOTTOM;
                setBackgroundImage(snakeHeader, "header_b.png");
            }
            break;
        case KeyEvent.VK_SPACE:
            // if(status){
            //     status = !status;
            // }else{
            //     status = !status;
            //     timer.notify();
            // }
            default:

    }

}

});

```

```

        this.setVisible(true);

    }

    /**
     * 设置指定的 label 的图标
     * @param label 标签
     * @param fileName 文件名
     */
    private void setBackgroundImage(JLabel label, String fileName) {
        ImageIcon icon = new ImageIcon("./src/com/snake/images/"+fileName);
        // 设置图片图标
        icon.setImage( icon.getImage());

        getScaledInstance(label.getWidth(), label.getHeight(), Image.SCALE_DEFAULT));
        label.setIcon(icon);
    }

    class SnakePanel extends JPanel{

        private static final long serialVersionUID = 1L;
        //构造器
        public SnakePanel() {
            init();
        }

        /**
         * 初始化
         */
        private void init() {
            //窗体大小
            this.setSize(SnakeFrame.WIDTH, SnakeFrame.HEIGHT);
            this.setLayout(null);
            //得分 label
            highestLabel = new JLabel();
            highestScore = Integer.parseInt(prop.getProperty("highest"));
            highestLabel.setText("历史最高分"+highestScore);
            highestLabel.setBounds(20, 20, 300, 30);
            this.add(highestLabel);
            //当前得分 label
            currentLabel = new JLabel("当前得分"+currentScore);
            currentLabel.setBounds(20, 60, 300, 30);
        }
    }

```

```

this.add(currentLabel);
//绘制蛇头
createHeader();
createBigHeader();
//新建线程绘制
new Thread(new Runnable() {
    @Override
    public void run() {
        createFruit();
    }
}).start();
timer = new Timer(200, new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
    }
});

//定时刷新绘制界面
timer = new Timer(200, new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        //获取蛇位置
        Point oldPoint = snakeHeader.getLocation();
        Point newPoint = null;
        //监听最后一次按钮的按键
        switch (dir) {
            case SnakeDirection.RIGHT: //右 移动坐标
                newPoint = new Point(oldPoint.x+CELL, oldPoint.y);
                break;
            case SnakeDirection.LEFT: //左
                newPoint = new Point(oldPoint.x-CELL, oldPoint.y);
                break;
            case SnakeDirection.BOTTOM: //下
                newPoint = new Point(oldPoint.x, oldPoint.y+CELL);
                break;
            case SnakeDirection.TOP: //上
                newPoint = new Point(oldPoint.x, oldPoint.y-CELL);
                break;
        }
        //移动完以后刷新坐标
        snakeHeader.setLocation(newPoint);
        isHeatWall();
        //判断是否吃到食物
        if(snakeHeader.getLocation().equals(fruit.getLocation())){
            eatBean();
        }
    }
});

```

```

//移动蛇的位置
move(oldPoint,bodies);

//大迷糊
Point bigOldPoint = bigSnakeHeader.getLocation();

Point bigNewPoint =null;
boolean flag = false;
int u=0;
do {
    //判断死循环 跳出
    u++;
    if(u > CELL){
        break;
    }
    flag = false;
    int i = random.nextInt(4);
    //判断反方向
    while
((big==0&&i==1) || (big==1&&i==0) || (big==2&&i==3) || (big==3&&i==2)) {
        i = random.nextInt(4);
    }
    big = i;
    switch (big) {
        case 3: //右 移动坐标
            bigNewPoint = new Point(bigOldPoint.x+CELL,
bigOldPoint.y);
            break;
        case 2: //左
            bigNewPoint = new Point(bigOldPoint.x-CELL,
bigOldPoint.y);
            break;
        case 1: //下
            bigNewPoint = new Point(bigOldPoint.x,
bigOldPoint.y+CELL);
            break;
        case 0: //上
            bigNewPoint = new Point(bigOldPoint.x,
bigOldPoint.y-CELL);
            break;
        default:
    }
    int x = bigNewPoint.getLocation().x;
    int y = bigNewPoint.getLocation().y;
    //撞墙
    if(x <20 || x >760 || y<20 || y>560){
        flag = true;
    }
}

```



```

    }

    // 大迷糊 自己撞自己
    for(int ii=1;ii<bigBodies.size();ii++){
        JLabel j = bigBodies.get(ii);
        int jx = j.getLocation().x;
        int jy = j.getLocation().y;
        if(x == (jx) && y == (jy)){
            flag = true;
            break;
        }
    }
    }while (flag);
    //设置位置并移动
    bigSnakeHeader.setLocation(bigNewPoint);
    move(bigOldPoint, bigBodies);
}

});
timer.start();

}

/**
 * 移动位置 将每个位置都向后面的位置迭代一次
 * @param oldPoint
 */
private void move(Point oldPoint, LinkedList<JLabel> bodies) {
    Point p = new Point();
    //遍历 整条蛇
    for(int i=1;i<bodies.size();i++){
        p = bodies.get(i).getLocation();
        bodies.get(i).setLocation(oldPoint);
        oldPoint = p;
    }
}

/**
 * 吃到食物的判定
 */
private void eatBean() {
    // 随机获取图片
    int index = random.nextInt( snakeBody.length);

```

```

        setBackgroundImage(fruit, snakeBody[index]);
        bodies.add(fruit);

        //增加得分
        currentScore++;
        currentLabel.setText("当前得分"+currentScore);

//
        new Thread( new Runnable() {
//
//
//            @Override
//            public void run() {
//
//                createFruit();
//            }
//        }).start();
        //创建新的食物
        new Thread( ()->{createFruit();}).start();

    }

    /**
     * 判断是否撞到墙 或 特定物体 结束游戏
     */
    private void isHeatWall() {
        // 获取 x y 轴坐标
        int x = snakeHeader.getLocation().x;
        int y = snakeHeader.getLocation().y;
        boolean flag= false;
        //自己撞墙
        if(x <0 || x >780 || y<0 || y>580){
            flag = true;
        }
        //自己撞自己
        for(int i=1;i<bodies.size();i++){
            JLabel j = bodies.get(i);
            int jx = j.getLocation().x;
            int jy = j.getLocation().y;
            if(x == jx && y == jy){
                flag = true;
            }
        }
        //自己撞大迷糊
        for(int i=0;i<bigBodies.size();i++){
            JLabel j = bigBodies.get(i);
            int jx = j.getLocation().x;
            int jy = j.getLocation().y;

```

```

        if(x == jx && y == jy){
            flag = true;
        }
    }
    if(flag){

        int op = -1;
        if(currentScore > highestScore){
            //新纪录
            op = JOptionPane.showConfirmDialog(null, "新纪录");
            prop.setProperty("highest", currentScore+"");
            try {
                //写入最新的得分
                FileWriter writer = new FileWriter(new
File("./src/score.properties"));
                prop.store(writer, null);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }else{
            op = JOptionPane.showConfirmDialog(null, "你死了");
        }

        //弹窗选择 是
        if(op == 0){
            reStart();
        }else{
            System.exit(0);
        }
    }
}

/**
 * 绘制 食物
 */
private void createFruit() {
    //创建得分标识
    fruit = new JLabel();
    //设置大小
    fruit.setSize(CELL, CELL);
    //随机下标
    int index = random.nextInt( fruits.length);
    //设置图片
    setBackgroundImage(fruit, fruits[index]);
    // 随机出现位置
    Point p = randomPoint(SnakeFrame.WIDTH/CELL, SnakeFrame.HEIGHT/CELL);

```

```

        System.out.println("x:"+p.x+" y:"+p.y);
        //设置位置
        fruit.setLocation(p);

        this.add(fruit);
        //绘制
        this.repaint();
    }

    /**
     * 绘制蛇头
     */
    private void createHeader() {
        snakeHeader = new JLabel();
        //设置大小
        snakeHeader.setSize(CELL, CELL);
        //设置透明度
        snakeHeader.setOpaque(false);
        //图片
        setBackgroundImage(snakeHeader, "header_r.png");
        //位置
        Point p = randomPoint((SnakeFrame.WIDTH/CELL)/2,
        (SnakeFrame.HEIGHT/CELL)/2);
        p.x = p.x+10*CELL;
        p.y = p.y+10*CELL;
        //设置蛇头位置
        snakeHeader.setLocation(p);
        bodies.add(snakeHeader);
        this.add(snakeHeader);
    }

    /**
     * 绘制
     */
    private void createBigHeader() {
        bigSnakeHeader = new JLabel();
        //设置大小
        bigSnakeHeader.setSize(CELL, CELL);
        //设置透明度
        bigSnakeHeader.setOpaque(false);
        //图片
        setBackgroundImage(bigSnakeHeader, "header_b.png");
        //位置
        Point p = randomPoint((SnakeFrame.WIDTH/CELL),
        (SnakeFrame.HEIGHT/CELL));
        //设置蛇头位置
        bigSnakeHeader.setLocation(p);
        bigBodies.add(bigSnakeHeader);
        this.add(bigSnakeHeader);
    }

```

```

        for(int i=0;i<10;i++){
            // 随机获取图片
            int index = random.nextInt( snakeBody.length);

            //创建得分标识
            JLabel fruit = new JLabel();
            //设置大小
            fruit.setSize(CELL, CELL);
            setBackgroundImage(fruit, bigSnakeBody[0]);
            // 随机出现位置
            Point pp = new Point(p.x+(CELL * (i+1)),p.y);
            //设置位置
            fruit.setLocation(pp);
            bigBodies.add(fruit);
            this.add(fruit);
            //绘制
        }
    // this.repaint();
}

/**
 * 随机坐标
 * @param xScale
 * @param yScale
 * @return
 */
private Point randomPoint(int xScale,int yScale){
    //获取随机的 x 轴 y 轴 坐标点
    Point point = new Point();
    int x = random.nextInt(xScale)*CELL;
    int y = random.nextInt(yScale)*CELL;
    point.setLocation(x, y);
    return point;
}

/**
 * 重绘制方法
 * @param g
 */
@Override
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    //绘制背景
    ImageIcon icon = new ImageIcon("./src/com/snake/images/bg.png");
    g.drawImage(icon.getImage(), 0, 0, SnakeFrame.WIDTH, SnakeFrame.HEIGHT,

```

```

null);

//设置颜色
g.setColor(Color.RED);
//绘制 宽 搞 网格
for(int i=1;i<HEIGHT/CELL;i++){
    g.drawLine(0, i*CELL, 800, i*CELL);
}

for(int i=1;i<WIDTH/CELL; i++){
    g.drawLine(i*CELL, 0, i*CELL, 600);
}

}

/**
 * 重新开始
 */
public void reStart() {

    if(currentScore > highestScore){
        highestScore = currentScore;
        highestLabel.setText("最高分"+highestScore);
    }
    currentScore = 0;
    currentLabel.setText("当前得分"+currentScore);
    //清除 界面上的东西 重置属性
    dir = 1;
    this.remove(fruit);
    for(JLabel body : bodies){
        this.remove(body);
    }
    for(JLabel body : bigBodies){
        this.remove(body);
    }
    bodies.clear();
    bigBodies.clear();
    //绘制界面上的东西
    createHeader();
    createBigHeader();
    createFruit();

    super.repaint();

}

```

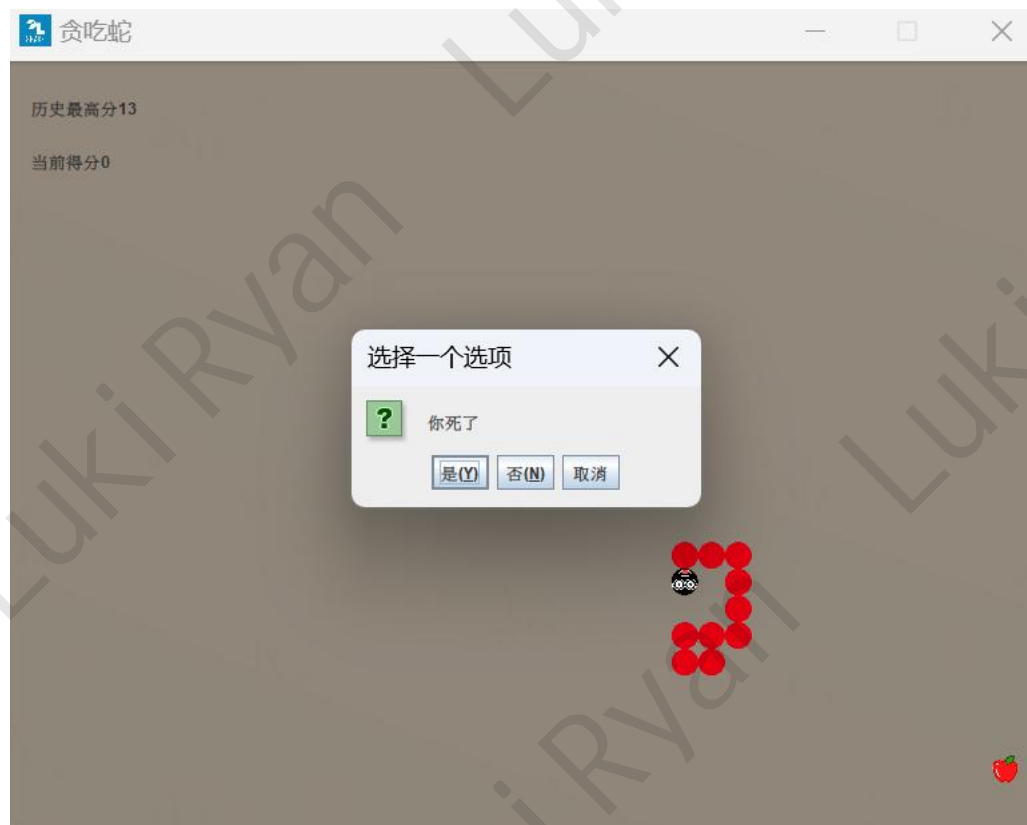
```
}  
}
```

### SnakeDirection.java

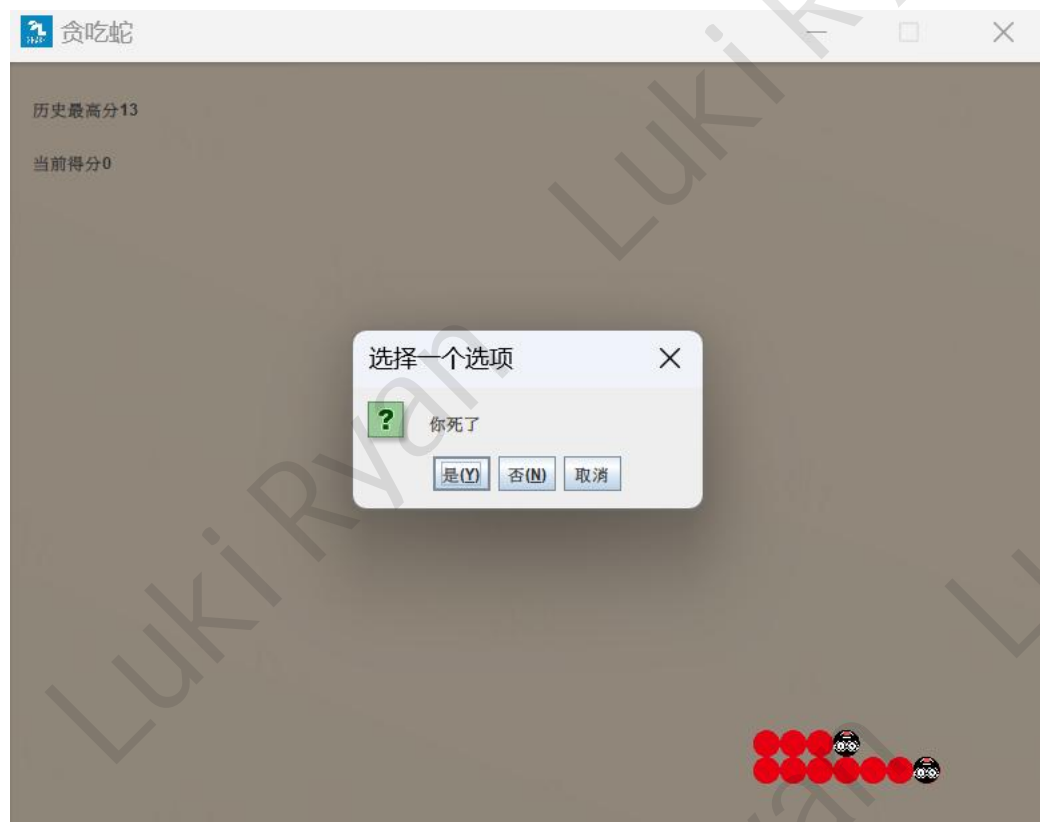
```
package com.snake.ui;  
  
/**  
 * 存储蛇运动方向的接口  
 * @author  
 *  
 */  
public interface SnakeDirection {  
  
    int LEFT = -1;  
  
    int RIGHT = 1;  
  
    int BOTTOM = -2;  
  
    int TOP = 2;  
  
}
```

## 4. 运行结果

### 4.1 撞墙死亡



#### 4.2 撞 NPC 死亡



#### 4.3 吃了食物后变长



指导老师评阅意见



指导老师： 年 月 日

填写内容时，可把表格扩大。