北京邮电大学软件学院

实验报告

课程名称： Java SE 程 序 设 计

项目名称： Java程序编制（基础练习）

项目完成人：

姓名： 罗舒婷

学号： 2017211951

指导教师： 崔毅东

日 期： 2018 年 12 月 28 日

（正文文字，请使用【正文缩进】样式）

（标题编号请使用自动编号。在样式表中选中标题级别样式即可自动编号。）

# 实验概述

## 实验目的

学习用Java，理解基本程序设计流程，掌握Java中多线程同步、Socket通信、继承、多态、接口、异常处理等，理解面向对象程序设计，掌握对象和类，继承和多态的特征和理解GUI程序设计。

## 实验内容

|  |  |
| --- | --- |
|  | 任务说明 |
|  | 1#设计一个继承抽象基类集合类的三角形子类，实现重写 |
|  | 2#设计一个几何基类和一个接口，继承该基类和接口的正方形类实现重写 |
|  | 3#设计GUI、三个线程，第一个线程和第二个线程生成字母，第三个线程比较大小 |
|  | 4#设计两个线程实现同时接收TCP和UDP的服务器 |
|  | 4#设计TCP和UDP两种客户端，给服务器发送内容，服务器比较内容 |

## 实验环境

Windows Eclipse

### 硬件环境

64位操作系统 基于x64的处理器

### 网络环境

不需要

### 软件环境

64位操作系统

## 实验结果

|  |  |
| --- | --- |
|  | 结果说明 |
|  | 控制台输出获得面积周长函数和重写的toString函数返回值 |
|  | 控制台看见调用howtoColor成功，成功实现了基类转换派生类 |
|  | GUI和控制台都可以看见线程A和B的执行，以及分数的比较和最后赢家 |
|  | GUI和控制台都可以看见TCP和UDP发送给服务器的数据，以及在服务器对接收到的数据的比较，以及游戏规则下的（石头剪刀布）最后赢家 |

# 实验内容

# Project #1分析设计

## 1.1问题分析

设计两个类，抽象基类和继承该基类的子类，子类里实现抽象基类三个函数的重写。

### 1.1.1 目标

实现继承和重写，测试函数。

### 1.1.2功能

#### 1.1.2.1功能一：生成实例

生成三角形子类的实例。

### 1.1.3性能

#### 1.1.3.1方法可知面积、周长、三边长

在调用方法getArea()\getPermieter()\toString() respectively return the results after caculation

#### 1.1.3.2健壮性

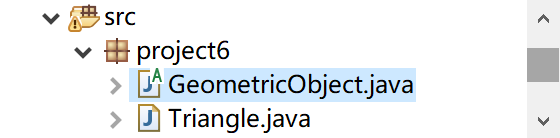
确保用户输入的是三角形，如果输入的三个边值不能生成三角形，则使用默认值边长1，1，1.

#### 1.1.3.3异常输入处理

当用户输入不符合规范时，信息框提醒用户输入不符合规范。

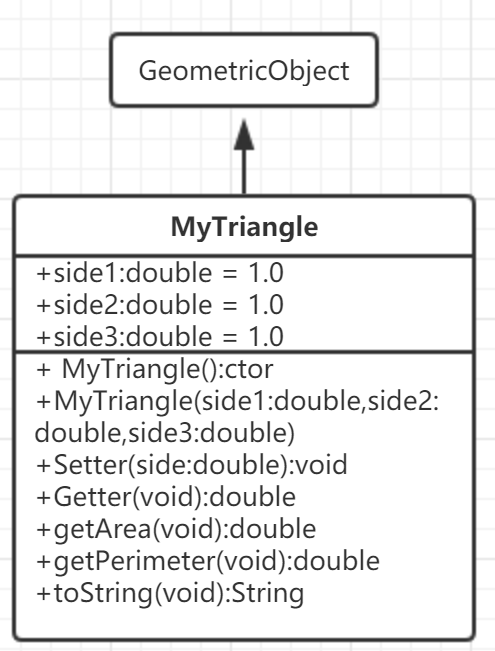
## 1.2设计方案

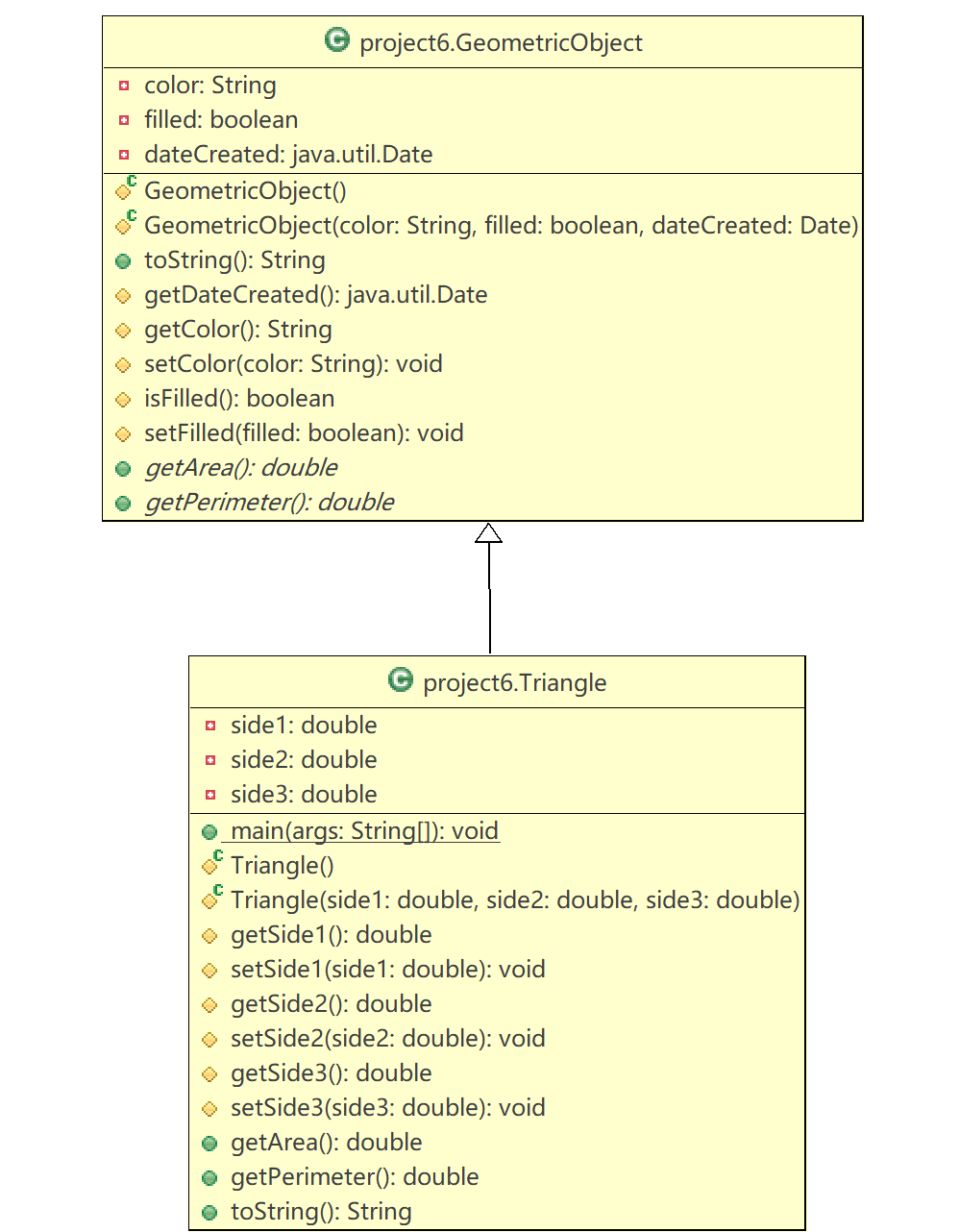
### 1.2.1模块划分



本项目包含两个模块

### 1.2.2类图





### 1.2.3算法

1.2.1.1算法一：计算面积的公式用了海伦公式，公式中三角形的面积等于三角形半周长乘以三角形半周长和三边分别的差值开根号

## 1.3实验结果

|  |  |
| --- | --- |
| 情况 | 实验结果截图 |
| 调用无参构造函数-成功；  调用有参构造函数；有参构造函数有异常输入； |  |

## 1.4 调试心得

抽象方法不能包含在非抽象类中。如果抽象父类的子类不能实现所有抽象方法，那么子类也必须定义为抽象的。三角形类不是抽象的子类，所以必须实现所有抽象方法，抽象方法是非静态的。抽象类是不能使用new操作符类初始化的，但仍然可以定义它的构造方法，这个构造方法在它的子类的构造方法中调用。可以定义一个不包含抽象方法的抽象类，在这种情况下不能使用new操作符创建该类的实例。这种类是用来定义新子类的基类的。

## 1.5源程序

/\*

\* GeometricObject.java

\*/

**package** project6;

**import** java.util.Date;

**public** **abstract** **class** GeometricObject {

**private** String color = "white";

**private** **boolean** filled;

**private** java.util.Date dateCreated;

/\*\* Construct a default geometric object \*/

**protected** GeometricObject() {

dateCreated = **new** java.util.Date();

}

**protected** GeometricObject(String color, **boolean** filled, Date dateCreated) {

**super**();

**this**.color = color;

**this**.filled = filled;

**this**.dateCreated = **new** java.util.Date();

}

@Override

**public** String toString() {

**return** "GeometricObject [color=" + color + ", filled=" + filled + ", dateCreated=" + dateCreated + "]";

}

**protected** java.util.Date getDateCreated() {

**return** dateCreated;

}

**protected** String getColor() {

**return** color;

}

**protected** **void** setColor(String color) {

**this**.color = color;

}

**protected** **boolean** isFilled() {

**return** filled;

}

**protected** **void** setFilled(**boolean** filled) {

**this**.filled = filled;

}

/\*\* Abstract method getArea \*/

**public** **abstract** **double** getArea();

/\*\* Abstract method getPerimeter \*/

**public** **abstract** **double** getPerimeter();

}

/\*

\* Triangle.java

\*/

package project6;

public class Triangle extends GeometricObject {

private double side1 = 1.0;

private double side2 = 1.0;

private double side3 = 1.0;

public static void main(String[] args) {

// TODO 自动生成的方法存根

Triangle t1 = new Triangle();

System.out.println("t1.getArea=" + t1.getArea() + " t1.Perimeter=" + t1.getPerimeter());

System.out.println(t1.toString());

System.out.println("");

System.out.println("t1 = new Triangle(5,12,13):");

t1 = new Triangle(5,12,13);

System.out.println("t1.getArea=" + t1.getArea() + " t1.Perimeter=" + t1.getPerimeter());

System.out.println(t1.toString());

System.out.println("");

System.out.println("t1 = new Triangle(1,2,3):");

t1 = new Triangle(1, 2, 3);

System.out.println("t1.getArea=" + t1.getArea() + " t1.Perimeter=" + t1.getPerimeter());

System.out.println(t1.toString());

}

protected Triangle() {

super();

}

protected Triangle(double side1, double side2, double side3) {

super();

if (side1 + side2 > side3 && side1 + side3 > side2 && side2 + side3 > side1) {

this.side1 = side1;

this.side2 = side2;

this.side3 = side3;

} else {

System.out.println("输入数不符合三角形三边关系，生成失败，使用默认值");

}

}

protected double getSide1() {

return side1;

}

protected void setSide1(double side1) {

this.side1 = side1;

}

protected double getSide2() {

return side2;

}

protected void setSide2(double side2) {

this.side2 = side2;

}

protected double getSide3() {

return side3;

}

protected void setSide3(double side3) {

this.side3 = side3;

}

@Override

public double getArea() {

double p = (side1 + side2 + side3) / 2;

double area = Math.sqrt(p \* (p - side1) \* (p - side2) \* (p - side3));

return area;

}

@Override

public double getPerimeter() {

// TODO 自动生成的方法存根

return side1 + side2 + side3;

}

@Override

public String toString() {

return "Triangle [toString():side1=" + side1 + " side2=" + side2 + " side3=" + side3 + "]";

}

}

# Project #2分析设计

## 2.1问题分析

|  |  |  |  |
| --- | --- | --- | --- |
| Analysis | | | |
| 接口 | Colorable | 方法 | HowtoColor |
| 基类 | GeometricObject |  |  |
| 继承子类 | Square |  |  |
| 主函数 | 通过基类类型的引用访问派生类对象  Java数组里没有元素  为每个数组元素赋值 存派生类 考察多态 | | |

### 2.1.1 目标

使用一个名为howToColor()的void方法设计一个名为Colorable的接口。可着色对象的每个类都必须实现可着色接口。设计一个名为Square的类，它extends了geometricObject并实现了Colorable。调用 如何显示如何给正方形着色的String消息 的重载函数。

### 2.1.2功能

#### 2.1.2.1功能一：GeometricObject基类

生成基类

#### 2.1.2.2功能二：

生成Square类，能调用方法howToColor()

### 2.1.3性能

#### 2.1.3.1知道信息的性能

生成的类信息的结果显示到屏幕上

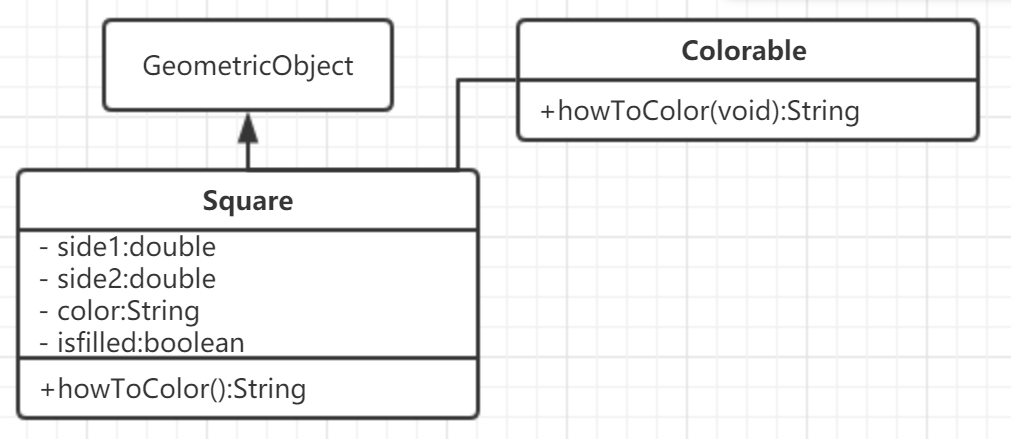
#### 2.1.3.2健壮性

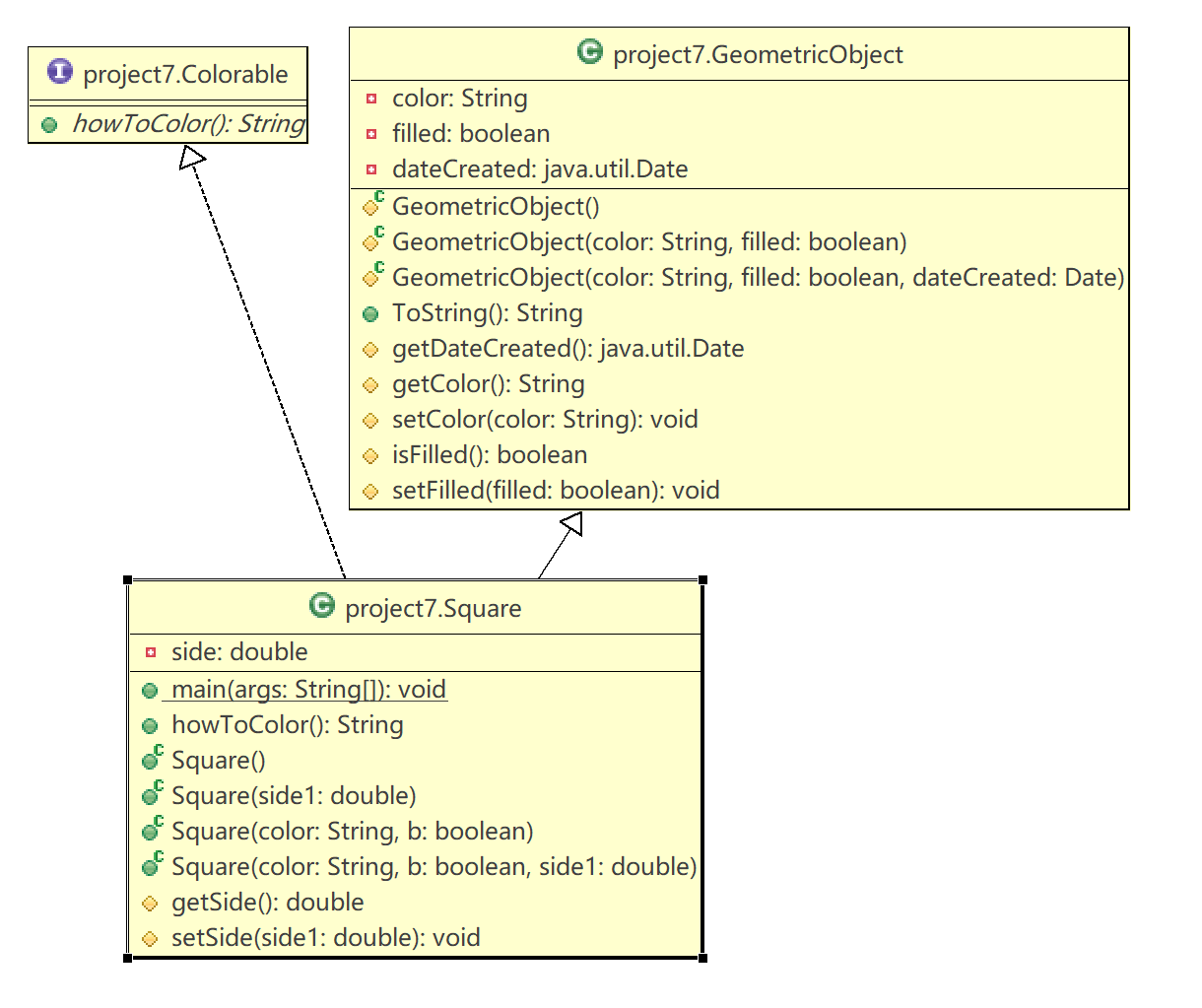
本程序不接受输入

## 2.2设计方案

### 2.2.1模块划分

### 2.2.2类图

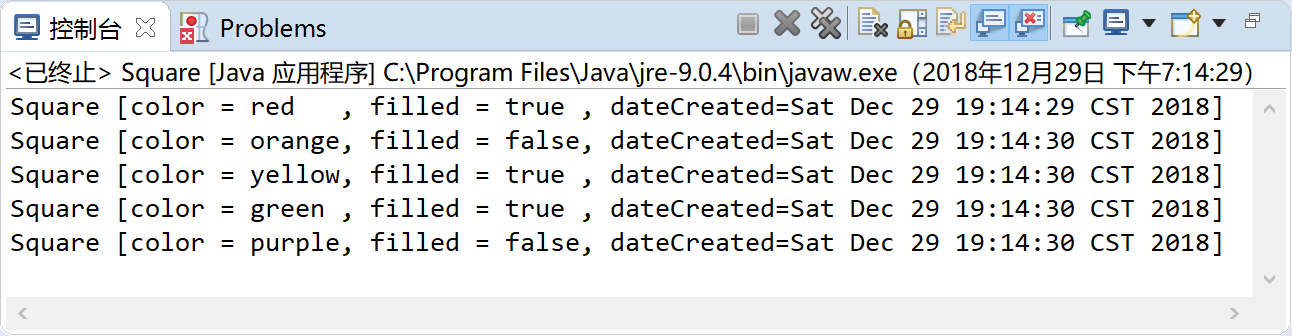




### 2.2.3算法

没有算法

## 2.3实验结果



## 2.4 调试心得

1.String.format(format,string)可以使输出内容整齐美观 2.JVM会基于调用方法时所用的确切对象来动态地决定调用哪个howToEat方法。构造方法可以调用重载的构造方法或者他的父类的构造方法，这种调用必须是构造方法的第一条语句。如果没有显式地调用他们中的任何一个，编译器就会把super()作为构造方法的第一条语句，它调用的式父类的无参构造函数。重写一个方法必须使用与他的父类中的方法相同的签名来定义子类中的方法。实例方法只有在可访问时才能重写，私有方法不能重写。静态方法和实例方法一样可以继承，但是静态方法不能重写！！如果一个方法的参数类型时父类，可以向该方法的参数传递任何子类，这称为多态。因为子类的实例总是它的父类的实例，所以总是可以将一个子类的实例转换成一个父类的变量。当把父类实例转换成他的子类变量时，必须使用转换记号(子类名)进行显式转换。可以用表达式obj instance of Aclass 对象名 instanceof类名测试一个对象是否是一个类的实例。多态意味着父亲的变量可指向子类对象。

## 2.5源程序

/\*

\* Colorable.java

\*/

**package** project7;

**public** **interface** Colorable {

**public** **abstract** String howToColor();

}

/\*

\* GeometricObject.java

\*/

**package** project7;

**import** java.util.Date;

**public** **class** GeometricObject {

**private** String color = "white";

**private** **boolean** filled;

**private** java.util.Date dateCreated;

/\*\* Construct a default geometric object \*/

**protected** GeometricObject() {

dateCreated = **new** java.util.Date();

}

**protected** GeometricObject(String color, **boolean** filled) {

**super**();

**this**.color = color;

**this**.filled = filled;

}

**protected** GeometricObject(String color, **boolean** filled, Date dateCreated) {

**super**();

**this**.color = color;

**this**.filled = filled;

**this**.dateCreated = **new** java.util.Date();

}

**public** String ToString() {

**return** "GeometricObject [color=" + color + ", filled=" + filled + ", dateCreated=" + dateCreated + "]";

}

**protected** java.util.Date getDateCreated() {

**return** dateCreated;

}

**protected** String getColor() {

**return** String.*format*("%-6s",color);

}

**protected** **void** setColor(String color) {

**this**.color = color;

}

**protected** **boolean** isFilled() {

**return** filled;

}

**protected** **void** setFilled(**boolean** filled) {

**this**.filled = filled;

}

}

/\*

\* Square.java

\*/

**package** project7;

**public** **class** Square **extends** GeometricObject **implements** Colorable {

**public** **static** **void** main(String[] args) {

//Create an array of five GeometricObject

GeometricObject[] object = **new** GeometricObject[5];

String []string = {"red","orange","yellow","green","purple"};

**boolean** []isfill= {**true**,**false**,**true**,**true**,**false**};

**for**(**int** i = 0; i<5;i++) {

object[i]=**new** GeometricObject();

object[i]= **new** Square(string[i],isfill[i]);

**if** (object[i] **instanceof** Square)

System.***out***.println(((Square) object[i]).howToColor());

}

}

**private** **double** side;

@Override

**public** String howToColor() {

// **TODO** 自动生成的方法存根

**return** "Square [color = " + getColor()

+", filled = " + String.*format*("%-5s",isFilled())

+", dateCreated=" + getDateCreated() +"]";

}

**public** Square() {

**super**();

**this**.side = 1;

}

**public** Square(**double** side1) {

**super**();

**this**.side = side1;

}

**public** Square(String color, **boolean** b) {

**super**();

setColor(color);

setFilled(b);

**this**.side = 1;

}

**public** Square(String color, **boolean** b,**double** side1) {

**super**();

setColor(color);

setFilled(b);

**this**.side = side1;

}

**protected** **double** getSide() {

**return** side;

}

**protected** **void** setSide(**double** side1) {

**this**.side = side1;

}

}

# Project #3分析设计

## 3.1问题分析

假设有一个Account账户里，里面有balance，线程A随机生成一个字母，相当于存储了这个字母对应的ASCII码值进入Account，在A存入的时候，Account被上锁，其他人不可干预操作以放冲突。反之，B也是随机，但是相当于取钱。ThreadC是在AB操作完以后，根据±balance的余额对AB进行分数的加减。对应UI在控制台和GUI输出。

在线程池里，有三个线程：ABC。A和B 不可以同时对Account操作。C一定要等AB都执行完了，才能进行比较。C在比较的时候。A和B都不能开始下一轮的生成，否则会干扰到C的分数判断。所以如果步骤不是A 那么A wait 如果A做完了，步骤轮到B ,叫醒B.如果步骤不是B ，那么Bwait.如果B做完了，步骤轮到C,叫醒C。如果步骤不是C，那么C wait.如果C做完了，那么步骤轮到第二轮开始的A，因为第一轮的A并不需要等其他线程。所以当开始了第二轮，那么C叫醒A。所以步骤的初始值就是A .

|  |  |  |  |
| --- | --- | --- | --- |
| 操作 线程 | ThreadA | ThreadB | ThreadC |
| Account | Balance++ | Balance-- | Balance(> < ==)0? |
| Points | 若>0 ->A.points+2;<0 ->B.points+2; ==0 -> A.points++=B.points++ | | |
| 逻辑顺序loop | 1->2 | 2->3 | 3>1 |

### 3.1.1 目标

1个线程池-三个线程-控制台输出表-GUI作为UI-线程A线程B生成字符且随机睡眠，线程C比较AB生成AB得分，最后统计最后结果。

### 3.1.2功能

#### 3.1.2.1功能一：生成字母

线程A、B生成随机字母。

#### 3.1.2.2功能二：比较大小

线程C比较两者大小，得出最后赢家

### 3.1.3性能

#### 3.1.3.1比较大小的性能

在线程AB生成2个字母的之后，C比较大小，根据规则给出得分，UI界面显示得分情况

#### 3.1.3.2健壮性

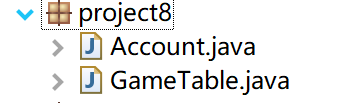
没有用户输入，其他都用try catch块处理了

#### 3.1.3.3异常

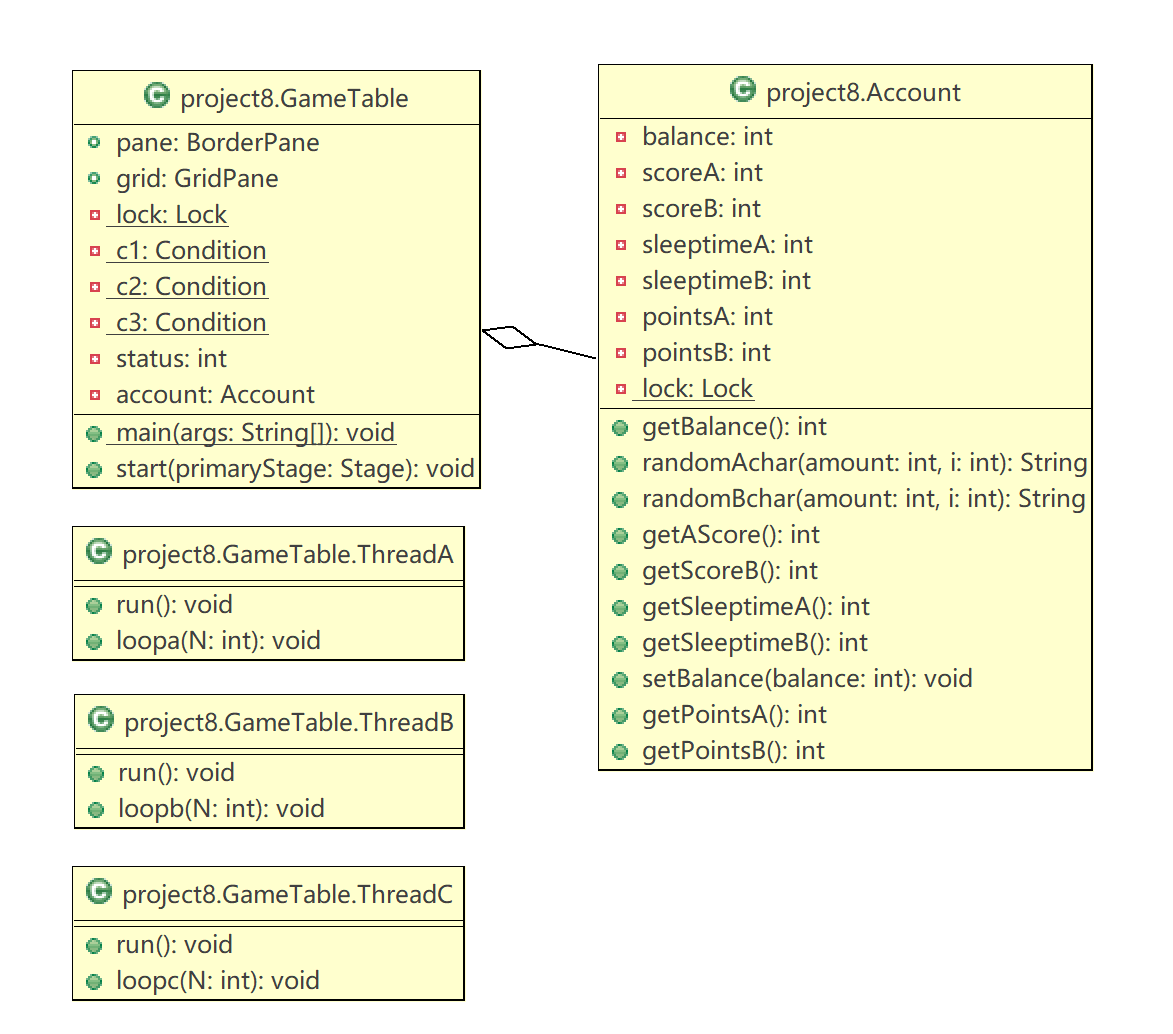
Try catch住了

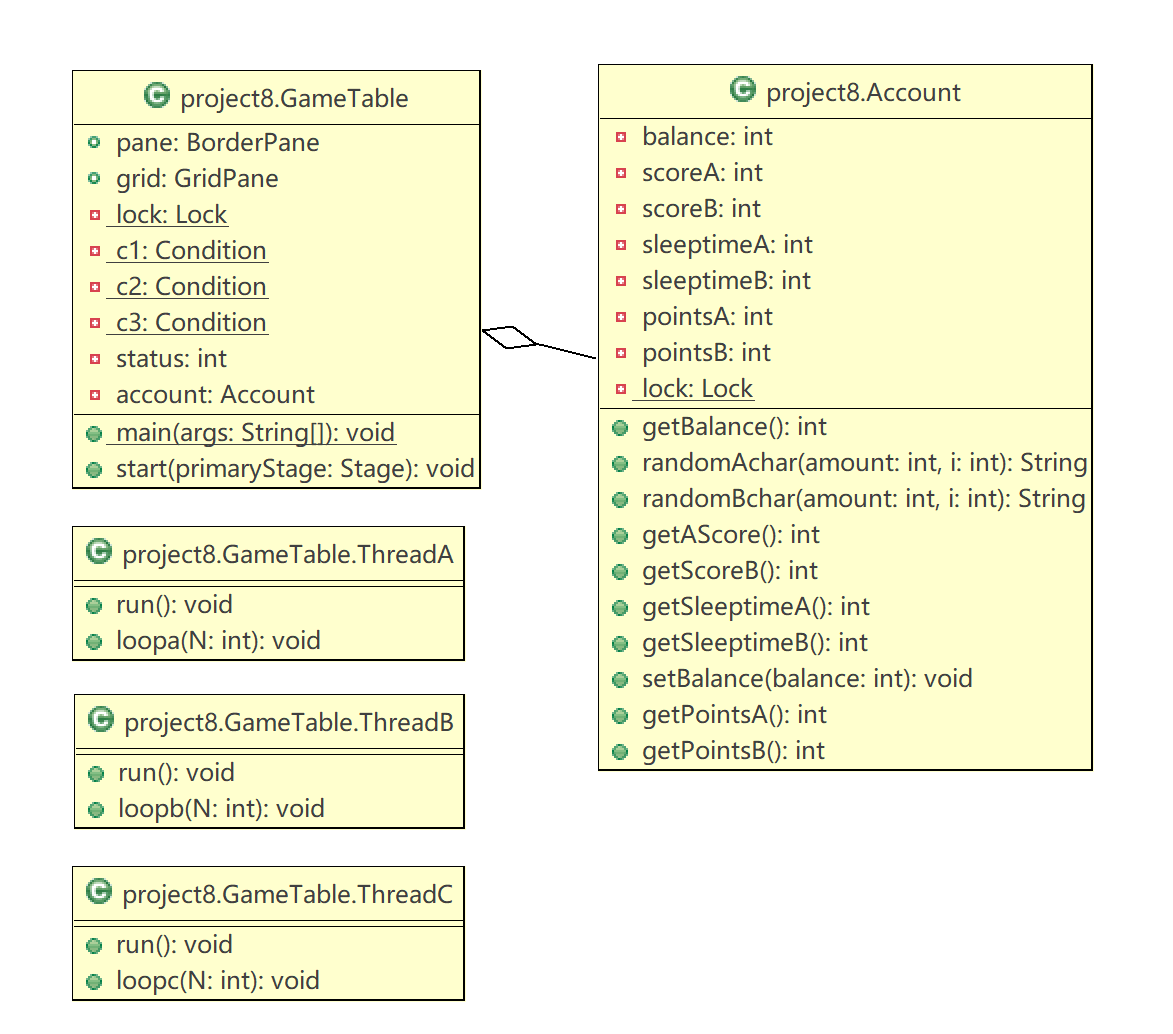
## 3.2设计方案

### 3.2.1模块划分

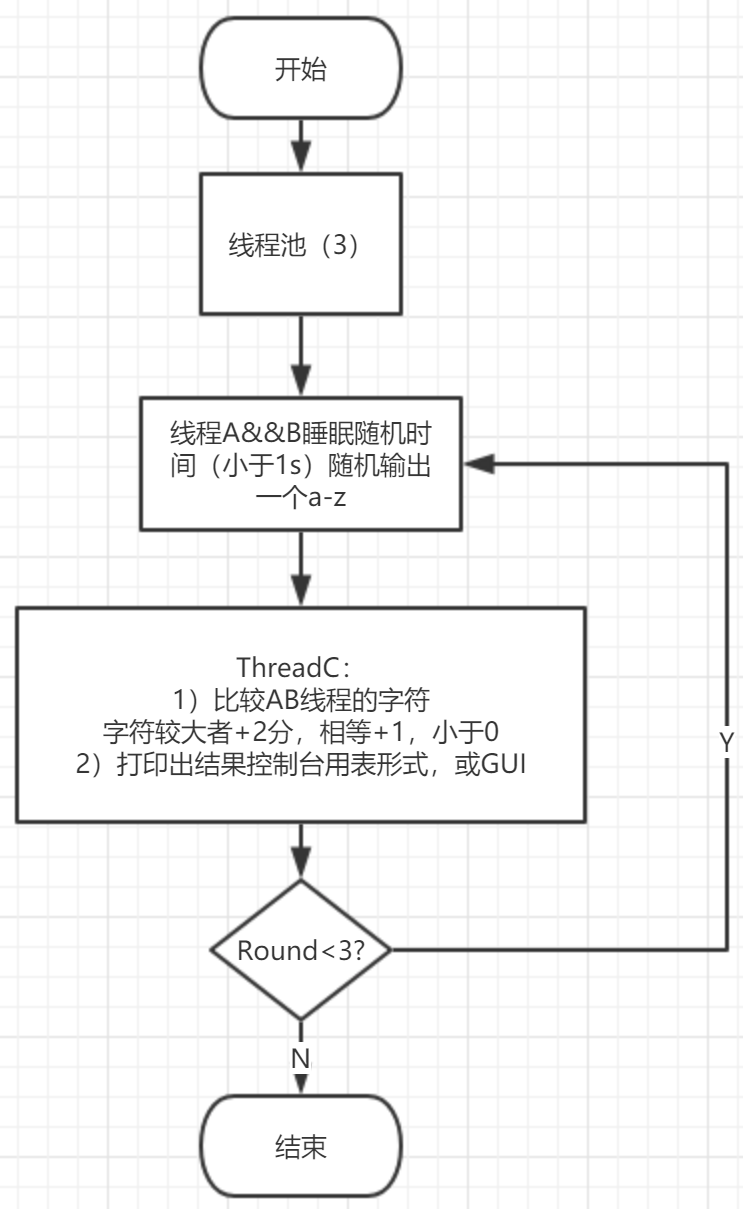


### 3.2.2类图





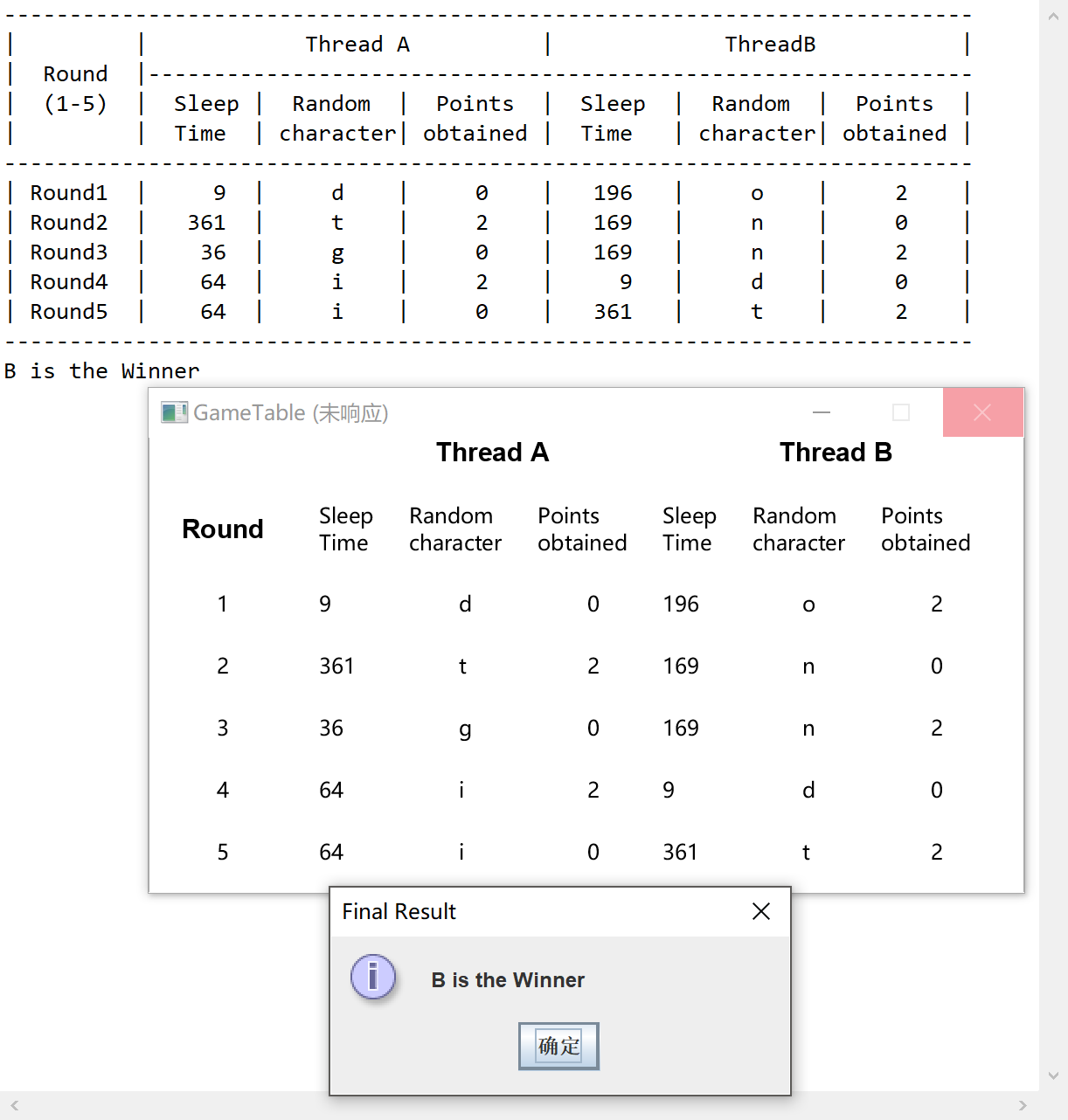
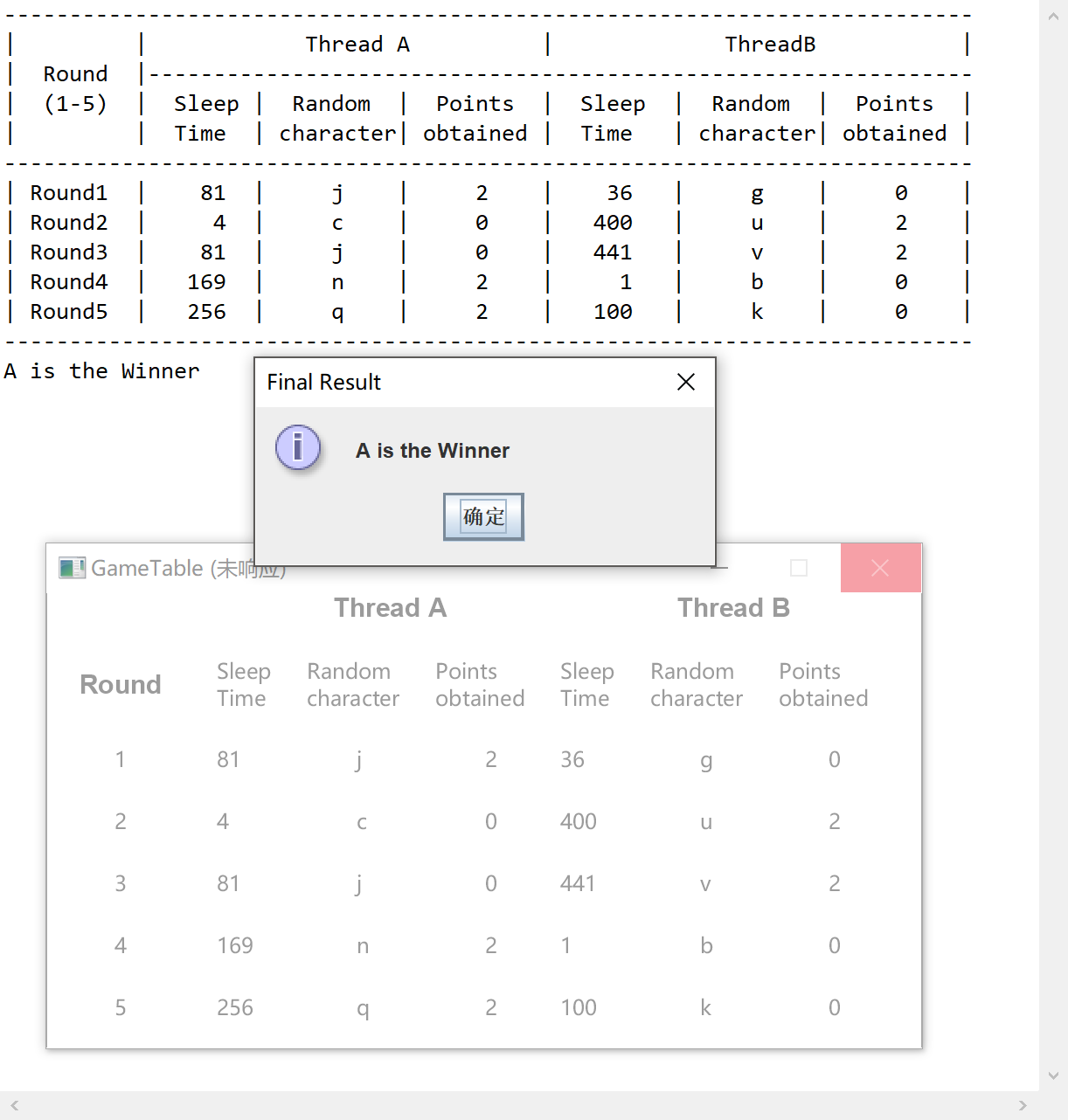
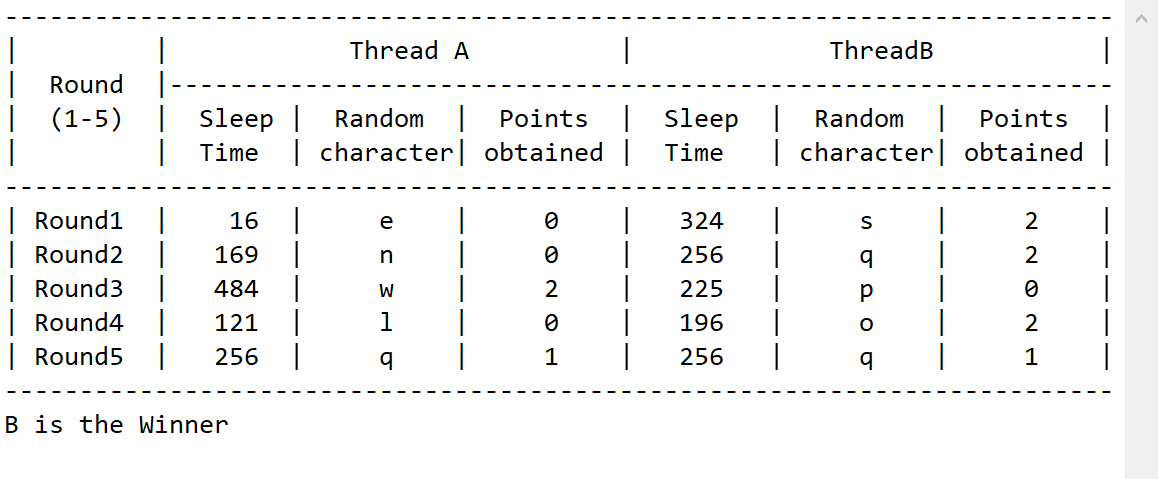
### 3.2.3算法



#### 3.2.1.1算法一：

设同一个锁，如果状态不是1，那么线程Ⅰ等待直到被叫醒，当线程一完事儿了，状态到2，叫醒线程Ⅱ，如果状态不是2，那么线程Ⅱ等待直到被叫醒，当完事儿了，状态到3，叫醒线程三，如果状态不是三，那么线程三等待直到被叫醒，当线程三完事了，状态又调至1，叫醒1。这样可以让三个线程轮流做事情。

## 3.3实验结果

## 3.4 调试心得

设同一个锁，如果状态不是1，那么线程Ⅰ等待直到被叫醒，当线程一完事儿了，状态到2，叫醒线程Ⅱ，如果状态不是2，那么线程Ⅱ等待直到被叫醒，当完事儿了，状态到3，叫醒线程三，如果状态不是三，那么线程三等待直到被叫醒。这样可以让三个线程轮流做事情。逻辑顺序很重要。每个任务都是Runnable接口的实例，或者继承Thread类，线程就是一个便于任务执行的对象。可以通过实现Runnable接口来定义人物类，通过使用Thread构造方法包住一个任务来创建线程。一个线程对象被创建之后，可以使用start()方法启动线程，可以使用Sleep()方法将线程转入休眠状态，一边其他线程获得运行的机会。线程对象从来不会直接调用run方法。到了执行某个线程的时候，Java虚拟机调用run方法。类必须覆盖run方法，告诉系统线程运行时将会做什么。为了避免线程破坏共享资源，可以使用同步的方法或块，同步方法在执行前需要获得一个锁，当同步方法是实例方法时，锁是在调用方法的对象上，当同步方法是静态类方法时锁是在方法所在的类上。在执行方法中某个代码块时，可以使用同语句获得任何对象上的锁，这个代码块称为同步块。可以使用显式锁和条件，以及对象的内置监视器来便于进程之间的通信。也可以使用信号量来限制访问共享资源的并行任务数量。如果两个或多个三个线程获取多个对象上的锁时，每个线程都有一个对象上的锁并等待另一个对象上的锁，这个时候可能发生死锁现象。使用资源排序可以避免死锁。

## 3.5源程序

/\*

\* GameTable.java:toShowGUI

\*/

package project8;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.concurrent.locks.Condition;

import java.util.concurrent.locks.Lock;

import java.util.concurrent.locks.ReentrantLock;

import javafx.application.Application;

import javafx.application.Platform;

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.GridPane;

import javafx.scene.text.Font;

import javafx.scene.text.FontWeight;

import javafx.scene.text.\*;

import javafx.stage.Stage;

public class GameTable extends Application {

public BorderPane pane = new BorderPane();

public GridPane grid = new GridPane();

private static Lock lock = new ReentrantLock();

private static Condition c1 = lock.newCondition();

private static Condition c2 = lock.newCondition();

private static Condition c3 = lock.newCondition();

private int status = 1;

public static void main(String[] args) {

launch(args);

}

@Override

public void start(Stage primaryStage) throws Exception {

// top center bottom

grid.setHgap(20);

grid.setVgap(20);

grid.setPadding(new Insets(0, 10, 0, 10));

// Round in column 1-2, row 1

Text round = new Text(" Round\t");

round.setFont(Font.font("Arial", FontWeight.BLACK, 15));

grid.add(round, 0, 1);

// ThreadA in column 2-4, row 1

Text ta = new Text("\t\tThread A");

ta.setFont(Font.font("Arial", FontWeight.BLACK, 15));

grid.add(ta, 1, 0, 3, 1);

// ThreadB in column 5-7, row 1

Text tb = new Text("\t\tThread B");

tb.setFont(Font.font("Arial", FontWeight.BLACK, 15));

grid.add(tb, 4, 0, 3, 1);

// item in column 2, row 2

Text st = new Text("Sleep\nTime");

grid.add(st, 1, 1);

Text rc = new Text("Random\ncharacter");

grid.add(rc, 2, 1);

Text po = new Text("Points\nobtained");

grid.add(po, 3, 1);

grid.add(new Text("Sleep\nTime"), 4, 1);

grid.add(new Text("Random\ncharacter"), 5, 1);

grid.add(new Text("Points\nobtained"), 6, 1);

for (int i = 1; i < 6; i++) {

grid.add(new Text("\t" + String.valueOf(i) + "\t"), 0, i + 1);

}

// thread threads concurrent

ExecutorService executor = Executors.newFixedThreadPool(3);

executor.execute(new ThreadA());

executor.execute(new ThreadB());

executor.execute(new ThreadC());

executor.shutdown();

// Console UI

System.out.println("--------------------------------------------------------------------------");

System.out.println("| | Thread A | ThreadB |");

System.out.println("| Round |---------------------------------------------------------------");

System.out.println("| (1-5) | Sleep | Random | Points | Sleep | Random | Points |");

System.out.println("| | Time | character| obtained | Time | character| obtained |");

System.out.println("--------------------------------------------------------------------------");

// Create a scene and place the pane

pane.setCenter(grid);

Scene scene = new Scene(pane, 500, 260);

primaryStage.setTitle("GameTable");

primaryStage.setScene(scene);

primaryStage.show();

}

private Account account = new Account();

public class ThreadA implements Runnable {

@Override

public synchronized void run() {

for (int i = 0; i < 5; i++) {

lock.lock();

try {

// 等待thread c

if (status != 1) {

c1.await();

}

// 执行线程a

loopa(i);

// invoke 2

status = 2;

c2.signal();

} catch (InterruptedException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

} finally {

lock.unlock();

}

}

}

public void loopa(int N) {

Platform.runLater(new Runnable() {

@Override

public void run() {

for (int i = 1; i <= 1; i++) {

Platform.runLater(() -> {

// random char column 3

grid.add(new Text("\t" + account.randomAchar((int) (Math.random() \* 25) + 1, N)), 2, 2 + N);

// sleep column 2 row 3-4-5-..

grid.add(new Text(String.valueOf(account.getSleeptimeA())), 1, 2 + N);

});

}

}

});

}

}

public class ThreadB implements Runnable {

@Override

public synchronized void run() {

for (int i = 0; i < 5; i++) {

lock.lock();

try {

// 等待 A

if (status != 2)

c2.await();

// 执行b

loopb(i);

// invoke c

status = 3;

c3.signal();

} catch (InterruptedException e) {

e.printStackTrace();

} finally {

lock.unlock();

}

}

}

public void loopb(int N) {

Platform.runLater(new Runnable() {

@Override

public void run() {

for (int i = 1; i <= 1; i++) {

// test

Platform.runLater(() -> {

grid.add(new Text("\t" + account.randomBchar((int) (Math.random() \* 25) + 1, N)), 5, 2 + N);

// sleep column 5 row 3-4-5-..

grid.add(new Text(String.valueOf(account.getSleeptimeB())), 4, 2 + N);

});

}

}

});

}

}

public class ThreadC extends Thread {

@Override

public synchronized void run() {

for (int i = 0; i < 5; i++) {

lock.lock();

try {

// 等待 B

if (status != 3)

c3.await();

// 执行C

loopc(i);

// invoke c

status = 1;

c1.signal();

} catch (InterruptedException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

} finally {

lock.unlock();

}

}

}

public void loopc(int N) {

Platform.runLater(new Runnable() {

@Override

public void run() {

for (int i = 1; i <= 1; i++) {

Platform.runLater(() -> {

// score a 4 row 3-4-5-..

grid.add(new Text("\t" + String.valueOf(account.getPointsA())), 3, 2 + N);

// score b column 7

grid.add(new Text("\t" + String.valueOf(account.getPointsB())), 6, 2 + N);

});

}

}

});

}

}

}

//Account.java

package project8;

import java.util.concurrent.locks.Lock;

import java.util.concurrent.locks.ReentrantLock;

import javax.swing.JOptionPane;

import javafx.application.Platform;

public class Account {

private int balance =0;

private int scoreA = 0;

private int scoreB = 0;

private int sleeptimeA =1;

private int sleeptimeB =1;

private int pointsA = 0;

private int pointsB = 0;

private static Lock lock = new ReentrantLock();

public int getBalance() {

return balance;

}

public String randomAchar(int amount,int i) {

lock.lock();

setBalance(0);

balance = amount;

try {

sleeptimeA =amount\*amount%1000;

Thread.sleep(sleeptimeA);

System.out.print("| Round"+(i+1)

+ " | "

+ String.format("%3d", sleeptimeA));

} catch (InterruptedException ex) {

ex.printStackTrace();

}

//amount是随机生成的char，getBalance是两者之差

//invoke randomBchar

lock.unlock();

System.out.print(" | "

+ (char) ('a' + amount)

+ " | ");

return String.valueOf((char) ('a' + amount));

}

public String randomBchar(int amount,int i) {

lock.lock();

try {

sleeptimeB = amount\*amount%1000;

Thread.sleep(sleeptimeB);

balance = balance - amount;

//System.out.println((char) ('a' + amount)+"\t");

pointsA =scoreA;

pointsB =scoreB;

if(balance>0) {

scoreA += 2;

}

else if(balance==0) {

scoreA += 1;

scoreB += 1;

}

else {

scoreB += 2;

}

getAScore();

getScoreB();

pointsA =scoreA-pointsA;

pointsB =scoreB-pointsB;

System.out.print(

String.format("%2d", pointsA)

+ " | "

+ String.format("%3d", sleeptimeB));

System.out.print(" | "

+ (char) ('a' + amount)

+ " | ");

System.out.print(

String.format("%2d", pointsB)

+ " |");

System.out.println();

//cond.await();

} catch (InterruptedException ex) {

ex.printStackTrace();

}

lock.unlock();

if(i==4) {

System.out.println("--------------------------------------------------------------------------");

if(getAScore()>getScoreB()) {

Platform.runLater(() -> {

JOptionPane.showMessageDialog(null, "A is the Winner", "Final Result", JOptionPane.INFORMATION\_MESSAGE);

});

System.out.println("A is the Winner");}

else if(getAScore()<getScoreB()) {

Platform.runLater(() -> {

JOptionPane.showMessageDialog(null, "B is the Winner", "Final Result", JOptionPane.INFORMATION\_MESSAGE);

});

System.out.println("B is the Winner");}

else {

Platform.runLater(() -> {

JOptionPane.showMessageDialog(null, "A is equal to B", "Final Result", JOptionPane.INFORMATION\_MESSAGE);

});

System.out.println("A is equal to B");

}

}

return String.valueOf((char) ('a' + amount));

}

public int getAScore() {

return scoreA;

}

public int getScoreB() {

return scoreB;

}

public int getSleeptimeA() {

return sleeptimeA;

}

public int getSleeptimeB() {

return sleeptimeB;

}

public void setBalance(int balance) {

this.balance = balance;

}

public int getPointsA() {

return pointsA;

}

public int getPointsB() {

return pointsB;

}

}

# Project #4分析设计

## 4.1问题分析

TCP让两台主机建立连接并交换数据流。UDP允许一台计算机上的应用程序向另一台计算机上的应用程序发送数据报。服务器是两个线程，接收TCP和UDP发来的数据。客户端是两个线程，一个TCP发送数据，一个UDP发送数据报。所以是基于多线程的互联网编程。

|  |  |  |  |
| --- | --- | --- | --- |
| 操作 线程 | Server.ThreadTCP | Server.ThreadUDP |  |
| MultClient.TCPClient | 发送data 接收 | 被唤醒 |  |
| MultClient.UDPClient | 发送data 接收 |  | 比较TCP UDP数据 |
| 判分 |  | 每次UDP执行时判分 | GUI&UI显示分数 |

### 4.1.1 目标

界面要美观。两个线程要跑的流畅。

### 4.1.2功能

#### 4.1.2.1功能一：有图形界面

实时展现情况

#### 4.1.2.2功能二：两个线程从客户端发送信息

两个客户端线程给服务器发送信息实现石头剪刀布游戏并且由服务器充当裁判，计算积分和赢家。

### 4.1.3性能

#### 4.1.3.1可视化的性能

可视化能玩石头剪刀布

#### 4.1.3.2健壮性

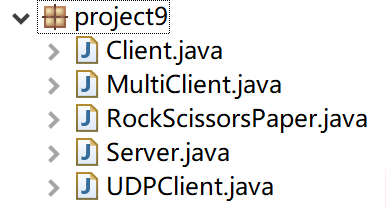
不接受输入

#### 4.1.3.3异常输入处理

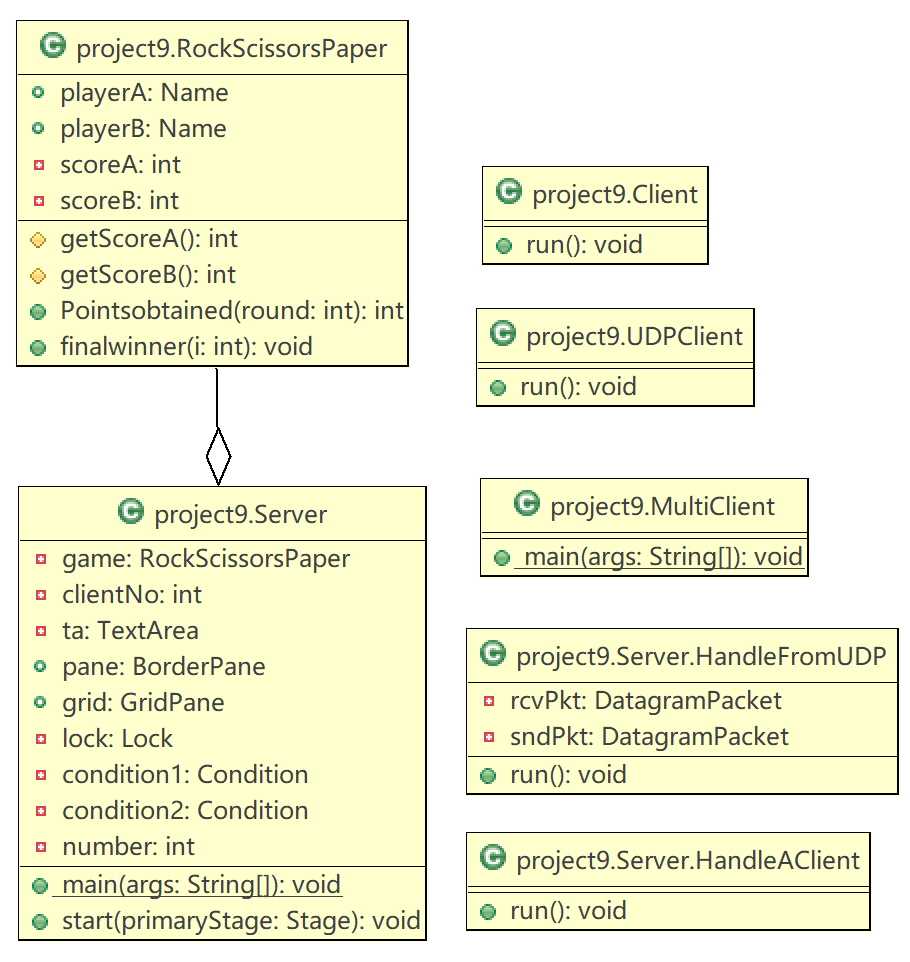
Try-catch

## 4.2设计方案

### 4.2.1模块划分



### 4.2.2类图

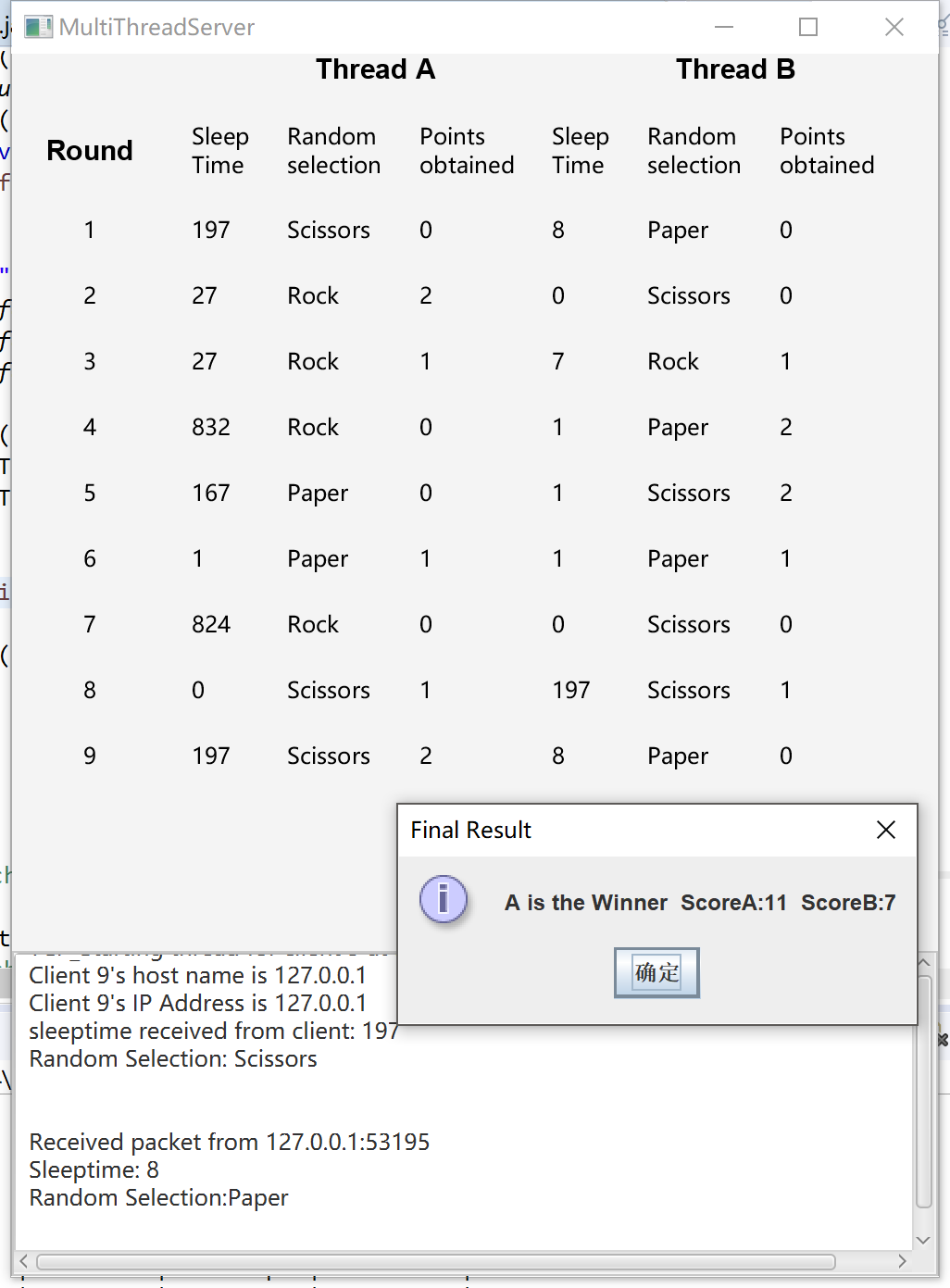


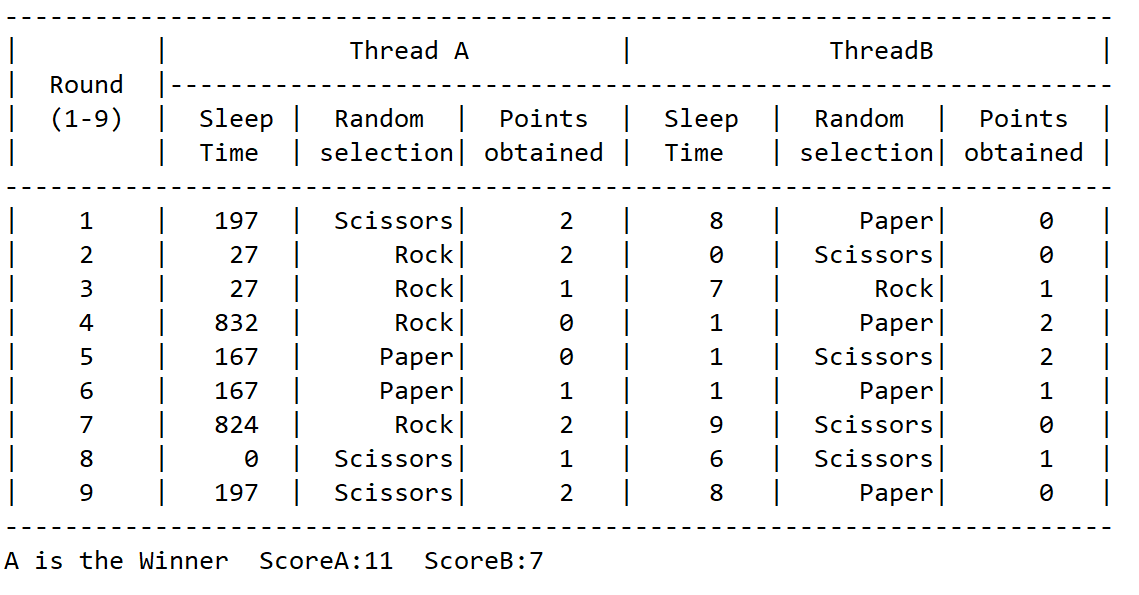
### 4.2.3算法

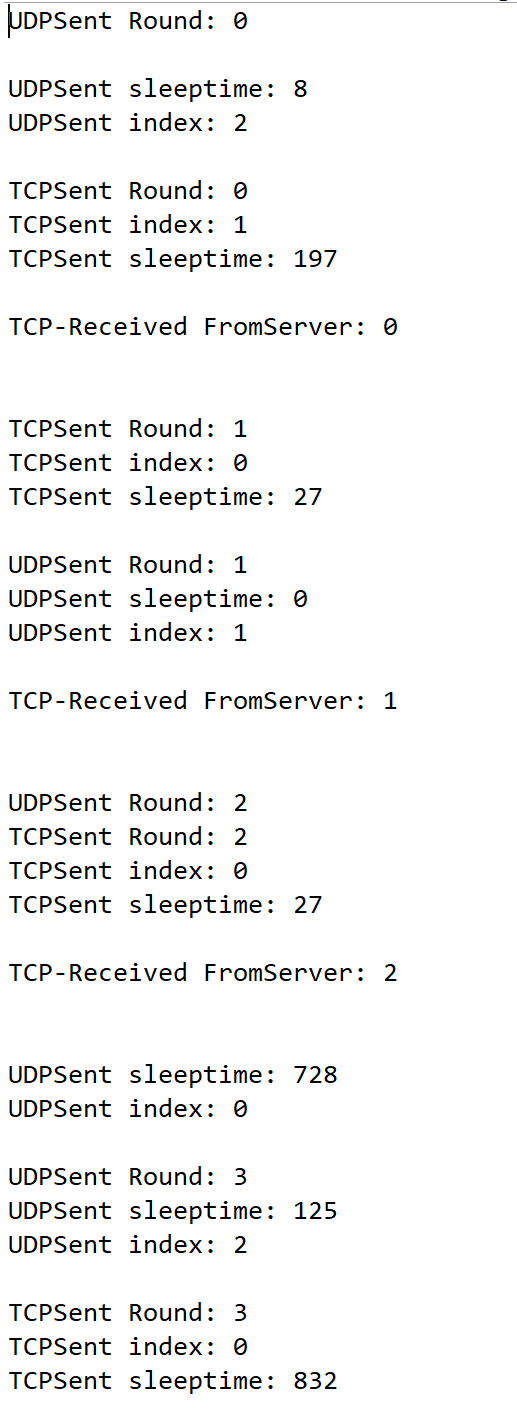
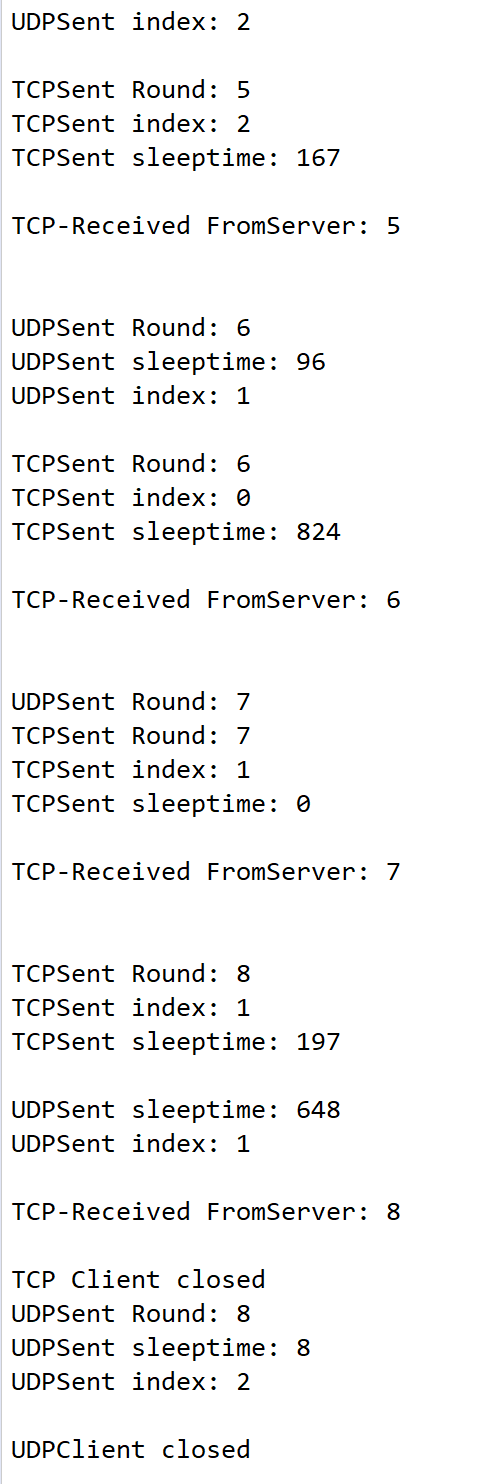
#### 4.2.1.1算法一：互相唤醒

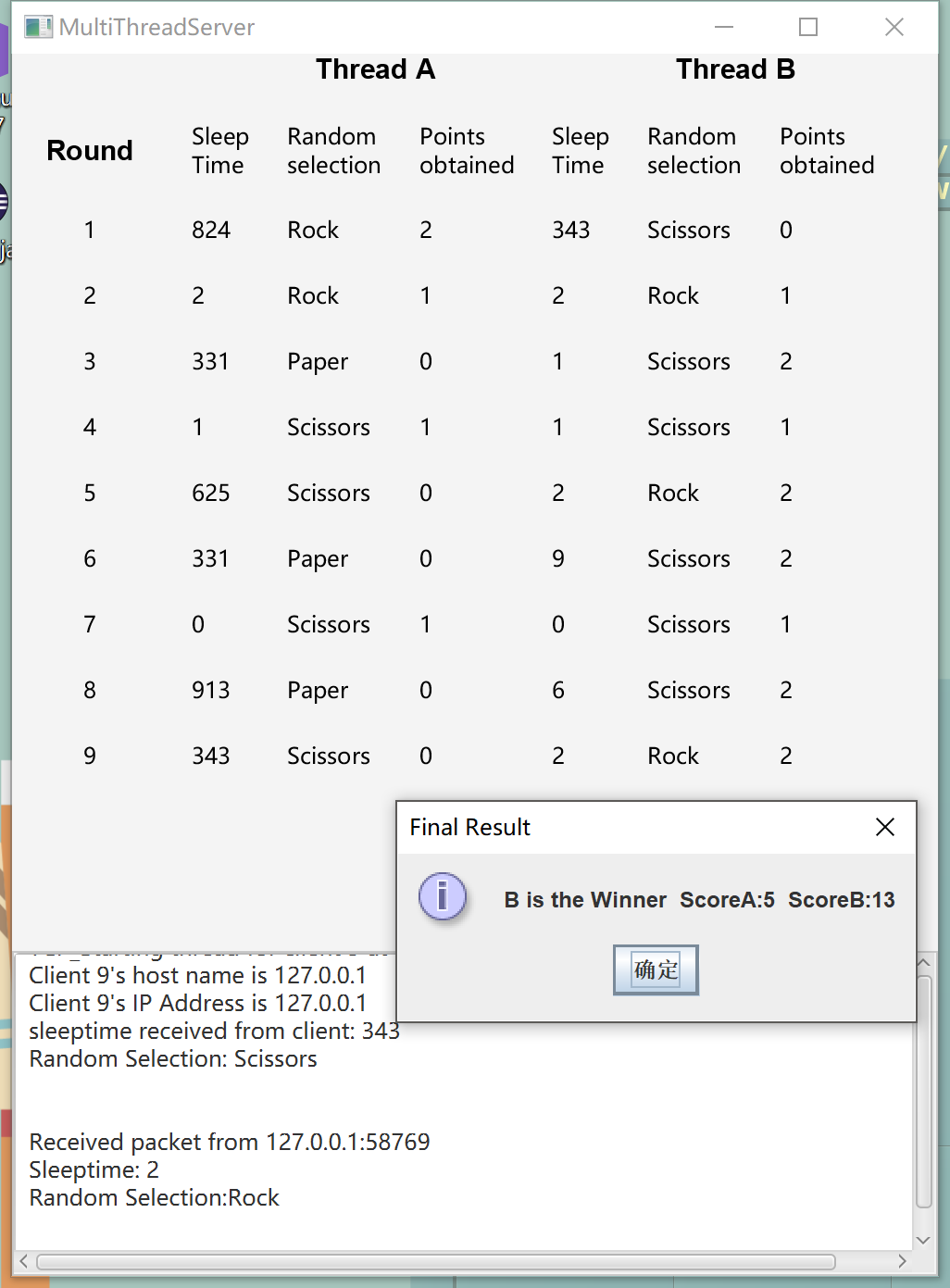
设同一个锁，如果状态不是1，那么线程Ⅰ等待直到被叫醒，当线程一完事儿了，状态到2，叫醒线程Ⅱ，如果状态不是2，那么线程Ⅱ等待直到被叫醒，当完事儿了，状态到1，叫醒线程1.

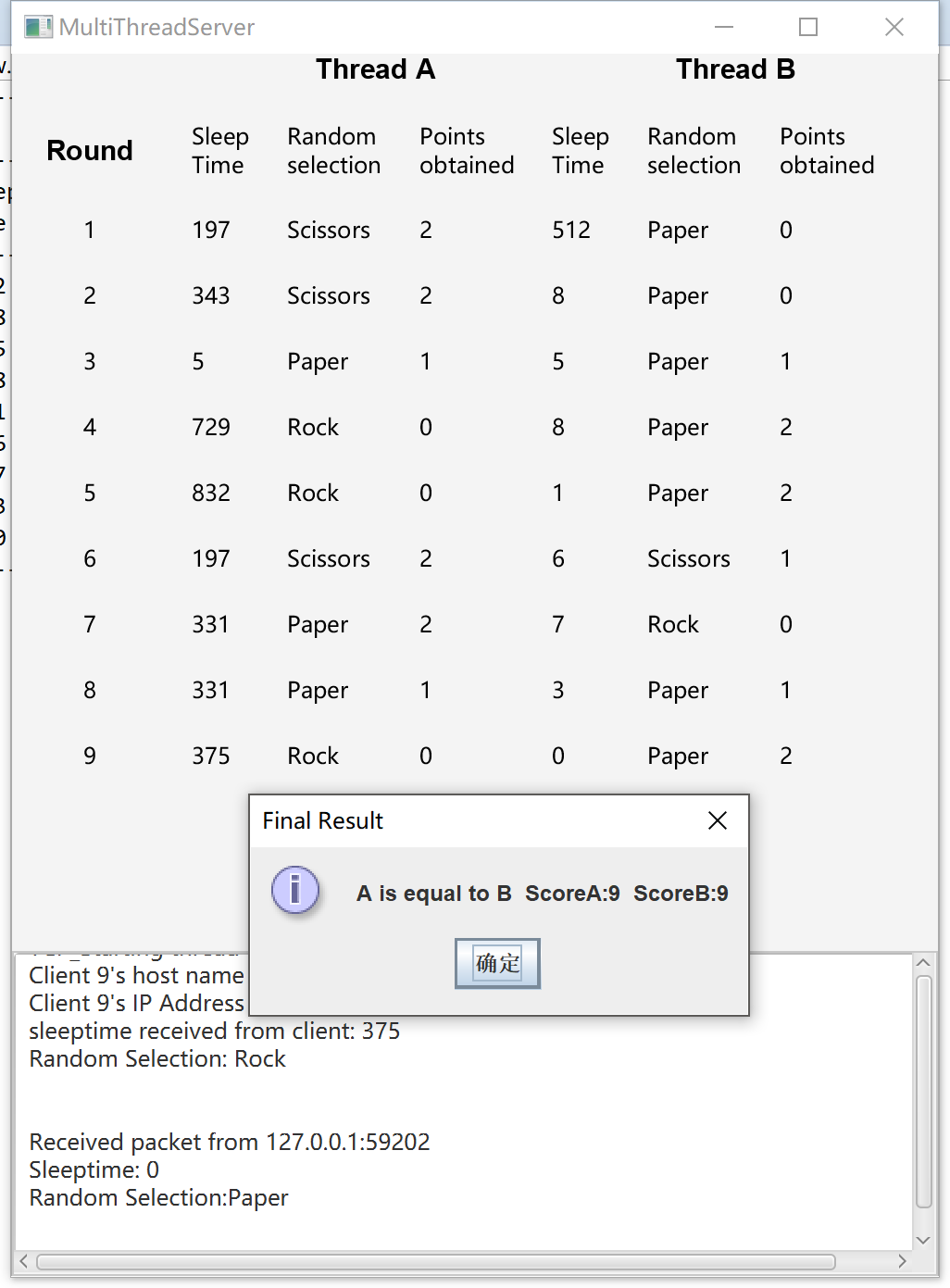
## 4.3实验结果



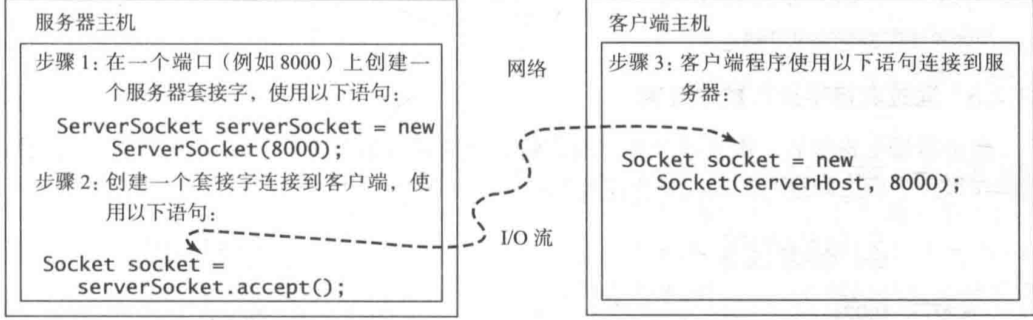






## 4.4 调试心得



客户端连接到服务器套接字之后，会请求与服务器进行连接

如果要发送和接受对象的话要Implements java.io.Serializable 不过我发的数据所以我没用到，但是一个程序是可以向另一个程序发送和接收对象的。服务器用In.readObject()

流套接字用TCP传输控制协议来进行数据传输，无损可靠。UDP容易丢包。要创建一个服务器首先使用语句new ServerSocket(port)获取一个服务器套接字。在创建服务器套接字之后，可以启动服务器，使用服务器的accept()方法监听连接请求。客户端通过new socket(serverName,port)来创建一个客户端套接字，用于向服务器发送连接请求。可以利用线程同时处理服务器的多个客户端。

UDP的时候，用new DatagramSocket(port) 每一次都要将byte[] buf 清空用零填充来接收新的数据，每次接收都需要new DatagramPacket(buf,buf.length); DatagramSocket的receive函数

枚举类.values()[index]可以到枚举类的一个成员

Udp客户端发送需要setPort.把数据转换成Bytes.放在buf里 把buf作为DatagramPacket的构造参数。sndPkt = **new** DatagramPacket(buf, buf.length); sndPkt.setPort(8000);

buf = Double.*valueOf*(round).toString().getBytes();sndPkt.setData(buf);!

用setData函数然后socket.send(sndPkt);

2）之前总是会分数判断不对，我纳闷了很久，设置断点发现，

game.playerA = Name.*values*()[index];

这句话执行后，我的分数就会错乱，B累积的分数总是会覆盖在A 上，导致分数累积错误。后来发现这是相当于一个赋值语句，我将一个新的枚举类型赋给了playerA，但是我的枚举类型是这样定义的

**public** **enum** Name {

***Rock***, ***Scissors***, ***Paper***;

// 成员变量

**private** **int** sleeptime;

**private** **int** score = 0;

**private** **int** point = 0;

…… setter&&getter……

}

所以我应该把分数放在外面orz,然后终于对啦。

## 4.5源程序

/\*\*

\* RockScissorsPaper.java

\*/

package project9;

import javax.swing.JOptionPane;

public class RockScissorsPaper {

public enum Name {

Rock, Scissors, Paper;

// 成员变量

private int sleeptime;

private int point = 0;

protected int getSleeptime() {

return sleeptime;

}

protected void setSleeptime(int sleeptime) {

this.sleeptime = sleeptime;

}

protected int getPoint() {

return point;

}

protected void setPoint(int point) {

this.point = point;

}

public int compareWith(Name rival) {

if (this == rival) {

return 0;

} else if ((this == Rock && rival == Scissors) || (this == Scissors && rival == Paper)

|| (this == Paper && rival == Rock)) {

return 1;

} else {

return -1;

}

}

}

public Name playerA;

public Name playerB;

private int scoreA = 0;

private int scoreB = 0;

protected int getScoreA() {

return scoreA;

}

protected int getScoreB() {

return scoreB;

}

public int Pointsobtained(int round) {

// Initialize

// score keep recording

int result = playerA.compareWith(playerB);

if (result > 0) {

// A get 2 points

playerB.point = 0;

playerA.point = 2;

scoreA += 2;

} else if (result < 0) {

// B get 2 points

// playerB.score = playerB.score + 2;

playerB.point = 2;

playerA.point = 0;

scoreB += 2;

} else {

// A is equals to B

scoreA++;

scoreB++;

playerB.point = 1;

playerA.point = 1;

}

return round;

}

public void finalwinner(int i) {

if (i == 9) {

System.out.println("--------------------------------------------------------------------------");

if (getScoreA() > getScoreB()) {

JOptionPane.showMessageDialog(null, "A is the Winner" + " ScoreA:" + scoreA + " ScoreB:" + scoreB,

"Final Result", JOptionPane.INFORMATION\_MESSAGE);

System.out.println("A is the Winner" + " ScoreA:" + scoreA + " ScoreB:" + scoreB);

} else if (getScoreA() < getScoreB()) {

JOptionPane.showMessageDialog(null, "B is the Winner" + " ScoreA:" + scoreA + " ScoreB:" + scoreB,

"Final Result", JOptionPane.INFORMATION\_MESSAGE);

System.out.println("B is the Winner" + " ScoreA:" + scoreA + " ScoreB:" + scoreB);

} else {

JOptionPane.showMessageDialog(null, "A is equal to B" + " ScoreA:" + scoreA + " ScoreB:" + scoreB,

"Final Result", JOptionPane.INFORMATION\_MESSAGE);

System.out.println("A is equal to B" + " ScoreA:" + scoreA + " ScoreB:" + scoreB);

}

}

}

}

//Server.java

package project9;

import java.io.\*;

import java.net.\*;

import java.util.Arrays;

import java.util.Date;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.concurrent.locks.Condition;

import java.util.concurrent.locks.Lock;

import java.util.concurrent.locks.ReentrantLock;

import javafx.application.Application;

import javafx.application.Platform;

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.ScrollPane;

import javafx.scene.control.TextArea;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.GridPane;

import javafx.scene.text.Font;

import javafx.scene.text.FontWeight;

import javafx.scene.text.Text;

import javafx.stage.Stage;

import project9.RockScissorsPaper.Name;

public class Server extends Application {

private RockScissorsPaper game = new RockScissorsPaper();

private int clientNo = 0;

private TextArea ta = new TextArea();

public BorderPane pane = new BorderPane();

public GridPane grid = new GridPane();

private Lock lock = new ReentrantLock();

private Condition condition1 = lock.newCondition();

private Condition condition2 = lock.newCondition();

private int number = 1;

public static void main(String[] args) {

launch(args);

}

@Override

public void start(Stage primaryStage) throws Exception {

// top center bottom

grid.setHgap(20);

grid.setVgap(20);

grid.setPadding(new Insets(0, 10, 0, 10));

// Round in column 1-2, row 1

Text round = new Text(" Round\t");

round.setFont(Font.font("Arial", FontWeight.BLACK, 15));

grid.add(round, 0, 1);

// ThreadA in column 2-4, row 1

Text threada = new Text("\t\tThread A");

threada.setFont(Font.font("Arial", FontWeight.BLACK, 15));

grid.add(threada, 1, 0, 3, 1);

// ThreadB in column 5-7, row 1

Text tb = new Text("\t\tThread B");

tb.setFont(Font.font("Arial", FontWeight.BLACK, 15));

grid.add(tb, 4, 0, 3, 1);

// item in column 2, row 2

Text st = new Text("Sleep\nTime");

grid.add(st, 1, 1);

Text rc = new Text("Random\nselection");

grid.add(rc, 2, 1);

Text po = new Text("Points\nobtained");

grid.add(po, 3, 1);

grid.add(new Text("Sleep\nTime"), 4, 1);

grid.add(new Text("Random\nselection"), 5, 1);

grid.add(new Text("Points\nobtained"), 6, 1);

ExecutorService executor = Executors.newFixedThreadPool(2);

executor.execute(new HandleAClient());

executor.execute(new HandleFromUDP());

executor.shutdown();

// Console UI

System.out.println("--------------------------------------------------------------------------");

System.out.println("| | Thread A | ThreadB |");

System.out.println("| Round |---------------------------------------------------------------");

System.out.println("| (1-9) | Sleep | Random | Points | Sleep | Random | Points |");

System.out.println("| | Time | selection| obtained | Time | selection| obtained |");

System.out.println("--------------------------------------------------------------------------");

// Create a scene and place the pane

pane.setCenter(grid);

pane.setBottom(new ScrollPane(ta));

Scene scene = new Scene(pane, 500, 660);

primaryStage.setTitle("MultiThreadServer");

primaryStage.setScene(scene);

primaryStage.show();

}

class HandleAClient implements Runnable {

@Override

public void run() {

ServerSocket serverSocket;

try {

serverSocket = new ServerSocket(8000);

Socket socket = serverSocket.accept();

for (int round = 0; round < 9; round++) {

lock.lock();

if (number != 1)

condition1.await();

// ta.appendText("MultiThreadServer started at " + new Date() + '\n');

clientNo++;

Platform.runLater(() -> {

// Display the client number

ta.appendText("TCP\_Starting thread for client " + clientNo + " at " + new Date() + '\n');

// Find the client's host name, and IP address

InetAddress inetAddress = socket.getInetAddress();

ta.appendText("Client " + clientNo + "'s host name is " + inetAddress.getHostName() + "\n");

ta.appendText("Client " + clientNo + "'s IP Address is " + inetAddress.getHostAddress() + "\n");

});

// Create data input and output streams

DataInputStream inputFromClient = new DataInputStream(socket.getInputStream());

DataOutputStream outputToClient = new DataOutputStream(socket.getOutputStream());

// Continuously serve the client

int r = inputFromClient.readInt();

int index = inputFromClient.readInt();

int sleeptime = inputFromClient.readInt();

// Record

Name.values()[index].setSleeptime(sleeptime);

// Name.values()[index].setPoint(0);

game.playerA = Name.values()[index];

outputToClient.writeInt(r);

System.out.print("| " + String.valueOf(r + 1) + " | " + String.format("%4d", sleeptime)

+ " | " + String.format("%9s", Name.values()[index].name()));

Platform.runLater(() -> {

ta.appendText("sleeptime received from client: " + sleeptime + '\n');

ta.appendText("Random Selection: " + Name.values()[index].name() + '\n');

ta.appendText("\n");

grid.add(new Text("\t" + String.valueOf(r + 1) + "\t"), 0, r + 2);

// random char column 6

grid.add(new Text(Name.values()[index].name()), 2, 2 + r);

// sleep column 5 row 3-4-5-..

grid.add(new Text(String.valueOf(Name.values()[index].getSleeptime())), 1, 2 + r);

});

number = 2;

condition2.signal();

lock.unlock();

}

} catch (IOException ex) {

ex.printStackTrace();

} catch (InterruptedException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

}

}

class HandleFromUDP implements Runnable {

private DatagramPacket rcvPkt;

private DatagramPacket sndPkt;

@Override

public void run() {

byte[] buf = new byte[1024];

try (DatagramSocket ss = new DatagramSocket(8000);) {

int i = 0;

while (i < 9) {

lock.lock();

if (number != 2)

condition2.await();

Arrays.fill(buf, (byte) 0);

rcvPkt = new DatagramPacket(buf, buf.length);

sndPkt = new DatagramPacket(buf, buf.length);

ss.receive(rcvPkt);

int round = (int) Double.parseDouble(new String(buf).trim());

// ta.appendText("UDPServer started at " + new Date() + '\n');

String app = "Received packet from " + rcvPkt.getAddress().getHostAddress() + ":" + rcvPkt.getPort()

+ "\n";

Platform.runLater(() -> {

ta.appendText("\n");

ta.appendText(app);

});

// 2.接收

Arrays.fill(buf, (byte) 0);

rcvPkt = new DatagramPacket(buf, buf.length);

ss.receive(rcvPkt);

// 并且赋值

int sleeptime = (int) Double.parseDouble(new String(buf).trim());

Arrays.fill(buf, (byte) 0);

rcvPkt = new DatagramPacket(buf, buf.length);

ss.receive(rcvPkt);

int index = (int) Double.parseDouble(new String(buf).trim());

Name.values()[index].setSleeptime(sleeptime);

// Name.values()[index].setPoint(0);

game.playerB = Name.values()[index];

Platform.runLater(() -> {

ta.appendText("Sleeptime: " + sleeptime + '\n' + "Random Selection:"

+ Name.values()[index].name() + '\n');

ta.appendText("\n");

// random char column 6

grid.add(new Text(Name.values()[index].name()), 5, 2 + round);

// sleep column 5 row 3-4-5-..

grid.add(new Text(String.valueOf(Name.values()[index].getSleeptime())), 4, 2 + round);

});

game.Pointsobtained(round);

// 发送分数到客户端

buf = new byte[1024];

Arrays.fill(buf, (byte) 0);

buf = Integer.valueOf(round).toString().getBytes();

sndPkt.setAddress(rcvPkt.getAddress());

sndPkt.setPort(rcvPkt.getPort());

sndPkt.setData(buf, 0, buf.length);

ss.send(sndPkt);

System.out.print("| " + String.format("%4d", game.playerA.getPoint()) + " | "

+ String.format("%4d", sleeptime) + " | "

+ String.format("%9s", Name.values()[index].name()) + "| "

+ String.format("%4d", game.playerB.getPoint()) + " |" + "\n");

Platform.runLater(() -> { // score a 4 row 3-4-5-..

grid.add(new Text(String.valueOf(game.playerA.getPoint())), 3, 2 + round);

grid.add(new Text(String.valueOf(game.playerB.getPoint())), 6, 2 + round);

});

i++;

game.finalwinner(i);

number = 1;

condition1.signal();

lock.unlock();

}

} catch (IOException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

} catch (InterruptedException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

}

}

}

//MultiClient.java

package project9;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

public class MultiClient {

public static void main(String[] args) {

ExecutorService executor = Executors.newFixedThreadPool(2);

executor.execute(new UDPClient() );

executor.execute(new Client());

executor.shutdown();

}

}

package project9;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.net.Socket;

public class Client implements Runnable {

@Override

public void run() {

try {

Socket socket = new Socket("127.0.0.1", 8000);

int round = 0;

while (true) {

int r = (int) (Math.random() \* 25) + 1;

int sleeptime = r \* r \* r % 1000;

int index = r % 3;

// IO Stream

DataOutputStream out = new DataOutputStream(socket.getOutputStream());

DataInputStream inpt = new DataInputStream(socket.getInputStream());

System.out.println("");

Thread.sleep(sleeptime);

// 发送轮

out.writeInt(round);

System.out.println("TCPSent Round: " + round);

// 发送index

out.writeInt(index);

System.out.println("TCPSent index: " + index);

// 发送睡眠时间

out.writeInt(sleeptime);

System.out.println("TCPSent sleeptime: " + sleeptime + '\n');

// 从服务器读取

round = inpt.readInt();

System.out.println("TCP-Received FromServer: " + round + "\n");

round++;

if (round == 9) {

break;

}

}

socket.close();

System.out.println("TCP Client closed");

} catch (IOException e) {

e.printStackTrace();

// TODO: handle exception

} catch (InterruptedException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

}

}

package project9;

import java.io.IOException;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.net.SocketException;

import java.net.UnknownHostException;

import java.util.Arrays;

public class UDPClient implements Runnable {

@Override

public void run() {

byte[] buf = new byte[1024];

DatagramPacket sndPkt;

DatagramPacket rcvPkt;

try (DatagramSocket socket = new DatagramSocket();) {

InetAddress server = InetAddress.getByName("localhost");

sndPkt = new DatagramPacket(buf, buf.length);

sndPkt.setAddress(server);

sndPkt.setPort(8000);

int round = 0;

while (true) {

int r = (int) (Math.random() \* 25) + 1;

int sleeptime = r \* r \* r % 1000;

int index = r % 3;

// 1.发送round

Arrays.fill(buf, (byte) 0);

buf = Double.valueOf(round).toString().getBytes();

sndPkt.setData(buf);

socket.send(sndPkt);

System.out.println("UDPSent Round: " + round);

Thread.sleep(sleeptime);

// 2.发送睡眠时间

buf = new byte[1024];

Arrays.fill(buf, (byte) 0);

buf = Double.valueOf(sleeptime).toString().getBytes();

sndPkt.setData(buf);

socket.send(sndPkt);

System.out.println("UDPSent sleeptime: " + sleeptime);

// 3.发送index

Arrays.fill(buf, (byte) 0);

buf = Double.valueOf(index).toString().getBytes();

sndPkt.setData(buf);

socket.send(sndPkt);

System.out.println("UDPSent index: " + index+'\n');

//从服务器接收

buf = new byte[1024];

rcvPkt = new DatagramPacket(buf, buf.length);

socket.receive(rcvPkt);

round = Integer.parseInt(new String(buf).trim());

round++;

if (round == 9)

break;

}

socket.close();

System.out.println("UDPClient closed");

} catch (SocketException e) {

e.printStackTrace();

} catch (UnknownHostException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}