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： 实验二：Java程序编制（基础练习）

项目完成人：

姓名： 王衔飞

学号：

指导教师： 程保中 孙艺

日 期： 年 月 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

通过使用Java语言进行基本程序的开发，练习继承、多态、接口、异常处理、多线程同步、Socket通信等。

## 实验内容

|  |  |
| --- | --- |
|  | 任务说明 |
|  | 创建一个类继承另一个类 |
|  | 创建一个类实现一个接口 |
|  | 多线程编程 |
|  | 网络多线程编程 |
|  |  |
|  |  |
|  |  |

## 实验环境

### 硬件环境

MacBook Air 2014 Core i5 1.4GHz 4GB Ram

### 网络环境

802.11ac：BUPT-portal

### 软件环境

macOS 10.15 Catalina JDK 1.8 IDE：IDEA Ultimate 2019.2

## 实验结果

完成实验。

# 实验内容

# Project #1分析设计

## N.1问题分析

### N.1.1 目标

### 创建一个父类叫GeometricObject，然后创建一个类Triangle并继承GeometricObject类。

### N.1.2功能

#### N.1.2.1功能一：输出信息

可以调用toString()方法输出当前对象的信息

#### N.1.2.2功能二：计算面积和周长

可以调用getArea() 和getPerimeter()方法来计算图形的面积和周长

### N.1.3性能

#### N.1.3.1计算的性能

非常快，无肉眼可见延时

#### N.1.3.2健壮性

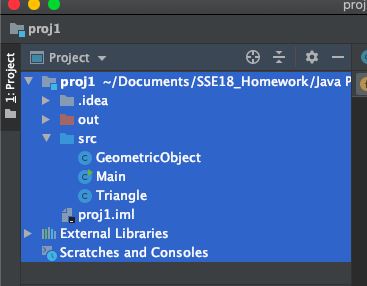
该程序没有输入

#### N.1.3.3异常输入处理

该程序没有输入

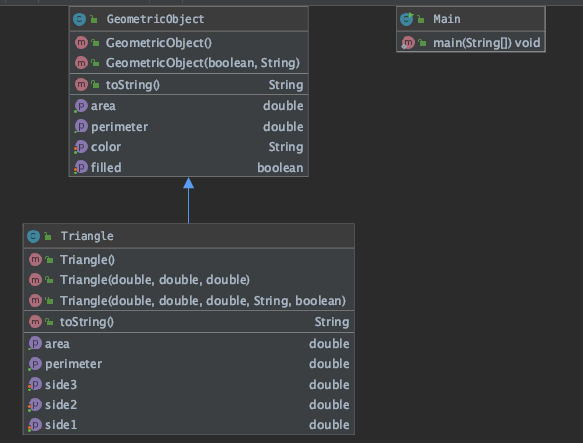
## N.2设计方案

### N.2.1模块划分



本项目包含三个模块 ，其中Main用于测试。

### N.2.2类图



### N.2.3算法

#### N.2.1.1算法一：计算面积：海伦公式，

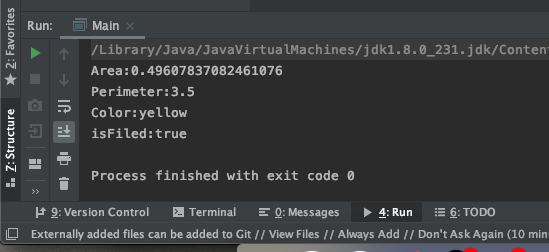
#### Math.sqrt((getPerimeter()/2)\*(getPerimeter()/2-side1)

#### \*(getPerimeter()/2-side2)\*(getPerimeter()/2-side3));

#### N.2.1.2算法二：计算周长：side1+side2+side3;

## N.3实验结果

运行结果满足题目要求，输出如图所示：



## N.4 调试心得

没有警告和错误，使用super调用父类函数可以提高编程效率。

## N.5源程序

/\* 文件名：Main.java

\*/

public class Main {

public static void main(String[] args) {

Triangle triangle = new Triangle(1,1.5,1,"yellow",true);

System.out.println("Area:"+triangle.getArea());

System.out.println("Perimeter:"+triangle.getPerimeter());

System.out.println("Color:"+triangle.getColor());

System.out.println("isFiled:"+triangle.isFilled());

}

}

/\* 文件名：GeometricObject.java

\*/

public class GeometricObject {

private boolean isFilled=false;

private String color;

public GeometricObject() {

}

public GeometricObject(boolean isFilled, String color) {

this.isFilled = isFilled;

this.color = color;

}

public boolean isFilled() {

return isFilled;

}

public void setFilled(boolean filled) {

isFilled = filled;

}

public String getColor() {

return color;

}

public void setColor(String color) {

this.color = color;

}

public double getPerimeter(){

return -1;

}

public double getArea(){

return -1;

}

@Override

public String toString() {

return " isFilled=" + isFilled + " color=" + color;

}

}

/\* 文件名：Triangle.java

\*/

public class Triangle extends GeometricObject{

private double side1=0,side2=0,side3=0;

public Triangle() {

}

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

this.side1 = side1;

this.side2 = side2;

this.side3 = side3;

}

public Triangle(double side1, double side2, double side3, String color,boolean isFilled) {

super(isFilled,color);

this.side1 = side1;

this.side2 = side2;

this.side3 = side3;

}

// accessor methods

public double getSide1() {

return side1;

}

public void setSide1(double side1) {

this.side1 = side1;

}

public double getSide2() {

return side2;

}

public void setSide2(double side2) {

this.side2 = side2;

}

public double getSide3() {

return side3;

}

public void setSide3(double side3) {

this.side3 = side3;

}

@Override

public double getPerimeter() {

return side1+side2+side3;

}

@Override

public double getArea() {

return Math.sqrt((getPerimeter()/2)\*(getPerimeter()/2-side1)

\*(getPerimeter()/2-side2)\*(getPerimeter()/2-side3));

}

@Override

public String toString() {

return "Triangle: side1 = " + side1 + " side2 = " + side2

+ " side3 = " + side3 + super.toString();

}

}

# Project #2分析设计

## N.1问题分析

### N.1.1 目标

### 创建一个父类叫GeometricObject，以及一个接口叫Colorable，然后创建一个类Square并继承GeometricObject类并实现Colorable接口。

创建一个Octagon 类继承GeometricObject类并实现 Comparable和Cloneable接口。

### N.1.2功能

#### N.1.2.1功能一：输出信息

可以调用toString()方法输出当前对象的信息

#### N.1.2.2功能二：克隆和比较

可以实现clone()方法和compareTo()方法。

### N.1.3性能

#### N.1.3.1计算的性能

非常快，无肉眼可见延时

#### N.1.3.2健壮性

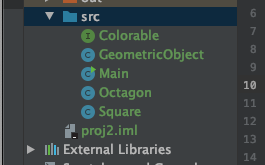
该程序没有输入

#### N.1.3.3异常输入处理

该程序没有输入

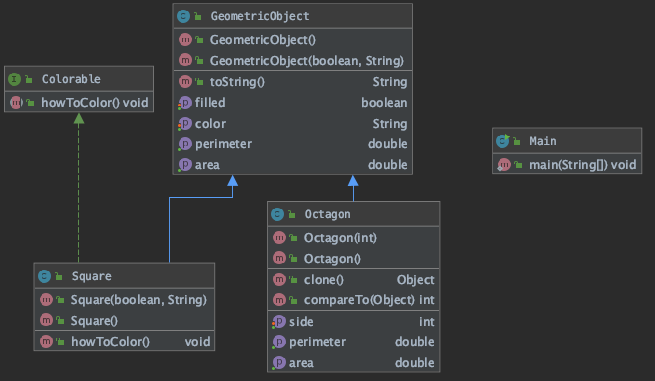
## N.2设计方案

### N.2.1模块划分



本项目包含三个模块 ，其中Main用于测试。

### N.2.2类图



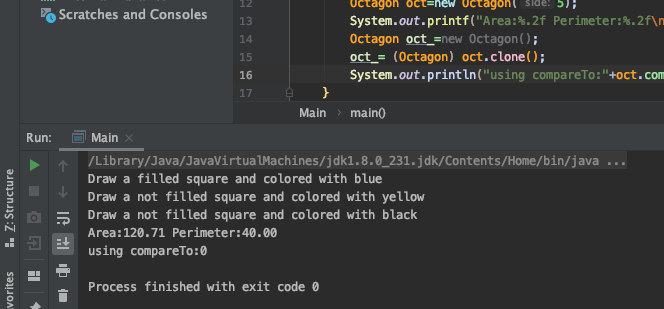
### N.2.3算法

#### N.2.1.1算法一：计算面积：题目中所给公式

#### N.2.1.2算法二：计算周长：8\*side;

## N.3实验结果

运行结果满足题目要求，输出如图所示：



## N.4 调试心得

没有警告和错误，clone方法可以直接调用上级的clone方法就行。

## N.5源程序

/\* 文件名：Main.java

\*/

public class Main {

public static void main(String[] args) throws CloneNotSupportedException {

// Write a test program that creates an array of five GeometricObject.

// For each object in the array, invoke its howToColor method if it is colorable.

GeometricObject[] gArrray={new GeometricObject(),new Square(true,"blue"),

new GeometricObject(),new Square(false,"yellow"),new Square()};

for (int i = 0; i <5 ; i++) if(gArrray[i] instanceof Colorable)((Square)gArrray[i]).howToColor();

// Write a test program that creates an Octagon object with side value 5

// and displays its area and perimeter. Create a new object

// using the clone method and compare the two objects using the compareTo method.

Octagon oct=new Octagon(5);

System.out.printf("Area:%.2f Perimeter:%.2f\n",oct.getArea(),oct.getPerimeter());

Octagon oct\_=new Octagon();

oct\_= (Octagon) oct.clone();

System.out.println("using compareTo:"+oct.compareTo(oct\_));

}

}

/\* 文件名：GeometricObject.java

\*/

public class GeometricObject {

private boolean isFilled=false;

private String color;

public GeometricObject() {

}

public GeometricObject(boolean isFilled, String color) {

this.isFilled = isFilled;

this.color = color;

}

public boolean isFilled() {

return isFilled;

}

public void setFilled(boolean filled) {

isFilled = filled;

}

public String getColor() {

return color;

}

public void setColor(String color) {

this.color = color;

}

public double getPerimeter(){

return -1;

}

public double getArea(){

return -1;

}

@Override

public String toString() {

return " isFilled=" + isFilled + " color=" + color;

}

}

/\* 文件名：Square.java

\*/

public class Square extends GeometricObject implements Colorable {

// Design a class named Square that extends GeometricObject and implements Colorable.

// Implement howToColor to display a message on how to color the square.

public Square(boolean isFilled, String color) {

super(isFilled, color);

}

public Square() {

}

@Override

public void howToColor() {

if(super.isFilled())System.out.println("Draw a square and colored with "+super.getColor());

else System.out.println("Draw square without filled");

}

}

/\* 文件名：Colorable.java

\*/

public interface Colorable {

// Design an interface named Colorable with a void method named howToColor().

// Every class of a colorable object must implement the Colorable interface.

void howToColor();

}

/\* 文件名：Octagon.java

\*/

public class Octagon extends GeometricObject implements Comparable,Cloneable {

private int side=0;

public Octagon(int side) {

this.side = side;

}

public Octagon() {

}

@Override

public Object clone() throws CloneNotSupportedException {

return super.clone();

}

@Override

public int compareTo(Object o) {

return side-((Octagon)o).getSide();

}

public int getSide() {

return side;

}

public void setSide(int side) {

this.side = side;

}

@Override

public double getPerimeter(){

return 8\*side;

}

@Override

public double getArea() {

return (2+4/Math.sqrt(2))\*side\*side;

}

}

# Project #3分析设计

## N.1问题分析

### N.1.1 目标

创建三个进程，两个用于产生随机数及随机睡眠，另一个用于比较和输出。

### N.1.2功能

N.1.2.1功能一：睡眠及等待

### N.1.3性能

#### N.1.3.1计算的性能

计算几乎没有时间

#### N.1.3.2健壮性

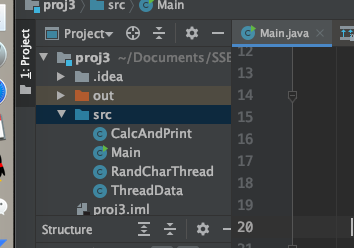
该程序没有输入，会抛出异常部分已进行处理

#### N.1.3.3异常输入处理

该程序没有输入

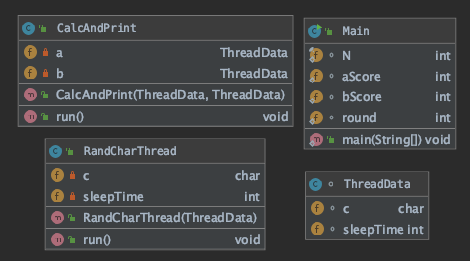
## N.2设计方案

### N.2.1模块划分



本项目包含四个模块。

### N.2.2类图

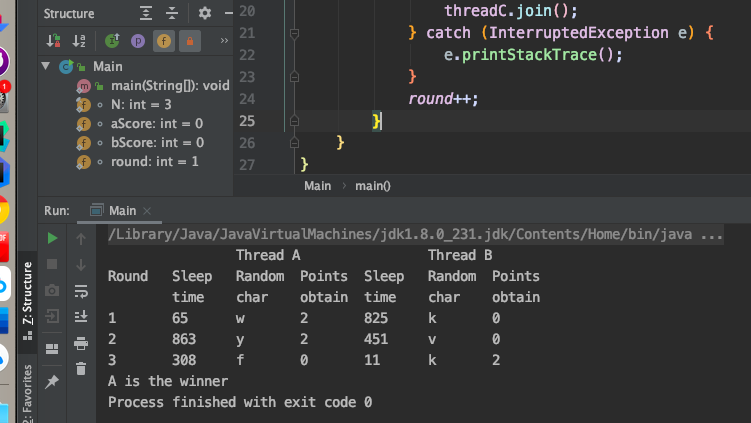


### N.2.3算法

#### N.2.1.1算法一：产生随机数 ：使用Random类中的nextInt()产生

## N.3实验结果

运行结果满足题目要求，输出如图所示：



## N.4 调试心得

没有警告和错误，该程序主要时间用在了等待上，判断和产生随机数线程顺序执行所以不需要处理临界区问题。

## N.5源程序

/\* 文件名：Main.java

\*/

public class Main {

static final int N = 3;

static int aScore=0,bScore=0,round=1;

public static void main(String[] args) {

System.out.println("\t\t\t\tThread A\t\t\t\tThread B");

System.out.println("Round\tSleep\tRandom\tPoints\tSleep\tRandom\tPoints\t");

System.out.println("\t\ttime\tchar\tobtain\ttime\tchar\tobtain\t");

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

ThreadData dataA = new ThreadData(),dataB = new ThreadData();

RandCharThread threadA = new RandCharThread(dataA);

RandCharThread threadB = new RandCharThread(dataB);

Thread threadC= new Thread(new CalcAndPrint(dataA,dataB));

try {

threadA.start();

threadB.start();

threadA.join();

threadB.join();

threadC.start();

threadC.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

round++;

}

}

}

/\* 文件名：ThreadData.java

\*/

class ThreadData {

char c=0;

int sleepTime=0;

}

/\* 文件名：CalcAndPrint.java

\*/

public class CalcAndPrint implements Runnable {

private ThreadData a, b;

public CalcAndPrint(ThreadData a, ThreadData b) {

this.a = a;

this.b = b;

}

@Override

public void run() {

if (a.c > b.c) {

System.out.println(Main.round+"\t\t"+a.sleepTime+"\t\t"

+a.c+"\t\t"+2+"\t\t"+b.sleepTime+"\t\t"+b.c+"\t\t"+0);

Main.aScore += 2;

} else if (a.c < b.c) {

System.out.println(Main.round+"\t\t"+a.sleepTime+"\t\t"

+a.c+"\t\t"+0+"\t\t"+b.sleepTime+"\t\t"+b.c+"\t\t"+2);

Main.bScore += 2;

} else {

System.out.println(Main.round+"\t\t"+a.sleepTime+"\t\t"

+a.c+"\t\t"+1+"\t\t"+b.sleepTime+"\t\t"+b.c+"\t\t"+1);

Main.aScore++;

Main.bScore++;

}

if(Main.round==Main.N){

if(Main.aScore>Main.bScore)System.out.print("A is the winner");

else if(Main.aScore<Main.bScore)System.out.print("B is the winner");

else System.out.print("None is the winner");

}

}

}

/\* 文件名：RandCharThread.java

\*/

import java.util.Random;

public class RandCharThread extends Thread{

private char c=0;

private int sleepTime=0;

public RandCharThread(ThreadData threadData) {

threadData.c=(char)(new Random().nextInt(26)+'a');

threadData.sleepTime = new Random().nextInt(1000);

this.c = threadData.c;

this.sleepTime = threadData.sleepTime;

}

@Override

public void run() {

try {

sleep(sleepTime);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

# Project #4分析设计

## N.1问题分析

### N.1.1 目标

分两个可执行程序，一个是客户端，分别用udp和tcp发送石头剪子布的游戏数据，另一个是服务器端，分别用udp和tcp接收游戏数据并统计和输出。

### N.1.2功能

N.1.2.1功能一：客户端：分别用udp和tcp发送石头剪子布的游戏数据

N.1.2.2功能二：服务器端：分别用udp和tcp接收游戏数据并统计和输出

### N.1.3性能

#### N.1.3.1计算的性能

计算几乎没有时间

#### N.1.3.2健壮性

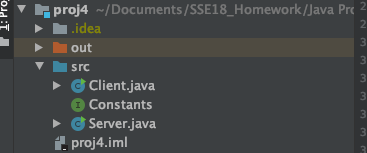
该程序没有输入，会抛出异常部分已进行处理

#### N.1.3.3异常输入处理

该程序没有输入

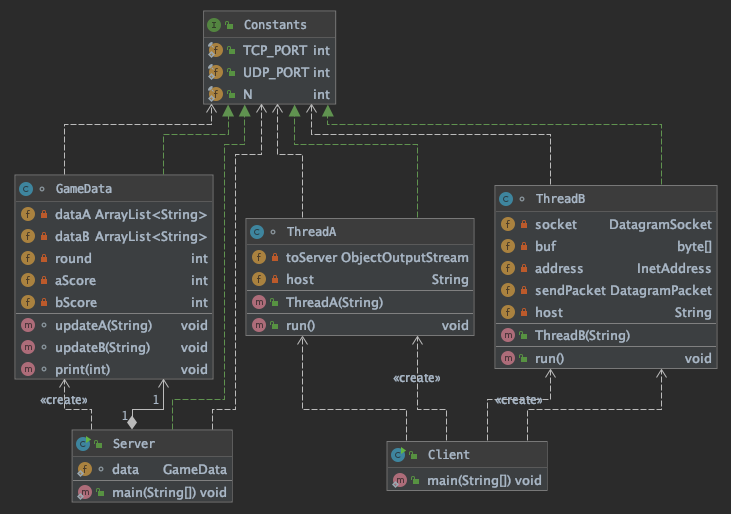
## N.2设计方案

### N.2.1模块划分



本项目包含三个模块。其中server与client均可执行

### N.2.2类图



### N.2.3算法

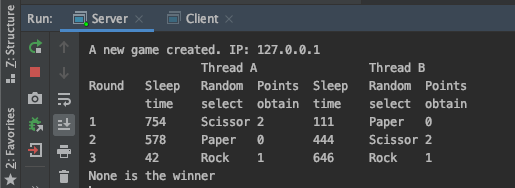
#### N.2.1.1算法一：产生随机数 ：使用Random类中的nextInt()产生

#### N.2.1.2算法二：传输：UDP和TCP统一以String对象传输，格式为睡眠时间-结果，TCP使用ObjectOutputStream对象，UDP使用String(byte [])构造方法。

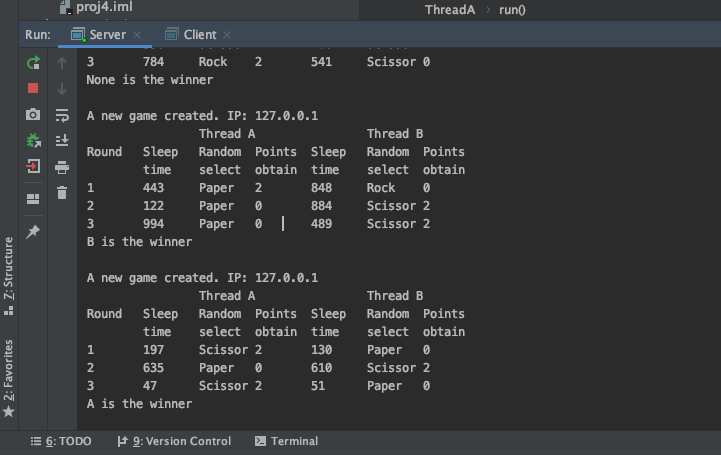
## N.3实验结果

运行结果满足题目要求，输出如图所示：

先运行Server再运行Client



不结束Server进程可多次运行Client，会将每次的游戏结果依次输出：



## N.4 调试心得

没有警告和错误，该程序主要时间用在了等待上，判断和产生随机数线程顺序执行所以不需要处理临界区问题。运行时必须先运行server再运行client否则将会抛出异常。

## N.5源程序

/\* 文件名：Client.java

\*/

import java.io.\*;

import java.net.\*;

import java.util.Arrays;

import java.util.Random;

public class Client {

public static void main(String argv[]) {

// TCP

new ThreadA("localhost").start();

// UDP

new ThreadB("localhost").start();

}

}

class ThreadA extends Thread implements Constants{

private ObjectOutputStream toServer = null;

private String host;

public ThreadA(String host) {

this.host = host;

}

@Override

public void run() {

try {

Socket socket = new Socket(host, TCP\_PORT);

//fromServer = new ObjectInputStream(socket.getInputStream());

toServer = new ObjectOutputStream(socket.getOutputStream());

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

int sleepTime = new Random().nextInt(1000);

int randHand = new Random().nextInt(3);

sleep(sleepTime);

switch (randHand) {

case 0:

toServer.writeObject(sleepTime + "-" + "Rock");

break;

case 1:

toServer.writeObject(sleepTime + "-" + "Paper");

break;

case 2:

toServer.writeObject(sleepTime + "-" + "Scissor");

break;

}

// System.out.println(sleepTime);

}

socket.close();

} catch (IOException|InterruptedException e) {

e.printStackTrace();

}

}

}

class ThreadB extends Thread implements Constants{

private DatagramSocket socket;

private byte[] buf = new byte[256]; // The byte array for sending and receiving datagram packets

private InetAddress address; // Server InetAddress

private DatagramPacket sendPacket; // The packet sent to the server

private String host;

public ThreadB(String host) {

this.host = host;

}

@Override

public void run() {

try {

Arrays.fill(buf, (byte) 0); // Initialize buffer for each iteration

socket = new DatagramSocket();

address = InetAddress.getByName(host);

sendPacket = new DatagramPacket(buf, buf.length, address, UDP\_PORT);

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

int sleepTime = new Random().nextInt(1000);

int randHand = new Random().nextInt(3);

sleep(sleepTime);

switch (randHand) {

case 0:

sendPacket.setData((sleepTime + "-" + "Rock").getBytes());

break;

case 1:

sendPacket.setData((sleepTime + "-" + "Paper").getBytes());

break;

case 2:

sendPacket.setData((sleepTime + "-" + "Scissor").getBytes());

break;

}

socket.send(sendPacket);

// System.out.println(sleepTime);

}

socket.close();

} catch (IOException | InterruptedException ex) {

ex.printStackTrace();

}

}

}

/\* 文件名：Server.java

\*/

import java.io.\*;

import java.net.\*;

import java.util.ArrayList;

import java.util.Arrays;

public class Server implements Constants {

static GameData data=new GameData();

public static void main(String[] argv){

// 接受TCP

new Thread(()->{

while (true)

try {

ServerSocket serverSocket = new ServerSocket(TCP\_PORT);

Socket socket = serverSocket.accept();

InetAddress inetAddress = socket.getInetAddress();

System.out.println("\nA new game created. IP: " + inetAddress.getHostAddress());

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

// ObjectOutputStream outputToClient = new ObjectOutputStream(socket.getOutputStream());

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

String gotStr=inputFromClient.readObject().toString();

//System.out.println("tcp"+gotStr);

data.updateA(gotStr);

}

} catch (IOException | ClassNotFoundException ignored) {

}

}).start();

// 接受UDP

new Thread(()->{

byte[] buf = new byte[64];

while (true)

try {

DatagramSocket socket = new DatagramSocket(UDP\_PORT);

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

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

Arrays.fill(buf, (byte) 0); // Initialize buffer for each iteration

socket.receive(receivePacket); // Receive from the client in a packet

//System.out.println("udp"+new String(buf).trim());

data.updateB(new String(buf).trim());

}

} catch (IOException ignored) {

}

}).start();

}

}

class GameData implements Constants{

private ArrayList<String> dataA=new ArrayList<>();

private ArrayList<String> dataB=new ArrayList<>();

private int round=0,aScore=0,bScore=0;

void updateA(String str){

dataA.add(str);

if(dataA.size()>round&&dataB.size()>round) print(round++);

}

void updateB(String str){

dataB.add(str);

if(dataA.size()>round&&dataB.size()>round) print(round++);

}

void print(int round){

if(round==0){

System.out.println("\t\t\t\tThread A\t\t\t\tThread B");

System.out.println("Round\tSleep\tRandom\tPoints\tSleep\tRandom\tPoints\t");

System.out.println("\t\ttime\tselect\tobtain\ttime\tselect\tobtain\t");

}

String[] a=dataA.get(round).split("-");

String[] b=dataB.get(round).split("-");

if (a[1].equals(b[1])) {

System.out.println((round+1)+"\t\t"+a[0]+"\t\t"+a[1]+"\t"+1+"\t\t"+b[0]+"\t\t"+b[1]+"\t"+1);

aScore++;

bScore++;

} else if ((a[1].equals("Rock")&&b[1].equals("Paper"))||(a[1].equals("Paper")&&b[1].equals("Scissor"))

||(a[1].equals("Scissor")&&b[1].equals("Rock"))) {

System.out.println((round+1)+"\t\t"+a[0]+"\t\t"+a[1]+"\t"+0+"\t\t"+b[0]+"\t\t"+b[1]+"\t"+2);

bScore += 2;

} else {

System.out.println((round+1)+"\t\t"+a[0]+"\t\t"+a[1]+"\t"+2+"\t\t"+b[0]+"\t\t"+b[1]+"\t"+0);

aScore += 2;

}

if(round==N-1){

if(aScore>bScore)System.out.println("A is the winner");

else if(aScore<bScore)System.out.println("B is the winner");

else System.out.println("None is the winner");

// reset and ready for a new game

this.round=aScore=bScore=0;

dataA.clear();

dataB.clear();

}

}

}

/\* 文件名：Constant.java

\*/

public interface Constants {

static final int TCP\_PORT = 8000;

static final int UDP\_PORT = 8001;

static final int N = 3;

}