java 实现游戏模拟实验报告

2025 年 3 月 19 日

一、类设计思路：

由 actor 实现 Canplay 接口，同时 actor 是所有职业的抽象基类，actor 中定义了诸如血量，攻击，防御，异常 buff 等所有职业共有的属性，也定义了 attack 和 defense 行为，由 action 函数将攻击与防御行为封装在一起，使每回合只需要分别调用两个玩家的 action 函数即可，而对于各个职业不同的部分，actor 中给出了 attackway 这一抽象方法供各个职业类重写，以此实现了多态性

由 Game 定义 play 方法供两种游戏模式重写

Game1：玩家选择狂战士或者冰封法师中的一个，系统选择另一个，系统会随机出招，而玩家会通过输入框输入出招

Game2：玩家输入两个角色的职业

RandPlay：由电脑控制两个角色随机出招

Play（无参）：测试用，电脑控制两个角色一直攻击

Play（有参）：由玩家每次输入两个字符串分别控制 player1 和 player2 的行为

冰封法师：每次攻击为敌方累积一点冰封值，冰封值到 2 会冰封敌人（一回合无法行动），若敌方不为冰封法师则对敌方造成两次攻击

狂战士：每次攻击前攻击力提升相当于损失血量百分之 50 的数值，若敌方是狂战士则对敌方造成两次攻击

血魔：每次攻击为敌方附加一层流血，每层流血使敌人每回合损失 20 血量，生命值首次低于 200 时会吸取敌方相当于攻击力百分百的血量

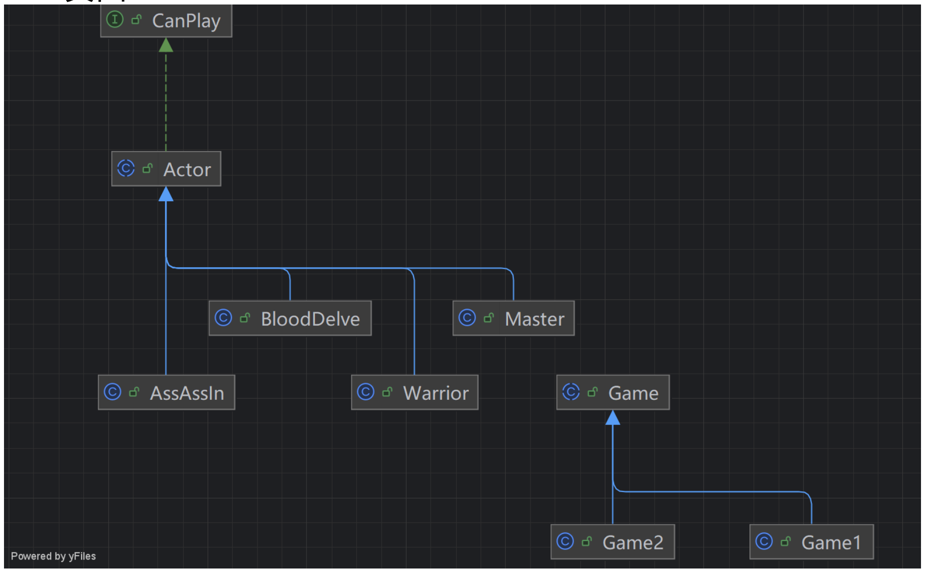
刺客：每回合进行判定，若判定成功则对敌方额外造成一次相当于攻击力百分之 200 的伤害

防御使防御力 *2，损失血量 = 敌方攻击力-自身防御力

暴击：每次进行判定，若判定成功则流失双倍血量

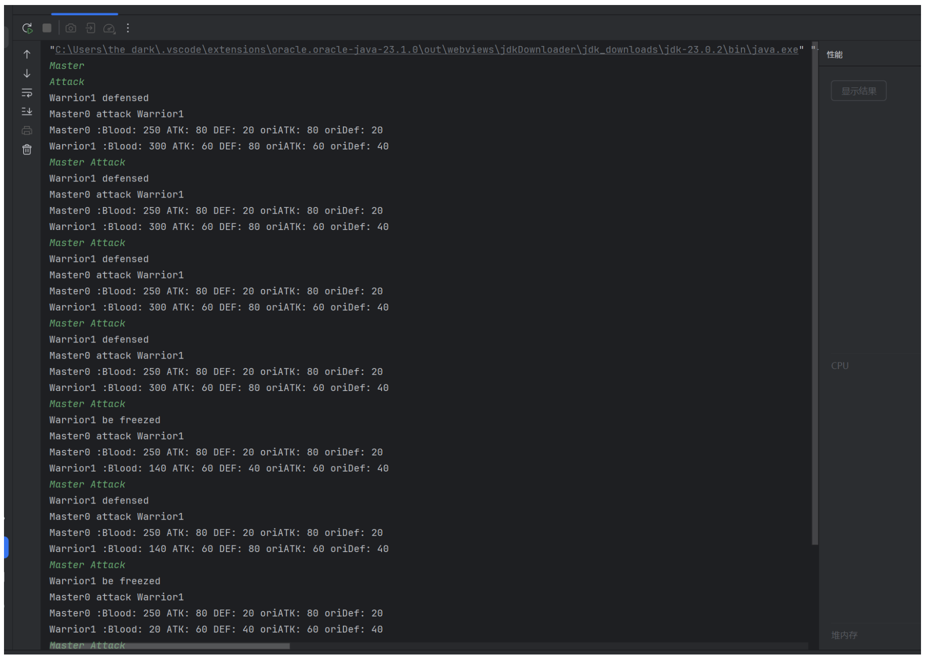
网络：使用两条线程，服务器每条线程监听一个客户端，两个客户端每回合分别将玩法传给服务器，服务器返回两个客户端传输的内容，若游戏结束，服务器返回-1

二. 类图



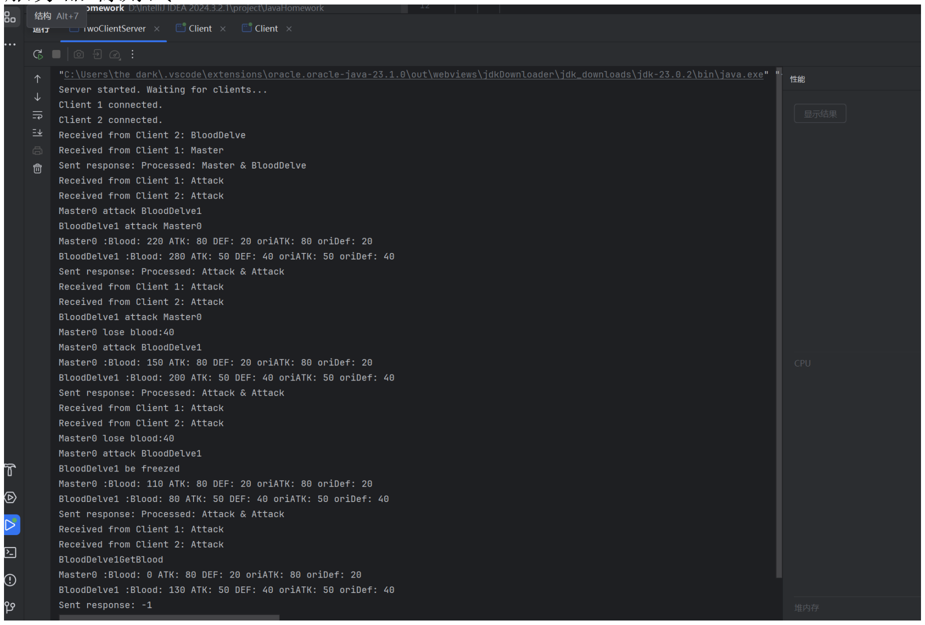
三. 测试案例设计

Game1 测试：



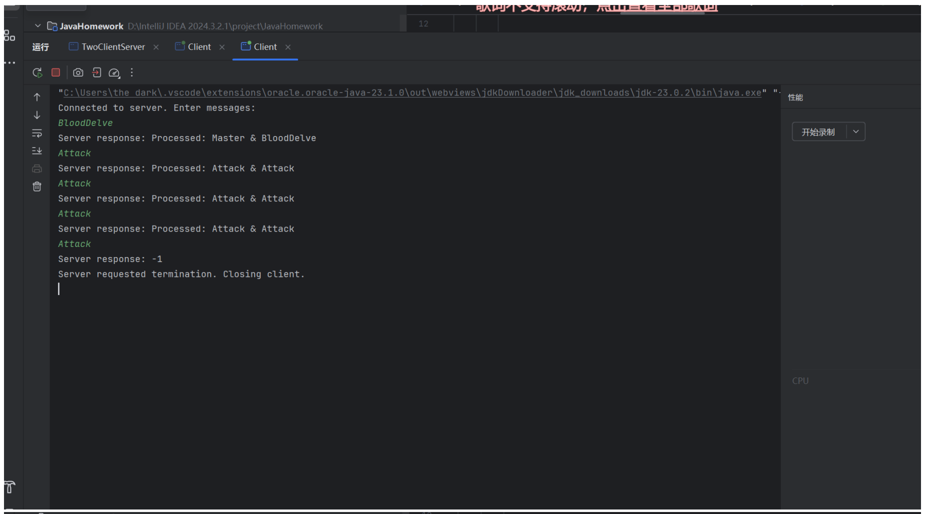
```
"C:\Users\the dark\.vscode\extensions\oracle.oracle-java-23.1.0\out\webviews\jdkDownloader\jdk_downloads\jdk-23.0.2\bin\java.exe"
Master
Attack
Warrior1 defensed
Master0 attack Warrior1
Master0 :Blood: 250 ATK: 80 DEF: 20 oriATK: 80 oriDef: 20
Warrior1 :Blood: 300 ATK: 60 DEF: 80 oriATK: 60 oriDef: 40
Master Attack
Warrior1 defensed
Master0 attack Warrior1
Master0 :Blood: 250 ATK: 80 DEF: 20 oriATK: 80 oriDef: 20
Warrior1 :Blood: 300 ATK: 60 DEF: 80 oriATK: 60 oriDef: 40
Master Attack
Warrior1 defensed
Master0 attack Warrior1
Master0 :Blood: 250 ATK: 80 DEF: 20 oriATK: 80 oriDef: 20
Warrior1 :Blood: 300 ATK: 60 DEF: 80 oriATK: 60 oriDef: 40
Master Attack
Warrior1 defensed
Master0 attack Warrior1
Master0 :Blood: 250 ATK: 80 DEF: 20 oriATK: 80 oriDef: 20
Warrior1 :Blood: 300 ATK: 60 DEF: 80 oriATK: 60 oriDef: 40
Master Attack
Warrior1 be freezed
Master0 attack Warrior1
Master0 :Blood: 250 ATK: 80 DEF: 20 oriATK: 80 oriDef: 20
Warrior1 :Blood: 140 ATK: 60 DEF: 40 oriATK: 60 oriDef: 40
Master Attack
Warrior1 defensed
Master0 attack Warrior1
Master0 :Blood: 250 ATK: 80 DEF: 20 oriATK: 80 oriDef: 20
Warrior1 :Blood: 140 ATK: 60 DEF: 80 oriATK: 60 oriDef: 40
Master Attack
Warrior1 be freezed
Master0 attack Warrior1
Master0 :Blood: 250 ATK: 80 DEF: 20 oriATK: 80 oriDef: 20
Warrior1 :Blood: 20 ATK: 60 DEF: 40 oriATK: 60 oriDef: 40
Master Attack
```

服务器端测试：

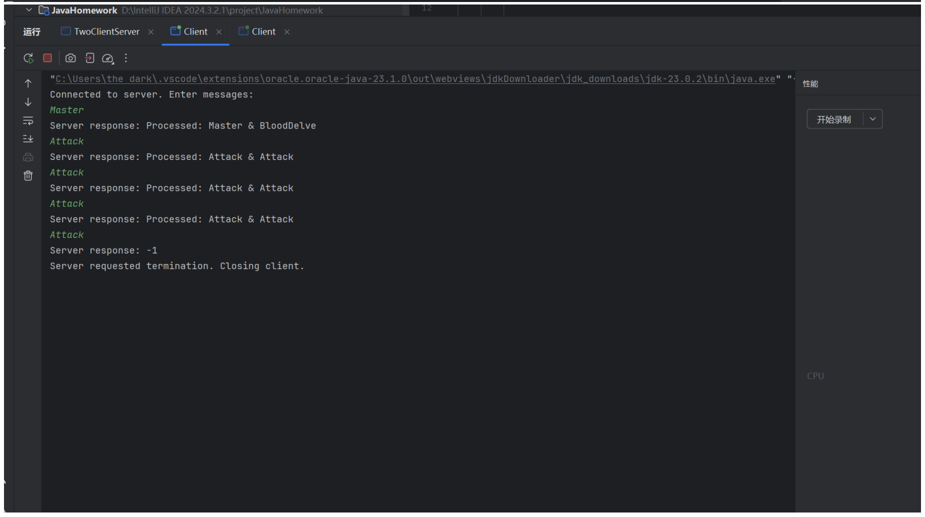


```
"C:\Users\the dark\.vscode\extensions\oracle.oracle-java-23.1.0\out\webviews\jdkDownloader\jdk_downloads\jdk-23.0.2\bin\java.exe"
Server started. Waiting for clients...
Client 1 connected.
Client 2 connected.
Received from Client 2: BloodDelve
Received from Client 1: Master
Sent response: Processed: Master & BloodDelve
Received from Client 1: Attack
Received from Client 2: Attack
Master0 attack BloodDelve1
BloodDelve1 attack Master0
Master0 :Blood: 220 ATK: 80 DEF: 20 oriATK: 80 oriDef: 20
BloodDelve1 :Blood: 280 ATK: 50 DEF: 40 oriATK: 50 oriDef: 40
Sent response: Processed: Attack & Attack
Received from Client 1: Attack
Received from Client 2: Attack
BloodDelve1 attack Master0
Master0 lose blood:40
Master0 attack BloodDelve1
Master0 :Blood: 150 ATK: 80 DEF: 20 oriATK: 80 oriDef: 20
BloodDelve1 :Blood: 200 ATK: 50 DEF: 40 oriATK: 50 oriDef: 40
Sent response: Processed: Attack & Attack
Received from Client 1: Attack
Received from Client 2: Attack
Master0 lose blood:40
Master0 attack BloodDelve1
BloodDelve1 be freezed
Master0 :Blood: 110 ATK: 80 DEF: 20 oriATK: 80 oriDef: 20
BloodDelve1 :Blood: 80 ATK: 50 DEF: 40 oriATK: 50 oriDef: 40
Sent response: Processed: Attack & Attack
Received from Client 1: Attack
Received from Client 2: Attack
BloodDelve1GetBlood
Master0 :Blood: 0 ATK: 80 DEF: 20 oriATK: 80 oriDef: 20
BloodDelve1 :Blood: 130 ATK: 50 DEF: 40 oriATK: 50 oriDef: 40
Sent response: -1
```

两个客户端测试：

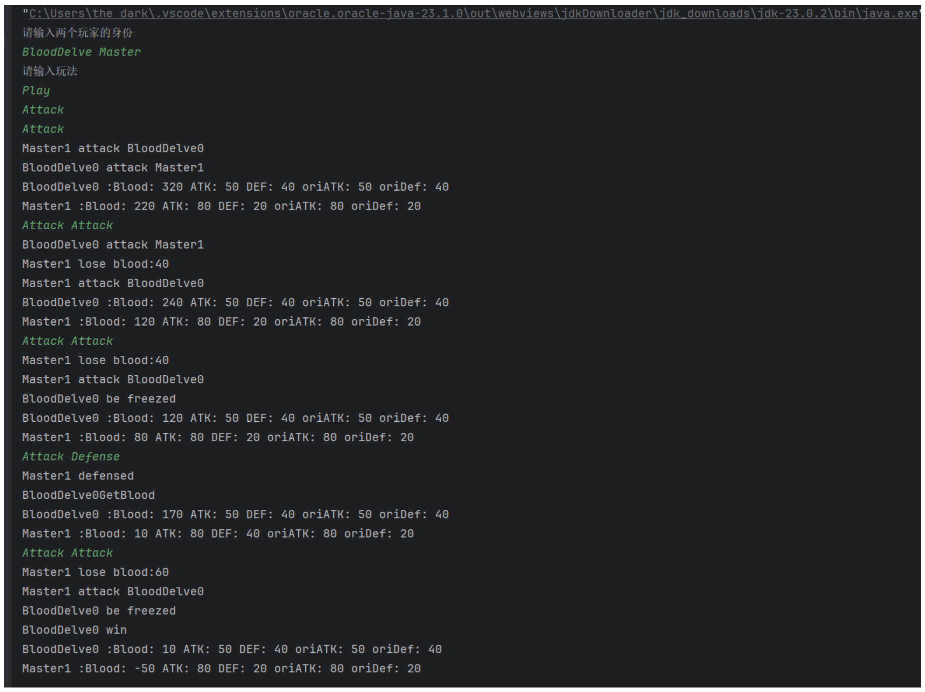

JavaHomework D:\IntelliJ IDEA 2024.3.2.1\project\JavaHomework
运行
TwoClientServer
Client
Client
性能
开始录制
CPU

```
"C:\Users\the dark\.vscode\extensions\oracle.oracle-java-23.1.0\out\webviews\jdkDownloader\jdk_downloads\jdk-23.0.2\bin\java.exe" "
Connected to server. Enter messages:
BloodDelve
Server response: Processed: Master & BloodDelve
Attack
Server response: Processed: Attack & Attack
Attack
Server response: Processed: Attack & Attack
Attack
Server response: Processed: Attack & Attack
Attack
Server response: -1
Server requested termination. Closing client.
```


JavaHomework D:\IntelliJ IDEA 2024.3.2.1\project\JavaHomework
运行
TwoClientServer
Client
Client
性能
开始录制
CPU

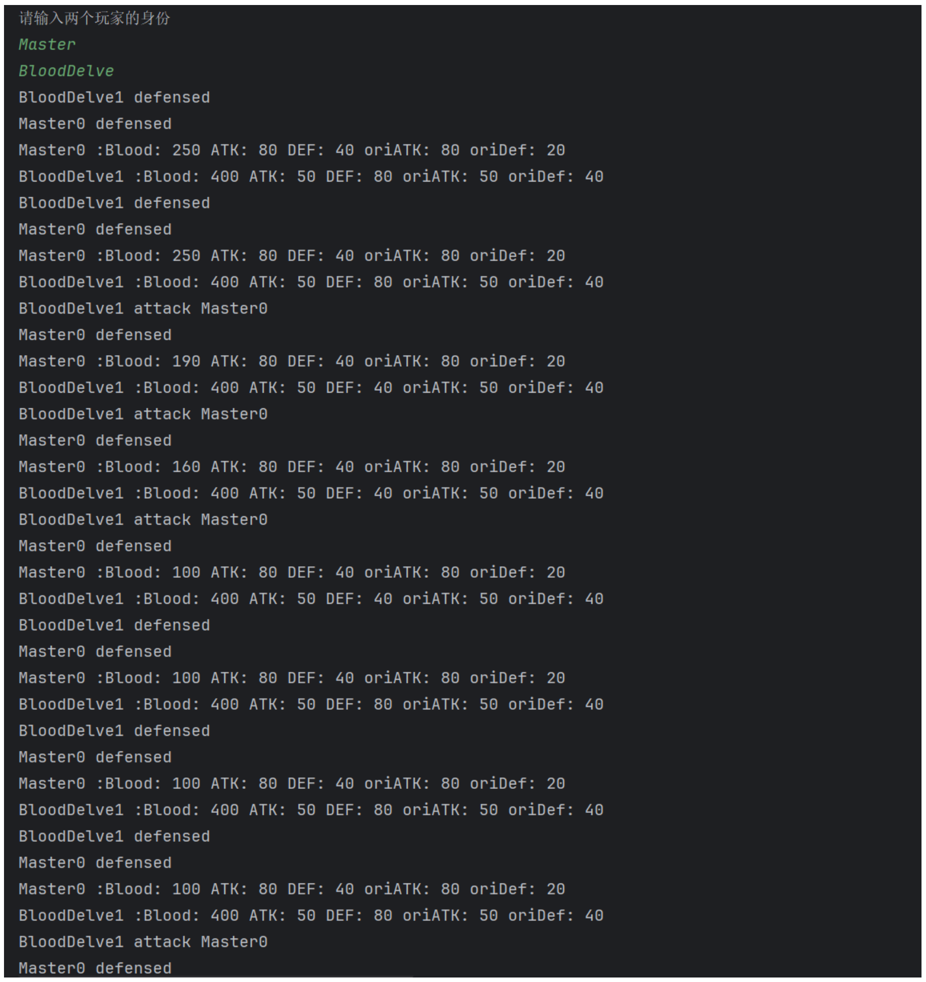
```
"C:\Users\the dark\.vscode\extensions\oracle.oracle-java-23.1.0\out\webviews\jdkDownloader\jdk_downloads\jdk-23.0.2\bin\java.exe" "
Connected to server. Enter messages:
Master
Server response: Processed: Master & BloodDelve
Attack
Server response: Processed: Attack & Attack
Attack
Server response: Processed: Attack & Attack
Attack
Server response: Processed: Attack & Attack
Attack
Server response: -1
Server requested termination. Closing client.
```

Game2 全输入测试：

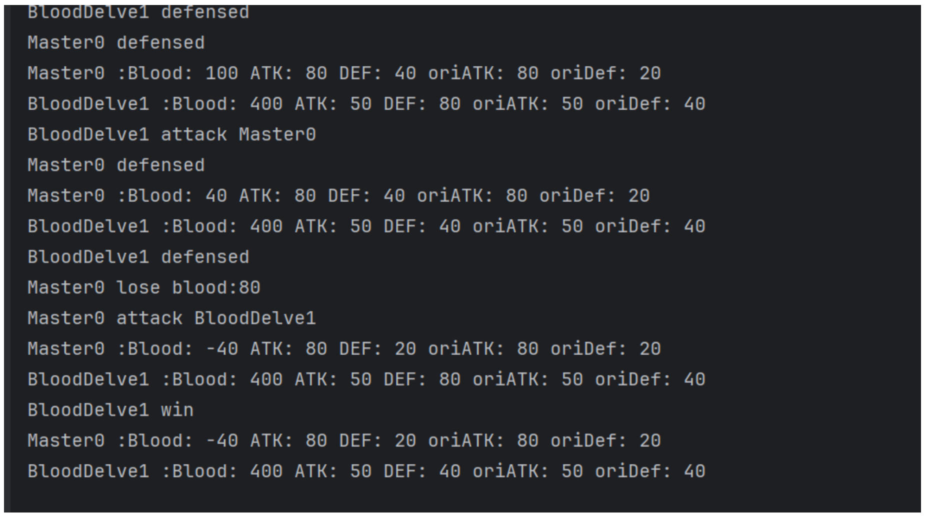


```
"C:\Users\the dark\.vscode\extensions\oracle.oracle-java-23.1.0\out\webviews\jdkDownloader\jdk_downloads\jdk-23.0.2\bin\java.exe
请输入两个玩家的身份
BloodDelve Master
请输入玩法
Play
Attack
Attack
Master1 attack BloodDelve0
BloodDelve0 attack Master1
BloodDelve0 :Blood: 320 ATK: 50 DEF: 40 oriATK: 50 oriDef: 40
Master1 :Blood: 220 ATK: 80 DEF: 20 oriATK: 80 oriDef: 20
Attack Attack
BloodDelve0 attack Master1
Master1 lose blood:40
Master1 attack BloodDelve0
BloodDelve0 :Blood: 240 ATK: 50 DEF: 40 oriATK: 50 oriDef: 40
Master1 :Blood: 120 ATK: 80 DEF: 20 oriATK: 80 oriDef: 20
Attack Attack
Master1 lose blood:40
Master1 attack BloodDelve0
BloodDelve0 be freezed
BloodDelve0 :Blood: 120 ATK: 50 DEF: 40 oriATK: 50 oriDef: 40
Master1 :Blood: 80 ATK: 80 DEF: 20 oriATK: 80 oriDef: 20
Attack Defense
Master1 defensed
BloodDelve0GetBlood
BloodDelve0 :Blood: 170 ATK: 50 DEF: 40 oriATK: 50 oriDef: 40
Master1 :Blood: 10 ATK: 80 DEF: 40 oriATK: 80 oriDef: 20
Attack Attack
Master1 lose blood:60
Master1 attack BloodDelve0
BloodDelve0 be freezed
BloodDelve0 win
BloodDelve0 :Blood: 10 ATK: 50 DEF: 40 oriATK: 50 oriDef: 40
Master1 :Blood: -50 ATK: 80 DEF: 20 oriATK: 80 oriDef: 20
```

Game2 全随机测试：



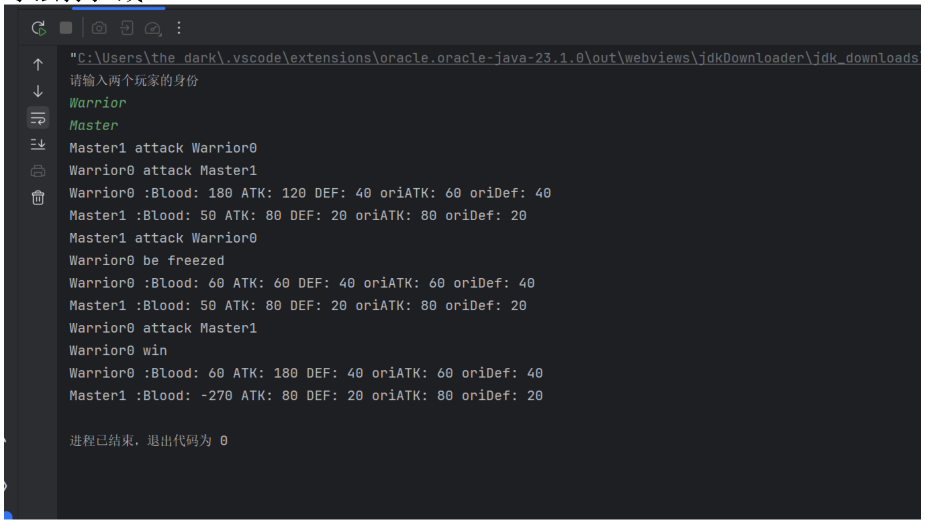
```
请输入两个玩家的身份
Master
BloodDelve
BloodDelve1 defensed
Master0 defensed
Master0 :Blood: 250 ATK: 80 DEF: 40 oriATK: 80 oriDef: 20
BloodDelve1 :Blood: 400 ATK: 50 DEF: 80 oriATK: 50 oriDef: 40
BloodDelve1 defensed
Master0 defensed
Master0 :Blood: 250 ATK: 80 DEF: 40 oriATK: 80 oriDef: 20
BloodDelve1 :Blood: 400 ATK: 50 DEF: 80 oriATK: 50 oriDef: 40
BloodDelve1 attack Master0
Master0 defensed
Master0 :Blood: 190 ATK: 80 DEF: 40 oriATK: 80 oriDef: 20
BloodDelve1 :Blood: 400 ATK: 50 DEF: 40 oriATK: 50 oriDef: 40
BloodDelve1 attack Master0
Master0 defensed
Master0 :Blood: 160 ATK: 80 DEF: 40 oriATK: 80 oriDef: 20
BloodDelve1 :Blood: 400 ATK: 50 DEF: 40 oriATK: 50 oriDef: 40
BloodDelve1 attack Master0
Master0 defensed
Master0 :Blood: 100 ATK: 80 DEF: 40 oriATK: 80 oriDef: 20
BloodDelve1 :Blood: 400 ATK: 50 DEF: 40 oriATK: 50 oriDef: 40
BloodDelve1 defensed
Master0 defensed
Master0 :Blood: 100 ATK: 80 DEF: 40 oriATK: 80 oriDef: 20
BloodDelve1 :Blood: 400 ATK: 50 DEF: 80 oriATK: 50 oriDef: 40
BloodDelve1 defensed
Master0 defensed
Master0 :Blood: 100 ATK: 80 DEF: 40 oriATK: 80 oriDef: 20
BloodDelve1 :Blood: 400 ATK: 50 DEF: 80 oriATK: 50 oriDef: 40
BloodDelve1 defensed
Master0 defensed
Master0 :Blood: 100 ATK: 80 DEF: 40 oriATK: 80 oriDef: 20
BloodDelve1 :Blood: 400 ATK: 50 DEF: 80 oriATK: 50 oriDef: 40
BloodDelve1 attack Master0
Master0 defensed
```



```
BloodDelve1 defensed
Master0 defensed
Master0 :Blood: 100 ATK: 80 DEF: 40 oriATK: 80 oriDef: 20
BloodDelve1 :Blood: 400 ATK: 50 DEF: 80 oriATK: 50 oriDef: 40
BloodDelve1 attack Master0
Master0 defensed
Master0 :Blood: 40 ATK: 80 DEF: 40 oriATK: 80 oriDef: 20
BloodDelve1 :Blood: 400 ATK: 50 DEF: 40 oriATK: 50 oriDef: 40
BloodDelve1 defensed
Master0 lose blood:80
Master0 attack BloodDelve1
Master0 :Blood: -40 ATK: 80 DEF: 20 oriATK: 80 oriDef: 20
BloodDelve1 :Blood: 400 ATK: 50 DEF: 80 oriATK: 50 oriDef: 40
BloodDelve1 win
Master0 :Blood: -40 ATK: 80 DEF: 20 oriATK: 80 oriDef: 20
BloodDelve1 :Blood: 400 ATK: 50 DEF: 40 oriATK: 50 oriDef: 40
```

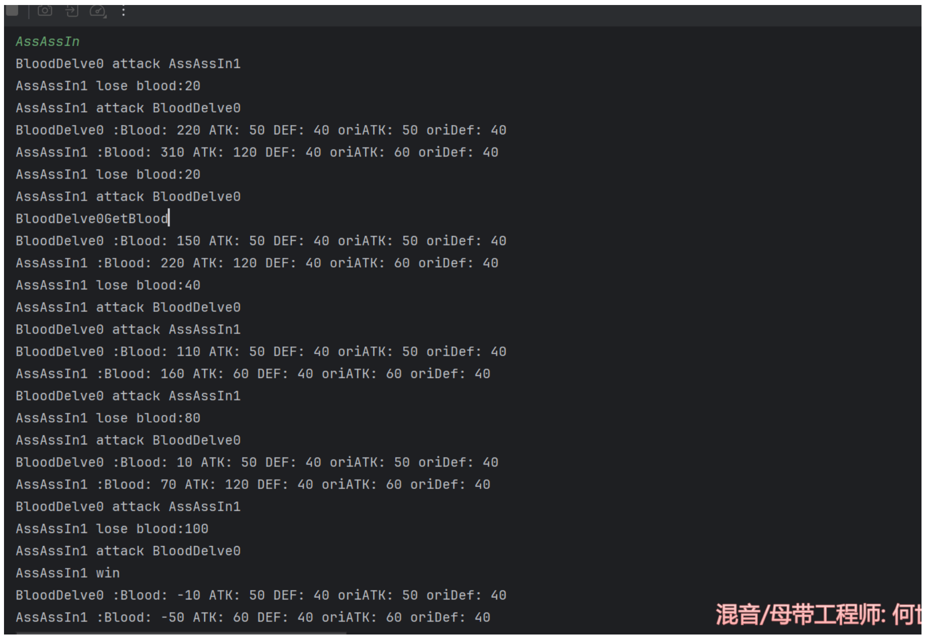
Game2 全攻击测试：

冰法打狂战士：



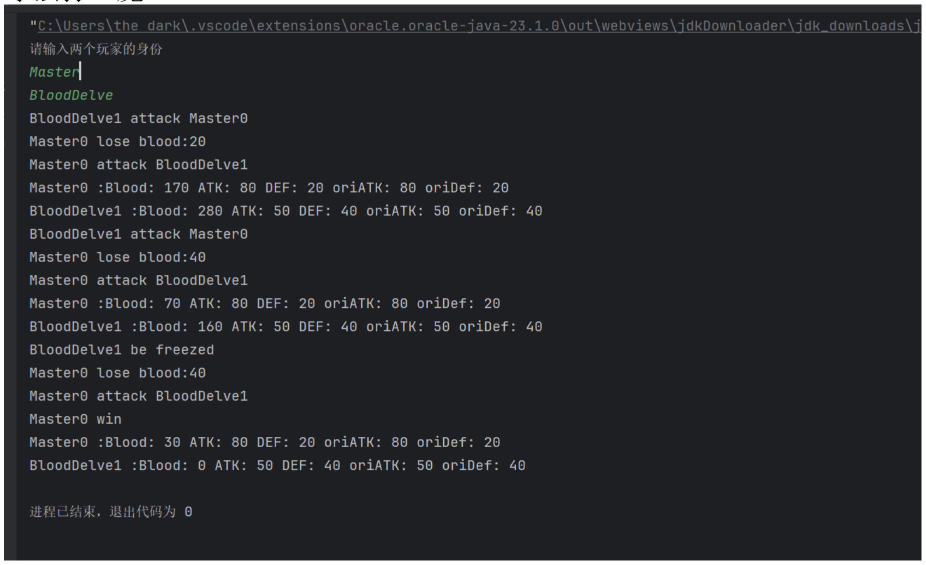
```
"C:\Users\the dark\.vscode\extensions\oracle.oracle-java-23.1.0\out\webviews\jdkDownloader\jdk_downloads
请输入两个玩家的身份
Warrior
Master
Master1 attack Warrior0
Warrior0 attack Master1
Warrior0 :Blood: 180 ATK: 120 DEF: 40 oriATK: 60 oriDef: 40
Master1 :Blood: 50 ATK: 80 DEF: 20 oriATK: 80 oriDef: 20
Master1 attack Warrior0
Warrior0 be freezed
Warrior0 :Blood: 60 ATK: 60 DEF: 40 oriATK: 60 oriDef: 40
Master1 :Blood: 50 ATK: 80 DEF: 20 oriATK: 80 oriDef: 20
Warrior0 attack Master1
Warrior0 win
Warrior0 :Blood: 60 ATK: 180 DEF: 40 oriATK: 60 oriDef: 40
Master1 :Blood: -270 ATK: 80 DEF: 20 oriATK: 80 oriDef: 20

进程已结束，退出代码为 0
```

血魔打刺客：



```
AssAssIn
BloodDelve0 attack AssAssIn1
AssAssIn1 lose blood:20
AssAssIn1 attack BloodDelve0
BloodDelve0 :Blood: 220 ATK: 50 DEF: 40 oriATK: 50 oriDef: 40
AssAssIn1 :Blood: 310 ATK: 120 DEF: 40 oriATK: 60 oriDef: 40
AssAssIn1 lose blood:20
AssAssIn1 attack BloodDelve0
BloodDelve0GetBlood
BloodDelve0 :Blood: 150 ATK: 50 DEF: 40 oriATK: 50 oriDef: 40
AssAssIn1 :Blood: 220 ATK: 120 DEF: 40 oriATK: 60 oriDef: 40
AssAssIn1 lose blood:40
AssAssIn1 attack BloodDelve0
BloodDelve0 attack AssAssIn1
BloodDelve0 :Blood: 110 ATK: 50 DEF: 40 oriATK: 50 oriDef: 40
AssAssIn1 :Blood: 160 ATK: 60 DEF: 40 oriATK: 60 oriDef: 40
BloodDelve0 attack AssAssIn1
AssAssIn1 lose blood:80
AssAssIn1 attack BloodDelve0
BloodDelve0 :Blood: 10 ATK: 50 DEF: 40 oriATK: 50 oriDef: 40
AssAssIn1 :Blood: 70 ATK: 120 DEF: 40 oriATK: 60 oriDef: 40
BloodDelve0 attack AssAssIn1
AssAssIn1 lose blood:100
AssAssIn1 attack BloodDelve0
AssAssIn1 win
BloodDelve0 :Blood: -10 ATK: 50 DEF: 40 oriATK: 50 oriDef: 40
AssAssIn1 :Blood: -50 ATK: 60 DEF: 40 oriATK: 60 oriDef: 40
```

冰法打血魔：



```
"C:\Users\the dark\.vscode\extensions\oracle.oracle-java-23.1.0\out\webviews\jdkDownloader\jdk_downloads\j
请输入两个玩家的身份
Master
BloodDelve
BloodDelve1 attack Master0
Master0 lose blood:20
Master0 attack BloodDelve1
Master0 :Blood: 170 ATK: 80 DEF: 20 oriATK: 80 oriDef: 20
BloodDelve1 :Blood: 280 ATK: 50 DEF: 40 oriATK: 50 oriDef: 40
BloodDelve1 attack Master0
Master0 lose blood:40
Master0 attack BloodDelve1
Master0 :Blood: 70 ATK: 80 DEF: 20 oriATK: 80 oriDef: 20
BloodDelve1 :Blood: 160 ATK: 50 DEF: 40 oriATK: 50 oriDef: 40
BloodDelve1 be freezed
Master0 lose blood:40
Master0 attack BloodDelve1
Master0 win
Master0 :Blood: 30 ATK: 80 DEF: 20 oriATK: 80 oriDef: 20
BloodDelve1 :Blood: 0 ATK: 50 DEF: 40 oriATK: 50 oriDef: 40

进程已结束，退出代码为 0
```

四. 心得与收获

使用继承封装多态三大特性让代码高效复用，只需要在子类中重写自己独特的部分即可，深刻感受到将所有共性都抽象到父类里这种设计思想带来的巨大便捷和效率提升

五. 源代码

```java
package GameSimulation;

import java.util.Random;

public abstract class Actor implements CanPlay {
    String name;
    int blood;
    int state;
    int ATK;
    int DEF;
    int character;
    int oriATK;
    int oriDef;
    enum Buff{NONE,LOSEBLOOD,FREEZE};
    Buff buff;
    int LoseBloodRound=0;
    static int RoolNum=0;
    Random rand = new Random();
    public Actor(String name, int blood, int state, int ATK,
        int DEF,int chara) {
        this.name = name+RoolNum;
        this.blood = blood;
        this.state = state;
        this.ATK = ATK;
        this.DEF = DEF;
        this.character = chara;
        oriATK=ATK;
        oriDef=DEF;
        buff=Buff.NONE;
        RoolNum++;
    }

    public Actor() {

    }

    @Override
    public void attack(Actor a) {
        state=1;
        int LoseBlood = ATK-a.DEF;
        if(rand.nextBoolean()) LoseBlood*=2;
        if(LoseBlood>0) a.blood-= LoseBlood;
    }

    @Override
    public String defense() {
        state=0;
        DEF*=2;
        return name+" defensed";
    }
    @Override
    public String toString()
    {
        return name+" :Blood: "+blood+" ATK: "+ATK+" DEF: "+DEF
            +" oriATK: "+oriATK+" oriDef: "+oriDef;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getBlood() {
        return blood;
    }

    public void setBlood(int blood) {
        this.blood = blood;
    }

    public int getState() {
        return state;
    }

    public void setState(String state) {
        if(state.equals("Attack")) this.state=1;
        else if(state.equals("Defense")) this.state=0;
    }
    public void setState(int state) {
        this.state = state;
    }

    public int getATK() {
        return ATK;
    }

    public void setATK(int ATK) {
        this.ATK = ATK;
    }

    public int getDEF() {
        return DEF;
    }
    abstract String attackway(Actor a);
    public void setDEF(int DEF) {
        this.DEF = DEF;
    }
    String action(Actor a) {
        if(ifdead()) return a.name+"␣dead";
        if(state==1) {
            return attackway(a);
        }
        else return defense();
    }
    public boolean getBuff()
    {
        switch(buff)
        {
            case NONE:break;
            case LOSEBLOOD:blood-=20*LoseBloodRound;
                System.out.println(name+"␣lose␣blood:"+20*
                    LoseBloodRound);break;
            case FREEZE:buff=Buff.NONE;return true;
        }
        return false;
    }
    void reset()
    {
        ATK=oriATK;
        DEF=oriDef;
    }
    boolean ifdead()
    {
        return blood<=0;
    }
}
```

```java
package GameSimulation;

public class AssAssIn extends Actor {
    AssAssIn()
    {
        super("AssAssIn",350,0,60,40,4);
    }
    @Override
    String attackway(Actor a) {
        if(getBuff()){return this.name+"␣be␣freezed";}
        attack(a);
        if(rand.nextBoolean()) {
            ATK*=2;
            attack(a);
        }
        return name+"␣attack␣"+a.name;
    }
}
```

```java
package GameSimulation;

public class BloodDelve extends Actor{
    boolean IfGetBlood=false;
    BloodDelve(){
        super("BloodDelve",400,0,50,40,3);
    }
    @Override
    String attackway(Actor a) {
        if(getBuff()) return this.name+"␣be␣freezed";
        if(this.character!=a.character) {a.buff=Buff.LOSEBLOOD;
            a.LoseBloodRound++;}
        attack(a);
        if(blood<200&&!IfGetBlood) {
            blood+=ATK;
            a.blood-=ATK;
            IfGetBlood=true;
            return name+"GetBlood";
        }
        return this.name+"␣attack␣"+a.name;
    }
}
```

```
package GameSimulation;

public interface CanPlay {
    abstract void attack(Actor a);
    abstract String defense();
}
```

```
package GameSimulation;

import java.io.*;
import java.net.*;

public class Client {
    public static void main(String[] args) {
        try (Socket socket = new Socket("localhost", 8080);
             BufferedReader in = new BufferedReader(new
                 InputStreamReader(socket.getInputStream()));
             PrintWriter out = new PrintWriter(socket.
                 getOutputStream(), true);
             BufferedReader userInput = new BufferedReader(new
                 InputStreamReader(System.in))) {

```

```java
            System.out.println("Connected␣to␣server.␣Enter␣messages:");

            // 启动线程监听服务器响应
            new Thread(() -> {
                try {
                    String response;
                    while ((response = in.readLine()) != null) {
                        System.out.println("Server␣response:␣" + response);

                        // 检测到服务端发送 "-1" 时关闭客户端
                        if ("-1".equals(response)) {
                            System.out.println("Server␣requested␣termination.␣Closing␣client.");
                            socket.close(); // 关闭 Socket 触发主线程退出

                            throw new IOException();
                        }
                    }
                } catch (IOException e) {
                    System.out.println("Disconnected␣from␣server.");
                }
            }).start();

            // 发送用户输入的消息
            String input;
            while ((input = userInput.readLine()) != null) {
                out.println(input);
            }

        } catch (IOException e) {
            // 捕获到 Socket 关闭的异常后正常退出
            System.out.println("Client␣terminated.");
        }
```

```java
    }
}
```

```java
package GameSimulation;

import java.util.Random;

public abstract class Game {
    static Random RandState = new Random();
    abstract void play();
}
```

```java
package GameSimulation;

import java.util.Scanner;

public class Game1 extends Game{
    static Master master;
    static Warrior warrior;
    public void play() {
        master=new Master();
        warrior=new Warrior();
        Scanner sc=new Scanner(System.in);
        int player=0;
        int whodead=0;
        while(master.blood>0 && warrior.blood>0){
            String Character=sc.next();
            String State=sc.next();
            if(Character.equals("Master")){
                player=1;
                master.setState(State);
                warrior.setState(RandState.nextInt()%2);
            }
            else if(Character.equals("Warrior")){
                player=2;
                warrior.setState(State);
                master.setState(RandState.nextInt()%2);
            }
            int whonext=RandState.nextInt()%2;
            if(whonext==1){
                System.out.println(master.action(warrior));
                if(warrior.blood<=0) {whodead=2;break;}
                System.out.println(warrior.action(master));
                if(master.blood<=0) {whodead=1;break;}
            }
            else
            {
                System.out.println(warrior.action(master));
                if(master.blood<=0) {whodead=1;break;}
                System.out.println(master.action(warrior));
                if(warrior.blood<=0) {whodead=2;break;}
            }
            System.out.println(master);
            System.out.println(warrior);
            warrior.reset();
            master.reset();
        }
        if(whodead==player) System.out.println("Drew!");
        else System.out.println("You are the winner!");
        System.out.println(warrior);
        System.out.println(master);
    }
}
```

```
package GameSimulation;

public class Game2 extends Game{
    Actor player1;
    Actor player2;
    public Game2(String player1,String player2){
        switch(player1){
            case"AssAssIn": this.player1=new AssAssIn();break;
            case"Master":this.player1=new Master();break;
            case"Warrior":this.player1=new Warrior();break;
            case"BloodDelve":this.player1=new BloodDelve();break;
        }
        switch(player2){
            case"AssAssIn": this.player2=new AssAssIn();break;
            case"Master":this.player2=new Master();break;
            case"Warrior":this.player2=new Warrior();break;
            case"BloodDelve":this.player2=new BloodDelve();
                break;
        }
    }
    public void play()
    {
        Play("Attack","Attack");
    }
    public void RandPlay()
    {
        int WhoPiror= RandState.nextInt()%2;
        player1.setState(RandState.nextInt()%2);
        player2.setState(RandState.nextInt()%2);
        if(WhoPiror==1)
        {
            System.out.println(player1.action(player2));
            if(player2.ifdead()) return;
            System.out.println(player2.action(player1));
            if(player1.ifdead()) return;
        }
        else {
            System.out.println(player2.action(player1));
            if(player1.ifdead()) return;
            System.out.println(player1.action(player2));
            if(player2.ifdead()) return;
        }
        System.out.println(player1);
        System.out.println(player2);
        player1.reset();
        player2.reset();
    }
    public void Play(String play1,String play2)
    {
        int WhoPiror= RandState.nextInt()%2;
        player1.setState(play1);
        player2.setState(play2);
```

```java
        if(player1.getState()==0) WhoPiror=1;
        if(player2.getState()==0) WhoPiror=0;
        if(WhoPiror==1)
        {
            System.out.println(player1.action(player2));
            if(player2.ifdead()) return;
            System.out.println(player2.action(player1));
            if(player1.ifdead()) return;
        }
        else {
            System.out.println(player2.action(player1));
            if(player1.ifdead()) return;
            System.out.println(player1.action(player2));
            if(player2.ifdead()) return;
        }
        System.out.println(player1);
        System.out.println(player2);
        player1.reset();
        player2.reset();
    }
}
```

```java
package GameSimulation;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {

       /* Game1 game1 = new Game1();
        game1.play();*/
        Scanner input = new Scanner(System.in);
        System.out.println("请输入两个玩家的身份");
        String player1 = input.next();
        String player2 = input.next();
        System.out.println("请输入玩法");
        String playway= input.next();
        Game2 game = new Game2(player1, player2);
        while(!game.player1.ifdead()&&!game.player2.ifdead()) {

            if(playway.equals("Random")) game.RandPlay();
            if(playway.equals("test")) game.play();
            if(playway.equals("Play")) {
                player1= input.next();
                player2= input.next();
                game.Play(player1,player2);
            }
        }
        if(game.player1.ifdead())
        {
            System.out.println(game.player2.name+"␣win");
        }
        else
        {
            System.out.println(game.player1.name+"␣win");
        }
        System.out.println(game.player1);
        System.out.println(game.player2);
    }
}
```

```java
package GameSimulation;

public class Master extends Actor{
    public int FreezeValue=0;
    public Master(){
        super("Master",250,0,80,20,1);
    }
    @Override
    String attackway(Actor a){
        if(getBuff()){return this.name+"␣be␣freezed";}
        if(this.character==a.character){
            attack(a);
        }
        else {
            attack(a);
            attack(a);
        }
        FreezeValue++;
        if(FreezeValue==2){FreezeValue=0;a.buff=Buff.FREEZE;}
        return this.name+"␣attack␣"+a.name;
    }

}
```

```java
package GameSimulation;
import java.io.*;
import java.net.*;
import java.util.concurrent.CyclicBarrier;

public class TwoClientServer {

    // 共享数据类，保存两个客户端的消息
    static class SharedData {
        static boolean ifFirst = true;
        private String client1Message;
        private String client2Message;
        private final CyclicBarrier barrier;
        private static Game2 game;
        public SharedData() {
            // 当两个线程到达屏障时，触发处理并回复
            this.barrier = new CyclicBarrier(2, this::
                processAndRespond);
        }

        public synchronized void setClient1Message(String msg)
            {
            this.client1Message = msg;
        }

        public synchronized void setClient2Message(String msg)
            {
            this.client2Message = msg;
        }

        // 处理消息并回复客户端
        private void processAndRespond() {
            String processed = "Processed:␣" + client1Message + "␣&␣" + client2Message;
            // 假设每个客户端处理程序持有自己的输出流
            if(ifFirst) {game=new Game2(client1Message,client2Message);ifFirst=false;}
            else{
                game.Play(client1Message,client2Message);
            }
            if(game.player1.ifdead()||game.player2.ifdead()) {processed="-1";
                System.out.println(game.player1);
                System.out.println(game.player2);
            }
            ClientHandler.client1Out.println(processed);
            ClientHandler.client2Out.println(processed);
            System.out.println("Sent␣response:␣" + processed);
        }
    }

    // 客户端处理线程
    static class ClientHandler implements Runnable {
        private Socket socket;
        private BufferedReader in;
        public static PrintWriter client1Out;
        public static PrintWriter client2Out;
        private final SharedData sharedData;
        private final int clientId;

        public ClientHandler(Socket socket, SharedData sharedData, int clientId) {
            this.socket = socket;
            this.sharedData = sharedData;
            this.clientId = clientId;
            try {
                this.in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
                PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
                // 根据客户端ID保存输出流
                if (clientId == 1) {
                    client1Out = out;
                } else {
                    client2Out = out;
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        @Override
        public void run() {
            try {
                while (true) {
                    String message = in.readLine();
                    if (message == null) break; // 客户端断开连
                        接

                    // 存储消息到共享数据
                    if (clientId == 1) {
                        sharedData.setClient1Message(message);
                    } else {
                        sharedData.setClient2Message(message);
                    }

                    System.out.println("Received␣from␣Client␣"
                        + clientId + ":␣" + message);

                    // 等待另一个客户端的消息
                    sharedData.barrier.await();
                }
            } catch (Exception e) {
                e.printStackTrace();
            } finally {
                try {
                    in.close();
                    socket.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    public static void main(String[] args) {
        SharedData sharedData = new SharedData();
        try (ServerSocket serverSocket = new ServerSocket(8080)
            ) {
            System.out.println("Server␣started.␣Waiting␣for␣
                clients...");

            // 等待第一个客户端连接
            Socket client1 = serverSocket.accept();
            System.out.println("Client␣1␣connected.");
            new Thread(new ClientHandler(client1, sharedData,
                1)).start();

            // 等待第二个客户端连接
            Socket client2 = serverSocket.accept();
            System.out.println("Client␣2␣connected.");
            new Thread(new ClientHandler(client2, sharedData,
                2)).start();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```java
package GameSimulation;

public class Warrior extends Actor{
    public Warrior(){
        super("Warrior",300,0,60,10,2);
    }
    @Override
    String attackway(Actor a)
    {
        if(getBuff()) return this.name+" be freezed";
        ATK=oriATK+(300-blood)/2;
        if(this.character==a.character)
        {
            attack(a);
            attack(a);
        }
        else
        {
            attack(a);
        }
        return name+" attack "+a.name;
    }
}
```