目录

Hero class

```
package three_kingdoms_kill;


import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Set;


public class Heroes {

	public String name;

	public int ordinary_hp;

	public int hp;


	public Heroes() {

		this.hp = 0;

	}


//	public int skills(Players thisplay,Players thatplay, HashMap cardMap,HashMap
paiduiMap,ArrayList redCamp,ArrayList blueCamp) {

//		return 0;

//	}

//	public int skills(Players thisplay,Players play[], HashMap cardMap,HashMap
paiduiMap,ArrayList redCamp,ArrayList blueCamp) {

//		return 0;

//	}


	public int skills(Players thisplay, Players thatplay, HashMap cardMap) {

		return 0;

	}
```

```
    public int skills(Players thisplay, Players play[], HashMap paiduiMap) {
        return 0;
    }

    public int skills(Players thisplay, Players play[], HashMap paiduiMap, ArrayList
redCamp, ArrayList blueCamp) {
        return 0;
    }

    public int skills(Players thisplay, HashMap paiduiMap) {
        return 0;
    }

    public int skills(Players thisplay, HashMap cardMap, HashMap paiduiMap) {
        return 0;
    }

    public int skills(Players thisplay, HashMap cardMap, HashMap paiduiMap, ArrayList
redCamp, ArrayList blueCamp) {
        return 0;
    }
}

class Liubei extends Heroes {
    public Liubei() {
        this.name = "刘备";
        this.ordinary_hp = this.hp = 4;
    }
```

```
//  仁德，出牌阶段，可以将一张手牌以任意分配方式交给其他角色并回复 1 点体力 ，分出的牌，对方无法拒绝。
    public int skills(Players thisplay, Players play[], HashMap cardMap) {

        if (this.hp < this.ordinary_hp && thisplay.cardNum > 0) {
            System.out.println(this.name + "的【仁德】技能开始-------->");

            Players p = thisplay.select_opponent(thisplay, play);

            System.out.println(this.name + "的血量" + this.hp + "少于" + this.ordinary_hp + "并且有" + thisplay.cardNum + "张牌");
            Set entrySet = thisplay.handCard.entrySet();
            Iterator it = entrySet.iterator();

            while (it.hasNext()) {
                HashMap.Entry entry = (HashMap.Entry) (it.next());
                Object key = entry.getKey();
                System.out.println("交换前的牌" + thisplay.handCard + p.handCard + "血量" + this.hp);
                thisplay.handCard.remove(key);
                thisplay.cardNum--;
                p.handCard.put(key, cardMap.get(key));
                p.cardNum++;
                this.hp++;
                System.out.println("交换后的牌" + thisplay.handCard + p.handCard + "血量" + this.hp);
                break;
            }
```

```
            System.out.println(this.name + "的【仁德】技能结束-------->");
        } else
            System.out.println(this.name + "未使用【仁德】技能");

        return 0;
    }
}

class Zhugeliang extends Heroes {
    public Zhugeliang() {
        this.name = "诸葛亮";
        this.ordinary_hp = this.hp = 3;
    }

//    “空城”虽然恢复血量，但诸葛亮一般不需要为了空城而刻意弃光手牌。
    public int skills(Players thisplay, HashMap paiduiMap) {
        if (thisplay.hero.hp < 0 || thisplay.hero.hp < thisplay.hero.ordinary_hp) {
            System.out.println(this.name + "的【空城】技能开始-------->");
            Set entrySet = thisplay.handCard.entrySet();
            Iterator it = entrySet.iterator();
//          System.out.println("空城之前"+thisplay.handCard);
//          for (int i = 0;i < thisplay.cardNum;i++) {//不能循环删除，只能循环删除一次
            while (it.hasNext()) {
                HashMap.Entry entry = (HashMap.Entry) (it.next());
                Object key = entry.getKey();
                Object value = entry.getValue();
```

```
                    thisplay.handCard.remove(key);
                    paiduiMap.put(key, 0);
                    thisplay.cardNum--;
                    break;
                }
//              }
                thisplay.hero.hp = thisplay.hero.ordinary_hp;
//              System.out.println("空城之后"+thisplay.handCard);
                System.out.println(this.name + "丢弃了所有手牌，【空城】技能结束
-------->");
            } else
                System.out.println(this.name + "未使用【空城】技能");

            return 0;
        }
}

class Guanyu extends Heroes {
    public Guanyu() {
        this.name = "关羽";
        this.ordinary_hp = this.hp = 4;
    }

    // "武圣"可将一切牌转化为杀
    public int skills(Players thisplay, Players play[], HashMap paiduiMap, ArrayList
redCamp, ArrayList blueCamp) {
        System.out.println(this.name + "的【武圣】技能开始-------->");
        Set entrySet = thisplay.handCard.entrySet();
```

```
        Iterator it = entrySet.iterator();
//      for (int i = 0;i < thisplay.cardNum;i++) {//不能循环删除，只能循环删除一次
        while (it.hasNext()) {

            HashMap.Entry entry = (HashMap.Entry) (it.next());
            Object key = entry.getKey();
            if (thisplay.handCard.size() != 0) {
                System.out.println(thisplay.hero.name + "有【杀】");
                thisplay.attack(thisplay, thisplay.select_opponent(thisplay, play),
paiduiMap, redCamp, blueCamp);

                thisplay.handCard.remove(key);
                paiduiMap.put(key, 0);
                thisplay.cardNum--;
                break;
            } else
                System.out.println(thisplay.hero.name + "没有【杀】，不能使用【武
圣】技能！");

        }
//      }
        System.out.println(this.name + "的【武圣】技能结束-------->");

        return 0;
    }
}

class Diaochan extends Heroes {
```

```
    public Diaochan() {
        this.name = "貂蝉";
        this.ordinary_hp = this.hp = 3;
    }

    //  “离间”是一个异常强大的技能，1 张牌必然能换敌方 1 点血量
    public int skills(Players thisplay, Players play[], HashMap paiduiMap, ArrayList
redCamp, ArrayList blueCamp) {
        System.out.println(this.name + "的【离间】技能开始-------->");

        Set entrySet = thisplay.handCard.entrySet();
        Iterator it = entrySet.iterator();
//      for (int i = 0;i < thisplay.cardNum;i++) {//不能循环删除，只能循环删除一次
        while (it.hasNext()) {

            HashMap.Entry entry = (HashMap.Entry) (it.next());
            Object key = entry.getKey();
            System.out.println(thisplay.hero.name + "有手牌【" + key + "】");
            //
            //System.out.println(redCamp.size());
            //System.out.println(blueCamp.size());// 2021 年 1 月 9 日 14:59:21
            //
            if ((int) redCamp.size() == 0 || (int) blueCamp.size() == 0)
                break;// 出牌之前先判断结束游戏没有
            Players p = thisplay.select_opponent(thisplay, play);

            System.out.println("并选择了对象" + p.hero.name + "," + p.hero.name
+ "被扣一滴血");
```

```
                p.hero.hp--;

                if (p.die()) {
                    if (p.camps == "红方") {
                        redCamp.remove(0);
                    } else
                        blueCamp.remove(0);
                    // ----------------------------------------------------
                } // 2021 年 1 月 9 日 15:01:53

                thisplay.handCard.remove(key);
                paiduiMap.put(key, 0);
                thisplay.cardNum--;
                break;
            }
//          }
            System.out.println(this.name + "的【离间】技能结束-------->");
            return 0;
        }
    }

    class Zhangfei extends Heroes {
        public Zhangfei() {
            this.name = "张飞";
            this.ordinary_hp = this.hp = 4;
        }

        // 咆哮：锁定技，你使用【杀】无次数限制。
```

```
    public int skills(Players thisplay, Players play[], HashMap paiduiMap, ArrayList
redCamp, ArrayList blueCamp) {

        System.out.println(this.name + "的【咆哮】技能开始-------->");

        Set entrySet = thisplay.handCard.entrySet();

        Iterator it = entrySet.iterator();


        while (it.hasNext()) {

            HashMap.Entry entry = (HashMap.Entry) (it.next());

            Object key = entry.getKey();

            Object value = entry.getValue();


            if ((String) value == "杀") {

                //

                System.out.println(thisplay.name + "有杀");

                thisplay.attack(thisplay, thisplay.select_opponent(thisplay, play),
paiduiMap, redCamp, blueCamp);

                thisplay.handCard.remove(key);

                paiduiMap.put(key, 0);

                thisplay.cardNum--;

//              defense(select_opponent(play));

                break;

            } else

                System.out.println(this.name + "不满足【咆哮】技能");

        }

        System.out.println(this.name + "的【咆哮】技能结束-------->");

        return 0;

    }

}
```

```
class Lvbu extends Heroes {
    public Lvbu() {
        this.name = "吕布";
        this.ordinary_hp = this.hp = 4;
    }

    // 无双：锁定技，使用的【杀】需两张【闪】才能抵消；
    public int skills(Players thisplay, Players play[], HashMap paiduiMap, ArrayList redCamp, ArrayList blueCamp) {
        System.out.println(this.name + "的【无双】技能开始-------->");
        Set entrySet = thisplay.handCard.entrySet();
        Iterator it = entrySet.iterator();

        while (it.hasNext()) {
            HashMap.Entry entry = (HashMap.Entry) (it.next());
            Object key = entry.getKey();
            Object value = entry.getValue();

            if ((String) value == "杀") {
                //
                System.out.println(thisplay.hero.name + "有【杀】");
                Players p = thisplay.select_opponent(thisplay, play);
                thisplay.attack(thisplay, p, paiduiMap, redCamp, blueCamp);
                thisplay.attack(thisplay, p, paiduiMap, redCamp, blueCamp);// 攻击两次
                thisplay.handCard.remove(key);
                paiduiMap.put(key, 0);
                thisplay.cardNum--;
```

```
//                  defense(select_opponent(play));
                    break;
                } else
                    System.out.println(thisplay.hero.name + "没有【杀】，不能使用【无双】技能！");
            }
            System.out.println(this.name + "的【无双】技能结束-------->");
            return 0;
        }
}

class Huatuo extends Heroes {
    public Huatuo() {
        this.name = "华佗";
        this.ordinary_hp = this.hp = 3;
    }

    //  “急救”能让华佗的所有红牌当桃用
    public int skills(Players thisplay, HashMap paiduiMap) {
        System.out.println(this.name + "的【急救】技能开始-------->");
        Set entrySet = thisplay.handCard.entrySet();
        Iterator it = entrySet.iterator();
        int ans = 0;
//      for (int i = 0;i < thisplay.cardNum;i++) {//不能循环删除，只能循环删除一次
        while (it.hasNext()) {

            HashMap.Entry entry = (HashMap.Entry) (it.next());
            Object key = entry.getKey();
```

```
                System.out.println(thisplay.name + "有桃，急救成功");
                ans = 1;
                thisplay.handCard.remove(key);
                paiduiMap.put(key, 0);
                thisplay.cardNum--;
                break;
            }
//          }

            System.out.println(this.name + "的【急救】技能结束-------->");
            return ans;// 返回 1 代表有牌替代桃使用
        }
}

class Huanggai extends Heroes {
    public Huanggai() {
        this.name = "黄盖";
        this.ordinary_hp = this.hp = 4;
    }

    // “苦肉”可以看做是主动发动的“遗计”，1 血换 2 牌，不产生收益。
    public int skills(Players thisplay, HashMap cardMap, HashMap paiduiMap, ArrayList redCamp, ArrayList blueCamp) {
        if (this.hp > 1) {
            System.out.println(this.name + "的【苦肉】技能开始-------->");
            System.out.println(this.name + "使用了【苦肉】技能，损失一滴血，抽取两张牌！");
            thisplay.selectCard(cardMap, 2, paiduiMap);
```

```
            this.hp--;

            if (thisplay.die()) {
                if (thisplay.camps == "红方") {
                    redCamp.remove(0);
                } else
                    blueCamp.remove(0);
                // ----------------------------------------------------
            } // 2021 年 1 月 9 日 15:01:53
            System.out.println(this.name + "的【苦肉】技能结束-------->");
        }
        return 0;
    }
}
```

Players class

```
package three_kingdoms_kill;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Random;
import java.util.Set;

public class Players {
    // 死亡
    boolean isdie = false;
//  （属性）名字
    String name;
//  序列号
    int No;
//  （属性）阵营
    String camps;
//  （属性）英雄（对象）
    Heroes hero;
//  （属性）手牌（对象）
    // 记录手牌
    HashMap handCard = new HashMap();
//  （属性）玩家持牌数量
    int cardNum = 0;
```

```
    // 选阵营
    void selectCamp(HashMap campMap, ArrayList redCamp, ArrayList blueCamp) {
        int k = new Random().nextInt(2);
        if (k == 0 && redCamp.size() <= 2) {
            redCamp.add(this.camps = (String) campMap.get(k));
        } else if (k == 1 && blueCamp.size() <= 2) {
            blueCamp.add(this.camps = (String) campMap.get(k));
        } else {
            this.selectCamp(campMap, redCamp, blueCamp);
        }
    }

    // 抽英雄
    void selectHero(HashMap heroMap, HashSet heroSet) {
        int k = 0;
        int lenth = heroSet.size();
        while (heroSet.size() == lenth) {
            k = new Random().nextInt(8);
            heroSet.add(heroMap.get(k));
            hero = (Heroes) heroMap.get(k);
        }
        // return (String)heroMap.get(k);
    }

//  得到卡牌 2021 年 1 月 5 日
    void selectCard(HashMap cardMap, int num, HashMap paiduiMap) {
        for (int i = 0; i < num; i++) {
```

```
                int lenth = handCard.size();
                while (handCard.size() <= lenth) {
                    int k = new Random().nextInt(66) + 1;// (1-66)
                    if ((int) paiduiMap.get(k) == 0) {
                        handCard.put(k, cardMap.get(k));
                        paiduiMap.put(k, 1);
                        cardNum++;
                    }
                }
            }
//          System.out.println(handCard);//测试
        }

        // 【出牌】
        void chupai(Players thisplay, Players play[], HashMap paiduiMap, ArrayList
redCamp, ArrayList blueCamp) {

            if ((int) redCamp.size() == 0 || (int) blueCamp.size() == 0) {
                System.out.println("游戏已结束！");
                return;
            } // 2021 年 1 月 9 日 14:15:40

            if (!thisplay.isdie) {
                // 遍历键值对中的值（杀闪桃）

                System.out.print(thisplay.hero.name + "有手牌");//

                Set entrySet2 = thisplay.handCard.entrySet();
```

```
            Iterator it2 = entrySet2.iterator();

            while (it2.hasNext()) {
                HashMap.Entry entry2 = (HashMap.Entry) (it2.next());
                Object value2 = entry2.getValue();
                System.out.print("  【" + value2 + "】  ");
            }
            System.out.println();

            Set entrySet = thisplay.handCard.entrySet();
            Iterator it = entrySet.iterator();

            while (it.hasNext()) {
                HashMap.Entry entry = (HashMap.Entry) (it.next());
                Object key = entry.getKey();
                Object value = entry.getValue();
//                  System.out.println("这次是"+value);
//                  System.out.println(key+"+"+value);
                if ((String) value == "杀") {
                    //
                    System.out.println(thisplay.hero.name + "有【杀】");

                    if ((int) redCamp.size() == 0 || (int) blueCamp.size() == 0)
                        break;// 出牌之前先判断结束游戏没有

                    Players p = select_opponent(thisplay, play);
                    System.out.println(thisplay.hero.name + "对" + p.hero.name +
"使用了【杀】");
```

```
                    attack(thisplay, p, paiduiMap, redCamp, blueCamp);
                    thisplay.handCard.remove(key);
                    paiduiMap.put(key, 0);
                    thisplay.cardNum--;
//                  defense(select_opponent(play));
                    break;
                }
            }

        } else {
            System.out.println(thisplay.hero.name + "已死亡！无法【出牌】");
        }

//      System.out.println(thisplay.name+"的"+thisplay.hero.name+"，手牌
"+thisplay.handCard+"，有"+thisplay.cardNum+"张牌");
        System.out.println();
    }

    // 【选对手】选择对手阵营英雄（主动）
    Players select_opponent(Players thisplay, Players play[]) {
//      System.out.println("----------【选对手】----------");//
        String camp = thisplay.camps;
        System.out.println(thisplay.hero.name + "的阵营是" + camp);
        int k = new Random().nextInt(6);
//      int i = 0;
        while (play[k].camps == camp || play[k].isdie) {
            k = new Random().nextInt(6);
//          if (i > 6) {
```

```
//                  System.out.println("超时");
//                  k = thisplay.No - 1;
//                  break;
//              }
//              i++;
            }
            System.out.println(thisplay.hero.name + "选择了" + play[k].camps + "对象" + play[k].hero.name);
//          System.out.println(play[k].hero.name+"的【手牌】"+play[k].handCard);
//          System.out.println("----------【选对手结束】-------------------");//
            return play[k];
        }

        // 【攻击】攻击（主动）
        void attack(Players thisplay, Players thatplay, HashMap paiduiMap, ArrayList redCamp, ArrayList blueCamp) {
//          System.out.println("----------------【攻击】-------------------");//
            if (!(boolean) thatplay.defense(thatplay, paiduiMap, redCamp, blueCamp)) {

                if (thisplay.die()) {
                    System.out.println(thisplay.hero.name + "已经死亡");
                    if (thisplay.camps == "红方") {
                        redCamp.remove(0);
                    } else
                        blueCamp.remove(0);

                    // ----------------------------------------------------
                } // 2021 年 1 月 8 日 21:23:17 添加
```

```
            if (thatplay.isdie) {
//                  if (thatplay.camps=="红方") {
//                      redCamp.remove(0);
//                  }
//                  else blueCamp.remove(0);
                    // ----------------------------------------------------2021 年 1 月 9 日 15:01:02
                    System.out.println(thatplay.hero.name + "已死亡，无法被攻击");
            } else {
                    System.out.println(thatplay.hero.name + "对" + thisplay.hero.name + "防御失败");
                    thatplay.hero.hp--;
                    if (thatplay.die()) {
                        if (thatplay.camps == "红方") {
                            redCamp.remove(0);
                        } else
                            blueCamp.remove(0);
                        // ----------------------------------------------------
                    } // 2021 年 1 月 9 日 15:01:53

                    System.out.println(thatplay.hero.name + "的 HP:" + thatplay.hero.hp);

            }

        } else {
```

```
                System.out.println(thatplay.hero.name + "对" + thisplay.hero.name + "
防御成功");
//              System.out.println(thatplay.hero.name+"的血量"+thatplay.hero.hp);
            }

//          System.out.println("----------------【攻击结束】-------------------");//
        }

        // 【防御】防御（被动）
        boolean defense(Players thisplay, HashMap paiduiMap, ArrayList redCamp,
ArrayList blueCamp) {
//          System.out.println("------------------------【防御】-------------------");
            boolean result = false;
            if (!thisplay.isdie) {
//              System.out.println(thisplay.hero.name+"的血量："+thisplay.hero.hp);
                Set entrySet = thisplay.handCard.entrySet();
                Iterator it = entrySet.iterator();

                while (it.hasNext() && result == false) {

                    HashMap.Entry entry = (HashMap.Entry) (it.next());
                    Object key = entry.getKey();
                    Object value = entry.getValue();
//                  System.out.println(thisplay.name+"【手牌】"+thisplay.handCard);
                    if ((String) value == "闪") {

                        System.out.println(thisplay.hero.name + "使用了【闪】，防御成
功");
```

```
                    thisplay.handCard.remove(key);
                    paiduiMap.put(key, 0);
                    thisplay.cardNum--;
//                  System.out.println(thisplay.hero.name+"防御后手牌"+thisplay.handCard);
                    result = true;
                    break;
                } else if ((String) value == "桃" && thisplay.hero.hp <= 1) {// 最后一血有桃就当做闪
                    System.out.println(thisplay.hero.name + "没有【闪】，但使用了【桃】，成功续命");
                    thisplay.handCard.remove(key);
                    paiduiMap.put(key, 0);
                    thisplay.cardNum--;
//                  System.out.println(thisplay.hero.name+"防御后手牌"+thisplay.handCard);
                    result = true;
                    break;
                } else {
//                  System.out.println(thisplay.hero.name+"没有【闪】，防御失败");
                    result = false;

                }

            }
//          System.out.println(thisplay.hero.name+"的血量："+thisplay.hero.hp);
//          if (thisplay.die()) {
//              if (thisplay.camps=="红方") {
```

```
//                      redCamp.remove(0);
//                  }
//                  else blueCamp.remove(1);
//                  //-----------------------------------------------------
//              }
        } else
            System.out.println(thisplay.hero.name + "已死亡，无法防御！");
//      System.out.println(thisplay.hero.name+"无手牌，无法防御！");
//      System.out.println("------------------------【防御结束】-------------------");
        return result;
    }

    // 判断是否死亡
    boolean die() {
        if (this.hero.hp <= 0 && !this.isdie) {
            this.isdie = true;
            System.out.println(this.hero.name + "死亡！");
        }

        return this.isdie;
    }

    // 手牌是否大于血量
    boolean selectCardMore() {
        boolean b = false;
        if (this.cardNum > this.hero.hp) {
            b = true;
        }
```

```
        return b;
    }

    // 清理手牌
    void clearCard(HashMap cardMap, HashMap paiduiMap, int num) {

//          System.out.println(this.hero.name+"的血量"+this.hero.hp+"，清理前的手牌"+this.handCard+"有："+this.cardNum+"张牌");
        // 测试 2
        int i = 1;
        Set entrySet = this.handCard.entrySet();
        Iterator it = entrySet.iterator();

        while (it.hasNext() && i <= num) {
            HashMap.Entry entry = (HashMap.Entry) (it.next());
            Object key = entry.getKey();
            this.handCard.remove(key);
            paiduiMap.put(key, 0);
            this.cardNum--;
            i++;
        }
//          System.out.print(this.hero.name+"的血HP"+this.hero.hp+",清理后的手牌");
//          //测试 3
//
//          Set entrySet2 = this.handCard.entrySet();
//          Iterator it2 = entrySet2.iterator();
//
//          while(it2.hasNext()) {
```

```
//              HashMap.Entry entry2 = (HashMap.Entry)(it2.next());
//              Object value2 = entry2.getValue();
//              System.out.print(" 【"+value2+"】 ");
//          }
//          System.out.println();

    }

    // 血量限制手牌
    void clearHpCard(HashMap cardMap, HashMap paiduiMap) {
//          System.out.println(this.hero.name+"的血量"+this.hero.hp+"，清理前的手牌"+this.handCard+"有："+this.cardNum+"张牌");
            // 测试 4
            int num = this.cardNum - this.hero.hp;
            if (num > 0) {
                int i = 1;
                Set entrySet = this.handCard.entrySet();
                Iterator it = entrySet.iterator();

                while (it.hasNext() && i <= num) {
                    HashMap.Entry entry = (HashMap.Entry) (it.next());
                    Object key = entry.getKey();
                    this.handCard.remove(key);
                    paiduiMap.put(key, 0);
                    this.cardNum--;
                    i++;
                }
            }
```

```
        System.out.print(this.hero.name + "的 HP:" + this.hero.hp + "，清理后的手牌");// 测试 5

        Set entrySet2 = this.handCard.entrySet();
        Iterator it2 = entrySet2.iterator();

        while (it2.hasNext()) {
            HashMap.Entry entry2 = (HashMap.Entry) (it2.next());
            Object value2 = entry2.getValue();
            System.out.print("【" + value2 + "】 ");
        }
        System.out.println();

    }
//  void remove() {}

}
```

Three_Kingdoms_Kill class

```
package three_kingdoms_kill;

import java.util.ArrayList;
import java.util.Collection;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Random;
import java.util.Set;

public class Three_Kingdoms_Kill {

    public static void main(String[] args) {

        System.out.println("--------------------欢迎来到三国杀--------------------");
        System.out.println();
        System.out.println("正在加载阵营……");
        // 阵营类 2020 年 12 月 22 日 10:43:58
        HashMap campMap = new HashMap();
        campMap.put(0, "红方");
        campMap.put(1, "蓝方");
        // 记录选中的阵营
        ArrayList redCamp = new ArrayList();
        ArrayList blueCamp = new ArrayList();
        System.out.println("【阵营】: " + campMap.get(0) + "、" + campMap.get(1));
```

```
System.out.println();

System.out.println("正在加载英雄……");
// 英雄类 2020 年 12 月 21 日 20:31:11
HashMap heroMap = new HashMap();
Liubei liub = new Liubei();
Zhugeliang zhugl = new Zhugeliang();
Guanyu guany = new Guanyu();
Diaochan diaoc = new Diaochan();
Zhangfei zhangf = new Zhangfei();
Lvbu lvb = new Lvbu();
Huatuo huat = new Huatuo();
Huanggai huangg = new Huanggai();

heroMap.put(0, liub);
heroMap.put(1, zhugl);
heroMap.put(2, guany);
heroMap.put(3, diaoc);
heroMap.put(4, zhangf);
heroMap.put(5, lvb);
heroMap.put(6, huat);
heroMap.put(7, huangg);
// 记录选中的英雄
HashSet heroSet = new HashSet();
System.out.println("【英雄】: " + liub.name + "、" + zhugl.name + "、" + guany.name + "、" + diaoc.name + "、"
                + zhangf.name + "、" + lvb.name + "、" + huat.name + "、" + huangg.name);
```

```
        System.out.println();

        System.out.println("正在加载手牌……");
        // 加载手牌类
//      三国杀中各种手牌数量,共 66:
        HashMap cardMap = new HashMap();
//      1、杀共有 30 张(1-30)
        for (int i = 1; i <= 30; i++) {
            cardMap.put(i, "杀");
        }
//      2、闪共有 24 张(31-54)
        for (int i = 31; i <= 54; i++) {
            cardMap.put(i, "闪");
        }
//      3、桃共有 12 张(55-66)
        for (int i = 55; i <= 66; i++) {
            cardMap.put(i, "桃");
        }
        // 牌堆
        HashMap paiduiMap = new HashMap();
        for (int i = 1; i <= 66; i++) {
            paiduiMap.put(i, 0);
        }
        System.out.println("【手牌】: " + cardMap.get(1) + "、" + cardMap.get(31)
+ "、" + cardMap.get(55));
        System.out.println();
```

```
System.out.println("正在生成玩家……");
System.out.print("【玩家】： ");
// 玩家初始化
Players play[] = new Players[6];
for (int i = 0; i < 6; i++) {
    play[i] = new Players();
    play[i].No = i + 1;
    play[i].name = "玩家" + play[i].No;
    play[i].selectCamp(campMap, redCamp, blueCamp);
    play[i].selectHero(heroMap, heroSet);
    play[i].selectCard(cardMap, 4, paiduiMap);// 开局每人四张手牌
    if (i == 5) {
        System.out.println(play[i].name);
    } else
        System.out.print(play[i].name + "、");

}
System.out.println();

// 游戏开始
--------------------------------------------------------------------------------
System.out.println("--------------------游戏开始--------------------");
System.out.println();
// 顺时针顺序出牌
int i = 1;// 回合数
int t = 0;// 循环数（测试）
while ((int) redCamp.size() != 0 && (int) blueCamp.size() != 0) {
```

```
            // (int)redCamp.size()!  =0&&(int)blueCamp.size()!=0
            // t < 100
            //  遍历 6 个玩家
            for (int no = 0; no < 6; no++) {
//                  System.out.println(redCamp);
//                  System.out.println(blueCamp);//测试 1

                    if ((int) redCamp.size() == 0 || (int) blueCamp.size() == 0)
                        break;
                    if (play[no].isdie)
                        continue;
                    System.out.println("--------------------第" + i + "回合
--------------------");
                    //
                    System.out.println("双方阵营：");
                    //
                    System.out.print("【" + campMap.get(0) + "】：");
                    for (int n = 0, o = 1; n < 6; n++) {
                        if (play[n].camps == "红方" && !play[n].isdie) {
                            System.out.print(" " + play[n].hero.name + "HP:" +
play[n].hero.hp + " ");
//                          if (o < 3)
//                          System.out.print("、");
//                          o++;
                        }
                    }
                    System.out.println();
```

```
                //
                System.out.print("【" + campMap.get(1) + "】： ");
                for (int n = 0, o = 1; n < 6; n++) {
                    if (play[n].camps == "蓝方" && !play[n].isdie) {
                        System.out.print(" " + play[n].hero.name + "HP:" +
play[n].hero.hp + " ");
//                      if (o < 3)
////                    System.out.print("、");
////                    o++;
                    }
                }
                System.out.println();
                //

//              System.out.println(redCamp);
//              System.out.println(blueCamp);//测试 1
                System.out.println();
                // 对于活人
                if (!play[no].isdie) {
                    System.out.println("每局抽取一张手牌");
                    play[no].selectCard(cardMap, 1, paiduiMap);// 每局抽一张牌。
                    int num = play[no].cardNum - play[no].hero.hp;
                    if (num > 0) {
                        for (int j = 0; j < num; j++) {
                            play[no].clearCard(cardMap, paiduiMap, 1);// 抽完牌丢
弃多余的牌//由于不能在循环中删除多个，所以循环删除一个
                        }
```

```
                    play[no].clearHpCard(cardMap, paiduiMap);// 血量限制手
牌
                }
            }
//          System.out.println(play[no].name+"是"+"【"+play[no].camps+"】"+"
【"+play[no].hero.name+"】"+play[no].handCard+play[no].cardNum+"张牌");
            System.out.println(play[no].name + "是" + "【" + play[no].camps +
"】" + "【" + play[no].hero.name + "】有"
                    + play[no].cardNum + "张牌");
            // 判断是否死亡，否则出牌
            if (play[no].hero.hp <= 0 && !play[no].isdie) {
                if (play[no].camps == "红方") {
                    redCamp.remove(0);
                    play[no].isdie = true;

                } else {
                    blueCamp.remove(0);
                    play[no].isdie = true;

                }
                System.out.println(play[no].hero.name + "死亡");
            } else {
                System.out.println("----【出牌】------------------");//
                play[no].chupai(play[no], play, paiduiMap, redCamp,
blueCamp);
                if ((int) redCamp.size() == 0 || (int) blueCamp.size() == 0)
                    break;// 出牌之前先判断结束游戏没有
                // 英雄技能
```

```
                if (play[no].hero.name == "刘备" && !play[no].isdie) {

                    play[no].hero.skills(play[no], play, cardMap);
                } else if (play[no].hero.name == "诸葛亮" && !play[no].isdie) {
                    play[no].hero.skills(play[no], paiduiMap);
                } else if (play[no].hero.name == "关羽" && !play[no].isdie) {
                    play[no].hero.skills(play[no], play, paiduiMap, redCamp,
blueCamp);
                } else if (play[no].hero.name == "貂蝉" && !play[no].isdie) {
                    play[no].hero.skills(play[no], play, paiduiMap, redCamp,
blueCamp);
                } else if (play[no].hero.name == "张飞" && !play[no].isdie) {
                    play[no].hero.skills(play[no], play, paiduiMap, redCamp,
blueCamp);
                } else if (play[no].hero.name == "吕布" && !play[no].isdie) {
                    play[no].hero.skills(play[no], play, paiduiMap, redCamp,
blueCamp);
                } else if (play[no].hero.name == "华佗" && !play[no].isdie) {
                    play[no].hero.skills(play[no], paiduiMap);
                } else if (play[no].hero.name == "黄盖" && !play[no].isdie) {
                    play[no].hero.skills(play[no], cardMap, paiduiMap, redCamp,
blueCamp);
                }

                System.out.println("----【出牌结束】------------------");
            }
            System.out.println("--------------------第" + i + "回合结束
--------------------");
            i++;
```

```
                System.out.println();
                System.out.println();
            }
            t++;

        }
        // 游戏结束
---------------------------------------------------------------------------------
        // 遍历所有英雄，找到唯一的存活的人
        if (redCamp.size() == 0) {
//                  System.out.println("蓝方获胜");
////                break;
            System.out.print("蓝方：");
            for (int n = 0; n < 6; n++) {
                if (play[n].camps == "蓝方" && !play[n].isdie) {
                    System.out.print(" " + play[n].hero.name + " ");
                }

            }
            System.out.print("获胜！！！");
        } else {
//                  System.out.println("红方获胜");
            System.out.print("红方：");
            for (int n = 0; n < 6; n++) {
                if (play[n].camps == "红方" && !play[n].isdie) {
                    System.out.print(" " + play[n].hero.name + " ");
                }
```

```
                }
                System.out.print("获胜！！！");
            }
//              for (int n = 0;n < 6;n++) {
//                  if(!play[n].isdie) {
//                      System.out.print(play[n].hero.name+"胜利，"+play[n].camps+"
胜利!");
//                  }
//              }

        }

}
```

游戏说明

一、 英雄杀休闲 3V3（150）

必须要实现的

1、 3V3 对战回合制游戏，总共 6 人游戏，3 个人一个阵营，都死亡的阵营失败。

2、 系统给每一方提供一定数量的英雄供选择，玩家选择的英雄不能重复，每个英雄拥有不同的血量上限（与初始血量相同）。

3、 玩家持牌数量≤血量。

4、 每一个英雄拥有不同的技能

5、 每轮可以抽取一张牌，牌包括

牌	功能
杀	攻击对方如果成功掉一点血
闪	在对方使用杀时可以用闪躲避
药	给自己加一点血

6、 可以使用牌

7、 提前设计好 6 个以上的英雄

选择实现的

1、 主动技能和被动技能

游戏的功能都已经实现，选择实现了主动与被动技能。

详细代码见压缩包

Big Assignment.zip	2021-01-09 下午 04:31	ZIP 压缩文件	34 KB