实验四 类与对象

学生姓名：黄晨箬　　学　　号：6109119066　　专业班级：计算机193班

实验类型：■ 验证 □ 综合 □ 设计 □ 创新　实验日期：2021.5.8　实验成绩：

一、实验目的

掌握Java语言面向对象的程序设计方法，理解对象的抽象和封装等概念，在调试实例程序后，能总结出面向对象的事务定义以及以对象的形式进行封装等内容。

二、实验内容

1、编写一个名为“复数”的类，包含复数的实部和虚部(数据成员），以及复数之间的基本算术运算：加、减（方法成员），并要求复数加减运算，程序运行中能对给定的复数进行运算，并打印运算的结果。

2、编写一个小游戏实现两人（Titan 和 Zues）格斗。

基本要求（Version1.0）：

（1） 创建一个 Titan 类,有 int 型数据成员 Energy；无参构造方法，可将 Energy 初始化为 800；带参构造方法，可将 Energy 初始化为参数指定值；数据成员 Energy 的设置器和读取器；成员方法 fight(Zues z)，每次调用此方法，随机生成一个 10~100 之间的整数攻击值，减少参数 z 的 Energy 值，并将结果输出，格式为“Titan 攻击 Zues，产生***点攻击值，Zues 当前 Energy 值为***”。

（2） 创建一个 Zues 类,有 int 型数据成员 Energy；无参构造方法，可将 Energy 初始化为 1000；带参构造方法，可将 Energy 初始化为参数指定值；数据成员 Energy 的设置器和读取器；成员方法 fight(Titan t)，每次调用此方法，随机生成一个 0~70 之间的整数攻击值，减少参数 z 的 Energy 值，并将结果输出，格式为“Zues 攻击 Titan，产生***点攻击值，Titan 当前 Energy 值为***”。

（3） 构建测试类，分别创建 Titan 对象和 Zues 对象，使用循环结构让 Titan 对象和 Zues 对象相互攻击，每次攻击完毕，判断 Titan 对象和 Zues 对象的 Energy 值，如果一方的 Energy 值小于 0 则停止循环，并输出结果：“Titan/Zues 的 Energy 值为***，已经失败，获胜者是 Titan/Zues！”

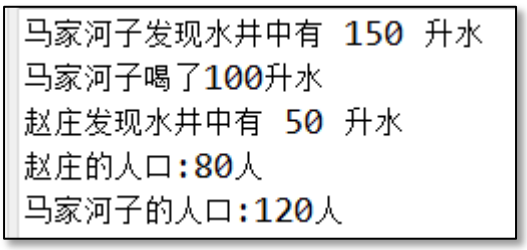
扩展要求（Version2.0）：在类中添加游戏角色的交手次数、经验值、生命值之间的关系，并断定角色决斗的胜负，游戏若能采用GUI界面设计更佳。

3、（共饮同井水）编写程序模拟两个村庄共用同一口井水。编写一个Village类(详见附件1：Village.java)，该类有一个静态的int型成员变量

waterAmount，用于模拟井水的水量。在主类Land 的main()方法中创建两个村庄，一个村庄改变了waterAmount 的值，另一个村庄查看waterAmount 的值。

(1) 画出Village类的UML类图。

(2) 请将Land.java源文件中[代码X](X:1--5)部分补充完整，使得程序运行效果如下图所示：



（3）在Land类的main方法中可否不定义name变量也能获得查看井水村庄的名字？若可行，修改程序使得能获得上图相同的运行结果。

(4) 【代码3】是否可以是Village.drinkWater(50)?为什么？

(5) 【代码4】是否可以是Village.getWaterAmount()?村庄的人数是否也可以采用Village.getPeopleNumber（）方式获得？

三、实验要求

1、加减运算能够接收不同类型的参数既可以实现复数与实数的加减、复数与复数的加减运算。

2、两游戏角色决斗。习题2中扩展部分规则参考：角色1（Titan）交手次数＋1，生命值－1,经验值＋2；角色2（Zue）交手次数＋1,生命值－2,经验值＋3。经验值每增加50时,生命值＋1；生命值<0判为负。生命值初始为1000,经验值初始为0。

3、给定二个不同的角色，判定交手的胜负关系。

4、实验报告给出决斗结果和交手次数

5、实验报告给出相关类图和所有源代码。

四、实验环境

1、PC微机；

2、DOS操作系统或 Windows 操作系统；

3、Eclipse程序集成环境。

五、实验步骤

内容一：

1、创建“复数”类Complex，定义实部、虚部成员

2、定义构造函数接收二个double参数用来初始化数据成员

3、定义二个复数运算函数plus()以及minus()各接收一个复数类型以实现复数与复数的加减运算。

4、定义二个复数运算函数plus()以及minus()各接收一个double类型以实现复数与与实数的加减运算。

4、定义一个打印方法。

5、在main()方法中创建复数对象并调用相关方法来验证。

内容二：

1、建立角色类，给出相应的成员，并能以生命值、经验值、能量值初始化角色对象。

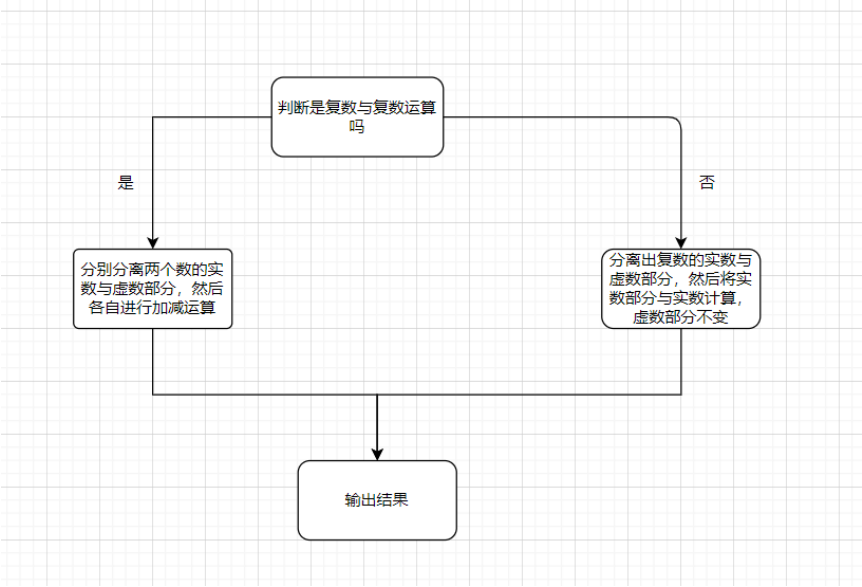
2、在角色类中建立fight方法，接收一个角色类型的参数并与之“战斗”，返回胜者信息。

3、在主函数中初始化二个角色，调用fight方法。

六、实验数据及处理结果

习题一：

[程序流程图]



[数据结构设计]

```
/*
    首先创建复数类并定义其构造函数，之后再根据复数的计算方式即实数与实数相加减，虚数与虚数相加减从而定义加减的方法函数。
    最后在main中创建并使用即可
*/
```

[程序源代码]

```
static class complex {
    double real;
    double imaginary;

    //定义构造
    complex(){
    }

    complex(double nowReal, double nowImaginary) {
        real = nowReal;
        imaginary = nowImaginary;
    }

    //实现复数与复数之间的加法计算
    String plus(complex num) {
        double addReal;
        double addImaginary;

        addReal = real + num.real;
        addImaginary = imaginary + num.imaginary;

        //区分实数部分是否为0的情况和虚数部分的正负情况
        if(addImaginary > 0) {
            if (addReal == 0) {
                return " + " + addImaginary + "i";
            }
            else {
                return addReal + " + " + addImaginary + "i";
            }
        }
        else if (addImaginary == 0) {
            return  addReal + " ";
        }
        else {
            if (addReal == 0) {
                return addImaginary + "i";
```

```
        else {
            return addReal + addImaginary + "i";
        }
    }
}

//计算复数与实数之间的加法
String plus(double x) {
    double plusReal = real + x;
    if (imaginary > 0) {
        return plusReal + " + " + imaginary + "i";
    }
    else if (imaginary == 0) {
        return plusReal + " ";
    }
    else {
        return plusReal + " " + imaginary;
    }
}

//计算复数与复数的减法
String minus(complex num) {
    double subReal;
    double subImaginary;

    subReal = real - num.real;
    subImaginary = imaginary - num.imaginary;
    if (subImaginary > 0) {
        if (subReal == 0) {
            return " + " + subImaginary + "i";
        }
        else{
            return subReal + " + " + subImaginary + "i";
        }
    }
```

```
        else if (subImaginary == 0) {
            return  subReal + " ";
        }
        else {
            if (subReal == 0) {
                return subImaginary + "i";
            }
            else {
                return subReal + subImaginary + "i";
            }
        }
    }

    //计算复数与实数的减法计算
    String minus(double x) {
        double minReal = real - x;
        if (imaginary > 0) {
            return minReal + " + " + imaginary + "i";
        }
        else if (imaginary == 0) {
            return minReal + " ";
        }
        else {
            return minReal + " " + imaginary;
        }
    }
}

    //定义一个打印方法
    static void prin(String x) { System.out.println(x); }
```

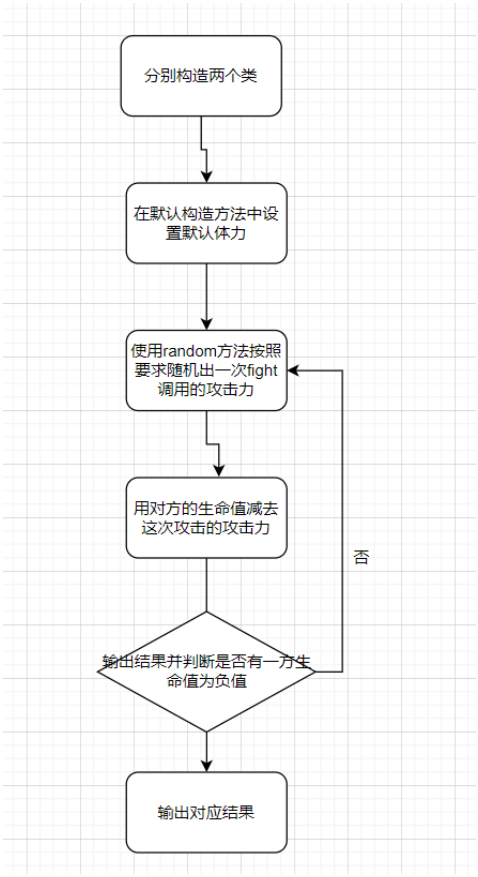
```java
public static void main(String args[]) {
    complex num1 = new complex( nowReal: 1,  nowImaginary: 1);
    complex num2 = new complex( nowReal: 1,  nowImaginary: -1);


    prin(num1.plus(num2));
    prin(num1.minus(num2));

}
```

[程序运行结果]

```
2.0
 + 2.0i
```

习题二：

[程序流程图]



[数据结构设计]

```
/*
    分别创建两个角色的类，并定义其默认的体力
    之后在fight方法中使用random定义造成的伤害，将其与对方的体力做减法运算
    当某次fight之后，双方有一方的体力为负数之后结束循环并输出对应的结果
 */
```

[程序源代码]

```
public class Game {

    //创建角色Titan类
    static class Titan {
        int Energy;

        Titan() {
            Energy = 800;
        }

        Titan(int e) {
            Energy = e;
        }

        void fight(Zues z) {
            int track = (int)(Math.random()*91)+10;
            z.Energy = z.Energy - track;
            System.out.println("Titan攻击Zues，产生" + track + "点攻击值，Zues当前Energy值为" + z.Energy);
        }
    }
```

```
    //创建角色Zues类
    static class Zues {
        int Energy;

        Zues() {
            Energy = 1000;
        }

        Zues(int x) {
            Energy = x;
        }

        void fight(Titan t) {
            Random r = new Random();
            int track = r.nextInt( bound: 70);
            t.Energy = t.Energy - track;
            System.out.println("Zues攻击Titan，产生" + track + "点攻击值，Titan当前Energy值为" + t.Energy);
        }
    }
```

```java
public static void main(String[] args) {
    Titan T = new Titan();
    Zues Z = new Zues();
    double i = 0;

    //判断T和Z的体力是否大于0，如果均大于0，则执行fight方法，如果T小于等于0，则输出结果，并跳出循环。

    while (T.Energy > 0 && Z.Energy > 0) {
        T.fight(Z);
        Z.fight(T);
        if (T.Energy <= 0) {
            System.out.println("Titan的Energy值为" + T.Energy +", 已经失败，获胜者是Zues");
        }
        else if (Z.Energy <= 0) {
            System.out.println("Zues的Energy值为" + Z.Energy + ", 已经失败，获胜者为Titan");
        }
    }
}
```

[程序运行结果]

```
Zues攻击Titan，产生7点攻击值，Titan当前Energy值为678
Titan攻击Zues，产生17点攻击值，Zues当前Energy值为601
Zues攻击Titan，产生6点攻击值，Titan当前Energy值为672
Titan攻击Zues，产生33点攻击值，Zues当前Energy值为568
Zues攻击Titan，产生40点攻击值，Titan当前Energy值为632
Titan攻击Zues，产生51点攻击值，Zues当前Energy值为517
Zues攻击Titan，产生59点攻击值，Titan当前Energy值为573
Titan攻击Zues，产生21点攻击值，Zues当前Energy值为496
Zues攻击Titan，产生27点攻击值，Titan当前Energy值为546
Titan攻击Zues，产生41点攻击值，Zues当前Energy值为455
Zues攻击Titan，产生67点攻击值，Titan当前Energy值为479
Titan攻击Zues，产生22点攻击值，Zues当前Energy值为433
Zues攻击Titan，产生8点攻击值，Titan当前Energy值为471
Titan攻击Zues，产生36点攻击值，Zues当前Energy值为397
Zues攻击Titan，产生12点攻击值，Titan当前Energy值为459
Titan攻击Zues，产生28点攻击值，Zues当前Energy值为369
Zues攻击Titan，产生23点攻击值，Titan当前Energy值为436
Titan攻击Zues，产生91点攻击值，Zues当前Energy值为278
Zues攻击Titan，产生60点攻击值，Titan当前Energy值为376
Titan攻击Zues，产生16点攻击值，Zues当前Energy值为262
Zues攻击Titan，产生66点攻击值，Titan当前Energy值为310
Titan攻击Zues，产生12点攻击值，Zues当前Energy值为250
Zues攻击Titan，产生66点攻击值，Titan当前Energy值为244
Titan攻击Zues，产生55点攻击值，Zues当前Energy值为195
Zues攻击Titan，产生15点攻击值，Titan当前Energy值为229
Titan攻击Zues，产生30点攻击值，Zues当前Energy值为165
Zues攻击Titan，产生23点攻击值，Titan当前Energy值为206
Titan攻击Zues，产生94点攻击值，Zues当前Energy值为71
Zues攻击Titan，产生54点攻击值，Titan当前Energy值为152
Titan攻击Zues，产生44点攻击值，Zues当前Energy值为27
Zues攻击Titan，产生69点攻击值，Titan当前Energy值为83
Titan攻击Zues，产生54点攻击值，Zues当前Energy值为-27
Zues攻击Titan，产生12点攻击值，Titan当前Energy值为71
Zues的Energy值为-27，已经失败，获胜者为Titan
```

习题三：

[程序流程图]

省略

[数据结构设计]

省略

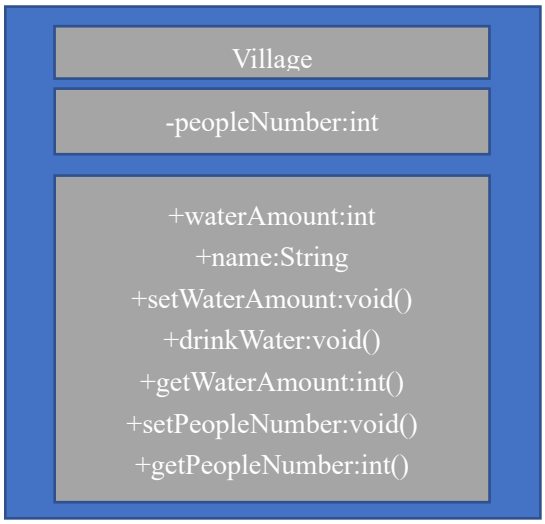
[程序源代码]

```java
public class Land {
    public static void main(String args[]) {
        Village.setWaterAmount(200); //用类名调用setWaterAmount(int m),并向参数传值200

        int leftWater = Village.waterAmount; //用Village类的类名访问waterAmount

        System.out.println("水井中有 "+leftWater+" 升水");
        Village zhaoZhuang,maJiaHeZi;
        zhaoZhuang = new Village( s: "赵庄");
        maJiaHeZi = new Village( s: "马家河子");
        zhaoZhuang.setPeopleNumber(80);
        maJiaHeZi.setPeopleNumber(120);

        zhaoZhuang.drinkWater( n: 50);//zhaoZhuang调用drinkWater(int n),并向参数传值50

        leftWater =maJiaHeZi.getWaterAmount();//maJiaHeZi调用getWaterAmount()方法
        String name=maJiaHeZi.name;
        System.out.println(name+"发现水井中有 "+leftWater+" 升水");
        maJiaHeZi.drinkWater( n: 100);

        leftWater =zhaoZhuang.getWaterAmount();//zhaoZhuang调用getWaterAmount()方法
        name=zhaoZhuang.name;
        System.out.println(name+"发现水井中有 "+leftWater+" 升水");

        int peopleNumber = zhaoZhuang.getPeopleNumber();
        System.out.println("赵庄的人口:"+peopleNumber+"人");
        peopleNumber = maJiaHeZi.getPeopleNumber();
        System.out.println("马家河子的人口:"+peopleNumber+"人");
    }
}
```

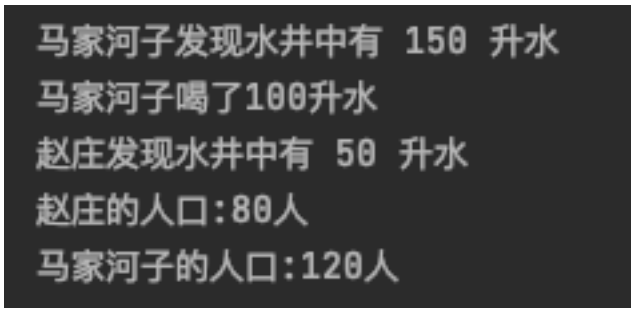
```java
public class Village {
        static int waterAmount;   //模拟水井的水量
        private int peopleNumber;         //村庄的人数
        String name;              //村庄的名字
        Village(String s) { name = s; }
        static void setWaterAmount(int m) {
            if(m>0)
                waterAmount = m;
        }
        void drinkWater(int n){
            if( waterAmount-n>=0) {
                waterAmount = waterAmount-n;
                System.out.println(name+"喝了"+n+"升水");
            }
            else
                waterAmount = 0;
        }
        static int getWaterAmount() { return waterAmount; }
        void setPeopleNumber(int n) { peopleNumber = n; }
        int getPeopleNumber() { return peopleNumber; }
    }
```

[程序运行结果]

（1）画出Village类的UML类图。



（2）请将Land. java源文件中[代码X](X:1--5)部分补充完整，运行得：



（3）在Land类的main方法中可否不定义name变量也能获得查看井水村庄的名字？若可行，修改程序使得能获得上图相同的运行结果。

要不定义name变量也能查看井水村庄的名字，只需要将name替换为对应村庄名.name即可。例如，将name修改为majiahezi.name。

（4）【代码3】是否可以是Village.drinkWater(50)?为什么？

不可以，因为我们无法从 static 上下文引用非 static 方法 'drinkWater(int)'

（5）【代码4】是否可以是Village.getWaterAmount()?村庄的人数是否也可以采用Village.getPeopleNumber（）方式获得？

代码四可以是Village.getWaterAmount，但是村庄人数不能采用Village.getPeopleNumber方式获得，因为我们无法从 static 上下文引用非 static 方法 'getPeopleNumber()'。

七、选做题（二选一）

1. 课本第10章习题347页10.4（Mypoint类）。编程定义一个点类“MyPoint”。编写一个测试类PointTest 来创建MyPoint 类的对象，创建两个点的（0，0）和（10，30.5）并显示它们之间的距离，测试该类。

```java
public class MypointNum {
    static class Mypoint {
        double x;
        double y;

        Mypoint() {
            x = 0;
            y = 0;
        }

        Mypoint(double a, double b) {
            x = a;
            y = b;
        }

        double distance(double a, double b) {
            double d_2 = Math.pow((x - a), 2) + Math.pow((y - b), 2);
            double d = Math.pow(d_2, 0.5);
            return d;
        }


        static double distance(Mypoint a, Mypoint b){
            return Math.pow(Math.pow((a.x - b.x), 2) + Math.pow((a.y - b.y), 2), 0.5);
        }

    }

    public static void main(String[] args) {
        Mypoint a = new Mypoint( a: 0,  b: 0);
        Mypoint b = new Mypoint( a: 10,  b: 30.5);

        System.out.println("其距离为" + a.distance(a,b));
        System.out.println("其距离为" + b.distance( a: 0, b: 0));
    }
```

```
其距离为32.09750769140807
其距离为32.09750769140807
```

2. 课本第10章习题349页10.10（Queue类）。编程定义一个队列类（Queue类），添加从1到20的20个数字到队列中，然后将这些数字移除并显示它们。

八、实验总结与体会

通过本次实验，更加深入的了解了类的相关知识内容，对对象的封装与使用更为娴熟。能够使用类的相关知识解决一些问题。在实验的过程中，遇到了一些static上下文的问题，在经过查阅资料之后对其有了更深刻的理解。