****

**Southwest university of science and technology**

JAVA实验报告

**复数类的设计**

|  |  |
| --- | --- |
| **学院名称** | 计算机科学与技术学院 |
| **专业班级** | 信安1802 |
| **学生姓名** | 侯天铭 |
| **学号** | 5120182377 |
| **指导教师** | 刘涌 |

2019年3月

**复数类的设计**

1. **实验目的**

分析复数类的设计过程和练习测试方法。

1. **实验内容**
2. **复数类的总体设计：**

Java语言中，一个类里面只含有属性和方法，属性是这个类中的每一个实例都具有的相同特征，方法则是这一个个实例都能完成的动作，或是操作。

就复数而言，每一个复数都具有的特征便是：由**实部**和**虚部**组成。因此实部，虚部便可以抽象为复数类的属性。再根据实部，虚部的取值范围均为实数，可进一步确定这两个属性皆应使用double定义。

同样的，对于复数而言，这个类中的实例所能完成的动作无非就是**复数的运算**，即加减乘除，以及求模。那么这些操作便可以对应地抽象为复数类中的方法，因为这是每一个复数都能进行的操作。

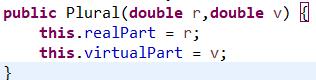
因此最终我们能得到复数类的大致结构，即属性由实部realPart，虚部virtualPart组成（皆为double类型）。方法由加plus()，减minus()，乘multiply()，（除法本次不予考虑），求模mod()组成。接下来只需要逐个完善这些个方法便可。

1. **测试类的设计：**
   1. **为何需要main()方法？：**

main()方法并**不是**面向对象程序的特征。因为前面说过，方法是一个类的所有实例都能进行的某种或某些操作，而生活中没有哪一个类是具有main()方法的，所以这个方法本身是无意义的。但是我们确实需要main()方法，是因为java程序规定了程序执行的入口在main()方法，因此main()仅仅起到一个执行程序的**入口**功能。

* 1. **如何创建复数对象？：**

创建对象，使用关键字new。假设我们定义了类Plural，可使用如下语句创建对象p1：Plural p1 = new Plural(); 而在创建对象的过程中，实际上是调用了Plural()这个构造方法。如果不加声明，我们创建了对象p1后需要依次为其实部，虚部赋值。如：p1.realPart = 2，p1.virtualPart = 3，这样才算完成了一个复数对象的创建。但我们还可以先在复数类中声明一个**构造方法**如下：



随后在创建对象时便可以直接赋值。如：Plural p1 = new Plural(2,3); 同样创建了一个实部为2，虚部为3的复数对象。

* 1. **如何从用户需求分析出发将数学表达式利用面向对象设计思想转换为程序表达式？：**

在本次实验课上，我们需要实现这样一个表达式：p5 = p1 + p2 – p3 \* p4（不考虑优先级）。对于用户而言，他想要的是输入p2，便能实现p1+p2，再输入p3，便能实现p1+p2后再减去p3，然后输入p4，就可以实现前两步运算的结果复数乘上p4。那么对于我们来说，实际上就是要将这些运算符，化为一个个的方法。通过分析可得：用户想要的是输入一个p2，然后完成p1+p2这个操作，因此对于plus()这个方法来讲，用户只想输入一个p2，便可以完成p1+p2，那么对于我们来讲，就需要使用p1这个对象来调用plus()这个方法，并且传入p2这个参数，从而完成加法的运算。并且运算结束后会得到一个新的复数，用户需要用它减去p3，那么我们就必须使这个新的复数去调用minus()方法，并把p3作为参数传入，来完成减法运算。之后便是同理，利用刚得到的新复数调用multiply()方法，并传入p4作为参数，完成乘法的运算并赋值给p5。最后利用p5调用求模mod()方法，求出该复数的模。由于就mod()方法而言，谁调用mod()便是求谁的模，因此不需要传入参数。

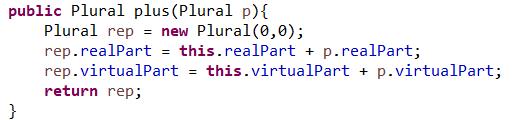
那么最终可将上述表达式化为如下表达式：

p5 = p1.plus(p2).minus(p3).multiply(p4)，然后求模。

1. **如何从程序表达式完善复数类的方法：**

在前面，我们得到了一个程序表达式：p5 = p1.plus(p2).minus(p3).multiply(p4)。以第一步plus()方法为例。

* 1. **传入参数**：首先我们是要计算p1+p2的值，p1，p2是两个复数，我们想要的是输入一个复数p2，程序便能完成复数p1+p2这样一个操作，也就是说，我们的输入只有一个值，p2，并且这个p2是一个复数。因此对于plus()方法而言，它的传入参数只有一个，并且是Plural类型的参数。
  2. **返回参数**：既然p1，p2都为复数，那么它们的和也应该为复数，也就是说复数对象p1调用它的plus()方法并传入参数p2，也就是p1+p2后所得的结果仍是一个复数。那么如果要将这个和返回并赋值给p5，这个返回值也应该是复数，即返回值也为Plural类型。
  3. **方法体的设计过程：**就plus()方法而言，首先我们知道这是一个实现复数相加的方法，而复数相加，我们知道是实部与实部相加，虚部与虚部相加。假如我们设p1+p2的和为复数rep（三者均为复数类的对象，均含有属性realPart和virtualPart）。那么rep的实部，即rep.realPart应该等于p1的实部加上p2的实部，同时rep的虚部，即rep.virtualPart应该等于p1的虚部加上p2的虚部。若是plus()方法定义如下：public Plural plus(Plural p){};则方法体内部使用p来访问p2的实部虚部，使用this来访问p1的实部虚部。可有如下代码：



我因为使用了构造方法，所以在创建rep对象时为其赋了初值，即实部虚部均为0。由此便完成了plus()方法的设计。

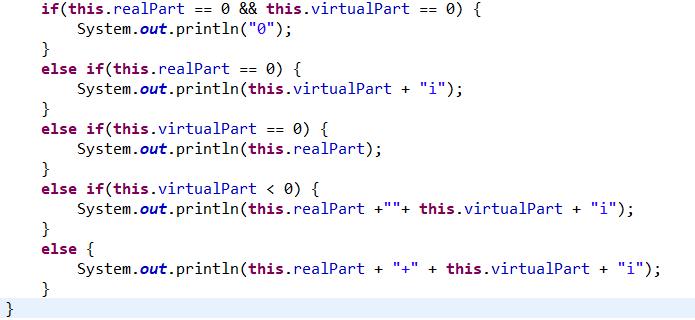
1. **完善printString()方法：**

考虑到所得结果复数中实部，虚部的取值会导致输出格式不同，因此该方法中应加入若干条件判断语句：

* 1. **实部虚部同时为0：**输出0即可。
  2. **实部为0：**只输出虚部加上i。
  3. **虚部为0：**只输出实部。
  4. **虚部为负数：**输出应去掉虚部前的’+’。
  5. **其他情况：**常规输出即可。

****

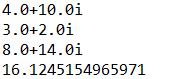
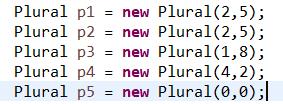
具体每一条的实现代码如下：



1. **实验分析**

对改进后的pringString()方法测试分析如下：

1. p1+p2结果实部虚部都为0，输出为0，正确。随后减去p3，虚部为复数，输出正确。
2. p1+p2实部为0，只输出虚部，输出正确。随后减去p3，虚部为0，只输出实部，输出正确。
3. 此为一般情况，输出正确。

以最后一种情况为例加法，减法，乘法，求模运算的输出结果均正确。