第 2 篇

面向对象程序设计篇

第4章　复数类

4.1　项目目标与任务

1.项目目标

(1)掌握 Java 中类的定义；

(2)掌握类中属性的声明和方法的定义；

(3)掌握声明和实例化对象的方法以及通过对象来调用属性和方法。

2.项目任务

该项目要定义复数类，并实现两个复数之间的加、减、乘操作。

4.2　项目分析

1.项目完成思路

根据项目任务描述的项目功能需求，本项目需要定义两个类，具体可以按照以下过程实现：

(1)定义复数类。

先定义一个复数类，该类中定义了表示实部和虚部的两个属性，定义复数类的构造方法能同时给实部和虚部赋值，定义实现复数加、减、乘的成员方法。

(2)定义测试类。

为了测试已经定义的复数类是否正确，需要定义带有 main 方法的测试类。在 main 方法中实例化两个复数类对象，通过对象调用复数类中的加、减、乘方法，并将计算结果的实部和虚部分别显示输出。

2.需解决的问题

(1)如何用 Java 语言定义一个复数类，如何定义复数类中的属性和方法？

(2)加、减、乘这三个成员方法需要定义几个形式参数，返回值类型是什么，怎么写方法体？

(3)如何声明并实例化复数类对象，如何通过复数类对象调用成员方法？

解决以上问题涉及的技术将在下一节技术准备中详细阐述。

4.3 技术准备

现实世界中我们看到的每一个具体事务如桌子、椅子、笔记本，乃至我们自己就是一个个具体的对象。如张三和李四在作为学生这点上有一些共同特点，都有学号、姓名、成绩等等，因此他们都是学生，但这些相同的特点在具体细节上又有所不同，因而他们成为学生类中迥然不同的两个对象。由此可见，在现实世界中是先有一个个具体的对象，为了更好的分析和认识这些对象，将这些具体的对象抽取出共有的特征而形成了类的概念。但在软件系统中为了描述真实世界中的对象则必须先描述类，这里先介绍如何使用 Java 语法来描述类。

4.3.1 类的定义

(1)示例代码。

【例 4-1】定义矩形类。

//定义类，类名为 Rectangle，文件名为 Rectangle.java。

```
class Rectangle {
    //定义属性，表示矩形的长和宽
    int length=1;
    int width=1;
    //定义方法，求矩形面积
    public int area(){
        int temp=length*width;
        return temp;
  }
}
```

编译该类后，除了保存的源文件 Rectangle.java 外，还生成了一个新的扩展名为 class 的文件：Rectangle.class，该文件就是编译类 Rectangle.java 生成的字节码文件。

(2)代码分析。

在例 4-1 中展示了在 Java 语言中如何定义一个类。首先类的定义是由关键字 class 描述的，其次类定义中可以包含属性定义和方法定义两个部分，这两部分是放在类定义的左右大括号之间。类中的属性表示该类对象的状态或者特征，而方法表示该类对象的行为，行为用于改变对象自身的状态，或者向其他对象发送消息。

Rectangle 是一个“矩形”类，要想描述一个矩形，必须指明这个矩形的长度和宽度。作为所有“矩形”对象的共有特征，在类定义时就需要定义两个属性 length 和 width，代表“矩形”的长度和宽度，这里我们给定了长和宽的初始值都是 1，其实我们完全可以不指定属性的初始值，而通过其他方法设定其值，关于如何设定我们在后续章节里会介绍。

在 Rectangle “矩形”类中还定义了一个方法 area，是用来求矩形面积的。我们知道矩形面积＝矩形的长度×矩形的宽度，因为矩形的长度和宽度已经作为类的属性定义过了，而属性在类定义体内(即类的左右大括号内)相当于全局变量，所以在 area 方法定义的头部我们不需要

再定义任何参数直接把定义过的属性拿过来用即可。但计算完矩形面积我们要把该值带出去，所以 area 方法的返回值不应该是 void 类型，同时因为 length 和 width 的类型都是 int 型所以二者乘积后的类型可以为 int 型，即方法的返回值类型定义为 int 型。

在这个示例程序中，属性声明放在方法定义之前。其实，在类定义中属性和方法可以按照任何顺序声明都是合法的。

(3)知识点。

下面以例 4-1 来说明在 Java 中定义(声明)一个类的基本语法结构。

①类的定义。

语法格式：

```
[修饰符] class 类名----------类头定义
{                    ----------类体定义
    [修饰符] 类属性定义
    [修饰符] 类方法定义
}
```

类的定义又可以称作类的声明，一般由两个部分组成：类头定义和类体定义。

②类头定义。

语法格式：

```
[修饰符]  class  类名(加[]的内容表示可选项)
```

例如：`class  Rectangle`

其中修饰符用来说明类的特殊性质，分为访问控制修饰符、抽象修饰符(abstract)、最终修饰符(final)三种，例 4-1 中的访问控制修饰符是缺省。类头定义中 class 是关键字，要小写。类名要符合 Java 语言定义标识符的规定，即：标识符可以由字母、数字、下划线或 $ 符号组成，对标识符的长度没有特别的限制；标识符必须以字母、下划线或 $ 符号开头；标识符区分大小写，不能使用系统的保留字。

【注意】class 要小写，Java 是大小写敏感的语言。

③类体定义。

类定义中包含在左右大括号之间的部分称作类体，类体定义主要完成对类的属性和方法的定义。

④类的属性。

定义属性的基本语法结构如下：

```
[修饰符] 变量类型 变量名=[变量初始值]
```

例如：

```
int  length;
```

属性的修饰符分为访问控制修饰符、静态修饰符(static)、最终修饰符(final)，而例 4-1 中属性 length 的访问控制修饰符是缺省的。

一个类的属性可以是简单类型，如 int、double、char 等，也可以是其他类的对象或数组等复杂的数据类型，如下例所示：

【例 4-2】其他类对象作本类属性。

```
//Face.java
class Eye{
    void open()
    {}
    void close()
    {}
}
class Face
{
    //Eye 类的对象作本类的属性
    Eye eyes;
    void smile()
    {}
}
```

在例 4-2 中 Eye 类对象 eyes 作 Face 类的属性。如果在引用(引用可以理解为是别名)eyes 属性前没有为其赋值(将其作为其他 Eye 类对象的引用,)系统会自动将 null(null 是 Java 系统提供的关键字,表示“空”)赋给 eyes。

【注意】

• 属性是指在类体左右大括号之间但在所有方法外定义的变量,如果是在类内且在方法内定义的变量是局部变量。

• 在类定义中,属性声明和方法定义可以采用任何顺序,并且属性如果没有给定初始值,系统会根据数据类型给其一个默认值。

⑤类的方法定义。

一个类的方法,是类和外部进行交互的途径。类的方法定义基本语法如下:

```
[修饰符] 返回值类型 方法名(形参列表)----------方法头
{
    局部变量定义
                                    ----------方法体
    语句序列
}
```

例如:

```
public int area()
{
    int temp=length*width;
    return temp;
}
```

其中类的方法的修饰符包括访问控制修饰符、静态修饰符(static)、抽象修饰符(abstract)、最终修饰符(final)。area 方法的访问控制修饰符是 public 的。

(4)举一反三。

下面仿照例 4-1 来完成一道练习，题目描述如下：

定义一个圆类，类名为 MyCircle。此类有一个属性 radius 表示半径，数据类型为整型，有一个方法 area 求圆的面积，完成此类的定义及编译。

4.3.2　创建对象

例 4-1 编译通过，如果试着运行一下这段代码，会看到有如下输出：

```
Exception in thread "main" java.lang.NoSuchMethodError: main
```

为什么系统会报错？如何才能测试 Rectangle 类？这是本节要解决的问题。

(1)示例代码。

为了回答前述两个问题，需要在例 4-1 上添加若干代码，完整程序如例 4-3 所示。

【例 4-3】 修订程序 4-1。

//添加代码，定义 Rectangle 的测试类。

```
public class RectangleTest{
//新添加代码!!
    public static void main(String[] args){
        //声明 Rectangle 对象 obj
        Rectangle obj;
         //实例化 Rectangle 对象 obj
         obj=new Rectangle();
         System.out.println("矩形长和宽是:"+obj.length+","+ob.width);
         /*通过对象名 obj 调用方法 area 计算面积
          *同时把计算结果赋给局部变量 result*/
         int result=obj.area();
         //输出面积值
         System.out.println("矩形面积是:"+result);
    }
}
    //原来例 4-1 的代码
    //Rectangle.java
    //定义类，类名为 Rectangle
    class Rectangle
    {
        //定义属性，表示矩形的长和宽
        int length=1;
        int width=1;
        //定义方法，求矩形面积
        public int area()
        {
            int temp=length*width;
            return temp;
        }
    }
```

编译运行得到如下结果：

```
矩形长和宽是:1,1
矩形面积是:1
```

(2)代码分析。

Rectangle 类和例 4-1 中是一样的，不再重复讲解，下面分析的重点集中在新添加的这段代码上。因为所有的 Java 程序都是由类组成的，为了测试 Rectangle 类的功能也需定义一个类即 RectangleTest 类，这个类的类头定义部分出现了一个新的修饰符 public，这是一个访问控制修饰符，被这个修饰符修饰的类原则上可以被任何其他类引用。这个测试类只有一个方法即 main 方法，main 方法是本程序执行的入口点，因此也把 RectangleTest 类称为主类。如果一个 Java 源文件中包含多个类定义，则此 Java 源文件的文件名必须和主类的类名一致，这就是在代码编辑时将源文件命名为 RectangleTest 的原因(思考：编辑例 4-2 时应如何命名源文件)。

main 方法的修饰符中出现了一个静态修饰符 static，被 static 修饰的方法是属于该类的方法，无需创建对象即可直接调用。关于 main 方法的方法头的构成目前只需按照例 4-3 中的书写记住即可。在 main 方法中完成了对 Rectangle 类的对象 obj 的声明及创建过程，分别对应 Rectangle obj 和 obj＝new Rectangle ()。其中 Rectangle obj 是对象的声明，这条语句只是简单地把类 Rectangle 和对象 obj 联系起来，此时并没有为 obj 分配内存空间，但该变量已有了一个特殊的值 null，表示对象尚未创建 new Rectangle ()是创建了类 Rectangle 的一个对象，也即在内存中为其分配空间。obj＝new Rectangle ()语句对 obj 而言，它的值是一个引用，是对 new Rectangle ()在内存中分配的一段存储地址的一个引用。

对象创建出来后，该对象的属性和方法就可以通过对象名加上“.”来引用，如例中“obj.length”通过对象名 obj 来引用 length 属性，而语句“result＝obj.area();”通过对象名 obj 来引用 area 方法，计算对象 obj 的面积值并赋给变量 result。

新添加的代码中还出现了一条具有向显示器输出功能的语句 System.out.println，这条语句中双引号中的内容原样显示在屏幕上，其中出现的“＋”起到连接字符串的作用，关于该语句的详细解释请参考后续章节。

(3)知识点。

①创建对象。

定义类的目的是为了使用类，使用类就要创建并操纵类的对象。创建一个对象就是在内存中为该对象开辟一段空间用来存放该对象的属性和方法。

语法格式 1：

```
类名  对象名;    对象名＝new 类名();
```

例如：

```
Rectangle obj;  obj＝new  Rectangle ();
```

语法格式 2：

```
类名    对象名＝new 类名();
```

例如：

```
Rectangle  obj=new  Rectangle();
```

【注意】new 需小写，两种语法格式等价。

②通过对象名调用方法。

创建对象后，如果希望对象能和外界交互，就要使用对象的方法。

语法格式：

```
对象名.方法(实参列表)
```

例如：

```
obj.area();
```

③通过对象名调用属性。

语法格式：

```
对象名.属性名
```

例如：

```
obj.width;
```

(4)举一反三。

下面请仿照例 4-3 来完成一道练习，题目描述如下。

有类 MyCircle 定义如下：

```
class  MyCircle
{
    int radius;
    double area()
    {   double PI=3.14159;
        return PI * radius * radius;
    }
}
```

为 MyCircle 类定义一个测试类，类名为 MyCircleTest，该测试类只有一个 main 方法，在 main 方法中声明实例化 MyCircle 类对象 obj，并将 obj 的半径 radius 赋值为 5，然后调用 area 方法计算面积值并输出到显示器上。

4.3.3　构造方法

在例 4-3 的类 Rectangle 中创建对象时是使用语句“new Rectangle ()”来实现的，这里的 Rectangle()就是构造方法，但是纵观 Rectangle 类定义，并没有定义构造方法，为什么在创建对象时可以直接调用？同时也应该注意到对象 obj 的长和宽都是预先设定的值 1，那么如何能创建出初始值多样的 Rectangle 类对象？本节将回答这两个问题。

(1)示例代码。

例 4-4 展示如何在类中定义带参的构造方法以方便创建出初始值多样的 Rectangle 对象。

【例 4-4】定义带参的构造方法。

```
//RectangleTest2.java
public class RectangleTest2
{
        //新修改代码!!
        public static void main(String[] args)
        {
                //调用带参的构造方法声明实例化 Rectangle 对象 obj
                Rectangle  obj=new Rectangle (2,3);
                //输出 obj 对象的长和宽属性值
                System.out.println("矩形的长和宽是:"+obj.length+","+obj.width);
                /*通过对象名 obj 调用方法 area 计算面积
                 *同时把计算结果赋给局部变量 result*/
                int result=obj.area();
                //输出面积值
                System.out.println("矩形面积是:"+result);
        }
}
  //Rectangle .java
    class Rectangle     //定义类,类名为 Rectangle
    {
        //定义属性,表示矩形的长和宽
        int length=1;
        int width=1;
        //定义方法,求矩形面积
        public int area()
        {
            int temp=length*width;
            return temp;
        }
        /*新添加代码,定义带参的构造方法分别为属性
         *length 和 width 赋值*/
        public Rectangle(int x,int y)
        {
            length=x;
            width=y;
        }
    }
```

编译运行得到如下结果:

```
矩形的长和宽是:2,3
矩形的面积是:6
```

(2)代码分析。

例 4-4 定义了两个类,即 Rectangle 和 RectangleTest。在 Rectangle 类中定义了一个新的带参数的构造方法,利用这个带参的构造方法再创建 Rectangle 对象时,可以根据给定的实参值创建出具有不同长度和宽度值的矩形对象,如 obj 的长度和宽度分别为 2 和 3。看一下这个构造方法,会发现它和前述介绍的类方法定义语法不太一样,最明显的地方就是它没有返回值类型,注意并不是说返回值类型为 void,而是没有,另外一个明显的特征就是构造方法的方法名

和类的名字一模一样。

下面在例 4-4 的 main 方法中做一点修改，main 方法改成如下：

```
public static void main(String args[])
{
    Rectangle  obj=new Rectangle ();//修改的地方!!
    System.out.println("矩形的长和宽是:"+obj.length+","+obj.width);
    int result=obj.area();
    System.out.println("矩形面积是:"+result);

}
```

再次编译修改后的代码，得到如下的错误提示：找不到符号，同时错误定位到刚刚修改过的那行代码。此例说明，当在类中只明确定义了带参数的构造方法后，则不能像没定义构造方法时那样，再调用无参的构造方法来创建类的对象。

如果希望例 4-4 修改后的 main 方法编译仍能通过，可以在 Rectangle 类定义中再添加一个构造方法即可，修改后的 Rectangle 如下所示。

```
//定义类,类名为 Rectangle
class Rectangle
{
    //定义属性,表示矩形的长和宽
    int length=1;
        int width=1;
    //定义方法,求矩形面积
    public int area()
    {
        int temp=length*width;
        return temp;
    }
    /定义带参的构造方法分别为属性 length 和 width 赋值*/
    public Rectangle(int x,int y)
    {
        length=x;
        width=y;
    }
    //添加一个无参的构造方法!!
    public Rectangle()
    {}
}
```

(3)知识点。

①构造方法是类的一种特殊方法，它的特殊性主要体现在如下几个方面：

- 构造方法的名字与类名完全相同；
- 构造方法没有返回值类型；
- 如果在定义一个类时没有定义构造方法，则系统会自动为该类生成一个构造方法，此构造方法的名字与类名完全相同，但它没有任何形式参数；
- 如果在定义类时只定义了带参的构造方法，则系统不会为其提供无参的构造方法，则此

时不可调用无参的构造方法来创建对象，除非又明确定义了无参的构造方法；

- 构造方法只能在用 new 创建类的对象时由系统调用。

②构造方法定义的语法格式如下：

[访问控制修饰符] 方法名(形参列表){方法体}

例如：

```
public Rectangle(int x,int y)
{
    length=x;
    width=y;
}
```

③构造方法调用的语法格式如下：

类名 对象名=new 构造方法(参数列表)

例如：

```
Rectangle  obj=new Rectangle (2,3);
```

引入构造方法是为了：

- 保证每个新创建的对象处于正常合理的状态；
- 引入灵活性，满足各种复杂操作的需要。

(4)举一反三。

请仿照例 4-4 完成如下题目：

定义一个笔记本类，该类有如下两个属性：颜色，数据类型为字符串(即 String 类)；cpu 型号，数据类型为字符串。该类有两个方法：(1)带两个参数的构造方法，完成对两个成员变量的初始化，两个参数分别是初始化时候需要的值(2)显示笔记本信息的方法 show()，该方法的功能是输出笔记本的颜色和 cpu 的型号。

定义一个笔记本类的测试类，该类只有一个 main 方法，在 main 方法中创建笔记本类的一个对象，并将其颜色初始化为“black”(黑色)，将其 cpu 型号初始化为“386”，然后调用 show 方法显示该对象的颜色及 cpu 型号。

(提示：定义字符串类型变量的语句是 String x；给字符串变量赋值时该值需用双引号引起来，如 x="wwww";)

4.4 项目学做

(1)定义复数类，有两个 double 型属性表示实部和虚部。

```
class ComplexNumber
{
    double  realPart;  //定义属性表示实部
    double  imagePart;  //定义属性表示虚部
}
```

(2)定义带参数的构造方法,给实部和虚部赋值。

```
ComplexNumber(int real, int image)
{
realPart=real;
    imagePart=image;
}
```

(3)计算两个复数的加法,因为方法是通过对象调用的,所以调用该方法的对象是参与运算的一个参数,因此形参只需一个即可。另外两个复数的加法结果也是复数,因此返回值类型是复数类型。

```
ComplexNumber  add (ComplexNumber another)
{
    double real=realPart+another.realPart;
    double image=imagePart+another.realPart;
    return new ComplexNumber(real, image);
}
```

(4)计算两个复数的减法,因为方法是通过对象调用的,所以调用该方法的对象是参与运算的一个参数,因此形参只需一个即可。另外两个复数的减法结果也是复数,因此返回值类型是复数类型。

```
ComplexNumber  sub(ComplexNumber another)
{
    double real=realPart-another.realPart;
    double image=imagePart-another.realPart;
    return new ComplexNumber(real, image);
}
```

(5)计算两个复数的乘法,因为方法是通过对象调用的,所以调用该方法的对象是参与运算的一个参数,因此形参只需一个即可。另外两个复数的乘法结果也是复数,因此返回值类型是复数类型。

```
ComplexNumber mul((ComplexNumber another)
{
    double real=realPart * another.realPart-imagePart * another.realPart;
    double image=imagePart * another.realPart-realPart * another.imagePart;
    return new ComplexNumber(real,image);
}
```

(6)定义含有 main 方法的测试类,在 main 方法中声明并实例化两个复数类对象 first 和 second,并调用加、减、乘法后输出结果的实部和虚部。

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

```
    {   //声明并实例化两个复数对象 first 和 second
        ComplexNumber first=new ComplexNumber(1,2);
        ComplexNumber second=new ComplexNumber(3,4);
     //将二者的和赋值给 addR,并输出其实部和虚部
        ComplexNumber addR=first.add(second);
        System.out.println("the sum realPart is:"+addR.realPart+"and imagePart is:"+addR.
      imagePart);
    //将二者的差赋值给 subR,并输出其实部和虚部
        ComplexNumber subR=first.sub(second);
        System.out.println("the sub realPart is:"+subR.realPart+"and imagePart is:"+subR.
      imagePart);
    //将二者的积赋值给 mulR,并输出其实部和虚部
        ComplexNumber mulR=first.mul(second);
        System.out.println("the sum realPart is:"+mulR.realPart+"and imagePart is:"+mulR.
      imagePart);
    }
}
```

在这个小项目里,我们定义了两个类一个是复数类 ComplexNumber 一个是测试类 Test,如果二者放在一个源文件中,则只能有一个类被 public 修饰。通常这个类也应该是 main 方法所在的类,如上所示我们将 Test 类定义为被 public 修饰,遵循 java 的源文件名应该和主类一致的原则,所以这个源文件被命名为 Test.java。当然如果我们一定要把这两个类都定义成被 public 修饰的也可以,但二者就要定义在两个源文件里了,并且这两个源文件一定被命名为 ComplexNumber.java 和 Test.java。

ComplexNumber 中有计算加,减,乘的三个成员方法,返回值都是 ComplexNumber 引用类型,也就是说测试类中的 addR,subR 和 mulR 都是这三个方法体中实例化出来的三个对象的引用。

可能有人会有疑惑:加,减,乘都是二元运算为什么只提供了一个形式参数?从测试代码可以看出在调用这三个方法的时候都是通过其中的一个对象 first 去调用的,所以 first 也是参与到这个二元运算中的一元,因此只需再提供一个参数就可以了。

有人会觉得 main 方法中在输出三个结果的时候把实部和虚部分开太繁琐,能不能直接输出 addR,subR,mulR 三个对象就自动能显示实部和虚部信息呢?要想达到这种效果,需要在 ComplexNumber 中添加如下的一个方法:

```
public String toString()
{
return  "realPart is:"+realPart+"and imagePart is:"+imagePart);
}
则 main 方法改成如下形式即可:
public static void main(String []args)
{
//声明并实例化两个复数对象 first 和 second
        ComplexNumber first=new ComplexNumber(1,2);
        ComplexNumber second=new ComplexNumber(3,4);
```

```
        //将二者的和赋值给 addR,并输出其实部和虚部
            ComplexNumber addR=first.add(second);
            System.out.println(addR);
        //将二者的差赋值给 subR,并输出其实部和虚部
            ComplexNumber subR=first.sub(second);
    System.out.println(subR);
        //将二者的积赋值给 mulR,并输出其实部和虚部
            ComplexNumber mulR=first.mul(second);
    System.out.println(mulR);
    }
```

如此修改后,在 main 方法中输出对象名即可自动调用 toString 方法,详情在后续中会有详述。

4.5　知识拓展

4.5.1　对象做方法的参数

前面介绍了对象可以作为另一个类的属性,同样对象也可以作为方法的参数,那么对象做方法的参数和基本类型做方法的参数有哪些不同呢? 这是我们这节要解决的问题。

(1)示例代码。

【例 4-5】基本数据类型做形参和对象做形参的区别。

```
// Test1.java
public class Test1
{
    public static void main(String [] args)
    {
        //声明并定义局部变量 local
        int local=0;
        //声明并创建对象
        One ex1=new One();
        //输出调用 add 方法前属性 a 的值,和局部变量 local
        System.out.println("before add ex1.a="+ex1.a+",local="+local);
        //方法调用
        ex1.add(ex1,local);
        //输出调用 add 方法后属性 a 的值,和局部变量 local 值
        System.out.println("after add ex1.a="+ex1.a+",local="+local);
    }
}
class One
{  int a;
    //构造方法
    public One()
    {
```

```
        a=0;
    }
    //方法定义,对象作形式参数
    public void add(One x,int y)
    {
        x.a++;
        y=y+1;
    }
}
```

编译运行得到如下结果：

```
before  add  ex1.a=0 , local=0
after   add  ex1.a=1, local=0
```

(2)代码分析。

例 4-5 类 One 有一个 add 方法，这个方法的两个形式参数分别由对象和基本数据类型充当，从测试类的运行结果能够看出在 add 体内对形参 y 的改变没有影响到实际参数 local，而对对象类型的形参 x 的修改影响到了实际参数 ex1。这种差别的产生是由于两个形参分别由对象和基本数据类型来充当造成的。因为方法调用时，基本数据类型作形参传递的是值，即是把实际参数 local 的值拷贝了一份副本，在 add 方法体内操纵的就是这个副本，所以方法体内的改变不能影响到实际参数。而对象作形参，方法调用时传递的是对象的引用，即此时形参、实参指向同一内存空间，所以对形参的改变能够影响到实际参数。这个过程可以如图 4-1 所示。

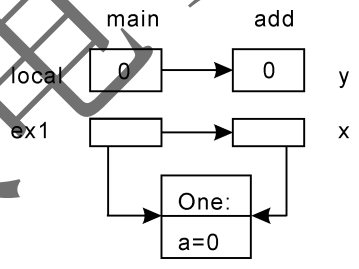

图 4-1　传值与传引用的区别

(3)知识点。

基本数据类型作形参和对象作形参有重要区别：

①基本数据类型作形参是实参将其值传递给形参，此时形参取得实参的一个副本，方法体内对形参的改变不会影响到方法外的变量值。

②对象作形参是实参将引用传递给形参，此时形参和实参指向同一内存空间，方法体内的任何改变都会影响实参对象。

(4)举一反三。

请给出如下程序的运行结果。

```
public class Ref
{
    int i=9;
    public static void main(String argv[])
    {
        Ref r=new Ref();
        r.amethod(r);
    }
    public void amethod(Ref r)
    {
        multi(r);
        System.out.println(i);
    }
    public void multi(Ref r)
    {
        r.i=r.i*2;
    }
}
```

4.5.2　终结器

前面介绍了如何声明并创建对象，其实对象和普通的变量一样都有其产生和消亡的过程，Java 对象的生命周期包括三个部分：对象的创建、对象的使用（包括对象变量的引用和对象方法的调用）和对象的释放。对类的对象来说如果用完就扔掉并不总是安全的选择。但是在前面所看到的所有示例中都没有涉及到对对象所占用空间的回收问题，因为 Java 提供了“垃圾收集”机制。在具体运行中，“垃圾收集”是由一个称为垃圾收集器的程序实现的。Java 运行系统会为对象对应的内存设标记，而当这个对象结束使用时会自动清除标记，有了这种标记设置和清除规则，垃圾收集器就可以周期性地扫描所有的 Java 对象有关的内存标记，将无标记的内存区列入可供分配的范畴，从而起到垃圾收集的作用。因为垃圾收集器涉及读写等操作，相对较慢，所以尽管扫描过程是周期进行的，但垃圾收集操作却以较低优先级留待系统空闲时才能得以完成。除了自动垃圾收集外，Java 运行系统也允许程序员调用方法 System.gc()来请求垃圾收集。此外，Java 系统开始运行时会自动调用一个名为 finalize()的方法，此方法的功能之一就是释放对象所占用的内存。正如构造方法是创建对象时所用的方法一样，finalize()方法就是当对象已经无用，需要销毁时回收对象所占空间时执行的方法，请看示例程序。

(1)示例代码。

【例 4-6】垃圾回收机制。

```
// FinalizeTest.java
class Ball
{
```

```
    public void finalize()
    {
        System.out.println("A ball has been free & collected");
    }
}
//测试类
public class FinalizeTest
{
    public static void main (String[] args)
    {
        //用这一行和下面一行分别测试运行结果;
        int number=100;
        //int number=10000;
        System.out.println("Start to create balls!");
        for(int j=0; j<number; j++)
        {
            Ball b=new Ball();
        }
        System.out.println("End of the program!");
    }
}
```

编译运行得到如下运行结果：

```
Start to create balls!
End of the program!
```

没有看到 finalize 执行的输出结果。把语句 int number=100;注释掉，把下一条语句前的注释符号去掉，再次编译运行，得到如下输出：

```
Start to create balls!
A ball has been free & collected
A ball has been free & collected
A ball has been free & collected
…
End of the program!
```

说明 finalize 方法被调用。但是发现在程序中并没有显示调用 finalize 方法，可见是由系统自动调用的。

(2)知识点。

①finalize 方法是系统在回收对象时候执行的方法。

②Java 中释放对象所占用的资源是由系统的垃圾回收机制自动完成的。

③终结器方法不能由编程人员显式调用执行。

④finalize 方法的原型如下：

```
protected void finalize();
```

4.6　强化训练

定义一个二维空间的点类，有横、纵坐标信息，有计算两点之间距离的方法，有将当前点的横、纵坐标移动一定距离到下一个位置的方法。定义一个测试类测试点的这两个方法。

4.7　课后习题

一、选择题

1. 设 A 为已定义的类名，下列声明 A 类的对象 a 的语句中正确的是(　　)。

(A) float　A　a　　(B) public　A　a＝A()

(C) A　a＝newint()　　(D) static　A　a＝new　A()

2. 设 A 为已定义的类名，下列声明 A 类的对象 a 的语句中正确的是(　　)。

(A) public　A a＝new A()　　(B) public　A　a＝A()

(C) A a＝new class()　　(D) a　A

3. 设 X、Y 均为已定义的类名，下列声明类 X 的对象 x1 的语句中正确的是(　　)。

(A) publicX　x1＝new　Y()　　(B) X　x1＝X()

(C) X　x1＝new　X()　　(D) int X　x1

4. 设 X、Y 为已定义的类名，下列声明 X 类的对象 x1 的语句中正确的是(　　)。

(A) static X x1　　(B) public X x1＝new　X(int 123)

(C) Y x1　　(D) X　x1＝X()

5. 设 i，j 为类 X 中定义的 int 型变量名，下列 X 类的构造方法中不正确的是(　　)。

(A) void X(int k){ i＝k; }　　(B) X(int k){ i＝k; }

(C) X(int m, int n){ i＝m; j＝n; }　　(D) X(){i＝0;j＝0; }

6. 有一个类 A，以下为其构造方法的声明，其中正确的是(　　)。

(A) public A(int x){…}　　(B) static A(int x){…}

(C) public a(int x){…}　　(D) void A(int x){…}

7. 设 i，j 为类 X 中定义的 double 型变量名，下列 X 类的构造方法中不正确的是(　　)。

(A) double　X(double k){ i＝k; return i; }　　(B) X(){i＝6;j＝8; }

(C) X(double m, double n){ i＝m; j＝n; }　　(D) X(double k){ i＝k; }

二、填空题

1. 通过类 MyClass 中的不含参数的构造方法，生成该类的一个对象 obj，可通过以下语句实现：_____。

2. 下面是一个类的定义，请完成程序填空。

```
public  class  _____
{
    int  x, y;
```

```
    Myclass(int  i, _____)   // 构造方法
    {
        x=i;   y=j;
    }
}
```

3. 下面是一个类的定义，请将其补充完整。

```
class  _____
{String   name;
    int    age;
    Student(  _____  s, int  i)
    {
        name=s;
        age=i;
    }
}
```

4. 下面是一个类的定义，请将其补充完整。

```
_____  A
{  String  s;
    _____  int  a=666;
    A(String s1)  {    s=s1;    }
    static int geta()  {    return  a;  }
}
```

5. 下面程序的功能是通过调用方法 max()求给定的三个数的最大值，请将其补充完整。

```
public class Class1{
   public static void main( String args[]){
     int i1=1234,i2=456,i3=-987;
     int MaxValue;
     MaxValue=_____;
     System.out.println("三个数的最大值:"+MaxValue);
   }
   public _____ int max(int x,int y,int z)
   {  int temp1,max_value;
      temp1=x>y? x:y;
      max_value=temp1>z? temp1:z;
      return max_value;
   }
}
```

三、程序阅读题

1. 下面是一个类的定义，根据题目要求回答以下问题。

```
class  B{
    private int x;
    privatechar y;
    public B(inti,char j){
        x=i; y=j;
    }
    public void show(){
```

```
        System.out.println("x="+x+"; y="+y);
    }
    public void methodC(int x){
        this.x=this.x+x;
        y++;
        show();
    }
}
```

(1)定义类 B 的一个对象 b,将类中的变量 x 初始化为 10、变量 y 初始化为'A',请写出相应的语句。

(2)若在(1)问基础上有方法调用语句:b.show();则输出如何?

(3)若在(1)问基础上增加语句:b.methodC(1);则输出为何?

2.阅读程序,回答问题。

```
public  class  Test52 {
    static  String  str1="Hello, Java world! \t";
    static  String  str2="Hello, students! ";
    public  static  void  main(String  args[])
    {   System.out.print(str1);  System.out.println(str2);   }
}
```

(1)这是哪种形式的 java 程序 ?

(2)程序的输出是什么?

四、编程题

编写账簿类 AccountBook 类。类中有三个属性:accountName(String 类型)、income(double 类型)和 outcome(double 类型)。要求设置收入/支出访问器方法时收入额/支出额不能为负数。定义通过收入和支出自动计算余额的方法 compute。再编写测试类,在测试类中用 AccoutnBook 类创建一个账簿对象,设置名称是"张三",收入 30,支出 10,并将账户名称"张三",该账户收入额和支出额及该账户的余额输出到控制台。