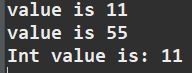
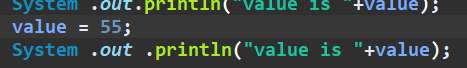
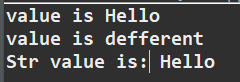
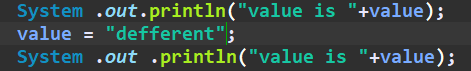
# 三次上机报告

## PPT练习

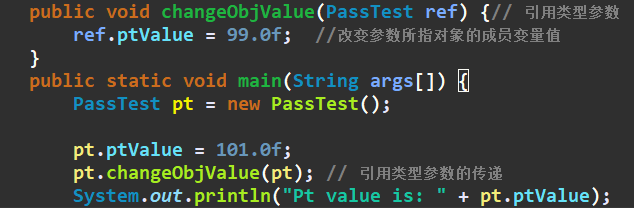
1. P15、P16、P17

在无论是基本类还是引用型变量作为形参传递给方法的时间，本质都是相同的。传递的都是变量的值，只不过基本类型变量中存储的是数值，而引用类型中存储的某个对象的（例如：“HELLO”字符串的地址）引用（即地址），所以在形参value存储的是“HELLO”字符串的地址，在输出后是字符串“HELLO”,然后让value变量指向“different”输出的就是“different”。所以在value=“different”的时间没有修改str的值的原理。只是将str的值“drfferent”给了value，并不是将str变量的地址给了value，所以不能够借助此方法来修改str。



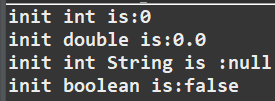
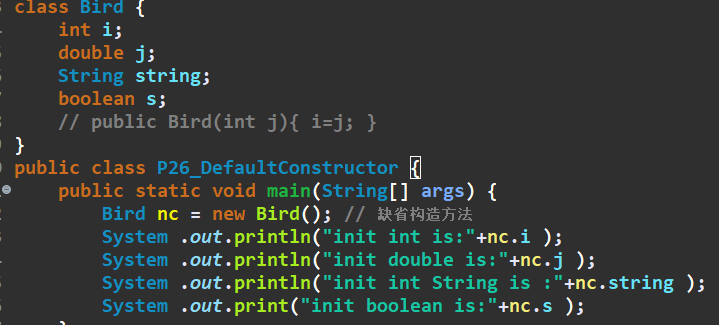


假如需要借助方法来更改，可以在传递引用类型的变量后，使用该变量（对象）的“.”运算符来修改其成员变量的值。(或许在这里的.也类似于C的解引用\*)

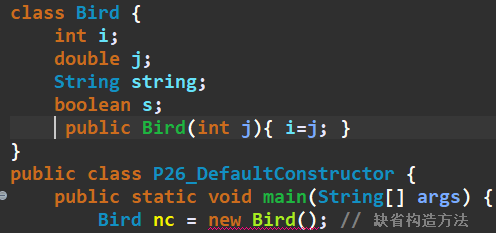
 

1. P26

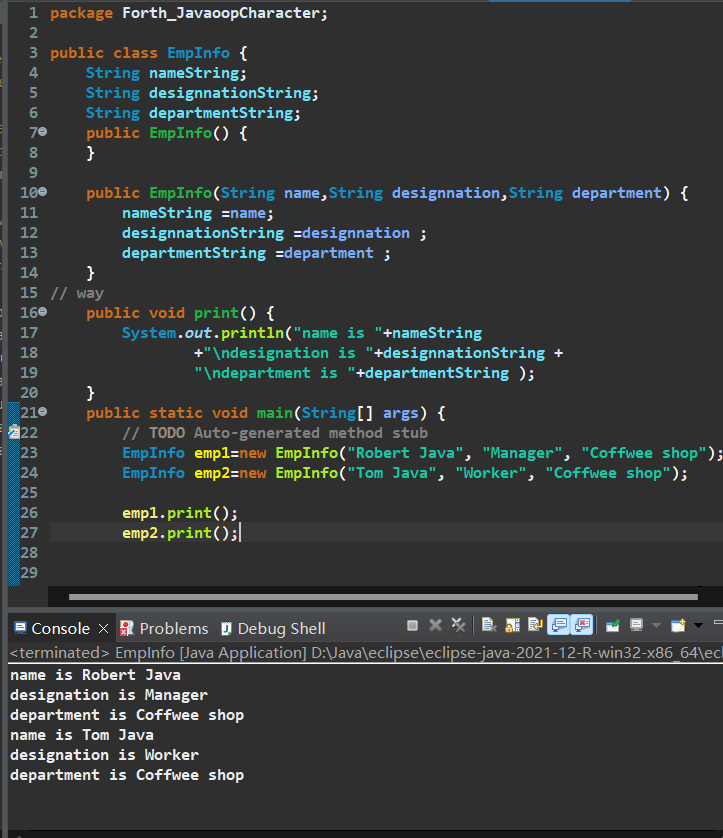
构造方法必须与类是同名的，如果没有，系统自动创建们没有变量的构造方法，其类成员变量的初始化（数值类：0、boolean：false、char：‘\0’、对象：null）。



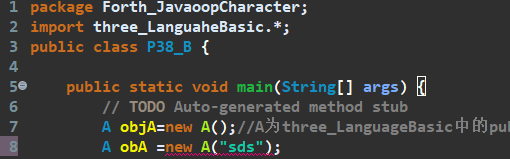
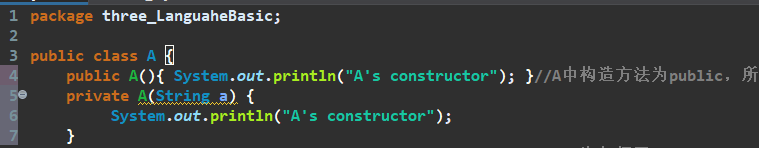
类中定义了构造方法后，系统不会添加默认构造方法。



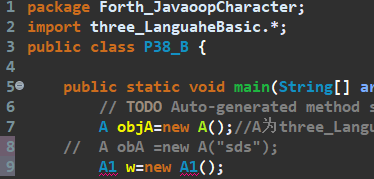
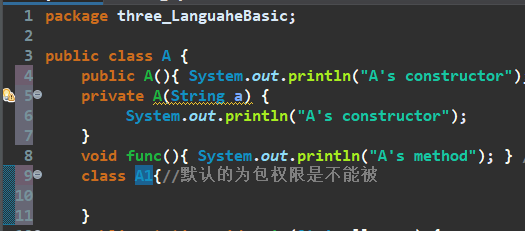
1. P27



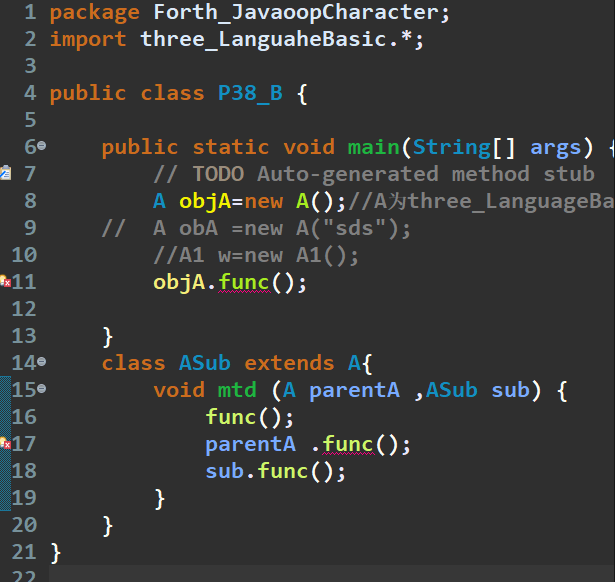
1. P38、P40、P42
   1. 在public类的A类中，其中的public类的构造方法可以被不是一个包中的B类进行访问，但是private类的构造方法就不能被不是一个包且不是相同类的类进行访问。不能生成该类的实例。



* 1. 默认的访问权限只有在相同的类或者是在同一个包中进行访问，不是同一个包的类也是不能访问的。



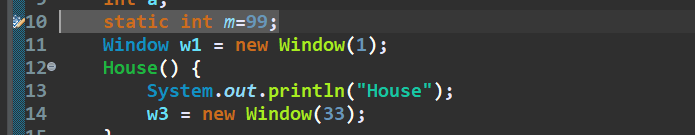
* 1. Protect可以被该成员所在的包中的类进行访问，并且也可以被他的子类访问无论是否在同一个包内。在forth包中引入three包中的所有类后，使得ASub继承于A类，则A类中的func（）方法的访问权限是Protected所以他的子类ASub可以访问func（）。Protected的方法在外包中只能够被其直接子类进行访问，即子类可以在外包中通过这个protected的方法来操纵C的某些被保护的成员变量。而且在外包类中，这个类本身的对象也是不可以访问它的Protected方法。而parent.func只是C类型的形参并不是旨在执行CSub的直接父类，所以也是不能访问C的protected方法。

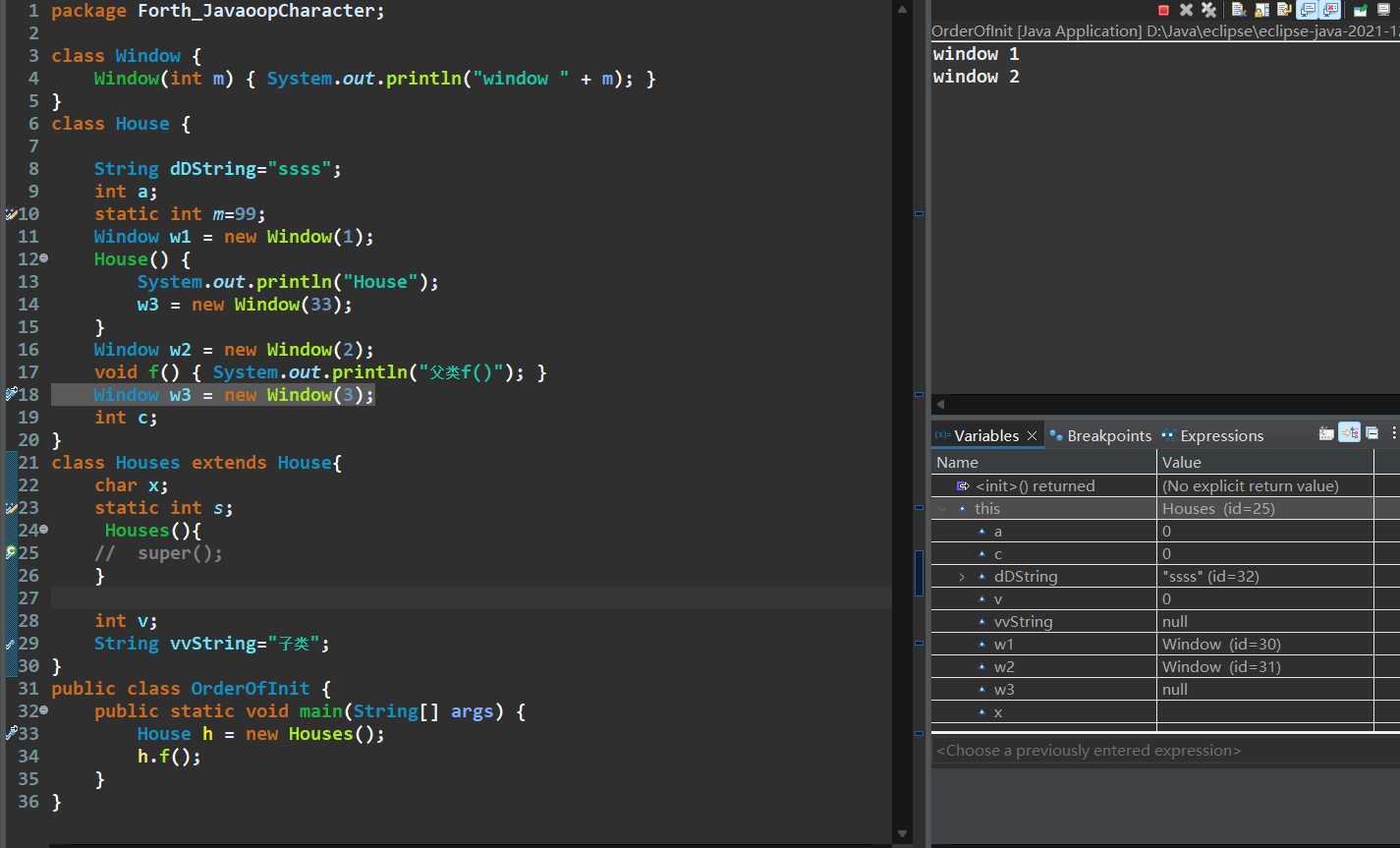


1. P51

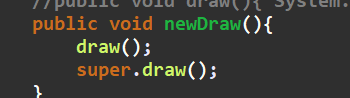
在Java中实例化一个对象的时间大分为以下几个步骤

* 1. 创建类的引用，如果该类或者继承的类中有static类的方法或者成员的时间，会先对static类型初始化，基本初始化将所有的实例变量初始化为相应的“0”。
  2. 之后调用子类的构造方法进行参数绑定，如果有显示的super（）调用的话会进行到直接父类的无参构造方法的调用，一直递归调用至Project根类，然后进行显式初始化，初始化后再进行构造方法体里面的内容，之后进入到其直接子类中，再进行显式初始化，直到最后一个子类的成员被显式初始化完毕。即创建一个实例化对象的步骤完成。

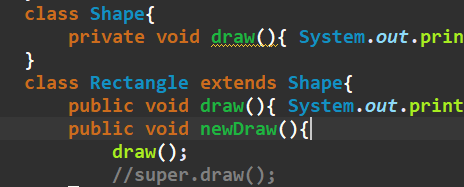




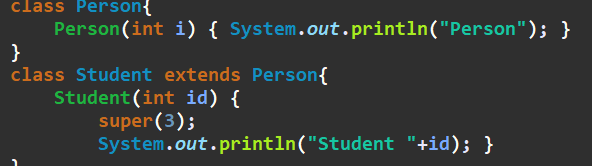
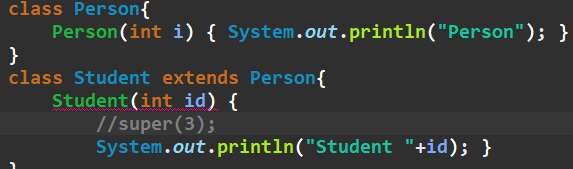
1. P61
   1. 在父类中继承的方法Public访问权限的draw()方法，如果在子类中没有重写该方法的时间，draw（）和super.draw()效果是相同的，都是调用父类的draw（）。



* 1. 如过父类中的draw（）方法访问权限是指为private时，子类将不能在继承，效果与final类似，如果有和父类同名的方法，则与父类同名的方法没关系。

。

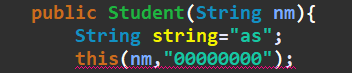
1. P65
   1. 子类对象初始化的一般步骤，为当前子类开辟空间，并将成员变量赋值为相应的“0”，绑定构造方法中的参数列表，调用父类的默认构造方法，进入父类中，开辟空间赋值为相应的“0”，如果子类没有显式的调用父类的构造方法，系统会自动的调用父类的缺省构造方法，如果父类不存在缺省构造方法会 报错。至此一直递归到父类的根部。
   2. 到达根部后，对父类进行显式初始化，在进行构造方法的其余部分，直到到达实例化的子类完毕。
   3. 在下图中展示了Person父类中没有添加缺省构造方法，并且它的子类Student中没有显式的调用有参数的构造方法，所以在其直接子类Student中出现报错。相反添加后会正常。



1. P66
   1. 在创建子类Sandwich的时间自动调用其直接父类PortableLnch的缺省构造方法

一直递归到“根部“Meal，然后执行根部Meal类构造方法的剩余部分，即输出语句“Meal（）”一直按照顺序到Sandwich（），执行完毕Sanwich（）的剩余构造方法体后，在进行显式其显式初始化（前面的父类没有进行显式初始化是因为没有为其显式初始化）。在显式初始化的过程中相应的调用对应的缺省构造方法，完成与Sanwich相同的一系列操作（前提是有相应的父类，和显式初始化的语块）。

1. P72
   1. 构造方法的重载，在一个构造方法中调用另外一个构造方法的时间，使用当前类的应用this来调用其他的构造方法，并且这个调用必须放在构造方法体的首行。

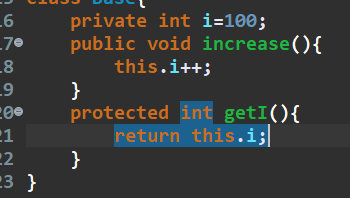
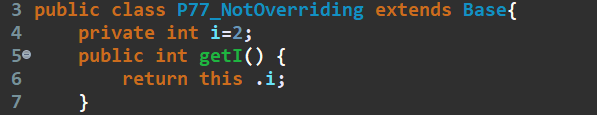


1. P73
   1. 初始化块和显式初始化是顺序相同的，在没有被显式初始化的J变量在初始化块中去调用J的时间报错，变量未定义。

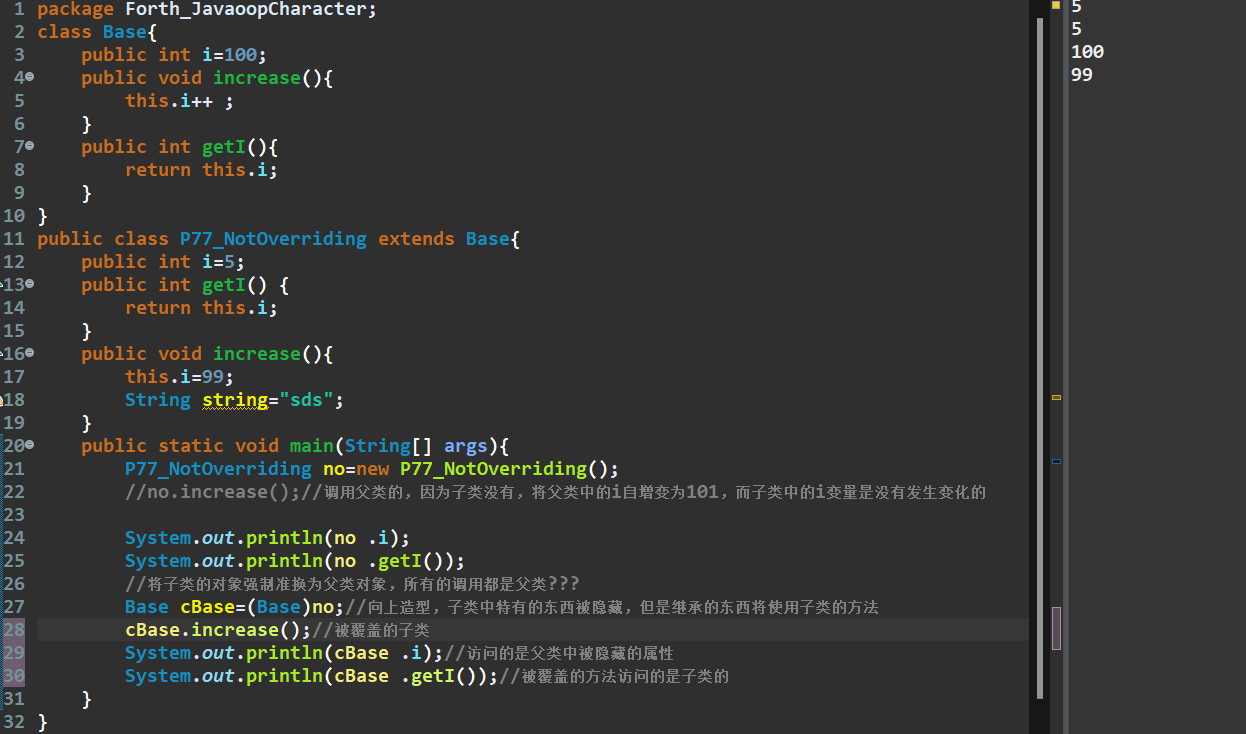


* 1. 在无参构构造方法中调用有参的构造方法的时间，必须使用当前类的引用this来调用，并且必须放在首行。否则直接使用构造方法名会被当做方法处理。
  2. 在子类中的构造方法，如果没有显式的调用父类的构造方法的时间会自动调用父类的缺省构造方法。访问父类的成员的时间可以使用super关键字。

1. P77 方法的重写
   1. 子类实例对象no.i访问的是子类中的对象，父类和子类有相同的成员的时间，子类变量覆盖了父类的成员。输出的是子类自己拥有的成员。



* 1. 存在被隐藏的属性i和被覆盖后的方法getI（）。被隐藏的属性在强制转换为父类后cBase访问到的是父类的属性，而强制转换后访问的被覆盖的方法是子类中的方法。



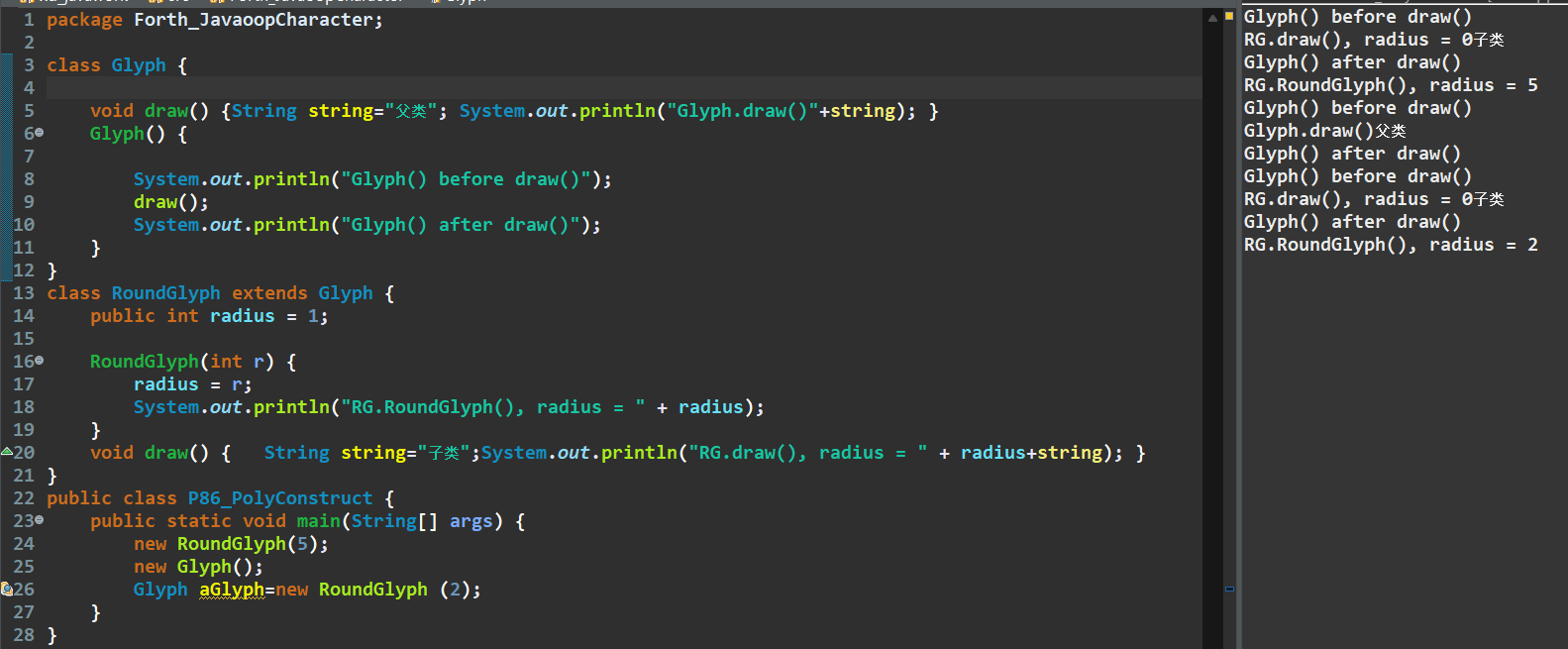
1. P78 没有覆盖的方法
   1. 方法f的访问权限是private不会被子类继承，也就不会被子类重写，覆盖。所以在强制转换（子->父）后访问f（）方法的时间直接访问父类的方法f（）。
   2. 如果不需要进行强制转换的时间访问父类的方法可以使用super。



1. 引用变量出现编译时类型（声明）和运行时类型（new）不一致的情况 编译时Father类型的 运行时是Son类型的。当变量的编译时类型和运行时类型不一致时，通过变量访问它所引用的对象的实例时，该实例变量的值由声明该变量的类型决定。通过变量访问它所引用的对象的方法时，该方法的行为由所引用的对象实际类型所决定
2. 在强制转换的时间声明类型和运行类型都是最后转向的类型。



1. P86
   1. 在创建一个拥有父类的子类对象实例的时间在其构造方法中，被调用的一般方法的行为具体看的是实例对象的类型，如果是子类，那么就是调用的子类的方法，如果是父类就调用的父类的方法；在者如果是向上造型对被覆盖的方法访问的就是子类中的方法，如果是属性的话访问的是父类中的属性。与上文中11是类似的属性的隐藏和方法的覆盖是一样的。



MyInterger和MynaturalInterger测试，处理任意位数的数据加减法

错误的输入结果显示

