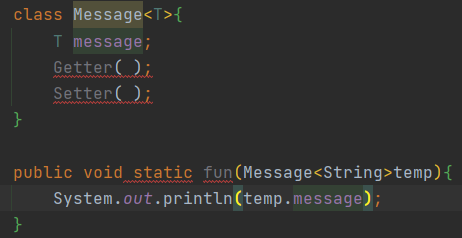
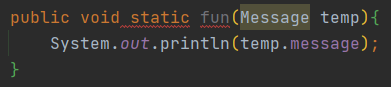
**一，泛型带来了新的问题——引用传递处理**



fun方法只能接受泛型为String的Message，其他的不能接受，那么有没有办法让fun可以接受任意泛型的Message呢？

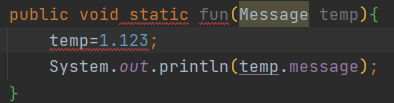
一种解决思路是——不要泛型



虽然这样可以让fun接受任意泛型的Message

但是又会衍生出一个新的问题

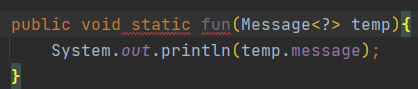
就是如果在fun中我们对temp的属性进行修改



那么不管temp的泛型是什么类型，都会被我们修改为1.123（double）

这就产生了安全隐患

所以我们采用**泛型通配符（ <?> ）**在解决这个问题



**泛型通配符<?>**

**①表示可能为任意类型，所以可以放入任意类型**

**②不可以对temp的属性进行修改，如果修改，编译时就会报错**

**二，泛型通配符的另外两种形式——<? extend 类> 和<? super 类>**

**①<? extend 类A>—定义类泛型的上界，表示这可能是A的泛型，也可能是A子类的泛型**

**例子：**

**ArrayList heroList<? Extends Hero> 表示这是一个Hero泛型或者Hero子类的泛型**

**heroList的泛型可能是Hero，ADHero，APHero**

**但可以确定的是——heroList中取出的东西一定可以转换为Hero**

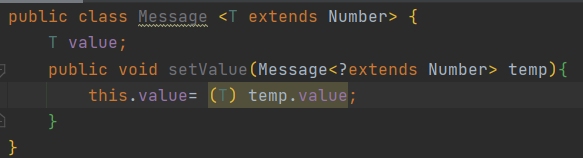
**但是我们不能往里加东西**

**因为放APHero就不满足<ADHero>,放ADHero就不满足<APHero>**

**所以对于一个 <?extends A>的泛型——我们可以从里面取东西，不能往里面放东西，所以我们无法创建对应的ArrayList，因为它无法放东西**

****

**因为<? Extends A>只能取东西，不能放东西，所以<? Extends A>一般用才方法的参数上**

****

**表示传进来的泛型temp中，可能含有Number，可能含有Number的子类**

**②<? super 类A>—规定了泛型的下界，表示这可能是A的泛型，也可能是A父类的泛型**

例子：

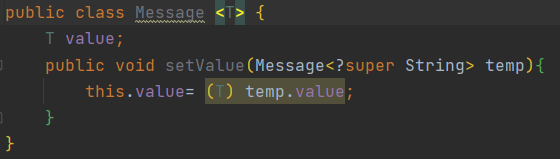
ArrayList heroList<? Super Hero>表示这可能是Hero的泛型，也可能是Hero父类的泛型

所以 heroList可能是Hero泛型，也可能是Object泛型

**对于这种情况，我们从中取元素出来无法知道元素是什么类型——有风险**

**但是我们往里面插入时没有风险的**

**所以 <?super A>一般用于 只插入，不取出**



Class Message<T>的<>中不能有super

**三，总结**

**如果希望 只取出，不插入——用<?extends A>**

**如果希望 只插入，不取出——用<?super A>**

**如果希望 能取出，能插入——用<?>**