逆变与协变

作者：flyue
原文链接：https://ld246.com/article/1546412671832
来源网站：链滴

<h5 id="Java中的逆变与协变">Java 中的逆变与协变</h5>
<pre><code class="language-java highlight-chroma"><span c
ass="highlight-cl">Number
um=new <s
an class="highlight-n">Integer(<span class="hi
hlight-mi">1);
<span class="high
ight-n">ArrayList<<span class="highlight-n"
Number> list</s
an>=new <span clas
="highlight-n">ArrayList<<span class="highli
ht-n">Integer>(); <span class="highlight-c1"
//type mismatch
<span cla
s="highlight-c1">List<span class="highlight-o"
<? extends Num
er> list1
span class="highlight-o">=new <span class="hi
hlight-n">ArrayList<<span class="highlight-n
>Number>();
<span class="high
ight-n">list1.add</
pan>(new <span cla
s="highlight-n">Integer(<span class="highlight
mi">1)); //error
<span cla
s="highlight-c1">list1<span class="highlight-o
>.add(<sp
n class="highlight-k">new Float<span class="hi
hlight-o">(1.2f));</
pan> //error
</code></pre>

里氏替换原则

<blockquote>
所有引用父类的地方必须能透明的使用其子类对象
</blockquote>

包含以下四层含义：

子类完全拥有父类的方法，且具体子类必须实现父类的抽象方法
子类可以增加自己的方法
当子类覆盖或者实现父类的方法时，方法的形参要比父类更为宽松
当子类覆盖或者实现父类的方法时，方法的返回值要比父类更严格

<pre><code class="language-java highlight-chroma"><span c
ass="highlight-cl">// 根据里氏替换原则，我们在实例对象时，可以
用其子类进行实例化
<span cla
s="highlight-c1">// 左侧需要的是Number 类型的对象，可以使用Integer子类
<span cla
s="highlight-c1">Number <span class="highlig

t-n">num=new</sp
n> Integer(<span cla
s="highlight-mi">1);
</code></pre>

<ol start="2">

逆变与协变定义
逆变与协变用来描述类型转换后的继承关系，其定义:
如果 A、B 表示类型, f(.) 表示类型转换，< 表示继承关系(比如 A<B, 表示 A 是 B 的子类)</
>

当 f(.)是 逆变 时，如果 A<B，那么 f(B)<f(A)
当 f(.)是 协变 时，如果 A<B，那么 f(A)<f(B)
当 f(.)是 不变 时，如果 A<B，那么上述两个式子都不成立。即 f(A)和 f(B
没有继承关系

<pre><code class="language-java highlight-chroma"><span c
ass="highlight-cl">// 数组是协变的 :令f(A)=[]A,有
<span cla
s="highlight-c1">// Integer[]数组可以使用Number[]接收，即Number[]的引用可以使用Integer[]
象
<span cla
s="highlight-c1">// 所以 Integer[] < Number[] ， Integer[]是Number[]的子类
<span cla
s="highlight-c1">Number<span class="highligh
-o">[] numbers=</s
an>new Integer<sp
n class="highlight-o">[3<span class="highligh
-o">];
<span class="high
ight-c1">// 泛型是不变的 :令f(A)=ArrayList<A>()
<span cla
s="highlight-c1">// ArrayList<Number> 不能接收ArrayList<Integer> ,反之也不行，
俩者之间没有继承关系
<span cla
s="highlight-c1">// 所以泛型是不变的
<span cla
s="highlight-c1">ArrayList<span class="highlig
t-o"><Number&gt
 numList=<
pan class="highlight-k">new ArrayList<span cla
s="highlight-o"><Integer<span class="highli
ht-o">>(); //error
<span cla
s="highlight-c1">ArrayList<span class="highlig
t-o"><Integer&gt
 intList=<sp
n class="highlight-k">new ArrayList<span class
"highlight-o"><Number<span class="highligh
-o">>(); //error
<span cla
s="highlight-c1">// 方法调用： 当调用方法result=method(n);

```
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1">// 传入的形参n可以替换为n的子类型，即f(method(n的子类)) &gt; f(method(n)
，传入形参是逆变
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1">// 返回值可以替换为method返回值的父类型 所以f（result父类）&gt;f(result)
所以方法返回值是协变
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span><span class="highlight-kd">public</span> <span class="highligh
-kd">static</span> <span class="highlight-n">Number</span> <span class="highlight-nf"
method</span><span class="highlight-o">(</span><span class="highlight-n">Number</s
an> <span class="highlight-n">num</span><span class="highlight-o">){</span>
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-k">return</span> <span class="highlight-mi">1</span><span class="highlight-o">
</span>
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
ight-o">}</span>
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
ight-n">Object</span> <span class="highlight-n">result</span><span class="highlight-o"
=</span><span class="highlight-n">method</span><span class="highlight-o">(</span><
pan class="highlight-k">new</span> <span class="highlight-n">Integer</span><span clas
="highlight-o">(</span><span class="highlight-mi">1</span><span class="highlight-o">))
</span> <span class="highlight-c1">//correct
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span><span class="highlight-n">Number</span> <span class="highlig
t-n">numResult</span><span class="highlight-o">=</span><span class="highlight-n">me
hod</span><span class="highlight-o">(</span><span class="highlight-k">new</span> <s
an class="highlight-n">Object</span><span class="highlight-o">());</span> <span class="h
ghlight-c1">//error
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span><span class="highlight-n">Integer</span> <span class="highligh
-n">result</span><span class="highlight-o">=</span><span class="highlight-n">method
/span><span class="highlight-o">(</span><span class="highlight-k">new</span> <span c
ass="highlight-n">Integer</span><span class="highlight-o">(</span><span class="highlig
t-mi">2</span><span class="highlight-o">));</span> <span class="highlight-c1">//error
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1">// 方法重写:
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1">// java1.5开始，子类覆盖父类方法时，允许返回更为具体的类型()
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1">// Sub&lt;Super ,返回值 Sub的返回值&lt; Super的返回值，协变
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1">// 方法的形参必须和父类保持一致（不变）
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-c1"></span><span class="highlight-kd">class</span> <span class="highlight
nc">Super</span><span class="highlight-o">{</span>
</span></span><span class="highlight-line"><span class="highlight-cl">    <span class="h
ghlight-kd">public</span> <span class="highlight-kd">abstract</span> <span class="highl
ght-n">Number</span> <span class="highlight-nf">method</span><span class="highlight
o">(</span><span class="highlight-n">Number</span> <span class="highlight-n">n</spa
><span class="highlight-o">)</span>
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
ight-o">}</span>
</span></span><span class="highlight-line"><span class="highlight-cl"><span class="high
ight-kd">class</span> <span class="highlight-nc">Sub</span> <span class="highlight-kd"
```

extends Super{</sp
n>
    <span class="h
ghlight-nd">@Override
    <span class="h
ghlight-kd">public Integer <span class="highli
ht-nf">method(NU
ber n){ <sp
n class="highlight-o">... }
<span class="high
ight-o">}
</code></pre>

总结

子类重写父类的方法时，形参类型必须一致，而返回值可以 返回 更具体的类型
java 的泛型必须一致，数组可以使用

