Java 核心技术(高阶)

第二章 Java泛型

第五节 Java类型的协变和逆变

华东师范大学 陈良育

类型变化关系(1)

- 面向对象语言，支持子类型(Subtyping)
- 类型变化关系(type variance)：
  - 更复杂类型中的子类型关系，与子类型之间的关系相关联。
  - Wiki：Variance refers to how subtyping between more complex types relates to subtyping between their components.

类型变化关系(2)

- 例子:
  - 假设Cat是Animal的子类，那么任何包含Cat的语句是否可被Animal语句来取代?
  - List<Cat> 是否可以被List<Animal>取代?
  - 一个返回值Cat的函数，是否可以被一个返回Animal的函数取代?
  - 一个List<Animal>中的Cat实例和Fish实例如何比较?

类型变化关系(3)

- Type Variance形式化定义:
  - A、B是类型，f(·)表示类型转换，≤表示继承关系，如A≤B，表示A继承于B
  - f(·)是协变(covariant)的，如果A ≤ B，有f(A) ≤f(B)
  - f(·)是逆变(contravariant)的，如果A ≤ B，有f(B) ≤f(A)
  - f(·)是不变(invariant)的，当上述两种都不成立，即f(A)和f(B)没有关系
  - f(·)是双变(bivariant)的，如果A ≤ B，有f(B) ≤f(A) 和f(A) ≤f(B)同时成立

Java数据类型变化(1)

- Java数组是协变的
  - String是Object的子类，String[]是Object[]的子类

```
class A{}                   //第一代
class B extends A{}         //第二代
class C extends B{}         //第三代
```

```
B[] array1 = new B[1];
array1[0] = new B();

A[] array2 = array1;

try {
    array2[0] = new A();
    // compile ok, runtime error
} catch (Exception ex) {
    ex.printStackTrace();
}

try {
    array2[0] = new C();
    // compile ok, runtime ok
} catch (Exception ex) {
    ex.printStackTrace();
}
```

Java数据类型变化(2)

- Java的(原始的)泛型是不变的
  - String是Object的子类，List<String>和List<Object>没有关系

```
class A{}                  //第一代
class B extends A{}        //第二代
class C extends B{}        //第三代
```

```
ArrayList<B> list1 = new ArrayList<B>();
list1.add(new B());

ArrayList<A> list2 = list1;
//compile error
```

- 泛型可采用通配符，支持协变和逆变(PECS原则)
  - ArrayList<? extends A> list3 = new ArrayList<B>(); //协变
  - ArrayList<? super B> list4 = new ArrayList<A>(); //逆变

Java数据类型变化(3)

- 复合情况
  - 数组协变，泛型不变

```
class A{}                    //第一代
class B extends A{}          //第二代
class C extends B{}          //第三代
```

```
public static void testArrayAndList()
{
    B[] r1 = test(new ArrayList<B>()); //compile error
    A[] r2 = test(new ArrayList<B>()); //compile error
    Object[] r3 = test(new ArrayList<Object>()); //compile error

    A[] r4 = test(new ArrayList<A>());
    Object[] r5 = test(new ArrayList<A>());
}

public static A[] test(ArrayList<A> list)
{
    return new A[1];
}
```

Java数据类型变化(4)

- 方法情况
  - JDK 1.4 重写的方法参数和返回值要求一样
  - JDK 1.5 +重写的方法，参数要求一样的，返回值是协变的

```
class Father
{
    public B f1(B obj)
    {
        System.out.println("Father.f1()");
        return new B();
    }
}

class Son extends Father
{
    public B f1(B obj)
    //public C f1(B obj)    //返回值是C，也是对的
    {
        System.out.println("Son.f1()");
        return new C();
    }
}
```

```
class A{}                   //第一代
class B extends A{}         //第二代
class C extends B{}         //第三代
```

```
Father foo = new Son();
foo.f1(new B());
```

总结

- Java类型变化(type variance)
  - 数组是协变的(covariant)
  - 原始的泛型是不变的(invariant)
  - <? extends A> 支持协变(covariant)
  - <? super B>支持逆变(contravariant)

谢 谢！