**类型通配符**

1、类型通配符一般是使用 ? 代替具体的类型参数。例如 List<?> 在逻辑上是 List<String>,List<Integer> 等所有 List<具体类型实参> 的父类。

实例

import java.util.\*;

public class GenericTest {

public static void main(String[] args) {

List<String> name = new ArrayList<String>();

List<Integer> age = new ArrayList<Integer>();

List<Number> number = new ArrayList<Number>();

name.add("icon");

age.add(18);

number.add(314);

getData(name);

getData(age);

getData(number);

}

public static void getData(List<?> data) {

System.out.println("data :" + data.get(0));

}

}

输出结果为：

data :icon

data :18

data :314

解析： 因为 getData() 方法的参数是 List<?> 类型的，所以 name，age，number 都可以作为这个方法的实参，这就是通配符的作用。

2、类型通配符上限通过形如List来定义，如此定义就是通配符泛型值接受Number及其下层子类类型。

实例

import java.util.\*;

public class GenericTest {

public static void main(String[] args) {

List<String> name = new ArrayList<String>();

List<Integer> age = new ArrayList<Integer>();

List<Number> number = new ArrayList<Number>();

name.add("icon");

age.add(18);

number.add(314);

//getUperNumber(name);//1

getUperNumber(age);//2

getUperNumber(number);//3

}

public static void getData(List<?> data) {

System.out.println("data :" + data.get(0));

}

public static void getUperNumber(List<? extends Number> data) {

System.out.println("data :" + data.get(0));

}

}

输出结果：

data :18

data :314

解析： 在 //1 处会出现错误，因为 getUperNumber() 方法中的参数已经限定了参数泛型上限为 Number，所以泛型为 String 是不在这个范围之内，所以会报错。

3、类型通配符下限通过形如 List<? super Number> 来定义，表示类型只能接受 Number 及其上层父类类型，如 Object 类型的实例。