* **通配符**
* 使用**类型通配符：？**

比如：List<?>，Map<?,?>。

List<?>是List<String>、List<Object>等各种泛型List的父类。

* **读取**List<?>的对象list中的元素时，永远是安全的，因为不管list的真实类型是什么，它包含的都是Object。
* **写入**list中的元素时，不行。因为我们不知道c的元素类型，我们不能向其中添加对象。
  + 唯一的例外是null，它是所有类型的成员。
* 将任意元素加入到其中不是类型安全的：

Collection<?> c = new ArrayList<String>();

c.add(new Object());//编译时错误

* **因为我们不知道c的元素类型，我们不能向其中添加对象**。
* add方法有类型参数E作为集合的元素类型。我们传给add的任何参数都必须是一个未知类型的子类。因为我们不知道那是什么类型，所以我们无法传任何东西进去。
* 唯一的例外的是null，它是所有类型的成员。
* 另一方面，我们可以调用get()方法并使用其返回值。返回值是一个未知的类型，但是我们知道，它总是一个Object。
* 有限制的通配符

<?> 允许所有泛型的引用调用

* 举例：
* <? extends Number>(无穷小,Number]

只允许泛型为Number及Number子类的引用调用。

* <? super Number>[Number,无穷大)

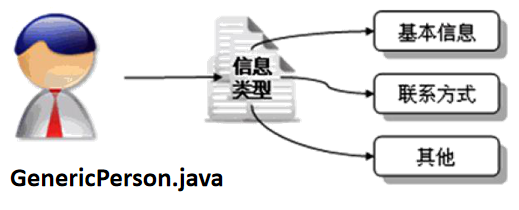
只允许泛型为Number及Number父类的引用调用。

* <? extends Comparable>

只允许泛型为实现Comparable接口的实现类的引用调用。

* 泛型应用

用户在设计类的时候往往会使用类的关联关系，例如，一个人中可以定义一个信息的属性，但是一个人可能有各种各样的信息（如联系方式、基本信息等），所以此信息属性的类型就可以通过泛型进行声明，然后只要设计相应的信息类即可。



/\*\*  
 \* 通配符 ?  
 \* List<A>、List<B>....都是List<?>的子类  
 \* <p>  
 \* ? extends A ：可以存放A及其子类  
 \* ? super A ：可以存放A及其父类  
 \*/  
 public void test3() {  
 List<?> list = null;  
 List<Object> list1 = new ArrayList<>();  
 List<String> list2 = new ArrayList<>();  
 list = list1;  
 list = list2;  
  
 show(list1);  
// show(list2);  
 show1(list1);  
 show1(list2);  
  
 List<? extends Number> list3 = null;  
 List<Integer> list4 = null;  
 list3 = list4;  
// list3 = list1;  
 List<? super Number> list5 = null;  
 list5 = list1;  
 }  
  
 public void show(List<Object> list) {  
  
 }  
  
 public void show1(List<?> list) {  
  
 }