泛型程序设计意味着编写的代码可以被很多不同类型的对象所重用。

ArrayList

类型变量使用大写形式

使用E表示集合的元素类型，K和V分别表示表的关键字和值的类型。

T（需要时还可以用临近的字母U和S）表示任意类型

泛型方法

public static <T> T getXXX(){

}

类型变量放在修饰符的后面，返回类型的前面

变量类型的限定

<T extends Comparable> T

一个类型变量或通配符可以有多个限定

T extends Comparable & Serializable

限定类型用&分隔，逗号用来分隔类型变量

在Java的继承中，可以根据需要拥有的多个接口超类型，但限定中至多有一个类，如果用一个类作为限定，它必须是限定列表中的第一个。

泛型代码和虚拟机

无论何时定义一个泛型类型，都自动提供一个相应的原始类型(raw type)。原始类型的名字就是删除类型参数后的泛型类型名。擦除（erased）类型变量，并替换为限定类型（无限定的变量用Object）

泛型转换

Java虚拟机中没有泛型，只有普通的类和方法

所有的类型参数都用它们的限定类型替换

桥方法被合成来保持多态

为保持类型安全性，必要时插入强制类型转换

约束与局限性

不能用基本类型实例化类型参数

运行时类型查询只适用于原始类型

不能创建参数化类型的数组

Varargs警告

不能实例化类型变量

泛型类的静态上下文中类型变量无效

不能抛出或捕获泛型类的实例

注意擦除后的冲突

泛型类型的继承规则

8通配符类型

<? extends xxxclass>

8.1通配符的超类型限定

? super xxxclass

带有超类型限定的通配符可以向泛型对象写入，带有子类型限定的通配符可以从泛型对象读取。

8.2无限定通配符

<?>