1. 泛型的学习网站：

<https://www.bilibili.com/video/BV1xJ411n77R?from=search&seid=13800166643041649361>

1. 学习笔记：
2. 泛型产生的背景

Java中常用的集合，没有指定类型的，一般都是存储的Object。相当于什么类型都可以进行存储。

但是当遇到大型项目，存储的数据类型多种多样时，不可能为每一个数据的输出输入都指定一个类型，所以就产生了泛型的概念。

定义：Java泛型是JDK5中引入的一个新特性，泛型提供了编译时的类型安全监测机制，该机制允许我们在编译时监测到非法的类型数据结构。

泛型的本质就是参数化类型，也就是所操作的数据类型被指定为一个参数。

好处：

1. 编译期间检查类型（类型安全）
2. 减少了类型转换（消除了强制类型的转换）
3. 泛型类、泛型接口

class 类名称<泛型标识，泛型标识， ...> {

public 泛型标识 变量名1;

protected 泛型标识 变量名2;

private 泛型标识 变量名3;

}

常用的泛型标识：T 、E 、K 、V

1. 泛型类

使用方法：

类型<具体的数据类型> 对象名 = new 类名<具体的数据类型>();

Java1.7后，后面的<>中的数据类型可以不写

类型<具体的数据类型> 对象名 = new 类名<>();

注意事项：

如果没有指定具体的数据类型，此时，操作类型是Object

泛型的类型参数只能是类类型，不能是基本的数据类型

泛型类型在逻辑上可以看成是多个不同的类型，但实际上都是相同类型

1. 从泛型类派生子类
2. 子类也是泛型类，子类和父类的泛型类型要一致

Class ChildGeneric<T> extends Generic<T>

1. 子类不是泛型类，父类要明确泛型的数据类型

Class ChildGeneric extends Generic<String>

1. 泛型类、接口

泛型接口的使用

1. 实现类不是泛型类，接口要明确数据类型
2. 实现类也是泛型类，实现类和接口的泛型类型要一致
3. 泛型方法

泛型类，是在实例化类的时候至明年泛型的具体类型

泛型方法，是在调用方法的时候指明泛型的具体类型

语法：

修饰符 <T,E,K ...> 返回值类型 方法名（形参列表） {

方法体...

}

泛型列表非常重要，可以理解为声明此方法为泛型方法

只有声明了<T>（泛型标识符）的方法才是泛型方式，泛型类中的 使用了泛型的成员的方法并不是泛型方法。

<T>表明该方法将使用泛型类型T，此时才可以在方法中使用泛型类型T

与泛型类的定义一样，此处T可以随便写为任意标识，常见的如T、E、K、V等形式的参数常用于表示泛型

泛型方法与可变参数

Public <E> void xxxx(E... e) {

For(E el:e){

Log.v(“xxx”, String.valueOf(e));

}

}

泛型方法总结：

泛型方法能使方法独立于类尔产生变化

如果static方法要使用泛型能力，就必须使其定义为泛型方法

1. 类型通配符

类型通配符一般是使用 “？” 代替具体的类型实参

所以，类型通配符是类型实参，而不是类型形参

1. 类型通配符的上限

语法

类/接口 <? extends 实参类型>

要求该泛型的类型，只能是实参实现，或者是实参类型的 子类型 实现

1. 类型通配符的下限

语法

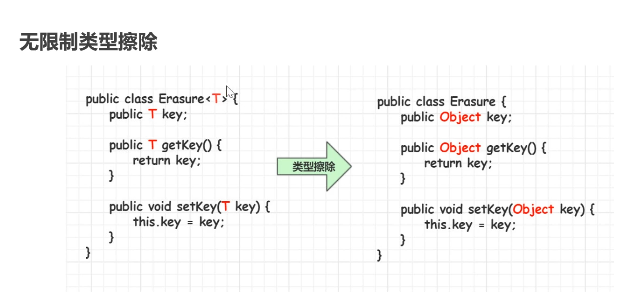
类/接口<? super 实参类型>

要求该泛型的类型，只能是实参实现，或者是实参类型的 父类类型 实现

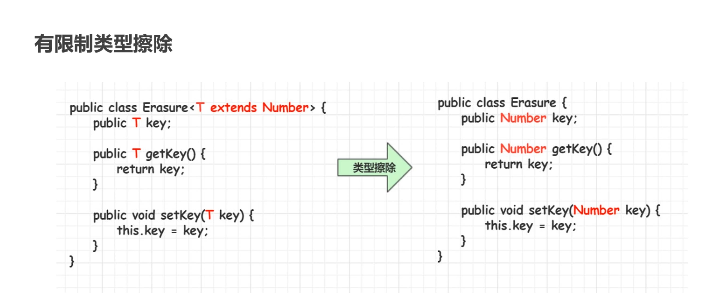
1. 类型擦除

泛型是java1.5版本才引进的概念，在这之前是没有泛型的。但是，泛型代码却能很好地与之前版本的代码兼容。那是因为，泛型信息只存在于编译阶段，在进入JVM之前，与泛型相关的信息会被擦除掉，我们称之为 —— 类型擦除。

两种类型擦除的方式

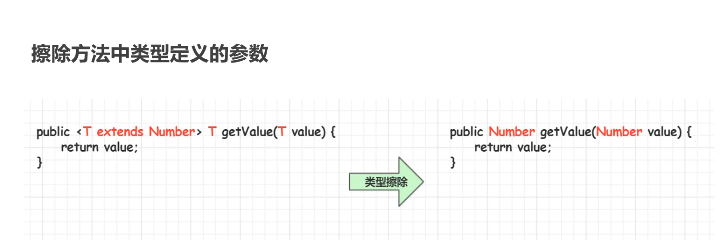


在编译的过程中，泛型标识会被Object去替换掉



在编译的过程中，泛型标识会被Number去替换掉

不仅在类中可以存在类型擦除，在类的方法中也会存在



桥接方法：

