**泛型**

## 泛型定义

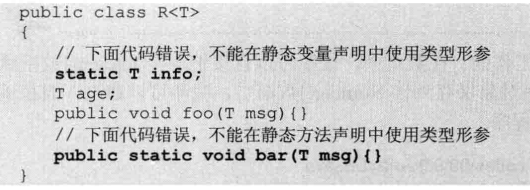
泛型 主要用于动态指定 变量的类型，既叫**参数化类型 也可叫 类型参数 或 类型实参。**

## 泛型使用范围

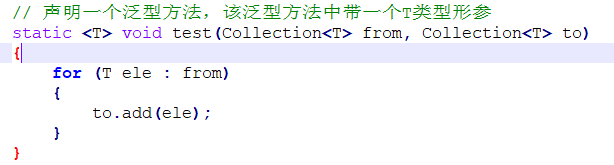
泛型 可用于在 类 接口 方法 变量 参数中。

## 静态方法中泛型使用

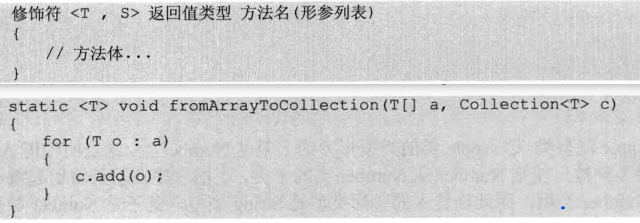
**1.泛型类 不能用在静态方法和静态声明中。**



**2.泛型方法 可用于静态方法中使用，泛型类不可用于静态方法。**



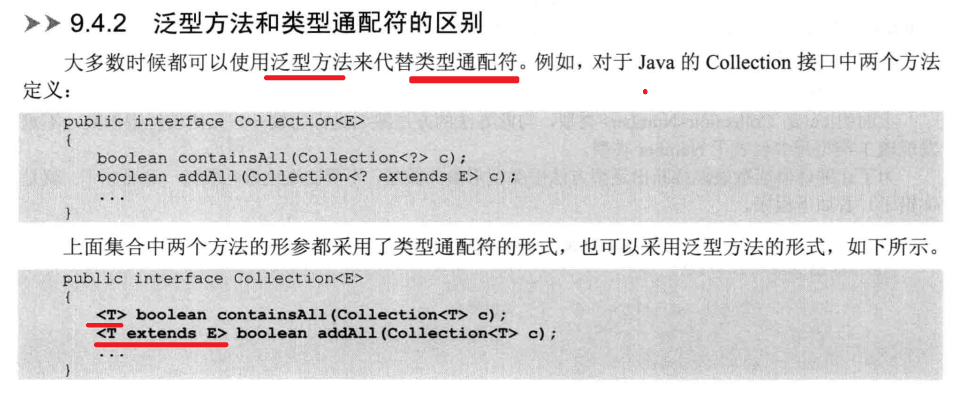
**方法定义**



## 通配符

泛型通配符是一个问号(?)代表泛型的父类。例如：List<?> 相当于 List<objcet>.

## 泛型方法与通配符区别



## 泛型的范围

**<? extend Object>** 此泛型表示任意一个Object子类 类 <= Object

适用于 单个参数 指定 该参数 类型范围

**<? super Integer>** 此泛型表示任意一个Integer父类 类 >= Integer

适用于多参数，当多个参数有依赖关系时候， 指定同意类型参数。

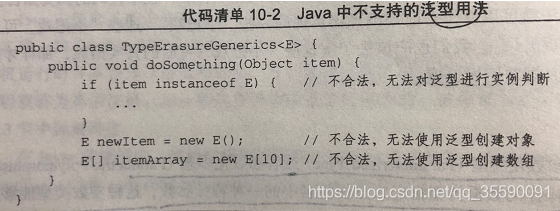
## 类型擦除的缺点

在别的语言里面，比如C#,也有实现泛型，但是实现的原理不一样，它使用的方式是：“具现化”，简单来说就是在C#里面，无论在源代码，还是在编译和，亦或是在运行期间，这些泛型的类型都是实际存在的，也就是说List<int>和List<long>他就是两个不同的类型。

而我们之前说过，java里面实现的方式是“类型擦除”，那么这种实现方式有哪些缺点呢？

1、**使用类型擦除直接导致了对于原始的数据类型无法支持，比如int，long这种**，因为java不支持Object类型和基本数据类型之间的强制类型转换，也就是说一旦类型擦除之后，就没法在进行 类型转换了。也正是这样，现在的泛型都是不支持原始类型的，比如ArrayList<Integer>，而不能使用ArrayList<int>。

2、**运行期间无法获得泛型类型信息。因为泛型都被擦除了**，都被替换成了裸类型。这样就导致了下面的程序都会报错，比如无法使用泛型来创建对象，或者数组。



原文链接：<https://blog.csdn.net/qq_35590091/article/details/107177889>

## 总结

1. T表示一个具体的类型。?表示多个类型(也就是任意一个类型)，是T的父类。
2. 泛型中通配符(?)表示的是数据范围，因此不能在泛型类的定义中。【**因为泛型在类中定义需要指定确定的类型信息**】。只能用于形参中。

形参定义->是指在定义函数时使用的参数.即方法的参数都是形参。

1. 而且通配符定义的类型只能查询数据，不能添加数据（**通配符指的就是 ?**）。

错误案例

