Java泛型是JDK 5引入的一个特性，它允许我们定义类和接口的时候使用参数类型，泛型在集合框架中被广泛使用。类型擦除是泛型中最让人困惑的部分，本篇文章将阐明什么是类型擦除，以及如何使用它。

**一个常见错误**

[复制代码](javascript:void(0);)

package simplejava;

import java.util.ArrayList;

public class Q29 {

public static void main(String[] args) {

ArrayList<String> al = new ArrayList<String>();

al.add("a");

al.add("b");

accept(al);

}

public static void accept(ArrayList<Object> al) {

for (Object o : al)

System.out.println(o);

}

}

[复制代码](javascript:void(0);)

以上代码看起来是没问题的，因为String是Object的子类。然而，这并不会工作，编译不会通过，并提示如下错误：  
The method accept(ArrayList<Object>) in the type Q29 is not applicable for the arguments (ArrayList<String>)

**List<Object>和List<String>**

原因在于类型擦除。记住：Java的泛型机制是在编译级别实现的。编译器生成的字节码在运行期间并不包含泛型的类型信息。

在编译之后，List<Object>和List<String>将变成List，Object和String类型信息对于JVM来说是不可见的。在编译阶段，编译器发现它们不一致，因此给出了一个编译错误。

**通配符和有界通配符**

List<? >表示List能包含任何类型的元素

[复制代码](javascript:void(0);)

public static void main(String args[]) {

ArrayList<Object> al = new ArrayList<Object>();

al.add("abc");

test(al);

}

public static void test(ArrayList<?> al) {

for (Object e : al) {// no matter what type, it will be Object

System.out.println(e);

// in this method, because we don’t know what type ? is, we can not

// add anything to al.

}

}

[复制代码](javascript:void(0);)

永远记住，泛型是一个编译时的概念。在这个例子中，由于我们不知道？，我们不能添加任何元素到al集合。如果想要添加的话，可以使用通配符。

List< Object > - List can contain Object or it’s subtype

List< ? extends Number > - List can contain Number or its subtypes.

List< ? super Number > - List can contain Number or its supertypes.

**与数组比较**

现在，我们知道了ArrayList <String >并不是ArrayList <Object >的子类型，不过，你需要知道如果两个泛型类型有相同的参数，它们的继承关系是依据其类型。如ArrayList<String>是Collecton<String>的子类型。

然而，数组却不同，它们在运行期间知道每个元素的类型且强制它们的元素为该类型，这叫reification。举个例子，Object[] objArray是String[] strArr的超类型。如果你尝试往存储整型的数组添加字符串对象，将会在运行期间得到一个ArrayStoreException异常。