# 泛型

## 泛型的概述

泛型:是JDK5中引入的特性,他提供编译时类型安全机制.

泛型的好处:

把运行时期的问题提前到了编译时期;

避免了强制类型转换.

## 泛型的使用场景

类后面: 泛型类

方法申明上: 泛型方法

接口后面: 泛型接口

## 泛型的定义格式

<类型> 指定一种类型的格式

尖括号里面可以任意书写,按照的变量定义即可,一般只写一个字母

常用的有,E,T,Q,M,K,V等

<类型1,类型2> 指定多种类型的格式,中间用逗号隔开

例如:<E, T> <Q, M> <K, V>

泛型类的定义格式:

public class 类名<E> {} 不一定是E,还可以是其他

泛型方法的定义格式:

修饰符<类型> 返回值类型 方法名(类型 变量/对象名){}

如:

public static <E> void method(E e){}

泛型接口的定义格式:

interface 接口名<E>{}

实现接口的类或者子接口,要么也是泛型类/接口,要么确定具体的数据类型

**示例:**

//泛型接口的泛型实现类,在类名后面需要声明我是泛型类,后面才可以使用泛型,否则报错.和泛型方法类似

class interImpl1<E> implements inter<E> {

@Override

public void show(E e) {

System.out.println(e);

}

}

## 泛型的通配符

类型通配符: <?>

如:

ArrayList<?> 表示数据类型未知的ArrayList,它的元素可以匹配所有类型

但是并不能把元素添加到ArrayList中了,获取出来的也是父类类型

类型通配符上下限: **ArrayList extends Number extends Object**

类型通配符上限: <?extends 类型>

比如: ArrayList <? extends Number> : 它表示的类型时Number或者他的子类

类型通配符下限:<?super 类型>

比如: ArrayList <? extends Number> : 它表示的类型时Number或者他的父类