## 泛型应用:

格式

1. 集合类<类类型> 变量名 = new 集合类<类类型>();

优点

1. 将运行时的异常提前至编译时发生。

2. 获取元素的时候无需强转类型，就避免了类型转换的异常问题

使用：

格式 通过<> 来指定容器中元素的类型.

什么时候使用泛型:当类中操作的引用数据类型不确定的时候,就可以使用泛型类.

## 泛型的定义

##### **1. 函数上的泛型定义**

声明的位置在static后返回值类型前。静态方法也可以。

当函数中使用了一个不明确的数据类型，那么在函数上就可以进行泛型的定义。

public <泛型的声明> 返回值类型 函数名( 泛型 变量名 ){

}

##### **2. 类上的泛型声明**

修饰符 class 类名<泛型>{

}

静态方法不可以使用类中定义的泛型，因为类中的泛型需要在对象初始化时指定具体的类型，而静态优先于对象存在。那么类中的静态方法就需要单独进行泛型声明，声明泛型一定要写在static后，返回值类型之前

泛型类细节:

1、创建对象的时候要指定泛型的具体类型

2、创建对象时可以不指定泛型的具体类型(和创建集合对象一眼)。默认是Object，例如我们使用集合存储元素的时候没有使用泛型就是那么参数的类型就是Object

3、类上面声明的泛型只能应用于非静态成员函数，如果静态函数需要使用泛型，那么

需要在函数上独立声明。

4、如果建立对象后指定了泛型的具体类型，那么该对象操作方法时，这些方法只能操作一种数据类型。

5、所以既可以在类上的泛型声明，也可以在同时在该类的方法中声明泛型。

##### **3. 泛型接口**

和泛型类相类似

##### 4. 泛型通配符

<? Extends E>？extends E: 接收E类型或者E的子类型。

<? Super E >? super E: 接收E类型或者E的父类型。

泛型是提供给javac编译器使用的，它用于限定集合的输入类型，让编译器在源代码级别上，即挡住向集合中插入非法数据。但编译器编译完带有泛形的java程序后，生成的class文件中将不再带有泛形信息，以此使程序运行效率不受到影响，这个过程称之为“擦除”。

泛型的基本术语，以ArrayList<E>为例：

<>念着typeof

ArrayList<E>中的E称为类型参数变量

ArrayList<Integer>中的Integer称为实际类型参数

整个称为ArrayList<E>泛型类型

整个ArrayList<Integer>称为参数化的类型ParameterizedType