泛型

概述

泛型，即“参数化类型”。就是将类型由原来的具体的类型参数化，类似于方法中的变量参数，此时类型也定义成参数形式（可以称之为类型形参），然后在使用/调用时传入具体的类型（类型实参）。

使用

泛型类

```
定义一个泛型类：
public class ClassName<T>{
    private T data;

    public T getData() {
        return data;
    }

    public void setData(T data) {
        this.data = data;
    }

}
```

泛型接口

```
public interface IntercaceName<T>{
    T getData();
}

实现接口时，可以选择指定泛型类型，也可以选择不指定， 如下：

指定类型：
    public class Interface1 implements IntercaceName<String> {
        private String text;
        @Override
        public String getData() {
            return text;
        }
    }
不指定类型：
    public class Interface1<T> implements IntercaceName<T> {
        private T data;
        @Override
        public T getData() {
            return data;
        }
    }
```

泛型方法

```
    private static <T> T 方法名(T a, T b) {}
```

泛型限制类型

```
1.  在使用泛型时，  可以指定泛型的限定区域 ，
        -    例如：  必须是某某类的子类或  某某接口的实现类，格式：
             <T extends 类或接口1 & 接口2>
```

泛型中的通配符 ？

```
类型通配符是使用？代替方法具体的类型实参。

1   <? extends Parent> 指定了泛型类型的上届
2   <? super Child> 指定了泛型类型的下届
3   <?> 指定了没有限制的泛型类型
```

作用

```
1、 提高代码复用率
2、 泛型中的类型在使用时指定，不需要强制类型转换（类型安全，编译器会检查类型）
```

注意

```
在编译之后程序会采取去泛型化的措施。
也就是说Java中的泛型，只在编译阶段有效。
在编译过程中，正确检验泛型结果后，会将泛型的相关信息擦出，并且在对象进入和离开方法的边界处添加类型检查和类型转换的方法。也就是说，泛型信息不会进入到运行时阶段。
```