今日完成任务3

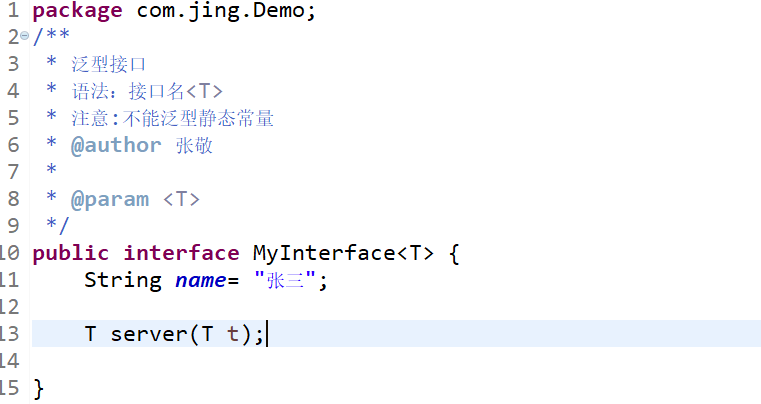
1. 泛型：参数化类型，把类型作为参数传递。

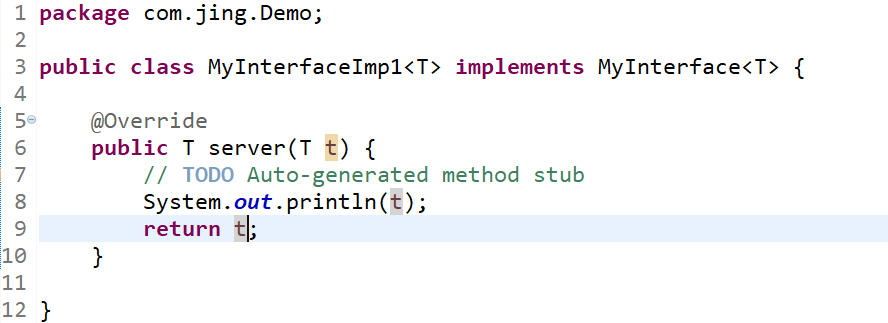
语法：<T,...>T为类型占位符，表示引用类型

①泛型类



②泛型接口





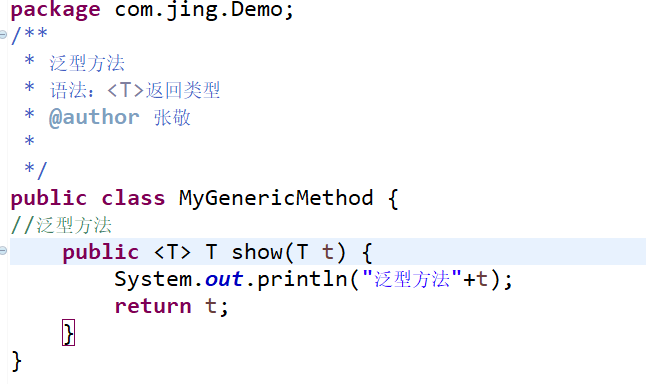




不需要传递数据类型，类型由传递的数据决定



③泛型方法



优点：1.提高代码的重用性

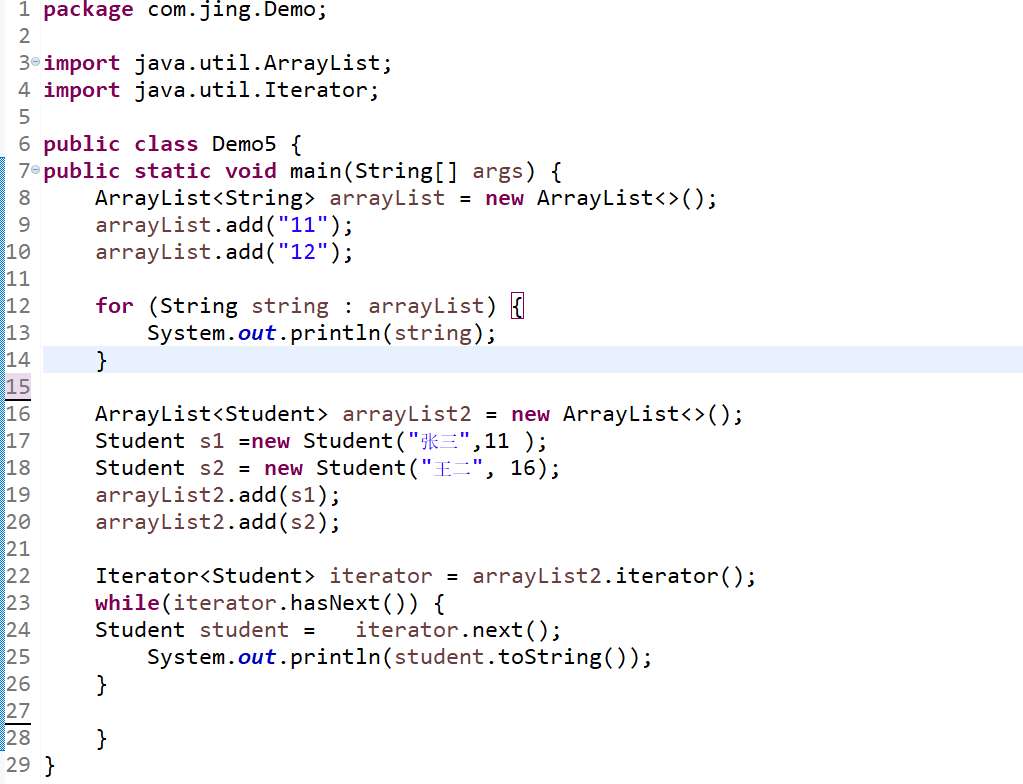
2.防止类型转换异常，提高代码的安全性

泛型集合：参数化类型、类型安全的集合，强制集合元素的类型必须一致。

特点：1.编译时即可检查，而非运行时抛出异常。

2.访问时，不必类型转换（拆箱）。

3.不同泛型之间引用不能相互赋值，泛型不存在多态。



1. 异常：程序在运行过程中出现的不正常的现象。出现异常不处理将终止程序的运行。

Throwable:可抛出，一切错误或异常的父类，位于java.lang包中。

|-Error :JVM、硬件、执行逻辑错误，不能手动处理。

|-StackOverflowError |-OutOfMemoryError

|-Exception:程序在运行和配置中产生的问题，可处理

|-RuntimeException:运行时异常，可处理，可不处理

|-CheckedException:检查时异常，必须处理

常见运行时异常

|  |  |
| --- | --- |
| 类型 | 说明 |
| NullPointException | 空指针异常 |
| ArrayIndexOutOfBoundsExcption | 数组越界异常 |
| ClassCastException | 类型转换异常 |
| NumberFormatException | 数字格式化异常 |
| ArithmeticException | 算数异常 |

异常的产生：当程序在运行时遇到不符合规范的代码或结果时，会产生异常或程序员使用throw关键字手动抛出异常。

异常的传递：按照方法的调用链反向传递，如始终没有处理异常，最终由JVM进行默认异常处理（打印堆栈跟踪信息）

异常处理

java的异常处理是通过5个关键字来实现的

1. try:执行可能产生的异常代码
2. catch:捕捉异常，并处理
3. finally:无论是否发生异常，代码总能执行
4. throw：手动抛出异常
5. throws:声明方法可能要抛出的各种异常。

异常处理try...catch

try{

}catch(Exception e){}

三种情况：

1. 正常请求 2.出现异常并处理 3.异常类型的不匹配

异常处理try...catch...finally

try{

可能出现异常的代码

}catch(Exception e){

异常处理相关代码，如：getMessage()、printStackTrace()

}finally{

是否发生异常都会执行，可以释放资源等。。。

}

1.finally块 是否发生异常都会执行，可以释放资源

2.finally块不执行的唯一情况，退出java虚拟机

异常处理 多重catch

try{

可能出现异常的代码

}catch(异常类型1){满足异常类型1执行的相关代码

}catch(异常类型2){满足异常类型2执行的相关代码

}catch(异常类型3){满足异常类型3执行的相关代码

}

1. 子类异常在前，父类异常在后
2. 发生异常时按顺序逐个匹配
3. 只执行第一个与异常类型匹配的catch语句
4. finally根据需要可写或不写

异常处理try...finally

try...finally不能捕获异常，仅仅用来当发生异常时，用于释放资源

一般用在底层代码，只释放资源不做异常处理，把异常向上抛出