**认识J2SE**

**一、实验简介**

在本章学习开始前，你应该具备一些[Java的基础知识](https://www.shiyanlou.com/courses/18)。我们将在本章来认识J2SE，并复习一下前面学过的面向对象的相关知识。

**注：所有的蓝色文字都是带超链接的，这些链接是本课程为你提供的扩展知识点，一定不要错过。**

本章知识点

* J2SE的定义和架构
* 面向对象

**二、什么是J2SE**

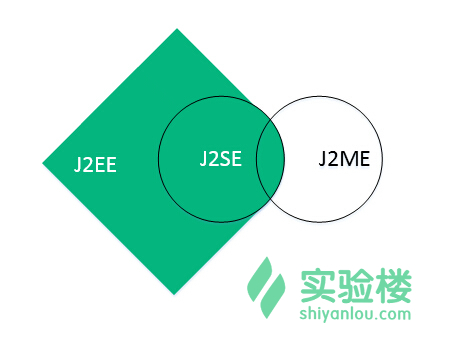
**1. J2SE的定义**

J2SE，全称为Java 2 Standard Edition。Java 2平台包括：标准版（J2SE）、企业版（J2EE）和微缩版（J2ME）三个版本。J2SE即Java 2的标准版，主要用于桌面应用软件的开发。

下面这段话是ORACLE对于Java SE的官方描述：

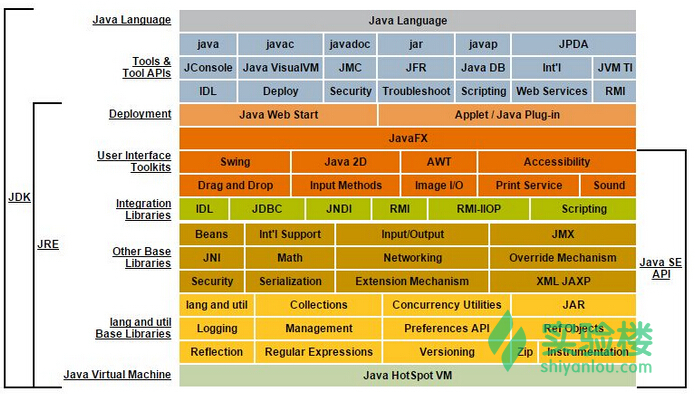
Java Platform, Standard Edition (Java SE) 可以让您在桌面和服务器以及目前要求较高的嵌入式环境中开发和部署 Java 应用程序。 Java 提供了当今应用程序所需要的丰富的用户界面、良好的性能、多功能性、可移植性和安全性。

J2SE与J2EE、J2ME之间的关系可以通过下图来表示：



**2. J2SE的架构**

J2SE的架构如下图所示，它主要包含了UI、集成库、语言和工具基础库、其他基础库、Java虚拟机等组件。



综上所述，将J2SE压缩一点再加上一些[CLDC](http://baike.baidu.com/link?url=YyhiiGfjgAR-CzCLq5nHUKwDvAKRCTC-JDmf_uAcYyqJerKQaB36sLlFNzoO9JtrutrOH1UnbAZjs7lrcmJKRK)等方面的特性就是J2ME；将其扩充一点再增加一些EJB等企业应用方面的特性就是J2EE。因此J2SE是J2EE的基础，建议从事Java的开发人员从J2SE开始学习。

**三、面向对象——巩固与提高**

在前面Java基础语法的学习中，你应该接触了一些面向对象的基础知识。面向对象的概念在Java的开发体系中无处不在，在今后的开发过程中，你也应该以面向对象的思想来看待和解决问题。

既然是面向对象，就会始终谈论到类和对象的概念，以及它们之间的关系：

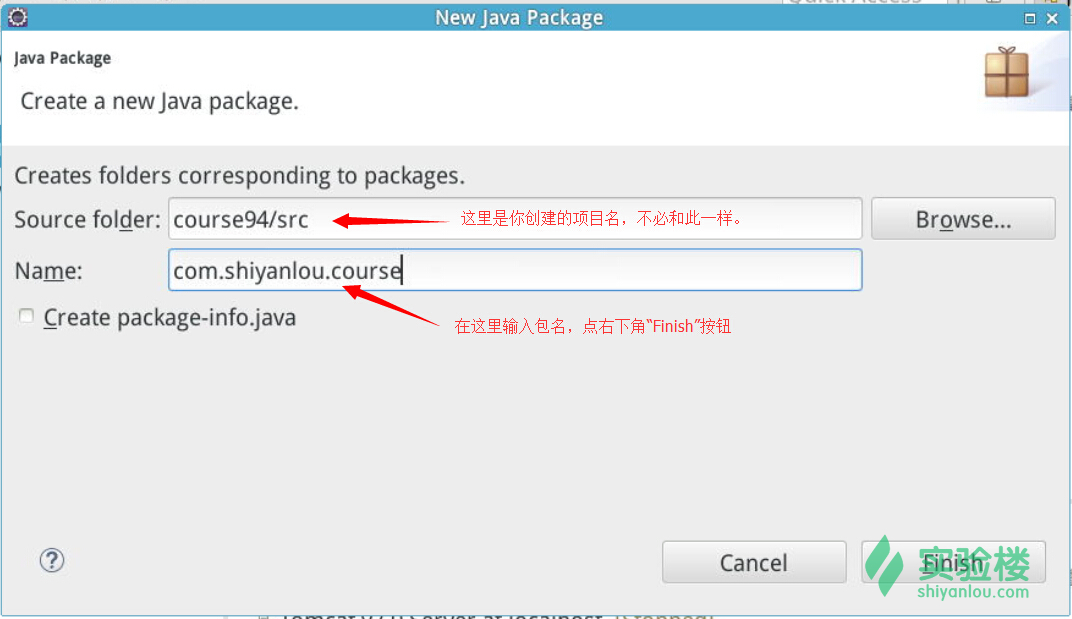
类是现实世界或思维世界中的实体在计算机中的反映，它将数据以及这些数据上的操作封装在一起。对象是具有类类型的变量。

类是对象的抽象，而对象是类的具体实例。类是抽象的，不占用内存，而对象是具体的，占用存储空间。类是用于创建对象的蓝图，它是一个定义包括在特定类型的对象中的方法和变量的软件模板。

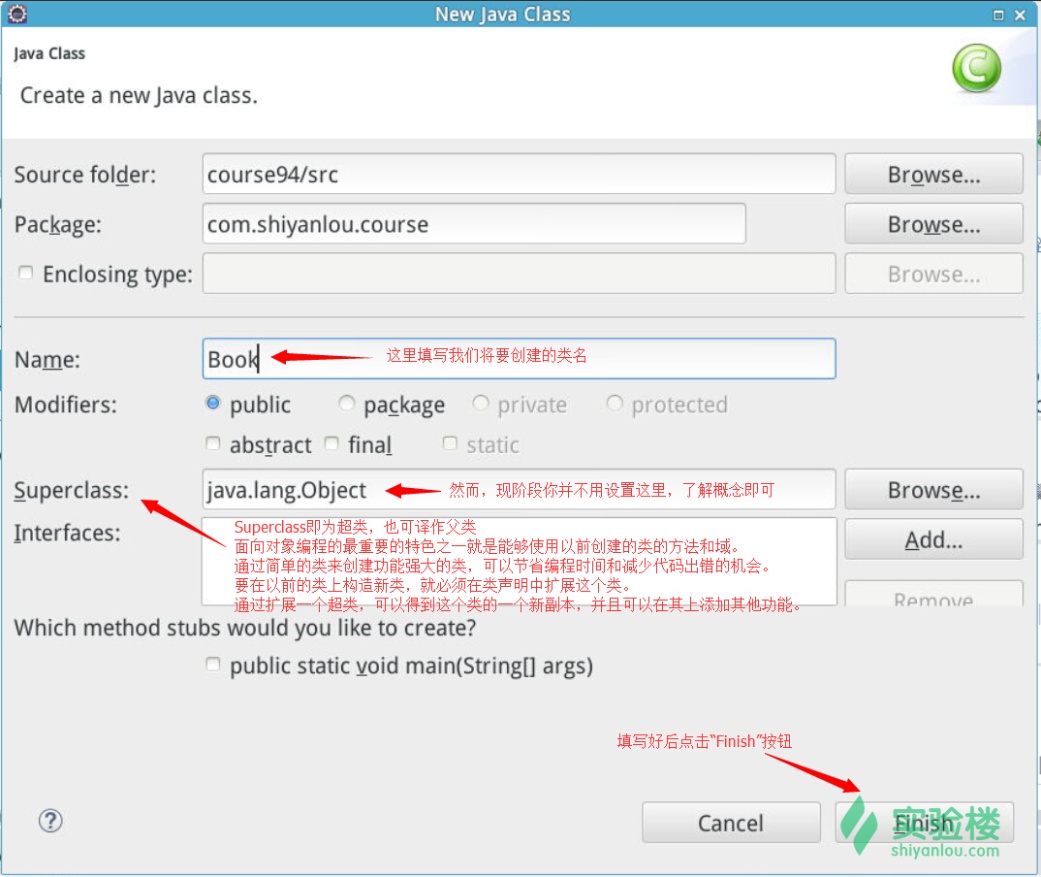
下面，我们通过一些例子来巩固面向对象的相关知识。

**1. 自定义图书类**

首先在Eclipse中创建一个项目HelloJ2SE，新建一个名为com.shiyanlou.course的包。



然后在这个包里面添加一个Book类。



对于一个图书对象，一般有书名、作者和编码等属性。我们利用这些常见属性来自定义这个图书类，创建的Book.java文件中主要代码如下：

**特别说明：在实验楼的实验环境中，暂不支持输入中文，因此你不必将代码片段中的注释也输入开发环境。该说明同样适用于本课程后续的章节。**

package com.shiyanlou.course;

//此为包名，如果你在创建包时已自动生成了该行，请忽略

public class Book {

private String name; // 定义书名

private String author; // 定义作者

private String ISBN; // 定义编码

//Tips: ISBN是国际标准书号，每本书背面的条形码即为此物

public Book(String name, String author, String ISBN) {

// 利用构造方法初始化域

this.name = name;

this.author = author;

this.ISBN = ISBN;

//Q:你清楚this的用法吗？

}

public String getName() { // 获得书名

return name;

}

public String getAuthor() { // 获得作者

return author;

}

public String getISBN() { // 获得编码

return ISBN;

}

}

为了测试我们刚刚自定义的图书类，我们在com.shiyanlou.course这个包中再创建一个名为Test的类文件，并在main()方法中创建一个Book对象。最后，我们在控制台输出这个Book对象的属性。

创建的Test.java文件中，主要的代码如下：

package com.shiyanlou.course;

public class Test {

public static void main(String[] args) {

Book book = new Book("This is my 1st book", "Peter", "1234567890");

// 创建Book对象并设定其各个属性

System.out.println("Book Name: " + book.getName());

// 输出这本书的名字

System.out.println("Book Author: " + book.getAuthor());

// 输出这本书的作者

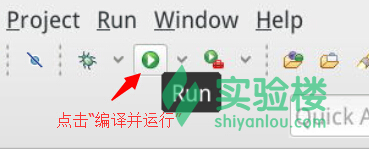
System.out.println("ISBN: " + book.getISBN());

// 输出这本书的编码

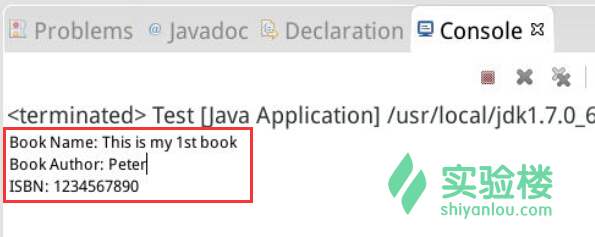
}

}

点击编译并运行。



如果在控制台中可以看到下图这样的信息，那么祝贺你，一个自定义的图书类就成功完成了。



当然，你也可以为Book对象添加更多的属性，并为每个属性编写get方法。

创建工程🡪创建包名🡪创建类:私有属性（成员变量）添加getter和setter

**2. 动态化实例类**

在面向对象的编程中，把用类创建对象的过程称为实例化。通常是使用有参数或无参数的构造方法来创建对象。其格式如下：

//有参数的情形

类名 对象名 = new 类名(参数1，参数2...参数n);

//例如下面这个例子

Person peter = new Person("Peter","boy");

//无参数的情形

类名 对象名 = new 类名();

//例如下面这个例子

Dog dog\_1 = new Dog();

你可以仿照自定义图书类的例子，通过自己创建相关的Person类和Dog类以及测试的方法来验证上述过程。

但是在Java中，类的实例化方法一共有四种途径：

1. 使用new操作符
2. 调用Class对象的newInstance()方法
3. 调用clone()方法，对现有实例的拷贝
4. 通过ObjectInputStream的readObject()方法反序列化类

我们在自定义图书类这个例子中，编写了有参数的构造方法public Book(String name, String author, String ISBN)。但如果类中没有定义构造方法，编译器便会自动添加一个无参数的构造方法。使用构造方法创建对象虽然常用，但并不灵活。因此，我们再来了解一下动态化实例类，也就是调用Class对象的newInstance()方法，通过反射创建对象。

了解一下反射的概念：

JAVA反射机制是在运行状态中，对于任意一个类，都能够知道这个类的所有属性和方法；对于任意一个对象，都能够调用它的任意一个方法和属性；这种动态获取的信息以及动态调用对象的方法的功能称为java语言的反射机制。

那么我们就开始来学习如何动态化实例类：

请在在Eclipse中创建一个项目DynamicClass，并在该项目中创建com.shiyanlou.course包。在该包中创建Test类，并编写main()方法。

在main()方法中创建一个File对象（你只需要知道File对象也是一个对象即可，[进一步了解File对象](http://docs.oracle.com/javase/7/docs/api/java/io/File.html)）。

最后，使用该对象在桌面创建一个文本文件。

主要的代码如下：

package com.shiyanlou.course;

import java.io.File;

import java.lang.reflect.Constructor;

//需要引用上述两个包

public class Test {

public static void main(String[] args) {

try {

Constructor<File> constructor = File.class.getDeclaredConstructor(String.class); //string.class是构造函数的参数类型

//获得File类的Constructor对象

System.out.println("Create File Object with reflection.");

//使用反射创建File对象

File file = constructor.newInstance("/home/shiyanlou/Desktop/MyFile.txt");

System.out.println("Use File Object to create MyFile.txt on desktop.");

//指定了创建的路径为桌面，名称为“MyFile.txt”

file.createNewFile(); //创建新的文件

System.out.println("File is created ?" + file.exists());

//验证文件是否创建成功

} catch (Exception e) {

e.printStackTrace();

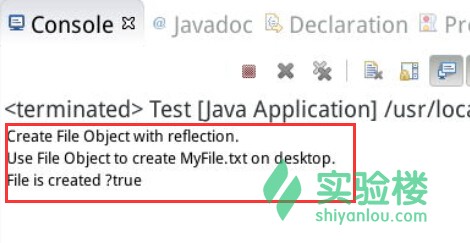
}

}

}

你可能会问为什么会用到try、catch以及Exception之类的奇怪的东西，这是由于上述代码会抛出大量的异常。我们通常不推荐这样写。不用担心，我们将在**错误处理**一章为你详细介绍他们的原理和用法。

点击编译并运行，如果在控制台可以看到下列信息则表示你成功了。



同时，你也应该可以在桌面上发现自己创建的这个MyFile.txt文件。



动态类：

Constructor<Book> constructor=Book.class.getDeclaredConstructor(String.class,String.class,float.class,String.class);  
Book book4=constructor.newInstance("java 动态类开发","Daniel",123,"1232222333");  
System.out.println(book4.toString());

**四、实验总结**

本章我们学习了J2SE的基本概念和架构，复习了面向对象中类和对象、动态实例化类的相关知识。建议大家再积累一些数据结构相关知识，以便更好地投入到后续的学习中。

J2SE的整个体系还是很庞大的，本课程会力所能及地涵盖其核心的知识点，如果你想更加深入地学习Java知识，我们还推荐你去阅读**《Java 2核心技术》**等书籍。

# 字符串与包装类

## 一、实验简介

在本章，我们将学习一些用于处理字符串的API以及包装类的相关知识。

本章知识点

* 字符串API
* 包装类及其应用

## 二、认识字符串类

### 1. 什么是字符串类

Java字符串类（java.lang.String）是Java中使用最多的类，也是最为特殊的一个类，很多时候，我们对它既熟悉又陌生。

首先你要知道什么是字符串：

字符串（String）是由数字、字母、下划线组成的一串字符，是编程语言中表示文本的数据类型。

你可以[查看更多关于字符串的知识](http://baike.baidu.com/link?url=yqDEZns6qOj21Eh-JTJ4Wk0txwUKJYhApEHD-CCAK85LTdhGmC4vxt-RtXr9dZw0so-PvCOIQeW7KIVIXvS1sK)。

操作字符串的常用方法有复制、计算其长度、查找字符串中的某些字符等。

然而，Java标准库中的String类包含很多的方法，要想记住所有的类和方法是一件不太可能的事情。最好的办法就是养成随时查阅API文档的习惯。在最后的参考文献中我们为你提供了官方API文档的链接，你可以在其中详细查看每一个操作字符串的方法。

**查阅API文档对于未来章节的学习也十分重要，请保持这个习惯。**

### 2. 用String类中的自带方法操作字符串

在Eclipse中创建项目StringUtil，新建一个com.shiyanlou.course包，并创建StringTools类。

这里会用到trim()、substring()、equals()和length()方法，

* String trim()，该方法返回一个新字符串，新的字符串删除了原始串中首尾的空格。
* String substring(int beginIndex, int endIndex)，该方法返回一个新字符串，新的字符串包含原来字符串中从beginIndex到endIndex-1的所有内容。
* boolean equals(Object other)，该方法用于比较两个字符串，如果原字符串与考虑大小写的情况下与other相等，则返回true。
* int length()，该方法返回一个字符串的长度。

这个项目首先需要获取来自用户的输入，然后去除输入的字符串中首尾多余的空格，再与字符串shiyanlou相比较，如果两个字符串相同，则截取该字符串前6位部分作为新的字符串。

主要的代码如下：

**同样，你不必在实验环境中输入这些中文注释。**

package com.shiyanlou.course;

import java.util.Scanner;

public class StringTools {

public static void main(String[] args) {

String a = new String();

String b = new String();

String c = new String();

//创建3个字符串对象

Scanner scan = new Scanner(System.in);// 创建扫描器

System.out.println("Please input a sentence with 10 more letters include space(s) : ");

// 获得用户输入的包含空格的字符串

a = scan.nextLine();

//将键盘上的输入赋到字符串a中

b = a.trim();

//将a中头部和尾部的空格去掉，赋值到b中

if(b.equals("shiyanlou"))

c = b.substring(0, 6);

else {

c = "";

}

//如果b字符串为shiyanlou，则取其第0个到第6个字符作为子串，赋值到c中

//如果b字符串不为shiyanlou，则c的值为空

System.out.println("a :"+a);

System.out.println("The length of a is :"+a.length());

System.out.println("b :"+b);

System.out.println("The length of b is :"+b.length());

System.out.println("c :"+c);

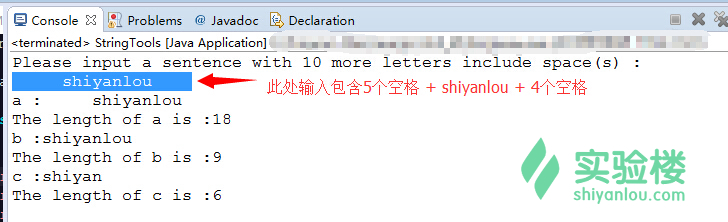
System.out.println("The length of c is :"+c.length());

//分别输出a、b、c三个字符串及其长度

}

}

点击编译并运行，在控制台先输入5个空格，紧接着输入shiyanlou，再输入4个空格并回车，可以看到输出结果如下：



你可能想问：我们在比较字符串的时候为什么不用==而用equals()呢？ 你可以[点此了解更多关于"=="和equals的信息](http://www.cnblogs.com/zhxhdean/archive/2011/03/25/1995431.html)。

对于基本数据类型（值类型），==比较的是两者的值；而对于符合数据类型（引用类型等），==比较的是两者在内存中的地址。

equals是后来实现，实现与object基类，也是调用==来实现的，因此没有被覆盖的情况下是比较内存地址，除非被覆盖。

## 三、认识包装类

### 1. 什么是包装类

我们知道，Java语言是面向对象的，但Java中的基本数据类型却不是面向对象的。这在实际使用过程中造成了诸多不便。为了解决此问题，Java语言的设计者们在设计类时，为每种基本数据类型也设计了一个对应的类。这八个和基本数据类型对应的类统称为包装类(Wrapper Class)，也称为外覆类或数据类型类。

包装类均位于java.lang包中，下表给出了包装类和基本数据类型的对应关系：

| **基本数据类型** | **对应包装类** |
| --- | --- |
| byte | Byte |
| boolean | Boolean |
| short | Short |
| char | Character |
| int | Integer |
| long | Long |
| float | Float |
| double | Double |

包装类的用途主要包含两种：

* 作为和基本数据类型对应的类类型存在，方便涉及到对象的操作。
* 包含每种基本数据类型的相关属性如最大值、最小值等，以及相关的操作方法。

下面我们通过一个例子来进一步认识包装类。

### 2. 比较Double类型

首先你应该知道double和Double的区别，前者是基本的数据类型，后者是引用类型，即包装类。我们可以直接使用普通运算符“==”对基本数据类型进行比较，但如果将“==”用于比较引用类型的话，只会判断其内存地址是否相同，并且结果通常是否定的。

在Eclipse中创建项目DoubleCompare，新建一个com.shiyanlou.course包，并创建DoubleCompare类。

在该类的main()方法中，定义两个含有数字的字符串并将其转换为Double类型的数据。

String s\_1 = "123.321";

String s\_2 = "567.765";

//显然，这里的123.321和567.765都不是数值，而是“一句话”

//你也可以自定义这些数字

Double num\_1 = Double.parseDouble(s\_1);

Double num\_2 = Double.parseDouble(s\_2);

//定义了Double类型的数据，并将字符串转换为双精度浮点数赋予其值

上面的parseDouble(String s)就是Double类中自带的将字符串中的数字转换为double类型的方法。

我们再将其输出。

System.out.println("number 1: " + num\_1);

System.out.println("number 2: " + num\_2);

然后我们来比较这两个数据，主要代码如下：

switch (num\_1.compareTo(num\_2)) {

case -1:

System.out.println("number 1 is smaller than number 2");

break;

case 0:

System.out.println("number 1 is equal to number 2");

break;

case 1:

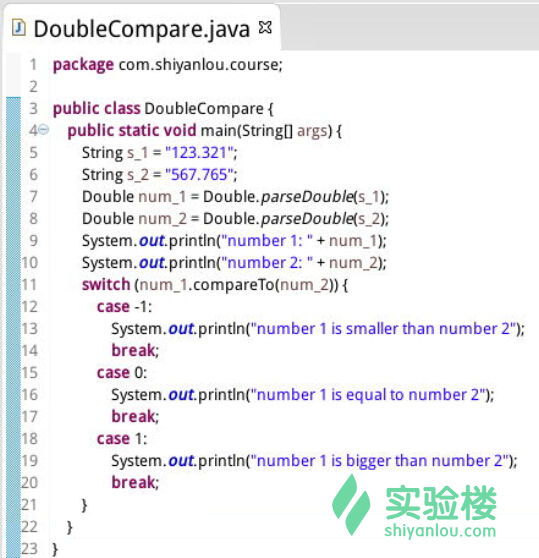
System.out.println("number 1 is bigger than number 2");

break;

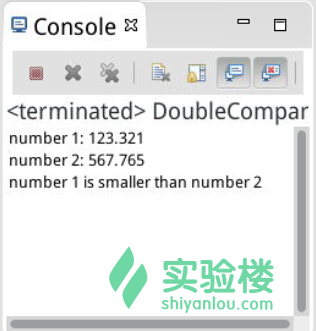
}

相信你能够理解这个分支语句的含义。需要特别说明的是，compareTo(Double anotherDouble)也是Double类提供的一个方法，用于比较两个Double类型数据的大小，返回值为int类型，-1、0、1分别代表小于、等于和大于。

该案例的完整代码如下图所示：



点击编译并运行，可以在控制台看到输出结果：



## 四、实验总结

在本章我们主要学习了字符串的定义和常用的生成子串、去除空格、比较等方法，和包装类的定义及其主要的操作方法。参考文档中为你提供了更多关于它们的知识，不妨试试字符串拼接等本课程还未用到的方法。在实验楼学习结束后，也不要忘记多看看这些API文档，多动手写写代码，天道酬勤！

try {  
 d3=Double.parseDouble("s\_2"); //抛出了异常就不会执行这句  
}catch(NumberFormatException e){  
 System.out.println("出错了");  
 System.out.println(d3); //12.0  
}

## 参考文档

关于字符串API，你可以查看Java SE官方文档获取更多更好玩的内容：

* [Java SE官方API文档 - Java.lang.String](http://docs.oracle.com/javase/7/docs/api/java/lang/String.html#method_detail)

以下是Java中一些包装类的官方技术文档：

* [Java SE官方API文档 - Java.lang.Byte](http://docs.oracle.com/javase/7/docs/api/java/lang/Byte.html)
* [Java SE官方API文档 - Java.lang.Boolean](http://docs.oracle.com/javase/7/docs/api/java/lang/Boolean.html)
* [Java SE官方API文档 - Java.lang.Integer](http://docs.oracle.com/javase/7/docs/api/java/lang/Integer.html)

我们没有给出所有的包装类参考文档，请你自己动手Ctrl+F体会一下查找的过程。

# 集合类框架

## 一、实验简介

在Java基础语法中，你可能接触到了数组这一概念。

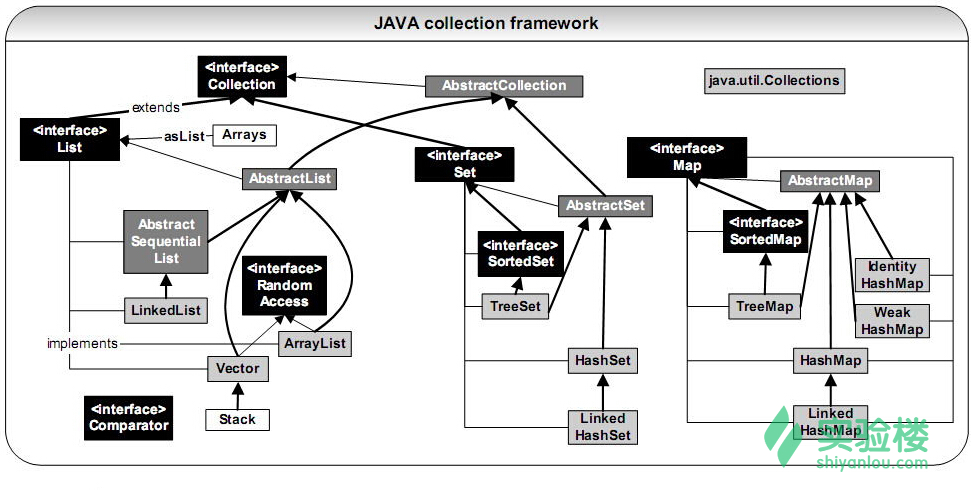
本章知识点：

* 集合类框架的概念
* ArrayList
* HashMap

## 二、什么是集合类框架

在Java语言中，Java语言的设计者对常用的数据结构和算法做了一些规范（接口）和实现（具体实现接口的类）。所有抽象出来的数据结构和操作（算法）统称为Java集合框架（Java Collection Framework）。 这意味着我们在具体应用时不必考虑数据结构和算法实现细节，只需要用这些类创建出来一些对象并加以应用就可以了，从而提高了编程效率。你可以[了解更多关于Java集合框架的知识](http://baike.baidu.com/link?url=hScyy3wtV3jrp3OyyLUs831YjOcXNX96jVkQ7RCozzmIDsO0M_yutaBY2X_lWQwJ2nc2xVelzNlLXnO5_FuBi_)。

Java.util 中的集合类包含 Java 中某些最常用的类。最常用的集合类是 List 和 Map。List 的具体实现包括 ArrayList 和 Vector，它们是可变大小的列表，比较适合构建、存储和操作任何类型对象元素列表。List 适用于按数值索引访问元素的情形。

下图是Java集合框架的结构图，从此处你可以大致了解到各种类型的集合之间的关系。 

## 三、认识ArrayList

ArrayList就是动态数组，支持自动改变大小，可以灵活的插入和删除元素。你可以[进一步了解ArrayList](http://baike.baidu.com/view/1827383.htm)。

ArrayList 定义了一些用于插入和删除元素的方法，了解一下：

* add(E e)，将指定的元素添加到此列表的尾部
* clear()，移除此列表中的所有元素
* contains(Object o)，如果此列表中包含指定的元素，则返回 true
* get(int index)，返回此列表中指定位置上的元素
* remove(int index)，移除此列表中指定位置上的元素
* toArray()，按从第一个到最后一个元素的顺序返回包含此列表中所有元素的数组

我们通过下面一个遍历ArrayList的实例来体会其用法，首先请在Eclipse中新建项目ArrayListExp，创建一个包com.shiyanlou.course和类ArrayListTraversal。

主要的代码如下，在注释里你可以看到对于各行代码的解说：

package com.shiyanlou.course;

import java.util.ArrayList;

import java.util.List;

public class ArrayListTraversal {

public static void main(String[] args) {

List<Integer> list = new ArrayList<Integer>();

// 创建列表list,利用多态

for (int i = 0; i < 10; i++) {

list.add(i);

// 使用add()方法，通过for循环向list中增加10个元素，每个元素即当前循环的次数值

}

System.out.println("Items in the list：");

System.out.println(list);

// 输出列表中全部的元素

System.out.println("Items in the list with odd index:");

for (int i = 1; i < list.size(); i += 2) {

// 输出列表中序号为奇数的元素，注意 i += 2 的用法

System.out.print(list.get(i)+" ");

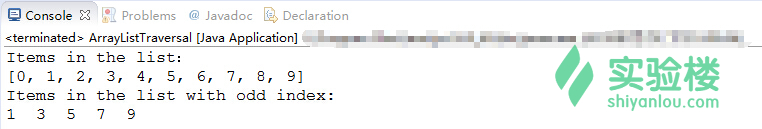
//使用get()方法从list中取出元素

}

}

}

编写完成后，点击编译并运行，我们可以在控制台看到下面的输出结果：



## 四、认识HashMap

Java 自带了各种 Map 类，这些 Map 类可归为三种类型：

* 通用 Map（HashMap、Hashtable、Properties、LinkedHashMap、IdentityHashMap等）
* 专用 Map（java.util.jar.Attributes、javax.print.attribute.standard.PrinterStateReasons等）
* 一个用于帮助实现您自己的 Map 类的抽象类

其中，通用Map用于在应用程序中管理映射，通常在 java.util 程序包中实现，可以直接使用。

Map 定义了几个用于插入和删除元素的方法，通过这些方法可以更改 Map 中的内容。

* clear()，从 Map 中删除所有映射
* remove(Object key)，从 Map 中删除键和关联的值
* put(Object key, Object value)，将指定值与指定键相关联
* clear()，从 Map 中删除所有映射
* putAll(Map t)，将指定 Map 中的所有映射复制到此 map

HashMap是基于哈希表的 Map 接口的实现。此实现提供所有可选的映射操作，并允许使用 null 值和 null 键。这里的键值对（key - value），你可以[进一步了解HashMap](http://baike.baidu.com/link?url=fNN0X2wuqs_2NCbawoSSQbpLNtvubZULgy5mcykcpjvh81kyJbt-dqfRIxpYTZNZgwQMMI2iwX_iOePk681KZa)。

下面我们使用HashMap及其中的一些方法来做一个简单的电话簿，目标是让用户通过输入待查询的姓名来获取其对应的电话号码。

首先请在Eclipse中新建一个项目PhoneBook，创建包com.shiyanlou.course和类PhoneBookByMap。主要的代码如下：

package com.shiyanlou.course;

import java.util.HashMap;

import java.util.Scanner;

public class PhoneBookByMap {

public static void main(String[] args) {

HashMap<String, String> phonebook = new HashMap<String, String>();

//创建一个HashMap对象，名称为phonebook，键值对的类型均为String

Scanner scan = new Scanner(System.in);

//创建扫描器用于获取用户的输入

String keyword = new String();

//创建一个名为keyword的字符串用于存放用户输入的关键词

phonebook = initPhoneBook(phonebook);

//利用自写的initPhoneBook方法为电话簿装入数据，完成其初始化的步骤

System.out.println("Please input a name that you want to search:");

//提示用户输入待查询的姓名

keyword = scan.nextLine();

//将用户的输入信息装入字符串keyword中

if(keyword.isEmpty()){

//使用isEmpty()方法判断用户是否真正输入了内容，如没有则提示其输入

System.out.println("Please input a name!");

}

else {

//若用户有输入某个关键词，则按照该关键词在phonebook中查找其对应的号码

System.out.println("The result is :");

System.out.println(queryPhone(phonebook, keyword));

//这里调用了自写的queryPhone()方法

}

}

private static HashMap<String, String> initPhoneBook(HashMap<String, String> phonebook){

//该方法用于完成指定HashMap的初始化，为其装入一些号码数据，你也可以自定义这些数据

//Q：你还清楚形参和实参的区别吗？

phonebook.put("Steve", "13012345678");

phonebook.put("Bob", "028-80001234");

phonebook.put("Peter", "182222233333");

//使用put()方法将姓名与电话号码相关联，存放在名为phonebook的HashMap中

//put()方法中，第一个参数为关键词key，第二个参数为其对应的值value

return phonebook;

//返回修改后的phonebook

}

private static String queryPhone(HashMap<String, String> phonebook, String keyword) {

String result = new String();

//创建存放结果的字符串result

result = phonebook.get(keyword);

//使用get()方法查询keyword所对应的电话号码，并赋给result

//put()方法中，参数为关键词key，返回值为其对应的值value，未找到对应值时，返回值为null

if(result == null)

return "Can not find this user.";

//如果未找到该用户的电话号码，则将查询结果修改为“未找到该用户”

return result;

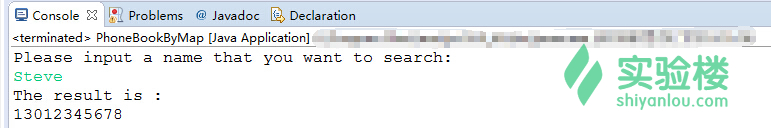
//返回查询结果

}

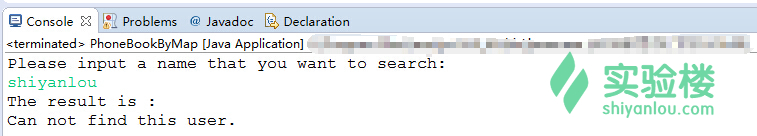
}

这个例子稍显复杂，需要你弄清楚其中各个模块之间的关系。检查代码无误后，点击编译并运行，在控制台分别输入无效姓名、有效姓名（例如Steve，注意大小写）和空值，观察下程序返回的结果是否正常。

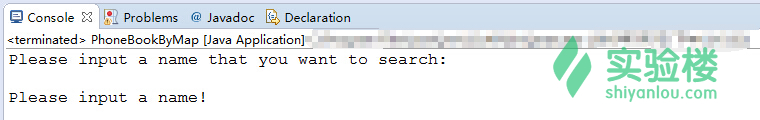
查找到号码的情况：



找不到号码的情况：



以及你什么都不输入，直接回车得到的提示：



## 五、实验总结

本章我们学习了Java中的集合类，并动手定义了ArrayList和HashMap，使用了其自带的方法来完成了诸如存取数据的工作。在课程结束之后，你也应该结合API文档，动手再使用一下这些集合类中的其他类型及其方法。

## 六、作业

请查阅资料，回答下面几个问题：

1. Collection与Collections的区别？
2. List、Set和Map是否继承自Collection接口？
3. ArrayList与Vector的区别？
4. HashMap与Hashtable的区别？

## 参考文档

* [Java的集合框架最全详解（图）](http://davidisok.iteye.com/blog/2106563)
* [Java SE官方API文档 - java.util.Collection](http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html)

# 数学工具

## 一、实验简介

Java为开发者提供了基本的数学工具类——Math类。Math类包含用于执行基本数学运算的方法，如初等指数、对数、平方根和三角函数。利用这些方法，我们可以对数值进行更加复杂的运算。

本章知识点：

* 查看数学常量
* 三角函数计算
* 随机数生成

下面我们通过一些例子来看Math类中都有哪些内容。

## 二、Math类中的常量

Math类自带了两个常量，分别是：

* 比任何其他值都更接近 e（即自然对数的底数）的 double 值。
* 比任何其他值都更接近 pi（即圆的周长与直径之比）的 double 值。

在Eclipse中创建一个MathConstant项目，新建一个com.shiyanlou.course包并创建一个ShowConstant类，你可以在main()方法中尝试打印出它们的值。

主要的代码如下：

package com.shiyanlou.course;

public class ShowConstant {

public static void main(String[] args) {

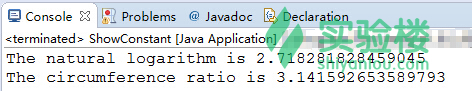
System.out.println("The natural logarithm is " + Math.E);

System.out.println("The circumference ratio is " + Math.PI);

}

}

点击编译并运行，你能得到如下图的结果：



## 三、如何利用数学工具

### 3.1 三角函数计算

如你在初高中所学，三角函数是数学领域里十分重要的一个分支，很多的问题通过使用三角函数能够变得容易。而Math类也提供了常用的三角、反三角函数的实现，我们动手来利用三角函数计算一些角度的正弦、余弦、正切值。

在Eclipse中创建一个MathTools项目，新建一个com.shiyanlou.course包并创建一个TriFunction类。

编写main()函数，这里将再次用到上个例子中的数学常量π，主要的代码如下：

package com.shiyanlou.course;

public class TriFunction {

public static void main(String[] args) {

System.out.println("sin30 = " + Math.sin(Math.PI / 6));

// 计算30°的正弦值，参数是用弧度表示的角，即π的六分之一

System.out.println("cos30 = " + Math.cos(Math.PI / 6));

// 计算30°的余弦值，这些计算三角函数的方法，其参数和返回值的类型都为double

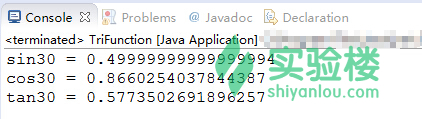
System.out.println("tan30 = " + Math.tan(Math.PI / 6));

// 计算30°的正切值

}

}

检查代码，点击编译并运行，可以在控制台看到计算的结果：



也许你会觉得简单的计算很枯燥，但在合适的地方用到它们可以让你的程序（比如游戏中飞行轨迹的计算）更加精彩。

### 3.2 生成彩票号码

有了之前的基础，我们来做个更好玩的项目——彩票生成器。目前市面上彩票的种类很多，我们就以大乐透为例吧。大乐透的规则是：从1~35中随机选取5个不重复的数字，从1~12中随机选取2个不重复的数字，这些数字组成了一个七位数。如果你买的号码与摇出来的号码相同，那么你一夜之间可能就不会当程序员了。

因为彩票号码是随机生成的，所以这个项目中我们也会带大家认识随机数的使用方法。首先请在Eclipse中创建一个GenerateLottery项目，新建一个com.shiyanlou.course包并创建一个Lottery类。

对于该项目所需要用到的API，你需要引入下面一些包：

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

import java.util.Random;

由于彩票号码的前半段和后半段的规则和个数略有不同，所以我们分别创建生成它们的方法。

首先来创建生成前段号码的方法getHeadNumber()，该方法需要从1到36中随机选取一个数作为号码（即摇号的过程）。对于个位数，为了显示上更整齐，所以我们在其十位再放一个“0”。再者，为了结果更加易读，我们还需要调用Collections.sort()方法对结果进行排序。

代码片段如下：

public List<String> getHeadNumber() {

List<String> list = new ArrayList<String>();

// 创建彩票的前半段号码集合

String lotteryNumber = "";

for (int i = 1; i < 36; i++) {

// 这个循环是为了初始化彩票的前半段号码集合

if (i < 10) {

list.add("0" + i + " ");

// 在集合中添加0~9的号码，因为是个位数，为了显示上更加人性化，所以需要在十位的地方添加“0”

} else {

list.add("" + i + " ");

// 向集合添加大于9的号码，即添加双位数

}

}

int roundIndex = 0;

// 创建并初始化随机数

List<String> lotteryList = new ArrayList<String>();

// 保存前段号码的List集合

for (int j = 0; j < 5; j++) {

int amount = list.size(); // 获取前半段号码的个数

Random r = new Random(); // 创建并实例化Random的对象

roundIndex = r.nextInt(amount); // 获取一个0到amount-1的随机数

lotteryNumber = list.get(roundIndex); // 获取彩票数字，即摇号的号码

lotteryList.add(lotteryNumber); // 将彩票号码添加lotteryList中

list.remove(roundIndex); // 移除刚刚产生的号码

}

Collections.sort(lotteryList);

// 对前半段号码进行排序，排序的目的是为了让结果更具可读性

return lotteryList;

//返回前半段号码

}

此处用到了Random()方法，在创建一个Random对象r后，你可以调用该对象的nextInt()方法得到一个随机数。r.nextInt(amount);中的amount是随机数的上限，产生的随机数为0到100（不含100）的整数。 你可以[了解更多Random的用法](http://www.cnblogs.com/ningvsban/p/3590722.html)。

和前半段号码类似，我们接着来写生成后半段号码的方法getRearNum()，代码片段如下：

public List<String> getRearNumber() {

List<String> list = new ArrayList<String>();

// 创建后半段号码集合，也就是最后两个球的数字

String lotteryNumber = "";

for (int i = 1; i < 13; i++) {

// 初始化后半段号码集合

if (i < 10) {

list.add("0" + i + " ");

// 添加0~9的号码，原理同前半段

} else {

list.add("" + i + " ");

// 添加大于9的号码

}

}

int roundIndex = 0;

//创建并初始化随机数

List<String> lotteryList = new ArrayList<String>();

// 保存后半段号码的List集合

for (int j = 0; j < 2; j++) {

int amount = list.size(); // 获取后半段号码的个数

Random r = new Random(); // 创建并实例化Random的对象

roundIndex = r.nextInt(amount); // 获取一个0到amount-1的随机数

lotteryNumber = list.get(roundIndex); // 摇号

lotteryList.add(lotteryNumber); // 将彩票号码添加lotteryList中

list.remove(roundIndex); // 移除刚刚产生的号码

}

Collections.sort(lotteryList);

// 对后半段号码进行排序

return lotteryList;

}

现在，根据规则我们已经能够产生对应的数字了。再接再厉，我们再做一个生成最终结果的方法，把这两个号段组装成整体的号码，并且按照用户的要求产生多少组号码，在控制台输出它们。

下面给出了这个方法的代码片段：

public void generateLottery(String groupNum) {

int groupNumber = 0;

//为了避免不必要的错误，一般在创建变量时都要为其赋初始值

groupNumber = Integer.parseInt(groupNum);

StringBuilder sbuiler = new StringBuilder();

// 创建字符串生成器对象，使用字符串生成器能够较为方便地在字符串中追加内容

for (int i = 0; i < groupNumber; i++) {

List<String> startList = getHeadNumber();

// 获得彩票前半段号码的集合

List<String> endList = getRearNumber();

// 获得彩票后半段号码的集合

for (int m = 0; m < startList.size(); m++) {

sbuiler.append(startList.get(m));

// append()即为追加方法，用于添加彩票的前半段号码到字符串生成器中

}

sbuiler.append(" ");

for (int n = 0; n < endList.size(); n++) {

sbuiler.append(endList.get(n));

// 添加彩票的后半段号码到字符串生成器中

}

sbuiler.append("\n");

}

System.out.println(sbuiler.toString());

//将每组生成好的彩票号码即时输出

}

为了验证我们的实现无误，在同一个包中创建一个名为LotteryTest的测试类，编写其main()方法。在其中创建一个彩票对象并调用其产生号码的方法。

LotteryTest.java中的内容为：

package com.shiyanlou.course;

import java.util.Scanner;

public class LotteryTest {

public static void main(String[] args) {

Lottery l = new Lottery();

Scanner scan = new Scanner(System.in);// 创建扫描器

System.out.println("Please input the number of lottery group(s) that you want to generate : ");

// 获得用户输入的需要生成的中奖号码个数

String groupNum = scan.nextLine();

l.generateLottery(groupNum);

}

}

检查一下代码，点击编译并运行，在控制台输入你要生成多少组的彩票号码，按下回车键你就能看到生成的结果。



最后，希望这些号码能给你带来好运。

## 四、实验总结

我们在这些实验中学习了调用查看数学常量，使用三角函数的相关方法，并且调用了随机数的生成方法。

这一节的实验稍微复杂了一点，不过能用自己写的程序来机选号码也是很有趣的一件事儿呢。

为什么我们先写了各个子功能，最后才来写主方法呢？这是因为在开发过程中，一般情况下，最好是让各个功能模块各司其职，main()方法中只负责数据的输入输出和调用子功能。这种设计思想就叫[模块化](http://baike.baidu.com/link?url=gX5UOScH8s40gJmV65zxEisTo5osO8NH2VHQ8UHUYsB7FW5C0hEB9CfkY-LlfXyV5MIgTc9fGKnIu7fvv0s-8K#4)设计。

除了我们刚刚用到的这些，Math类中更多的方法可以在文末参考文献中提供的官方API文档中查看。

## 五、作业

在数学领域，角的大小可以通过角度和弧度来度量，请查阅API文档，使用Math类中的toRadians()和toDegrees()方法，实现角度和弧度之间的转换。

## 参考文档

* [Java数学工具包java.lang.Math类](http://blog.csdn.net/itlwc/article/details/10077305)
* [Java SE官方API文档 - Java.lang.Math](http://docs.oracle.com/javase/7/docs/api/java/lang/Math.html)
* [Java SE官方API文档 - Java.util.Random](http://docs.oracle.com/javase/7/docs/api/java/util/Random.html)

# 错误处理

## 一、实验简介

还记得在第一章“认识J2SE”的例子中我们就见识了try和catch结构吗？它们就是用于捕获异常的关键词。

在真实的程序运行过程中，诸如除0溢出、数组越界、文件找不到等错误的发生，将阻止程序的正常运行。而在Java世界里，这些错误我们通常称之为异常。本章我们就将学习Java中的异常机制，观察程序中的错误是如何发生并解决的，你还有机会自己动手定义异常类。

本章知识点：

* 常见异常的产生
* 自定义异常类
* 捕获异常

## 二、常见异常

异常指不期而至的各种状况，它在程序运行的过程中发生。作为开发者，我们都希望自己写的代码永远都不会出现bug，然而现实告诉我们并没有这样的情景。如果用户在程序的使用过程中因为一些原因造成他的数据丢失，这个用户就可能不会再使用该程序了。所以，对于程序的错误以及外部环境能够对用户造成的影响，我们应当及时报告并且以适当的方式来处理这个错误。

之所以要处理异常，也是为了增强程序的健壮性。

异常都是从Throwable类派生出来的，而Throwable类是直接从Object类继承而来。你可以在Java SE官方API文档中获取更多关于它们的知识。

异常通常有四类：

* Error：系统内部错误，这类错误由系统进行处理，程序本身无需捕获处理
* Exception：可以处理的异常
* RuntimeException：可以捕获，也可以不捕获的异常
* 继承Exception的其他类：必须捕获，通常在API文档中会说明这些方法抛出哪些异常

我们先来体验3种较为简单的异常。

### 1. 算术异常（ArithmeticException）

Java SE官方文档对于算术异常的定义是：

当出现异常的运算条件时，抛出此异常。例如，一个整数“除以零”时，抛出此类的一个实例。

关于除以零还有[更有趣的故事](http://www.guokr.com/article/440472/" \t "_blank)。

所以，我们来抛出一个算术异常看看吧。在Eclipse中新建一个项目AriExption，创建com.shiyanlou.course包，新建一个AriExceptionTest类。

我们需要在这个类的main()方法中模拟出现算术异常的情况， 主要的代码如下：

package com.shiyanlou.course;

public class AriExceptionTest {

public static void main(String[] args) {

System.out.println("Example 1: -1.0 / 0 = " + (-1.0 / 0));

// 演示负浮点数除0

System.out.println("Example 2: +1.0 / 0 = " + (+1.0 / 0));

// 演示正浮点数除0

System.out.println("Example 3: -1 / 0 = " + (-1 / 0));

// 演示负整数除0

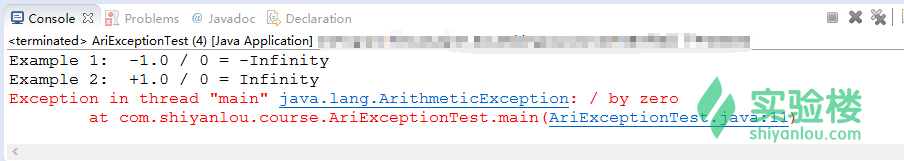
System.out.println("Example 4: +1 / 0 = " + (+1 / 0));

// 演示正整数除0

}

}

在控制台可以看到该异常的输出信息：



可以看到，实际上程序在运行到Example 3的时候就已经出现了算数异常。当代码抛出一个异常的同时，也终止了对剩余代码的处理，所以Example 4根本没有机会运行。

那么Example 1和2中为什么能出现结果呢？

这是由于在Java中，浮点数（无论是float还是double类型的浮点数）被0除，并不会引发算术异常。具体说来，是操作系统为了保护应用软件而已经处理了该异常，不再抛出，最终运算结果是无穷大。

### 2. 数组下标越界异常（ArrayIndexOutOfBoundsException）

Java SE官方文档对于数组下标越界异常的定义是：

用非法索引访问数组时抛出的异常。如果索引为负或大于等于数组大小，则该索引为非法索引。

通俗的来说，例如你的数组大小是6，你能访问的范围即0到5，如果你用下标13去访问其中的元素，就会遇到此问题。

动手试一下，在Eclipse中新建一个项目ArrayExption，创建com.shiyanlou.course包，新建一个ArrayExceptionTest类。

主要的代码如下：

package com.shiyanlou.course;

import java.util.Arrays;

public class ArrayExceptionTest {

public static void main(String[] args) {

int[] array = new int[5];

// 声明一个长度为5的整型数组array

Arrays.fill(array, 8);

// 将该数组所有元素赋值为8

for (int i = 0; i < 6; i++) {

// 用遍历的方式，输出所有数组元素，注意此处的控制条件 i < 6

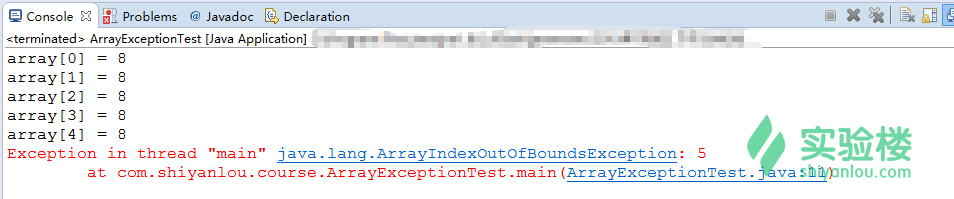
System.out.println("array[" + i + "] = " + array[i]);

//下标将会增加到5，显然是超出了数组array的范围（0到4），程序将在第6次循环前抛出异常

}

}

}



### 3. 空指针异常（NullPointerException）

Java SE官方文档对于空指针异常的定义是：

当应用程序试图在需要对象的地方使用 null 时，抛出该异常。这种情况包括：

* 调用 null 对象的实例方法。
* 访问或修改 null 对象的字段。
* 将 null 作为一个数组，获得其长度。
* 将 null 作为一个数组，访问或修改其时间片。
* 将 null 作为 Throwable 值抛出。

应用程序应该抛出该类的实例，指示其他对 null 对象的非法使用。

同样，我们来尝试抛出一个此类异常。在Eclipse中新建一个项目NullPointerExption，创建com.shiyanlou.course包，新建一个NullPointerExceptionTest类。

主要的代码如下：

package com.shiyanlou.course;

public class NullPointerExceptionTest {

public static void main(String[] args) {

String s= null;

// 将字符串设置为null

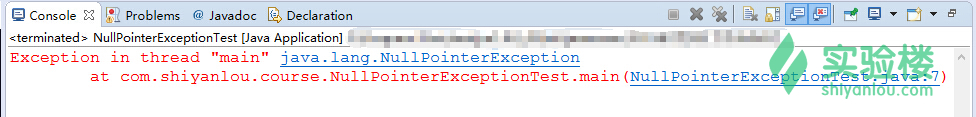
System.out.println(s.toUpperCase());

// 尝试将字符串转换成大写，看看会发生什么

}

}

点击编译并运行，可以看到，红色的异常信息又“如约而至”了。



除了上述3种，异常的种类还有很多很多，比如IllegalAccessException、FileNotFoundException以及在操作数据库时抛出的异常，你可以在API文档中查看它们的产生原因。再者，例如在Android开发过程中，我们也会经常看到控制台报出的各式各样的错误信息，这些异常是平时我们没见过的，它们又是怎么来的呢？我们不妨来了解一下自定义的异常类。

## 三、自定义异常类

尽管Java SE的API已经为我们提供了数十种异常类，然而在实际的开发过程中，你仍然可能遇到未知的异常情况。此时，你就需要对异常类进行自定义。

自定义一个异常类非常简单，只需要让它继承Exception或其子类就行。在自定义异常类的时候，建议同时提供无参构造方法和带字符串参数的构造方法，后者可以为你在调试时提供更加详细的信息。

百闻不如一见，下面我们尝试自定义一个算术异常类。

在Eclipse中创建项目MyException，创建com.shiyanlou.course包，并添加一个MyAriException类。

主要的代码如下：

package com.shiyanlou.course;

public class MyAriException extends ArithmeticException {

//自定义异常类，该类继承自ArithmeticException

public MyAriException() {

}

//实现默认的无参构造方法

public MyAriException(String msg) {

super(msg);

}

//实现可以自定义输出信息的构造方法，将待输出信息作为参数传入即可

}

在同个包内再添加一个ExceptionTest类作为测试用，在该类的main()方法中，可以尝试抛出自定义的异常。

代码片段如下：

package com.shiyanlou.course;

import java.util.Arrays;

public class ExceptionTest {

public static void main(String[] args) {

int[] array = new int[5];

//声明一个长度为5的数组

Arrays.fill(array, 5);

//将数组中的所有元素赋值为5

for (int i = 4; i > -1; i--) {

//使用for循环逆序遍历整个数组，i每次递减

if (i == 0) {

// 如果i除以了0，就使用带异常信息的构造方法抛出异常

throw new MyAriException("There is a exception occured.");

}

System.out.println("array[" + i + "] / " + i + " = " + array[i] / i);

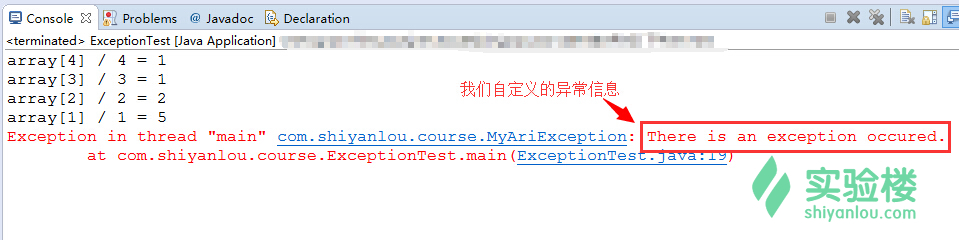
// 如果i没有除以0，就输出此结果

}

}

}

检查一下代码，点击编译并运行，期待中的自定义错误信息就展现在控制台中了：



## 四、捕获异常

当我们在编程时遇到了异常不要紧，除了可以将异常抛出，还可以将异常捕获。通常使用try和catch语句块来捕获异常，有时候还会用到finally。

对于上述三个关键词所构成的语句块，try语句块是必不可少的，catch和finally语句块可以根据情况选择其一或者全选（catch和finally语句块至少有一个）。你可以把可能发生错误或出现问题的语句放到try语句块中，将异常发生后要执行的语句放到catch语句块中，而finally语句块里面放置的语句，不管异常是否发生，它们都会被执行。

你可能想说，那我把所有有关的代码都放到try语句块中不就妥当了吗？可是你需要知道，捕获异常对于系统而言，其开销非常大，所以应尽量减少该语句块中放置的语句。

try {  
 throw new MyException("my exception");  
}finally {  
 System.out.println("zhixingle");  
}  
  
System.out.println("不会被执行"); //没有catch，如果发生异常会执行finally里面的但不会执行之后的语句；相当于jvm补充了catch语句并且在finally执行完后输出异常信息。

下面我们通过一个实例来练习如何捕获异常。请在Eclipse中新建项目CatchExp，创建包com.shiyanlou.course和类CatchException，添加下面的关键代码：

package com.shiyanlou.course;

public class CatchException {

public static void main(String[] args) {

try {

// 下面定义了一个try语句块

System.out.println("I am try block.");

Class<?> tempClass = Class.forName("");

// 声明一个空的Class对象用于引发“类未发现异常”

System.out.println("Bye! Try block.");

} catch (ClassNotFoundException e) {

// 下面定义了一个catch语句块

System.out.println("I am catch block.");

e.printStackTrace();

//printStackTrace()的意义在于在命令行打印异常信息在程序中出错的位置及原因

System.out.println("Goodbye! Catch block.");

} finally {

// 下面定义了一个finally语句块

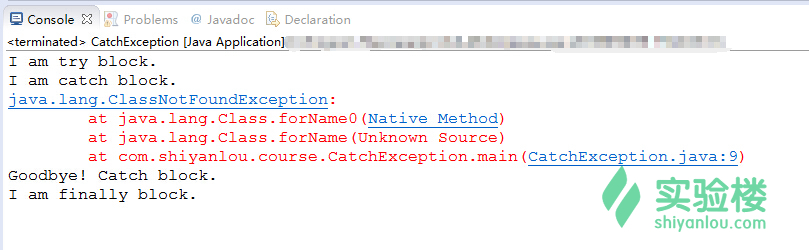
System.out.println("I am finally block.");

}

}

}

编写完成后，请点击编译并运行，可以在控制台看到如下的输出信息：



请你结合这些输出语句在源代码中的位置，再来体会一下三个语句块的作用。

## 四、实验总结

在本章，我们Eclipse中的控制台可谓是红成了一片海洋，希望你不要反感这些红色的异常信息。相反，这些信息还有利于你去定位程序中的错误。本章我们还学习了自定义异常类和捕获异常等手段。通过它们，你可以不断地增强程序的健壮性，真正使程序在错综复杂的运行和使用环境中具有“金刚不败之身”。

## 五、作业

试问，若在try语句块里有一个return语句，同时这个try语句块后面还有一个finally语句块，请问finally语句块里的代码会不会被执行？若答案是肯定的，那么会在什么时候被执行？是在return语句之前。

在try-catch-finally里面包含return语句时不好的编程习惯。

## 参考文档

* [J2SE官方API文档 - ArithmeticException](http://docs.oracle.com/javase/7/docs/api/java/lang/ArithmeticException.html)
* [J2SE官方API文档 - ArrayIndexOutOfBoundsException](http://docs.oracle.com/javase/7/docs/api/java/lang/ArrayIndexOutOfBoundsException.html)
* [J2SE官方API文档 - NullPointerException](http://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html)

# 输入输出

## 一、实验简介

还记得一次又一次地在程序中输入和输出数据吗？如果程序不能够读写数据，那么程序的功能会受到极大的限制。本实验就将为你介绍Java中的输入输出体系，你将有机会自己动手体会一下读写文件的过程。

本章知识点：

* Java的输入输出机制
* 字节流和字符流
* 文件的读取和写入

## 二、Java中的“流”

在大多数程序中，都需要对输入输出进行处理。例如我们在前面各个章节中需要获取用户从键盘上的输入，需要在控制台输出结果等等。除此之外还有从文件中读取数据，向文件中写入数据等等。在Java中，我们把这些不同类型的输入输出源抽象地称为流，也就是Stream；在里面输入输出的数据则称为数据流（Data Stream），它们通常具有统一的接口。

于是我们得到了数据流的定义：

一个Java I/O对象叫做数据流。读取数据的对象叫做输入流，写入数据的对象叫做输出流。

针对其面向的不同角度，我们大致可以将流分为下面几种类型：

* 按照数据流的方向不同分为输入流和输出流。这种分类不是绝对的，例如在向一个文件写入数据时，它就是输出流；而在读取数据时，它就是输入流。
* 按照处理数据的单位不同分为字节流和字符流。
* 按照功能的不同分为节点流和处理流。

需要特别说明，节点流是从特定的数据节点（文件、数据库、内存等）读写数据；处理流是连接在已有的流上，通过对数据的处理为程序提供更多功能。

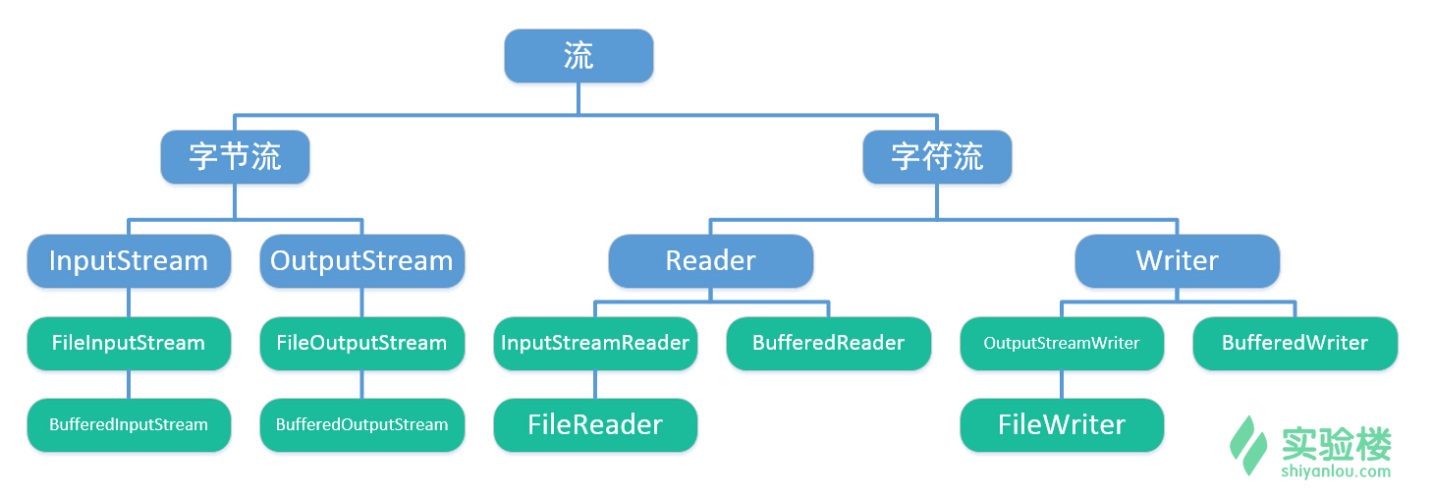
在Java环境中，java.io包提供了大多数的类和接口来实现输入输出管理。一些标准的输入输出则来自java.lang包中的类，但它们都是继承自java.io中的类。我们可以将输入流理解为数据的提供者，而把输出流理解为数据的接收者。在最初的时候，这些派生自抽象类InputStream和OutputStream的输入输出类是面向8位的字节流的。但为了支持国际化，又引入了派生自抽象类Reader和Writer的类层次，用于读写一些双字节的Unicode字符。

**因此，在学习java的输入输出上，我们希望你以字节流和字符流作为区分来学习。**

如果需要概括一下，则可以得到下面的定义：

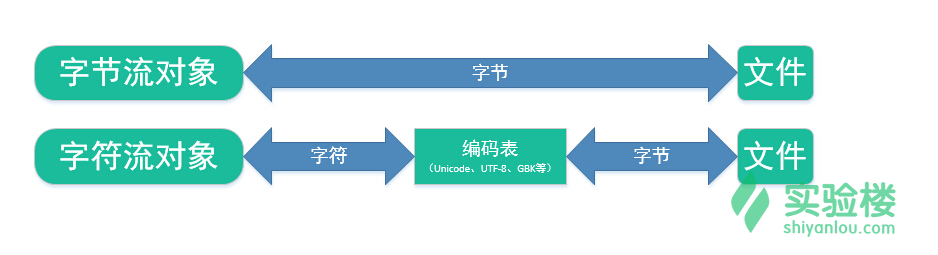
* 字节流：表示以字节为单位从stream中读取或往stream中写入信息。通常用来读取二进制数据。
* 字符流：以Unicode字符为单位从stream中读取或往stream中写入信息。

按照这样的定义，Java中流的层级结构可以通过下图来表示：



图中蓝色的部分均为抽象类，而绿色的部分则为派生类，是可以直接使用的。

而下图简要说明了字节流和字符流的区别，你也可以[进一步了解字节流与字符流的区别](http://blog.csdn.net/cynhafa/article/details/6882061)。



我们知道Java是一门面向对象的语言，所以为了能够永久地保存对象的状态，java.io包还以字节流为基础，通过实现ObjectInput和ObjectOutput接口提供了对象流。在此仅作引入，你可以通过查阅API手册来详细了解它们。

## 三、读写一个文件

### 1. 使用FileInputStream读取文件

我们在之前的实验中用到了java.util.Scanner来读取系统的标准输入流System.in。

Scanner scan=new Scanner(System.in);  
int i=0;  
while(true){  
 System.out.println("please input some words: \r");  
 System.out.println(scan.nextLine());  
 i++;  
 if(i>10){  
 break;  
 }  
}

这一次，我们来动手学习如何使用FileInputStream来读取文件。

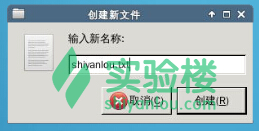
开始编码之前，请在实验环境完成新建文本文件的操作：

你也可以通过linux命令来完成该操作，想一想需要哪些命令？

首先右键单击桌面空白处，创建一个文件。



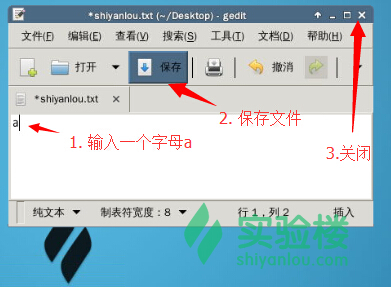
在弹出的对话框中输入文件名shiyanlou.txt，并点击创建按钮。



这样你在桌面上就创建好了一个我们需要的文本文件了。



双击这个文件的图标，用gedit编辑器打开此文件，在其中输入内容a，然后保存并关闭。



新建文件的过程就已经完成了。

在上一个实验中我们已经知道：在所有的输入输出过程中，都可能发生一些异常。所以我们得用try和catch结构来进行错误处理。

请接着在Eclipse中新建项目FileIO，创建包com.shiyanlou.course和包含main()方法的类ReadFileTest。

字节流中读取文件的方式主要有两种方法：

* int read()，从此输入流中读取一个数据字节。返回：下一个数据字节。如果已到达文件末尾，则返回 -1。
* int read(byte[] b)，从此输入流中将最多 b.length 个字节的数据读入一个 byte 数组中。返回：读入缓冲区的字节总数，如果因为已经到达文件末尾而没有更多的数据，则返回 -1。

这里使用前者来读取，代码片段如下，我们在注释中继续进行说明。

package com.shiyanlou.course;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

public class ReadFileTest {

public static void main(String[] args) {

try {

FileInputStream file = new FileInputStream("/home/shiyanlou/Desktop/shiyanlou.txt");

//声明一个文件输入流file，并指明该文件在系统中的路径以方便定位

int data = 0;

//声明一个整型变量用于存放读取的数据

while ((data = file.read()) != -1) {

//在while循环中使用read()方法持续读取file，数据赋到data中

//如果读取失败或者结束，则将返回-1，这个特殊的返回值可以作为读取结束的标识

System.out.println(data);

//输出读取到数据

}

file.close();

//文件读取结束之后一定要通过close()方法关闭这个流对象

} catch (FileNotFoundException e) {

//如果文件未找到，则捕获该异常

e.printStackTrace();

} catch (IOException e) {

//如果输入输出时发生错误，则捕获该异常

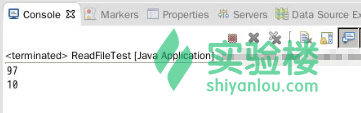
e.printStackTrace();

}

}

}

检查一下代码，点击编译并运行，可以在控制台看到读取的文件数据。



为什么我们在文本文件中输入的是a，而输出的结果是97和10呢？

这是因为文件是GBK编码，而文件类型为ANSI，a的ANSI编码恰好是97，10则代表了换行符。

需要在读取的时候将97等数字变为相应的字符：（char）data

public String readFromFile(String fileName) throws Exception{  
 if(null==fileName){  
 throw new Exception("fileName is null");  
 }  
  
 File file=new File(fileName);  
 if(!file.exists()){  
 throw new Exception("文件不存在");  
 }  
  
 FileInputStream fis=null;  
 try {  
 fis=new FileInputStream(file);  
 }catch(IOException e){  
 e.printStackTrace();  
 return null;  
 }  
  
 StringBuilder sbuilder=new StringBuilder();  
 int a;  
 while(-1!=(a=fis.read())){ //这样来判断  
 char c=(char)a; //将读取到的ANSI码转换为char  
 sbuilder.append(c);  
 }  
  
 fis.close(); //读取之后要关闭资源  
  
 return sbuilder.toString();  
}

中文等多字节内容这样每个字节的读取可能最后转换为char进而转换为string的时候会有问题，可能导致乱码，最好采用字符流的方式读取。

### 2. 读取文件的其他方式

如果你想使用BufferedInputStream来读取文件，只需要在创建对象时使用下面的格式即可：

BufferedInputStream file = new BufferedInputStream(new FileInputStream("文件的路径"));

其他的使用方法与FileInputStream一样，此处不再赘述，你不妨动手试试。

上述方法都是对于字节流的，那么对于字符流呢？

查阅API文档可以知道，字节流使用的数组是字节数组byte[] bt，而字符流使用的数组是字符数组char[] chs。

将上个程序的try语句块中的程序改为下面这些，就可以使用字符流的方式来读取文件。

FileReader file = new FileReader("/home/shiyanlou/Desktop/shiyanlou.txt");

//声明一个文件输入流file，并指明该文件在系统中的路径以方便定位

int data = 0;

//声明一个整型变量用于存放读取的数据

while((data=file.read())!=-1){

//在while循环中使用read()方法持续读取file，数据赋到data中

//如果读取失败或者结束，则将返回-1，这个特殊的返回值可以作为读取结束的标识

System.out.print((char)data);

//输出读取到数据

}

file.close();

//一定要记得读取结束后要关闭文件

这时候你编译并运行，看到的输出结果就不再是ASCI编码了，而是a：



这是因为我们在输出前进行了强制类型转换。

同样，如果你想使用BufferedReader来读取文件，则按照下面的格式来创建对象：

BufferedReader file = new BufferedReader(new FileReader(“文件的路径”));

BufferedReader有一个特有方法readLine();，它在读到文件末尾时返回null。

### 3. 使用FileOutputStream写入文件

在熟悉了读取文件的过程后，写入文件就显得比较简单了。我们动手来实践一下。

在Eclipse中，请继续在同一个项目FileIO的同个com.shiyanlou.course包下面创建类WriteFileTest。

代码片段如下：

package com.shiyanlou.course;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

public class WriteFileTest {

public static void main(String[] args) {

try {

String path = "/home/shiyanlou/Desktop/newfile.txt";

//这一次我们提前声明一个字符串用于存放待写入文件的路径

String content = "shiyanlou";

//声明一个字符串存放待写入的内容

FileOutputStream file = new FileOutputStream(path);

//声明一个文件输出流对象，并指定路径

file.write(content.getBytes());

//通过write()方法将数据写入到文件中

//getBytes()方法是将字符串转化为二进制序列

file.close();

//记得关闭文件

System.out.println("File created successfully.");

//提示用户创建成功

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

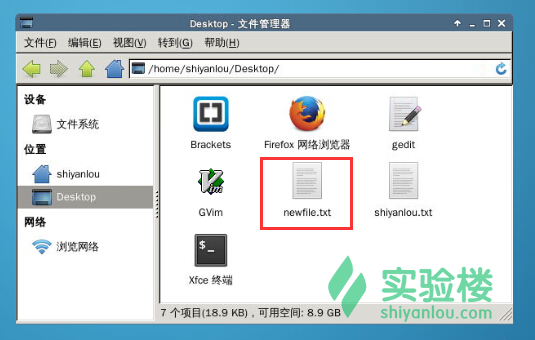
}

}

编译并运行，在控制台可以看到写入成功的提示。



在桌面上，也能看到这个文件。



双击打开这个文件，可以看到我们刚刚写入的内容。



## 四、实验总结

本章详细介绍了Java中输入输出的机制，展示了这个体系下的结构，并结合了两个主要的实验来体会输入输出的用法。但是这是远远不够的，对于一些较为复杂的输入输出方法，希望你在本实验的基础上进行更加深入的学习。

## 五、作业

请查阅API文档，使用BufferedWriter写一个名为bfFile.txt文件，内容为My First Buffered File.。

## 参考文档

* [CSDN博文 - Java 7之传统I/O第1篇- 输入输出流基础框架](http://blog.csdn.net/mazhimazh/article/details/17839091)
* [Java SE官方API文档 - java.io.FileInputStream](http://docs.oracle.com/javase/7/docs/api/java/io/FileInputStream.html)
* [Java SE官方API文档 - java.io.FileOutputStream](http://docs.oracle.com/javase/7/docs/api/java/io/FileOutputStream.html)
* [Java SE官方API文档 - java.io.BufferedInputStream](http://docs.oracle.com/javase/7/docs/api/java/io/BufferedInputStream.html)
* [Java SE官方API文档 - java.io.BufferedOutputStream](http://docs.oracle.com/javase/7/docs/api/java/io/BufferedOutputStream.html)
* [Java SE官方API文档 - java.io.FileReader](http://docs.oracle.com/javase/7/docs/api/java/io/FileReader.html)
* [Java SE官方API文档 - java.io.FileWriter](http://docs.oracle.com/javase/7/docs/api/java/io/Filewriter.html)
* [Java SE官方API文档 - java.io.BufferedReader](http://docs.oracle.com/javase/7/docs/api/java/io/BufferedReader.html)
* [Java SE官方API文档 - java.io.BufferedWriter](http://docs.oracle.com/javase/7/docs/api/java/io/BufferedWriter.html)

# 枚举和泛型

## 一、实验简介

在本章我们将学习Java中的枚举类型和

本章知识点

* 枚举类型
* 泛型

## 二、认识枚举类型

在有些情况下，变量的取值是限定在一个有限的集合内的。比如说，性别只有男和女，星期也只有星期一、星期二、星期三、…、星期天等。你可以将这些元素编码为1、2、3之类的，但这么做可能会让你的程序存在一些隐患，比如0通常还有false的含义。

从JDK 5.0开始，开发者可以自定义枚举类型，它的含义如下：

枚举类型包含有限个命名的值。

比如将星期声明为枚举类型的变量，你可以这样做：

enum Week { MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY };

这个声明定义的类型实际上是一个类，其中有7个实例。

和所有的类一样，你可以枚举类型中添加其构造器，以及一些域和方法。所以上面的声明你可以再补充一些内容：

package com.shiyanlou.course

public enum Week {

MONDAY("MON"), TUESDAY("TUES"), WEDNESDAY("WED"), THURSDAY("THUR"),

FRIDAY("FRI"), SATURDAY("SAT"), SUNDAY("SUN"); //这五句话相当于是利用构造方法实例化了类，构造方法下面创建。

//必须要先定义枚举类型，才能定义方法

private String abbr;

//声明的一个字符串，用于保存缩写信息

private Week(String abbr) { //该构造方法使得Monday(“Mon”)中Mon为Monday的缩写。

this.abbr = abbr;

}

//声明的一个私有构造方法，必须是私有构造函数，该方法只能在构造枚举常量时被调用，可以使枚举元素具有指定的缩写信息，这个构造函数怎么定义，上面的Monday格式就怎么写，从而week的属性做相应的改变

public String getAbbr() {

return abbr;

}

//通过该方法获得枚举元素的缩写

}

所有的枚举都是这个week的实例。

所有的枚举类型都继承了java.lang.Enum类，所以Enum类的一些方法也适用于这些枚举类型。例如：

* equals(Object other)，当指定对象等于此枚举常量时，返回 true。
* toString()，返回枚举常量的名称，它包含在声明中。
* valueOf(Class enumType, String name)，返回带指定名称的指定枚举类型的枚举常量。

下面通过一个例子来学习枚举元素的基本特性，请在Eclipse中新建项目LearnEnum，创建包com.shiyanlou.course，在这个包里面创建一个上述的Week类（代码在上方）和EnumPractice类。我们在代码的注释中继续来讲解它们的用法。

Week类你应该已经写好了，下面我们在EnumPractice类中添加一些代码：

package com.shiyanlou.course;

public class EnumPractice {

public static void main(String[] args) {

for(Week week : Week.values()) {

System.out.println("The order of " + week + " is " + week.ordinal());

//使用ordinal()方法查看枚举元素的顺序

System.out.println("Compare to MONDAY : " + week.compareTo(Week.MONDAY));

//用compareTo()方法来将当前枚举元素与Week.MONDAY比较

System.out.println("Equal to MONDAY ? " + week.equals(Week.MONDAY));

//用equals()方法来将当前枚举元素与Week.MONDAY比较

System.out.println("Equal to MONDAY by == ? " + ( week == Week.MONDAY ));

//用比较运算符==来将当前枚举元素与Week.MONDAY比较

//从这里你应该可以看到，可以直接使用==来比较两个枚举元素

System.out.println("Name : " + week.name());

//使用name()方法获得枚举元素的名称

System.out.println("Abbreviation : " + week.getAbbr());

//使用自定义的getAbbr()方法输出星期的缩写

System.out.println("-------------------");

//在每次循环快要结束时，打印一条分割线用于区分

}

}

}

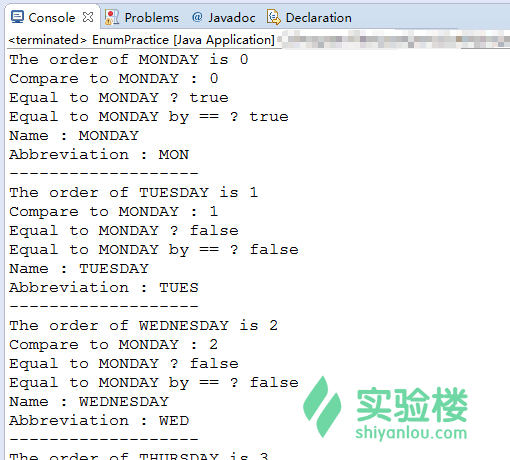
这段程序中的for循环使用了下列条件格式：

for (循环变量类型 循环变量名称 : 要被遍历的对象) {

循环体

}

检查一下代码，点击编译并运行，在控制台输出信息中再来理解一下各个方法的含义。这里我们截取了部分结果，如下图所示：



枚举就相当于定义类，只是这种类有一些限制：

实例化在类定义的内部，且不是通过new关键字的实现

构造方法必须是私有的

package com.shiyanlou.test;  
  
/\*\*  
 \* Created by DaiYan on 2017/2/4.  
 \*/  
public enum Country {  
 China("中国","China","北京"),  
 USA("美国","America","华盛顿"),  
 UK("英国","Britan","伦敦");  
 //上面内容是调用下面的构造方法来对类Country的实例化  
 private String zhName;  
 private String enName;  
 private String capital;  
  
 private Country(String zhName,String enName,String capital){  
 this.zhName=zhName;  
 this.enName=enName;  
 this.capital=capital;  
 }  
  
 public String getZhName(){  
 return this.zhName;  
 }  
  
 public String getEnName(){  
 return this.enName;  
 }  
  
 public String getCapital(){  
 return this.capital;  
 }  
}

## 三、认识泛型

泛型是程序设计语言的一种特性，它允许程序员在编写代码时定义一些可变的部分，而这些部分必须在使用前作出指明。

Java 泛型的参数只能代表类，不能代表个别对象。在对参数命名时，通常使用单个大写字母，例如对于任意类型可以使用大写字母T。

所谓的泛型类就是包含一个或多个类型参数的类。定义泛型类时只需要在类名后面加上<和>并在尖括号内指明类型参数即可。具体的规则如下：

1. 泛型的类型参数只能是类类型（包括自定义类），不能是基本数据类型。
2. 同一种泛型可以对应多个版本（因为参数类型是不确定的），不同版本的泛型类实例是不兼容的。
3. 泛型的类型参数可以有多个。例如K和V。
4. 泛型的参数类型可以使用extends语句。例如。习惯上称为“有界类型”。
5. 泛型的参数类型还可以是通配符类型。例如Class<?> classType = Class.forName("java.lang.String");。

我们用泛型在Java中实现一个具有栈的基本功能的类。

请在Eclipse中新建项目GenericStack，创建包com.shiyanlou.course。我们将会定义两个类，分别是栈结构的泛型类GenericStack和测试类GenericStackTest。

首先在GenericStack类中添加代码。我们需要做的是选择基本的容器来制作这个栈（这里选择了集合类中的LinkedList作为容器），并为栈添加入栈和出栈的方法。当然，你还可以使这个栈具有更易用的特性，为其添加判断空栈的方法。主要的代码如下：

package com.shiyanlou.course;

import java.util.LinkedList;

public class GenericStack<T> {

//定义了一个参数类型为T的泛型类

private LinkedList<T> stackContainer = new LinkedList<T>();

//使用T类型的LinkedList（链表）作为栈的容器，容器定义也是泛型定义

public void push(T t) {

//自定义了一个入栈的方法，其中调用了addFirst()方法

//addFirst()方法可以在栈的顶端加入元素

stackContainer.addFirst(t);

}

public T pop() {

//定义了一个出栈的方法，其中调用了removeFirst()方法

//removeFirst()方法可以将栈顶端的元素移除并取出作为返回值

return stackContainer.removeFirst();

}

public boolean isEmpty() {

//定义了一个判断是否是空栈的方法，调用了LinkedList本身的isEmpty(0方法

return stackContainer.isEmpty();

}

}

接着我们在测试类GeniricStackTest中来验证一下我们对于泛型类的定义是否正确。代码片段如下：

package com.shiyanlou.course;

public class GenericStackTest {

public static void main(String[] args) {

GenericStack<String> stack = new GenericStack<String>();

//我们在创建GenericStack对象时填入了String类型，表明这个栈里面只能存放字符串元素

//声明了一个GenericStack对象，名字为stack

System.out.println("Now add some words into stack.");

stack.push("ShiYanLou");

System.out.println("word - 'ShiYanLou' added.");

stack.push("OnlineLearning");

System.out.println("word - 'OnlineLearning' added.");

//添加两个元素到栈里面去

//Q：现在哪个词在栈顶？

System.out.println("Now pop words from stack.");

while(!stack.isEmpty()) {

//只要栈不为空，就取出其中的元素并打印出来

String s = (String)stack.pop();

//这里为了确保下一步执行的正确，用到了强制类型转换

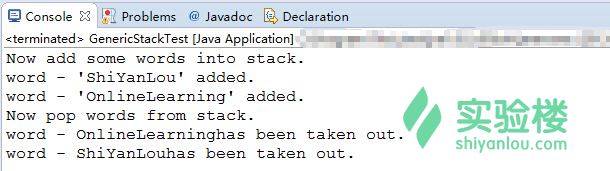
System.out.println("word - " + s + "has been taken out.");

}

}

}

检查一下代码是否有误，点击编译并运行，在控制台可以看到下面这样的输出结果：



当然你也可以自定义这些栈中的信息。

至此，你应该学会了泛型类的定义、声明即其中包含的方法的使用。

## 四、实验总结

本章我们学习了枚举类型和泛型。古人有云：

“此类繁多，不能枚举。” ——清 李渔·《闲情偶寄》

再对比一下Java中枚举的定义，详细你应该有更深的体会。

然而泛型的使用还与反射机制有关系，希望你查阅资料进行更加深入的学习。总之，泛型的好处是在编译的时候检查类型安全，并且所有的强制转换都是自动和隐式的，能够提高代码的重用率。

## 五、作业

请仿照上面泛型类的例子，自定义一个数组的泛型类GenericArray，要求：

1. 具有基本的数组特征；
2. 具有在指定位置插入元素的方法put(int index, T item)；
3. 能够获得指定位置元素的方法get(int index)；

package com.shiyanlou.test;  
  
import java.util.ArrayList;  
  
/\*\*  
 \* Created by DaiYan on 2017/2/4.  
 \*/  
public class GenericArray<T> {  
  
 //定义一个容器  
 private ArrayList<T> container;  
  
 public GenericArray(){  
 this.container=new ArrayList<T>();  
 }  
  
 public T get(int index){  
 return this.container.get(index);  
 }  
  
 public void put(T value){  
 container.add(value);  
 }  
  
 public void rm(int index){  
 container.remove(index);  
 }  
  
 public int length(){  
 return this.container.size();  
 }  
  
 public boolean isEmpty(){  
 return this.container.isEmpty();  
 }  
  
}

## 参考文档

* [Jinuxwu的专栏 - Java泛型详解](http://blog.csdn.net/jinuxwu/article/details/6771121)
* [Java SE官方API文档 - java.lang.Enum](http://docs.oracle.com/javase/7/docs/api/java/lang/Enum.html)