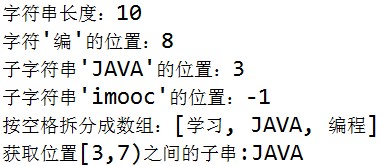
Java异常类的祖先类：Throwable

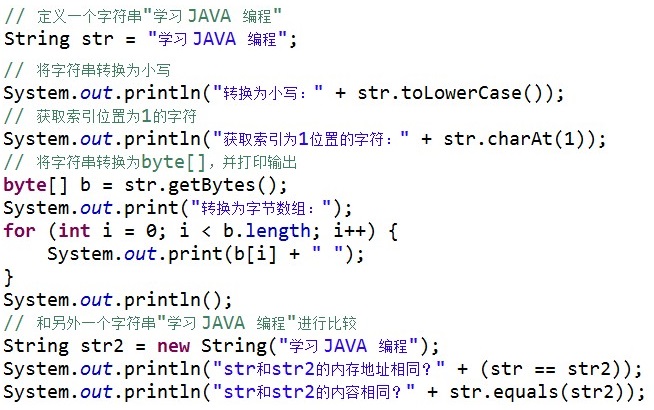
子类：①error（虚拟机溢出，线程死锁）②exception（非检查异常runtimeexception和检查异常）

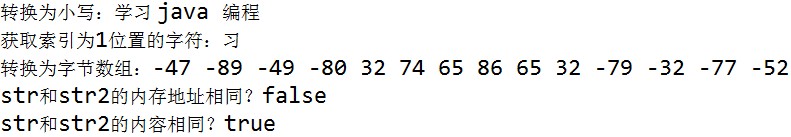


字符串的应用规则：

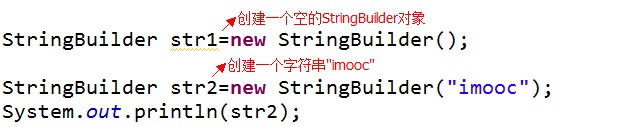




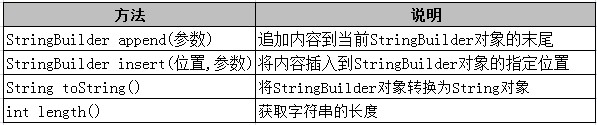




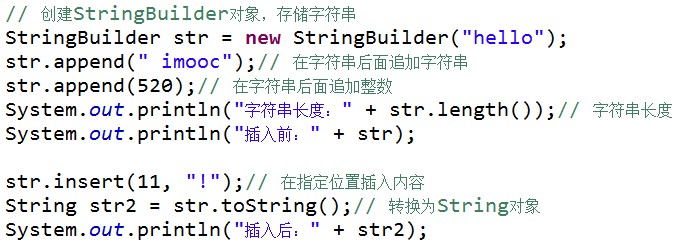
为了避免程序运行时会额外创建一个对象，保存 字符串。当频繁操作字符串时，就会额外产生很多临时变量。使用 StringBuilder 或 StringBuffer 就可以避免这个问题。至于 StringBuilder 和StringBuffer ，它们基本相似，不同之处，StringBuffer 是线程安全的，而 StringBuilder 则没有实现线程安全功能，所以性能略高。因此一般情况下，如果需要创建一个内容可变的字符串对象，应优先考虑使用 StringBuilder 类。



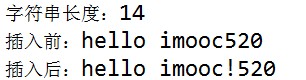
StringBuilder 类提供了很多方法来操作字符串：



例如：在下面的示例代码中，创建了 StringBuilder 对象，用来存储字符串，并对其做了追加和插入操作。这些操作修改了 str 对象的值，而没有创建新的对象，这就是 StringBuilder 和 String 最大的区别。



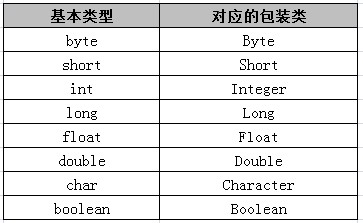
运行结果：



**Java 中的包装类**

相信各位小伙伴们对基本数据类型都非常熟悉，例如 int、float、double、boolean、char 等。基本数据类型是不具备对象的特性的，比如基本类型不能调用方法、功能简单。。。，为了让基本数据类型也具备对象的特性， Java 为每个基本数据类型都提供了一个包装类，这样我们就可以像操作对象那样来操作基本数据类型。

基本类型和包装类之间的对应关系：

[](http://img.mukewang.com/53abd8450001911703630223.jpg)

包装类主要提供了两大类方法：

1. 将本类型和其他基本类型进行转换的方法

2. 将字符串和本类型及包装类互相转换的方法

**Java 中基本类型和包装类之间的转换**

基本类型和包装类之间经常需要互相转换，以 Integer 为例（其他几个包装类的操作雷同哦）：

[http://img.mukewang.com/53abe23d0001da4506190047.jpg](http://img.mukewang.com/53abe23d0001da4506190047.jpg)

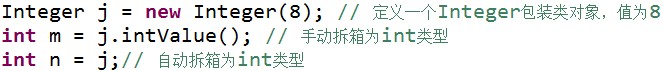
在 JDK1.5 引入自动装箱和拆箱的机制后，包装类和基本类型之间的转换就更加轻松便利了。

那什么是装箱和拆箱呢？我们分别来看下

**装箱：**把基本类型转换成包装类，使其具有对象的性质，又可分为手动装箱和自动装箱

[](http://img.mukewang.com/53abe4590001373504390074.jpg)

**拆箱：**和装箱相反，把包装类对象转换成基本类型的值，又可分为手动拆箱和自动拆箱

[](http://img.mukewang.com/53abe5210001652606630071.jpg)

**Java 中基本类型和字符串之间的转换**

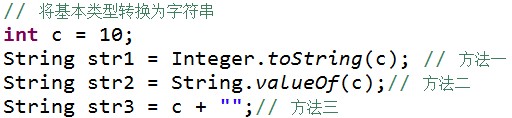
在程序开发中，我们经常需要在基本数据类型和字符串之间进行转换。

其中，基本类型转换为字符串有三种方法：

1. 使用包装类的 toString() 方法

2. 使用String类的 valueOf() 方法

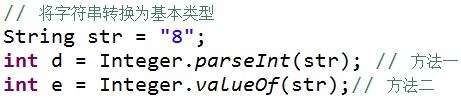
3. 用一个空字符串加上基本类型，得到的就是基本类型数据对应的字符串

[](http://img.mukewang.com/53abea61000151e105120118.jpg)

再来看，将字符串转换成基本类型有两种方法：

1. 调用包装类的 parseXxx 静态方法

2. 调用包装类的 valueOf() 方法转换为基本类型的包装类，会自动拆箱

[](http://img.mukewang.com/53abeaad000109af04610098.jpg)

**使用 Date 和 SimpleDateFormat 类表示时间**

在程序开发中，经常需要处理日期和时间的相关数据，此时我们可以使用 java.util 包中的 Date 类。这个类最主要的作用就是获取当前时间，我们来看下 Date 类的使用：

[http://img.mukewang.com/53b2125e0001429505470051.jpg](http://img.mukewang.com/53b2125e0001429505470051.jpg)

使用 Date 类的**默认无参构造方法**创建出的对象就代表**当前时间**，我们可以直接输出 Date 对象显示当前的时间，显示的结果如下：

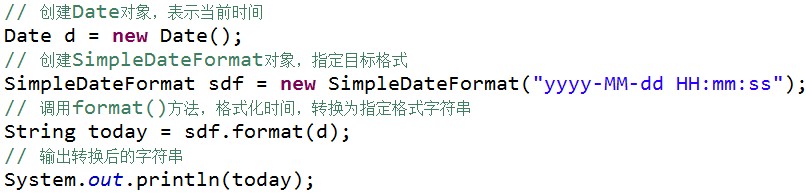
[http://img.mukewang.com/53b21db0000143ed03720025.jpg](http://img.mukewang.com/53b21db0000143ed03720025.jpg)

其中， Wed 代表 Wednesday (星期三)， Jun 代表 June (六月)， 11 代表 11 号， CST 代表 China Standard Time (中国标准时间，也就是北京时间，东八区)。

从上面的输出结果中，我们发现，默认的时间格式不是很友好，与我们日常看到的日期格式不太一样，如果想要按指定的格式进行显示，如 2014-06-11 09:22:30 ，那该怎么做呢？

此时就到了 java.text 包中的 SimpleDateFormat 类大显身手的时候了！！可以使用 SimpleDateFormat 来对日期时间进行格式化，如可以将日期转换为指定格式的文本，也可将文本转换为日期。

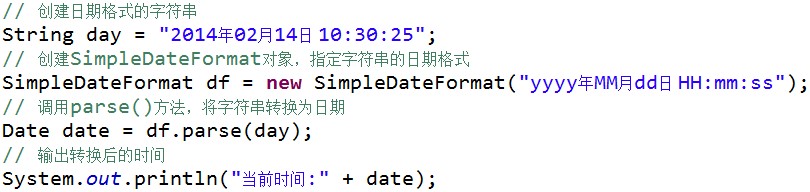
1. 使用 format() 方法将日期转换为指定格式的文本

[](http://img.mukewang.com/53b22d780001e5f808060193.jpg)

代码中的 “yyyy-MM-dd HH:mm:ss” 为预定义字符串， yyyy 表示四位年， MM 表示两位月份， dd 表示两位日期， HH 表示小时(使用24小时制)， mm 表示分钟， ss 表示秒，这样就指定了转换的目标格式，最后调用 **format()**方法将时间转换为指定的格式的字符串。

运行结果：**2014-06-11  09:55:48**

2. 使用 parse() 方法将文本转换为日期

[](http://img.mukewang.com/53b22ddf00012edf08100192.jpg)

代码中的 “yyyy年MM月dd日 HH:mm:ss” 指定了字符串的日期格式，调用 **parse()**方法将文本转换为日期。

运行结果： [http://img.mukewang.com/53b22e4f000108b104570025.jpg](http://img.mukewang.com/53b22e4f000108b104570025.jpg)

一定要注意哦：

1、 调用 SimpleDateFormat 对象的 parse() 方法时可能会出现转换异常，即 ParseException ，因此需要进行异常处理

2、 使用 Date 类时需要导入 java.util 包，使用 SimpleDateFormat 时需要导入 java.text 包

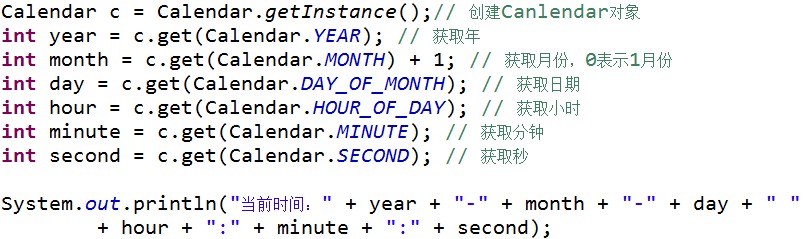
<http://www.imooc.com/code/2335>更改时间格式

**Calendar 类的应用**

Date 类最主要的作用就是获得当前时间，同时这个类里面也具有设置时间以及一些其他的功能，但是由于本身设计的问题，这些方法却遭到众多批评，不建议使用，更推荐使用 Calendar 类进行时间和日期的处理。

java.util.Calendar 类是一个抽象类，可以通过调用 getInstance()静态方法获取一个 Calendar 对象，此对象已由当前日期时间初始化，即默认代表当前时间，如 Calendar c = Calendar.getInstance();

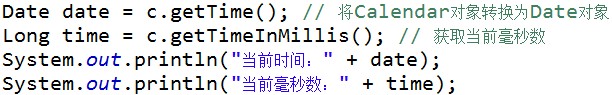
那么如何使用 Calendar 获取年、月、日、时间等信息呢？我们来看下面的代码：

[](http://img.mukewang.com/53b23219000153b608030239.jpg)

其中，调用 Calendar 类的 getInstance() 方法获取一个实例，然后通过调用 get() 方法获取日期时间信息，参数为需要获得的字段的值， Calendar.Year 等为 Calendar 类中定义的静态常量。

运行结果： [http://img.mukewang.com/53b232540001945f03370024.jpg](http://img.mukewang.com/53b232540001945f03370024.jpg)

Calendar 类提供了 getTime() 方法，用来获取 Date 对象，完成 Calendar 和 Date 的转换，还可通过 getTimeInMillis() 方法，获取此 Calendar 的时间值，以毫秒为单位。如下所示：

[](http://img.mukewang.com/53b232b200015b8306120095.jpg)

运行结果：

[http://img.mukewang.com/53b232d2000108bf04650052.jpg](http://img.mukewang.com/53b232d2000108bf04650052.jpg)

**使用 Math 类操作数据**

Math 类位于 java.lang 包中，包含用于执行基本数学运算的方法， Math 类的所有方法都是静态方法，所以使用该类中的方法时，可以直接使用类名.方法名，如： Math.round();

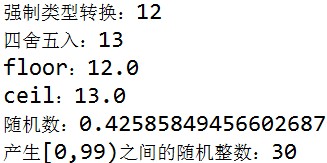
常用的方法：

[](http://img.mukewang.com/53b22ff10001b6d104810124.jpg)

通过案例我们来认识一下他们的使用吧！！

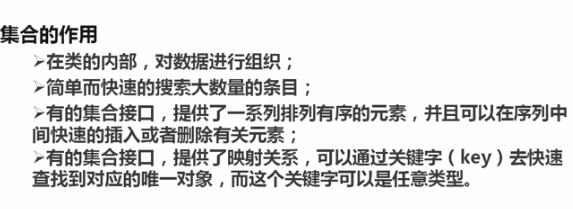
[](http://img.mukewang.com/544ef6360001036e07280313.jpg)

运行结果：

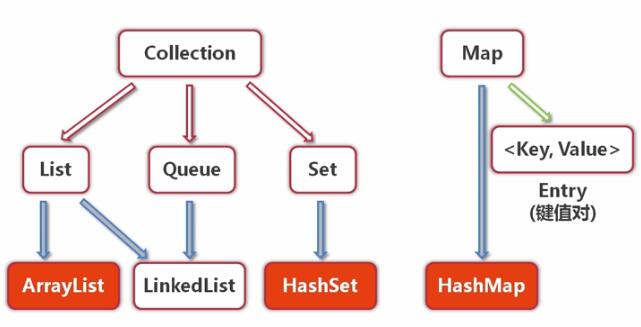
[](http://img.mukewang.com/53b230420001ba8403270163.jpg)

PS： Math 类还提供了许多其他方法，各位小伙伴们可以注意关注 wiki ，查阅更多信息

集合类：一种工具类，就像是容器，存储任意数量具有共同属性的对象



集合的实现类：



# JAVA数组的操作

## 定义一个数组

String [ ] aArray = new String[5];

String [ ] bArray = {“a”, ”b”, ”c”, ”d” ,”e”};

String [ ] aArray = new String[ ]{“a”, ”b”, ”c”, ”d” ,”e”};

第一种定义了一个数组，并且指定了数组的长度，称为动态定义

第二、三种在分配内存空间的同时还做了初始化

## 打印数组中的元素

Int [ ] intArray = {1,2,3,4,5};

String intArrayString = Arrays.toString(intArray); //print directly will print reference value

System.out.println(intArray); //[@7150bd4d

System.out.println(intArrayString); //[1,2,3,4,5]

这里的重点说明了JAVA中数组的引用和值的区别，第三行的直接打印intArray，输出的是乱码，因为intArray仅仅是一个地址引用，第四行则是真正的数组值，因为它经过了Arrays.toString()的转换。注意引用和值的区别。

## 从Array中 创建ArrayList

String [ ] stringArray = {“a”, ”b”, ”c”, ”d” ,”e”};

ArrayList<String> arrayList = new ArrayList<String>(Arrays.asList(stringArray));

System.out.println(arrayList); //[a, b, c, d, e]

ArrayLis是动态链表，我们可以更方便地对ArrayList进行增删改，我们并不需要循环Array将每一个元素加到ArrayList中，用以上的代码就可以简单地实现转换。

## 检查数组中是否包含某一个值

String [ ] stringArray =  {“a”, ”b”, ”c”, ”d” ,”e”};

Boolean b = Arrays.asList(stringArray).contains(“a”);

System.out.println(b); //true

先使用Arrays.asList()将转换成List<String>,这样就可以用动态链表的contains函数来判断是否包含在链表中。

## 连接两个数组

int [ ] intArray =  {1, 2, 3, 4, 5};

int [ ] intArray2 = {6, 7, 8, 9, 10};

int [ ] combindIntArray = ArrayUtils.addAll (intarray , intArray2);

ArrayUtils是Apache提供的数组处理类库，其addAll方法可以很方便地将两个数组连接成一个数组。

## 声明一个数组内链

method ( new String[ ] {“a”, ”b”, ”c”, ”d” ,”e”} );

## 将数组中的元素以字符串的形式输出

//containing the provided list of elements

//Apache common lang

String j = StringUtils.join(new String[ ] {“a”, ”b”, ”c” }, “ ,”);

System.out.println(j); //a,b,c

同样利用StringUtils中的join方法，可以将数组中的元素以一个字符串的形式输出

## 将Array转换成set集合

Set<String> set = new HashSet<String>(Arrays.asList(stringArray));

System.out.println(set); //[d,e,b,c,a]

在JAVA中使用set。可以方便地将需要的类型以集合类型保存在一个变量中，主要应用在显示列表。同样可以先将Array转换成List，然后将List转换成Set

## 数组翻转

int [ ] intArray = {1, 2, 3, 4, 5};

ArrayUtils.reverse(intArray);

System.out.println(Arrays.toString(intArray)); //[5, 4, 3, 2, 1]

依然用到了ArrayUtils

## 从数组中移除一个元素

int [ ] intArray = {1, 2, 3, 4, 5};

int [ ] remove = ArrayUtils.removeElement(intArray, 3);//create a new array

System.out.println(Arrays.toString(remove));

## Int转换为byte

byte [ ] bytes = ByteBuffer.allocate(4).putInt(8).array();

for(byte t:bytes)

{

System.out.format(“0x%x”,t);

}