

程序运行时静态初始化块最先被执行，然后执行普通初始化块，最后才执行构造方法。由于静态初始化块只在类加载时执行一次，所以当再次创建对象 hello2 时并未执行静态初始化块。



### Java 中的成员内部类

**成员内部类的使用方法：**

1、 Inner 类定义在 Outer 类的内部，相当于 Outer 类的一个成员变量的位置，Inner 类可以使用任意访问控制符，如 public 、 protected 、 private 等

2、 Inner 类中定义的 test() 方法可以直接访问 Outer 类中的数据，而不受访问控制符的影响，如直接访问 Outer 类中的私有属性a

3、 定义了成员内部类后，必须使用外部类对象来创建内部类对象，而不能直接去 new 一个内部类对象，即：内部类 对象名 = 外部类对象.new 内部类( );

4、 编译上面的程序后，会发现产生了两个 .class 文件

IMG_256

其中，第二个是外部类的 .class 文件，第一个是内部类的 .class 文件，即成员内部类的 .class 文件总是这样：外部类名$内部类名.class

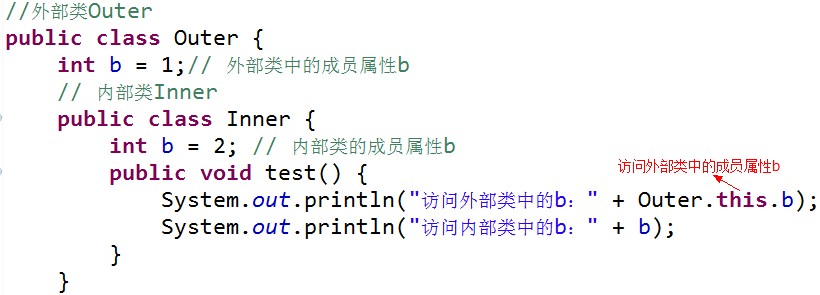
另外，友情提示哦：

1、 外部类是不能直接使用内部类的成员和方法滴



可先创建内部类的对象，然后通过内部类的对象来访问其成员变量和方法。

2、 如果外部类和内部类具有相同的成员变量或方法，内部类默认访问自己的成员变量或方法，如果要访问外部类的成员变量，可以使用 this 关键字。如：



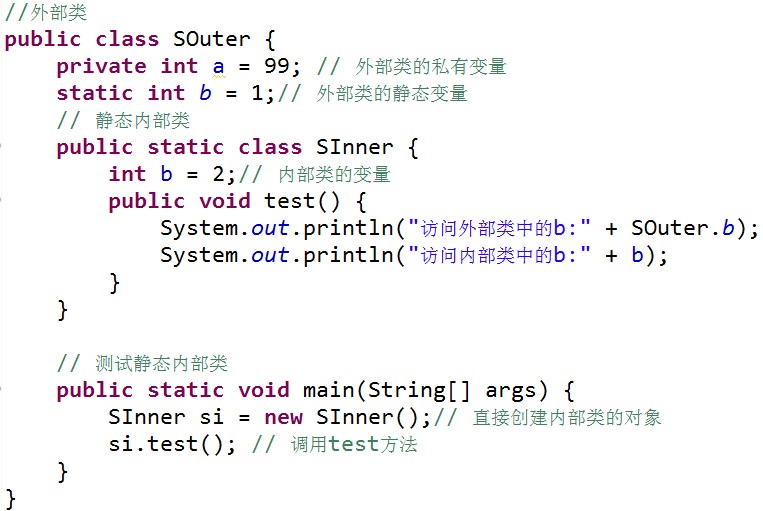
### **Java 中的静态内部类**

静态内部类是 static 修饰的内部类，这种内部类的特点是：

1、 静态内部类不能直接访问外部类的非静态成员，但可以通过 **new 外部类().成员** 的方式访问

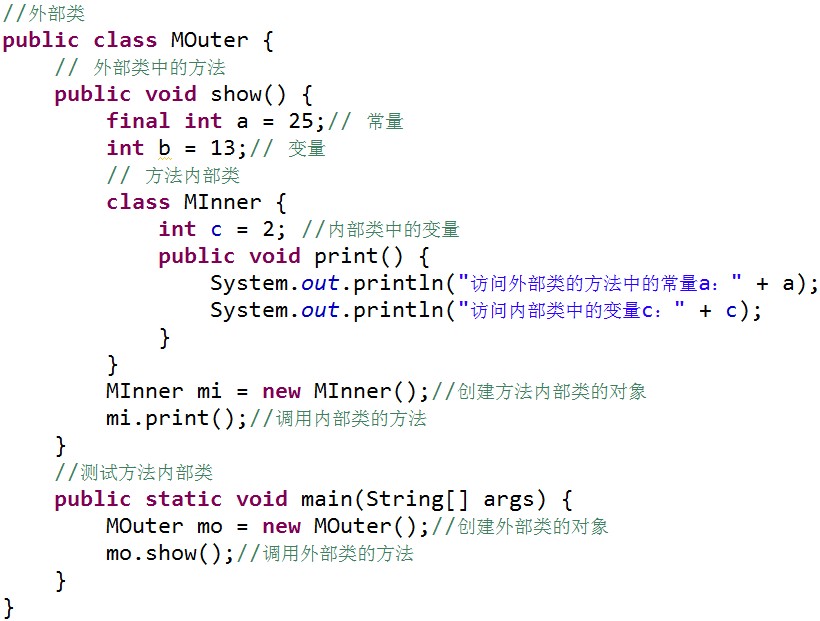
2、 如果外部类的静态成员与内部类的成员名称相同，可通过“类名.静态成员”访问外部类的静态成员；如果外部类的静态成员与内部类的成员名称不相同，则可通过“成员名”直接调用外部类的静态成员

3、 创建静态内部类的对象时，不需要外部类的对象，可以直接创建 **内部类 对象名= new 内部类();**

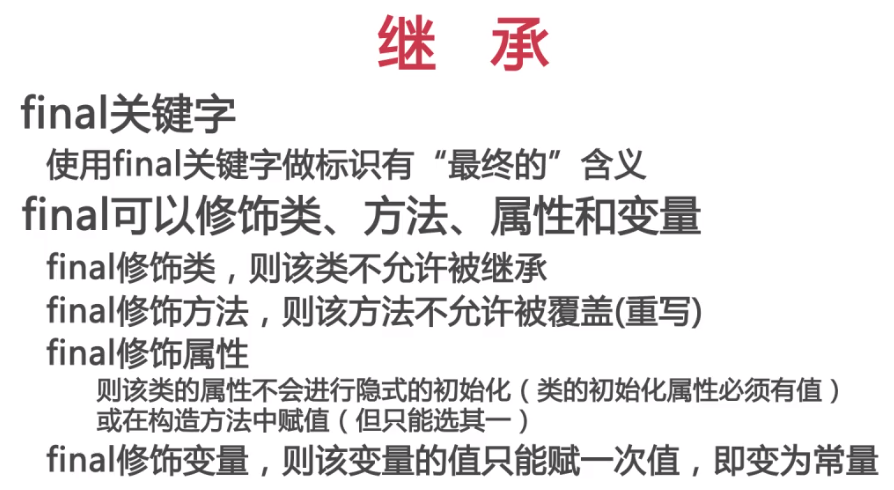


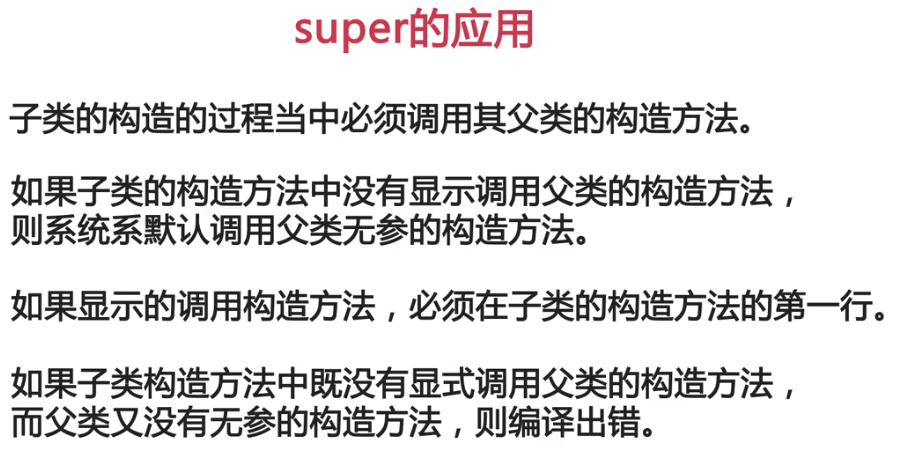
### **Java 中的方法内部类**

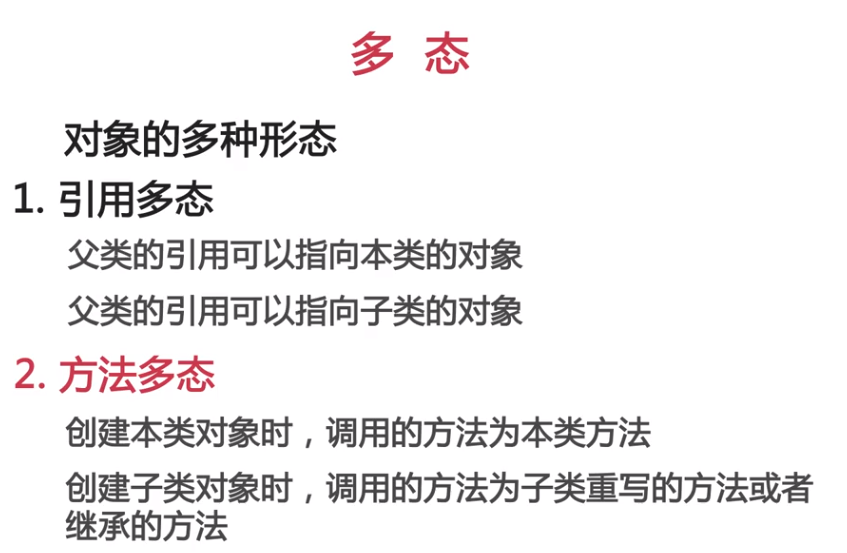
方法内部类就是内部类定义在外部类的方法中，方法内部类只在该方法的内部可见，即只在该方法内可以使用。

[](http://img.mukewang.com/539ea96700013ca708200621.jpg)

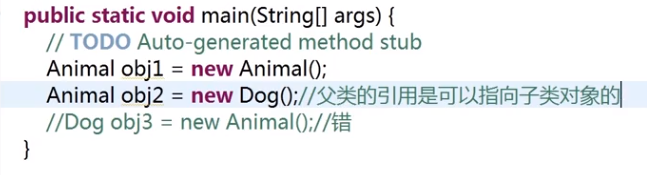
一定要注意哦：由于方法内部类不能在外部类的方法以外的地方使用，因此方法内部类不能使用访问控制符和 static 修饰符。

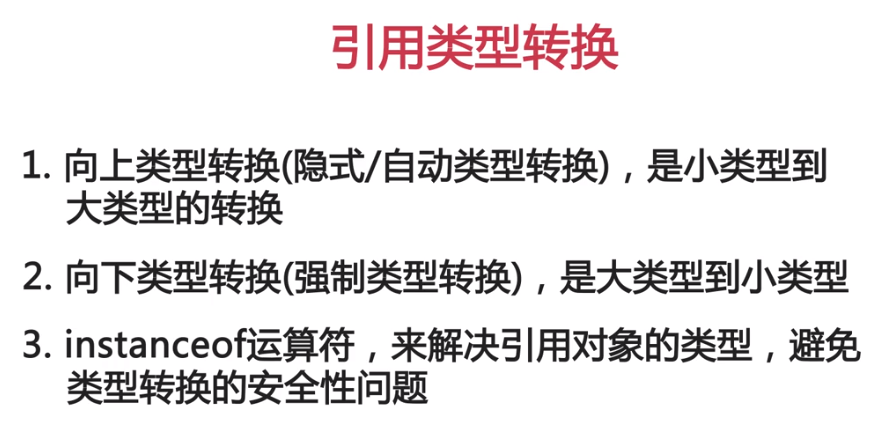


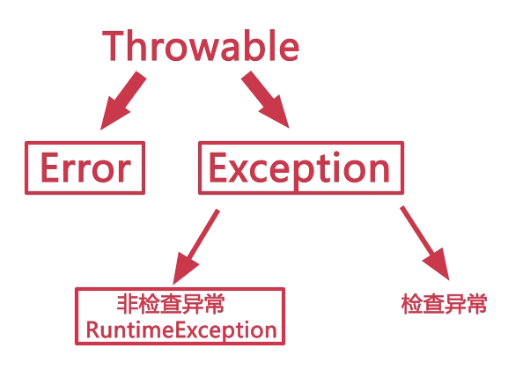


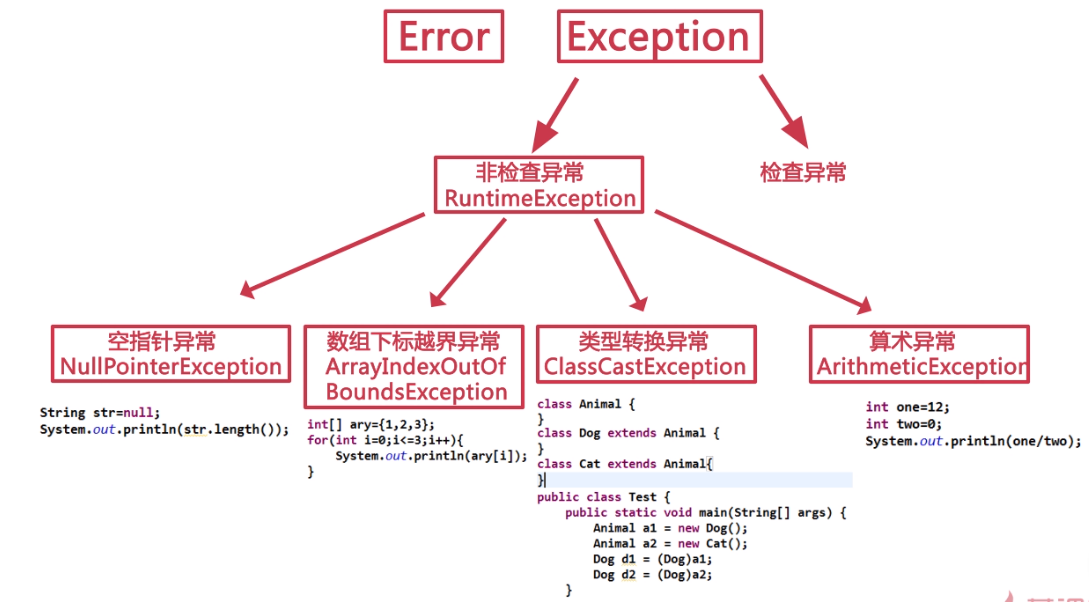


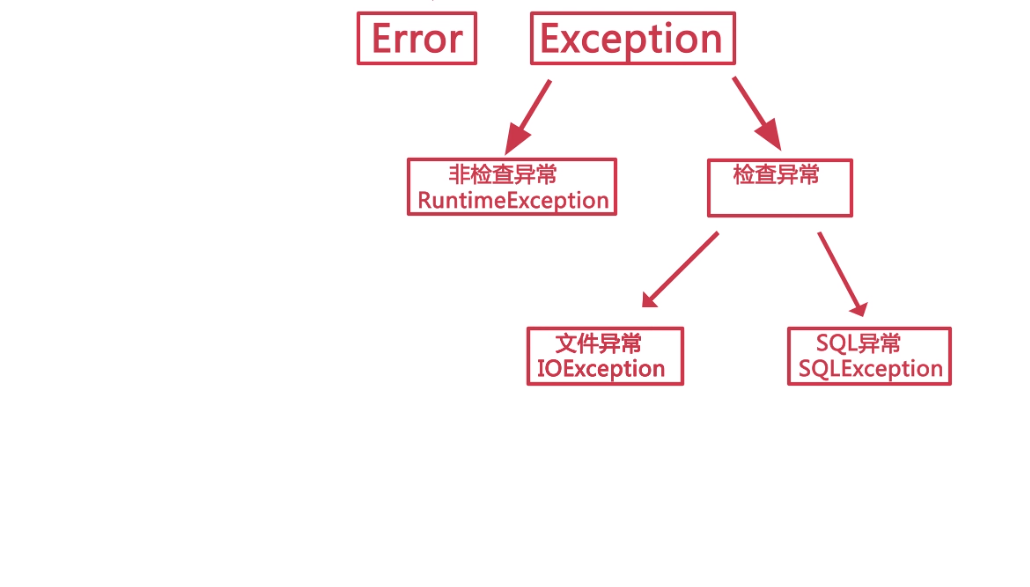
继承是多态的实现基础

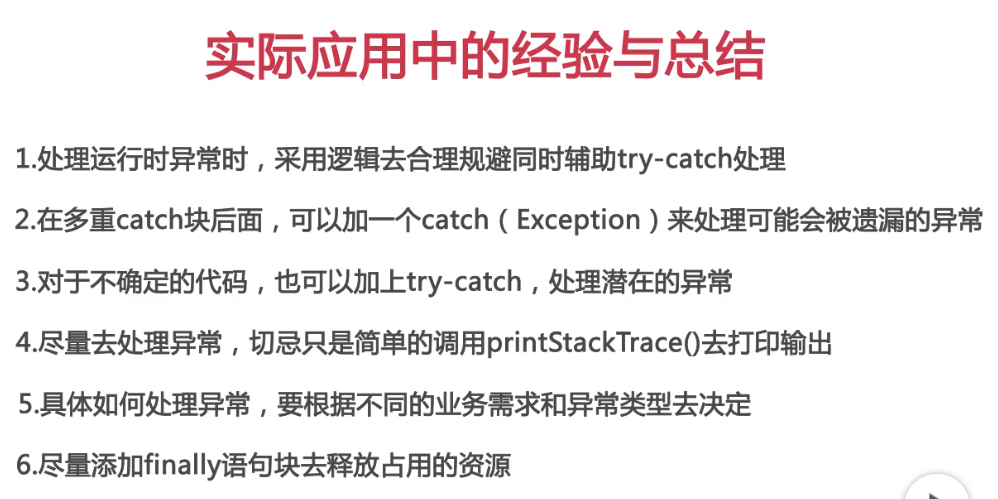


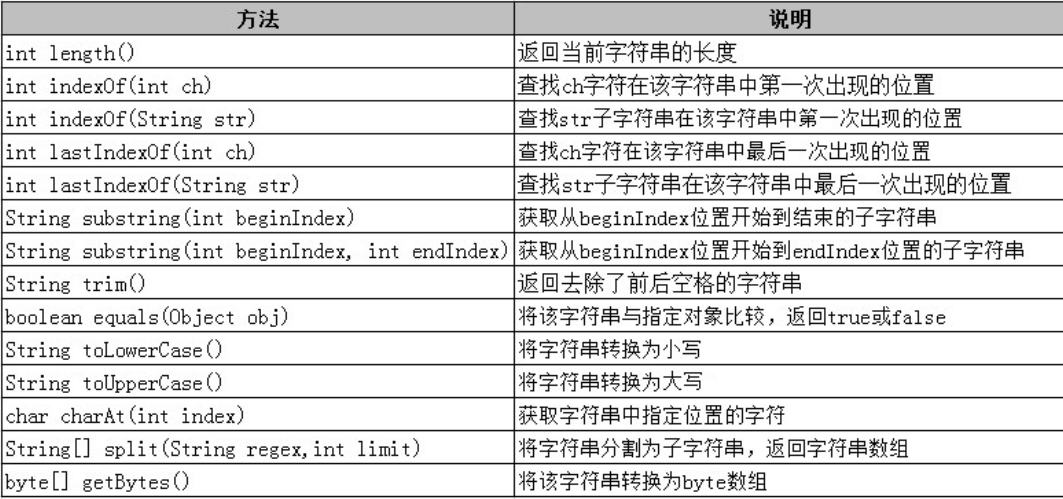


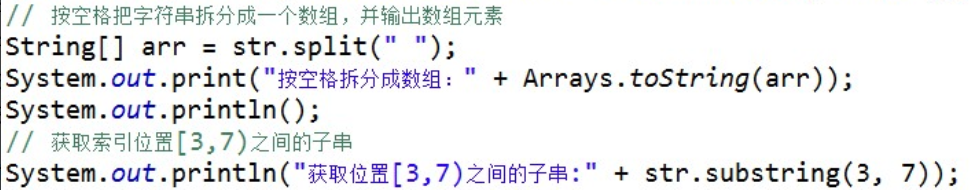






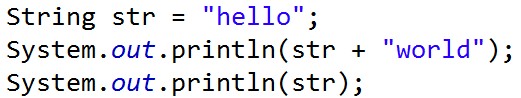






### 认识 Java 中的 StringBuilder 类

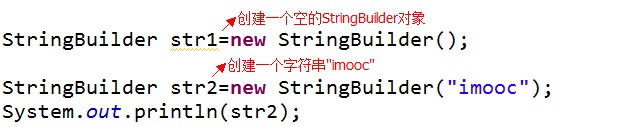
String 类具有是不可变性。如



运行结果：IMG_256

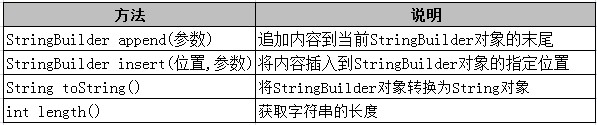
从运行结果中我们可以看到，程序运行时会额外创建一个对象，保存 "helloworld"。当频繁操作字符串时，就会额外产生很多临时变量。使用 StringBuilder 或 StringBuffer 就可以避免这个问题。至于 StringBuilder 和StringBuffer ，它们基本相似，不同之处，StringBuffer 是线程安全的，而 StringBuilder 则没有实现线程安全功能，所以性能略高。因此一般情况下，如果需要创建一个内容可变的字符串对象，应优先考虑使用 StringBuilder 类。

那么如何定义 StringBuilder 类的对象呢？ 我们来看下面的代码：

[](http://img1.sycdn.imooc.com/53a7d1f70001be9d06340127.jpg)

运行结果： imooc

StringBuilder 类提供了很多方法来操作字符串：



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

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

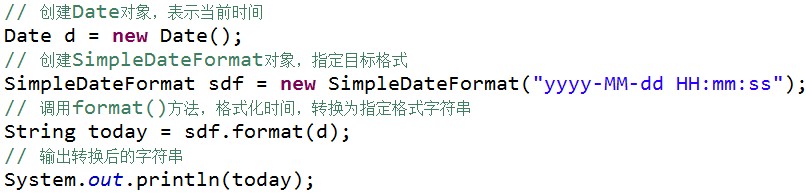
IMG_256

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：

IMG_256

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)。

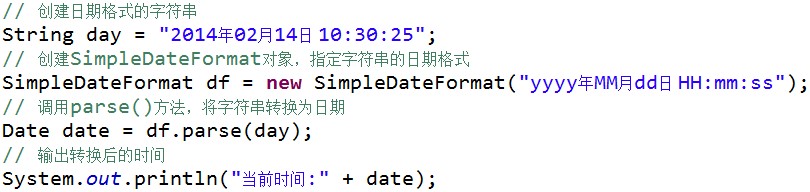
1. 使用 format() 方法将日期转换为指定格式的文本



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

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：

IMG_256

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

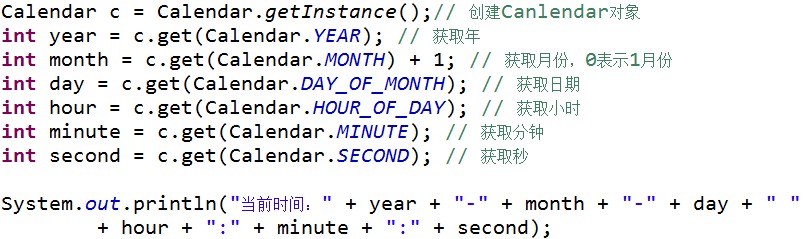
2、 使用 Date 类时需要导入 java.util 包，使用 SimpleDateFormat 时需要导入 java.text 包

### 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();

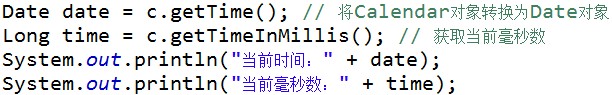
那么如何使用 Calendar 获取年、月、日、时间等信息呢？我们来看下面的代码：

[](http://img1.sycdn.imooc.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： [IMG_257](http://img1.sycdn.imooc.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://img1.sycdn.imooc.com/53b232b200015b8306120095.jpg)

运行结果：

[IMG_259](http://img1.sycdn.imooc.com/53b232d2000108bf04650052.jpg)

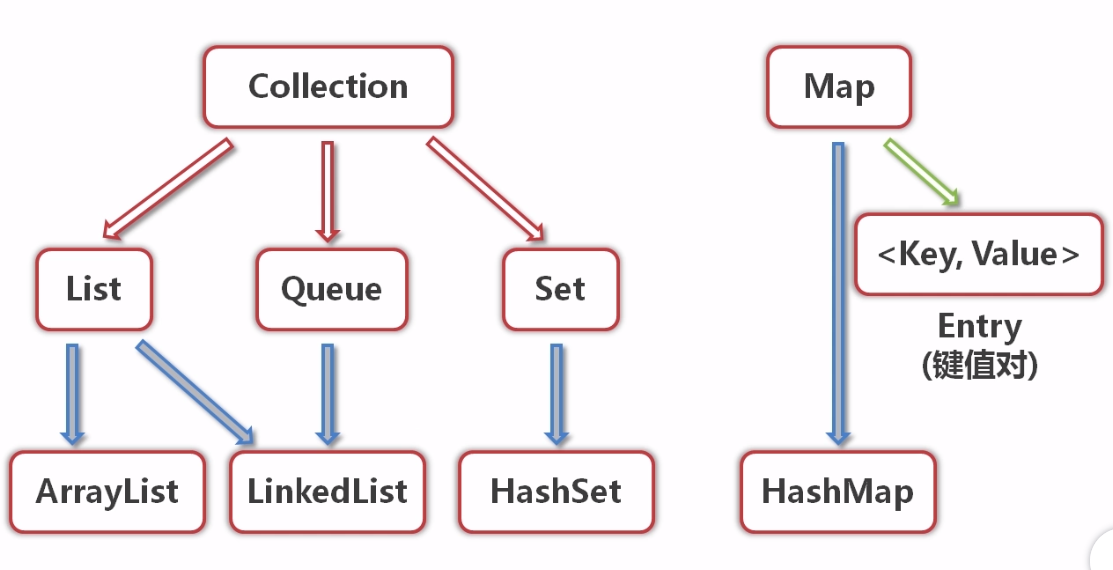
### 使用 Math 类操作数据

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://img1.sycdn.imooc.com/53b22ff10001b6d104810124.jpg)





对象存入集合都变成Object类型，取出时需要类型转换

