复习：

1. 包装类：针对每一个基本类型都提供了对应的类形式 int - Integer, char - Character；自动封箱 --- 指的是将基本类型的变量直接赋值给对应的引用类型的对象。自动封箱底层是调用了对应类身上的valueOf方法，对于整数在自动封箱过程中会有范围(-128~127)判断。自动拆箱是指将引用类型的对象直接赋值给对应的基本类型的变量。自动拆箱在底层是调用了对应对象身上的\*\*\*Value方法。自动封箱/拆箱都是JDK1.5的特性之一。包装类的对象的实际值只要相同则它的哈希码就一定是相同的。

2. 数学类：

Math：最终类。针对基本类型提供了初等数学运算。

BigDecimal：用于精确运算小数的类，但是要求参数以字符串形式传递。

BigInteger：用于运算任意大的整数，要求整数以字符串或者是对应的补码的字节数组形式来传递。

DecimalFormat：用于对数字进行格式化的类。

3. 日期类

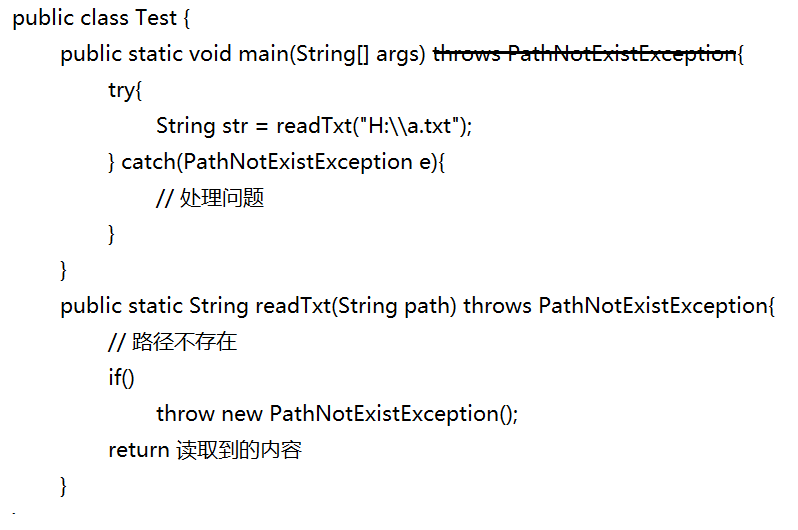
Date：表示日期的类。重点掌握字符串和日期之间的转换 --- SimpleDateFormat：parse将字符串转化为日期，format将日期转化为字符串。

Calendar：表示日历的类。从JDK1.2开始，Java推荐使用Calendar

## 时间包

在JDK1.8中对时间体系进行了全新的详细的划分，划分出来一个详细的时间体系的包 --- java.time

## 异常



异常是Java中一套用于问题的反馈和处理的机制

Throwable --- 异常的顶级父类

Error --- 错误表示合理（语法上还是逻辑上都是成立的）的应用程序中出现了严重的问题，而且这个问题不应该试图捕获 --- 意味着在程序中，错误一旦出现不能处理 --- StackOverflowError, [OutOfMemoryError](mk:@MSITStore:C:\\Users\\admin\\Desktop\\Java8%20API.chm::/java/lang/../../java/lang/OutOfMemoryError.html" \o "class in java.lang)

Exception --- 表示合理的应用程序想要捕获的问题，也因此可以处理。处理方式：要么继续抛出，要么进行捕获处理。

编译时异常：在编译时期就已经出现要求必须处理。

CloneNotSupportedException

UnsupportedEncodingException

ParseException

运行时异常：在编译时期不出现到运行的时候才出现。可以处理可以不处理 --- RuntimeException

ArithmeticException

ArrayIndexOutOfBoundsException

NullPointerException

ClassCastException

StringIndexOutOfBoundsException

NumberFormatException

自定义异常：写一个类继承Exception或者是其子类。如果是继承了RuntimeException及其子类，那么这个时候定义的是一个运行时异常；如果继承的是其他的Exception，那么定义的就是编译时异常

### 异常的捕获方式

A. 如果多个异常的处理方式各不一样，可以使用多个catch分别捕获分别处理

B. 如果所有异常的处理方式都一样，可以捕获这些异常的父类进行统一的处理

C. 如果多个异常进行了分组，那么同一组的异常之间用 | 隔开进行分组处理 --- 从JDK1.7开始

总结：方法的重载和方法的重写

方法的重载和重写都是行为多态。

方法的重载指在同一个类中存在方法名一致而参数列表不同的方法，和修饰符、返回值类型以及异常没有关系。重载本身是一种编译时多态。

方法的重写指在父子类中存在方法签名一致的非静态方法。子类在重写父类方法的时候，子类权限修饰符的范围要大于等于父类权限修饰符的范围。如果父类中的方法的返回值类型是基本类型/void，那么子类重写的方法的返回值类型要与父类一致。如果父类的方法的返回值类型是引用类型，那么子类重写的方法的返回值类型要么和父类一致，要么是父类方法返回值类型的子类。另外，子类重写的方法抛出的编译时异常不能超过父类方法的编译时异常的范围。方法的重写是一种运行时多态。

注意：异常从抛出的地方开始，后续代码停止执行。

finally --- 无论出现异常与否都需要执行一次。

如果在项目的开发期间遇到异常，记录栈轨迹，找异常来源进行改正；如果项目已经上线，记录错误日志，往往跳转错误页面

## 集合 - Collection<E>

存储多个同一类型的数据的容器 --- 大小不固定

<E> - 泛型 - 在集合中的作用是用于表示元素类型。- 由于泛型的限制，集合中只能存储对象。

String[] arr; arr的数据类型是数组，元素类型是String

Collection<String> c; c的数据类型是集合，元素类型是String

5,7, 2 --- Collection<Integer>

Collection<int[]> c; --- 表示集合中存储的元素是数组

### List - 列表

有序（保证元素的存入顺序）的集合 --- 存在了下标，因此能够通过下标来操作列表中的元素

#### ArrayList - 顺序表

底层是依靠的数组来存储的数据。默认初始容量是10，每次扩容是在原来的基础上增加一半，基于右移。