1.Object类

1、Object类的equals方法

public boolean equals(Object obj):

equals方法源码：

public boolean equals(Object obj){

return(this **==** obj);

}

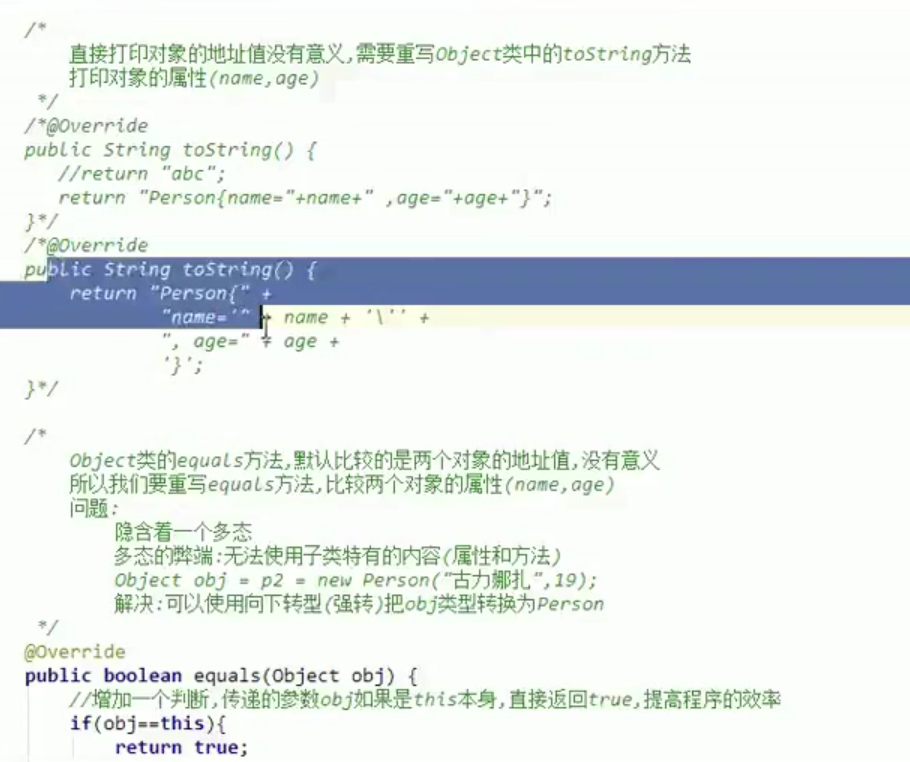
参数： Object obj:可以传递任意的对象

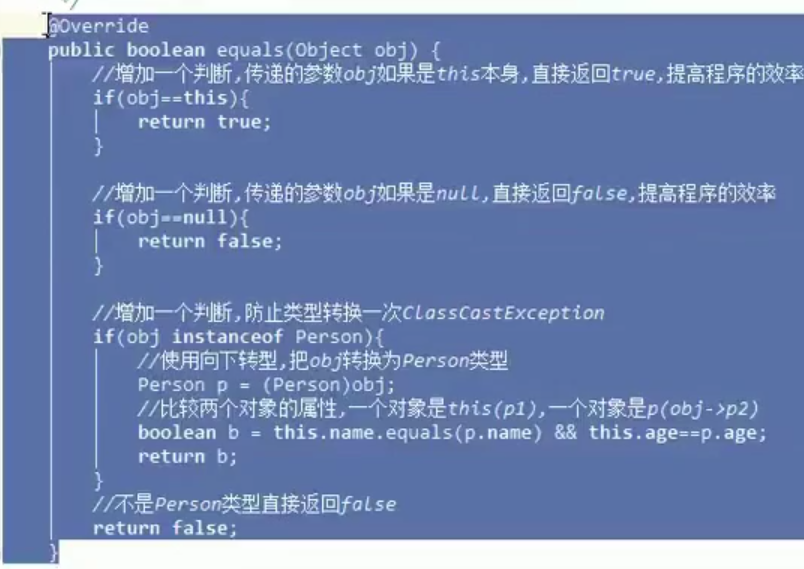
**==**比较运算符,返回一个boolean值

基本数据类型：比较的是值

引用数据类型：比较的是两个对象的地址值

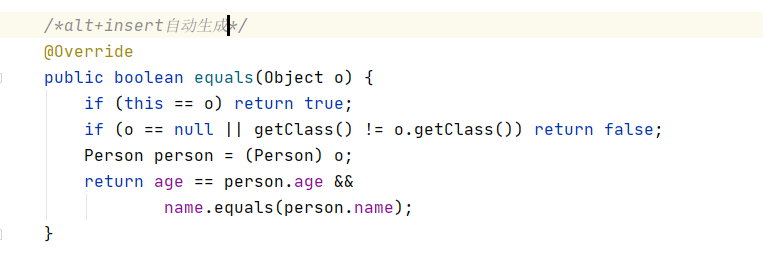
2、重写Object类当中的equals方法





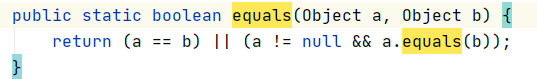




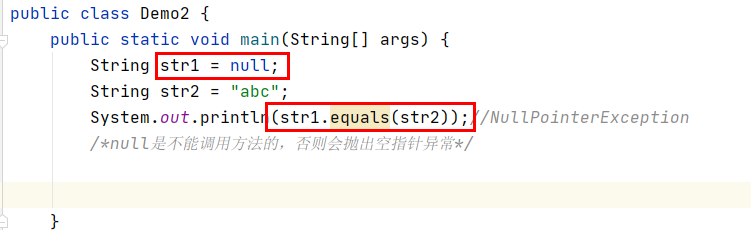


**Objects**中的equals方法（不是Object）

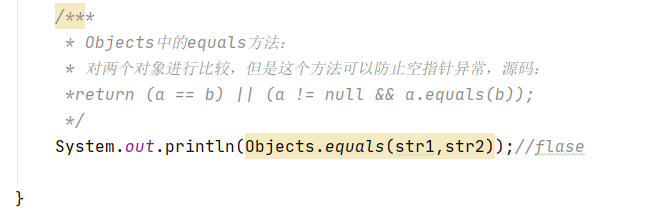
该equals（）的作用就是**防止空指针异常**



1、null是不能调用方法的，否则就会抛出空指针异常，如下：



2.Objects中的equals（Object obj1，Object obj2）；



2.Date类

1、java.util.Date类，表示特定的瞬间，精确到毫秒。一个表示日期和时间的类。

毫秒：1/1000秒。

毫秒值的作用：可以对时间和日期进行计算。

如何计算？

把日期转换为毫秒：

当前日期：2088-01-01

时间原点（0毫秒）：1970年1月1日00:00:00（英国格林威治）

其实就是计算当前日期到时间原点之间一共经历了多少毫秒。

备注： 中国属于东八区，会把时间增加8个小时：1970.01.01.08:00:00

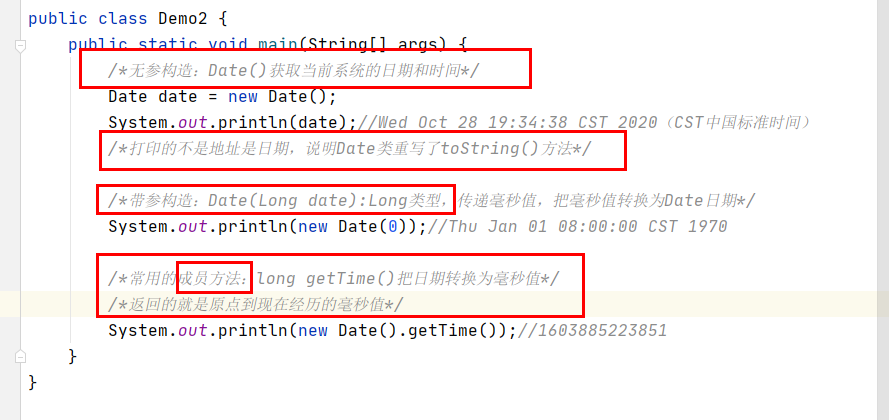


2、Date类的构造方法和成员方法

1、无参构造Date（）

2、带参构造Date（long date）

3、成员方法：public long getTime() [相当于System.currentTimeMillis()]



3.DateFormat类和SimpleDateFormat类

Date（日期） format（格式化） 【日期/时间格式化子类的抽象类】

作用：格式化（也就是日期—>文本） 解析（文本—>日期）

Java.text.DateFormat

public abstract class DateFormat extends Format (抽象类)

成员方法：

1.String format（Date date）：格式化

2.Date parse（String source）：解析

子类：**SimpleDateFormat**

java.text.SimpleDateFormat extends DateFormat

构造方法：

可以空参构造，但是一般使用：SimpleDateFormat(String pattern);

用给定的模式和默认语言环境的日期格式符号构造SimpleDateFormat

参数：String pattern传递指定的模式。



写对应的模式，会吧模式替换成对应的日期和时间。

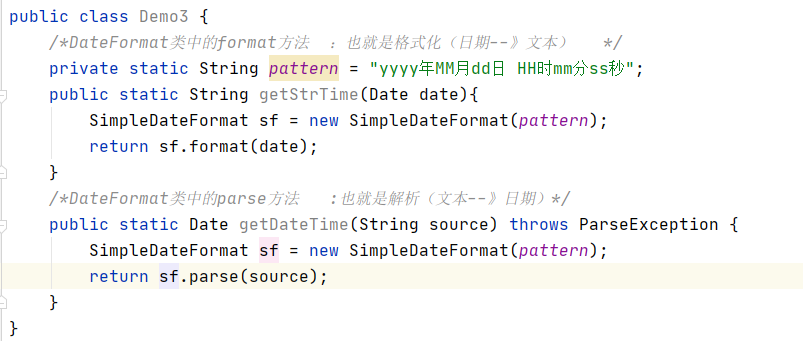
**yyyy-MM-dd HH:mm:ss**

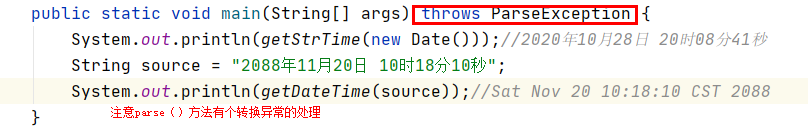
**注意事项：**

模式中的字母不能更改，但是连接的符号可以改变。

**yyyy年MM月dd日 HH时mm分ss秒**

**两种方法的使用：**





**注意：** public Date **parse**（String source） throws ParseException

parse方法声明了一个异常叫ParseException

如果字符串和构造方法的模式不一样，那么程序就会抛出这个异常。

调用一个抛出异常的方法，就必须得处理这个异常，要么**throws**继续抛出这个异常，要么try catch自己处理。

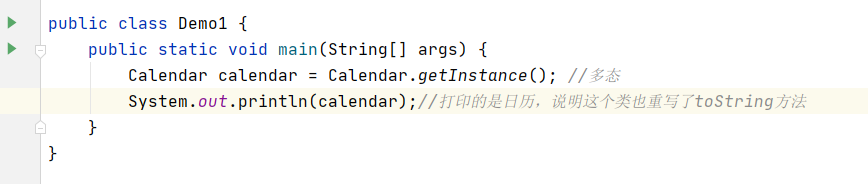
3.Calendar类

Java.util.Calendar类：日历类

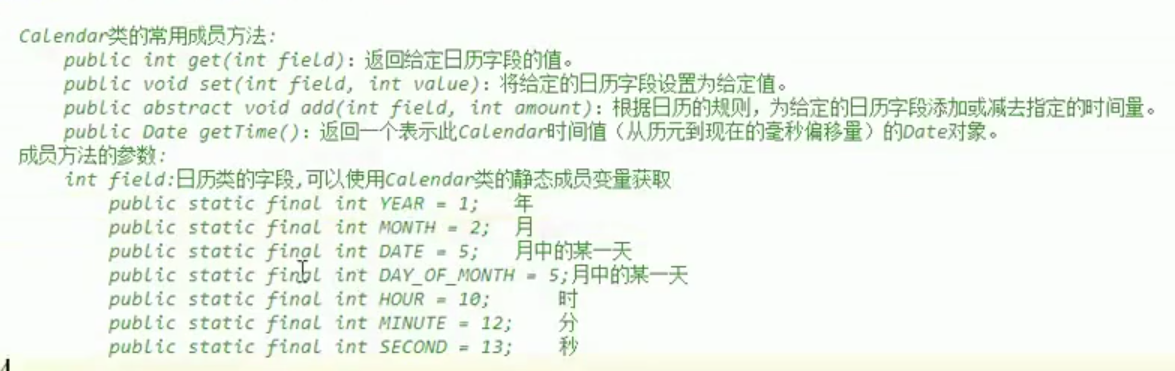
1、Calendar类是一个**抽象类**，里边提供了很多操作日历字段的方法(YEAR,MONTH,DAY\_OF\_MONTH,HOUR）

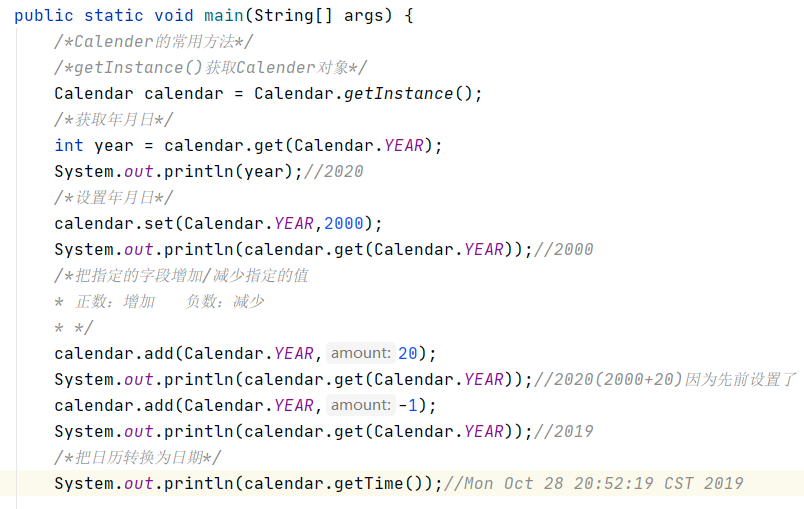
2、Calendar类无法直接创建对象使用，里面有一个静态方法叫getInstance（）；该方法**返回了**Calender对象的**子类对象。**

static **Calender** getInstance（）使用默认时区和语言环境获得一个日历。



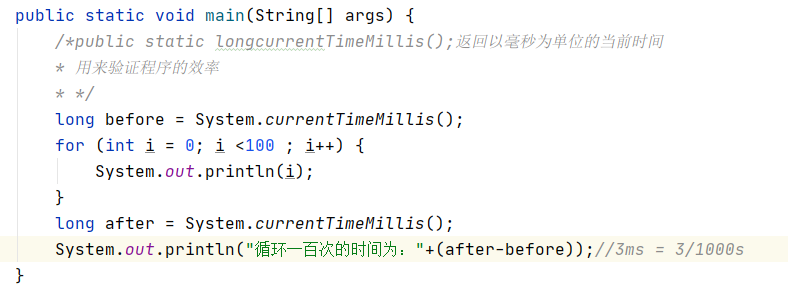
3、常用方法



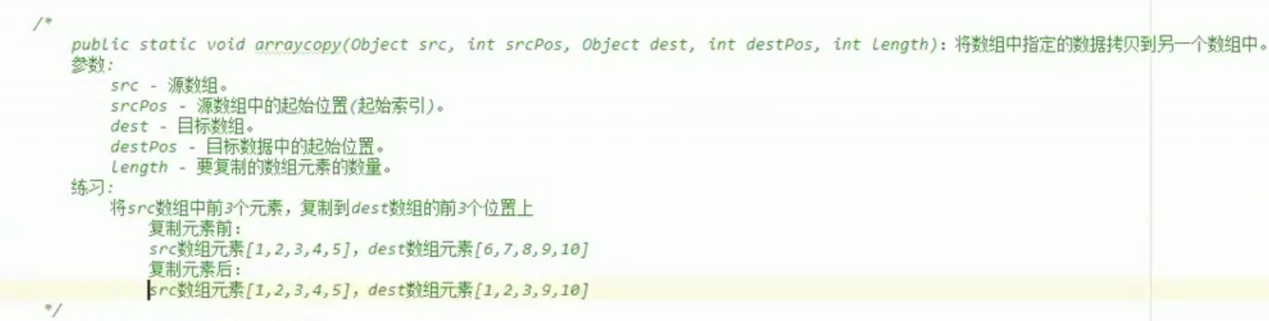


3.System类

1、public static long currentTimeMillis();【用来查看程序的效率】



2、





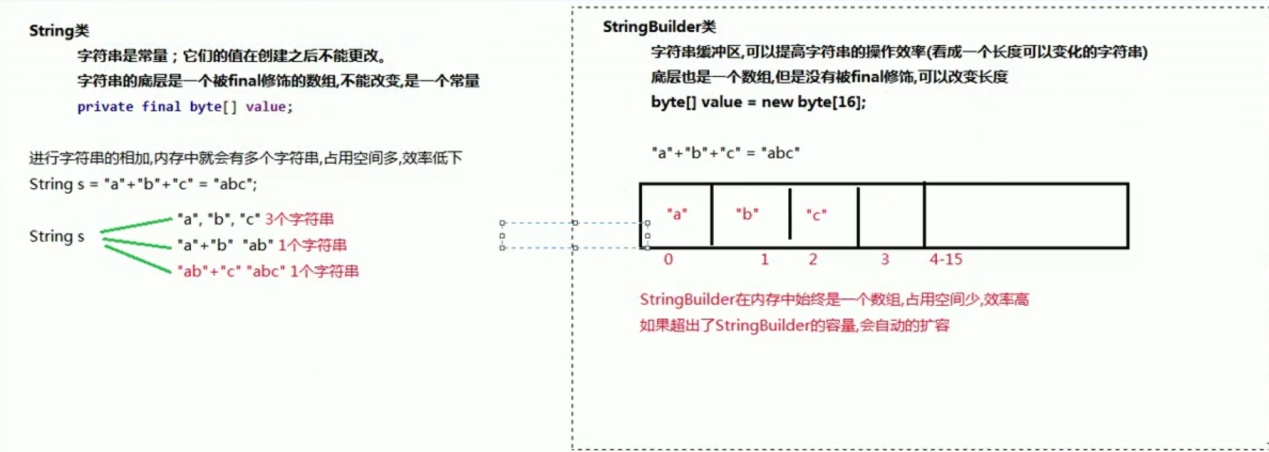
4.StringBuilder类 (java.lang.StringBuilder)

字符串缓冲区，可以提高字符串的操作效率（看成一个长度可变的字符串）

底层也是一个byte[]数组，但是没有被final修饰，可以改变长度（初始大小16）

（看到缓冲区三个字就知道可以提高效率）

如果超出了StringBuilder的容量，会自动扩容。



2、StringBuilder的构造方法和API

1、空参构造：StringBuilder():构造一个不带任何字符的字符串生成器，初始容量16字符。

2、有参构造：StringBuilder(String str):构造一个字符串生成器，并初始化为指定的字符串内容。

3、API:

成员方法1：

public StringBuilder append(…):添加任意类型数据的字符串形式，并**返回当前对象自身**。



成员方法2：

public String toString（）：将当前StringBuilder对象转换成为String对象。

StringBuilder和String可以相互转换：

String—》StringBuilder：可以使用StringBuilder的构造方法（带参构造，参数String类型）。

StringBuilder—》String：可以使用StringBuilder中的toString方法



5.包装类

1、基本数据类型，使用起来非常方便，但是没有对应的方法来操作这些基本类型的数据，可以使用一个类，把基本数据类型的数据装起来，在类中定义一些方法，这个类叫做包装类，我们可以使用类中的方法来操作这些基本类型的数据。

2、装箱与拆箱

①**装箱**：基本类型的数据，包装到包装类中（**基本类型的数据—>包装类**）

**四种做法：**

1.构造方法：

1、Integer(int value)构造一个新分配的Integer对象，它表示指定的int值。

2、Integer（String s）构造一个新分配的Integer对象，它表示String参数所指示的int值。 注意:传递的字符串，必须是基本类型的字符串，否则会抛出异常。【“100”正确，”a” 错误】

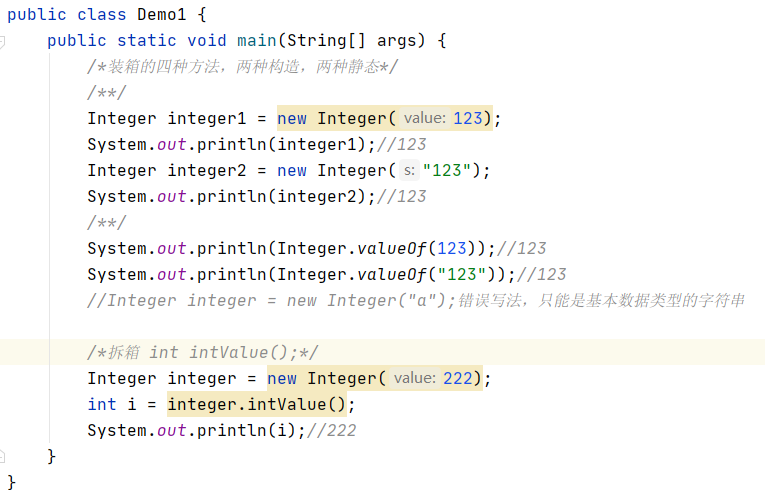
2.静态方法：

3、static Integer valueOf(int i):返回一个表示指定的int值的Integer实例。

4、static Integer valueOf(String s):返回保存指定的String的值的Integer对象。

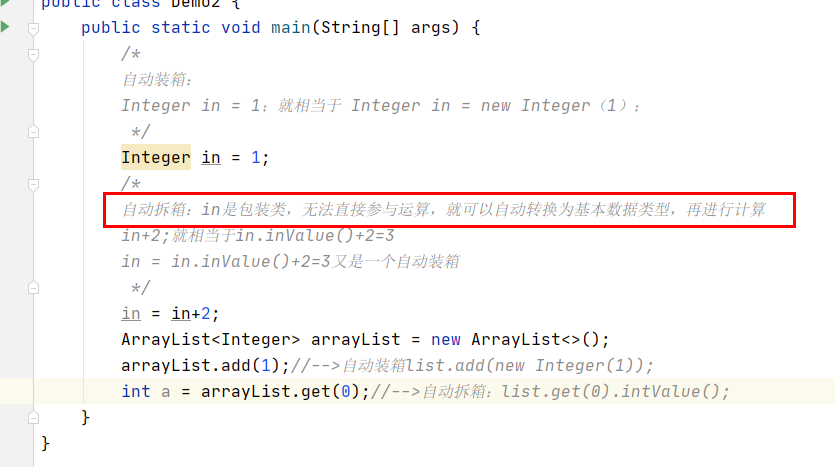
②**拆箱**：在包装类中取出基本类型的数据（**包装类—>基本类型的数据**）

成员方法： int intValue():以int类型返回该Integer的值。



3、自动装箱与自动拆箱

基本类型的数据和包装类之间可以自动的相互转换。JDK1.5之后才有。



4、基本类型与字符串类型间的相互转换

①基本类型—》字符串（String）

1、基本类型的值+“”；最简单的方法（工作中常用）

2、包装类的静态方法toString()，不是Object类的toString(),重载了。

3、String类的静态方法valueOf(参数)

static String valueOf(int i):返回int参数的字符串表示形式。

②字符串（String）--》基本类型

使用包装类的静态方法parsexxx(“数值类型的字符串”);

Integer:static int parseInt(String s);

Double :static double parseDouble（String s）;

