常用类

java.util.Date

概述

> java中的日期类，表示特定的瞬间，精确到毫秒

构造方法

1. `public Date()` :创建保存当前时间的一个Date对象
2. `public Date(long date)` :创建指定时间的Date对象
    - `date` :距离1970年1月1日凌晨的毫秒数

常用方法

1. `public long getTime()` :返回自1970年1月1日00:00:00 此Date对象表示的毫秒数
2. `public boolean before(Date when)` :此日期是否在指定日期之前
3. `public boolean after(Date when)` :此日期是否在指定日期之后
4. 还有一些过时方法不再推荐使用，详见**API**文档

例子

示例代码

```
Date date1 = new Date();
Date date2 = new Date(1470297241281L);
System.out.println(date1);
System.out.println(date2);
System.out.println(date1.before(date2));//false
System.out.println(date1.after(date2));//true
/********以下方法已过时********/
System.out.println(date1.getYear());//年:116(2016-1900)
System.out.println(date1.getMonth());//月:7(0-11)
System.out.println(date1.getDate());//4号
System.out.println(date1.getDay());//周四
```

执行结果

```
Thu Aug 04 21:07:31 CST 2016
Thu Aug 04 15:54:01 CST 2016
false
true
116
7
4
4
```

java.text.SimpleDateFormat

概述

> 是一个以与语言环境有关的方式来格式化和解析日期的具体类。它允许进行格式化（日期 -> 文本）、解析（文本 -> 日期）和规范化

构造方法

1. `SimpleDateFormat()` :用默认的模式和默认语言环境的日期格式符号构造 SimpleDateFormat
2. `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

日期和时间模式

年	月	日	时(12小时制)	时(24小时制)	分	秒	上午/下午
yyyy	MM	dd	hh	HH	mm	ss	a

常用方法

1. `public final String format(Date date)` :将一个 Date 格式化为日期/时间字符串
2. `public Date parse(String source)` :从给定字符串的开始解析文本，以生成一个日期

例子

示例代码

```
Date date = new Date();
SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss a");
//将Date根据指定格式转换为字符串
System.out.println(format.format(date));
String str = "2016-08-01 14:00:00 下午";
//将字符串按照指定格式转换为Date
Date date1 = format.parse(str);
System.out.println(date1);
```

执行结果

```
Thu Aug 04 21:26:51 CST 2016
2016-08-04 21:26:51 下午
Mon Aug 01 14:00:00 CST 2016
```

java.util.Calendar

概述

> Calendar类是一个抽象类，为特定瞬间与一组诸如YEAR、MONTH、DAY_OF_MONTH、HOUR等日历字段之间的转换提供了一些方法，并为操作日历字段（例如获得下星期的日期）提供了一些方法。瞬间可用毫秒值来表示，它是距历元（即格林威治标准时间1970年1月1日的00:00:00.000）的偏移量

获取对象

`getInstance()` :获取表示当前时间的Calendar对象

常用字段

1. `YEAR` :年
2. `MONTH` :月（加一）
3. `DAY_OF_MONTH` :日
4. `DAY_OF_WEEK` :周（减一）
5. `HOUR` :小时（12）

6. `HOUR_OF_DAY`:小时（24）
7. `MINUTE`:分钟
8. `SECOND`:秒

常用方法

1. `public int get(int field)`:根据`field`获取指定的日期字段
2. `public void set(int field, int value)`:为指定`field`赋值
3. `public void set(int year, int month, int date, int hourOfDay, int minute, int second)`:设置字段`YEAR`、`MONTH`、`DAY_OF_MONTH`、`HOUR`、`MINUTE`和`SECOND`的值。

例子

示例代码

```
Calendar c = Calendar.getInstance();
System.out.println(c);
c.set(2015, 6, 1, 14, 0, 0);
//年
//c.set(Calendar.YEAR, 2015);
System.out.println(c.get(Calendar.YEAR));
//月
//c.set(Calendar.MONTH, 11);
System.out.println(c.get(Calendar.MONTH));//(0-11)
//日
System.out.println(c.get(Calendar.DAY_OF_MONTH));
//星期（从周日到周六：1-7，需要减一）
System.out.println(c.get(Calendar.DAY_OF_WEEK));
//小时
System.out.println(c.get(Calendar.HOUR));//12小时制
System.out.println(c.get(Calendar.HOUR_OF_DAY));//24小时制
//分钟
System.out.println(c.get(Calendar.MINUTE));
//秒
System.out.println(c.get(Calendar.SECOND));
```

执行结果

```
java.util.GregorianCalendar[time=1470361774768,areFieldsSet=true,areAllFieldsSet=true,lenient=true
,zone=sun.util.calendar.ZoneInfo[id="Asia/Shanghai",offset=28800000,dstSavings=0,useDaylight=false
,transitions=19,lastRule=null],firstDayOfWeek=1,minimalDaysInFirstWeek=1,ERA=1,YEAR=2016,MONTH=7,W
EEK_OF_YEAR=32,WEEK_OF_MONTH=1,DAY_OF_MONTH=5,DAY_OF_YEAR=218,DAY_OF_WEEK=6,DAY_OF_WEEK_IN_MONTH=1
,AM_PM=0,HOUR=9,HOUR_OF_DAY=9,MINUTE=49,SECOND=34,MILLISECOND=768,ZONE_OFFSET=28800000,DST_OFFSET=
0]
2015
6
1
4
2
14
0
0
```

java.lang.Math

概述

> 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。它是一个final类，其中定义的都是一些常量和静态方法。

常用方法

1. `static double sqrt(double a)` :返回a的平方根
2. `static double ceil(double a)` :向上取整
3. `static double floor(double a)` :向下取整
4. `static double pow(double a,double b)` :返回a的b次方
5. `static long round(double a)` :对a四舍五入
6. `static int round(float a)` :对a四舍五入
7. `static double random()` :返回大于等于0小于1的double类型随机数

例子

示例代码

```
System.out.println(Math.sqrt(4));//2
System.out.println(Math.ceil(4.1));//5
System.out.println(Math.floor(4.9));//4
System.out.println(Math.pow(3, 2));//8
System.out.println(Math.round(3.94));//4
System.out.println(Math.round(3.14f));//3
System.out.println(Math.random());
```

执行结果

```
2.0
5.0
4.0
9.0
4
3
0.5149851216956421
```

java.util.Random

概述

> 专业的随机数工具类，功能强大

构造方法

1. `public Random()` :创建一个随机数对象
2. `public Random(long seed)` :根据指定的种子数创建随机数对象，相同种子数的随机数对象，相同次数生成的随机数相同

常用方法

1. `public void nextBytes(byte[] bytes)` :随机生成`bytes.length`个`byte`类型的随机数，放到`byte`数组中
2. `public int nextInt()` :随机生成一个int类型随机数
3. `public int nextInt(int n)` :随机生成一个大于等于0小于n的int类型随机数
4. `public boolean nextBoolean()` :随机生成一个布尔类型随机数
5. `public float nextFloat()` :随机生成一个`float`类型随机数
6. `public double nextDouble()` :随机生成一个double类型随机数

例子

示例代码

```
Random random = new Random();
byte[] bytes = new byte[5];
System.out.println(Arrays.toString(bytes));
random.nextBytes(bytes);
System.out.println(Arrays.toString(bytes));
System.out.println(random.nextInt(3));
System.out.println(random.nextInt());
System.out.println(random.nextBoolean());
System.out.println(random.nextDouble());
System.out.println(random.nextFloat());
System.out.println(random.nextLong());
```

执行结果

```
[0, 0, 0, 0, 0]
[-108, 3, 74, 88, 74]
1
-353298479
false
0.02247339367434331
0.60528654
57285040312506508
```