# 尚马教育 JAVA 基础课程

# 常用工具类

文档编号：A13

创建日期： 2017-04-12

最后修改日期：2019-09-23

版 本 号：V3.0

电子版文件名：尚马教育-第一阶段-13.常用工具类专题课程.docx

**文档修改记录：**

|  |  |  |  |
| --- | --- | --- | --- |
| 更新日期 | 更新作者 | 更新说明 | 版本号 |
| 2017-07-30 | 张元林 | 初始版本 | V1.0 |
| 2018-08-01 | 王绍成 | Java基础版本更新 | V2.0 |
| 2019-08-09 | 徐丽莎 | Java基础版本更新 | V3.0 |

**主讲人：**

**徐丽莎**

目录

[尚马教育 JAVA 基础课程 1](#_Toc8080)

[常用工具类 1](#_Toc427)

[1. 日期类 2](#_Toc32747)

[1.1. Date 3](#_Toc9703)

[1.1.1. 常用构造 3](#_Toc26218)

[1.1.2. 常用方法 3](#_Toc3947)

[1.1.3. 案例 3](#_Toc3837)

[1.2. Calendar 4](#_Toc12789)

[1.2.1. 常用方法 4](#_Toc28592)

[1.2.2. 案例 5](#_Toc24456)

[1.3. Instant 5](#_Toc32277)

[1.3.1. 常用方法 6](#_Toc23707)

[1.3.2. 案例 6](#_Toc17714)

[1.4. LocalDate 7](#_Toc9954)

[1.4.1. 常用方法 7](#_Toc30630)

[1.4.2. 案例 7](#_Toc2897)

[1.5. LocalDateTime 8](#_Toc18460)

[1.5.1. 案例 8](#_Toc19784)

[1.6. LocalTime 8](#_Toc25166)

[1.6.1. 案例 8](#_Toc28680)

[2. 格式化类 9](#_Toc28964)

[2.1. DateFormat 9](#_Toc31968)

[2.1.1. Date转换成String 9](#_Toc17070)

[2.1.2. String转换成Date 9](#_Toc13235)

[2.2. DateTimeFormatter 9](#_Toc24840)

[2.3. NumberFormat 11](#_Toc16585)

[2.3.1. 案例 11](#_Toc28536)

[3. 随机数类 12](#_Toc10758)

[3.1. Random/ThreadLocalRandom 12](#_Toc21816)

[3.1.1. 常用方法 12](#_Toc30462)

[3.1.2. 案例 12](#_Toc19401)

[4. 编码解码类 13](#_Toc29310)

[4.1. Base64 13](#_Toc21711)

[4.2. MessageDigest 14](#_Toc6526)

[5. 作业 14](#_Toc31674)

## 日期类

* java.util包是Java内置的一个工具包，其中包含一系列常用的工具类；
* 如处理日期和日历的类以及一些集合类；
* java.util包不会默认导入，如果要使用到该包中的类，则必须在程序的开始部分显式地声明如下语句：import java.util.\*;

### Date

* Date类对象用来表示时间和日期；该类提供一系列操纵日期和时间各组成部分的方法
* Date类最多的用途是获取系统当前的日期和时间。

#### 常用构造

* Date类的构造方法有6种重载方式，很多已经过时，以下是比较常用的2种。

|  |
| --- |
|  |

#### 常用方法

|  |
| --- |
|  |

#### 案例

|  |
| --- |
| **public** **class** DateDemo {  **public** **static** **void** main(String[] args) {  Date date = **new** Date(); // 获得当前的系统日期和时间  System.***out***.println("今天的日期是：" + date);  **long** time = date.getTime(); // 获得毫秒数  System.***out***.println("自1970年1月1日起以毫秒为单位的时间(GMT):" + time);  // 截取字符串中表示时间的部分  String strDate = date.toString();  String strTime = strDate.substring(11, (strDate.length() - 4));  System.***out***.println(strTime);  strTime = strTime.substring(0, 8);  System.***out***.println(strTime);  }} |

### Calendar

* Calendar类也是用来操作日期和时间的类，但它可以以整数形式检索类似于年、月、日之类的信息；
* Calendar类是抽象类，无法实例化，要得到该类对象只能通过调用getInstance方法来获得；
* Calendar对象提供为特定语言或日历样式实现日期格式化所需的所有时间字段
* Jdk1.8之前推荐使用Calendar替换Date。

#### 常用方法

|  |
| --- |
|  |

#### 案例

|  |
| --- |
| **public** **class** CalendarDemo {  **public** **static** **void** main(String[] args) {  // 创建包含有当前系统时间的Calendar对象  Calendar cal = Calendar.getInstance();  // 打印Calendar对象的各个组成部分的值  System.***out***.print("当前系统时间：");  System.***out***.print(cal.get(Calendar.YEAR) + "年");  System.***out***.print((cal.get(Calendar.MONTH) + 1) + "月");  System.***out***.print(cal.get(Calendar.DATE) + "日 ");  System.***out***.print(cal.get(Calendar.HOUR) + ":");  System.***out***.print(cal.get(Calendar.MINUTE) + ":");  System.***out***.println(cal.get(Calendar.SECOND));  // 将当前时间添加30分钟，然后显示日期和时间  cal.add(Calendar.MINUTE, 30);  Date date = cal.getTime();  System.***out***.println("将当前时间添加30分钟后的时间：" + date);  }} |

### Instant

* JDK8的应用，推荐使用 Instant 代替 Date。
* Instant 时间线上的瞬间点，采用UTC的时间刻度。此类是不可变的且是线程安全的。在java.time.\*包下 1.8 版本才有。

#### 常用方法

|  |
| --- |
|  |

#### 案例

|  |
| --- |
| **public** **class** InstantDemo {  **private** **static** **void** test1() {  // Instant instant =  // Instant.now(Clock.offset(Clock.systemUTC(),Duration.ofHours(8)));  Instant instant = Instant.now();  // System.out.println(instant);  // 2019-12-12T03:03:22.278Z 8 UTC  // +8H  instant = instant.plusSeconds(8 \* 60 \* 60);// 加8个小时 转换成秒数  System.***out***.println(instant);  // instant = instant.plus(Duration.ofHours(8));  // instant = instant.plus(8, ChronoUnit.HOURS);  // int year = instant.get(ChronoField.MILLI\_OF\_SECOND);  // instant.getLong(field);  System.***out***.println(Instant.now());  **long** duration = instant.until(Instant.now(), ChronoUnit.HOURS);// 获得2个时间对象的间隔 转换成特定的时间单位  System.***out***.println(duration);}} |

### LocalDate

* LocalDate 获得当前系统的时间 2007-12-03 线程安全且不可变。

#### 常用方法

|  |
| --- |
|  |

#### 案例

|  |
| --- |
| **public** **class** LocalDateDemo {  **private** **static** **void** test1() {  LocalDate localDate = LocalDate.now();  System.***out***.println(localDate);// 2019-03-14  System.***out***.println(localDate.getYear());  System.***out***.println(localDate.getDayOfWeek());// THURSDAY  localDate = localDate.minusMonths(1);  System.***out***.println(localDate);  LocalDate date1 = LocalDate.of(2018, 3, 20);  System.***out***.println(date1);  }  } |

### LocalDateTime

* LocalDateTime 获得当前系统的时间 2007-12-03T10:15:30 线程安全且不可变

#### 案例

|  |
| --- |
| **public** **class** LocalDateTimeDemo {  **private** **static** **void** testLocalDateTime() {  LocalDateTime localDateTime = LocalDateTime.now();  System.***out***.println(localDateTime);//2019-12-12T14:54:00.375  localDateTime = localDateTime.minusMonths(10);  System.***out***.println(localDateTime);  System.***out***.println(**new** Date());// 2019年12月12日 12:00:00 星期一 am  }  } |

### LocalTime

* LocalTime 获得当前系统的时间 10:15:30 线程安全且不可变

#### 案例

|  |
| --- |
| **public** **class** LocalTimeDemo {  **private** **static** **void** testLocalTime() {  LocalTime localTime = LocalTime.now();  System.***out***.println(localTime);// 14:48:12.525  LocalTime localTime2 = LocalTime.of(23, 59, 59);  System.***out***.println(localTime2);  System.***out***.println(localTime.getHour());  System.***out***.println(localTime.getMinute());  System.***out***.println(localTime.plusHours(2));  System.***out***.println(localTime.minusMinutes(50));  }  } |

## 格式化类

* 当数据格式并不满足需求的时候，我们可以对指定的不同类型的数据进行格式化操作，转换成特定的格式的数据。

### DateFormat

* 在显示日期的时候，可以使用很多格式，这时我们就可以借助java.text包中的DateFormat类及其子类进行调整
* DateFormat属于抽象类，我们使用子类SimpleDateFormat ，只针对于Date类型数据进行格式化操作。

#### Date转换成String

|  |
| --- |
|  |

#### String转换成Date

|  |
| --- |
|  |

### DateTimeFormatter

* 如果是jdk1.8相关的几个日期类做转换的话，就必须使用DateTimeFormatter

|  |
| --- |
| **private** **static** **void** testInatsnt() {  // 将Instant 转换成string DateTimeFormatter  DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss")  .withZone(ZoneId.systemDefault());  String str = formatter.format(instant1);  // String 转换成 Instant  Instant parse = Instant.parse("2019-03-14T06:33:13.034Z");// 默认是这种格式的时间 否则无法转换  } |
| **private** **static** **void** testLocalDate() {  // LocalDate 与 String相互转换  // LocalDate.parse(text, formatter)  LocalDate date1 = LocalDate.of(2018, 3, 20);  DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy年MM月dd");  String str = formatter.format(date1);  System.***out***.println(str);// 2018年03月20  date1 = LocalDate.parse(str, formatter);// 2007-12-03  System.***out***.println(date1);  } |
| **private** **static** **void** testLocalDate() {  LocalTime localTime = LocalTime.now(ZoneId.systemDefault());  DateTimeFormatter formatter = DateTimeFormatter.ofPattern("HH时mm分ss");  String str = formatter.format(localTime);  System.***out***.println(str);  localTime = LocalTime.parse(str,formatter);  System.***out***.println(localTime);  } |

### NumberFormat

* 要格式化数字，可以使用NumberFormat类来实现
* NumberFormat也是抽象类，必须使用子类DecimalFormat
* //#.## 保留两位小数
* //#.##% 自动百分比计数并保留两位小数
* //,### 每三位以逗号分隔

#### 案例

|  |
| --- |
| **public** **static** **void** main(String[] args) {  // #.## 保留两位小数  // #.##% 自动百分比计数并保留两位小数  // ,### 每三位以逗号分隔  // 创建格式化数字对象  // NumberFormat numberFormat = new DecimalFormat("##.###");  // String result = numberFormat.format(100.4567);  // System.out.println(result);  // System.out.println(Math.round(1.5567));  // NumberFormat numberFormat = new DecimalFormat("#.##%");  // String result = numberFormat.format(0.456777);  // System.out.println(result);  NumberFormat numberFormat = **new** DecimalFormat("###,###.###");  String result = numberFormat.format(4567778827.626548);  System.***out***.println(result);  } |

## 随机数类

### Random/ThreadLocalRandom

* Random类专门用来生成随机数；
* ThreadLocalRandom是Random的子类。
* 在并发程序中使用ThreadLocalRandom而不是共享的Random对象通常会遇到更少的开销和争用。 当多个任务在线程池中并行使用随机数时，使用ThreadLocalRandom是特别合适的。

#### 常用方法

|  |
| --- |
|  |
|  |

#### 案例

|  |
| --- |
| **public** **class** RandomDemo {  **public** **static** **void** main(String[] args) {  // 创建一个Random类对象  Random rand = **new** Random();  // 随机生成20个随机整数，并将其显示出来  **for** (**int** i = 0; i < 20; i++) {  **int** num = rand.nextInt();  System.***out***.println("第" + (i + 1) + "个随机数是：" + num);  }  }  } |

## 编码解码类

### Base64

* Base64 加解密。jdk1.8可正常使用。
* Base64.Encoder Base64.Decoder
* 该类使用RFC 4648和RFC 2045中规定的Base64编码方案来实现用于解码字节数据的解(编)码器。

|  |
| --- |
| **public** **static** **void** main(String[] args) {  // Base64  // 注册 对密码进行加密  String pass = "1234";  Encoder encoder = Base64.getEncoder();// 获得编码器  String encodePass = encoder.encodeToString(pass.getBytes());  System.***out***.println(encodePass);// MTIzNA==  // 登陆  Decoder decoder = Base64.getDecoder();// 获得解码器  **byte**[] by = decoder.decode(encodePass);  String str = **new** String(by);  System.***out***.println(str);  **if** (pass.equals(str)) {  System.***out***.println("用户登陆成功");  }  } |

### MessageDigest

* MessageDigest类为应用程序提供消息摘要算法的功能，如SHA-1或SHA-256或MD5。 消息摘要是采用任意大小的数据并输出固定长度散列值的安全单向散列函数。

|  |
| --- |
| **public** **static** **void** main(String[] args) {  // MD5加密 信息摘要算法的一种  **try** {  // 1.获得messageDigest对象  MessageDigest messageDigest = MessageDigest.getInstance("MD5");  // 2.传输需要加密数据  messageDigest.update(pass.getBytes());  // 3.加密  **byte**[] encodeBy = messageDigest.digest();  System.***out***.println(Arrays.toString(encodeBy));  // 将字节数组转换成BigInteger实例  BigInteger bigInteger = **new** BigInteger(1, encodeBy);  String pass1 = bigInteger.toString(16);  System.***out***.println(pass1);  } **catch** (NoSuchAlgorithmException e) {  e.printStackTrace();} |

## 作业

1. 将长时间格式时间转换为字符串 yyyy-MM-dd HH:mm:ss
2. 使用Calendar类的相关属性以及方法！打印出某年某个月的日历信息！（要求年月日由命令行输入）
3. 从命令行输入一个字符串！要求从中随机选择6个字符组成验证码！
4. 获取一个月中的最后一天
5. 计算两个时间相差几个小时
6. 一个方法，要求传入时间和间隔天数，返回新的日期
7. 要求写一个工具类，可以自定义获取N个随机数字或字母（0~9,a~z,A~Z混合一起）