第十三章 可变字符串类和日期相关类

13.1 可变字符串类（重点）

13.1.1 基本概念

- 由于String类描述的字符串内容是个常量不可改变，当需要在Java代码中描述大量类似的字符串时，只能单独申请和存储，此时会造成内存空间的浪费。
- 为了解决上述问题，可以使用java.lang.StringBuilder类和java.lang.StringBuffer类来描述字符序列可以改变的字符串，如："ab"。
- StringBuffer类是从jdk1.0开始存在，属于线程安全的类，因此效率比较低。
- StringBuilder类是从jdk1.5开始存在，属于非线程安全的类，效率比较高。

13.1.2 StringBuilder类常用的构造方法

方法声明	功能介绍
StringBuilder()	使用无参方式构造对象，容量为16
StringBuilder(int capacity)	根据参数指定的容量来构造对象，容量为参数指定大小
StringBuilder(String str)	根据参数指定的字符串来构造对象，容量为：16+字符串长度

13.1.3 StringBuilder类常用的成员方法

方法声明	功能介绍
int capacity()	用于返回调用对象的容量
int length()	用于返回字符串的长度，也就是字符的个数
StringBuilder insert(int offset, String str)	插入字符串并返回调用对象的引用，就是自己。
StringBuilder append(String str)	追加字符串
StringBuilder deleteCharAt(int index)	将当前字符串中下标为index位置的单个字符删除
StringBuilder delete(int start，int end)	删除字符串
StringBuilder replace(int start，int end，String str)	替换字符串
StringBuilder reverse()	字符串反转

- 注意

  作为参数传递的话，方法内部String不会改变其值，StringBuffer和StringBuilder会改变其值。

13.1.4 返回值的设计

- StringBuilder的很多方法的返回值均为StringBuilder类型。这些方法的返回语句均为：return this。
- 由此可见，这些方法在对StringBuilder所封装的字符序列进行改变后又返回了该对象的引用。基于这样设计的目的在于可以连续调用。

13.2 Java8之前的日期相关类（熟悉）

13.2.1 System类的概述

（1）基本概念

- Java.lang.System类中提供了一些有用的类字段和方法。

（2）常用的方法

方法声明	功能介绍
static long currentTimeMillis()	返回当前时间与1970年1月1日0时0分0秒之间以毫秒为单位的时间差

13.2.2 Date类的概述

（1）基本概念

- java.util.Date类主要用于描述特定的瞬间，也就是年月日时分秒，可以精确到毫秒。

（2）常用的方法

方法声明	功能介绍
Date()	使用无参的方式构造对象，也就是当前系统时间
Date(long date)	根据参数指定毫秒数构造对象， 参数为距离1970年1月1日0时0分0秒的毫秒数
long getTime()	获取调用对象距离1970年1月1日0时0分0秒的毫秒数
void setTime(long time)	设置调用对象为距离基准时间time毫秒的时间点

13.2.3 SimpleDateFormat类的概述

（1）基本概念

- java.text.SimpleDateFormat类主要用于实现日期和文本之间的转换。

（2）常用的方法

方法声明	功能介绍
SimpleDateFormat()	使用无参方式构造对象
SimpleDateFormat(String pattern)	根据参数指定的模式来构造对象，模式主要有: y-年 M-月 d-日 H-时 m-分 s-秒
final String format(Date date)	用于将日期类型转换为文本类型
Date parse(String source)	用于将文本类型转换为日期类型

13.2.4 Calendar类的概述

（1）基本概念

- java.util.Calender类主要用于描述特定的瞬间，取代Date类中的过时方法实现全球化。
- 该类是个抽象类，因此不能实例化对象，其具体子类针对不同国家的日历系统，其中应用最广泛的是GregorianCalendar（格里高利历），对应世界上绝大多数国家/地区使用的标准日历系统。

（2）常用的方法

方法声明	功能介绍
static Calendar getInstance()	用于获取Calendar类型的引用
void set(int year, int month, int date, int hourOfDay, int minute, int second)	用于设置年月日时分秒信息
Date getTime()	用于将Calendar类型转换为Date类型
void set(int field, int value)	设置指定字段的数值
void add(int field, int amount)	向指定字段增加数值

（3）多态的使用场合

- 通过方法的参数传递形成多态；
  ```
  public static void draw(Shape s){
      s.show();
  }
  draw(new Rect(1, 2, 3, 4));
  ```
- 在方法体中直接使用多态的语法格式
  ```
  Account acc = new FixedAccount();
  ```
- 通过方法的返回值类型形成多态
  ```
  Calender getInstance(){
     return new GregorianCalendar(zone, aLocale);
   }
  ```

13.3 Java8中的日期相关类（熟悉）

13.3.1 Java8日期类的由来

JDK 1.0中包含了一个java.util.Date类，但是它的大多数方法已经在JDK 1.1引入Calendar类之后被弃用
了。而Calendar并不比Date好多少。它们面临的问题是：

- Date类中的年份是从1900开始的，而月份都从0开始。
- 格式化只对Date类有用，对Calendar类则不能使用。
- 非线程安全等。

13.3.2 Java8日期类的概述

- Java 8通过发布新的Date-Time API来进一步加强对 日期与时间的处理。
- java.time包：该包日期/时间API的基础包。
- java.time.chrono包：该包提供对不同日历系统的访问。
- java.time.format包：该包能够格式化和解析日期时间对象。
- java.time.temporal包：该包包含底层框架和扩展特性。
- java.time.zone包：该包支持不同时区以及相关规则的类。

13.3.3 LocalDate类的概述

（1）基本概念

- java.time.LocalDate类主要用于描述年-月-日格式的日期信息，该类不表示时间和时区信息。

（2）常用的方法

方法声明	功能介绍
static LocalDate now()	在默认时区中从系统时钟获取当前日期

13.3.4 LocalTime类的概述

（1）基本概念

- java.time.LocalTime 类主要用于描述时间信息，可以描述时分秒以及纳秒。

（2）常用的方法

方法声明	功能介绍
static LocalTime now()	从默认时区的系统时间中获取当前时间
static LocalTime now(ZoneId zone)	获取指定时区的当前时间

13.3.5 LocalDateTime类的概述

（1）基本概念

- java.time.LocalDateTime类主要用于描述ISO-8601日历系统中没有时区的日期时间，如2007-12-03T10:15:30。

（2）常用的方法

方法声明	功能介绍
static LocalDateTime now()	从默认时区的系统时间中获取当前日期时间
static LocalDateTime of(int year, int month, int dayOfMonth, int hour, int minute, int second)	根据参数指定的年月日时分秒信息来设置日期时间
int getYear()	获取年份字段的数值
int getMonthValue()	获取1到12之间的月份字段
int getDayOfMonth()	获取日期字段
int getHour()	获取小时数
int getMinute()	获取分钟数
int getSecond()	获取秒数
LocalDateTime withYear(int year)	设置为参数指定的年
LocalDateTime withMonth(int month)	设置为参数指定的月
LocalDateTime withDayOfMonth(int dayOfMonth)	设置为参数指定的日
LocalDateTime withHour(int hour)	设置为参数指定的时
LocalDateTime withMinute(int minute)	设置为参数指定的分
LocalDateTime withSecond(int second)	设置为参数指定的秒
LocalDateTime plusYears(long years)	加上参数指定的年
LocalDateTime plusMonths(long months)	加上参数指定的月
LocalDateTime plusDays(long days)	加上参数指定的日
LocalDateTime plusHours(long hours)	加上参数指定的时
LocalDateTime plusMinutes(long minutes)	加上参数指定的分
LocalDateTime plusSeconds(long seconds)	加上参数指定的秒
LocalDateTime minusYears(long years)	减去参数指定的年
LocalDateTime minusMonths(long months)	减去参数指定的月
LocalDateTime minusDays(long days)	减去参数指定的日
LocalDateTime minusHours(long hours)	减去参数指定的时
LocalDateTime minusMinutes(long minutes)	减去参数指定的分
LocalDateTime minusSeconds(long seconds)	减去参数指定的秒

13.3.6 Instant类的概述

（1）基本概念

- java.time.Instant类主要用于描述瞬间的时间点信息。

（2）常用的方法

方法声明	功能介绍
static Instant now()	从系统时钟上获取当前时间
OffsetDateTime atOffset(ZoneOffset offset)	将此瞬间与偏移量组合以创建偏移日期时间
static Instant ofEpochMilli(long epochMilli)	根据参数指定的毫秒数来构造对象，参数为距离1970年1月1日0时0分0秒的毫秒数
long toEpochMilli()	获取距离1970年1月1日0时0分0秒的毫秒数

13.3.7 DateTimeFormatter类的概述

（1）基本概念

- java.time.format.DateTimeFormatter类主要用于格式化和解析日期。

（2）常用的方法

方法声明	功能介绍
static DateTimeFormatter ofPattern(String pattern)	根据参数指定的模式来获取对象
String format(TemporalAccessor temporal)	将参数指定日期时间转换为字符串
TemporalAccessor parse(CharSequence text)	将参数指定字符串转换为日期时间