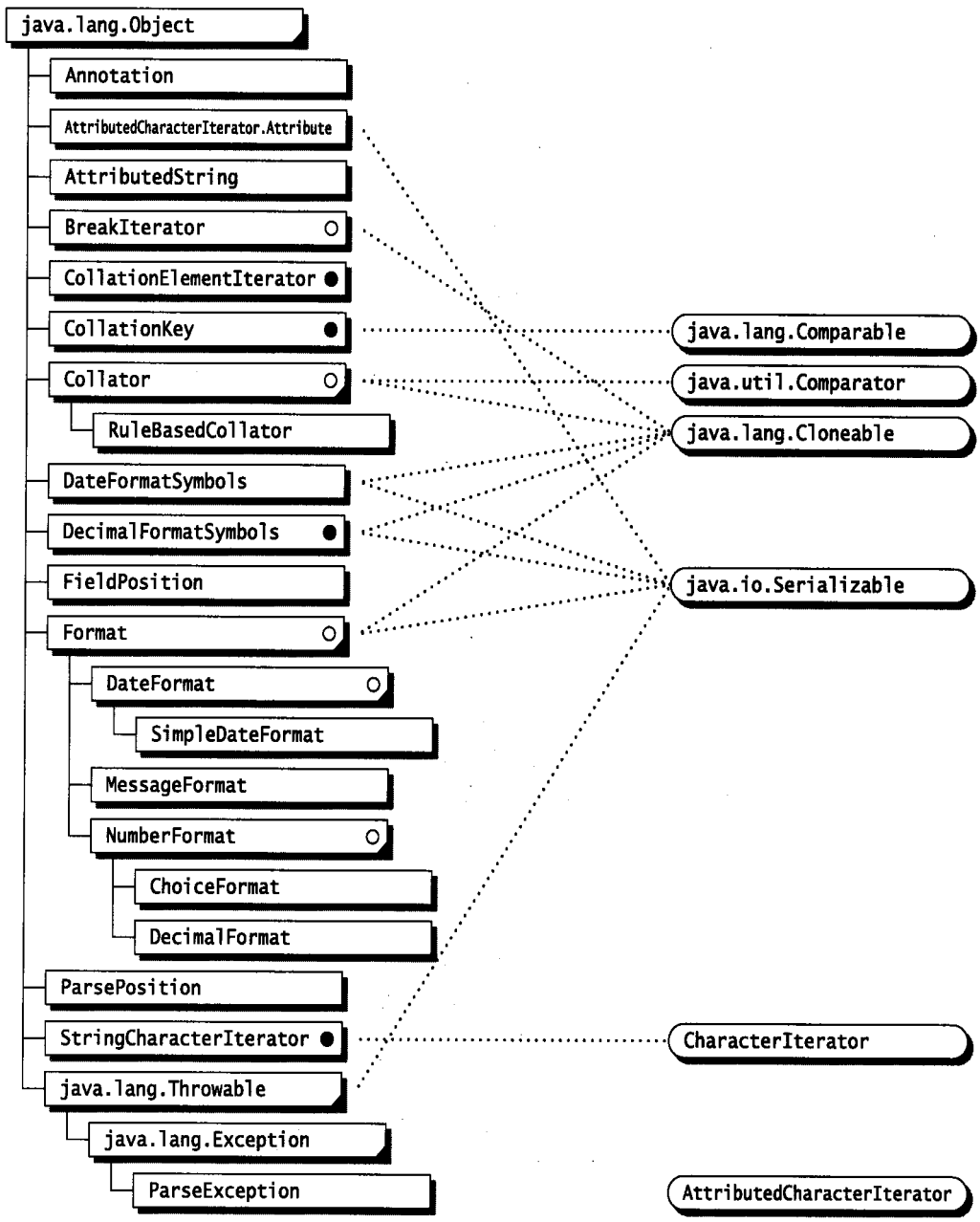
java.text



描述

java.text包包括以一种独立于自然语言的方式处理文本、日期、数字和消息的类和接口。这意味着主要应用程序或applet可以一种独立于语言的方式编写，而且，可以请求分离的、动态链接的本地化资源。这样就增加了灵活性，随时可以加入本地化的新语言。

这些类可以格式化日期、数字和消息，进行语义分析，对字符串进行查找和排序，及对字符、词、句子和断行进行连接。该包主要包括了四个类和接口组：

- 格式化和语义分析类
- 字符串整理类
- 迭代字符串的类
- 与文本属性相关的类

格式化和语义分析类

如图6所示，格式化是把数字和日期对象转变成字符串，而语义分析是格式化的逆操作，把字符串转化为数字和日期对象。

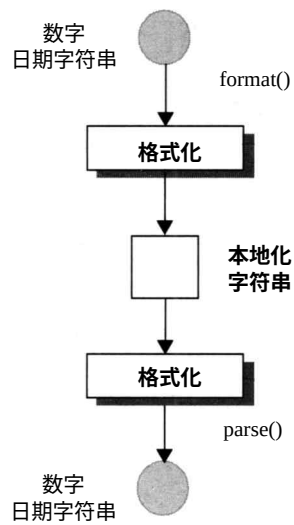


图6 对象格式化和字串语义分析

日期和数字在内部以独立于地区的方式表示。例如，日期以毫秒保存(从1970年1月1日，00：00：00 GMT算起)。当这些对象被打印或显示时，它们必须被转化成本地化的字符串。日期字符串中与地区相关的部分，如当地时区的字符串，单独从与当地相关的资源包中导入。

format()方法把Date对象从-604 656 780 000毫秒转化成如格式“Tuesday，November 3，1997 9：47am CST”这样的美式英语写法。图6显示了Format的子类的format()方法是如何将Number、Date、String和其他对象的实例格式化为与地区相关的字符串的。

相反，parseObject()方法(以及在子类中的parse()方法)执行相反的操作，它把当地化的字符串经过语义分析转化成Number、Date、String对象。图6显示了parse()方法是如何对format()方法加以补充的。任何一个被format()方法格式化的字符串可以由parseObject()方法进行解析。

Java.text包提供了六个Format的子类，用于格式化日期、数字和消息： DateFormat、SimpleDateFormat、NumberFormat、DecimalFormat、ChoiceFormat和MessageFormat。

串校对

术语校对的意思是判定两个或多个字符串的排列顺序。它也能说明两个字符串是否相等。Collator类和它的子类RuleBasedCollator用于对字符串进行地区敏感的比较。可以使用这些类对自然语言文本进行查找和按字母进行排序。它们可以区分基于基本字符的字符、重音标志和大小写属性。

Collator是一个抽象的基类。子类实现了具体的校对策略。现在提供的子类RuleBasedCollator对于大部分语言可用。其他子类用于处理更加专门的需求。CollationElementIterator提供一个迭代器用于把每一个与地区相关的字符串的字符根据一个具体的Collator对象中的规则进行迭代。CollationKey根据一个具体的Collator对象中的规则通过

声明一个字符串作为排序关键字，对字符串进行快速排序。

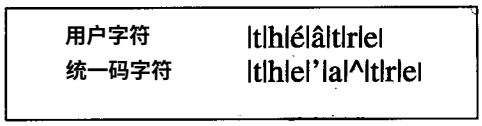


图7 用户字符串

字符串迭代

Java类char和Character代表统一码字符，有时把统一码字符组合在一起构造更复杂的字符，叫做用户字符，用户字符有它自己的语义。BreakIterator类使这些用户字符可能被迭代。一个断点迭代器可以找到一个字符、词、句子的边界，或者潜在的换行边界。这使得一个程序可以为不同的文本操作正确地选择字符，如使一个字符高亮显示，剪切掉一个字，移动到下一句，或者在行末进行换行。这些操作以对地区敏感的方式执行，这意味着它们必须注意在特定地区下的文本的边界。

图7显示了法语单词“ theatre ”是如何由九个统一码字符组成。用户字符可以是一笔或多笔。在拉丁语中(有字母A～Z)，一个字符是一个词的一部分，而在表意语言 (如中文、日文、朝鲜文)中，一个字符可以表达一个完整的意思或更大的词的一部分。例如，包括“ ä ”的用户字符和在一些中东和亚洲语言中的字符。

文本属性

文本可以有属性，例如语言、发音、字体、大小和字型。每一个属性有一个关键字和与此相关的值。在文本中的每一个字符可以有一个关键字 /值对集，其中没有两个关键字是相同的。关键字/值对可被加在一个或多个字符区域中。例如，一个有 LANGUAGE属性的字符串可以有JAPANESE地区值作为它的值。字符串的一部分可以有 READING属性，该属性定义如何发音 —— 它的值只是一个发音字符串。该字符串可以包括在一个注释中，以说明它不能和邻近属性合并。多属性和注释不同以被应用于可重叠的字符串子区域中。字符串属性类允许定义属性常量和注释值，允许存储属性化的字符串，允许重复字符和它们的关键字 /值对。

对地区敏感的类

java.text包中有许多类是地区敏感的，这就意味着你必须为每一个地区创建不同的实例。表1列出了地区敏感的和独立于地区的类。

表1 地区敏感的和独立于地区的类

地区敏感的类	独立于地区的类
NumberFormat	Format
DecimalFormat	ChoiceFormat
DecimalFormatSymbols	FieldPosition
MessageFormat	ParsePosition
DateFormat	ParseException
SimpleDateFormat	StringCharacterIterator
DateFormatSymbols	CharacterIterator
Collator	AttributedString
RuleBasedCollator	AttributedCharacterIterator
CollationElementIterator	AttributedCharacterIterator.Attribute
CollationKey	Annotation
BreakIterator	

版本1.2中所作的修改

格式类

SimpleDateFormat有了新的设置和获取方法，用于说明在格式化了的字符串中那里是两位数字年份的开始 。

DecimalFormat有四个新方法用于设置小数和整数的最大和最小位数。这样更多的统一码数(如上标)可以被当作十进制数进行语义分析。指数“ E ”已经被当作标号加进来了。而且，还介绍了一种新的分隔符 —— 货币十进制分隔符。当货币符号 (\u00a4)在符号集中，货币十进制分隔符被应用；否则，普通的十进制分隔符被应用。

DecimalFormatSymbols有了新的获取和设置通货符号和货币十进制分隔符的方法。

FieldPosition有了新的设置开始和结束索引的方法，当创建Format的子类时这些方法将被应用。

ParsePosition有了新的设置和获取错误索引的方法，它们用于精确定位语义错误。

整理类

CollationElementIterator有用于访问偏移量、设置文本、获取最大扩展，以及向后移动迭代器的新方法。

CollationKey已经被修改以实现Comparable接口。

Collator已经被修改以实现 Comparator接口。它不再实现 Serializable。在版本 1.1中，Collator的序列化没有被实现，并且总是抛出一个异常。

RuleBasedCollator中的getCollationElementIterator()有了新的重载形式，它以 CharacterIterator对象作为参数。

迭代器类

BreakIterator不再实现Serializable。在版本1.1中，BreakIterator的序列化没有被实现，并且总是抛出一个异常。现在加入了两个方便的方法： isBoundary()和preceding()。

在CharacterIterator和StringCharacterIterator中，current()、first()、last()、next()、previous()、setIndex()的语义已经修改，现在它们可以接受由 getEndIndex()返回的状态。

文本属性类

文本属性类： AttributedCharacterIterator、AttributedCharacterIterator.Attribute、AttributedString和Annotation已经被加到java.text包中。

类与接口概述

普通格式

Format类定义了用于把一个 Date或Number对象(或原始数字)格式化成一个本地化字符串，或把一个本地化字符串经语义分析转化成 Number、Date、String或者其他对象的操作。 format()方法接受一个 FieldPosition对象作为一个参数。 parse()方法的一些语法形式接受 ParsePosition对象作为参数。

Format	所有格式类的抽象超类。它提供了对数字、日期、字符串和其他对象进行格式化和语法分析的方法。

FieldPosition	一个包含域常量及数字与日期域的起止索引的具体类。
ParsePosition	一个包含字符串在进行语法分析时的当前位置的具体类。

数字格式化

下面的类定义了用于把一个 Number对象或原始数字格式化成一个本地化的字符串，或把一个本地化的字符串经语义分析转化成 Number对象的操作。 NumberFormat是一个抽象超类，DecimalFormat和ChoiceFormat是它的子类。 DecimalFormat使用DecimalFormatSymbols。

NumberFormat	一个抽象的超类，它提供了一些基本域以及用于把Number对象和数字格式化为本地字符串或者通过语义分析把本地化的字符串转换为Number对象的方法。
DecimalFormat	一个具体类，使用它可以把 Number对象和数字格式化为本地字符串或者通过语义分析把本地化的字符串转换为Number对象。
DecimalFormatSymbols	一个具体类，用于访问本地化的数字字符串，如组分割符、小数点和百分号。它在DecimalFormat类中使用。
ChoiceFormat	一个具体类，为了处理用户消息中的复数和名字序列使用此类把字符串映射到相应的数字区间上。

日期和时间格式化

下面的类定义了用于把一个Date对象格式化成一个本地化的字符串，或把一个本地化的字符串经语义分析转化成Date对象。 DateFormat是抽象超类， SimpleDateFormat是它的子类。

DateFormat	一个抽象超类，它提供了一些基本域以及用于把 Date对象格式化为本地字符串或者通过语义分析把日期或时间字符串转换为Date对象的方法。
SimpleDateFormat	一个具体类，使用它可以把 Date对象格式化为本地字符串或者通过语义分析把日期或时间字符串转换为 Date对象。
DateFormatSymbols	一个具体类，用于访问本地化的日期时间字符串，如月份的名称、一周中的星期几和时区。它在 DateFormat类中使用。

消息格式化

Message类用于产生于特定语言相关的用户消息，它是 Format的子类。

MessageFormat	一个产生包含数字、货币符、百分比、日期、时间和字符串变量的与特定语言相关的用户消息。

异常

ParseException是在这个包中定义的唯一的异常。它不是 RuntimeException的子类，所以，它必须在throws语句中被捕获或定义。

ParseException	在对一个字符串进行语义分析时，如果发生了一个不可预料的错误，该异常将被抛出。

字符串整理

下面的类定义了用于执行对地区敏感的字符串进行比较的操作，以便对自然语言的文本进行查找和按字母排序。Collator是抽象超类，RuleBasedCollator是它的子类。RuleBasedCollator使用CollationElementIterator和CollationKey。

Collator	一个抽象超类，用于执行对地址区敏感的字符串进行比较的操作，以便对自然语言文本进行查找或按字母顺序排序。
RuleBasedCollator	一个具体类，用于进行地区敏感的字符串比较，以便对自然语言文本进行查找或按字母顺序排序。
CollationElementIterator	一个具体的迭代器类，它根据在一个Collator对象中说明的规则对一个与地区相关的字符串中的每一个字符进行迭代。
CollationKey	一个具体类，代表在一个 Collator对象中说明的规则下的一个字符串的排序关键字。它用于对字符串进行快速排序。

字符串迭代

下面的类定义了从以用户字符为基础的文本字符串中寻找和获取逻辑断点的操作。

BreakIterator	一个抽象超类，在其中定义一个用于在文本字符串(字符、词、句子、潜在的行断点)中查找和获取逻辑断点位置的操作。
StringCharacterIterator	一个具体类，用于在一个统一码字符串中进行向前或向后迭代的类。它实现了CharacterIterator接口。
CharacterIterator	一个用于在一个统一码字符串中进行向前或向后迭代的接口。

文本属性类

文本属性类提供了把属性和一部分文本相关联的能力。一个属性是一种特性，例如语种、发音、字体、大小或颜色。这些类允许定义属性关键字/值对和注释值，允许存储属性化的字符串，允许迭代字符和它们的关键字/值对。

1.2 AttributedCharacterIterator	一个用于在文本及它的属性信息中进行迭代，获取属性关键字和值，判断属性如何操作的接口。
1.2 AttributedCharacterIterator.Attribute	一个静态内部类，用于定义属性关键字常量和用于序列化、比较、排序和调试这些关键字的方法。
1.2 Annotation	一个具体类，被用作一个文本属性值的包容器，这个值只在它当前的文本范围内适用，如发音。
1.2 AttributedString	一个用于包括文本及其属性信息的具体类。