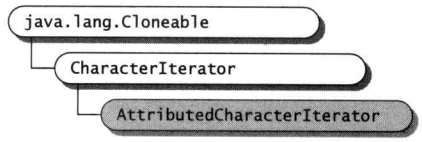
java.text

AttributedCharacterIterator



语法

public interface AttributedCharacterIterator extends CharacterIterator

描述

AttributedCharacterIterator是一个通过属性文本为迭代程序定义方法的接口，它由统一码字符和一些相关的属性信息所组成。这个迭代器的目的是实现对字符及属性关键字和值，同时也包括对属性开始和停止使用的字符下标的检索。返回的字符标志受到迭代器的范围的限制，返回的属性信息为包含当前字符的场所限制。

这个迭代器是通过调用文本存储对象的 getIterator()来创建。java.text包包括属性文本的一个实现:AttributedString。这个文本存储的对象是基于 String的 。或者，文本存储器可以是基于其他类型字符存储器的自定义存储类型。下面的部分描述了术语属性、属性关键字、属性值、属性场。

属性

属性是一个可以被增加到文本中以丰富和标识它的特征。属性包括了语言、发音、字体、大小、样式、对齐、颜色、场指示、上标、宽度等等。每一个属性都可以被加入到整个或部分文本中。属性由关键字/值对组成，这里关键字必须是 AttributedCharacterIterator.Attribute类型，值可以是任意对象或null。特殊的关键字可能需要特殊类型的值相对应。

一个给定字符的位置可以与一个或更多的单一关键字相联系，其中的每一个都没有值或其中一个可能是空值。为关键字在一个字符位置增加另一个值会替换掉前面的值。

在图8中，关键字LANGUAGE对文本第一部分值是 Locale.ENGLISH，对第二部分的值是 Locale.FRENCH。整个文本中关键字SIZE值是12(表示12磅)。

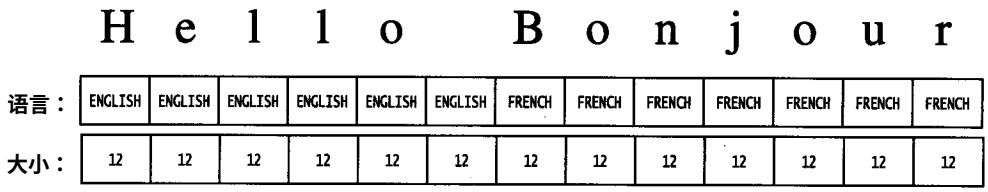


图8 每个字符每个属性允许至多一个值

注意：术语属性并非在整个 java.text包均用法统一；这种不一致性可能会造成混淆。根据上下文，它会指三个不同的情况：属性关键字，属性值，或属性关键字／值对。例如，类 AttributedCharacterIterator.Attribute 定义属性关键字，方法 getAttribute()(在Attributed

CharacterIterator中)返回属性值，方法getAttribute()返回属性关键字／值对。为避免这种混淆，本书使用统一的术语属性关键字和属性值分别指一个关键字和一个值，用属性关键字/值对或(偶尔)属性表示一个关键字／值对。

属性关键字

属性关键字用于AttributedCharacterIterator(和AttributedString)实例中，标识可以增加到文本范围的有用的元信息。如LANGUAGE、FONT、SIZE和WEIGHT。属性关键字必须是单一的且是AttributedCharacterIterator.Attribute类型。每一个属性关键字都可以通过关键字／值对的映射与一个值相关联。

属性关键字是由内部类AttributedCharacterIterator.Attribute或它的一个子类所定义。这个类也可以为序列化、比较、排序和调试那些关键字定义有用的方法。在这个类中定义的特殊的属性关键字是静态域常量LANGUAGE、READING和INPUT_METHOD_SEGMENT。其他的属性关键字，例如FONT、SIZE和WEIGHT，在子类例如java.awt.font.TextAttribute中定义。

可以使用预定义的关键字，或通过继承AttributedCharacterIterator.Attribute类创建自定义的属性关键字。为了更加有效，属性关键字必须保证唯一。也就是，属性关键字表示一个且仅指一个对象。这样就能够通过使用运算符==而不是equals()来比较不同的属性关键字。

属性值

一个属性关键字可以是未定义的、空的，或一个允许使用的类型值 (如本书中在它的域描述所指定)。例如，LANGUAGE关键字要求一个Locale值对。属性值是不可改变的或不能被客户或存储器所改变的。它们通过引用传递，而不能是克隆的。这是为了获得更好的性能和避免其他对象“隐藏在迭代器背后”进行对属性值的更改。

添加和获取属性关键字/值对

可以通过使用存储类的方法增加属性关键字/值，例如在AttributedString中的add Attribute()。可以通过使用AttributedCharacterIterator中的方法获得属性关键字/值对，例如getAttribute()。方法add Attribute()并不检查值的类型;如果传给它的是一个对象(而不是一个基本元素)，那么它接受这个值。

多个属性关键字/值对可以归入映象。这使得可以立即设置和获得多重属性，例如使用addAttributes()。

因为一个属性值可以是一个注释，当从一个迭代器获得属性时，需要考虑检查这个值是否可能是一个注释。

属性场

为AttributedCharacterIterator的实例定义属性场是很方便的。一个属性场是属性关键字和值没有改变的最大文本范围。更进一步讲，一个场是这样定义的:

- 在整个范围属性值是未定义的或空的
- 属性值是定义的，而且在整个范围中有着同样非空的值

getRunStart()和getRunLimit()方法为整个文本的场返回下标，这些场受到迭代器范围的限制。

对于这个类中的所有get方法，返回的属性信息受到包含当前字符的场的限制。

注释

一个注释对象是属性值的一个特殊类型，它的行为像一种“块”的属性。这种包容器避免了属性值被认为是较大的属性场的一部分或被分解成小的场。下面是注释的特征：

- 属性关键字/值对应用的文本范围对于范围的语义很重要。这意味着属性不能被应用于它应用的文本范围的子范围，并且，如果两个相近的文本范围对于属性关键字有相同的值，属性仍不能被作为一个整体应用于复合范围。
- 如果指定的文本被改变，属性的关键字或它的值通常不能应用。

要想获得关于注释的更多信息，可以参考Annotation类。

使用

给定属性文本，可以通过调用getIterator()来为它创建一个迭代器：

AttributedCharacterIterater iter = attrStr.getIterator();

可以使用在AttributedCharacterIterator中的任意方法或那些由CharacterIterator继承的方法来转到任意的字符或属性，并为一个或更多的属性获得场起始和结束的字符位置。

成员概述	
属性关键字/值方法	
getAllAttributeKeys()	检索在整个迭代器的文本范围定义的属性关键字的集合。
getAttribute()	检索当前字符的属性值，以获得一个特殊的关键字。
getAttributes()	检索当前字符的关键字/值对的映射表。
属性场方法	
getrRunStart()	检索包括当前字符的属性场第一个字符的下标。
getRunLimit()	检索包括当前字符的属性场之后的字符的下标志。

参见

AttributedCharacterIterator. Attribute、AttributedString、Annotation。

示例

这个例子创建了一个最初没有属性的属性字符串，然后为它增加两个属性。它为整个字符串增加了法语属性，然后为描述如何对每个单词发音增加了读属性。例如，法语单词“ vous”发音“ vu”(用法语语音)。发音字符串是一个READING属性值，它封装在注释中。每一个注释对象保证了发音只对它的原始单词有效，而对单词的子字符串或父字符串无效 (参见Annotation类)。然后读属性场被打印出来。

```
import java.text.AttributedString;
import java.text.AttributedCharacterIterator;
import java.text.Annotation;
import java.util.Locale;

class Main {
    public static void main(String[] args) {
        // Create an attributed string.
```

```
        AttributedString attrStr = new AttributedString("Vous l'aimez.");

        // Add a French language attribute key/value pair to entire string.
        attrStr.addAttribute(
            AttributedCharacterIterator.Attribute.LANGUAGE,
            Locale.FRENCH);

        // Assign READING constant to a variable.
        AttributedCharacterIterator.Attribute attrREAD =
            AttributedCharacterIterator.Attribute.READING;

        // Add reading attribute key/value pairs for each subrange.
        attrStr.addAttribute(attrREAD, new Annotation("vu"), 0, 4);  // Vous
        attrStr.addAttribute(attrREAD, new Annotation("l"), 5, 7);   // l'
        attrStr.addAttribute(attrREAD, new Annotation("eme"), 7, 12); // aimez

        // Get the string's iterator and print the string.
        AttributedCharacterIterator iter = attrStr.getIterator();
        printText(iter);

        // Print each run of the READING attribute.
        iter.first();
        while (iter.current() != AttributedCharacterIterator.DONE) {
            printToRunLimit(iter);
        }
    }

    public static void printText(AttributedCharacterIterator aci){
        // print the entire text using the character iterator
        do {
            System.out.print(aci.current());
            aci.next();
        } while (aci.current() != AttributedCharacterIterator.DONE) ;
        System.out.println("");
    }

    // Print the run start, run limit, string and attribute value.
    public static void printToRunLimit(AttributedCharacterIterator aci) {
        // Get the run limit of the READING attribute.
        int runStart = aci.getRunStart(
            AttributedCharacterIterator.Attribute.READING);
        printInColumn("" + runStart, 5);
        int runLimit = aci.getRunLimit(
            AttributedCharacterIterator.Attribute.READING);
        printInColumn("" + runLimit, 5);
        Object attr = aci.getAttribute(
            AttributedCharacterIterator.Attribute.READING);

        // Print to the run limit
        System.out.print("\"");
        while (aci.getIndex() < runLimit) {
            System.out.print(aci.current());
            aci.next();
        }
        System.out.print("\"   ");
        printInColumn("", 10 - (runLimit - runStart));
        System.out.println(attr);
    }

    // Print string, then pad spaces to a particular vertical column
    static void printInColumn(String str, int col) {
        System.out.print(str);
        for (int p = str.length(); p < col; ++p) {
            System.out.print(" ");
        }
    }
}
```

输出

```
> java Main
Vous l'aimez.
0    4    "Vous"          java.text.Annotation[value=vu]
4    5    " "             null
5    7    "l'"            java.text.Annotation[value=l]
7    12   "aimez"         java.text.Annotation[value=eme]
12   13   "."             null
```

getAllAttributeKeys()

目的 检索在整个迭代器的文本范围定义的属性关键字的集合。

语法 public abstract Set getAllAttributeKeys()

返回 在整个迭代器的文本范围定义的所有属性关键字的非空集合。如果没有属性被定义，那么集合为空。(按照定义，一个集合没有重复元素。)

示例 这个示例创建了一个单词的AttributedString实例，然后为它增加了LANGUAGE和READING属性。程序使用getAllAttributeKeys()打印出所有关键字，然后使用getAttributes()打印出所有属性关键字/值对。

```
import java.text.AttributedString;
import java.text.AttributedCharacterIterator;
import java.text.Annotation;
import java.util.Locale;

class Main {
    public static void main(String[] args) {

        // Create an attributed string.
        AttributedString attrStr = new AttributedString("Vous");

        // Add a French language attribute key/value pair to entire string.
        attrStr.addAttribute(
            AttributedCharacterIterator.Attribute.LANGUAGE,
            Locale.FRENCH);

        // Assign READING constant to a variable.
        AttributedCharacterIterator.Attribute attrREAD =
            AttributedCharacterIterator.Attribute.READING;

        // Add reading attribute key/value pairs for each subrange.
        attrStr.addAttribute(attrREAD, new Annotation("vu"), 0, 4);  // Vous

        // Get the string's iterator and print the string.
        AttributedCharacterIterator iter = attrStr.getIterator();
        iter.getAttributes();

        // Print each of the keys.
        iter.first();
        System.out.println("Keys: ");
        printAttrKeys(iter);
        System.out.println("");

        // Print each of the key/value pairs.
        iter.first();
        System.out.println("Key/Value Pairs: ");
        printAttrKeyValuePairs(iter);
    }

    // Print each of the keys.
    public static void printAttrKeys(AttributedCharacterIterator aci) {
        Object[] attrArray = aci.getAllAttributeKeys().toArray();
```

```
            for(int i=0; i<attrArray.length; i++) {
                System.out.println(attrArray[i]);
            }
        }

        // Print each of the key/value pairs.
        public static void printAttrKeyValuePairs(
                                        AttributedCharacterIterator aci) {
            Object[] attrKeyArray = aci.getAttributes().keySet().toArray();
            Object[] attrValueArray = aci.getAttributes().values().toArray();
            for(int i=0; i<attrKeyArray.length; i++) {
                System.out.println(attrKeyArray[i]);
                System.out.println(attrValueArray[i]);
                System.out.println("");
            }
        }
    }
```

输出

```
> java Main
Keys:
java.text.AttributedCharacterIterator$Attribute(language)
java.text.AttributedCharacterIterator$Attribute(reading)

Key/Value Pairs:
java.text.AttributedCharacterIterator$Attribute(language)
fr

java.text.AttributedCharacterIterator$Attribute(reading)
java.text.Annotation[value=vu]
```

getAttribute()

目的	检索当前字符的属性值，以获得一个特殊的关键字。
语法	public abstract Object getAttribute(AttributedCharacterIterator.Attribute attribute)
参数	
attribute	被请求值的属性的非空关键字。
返回	这个迭代器的当前字符指定属性的值。如果没有为这个字符定义属性关键字或属性值，或属性值为null时，返回null。
示例	参见类的示例。

getAttributes()

目的	检索当前字符的关键字/值对的映象。
语法	public abstract Map getAttributes()
返回	对当前字符的属性关键字 /值的映象 (按照定义，映射表不能包含相同的关键字)。
示例	参见getAllAttributeKeys()。

getRunLimit()

目的	检索包括当前字符的属性场之后的字符的注释。
语法	public abstract int getRunLimit() public abstract int getRunLimit(AttributedCharacterIterator.Attribute attribute) public abstract int getRunLimit(Set attributes)

描述	这种方法查找包括当前字符的属性场，并返回场之后的 int型的第一个字符的下标(基于0的字符位置)。所有的形式都是基于包含当前字符的场的。 第一种形式返回对于所有属性关键字的场之后的第一个字符的下标。这种场是所有属性都未改变的文本的最大范围。 第二种形式返回对于给定属性关键字的场之后的第一个字符的下标。 第三种形式返回对于给定属性关键字集合的场之后的第一个字符的下标。这种场是所有属性都未改变的文本的最大范围。
参数	
attribute	属性关键字，AttributedCharacterIterator.Attribute的实例，它的场被要求给予限制。
attributes	场被要求给予限制的属性关键字集合。这些关键字是 AttributedCharacterIterator. Attribute的实例。
返回	对于包含当前值的给定属性的场之后的第一个字符的下标。
示例	见类的示例。

getRunStart()

目的	检索包括当前字符的属性场第一个字符的下标。

语法	`public abstract int getRunStart()`
	`public abstract int getRunStart(`
	`  AttributedCharacterIterator.Attribute attribute)`
	`public abstract int getRunStart(Set attributes)`

描述	这种方法查找包括当前字符的属性场，并返回之后 int型的第一个字符的下标(基于0的字符位置)。所有的形式都是基于包含当前字符的场的。 第一种形式返回对于所有属性的场的第一个字符。 第二种形式返回对于给定属性关键字的场的第一个字符的下标。 第三种形式返回对于给定属性关键字集合的场的第一个字符的下标。这种场是所有属性都未改变的文本的最大范围。
参数	
attribute	属性关键字，AttributedCharacterIterator.Attribute的实例，它的场被要求给予限制。
attributes	场被要求给予限制的属性关键字设置，这些关键字是 AttributedCharacterIterator.Attribute的实例。
返回	对于包含当前字符的给定属性的场的第一个字符的下标。
示例	见类的示例。