String vs. StringBuffer

Der Inhalt eines String-Objektes kann nicht verändert werden, während ein StringBuffer-Objekt eine unbegrenzte Anzahl veränderbarer Zeichen aufnehmen und effizient eine Zeichenkette hinzufügen kann.

Class String

General methods	description
charAt(int)	Returns the character at the specified index.
getChars(int, int, char[], int)	Copies characters from this string into the destination character array.
length()	Returns the length of this string.
replace(char, char)	Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar.
toCharArray()	Converts this string to a new character array.
toString()	This object (which is already a string!) is itself returned.
toLowerCase(...)	Converts this string to lowercase.
toUpperCase(...)	Converts this string to uppercase.
valueOf(...)	Returns the string representation of the argument.

Beachte:

length bei einem Array ist eine Variable, length für einen String ist eine Methode!

```
String s;
char[] ca;

...
l = s.length();  // Methode
l = ca.length;   // Variable
```

Class String (cont.)

methods for search/substring	description
endsWith(String)	Tests if this string ends with the specified suffix.
startsWith(String, ...)	Tests if this string starts with the specified prefix.
indexOf(int, ...)	Returns the index within this string of the first occurrence of the specified character.
indexOf(String, ...)	Returns the index within this string of the first occurrence of the specified substring.
indexOf(String, int)	Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.
lastIndexOf(int, ...)	Returns the index within this string of the last occurrence of the specified character.
lastIndexOf(String, ...)	Returns the index within this string of the rightmost occurrence of the specified substring.
lastIndexOf(String, int)	Returns the index within this string of the last occurrence of the specified substring.
substring(int, ...)	Returns a new string that is a substring of this string.

methods for comparison	description
compareTo(String)	Compares two strings lexicographically.
equals(Object)	Compares this string to the specified object.
equalsIgnoreCase(String)	Compares this String to another object.

Class StringBuffer

methods	description
append(...)	**Appends the string representation of the argument to the string buffer.**
capacity()	**Returns the current capacity of the String buffer.**
charAt(int)	**Returns the character at a specific index in this string buffer.**
ensureCapacity(int)	**Ensures that the capacity of the buffer is at least equal to the specified minimum.**
getChars(int, int, char[], int)	**Characters are copied from this string buffer into the destination character array dst.**
insert(int, ...)	**Inserts the string representation of the argument into this string buffer.**
length()	**Returns the length (character count) of this string buffer.**
reverse()	**The character sequence contained in this string buffer is replaced by the reverse of the sequence.**
setCharAt(int, char)	**The character at the specified index of this string buffer is set to ch.**
setLength(int)	**Sets the length of this String buffer. If the new length is less than the current length of the string buffer, the string buffer is truncated to contain exactly the number of characters given by the new length.**
toString()	**Converts to a string representing the data in this string buffer.**

Beispiel StringDemo

```
/**
 *
 * Aufgabe:
 * --------
 * 1) Strings, Stringmethoden
 * 2) Character Arrays
 *
 * @author  Peter René Dietmüller
 * @version 1.0, 11/98
 *
 **/
public class StringDemo {

  public static void main(String args[]) {

    int    i;
    String s;

    /* -- Strings -- */
    s = "J a v A";
    System.out.println("content >" + s);
    System.out.println("length  >" + s.length());
    System.out.println("[1, 3]  >" + s.substring(1,3));
    System.out.println("[3, ]   >" + s.substring(3));
    System.out.println("start   >" + s.substring(0, 1));
    System.out.println("end     >" +
                       s.substring(s.length() - 1));
    System.out.println("start=J >" + s.startsWith("J"));
    System.out.println("end=JA  >" + s.endsWith("JA"));
    System.out.println("UpCase  >" + s.toUpperCase());
    System.out.println("LoCase  >" + s.toLowerCase());
    System.out.println();
```

```
content >J a v A
length  >7
[1, 3]  > a
[3, ]   > v A
start   >J
end     >A
start=J >true
end=JA  >false
UpCase  >J A V A
LoCase  >j a v a
```

```
    for (i = 0; i < s.length(); i++)
      System.out.println(i + ". char.: " + s.charAt(i));
    System.out.println();
```

```
0. char.: J
1. char.:
2. char.: a
3. char.:
4. char.: v
5. char.:
6. char.: A
```

```
    /* -- String: trim; NUR Anfang und Ende -- */
    s = "  J a v  A  ";
    System.out.println("content >" + s);
    System.out.println("length  >" + s.length());
    System.out.println("trim    >" + s.trim());
    System.out.println();
```

```
content >  J a v  A
length  >12
trim    >J a v  A
```

```
    /* -- String: equals, compareTo -- */
    String s1 = "JaVa";
    String s2 = "   JaVa  ";
    System.out.println("s1: " + s1);
    System.out.println("s2: " + s2);
    System.out.println("equals             " +
                       s1.equals(s2));
    System.out.println("equalsIgnoreCase:  " +
                       s1.equalsIgnoreCase(s2));
    System.out.println("equals(trim):      " +
                       s1.equals(s2.trim()));
```

```
s1: JaVa
s2:    JaVa
equals             false
equalsIgnoreCase:  false
equals(trim):      true
```

```
    System.out.println("compareTo s1,s2:   " +
                       s1.compareTo(s2));
    System.out.println("compareTo s2,s1:   " +
                       s2.compareTo(s1));
    System.out.println("compareTo(trim):   " +
                       s1.compareTo(s2.trim()));
    System.out.println("compareTo(LoCase): " +
                       s1.compareTo(s.toLowerCase()));
    System.out.println();
```

```
compareTo s1,s2:   42
compareTo s2,s1:   -42
compareTo(trim):   0
compareTo(LoCase): 42
```

```
    /* -- String: indexOf, lastIndexOf -- */
    s = "Der Bruder kommt nach der LVA.";
    System.out.println(s);
    for(i = 0; i < 6; i++)
      System.out.println("Vorkommen: " +
                         s.indexOf("der"));
    System.out.println();
```

```
Der Bruder kommt nach der LVA.
Vorkommen: 7
Vorkommen: 7
Vorkommen: 7
Vorkommen: 7
Vorkommen: 7
Vorkommen: 7
```

```
    System.out.println("!!und nun richtig!!");
    i = s.indexOf("der");
    while (i >= 0) {
      System.out.println("Vorkommen: " + i);
      i = s.indexOf("der", i + 1);
    }
    System.out.println();
```

```
!!und nun richtig!!
Vorkommen: 7
Vorkommen: 22
```

```
    System.out.println("!!und nun von hinten!!");
    i = s.lastIndexOf("der");
    while (i >= 0) {
      System.out.println("Vorkommen: " + i);
      i = s.lastIndexOf("der", i - 1);
    }
    System.out.println();
```

```
!!und nun von hinten!!
Vorkommen: 22
Vorkommen: 7
```

```
    /* -- String: replace -- */
    System.out.println("Ersetzen d durch s:");
    System.out.println(s.replace('d','s'));
    System.out.println();
```

```
Ersetzen d durch s:
Der Bruser kommt nach ser LVA.
```

```
    /* -- Character Arrays -- */
    // char ca[] = "java"; geht nicht!!
    char ca[] = {'j','a','v','a'};
    System.out.println("content: " + ca);
    System.out.println("length:  " + ca.length);
    System.out.println("start:   " + ca[0]);
    System.out.println("end:     " + ca[ca.length - 1]);
    for (i = 0; i < ca.length; i++)
      System.out.println(i + ". valueOf: " +
                         String.valueOf(ca[i]));
    System.out.println();
```

```
content: java
length:  4
start:   j
end:     a
0. valueOf: j
1. valueOf: a
2. valueOf: v
3. valueOf: a
```

```
    /* -- Character Array --> String -- */
    ca[0] = 'J';
    ca[1] = 'a';
    ca[2] = 'v';
    ca[3] = 'a';
    s = new String(ca);
    System.out.println("content: " + s);
    System.out.println("length:  " + s.length());
    System.out.println();
```

```
content: Java
length:  4
```

```
    /* -- Character Array: teilweise leer -- */
    ca = new char[5];
    ca[1] = 'x';
    ca[3] = 'y';
    System.out.println("content: " + ca);
    System.out.println("length:  " + ca.length);
    for (i = 0; i < ca.length; i++)
      System.out.println(i + ". char: " + ca[i]);
    System.out.println();
```

```
content:
length:  5
0. char:
1. char: x
2. char:
3. char: y
4. char:
```

```
    /* -- String --> Character Array -- */
    ca = s.toCharArray();
    System.out.println("s.toCharArray(): " + ca);
```

```
s.toCharArray(): Java
```

```
    ca = new char[s.length()];
    s.getChars(2, 4, ca, 1);
    System.out.println("s:                " + s);
    System.out.println("getchars(2, 4, 1): " + ca);
```

```
s:                Java
getchars(2, 4, 1):
```

```
    ca = new char[s.length()];
    s.getChars(0, 2, ca, 0);
    System.out.println("s:                  " + s);
    System.out.println("getchars(0, 2, 0): " + ca);

s:                  Java
getchars(0, 2, 0): Ja

  }

}
```

Beispiel StringBufferDemo

```
/**
 *
 * Aufgabe:
 * --------
 * 1) Automatische Erweiterung eines StringBuffers
 * 2) Einlesen einer Datei in einen StringBuffer
 *
 * @author  Peter René Dietmüller
 * @version 1.0, 11/98
 *
 **/

import java.io.*;

public class StringBufferDemo {

  /* -- Hauptprogramm -- */
  public static void main(String[] args)
                    throws java.io.IOException {

    BufferedReader in;
    String         line;
    StringBuffer   buffer;

    /* -- StringBuffer in 10er Schritten erweitern -- */
    buffer = new StringBuffer();
    for (int i = 0; i < 500; i++) {
      System.out.println(buffer.length() + " " +
                         buffer.capacity());
      buffer.append("1234567890");
    }

    /* -- Einlesen einer Datei -- */
    buffer = new StringBuffer();
    in = new BufferedReader(
             new InputStreamReader(System.in));
    line = in.readLine();    // erste Zeile lesen
    while (line != null) {
      buffer.append(line);
      buffer.append("\n");
      line = in.readLine();  // nächste Zeile lesen
    }
```

```
    System.out.println("Ergebnis:");
    System.out.println("=========");
    System.out.println("Größe: " + buffer.capacity());
    System.out.println("Länge: " + buffer.length());
    System.out.println("Inhalt:");
    System.out.println(buffer);

  }

}
```

Ausgabe StringBufferDemo

```
0 16
10 16
20 34
30 34
40 70
50 70
60 70
70 70
80 142
90 142
100 142
110 142
120 142
130 142
140 142
150 286
(... einige Zeilen gelöscht ...)
280 286
290 574
(... einige Zeilen gelöscht ...)
570 574
580 1150
(... einige Zeilen gelöscht ...)
1150 1150
1160 2302
(... einige Zeilen gelöscht ...)
2300 2302
2310 4606
(... einige Zeilen gelöscht ...)
4600 4606
4610 9214
(... einige Zeilen gelöscht ...)
4990 9214
Ergebnis:
=========
Größe: 2302
Länge: 2069
Inhalt:
; for 16-bit app support
[fonts]
(... einige Zeilen gelöscht ...)
```

Beispiel WordCount

```
/**
 *
 * Aufgabe:
 * --------
 * Schreiben Sie ein Programm, das zeilenweise vom
 * Standardeingabestrom (stdin) einliest und die Worte
 * in diesem Eingabestrom zählt.
 * Worte sind durch ein oder mehrere Trennzeichen
 * voneinander getrennt. Als Trennzeichen sollen
 * die Zeichen ",. :;+*?!()<>" (und neue Zeile) ver-
 * wendet werden.
 * Dem Programm sollen auch andere Trennzeichen als
 * Parameter übergeben werden können. Dann sollen
 * die übergebenen Trennzeichen statt der Standard-
 * trennzeichen verwendet werden.
 *
 * @author  Peter René Dietmüller
 * @version 1.0, 11/98
 *
 **/


import java.io.*;

public class WordCount {


  public static boolean IsDel(char c, String del) {

    int i;

    /* -- Ist aktuelles Zeichen ein Delimiter ? -- */
    i = 0;
    while ((i < del.length()) && (c != del.charAt(i)))
      i++;

    return (i < del.length());

  }
```

```
  public static int CountWords(String s, String del)
  {
    int  i;
    int  wc;  // word count

    wc = 0;
    i = 0;
    while (i < s.length()) {

      /* -- Delimiter überlesen -- */
      while ((i < s.length()) &&
             (IsDel(s.charAt(i), del)))
        i++;

      /* -- neues Wort ? -- */
      if (i < s.length()) wc++;

      /* -- Wort überlesen -- */
      while ((i < s.length()) &&
             (!IsDel(s.charAt(i), del)))
        i++;

    }

    return wc;

  }


  public static void main(String[] args)
                    throws IOException {

    String         delimiters;
    BufferedReader in;
    String         line;
    int            wc;

    /* -- Delimiters bestimmen -- */
    if (args.length == 0)
      delimiters =new String(",. :;+*?!()<>");
    else
      delimiters = args[0];
```

```
    /* -- Datei einlesen und Worte zählen -- */
    wc = 0;
    in = new BufferedReader(
             new InputStreamReader(System.in));
    line = in.readLine();
    while (line != null) {
      System.out.println(line);
      wc += CountWords(line, delimiters);
      line = in.readLine();
    }

    /* -- Ergebnis -- */
    System.out.print("Sie haben " + wc);
    System.out.print(wc == 1 ? " Wort" : " Worte");
    System.out.print(" eingegeben!\n");

   }
}
```

Worthäufigkeiten

```
public class WortHaeufigkeiten
{
    // Speichern der einzelnen Wörter
    private static Vector woerter=new Vector();
    // Speichern der zugehörigen Häufigkeiten
     private static Vector haeufigkeit=new Vector();
    // Trennzeichen
    private static final String delimiter=
                                 " \r\n\t,.:;!'()\"";

/* Alle Trennzeichen im String zeile überlesen, wobei
   bei startIndex begonnen wird. Der Index des ersten
   nachfolgenden Zeichens, das kein Trennzeichen mehr
   ist wird zurückgegeben */
private static int readWhitespaces(
                              String zeile,int startIndex)
{
    int curIndex=startIndex;
    while(curIndex<zeile.length() &&
            delimiter.indexOf(zeile.charAt(curIndex))>=0)
        curIndex++;
    return curIndex;
}
```

```
// Wörter in zeile herausschälen und Häufigkeit erhöhen
private static void ZaehleWoerter(String zeile)
{
    int index=0;
    // Etwaige führende Trennzeichen überlesen
    index=readWhitespaces(zeile,index);
    while(index<zeile.length())
    {
        // Wort lesen
        int startIndex=index;
        int endIndex=index;
        while(endIndex<zeile.length() &&
             delimiter.indexOf(zeile.charAt(endIndex))<0)
            endIndex++;
        // endIndex steht auf erstem Whitespace nach Wort
        String wort=zeile.substring(startIndex,endIndex);
        index=endIndex;
        // Wort suchen
        Enumeration enum=woerter.elements();
        int wortIndex=-1;
        while(enum.hasMoreElements() && wortIndex<0)
        {
            String cur=(String)enum.nextElement();
            if(cur.equalsIgnoreCase(wort))
            {
                // Wort gefunden. Aus Häufigkeitsvektor
                // element mit selber Position erhöhen
                wortIndex=woerter.indexOf(cur);
                Integer anzahl=(Integer)
                        haeufigkeit.elementAt(wortIndex);
                haeufigkeit.setElementAt(new Integer(
                         anzahl.intValue()+1),wortIndex);
            }
        }
        if(wortIndex<0)
        {
            // Neu eintragen, da noch nicht enthalten
            woerter.addElement(wort);
            haeufigkeit.addElement(new Integer(1));
        }
        // Trennung mit Whitespaces überlesen
        index=readWhitespaces(zeile,index);
    }
}
```

```
static public void main(String args[])
{
    BufferedReader in;
    String zeile;
    System.out.println("Worthäufigkeiten zählen:");
    // Argumente prüfen
    if(args.length!=1) {
        System.out.println("Benutzung: WortHaeufigkeiten
Source-Datei");
        System.exit(1);
    }
    // Datei einlesen und Worthäufigkeiten zählen
    try
    {
        in=new BufferedReader(new FileReader(args[0]));
        zeile=in.readLine();
        while(zeile!=null)
        {
            System.out.println(zeile);
            ZaehleWoerter(zeile);
            zeile=in.readLine();
        }
    }
    catch(IOException e){
        System.out.println(e.toString());
        System.exit(1);
    }
    // Ergebnis ausgeben
    System.out.println("\n\nAnzahl der Wörter: "+
                       woerter.size());
    Enumeration enum=woerter.elements();
    while(enum.hasMoreElements())
    {
        String wort=(String)enum.nextElement();
        int wortIndex=woerter.indexOf(wort);
        Integer anzahl=(Integer)
                        haeufigkeit.elementAt(wortIndex);
        System.out.println("Das Wort \""+wort+
                         "\" kommt "+anzahl+" mal vor.");
    }
    System.out.println("\nEnde der Häufigkeitsliste");
}

}
```