StringBuffer, StringBuilder, StringTokenizer

In the following example you will learn about StringBuffer class. This example explains how you can use functions provided by the StringBuffer class like append, insert, reverse, setCharAt, charAt, length, deleteCharAt, substring, delete, capacity etc. to manipulate the string operation in your program.

The StringBuffer class is used to represent characters that can be modified. This is simply used for concatenation or manipulation of the strings.

StringBuffer is mainly used for the dynamic string concatenation which enhances the performance. A string buffer implements a mutable sequence of characters. A string buffer is like a String, but can be modified. At any point in time it contains some particular sequence of characters, but the length and content of the sequence can be changed through certain method calls. There are some functions used in the given example. All the functions have been explained below with example :

• append()
  This is the append() function used for the concatenate the string in string buffer. This is better to use for dynamic string concatenation. This function works like a simple string concatenation such as : String str = str + "added string";

• insert()
  This is the insert() function used to insert any string or character at the specified position in the given string.

• reverse()
  This is the reverse() function used to reverse the string present in string buffer.

• setCharAt()
  This is the setCharAt() function which is used to set the specified character in buffered string at the specified position of the string in which you have to set the given character.

• charAt()
  This is the charAt() function which is used to get the character at the specified position of the given string.
substring()

This is the substring() function which is used to get the sub string from the buffered string from the initial position to end position (these are fixed by you in the program).

deleteCharAt()

This is the deleteCharAt() function which is used to delete the specific character from the buffered string by mentioning that's position in the string.

length()

This is the length() function is used to finding the length of the buffered string.

delete()

This is the delete() function is used to delete multiple character at once from \( n \) position to \( m \) position (\( n \) and \( m \) are will be fixed by you.) in the buffered string.

capacity()

This is the capacity() function is used to know about the current characters kept which is displayed like : number of characters + 6.

Code for the program:

```java
import java.io.*;
public class StringBuffer{
    public static void main(String[] args) throws Exception{
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        String str;
        try{
            System.out.print("Enter your name: ");
            str = in.readLine();
            str += ", This is the example of StringBuffer class and it's functions."
            //Create a object of StringBuffer class
            StringBuffer strbuf = new StringBuffer();
            strbuf.append(str);
            System.out.println(strbuf);
            strbuf.delete(0,str.length());
            //append()
            strbuf.append("Hello");
            strbuf.append("World"); //print HelloWorld
            System.out.println(strbuf);
            //insert()
            strbuf.insert(5,"_Java "); //print Hello_Java World
            System.out.println(strbuf);
            //reverse()
            strbuf.reverse();
        }
    }
}
```
System.out.print("Reversed string : ");
System.out.println(strbuf);                 // print dlroW avaJ_olleH
strbuf.reverse();
System.out.println(strbuf);               // print Hello_Java World

//setCharAt()
strbuf.setCharAt(5, ' ');
System.out.println(strbuf);               // print Hello J ava World

//charAt()
System.out.print("Character at 6th position : ");
System.out.println(strbuf.charAt(6));      // print J

//substring()
System.out.print("Substring from position 3 to 6 : ");
System.out.println(strbuf.substring(3,7));  // print lo J

//deleteCharAt()
strbuf.deleteCharAt(3);
System.out.println(strbuf);               // print Helo java World

//capacity()
System.out.print("Capacity of StringBuffer object : ");
System.out.println(strbuf.capacity());     // print 21

//delete() and length()
strbuf.delete(6,strbuf.length());
System.out.println(strbuf);               // no anything

} catch (StringIndexOutOfBoundsException e){
    System.out.println(e.getMessage());
}
}

public class Main {
    public static void main(String[] args) {
        String result = "Hello World";
}

StringBuffer

StringBuilder

The StringBuilder class is a drop-in replacement for StringBuffer in cases where thread safety is not an issue. Because StringBuilder is not synchronized, it offers faster performance than StringBuffer.

In general, you should use StringBuilder in preference over StringBuffer. In fact, the J2SE 5.0 javac compiler normally uses StringBuilder instead of StringBuffer whenever you perform string concatenation as in

System.out.println("The result is "+ result);

All the methods available on StringBuffer are also available on StringBuilder, so it really is a drop-in replacement.
StringTokenizer

The java.util.StringTokenizer allows you to break a string into substrings, or tokens, that are separated by delimiters. The delimiters are whitespace (spaces, carriage returns, etc.) by default but you can defined others.

A StringTokenizer provides an Enumeration object that steps through the tokens:

```java
String str = "This is a string object";
StringTokenizer st = new StringTokenizer (str);
while (st.hasMoreTokens ()) {
    System.out.println (st.nextToken ());
    ...
}
```

On the console this shows:

This
is
a
string
object

An overloaded constructor allows you to specify the delimiters. For example,

```java
String str = "A*bunch*of*stars";
StringTokenizer st = new StringTokenizer (str,"*");
```

This breaks the string into the tokens separated by the "]*" character.

- Exercises:

  1- Create an application program that read in a text file and counts the number of instances of each word. It displays the list of words and the number of occurrences beside each word.

  2- To "capitalize" a string means to change the first letter of each word in the string to upper case (if it is not already upper case). For example, a capitalized version of "Now is the time to act!" is "Now Is The Time To Act!". Write a subroutine named printCapitalized that will print a capitalized version of a string to standard output. The string to be printed should be a parameter to the subroutine. Test your subroutine with a main() routine that gets a line of input from the user and applies the subroutine to it.

Note that a letter is the first letter of a word if it is not immediately preceded in the string by another letter. Recall that there is a standard boolean-valued function Character.isLetter(char) that can be used to test whether its parameter is a letter. There is another standard char-valued function, Character.toUpperCase(char), that returns a capitalized version of the single character passed to it as a parameter. That is, if the parameter is a letter, it returns the upper-case version. If the parameter is not a letter, it just returns a copy of the parameter.