**STRINGBUFFER AND STRINGBUILDER**

* *StringBuffer* and *StringBuilder* classes are mutable as they do not create any new string object when manipulated.
* Therefore, when we need to do various manipulations, such as appending, concatenating, and deleting with string literals, we should use *StringBuilder* and *StringBuffer*.

**Java StringBuffer Class**

* Java StringBuffer class is used to create mutable (modifiable) String objects.
* The StringBuffer class in Java is the same as String class except it is mutable i.e. it can be changed.

**IMPORTANT CONSTRUCTORS OF STRINGBUFFER CLASS**

**StringBuffer()**

* It creates an empty StringBuffer with the initial capacity of 16.

**StringBuffer(String str)**

* It creates a StringBuffer with specified string.

**StringBuffer(int capacity)**

* It creates a StringBuffer with the specified capacity as length.

**MOST COMMONLY USED METHODS OF STRINGBUFFER CLASS**

**StringBuffer append(String obj)**

* Appends the argument to the string buffer.

**Example:**

public class StringBufferExample{

public static void main(String[] args){

StringBuffer sb = new StringBuffer(“Java”);

sb.append(“is a high-level programming language.”);

System.out.println(sb);

}

}

**Output:**

Java is a high-level programming language.

**StringBuffer delete(int start, int end)**

* Deletes the sequence from start to end in the char sequence.

**Example:**

public class StringBufferExample1{

public static void main(String[] args){

StringBuffer sb = new StringBuffer(“Java is a programming language”);

sb.delete(0, 9);

System.out.print(“After deletion: ”+sb);

}

}

**Output:**

After deletion: programming language

**StringBuffer insert(int offset, String obj)**

* Inserts the second argument into the string buffer.
* The first argument indicates the index before which the data is to be inserted.

**Example:**

public class StringBufferExample2{

public static void main(String[] args){

StringBuffer sb = new StringBuffer(“Java programming language”);

sb.insert(4, “ is a”);

System.out.print(“After insertion: ”+sb);

}

}

**Output:**

After insertion: Java is a programming language

**StringBuffer reverse()**

* The sequence of the characters in the string buffer is reversed.

**Example:**

public class StringBufferExample3{

public static void main(String[] args){

StringBuffer sb = new StringBuffer(“mugilan”);

System.out.print(“After reverse: ”+ sb.reverse ());

}

}

**Output:**

After reverse: naligum

**StringBuffer capacity()**

* The capacity() method of the StringBuffer class returns the current capacity of the buffer.
* The default capacity of the buffer is 16.
* If the number of character increases from its current capacity, it increases the capacity by ***(oldcapacity\*2)+2***.
* For example if your current capacity is 16, it will be (16\*2)+2=34.

**Example:**

public class StringBufferExample4{

public static void main(String[] args){

StringBuffer sb = new StringBuffer();

System.out.println(sb.capacity());//default 16

sb.append(“Java”);

System.out.println(sb.capacity());// now 16

sb.append(“is a high-level language uses interpreter”);

System.out.println(sb.capacity());//now (16\*2)+2=34+12=46

}

}

**Output:**

16

16

46

**StringBuffer ensureCapacity()**

* The ensureCapacity() method of the StringBuffer class ensures that the given capacity is the minimum to the current capacity.
* If it is greater than the current capacity, it increases the capacity by (oldcapacity\*2)+2.
* For example if your current capacity is 16, it will be (16\*2)+2=34.

**Example:**

package com.demo.JavaCode;  
  
public class StringBufferCapacityExample {  
 public static void main(String[] args){  
 StringBuilder sb = new StringBuilder();  
 System.out.println(sb.capacity());  
 sb.append("Java");  
 System.out.println(sb.capacity());  
 sb.append("It is high level language uses interpreter");  
 System.out.println(sb.capacity());//34+12  
 sb.ensureCapacity(10);  
 System.out.println(sb.capacity());  
 sb.ensureCapacity(55);  
 System.out.println(sb.capacity());//46\*2=92+2=94  
 }  
}

**Output:**

16

16

46

46

94

**Java StringBuilder Class**

* Java StringBuilder class is used to create mutable (modifiable) String objects.
* The StringBuilder class in Java is the same as String class except it is mutable i.e. it can be changed.

**IMPORTANT CONSTRUCTORS OF STRINGBUILDER CLASS**

**StringBuilder()**

* It creates an empty StringBuilder with the initial capacity of 16.

**StringBuilder(String str)**

* It creates a StringBuilder with specified string.

**StringBuilder(int capacity)**

* It creates a StringBuilder with the specified capacity as length.

**MOST COMMONLY USED METHODS OF STRINGBUILDER CLASS**

**StringBuilder append(String obj)**

* Appends the argument to the string builder.

**Example:**

public class StringBuilderExample{

public static void main(String[] args){

StringBuilder sb = new StringBuilder (“Java”);

sb.append(“is a high-level programming language.”);

System.out.println(sb);

}

}

**Output:**

Java is a high-level programming language.

**StringBuilder delete(int start, int end)**

* Deletes the sequence from start to end in the char sequence.

**Example:**

public class StringBuilderExample1{

public static void main(String[] args){

StringBuilder sb = new StringBuilder (“Java is a programming language”);

sb.delete(0, 9);

System.out.print(“After deletion: ”+sb);

}

}

**Output:**

After deletion: programming language

**StringBuilder insert(int offset, String obj)**

* Inserts the second argument into the string builder.
* The first argument indicates the index before which the data is to be inserted.

**Example:**

public class StringBuilderExample2{

public static void main(String[] args){

StringBuilder sb = new StringBuilder(“Java programming language”);

sb.insert(4, “ is a”);

System.out.print(“After insertion: ”+sb);

}

}

**Output:**

After insertion: Java is a programming language

**StringBuilder reverse()**

* The sequence of the characters in the string builder is reversed.

**Example:**

public class StringBuilderExample3{

public static void main(String[] args){

StringBuilder sb = new StringBuilder (“mugilan”);

System.out.print(“After reverse: ”+ sb.reverse ());

}

}

**Output:**

After reverse: naligum

**StringBuilder capacity()**

* The capacity() method of the StringBuilder class returns the current capacity of the buffer.
* The default capacity of the builder is 16.
* If the number of character increases from its current capacity, it increases the capacity by ***(oldcapacity\*2)+2***.
* For example if your current capacity is 16, it will be (16\*2)+2=34.

**Example:**

public class StringBuilderExample4{

public static void main(String[] args){

StringBuilder sb = new StringBuilder();

System.out.println(sb.capacity());//default 16

sb.append(“Java”);

System.out.println(sb.capacity());// now 16

sb.append(“is a high-level language uses interpreter”);

System.out.println(sb.capacity());//now (16\*2)+2=34+12=46

}

}

**Output:**

16

16

46

**StringBuilder ensureCapacity()**

* The ensureCapacity() method of the StringBuilder class ensures that the given capacity is the minimum to the current capacity.
* If it is greater than the current capacity, it increases the capacity by (oldcapacity\*2)+2.
* For example if your current capacity is 16, it will be (16\*2)+2=34.

**Example:**

package com.demo.JavaCode;  
public class StringBuilderCapacityExample {  
 public static void main(String[] args){  
 StringBuilder sb = new StringBuilder();  
 System.out.println(sb.capacity());  
 sb.append("Java");  
 System.out.println(sb.capacity());  
 sb.append("It is high level language uses interpreter");  
 System.out.println(sb.capacity());//34+12  
 sb.ensureCapacity(10);  
 System.out.println(sb.capacity());  
 sb.ensureCapacity(55);  
 System.out.println(sb.capacity());//46\*2=92+2=94  
 }  
}

**Output:**

16

16

46

46

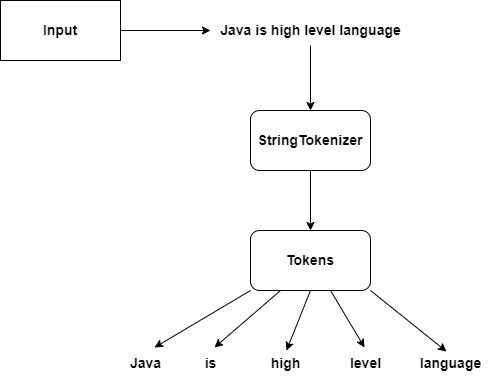
94

**StringTokenizer in Java**

* It allows you to break a string into tokens.
* It is a simple way to break a string.
* It is a legacy class of java.
* In the StringTokenizer class, the delimiters can be provided at the time of creation or one by one to the tokens.

**Note:**

* The StringTokenizer class is depreciated now.
* It is recommended to use the split() method of the string class or the Pattern class that belongs to the java.util.regex package.



**CONSTRUCTORS OF STRINGTOKENIZER CLASS**

3 Constructors defined in StringTokenizer class.

**StringTokenizer(String str)**

* It creates StringTokenizer with specified string.

**StringTokenizer(String str, String delim)**

* It creates StringTokenizer with specified string and delimiter.

**StringTokenizer(String str, String delim, boolean returnValue)**

* It creates StringTokenizer with specified string, delimiter and returnValue.
* If return value is true, delimiter characters are considered to be tokens.
* If it is false, delimiter characters serve to separate tokens.

**METHODS OF STRINGTOKENIZER CLASS**

6 methods of StringTokenizer class.

**boolean hasMoreTokens()**

* It checks if there is more tokens available.

**String nextToken()**

* It returns the next token from the StringTokenizer object.

**String nextToken(String delim)**

* It returns the next token based on the delimiter.

**boolean hasMoreElements()**

* It is the same as hasMoreTokens() method.

**Object nextElement()**

* It is the same as nextToken() but its return type is Object.

**int countTokens()**

* It returns the total number of tokens.

**Example: hasMoreTokens(), nextToken()**

public class StringTokenizerExample{

public static void main(String[] args){

StringTokenizer st = new StringTokenizer(“Java is high level language”, “ ”);

while(st.hasMoreTokens()){

System.out.println(st.nextToken());

}

}

}

**Output:**

Java

is

high

level

language

**Example: nextToken(String delim)**

public class StringTokenizerExample1{

public static void main(String[] args){

StringTokenizer st = new StringTokenizer(“Java,is,high level,language”);

System.out.println(“Next token is: ”+st.nextToken(“,”));

}

}

**Output:**

Next token is: Java

**Example: hasMoreElements(), nextToken()**

public class StringTokenizerExample2{

public static void main(String[] args){

StringTokenizer st = new StringTokenizer(“Java is high level language”);

while(st.hasMoreElements()){

System.out.println(st.nextToken());

}

}

}

**Output:**

Java

is

high

level

language

**Example: hasMoreTokens(), nextElement()**

public class StringTokenizerExample2{

public static void main(String[] args){

StringTokenizer st = new StringTokenizer(“Java is high level language”);

while(st.hasMoreTokens()){

System.out.println(st.nextElement());

}

}

}

**Output:**

Java

is

high

level

language

**Example: countTokens()**

public class StringTokenizerExample1{

public static void main(String[] args){

StringTokenizer st = new StringTokenizer(“Java is high level language”);

System.out.println(“Total no. of tokens: ”+st.countTokens(“,”));

}

}

**Output:**

Total no. of tokens: 5