String Buffer and String Builder

Prepared by

Dr. Rajesh Kumar Ojha
Asst. Prof., CSE, Silicon University

StringBuffer class

StringBuffer class

- Java StringBuffer class is used to created mutable (modifiable) string.
- The StringBuffer class in java is same as String class except it is mutable i.e. it can be changed.

Constructors of StringBuffer class

- **StringBuffer():** creates an empty string buffer with the initial capacity of 16.
- **StringBuffer(String str):** creates a string buffer with the specified string.
- **StringBuffer(int capacity):** creates an empty string buffer with the specified capacity as length.

Methods of StringBuffer class

- **public synchronized StringBuffer append(String s):** is used to append the specified string with this string. The append() method is overloaded like append(char), append(boolean), append(int), append(float), append(double) etc.
- **public synchronized StringBuffer insert(int offset, String s):** is used to insert the specified string with this string at the specified position. The insert() method is overloaded like insert(int, char), insert(int, boolean), insert(int, int), insert(int, float), insert(int, double) etc.
- **public synchronized StringBuffer replace(int startIndex, int endIndex, String str):** is used to replace the string from specified startIndex and endIndex.
- **public synchronized StringBuffer delete(int startIndex, int endIndex):** is used to delete the string from specified startIndex and endIndex.

Methods of StringBuffer class

- **public synchronized StringBuffer reverse():** is used to reverse the string.
- **public int capacity():** is used to return the current capacity.
- **public void ensureCapacity(int minimumCapacity):** is used to ensure the capacity at least equal to the given minimum.
- **public char charAt(int index):** is used to return the character at the specified position.
- **public int length():** is used to return the length of the string i.e. total number of characters.
- **public String substring(int beginIndex):** is used to return the substring from the specified beginIndex.
- **public String substring(int beginIndex, int endIndex):** is used to return the substring from the specified beginIndex and endIndex.

StringBuffer append()

The append() method concatenates the given argument with this string.

```
class A{
        public static void main(String args[]){
        StringBuffer sb=new StringBuffer("Hello ");
        sb.append("Java");//now original string is changed
        System.out.println(sb);//prints Hello Java
        }
}
```

StringBuffer insert()

The insert() method inserts the given string with this string at the given position.

```
class A{
        public static void main(String args[]){
        StringBuffer sb=new StringBuffer("Hello ");
        sb.insert(1,"Java");//now original string is changed
        System.out.println(sb);//prints HJavaello
        }
}
```

StringBuffer replace()

The replace() method replaces the given string from the specified beginIndex and endIndex.

```
class A{
        public static void main(String args[]){
        StringBuffer sb=new StringBuffer("Hello");
        sb.replace(1,3,"Java");
        System.out.println(sb);//prints HJavalo
        }
}
```

StringBuffer delete()

The delete() method of StringBuffer class deletes the string from the specified beginIndex to endIndex.

```
class A{
        public static void main(String args[]){
        StringBuffer sb=new StringBuffer("Hello");
        sb.delete(1,3);
        System.out.println(sb);//prints Hlo
        }
}
```

StringBuffer reverse()

- The reverse() method of StringBuilder class reverses the current string.

```
class A{
        public static void main(String args[]){
        StringBuffer sb=new StringBuffer("Hello");
        sb.reverse();
        System.out.println(sb);//prints olleH
        }
}
```

StringBuilder class

StringBuilder class

Java StringBuilder class is used to create mutable (modifiable) string. The Java StringBuilder class is same as StringBuffer class except that it is non-synchronized.

Constructors of StringBuilder class

- **StringBuilder():** creates an empty string Builder with the initial capacity of 16.
- **StringBuilder(String str):** creates a string Builder with the specified string.
- **StringBuilder(int length):** creates an empty string Builder with the specified capacity as length.

Methods of StringBuilder Class

Method	Description
public StringBuilder append(String s)	is used to append the specified string with this string. The append() method is overloaded like append(char), append(boolean), append(int), append(float), append(double) etc.
public StringBuilder insert(int offset, String s)	is used to insert the specified string with this string at the specified position. The insert() method is overloaded like insert(int, char), insert(int, boolean), insert(int, int), insert(int, float), insert(int, double) etc.
public StringBuilder replace(int startIndex, int endIndex, String str)	is used to replace the string from specified startIndex and endIndex.
public StringBuilder delete(int startIndex, int endIndex)	is used to delete the string from specified startIndex and endIndex.
public StringBuilder reverse()	is used to reverse the string.
public void ensureCapacity(int minimumCapacity)	is used to ensure the capacity at least equal to the given minimum.

Methods of StringBuilder Class

Method	Description
public int capacity()	is used to return the current capacity.
public char charAt(int index)	is used to return the character at the specified position.
public int length()	is used to return the length of the string i.e. total number of characters.
public String substring(int beginIndex)	is used to return the substring from the specified beginIndex.
public String substring(int beginIndex, int endIndex)	is used to return the substring from the specified beginIndex and endIndex.

StringBuilder append()

The append() method concatenates the given argument with this string.

```
class A{
        public static void main(String args[]){
        StringBuilder sb=new StringBuilder("Hello ");
        sb.append("Java");//now original string is changed
        System.out.println(sb);//prints Hello Java
        }
}
```

StringBuilder insert()

The insert() method inserts the given string with this string at the given position.

```
class A{
        public static void main(String args[]){
        StringBuilder sb=new StringBuilder("Hello ");
        sb.insert(1,"Java");//now original string is changed
        System.out.println(sb);//prints HJavaello
        }
}
```

StringBuilder replace()

- The replace() method replaces the given string from the specified beginIndex and endIndex.

```
class A{
        public static void main(String args[]){
        StringBuilder sb=new StringBuilder("Hello");
        sb.replace(1,3,"Java");
        System.out.println(sb);//prints HJavalo
        }
}
```

StringBuilder delete()

- The delete() method of StringBuilder class deletes the string from the specified beginIndex to endIndex.

```
class A{
        public static void main(String args[]){
        StringBuilder sb=new StringBuilder("Hello");
        sb.delete(1,3);
        System.out.println(sb);//prints Hlo
        }
}
```

StringBuilder reverse()

- The reverse() method of StringBuilder class reverses the current string.

```
class A{
        public static void main(String args[]){
        StringBuilder sb=new StringBuilder("Hello");
        sb.reverse();
        System.out.println(sb);//prints olleH
        }
}
```

StringBuffer vs StringBuilder

StringBuffer	**StringBuilder**
1. StringBuffer is *synchronized* i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously.	1. StringBuilder is *non-synchronized* i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously.
2. StringBuffer is *less efficient* than StringBuilder.	2. StringBuilder is *more efficient* than StringBuffer.

toString()

- The toString() method returns the string representation of the object.
- If you print any object, java compiler internally invokes the toString() method on the object. So overriding the toString() method, returns the desired output, it can be the state of an object etc. depends on your implementation.

```
class Student{
          int rollno;
          String name;
          String city;
          Student(int rollno, String name, String city){
          this.rollno=rollno;
          this.name=name;
          this.city=city;
          }
 public String toString(){
//overriding the toString() method
return rollno+" "+name+" "+city;
}
public static void main(String args[]){
        Student s1=new Student(101,"Raj","lucknow");
        Student s2=new Student(102,"Vijay","ghaziabad");
        System.out.println(s1);//compiler writes here s1.toString()
        System.out.println(s2);//compiler writes here s2.toString()
}
}
```

```
O/P:101 Raj lucknow
102 Vijay ghaziabad
```

StringBuilder class

StringTokenizer in Java

- The **java.util.StringTokenizer** class allows you to break a string into tokens. It is simple way to break string.
- It doesn't provide the facility to differentiate numbers, quoted strings, identifiers etc. like StreamTokenizer class. We will discuss about the StreamTokenizer class in I/O chapter.

Constructors of StringTokenizer class

Constructor	Description
StringTokenizer(String str)	creates StringTokenizer with specified string.
StringTokenizer(String str, String delim)	creates StringTokenizer with specified string and delimeter.
StringTokenizer(String str, String delim, boolean returnValue)	creates StringTokenizer with specified string, delimeter 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

Public method	Description
boolean hasMoreTokens()	checks if there is more tokens available.
String nextToken()	returns the next token from the StringTokenizer object.
String nextToken(String delim)	returns the next token based on the delimeter.
boolean hasMoreElements()	same as hasMoreTokens() method.
Object nextElement()	same as nextToken() but its return type is Object.
int countTokens()	returns the total number of tokens.

Example

```
import java.util.StringTokenizer;
public class Simple{
 public static void main(String args[]){
   StringTokenizer st = new StringTokenizer("my name is Jadoo"," ");
     while (st.hasMoreTokens()) {
         System.out.println(st.nextToken());
     }
   }
}
```

Output:

```
my
name
is
Jadoo
```

Thank you