ECAP615

Programming in Java

Harjinder Kaur

Assistant Professor

Learning Outcomes

After this lecture, you will be able to

- learn the basic concept of StringBuffer and StringBuilder Class.
- understand the different constructors of StringBuffer and StringBuilder Class.
- implement the various methods of StringBuffer and StringBuilder Class.
- Differentiate between StringBuffer and StringBuilder Class.

StringBuffer Class

- Java StringBuffer class is used to create mutable string.
- The StringBuffer class in java is same as String class except it is mutable i.e. it can be changed.
- StringBuffer may have characters and substrings inserted in the middle or appended to the end.

StringBuffer Class

Following are the important points about StringBuffer:

- A string buffer is like a String, but can be modified.
- It contains some particular sequence of characters, but the length and content of the sequence can be changed through certain method calls.
- They are safe for use by multiple threads.
- Every string buffer has a capacity.

StringBuffer Constructors

Constructor	Description
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.

StringBuffer Constructors

- StringBuffer( ): It reserves room for 16 characters without reallocation.

Example:

```
StringBuffer s=new StringBuffer();
```

- StringBuffer(int size): It accepts an integer argument that explicitly sets the size of the buffer.

Example:

```
StringBuffer s=new StringBuffer(20);
```

StringBuffer Constructors

- StringBuffer(String str): It accepts a String argument that sets the initial contents of the StringBuffer object and reserves room for 16 more characters without reallocation.

  Example:

  StringBuffer s=new StringBuffer(“Welcome");

StringBuffer Methods

Method	Description
append(String s)	It is used to append the specified string with this string.
insert(int offset, String s)	It is used to insert the specified string with this string at the specified position.
replace(int startIndex, int endIndex, String str)	is used to replace the string from specified startIndex and endIndex.
delete(int startIndex, int endIndex)	is used to delete the string from specified startIndex and endIndex.

StringBuffer Methods

Method	Description
reverse()	It is used to reverse the string.
capacity()	It is used to return the current capacity.
ensureCapacity(int minimumCapacity)	It is used to ensure the capacity at least equal to the given minimum.
charAt(int index)	It is used to return the character at the specified position.
length()	It is used to return the length of the string i.e. total number of characters.
substring(int beginIndex)	It is used to return the substring from the specified beginIndex.
substring(int beginIndex, int endIndex)	It is used to return the substring from the specified beginIndex and endIndex.

StringBuffer append() method Example

```
class StringBufferExample{

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() method Example

```
class StringBufferExample2{

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() method Example

```
class StringBufferExample3{

public static void main(String args[]){

StringBuffer sb=new StringBuffer("Hello");

sb.replace(1,3,"Java");

System.out.println(sb); //prints HJavalo

}

}
```

StringBuffer delete() method Example

```
class StringBufferExample4{

public static void main(String args[]){

StringBuffer sb=new StringBuffer("Hello");

sb.delete(1,3);

System.out.println(sb);//prints Hlo

}

}
```

StringBuffer reverse() method Example

```
class StringBufferExample5{

public static void main(String args[]){

StringBuffer sb=new StringBuffer("Hello");

sb.reverse();

System.out.println(sb); //prints olleH

}

}
```

StringBuffer capacity() method Example

```
class StringBufferExample6{

public static void main(String args[]){

StringBuffer sb=new StringBuffer();

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

sb.append("Hello");

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

sb.append("java is my favourite language");

System.out.println(sb.capacity()); //now (16*2)+2=34 i.e (oldcapacity*2)+2

}

}
```

StringBuffer ensureCapacity() method Example

```
class StringBufferExample7{

public static void main(String args[]){

StringBuffer sb=new StringBuffer();

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

sb.append("Hello");

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

sb.append("java is my favourite language"
);

System.out.println(sb.capacity());

//now (16*2)+2=34 i.e (oldcapacity*2)+
2

sb.ensureCapacity(10);

//now no change

System.out.println(sb.capacity());

//now 34

sb.ensureCapacity(50);

//now (34*2)+2

System.out.println(sb.capacity());

//now 70

}

}
```

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.
- It is available since JDK 1.5.
- StringBuilder class provides an API similar to StringBuffer, but unlike StringBuffer, it doesn't guarantee thread safety.

StringBuilder Constructors

Constructor	Description
StringBuilder()	Creates an empty string builder with a default capacity of 16 (16 empty elements).
StringBuilder(CharSequence cs)	Constructs a string builder containing the same characters as the specified CharSequence, plus an extra 16 empty elements trailing the CharSequence.
StringBuilder(int initCapacity)	Creates an empty string builder with the specified initial capacity.
StringBuilder(String s)	Creates a string builder whose value is initialized by the specified string, plus an extra 16 empty elements trailing the string.

StringBuilder Methods

- StringBuilder Length and Capacity

```
// creates empty builder, capacity 16

StringBuilder sb = new StringBuilder();

// adds 5 character string at beginning
sb.append("Hello");

System.out.println("StringBuilder length = "+sb.length());
// prints 5

System.out.println("StringBuilder capacity =
"+sb.capacity()); // prints 16
```

Append()

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

Insert()

```
StringBuilder sb = new
StringBuilder("HellWorld");

sb.insert(4, "o ");

System.out.println(sb); // prints Hello World
```

replace(int startIndex, int endIndex, String str)

```
StringBuilder sb = new StringBuilder("Hello
World!");

sb.replace(6,11,"Earth");

System.out.println(sb); // prints Hello Earth!
```

delete(int startIndex, int endIndex)

```
StringBuilder sb = new
StringBuilder("JournalDev.com");

sb.delete(7,14);

System.out.println(sb); // prints Journal
```

Capacity()

```
StringBuilder sb=new StringBuilder();

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

sb.append("Java");

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

sb.append("Hello StringBuilder Class!");

System.out.println(sb.capacity()); // (16*2)+2
```

Reverse()

```
StringBuilder sb = new StringBuilder("lived");

sb.reverse();

System.out.println(sb);// prints devil
```

StringBuffer v/s StringBuilder

StringBuffer	StringBuilder
StringBuffer is *synchronized* i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously.	StringBuilder is *non-synchronized* i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously.
Operates slower due to thread safety feature	Better performance compared to StringBuffer
Has some extra methods – substring, length, capacity etc.	Not needed because these methods are present in String too.
Introduced in Java 1.2	Introduced in Java 1.5 for better performance.
StringBuffer is *less efficient* than StringBuilder.	StringBuilder is *more efficient* than StringBuffer.

That’s all for now…