
# Java

# Классы

# StringBuffer и StringBuilder

Лекция #4 B

Пустовалова О.Г.
доцент. каф. мат.мод.
ИММиКН ЮФУ

# Содержание

# StringBuffer и StringBuilder

- Объекты String являются **неизменяемыми**, поэтому все операции, которые изменяют строки, фактически приводят к созданию новой строки, что сказывается на **производительности приложения**.

- Для решения этой проблемы, чтобы работа со строками проходила с меньшими издержками в Java были добавлены классы **StringBuffer** и **StringBuilder**.

- По сути они напоминает расширяемую строку, которую можно изменять без ущерба для производительности.

# Отличие StringBuffer от StringBuilder

- Эти классы похожи, практически двойники, они имеют одинаковые конструкторы, одни и те же методы, которые одинаково используются.

- Единственное их различие состоит в том, что класс **StringBuffer** синхронизированный и **потокобезопасный**. То есть класс StringBuffer удобнее использовать в многопоточных приложениях, где объект данного класса может меняться в различных потоках.

- Если же речь о многопоточных приложениях не идет, то лучше использовать класс **StringBuilder**, который не потокобезопасный, но при этом работает **быстрее, чем StringBuffer** в однопоточных приложениях.

# Скорость обработки String, StringBuffer, StringBuilder

| | | |
|---|---|---|
| 1. StringBuilder | изменяемый | Один поток |
| 2. StringBuffer | изменяемый | Много потоков |
| 3. String | неизменяемый | Один поток |

# Четыре конструктора StringBuffer

```
StringBuffer()
StringBuffer(int capacity)
StringBuffer(String str)
StringBuffer(CharSequence chars)
```

С помощью метода **capacity()** можно получить количество символов, для которых зарезервирована память.

```
StringBuilder sb1=new StringBuilder();
StringBuilder sb2=new StringBuilder(20);
StringBuilder sb3=new StringBuilder("ABC");

System.out.println(sb1.capacity());
System.out.println(sb2.capacity());
System.out.println(sb3.capacity());
```

```
16
20
19
```

# Четыре конструктора StringBuffer

```
StringBuilder()
StringBuilder(int capacity)
StringBuilder(String str)
StringBuilder(CharSequence chars)
```

```
StringBuffer sb1=new StringBuffer();
StringBuffer sb2=new StringBuffer(20);
StringBuffer sb3=new StringBuffer("ABC");

System.out.println(sb1.length());
System.out.println(sb2.length());
System.out.println(sb3.length());
```

```
0
0
3
```

https://www.tutorialspoint.com/java/lang/java_lang_stringbuffer.htm

# Четыре конструктора StringBuilder

```
StringBuilder sb = new StringBuilder("ABCD");
System.out.println(sb.length());
System.out.println(sb.capacity());
```

```
4
20
```

```
sb.ensureCapacity(32);
System.out.println(sb.capacity());
```

```
42
```

Финальная емкость может отличаться в большую сторону.
В целях повышения эффективности Java может дополнительно выделять память.

Длина строки, которую можно получить с помощью метода length(), в StringBuffer остается прежней - 4 символа.

# StringBuffer and StringBuilder Class methods

# StringBuffer and StringBuilder class methods

- **StringBuilder** имеет похожие методы с **StringBuffer**.
- Используйте **StringBuffer** в **многопоточном** приложении.
- Используйте **StringBuilder** в **однопоточном** приложении.
- Производительность **StringBuilder** выше, чем **StringBuffer** и **String**.

https://www.tutorialspoint.com/java/lang/java_lang_stringbuffer.htm
https://www.tutorialspoint.com/java/lang/java_lang_stringbuilder.htm

# StringBuffer class methods

1 StringBuffer append(**boolean** b)
*// This method appends the string representation of the boolean argument to the sequence*
2 StringBuffer append(**char** c)
*// This method appends the string representation of the char argument to this sequence.*
3 StringBuffer append(**char**[] str)
*// This method appends the string representation of the char array argument to this sequence.*
4 StringBuffer append(**char**[] str, **int** offset, **int** len)
*// This method appends the string representation of a subarray of the char array argument to this sequence.*
5 StringBuffer append(CharSequence s)
*// This method appends the specified CharSequence to this sequence.*
6 StringBuffer append(CharSequence s, **int** start, **int** end)
*// This method appends a subsequence of the specified CharSequence to this sequence.*

https://www.tutorialspoint.com/java/lang/java_lang_stringbuffer.htm

# StringBuffer class methods

```
7   StringBuffer append(double d)
// This method appends the string representation of the double
argument to this sequence.
8  StringBuffer append(float f)
// This method appends the string representation of the float argument
to this sequence.
9  StringBuffer append(int i)
// This method appends the string representation of the int argument to
this sequence.
10 StringBuffer append(long lng)
// This method appends the string representation of the long argument
to this sequence.
11 StringBuffer append(Object obj)
// This method appends the string representation of the Object
argument.
12 StringBuffer append(String str)
// This method appends the specified string to this character sequence.
13 StringBuffer append(StringBuffer sb)
// This method appends the specified StringBuffer to this sequence.
```

# StringBuffer class methods

14 StringBuffer appendCodePoint(**int** codePoint)

*// This method appends the string representation of the codePoint argument to this sequence.*

15 **int** capacity()

*// This method returns the current capacity.*

16 **char** charAt(**int** index)

*// This method returns the char value in this sequence at the specified index.*

17 **int** codePointAt(**int** index)

*// This method returns the character (Unicode code point) at the specified index*

18 **int** codePointBefore(**int** index)

*// This method returns the character (Unicode code point) before the specified index*

19 **int** codePointCount(**int** beginIndex, **int** endIndex)

*// This method returns the number of Unicode code points in the specified text range of this sequence*

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

*// This method removes the characters in a substring of this sequence.*

# StringBuffer class methods

21 StringBuffer deleteCharAt(**int** index)
*// This method removes the char at the specified position in this sequence*
22 **void** ensureCapacity(**int** minimumCapacity)
*// This method ensures that the capacity is at least equal to the specified minimum.*
23 **void** getChars(**int** srcBegin, **int** srcEnd, **char**[] dst, **int** dstBegin)
*// This method characters are copied from this sequence into the destination character array dst.*
24 **int** indexOf(String str)
*// This method returns the index within this string of the first occurrence of the specified substring.*
25 **int** indexOf(String str, **int** fromIndex)
*// This method returns the index within this string of the first occurrence of the specified substring, starting at the specified index.*
26 StringBuffer insert(**int** offset, **boolean** b)
*// This method inserts the string representation of the boolean argument into this sequence.*

# StringBuffer class methods

27 StringBuffer insert(**int** offset, **char** c)
*// This method inserts the string representation of the char argument into this sequence.*
28 StringBuffer insert(**int** offset, **char**[] str)
*// This method inserts the string representation of the char array argument into this sequence.*
29 StringBuffer insert(**int** index, **char**[] str, **int** offset, **int** len)
*// This method inserts the string representation of a subarray of the str array argument into this sequence.*
30 StringBuffer insert(**int** dstOffset, CharSequence s)
*// This method inserts the specified CharSequence into this sequence.*
31 StringBuffer insert(**int** dstOffset, CharSequence s, **int** start, **int** end)
*// This method inserts a subsequence of the specified CharSequence into this sequence.*
32 StringBuffer insert(**int** offset, **double** d)
*// This method inserts the string representation of the double argument into this sequence.*
33 StringBuffer insert(**int** offset, **float** f)
*// This method inserts the string representation of the float argument into this sequence.*

# StringBuffer class methods

```
34  StringBuffer insert(int offset, int i
// This method inserts the string representation of the second int
argument into this sequence.
35 StringBuffer insert(int offset, long l)
// This method inserts the string representation of the long argument
into this sequence.
36 StringBuffer insert(int offset, Object obj)
// This method inserts the string representation of the Object argument
into this character sequence.
37 StringBuffer insert(int offset, String str)
// This method inserts the string into this character sequence.
38 int lastIndexOf(String str)
// This method returns the index within this string of the rightmost
occurrence of the specified substring.
39 int lastIndexOf(String str, int fromIndex)
// This method returns the index within this string of the last occurrence
of the specified substring.
40 int length()
// This method returns the length (character count).
```

# StringBuffer class methods

41 **int** offsetByCodePoints(**int** index, **int** codePointOffset)

*// This method returns the index within this sequence that is offset from the given index by codePointOffset code points.*

42 StringBuffer replace(**int** start, **int** end, String str)

*// This method replaces the characters in a substring of this sequence with characters in the specified String.*

43 StringBuffer reverse()

*// This method causes this character sequence to be replaced by the reverse of the sequence.*

44 **void** setCharAt(**int** index, **char** ch)

*// The character at the specified index is set to ch.*

45 **void** setLength(**int** newLength)

*// This method sets the length of the character sequence.*

46 CharSequence subSequence(**int** start, **int** end)

*// This method returns a new character sequence that is a subsequence of this sequence.*

47 String substring(**int** start)

*// This method returns a new String that contains a subsequence of characters currently contained in this character sequence*

# StringBuffer class methods

48 String substring(**int** start, **int** end)

*// This method returns a new String that contains a subsequence of characters currently contained in this sequence.*

49 String toString()

*// This method returns a string representing the data in this sequence.*

50 **void** trimToSize()

*// This method attempts to reduce storage used for the character sequence.*

# Примеры использования

# StringBuffer  и StringBuilder

## append

## class method

# java.lang.StringBuffer.append() method

```
StringBuilder sb = new StringBuilder(10);
```


```
sb.append("Hello   !");
```


```
sb.append('!');
```

# java.lang.StringBuffer.append() method

```
StringBuilder sb = new StringBuilder("2+2=4 - ");

sb.append(true);

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

**2+2=4 – true**

# java.lang.StringBuffer.append() method

```
StringBuilder str = new StringBuilder("compile ");
System.out.println("string = " + str);

// char array
char[] cArr = new char[]{'o','n','l','i','n','e'};

/* appends the string representation of char array argument
to this StringBuilder */
str.append(cArr);

// print the StringBuilder after appending
System.out.println("After append = " + str);
```

**string = compile**
**After append = compile online**

# java.lang.StringBuffer.append() method

```
StringBuilder str = new StringBuilder("I am ");
System.out.println("string = " + str);

// char array
char[] cArr = new char[]
        {'a','d','m','i','n','i','s','t','r','a','t','o','r'};

/* appends the string representation of char array argument to
   this StringBuilder with offset at index 0 and length as 5 */
str.append(cArr, 0, 5);

// print the StringBuilder after appending
System.out.println("After append = " + str);
```

```
string = I am
After append = I am admin
```

# java.lang.StringBuffer.append() method

```
StringBuilder str = new StringBuilder("I am ");
System.out.println("string = " + str);

CharSequence cSeq = "admin";

// appends the CharSequence
str.append(cSeq);

// print the StringBuilder after appending
System.out.println("After append = " + str);
```

**string = I am**
**After append = I am admin**

# java.lang.StringBuffer.append() method

```
StringBuilder str = new StringBuilder(" 2+2 = ");

str.append(4);
```

**2+2 = 4**

```
StringBuilder str = new StringBuilder(" Pi = ");

str.append(Math.PI);
```

**Pi = 3.141592653589793**

## Примеры использования

## StringBuffer  и StringBuilder

**capacity, charAt, codePointAt, codePointBefore, codePointCount**

**class methods**

# java.lang.StringBuffer. capacity() method

- Пустой класс StringBuffer содержит емкость по умолчанию 16 символов.
- Метод capacity() класса Java StringBuffer возвращает текущую емкость строкового буфера.
- Если число символов увеличивается, то емкость увеличивается на емкость нагрузки*2+2.

```
StringBuilder sb1=new StringBuilder();
System.out.println(sb1.capacity());
```

→ **16**

```
sb1=sb1.append("1234567890ABCDEF*");
System.out.println(sb1.capacity());
```

→ **34** (16*2+2)

# java.lang.StringBuffer. charAt, codePointAt, codePointBefore

```
StringBuilder sb1=new StringBuilder("12345");
System.out.println(sb1.charAt(4));
```

→ **5**

```
StringBuilder sb1=new StringBuilder("12345");
System.out.println(sb1.codePointAt(4));
```

→ **53**

```
StringBuilder sb1=new StringBuilder("12345");
System.out.println(sb1. codePointBefore(4));
```

→ **52**

# java.lang.StringBuffer. codePointCount

Метод **codePointCount** класса StringBuffer возвращает количество кодовых точек Unicode в указанном диапазоне.

```
StringBuilder sb1=new StringBuilder("012345");

System.out.println(sb1.codePointCount(0,1));
```

→ **1**

```
System.out.println(sb1.codePointCount(0,3));
```

→ **3**

# Примеры использования

# StringBuffer  и StringBuilder

## delete, deleteCharAt, ensureCapacity

## class methods

# java.lang.StringBuffer. delete

Метод **delete** класса StringBuffer используется для удаления символов в строке.

```
StringBuffer sbf = new StringBuffer("123456789");

sbf.delete(1, 5);
```

**16789**

# java.lang.StringBuffer. deleteCharAt

Метод **deleteCharAt** удаляет символ в указанной позиции.

```
StringBuffer sbf = new StringBuffer("ABC");

sbf. deleteCharAt(1);
```

**AC**

# java.lang.StringBuffer. ensureCapacity

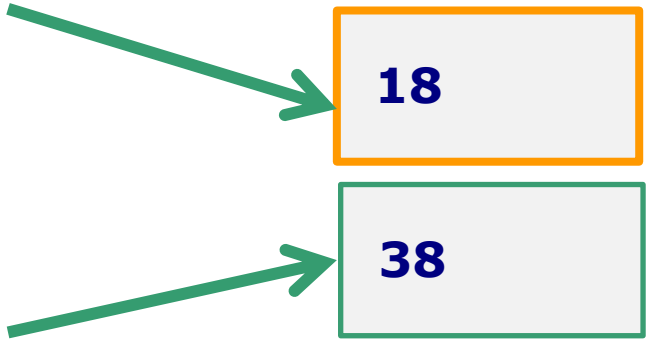
Метод **ensureCapacity** гарантирует, что емкость по крайней мере равна указанному минимуму.

```
StringBuffer buff = new StringBuffer("12");

System.out.println("Old Capacity = " + buff.capacity());

buff.ensureCapacity(30);

System.out.println("New Capacity = " + buff.capacity());
```



# Примеры использования

# StringBuffer  и StringBuilder

## getChars, indexOf, lastIndexOf, length

## class methods

# java.lang.StringBuffer. getChars

Метод getChars (int srcBegin, int srcEnd, char[] dst, int dstBegin) класса StringBuffer копирует символы в массив.

```
StringBuffer str = new StringBuffer("ABCDE");

char[] array = {'0','1','2','3','4','5','6'};

str.getChars(0, 2, array, 3);

for (int i = 0; i < array.length; i++)
    System.out.print(array[i] + " ");
```

**0 1 2 A B 5 6**

# java.lang.StringBuffer. indexOf

Метод **indexOf** возвращает индекс в этой строке первого вхождения указанной подстроки.

Метод **lastIndexOf** возвращает индекс в этой строке последнего вхождения указанной подстроки.

```
StringBuffer buff = new StringBuffer("ABCD QWERTY");

System.out.println(buff.indexOf("CD"));
```

**2**

```
// returns -1 as substring is not found
System.out.println(buff.indexOf("123"));
```

**-1**

# Примеры использования

# StringBuffer  и StringBuilder

# insert

# class methods

# java.lang.StringBuffer. insert

Метод **insert** вставляет символы в данную строку буфера (StringBuffer). Подстрока добавляется перед указанным индексом.

```
StringBuffer sb = new StringBuffer("abcd");

sb.insert(3,"123");
```

**abc123d**

# Примеры использования

# StringBuffer  и StringBuilder

# offsetByCodePoints, replace, reverse

# class methods

# java.lang.StringBuffer. offsetByCodePoints

Метод **OffsetByCodePoints** (int index,int codePointOffset) возвращает индекс в этой последовательности, который смещен от заданного индекса кодовыми точками codePointOffset.

```
// initialize the StringBuffer object
StringBuffer sb = new StringBuffer("javatutorialhq.com");
System.out.println("Contents of buffer:" + sb);

// get the offsetByCodePoints on index of 2 and an offset of 5
int index = 2;
int codePointOffset = 5;

System.out.println("Result :"
    + sb.offsetByCodePoints(index, codePointOffset));
```

→ **7**

# java.lang.StringBuffer. replace

Метод **replace** заменяет указанной подстрокой символы с начала указанного индекса до конца указанного индекса.

```
StringBuffer sb = new StringBuffer("A**B**C");

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

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

→ **A===B**C**

# java.lang.StringBuffer. reverse

```
StringBuffer sb = new StringBuffer("ABCD");

System.out.println(sb.reverse());
```

**DCBA**

## Примеры использования

## StringBuffer и StringBuilder

**setCharAt, setLength, subSequence, substring, toString, trimToSize**

**class methods**

# java.lang.StringBuffer. setCharAt

Метод setCharAt () устанавливает символ в позицию с указанным индексом.

```
StringBuffer buff = new StringBuffer("A*C");

buff.setCharAt(1, 'B');
```

**ABC**

# java.lang.StringBuffer. setLength

Метод setLength устанавливает новую длину. Если длина меньше, чем количество символов, то содержимое обрезается до заданного значения.

```
StringBuffer buff = new StringBuffer("12345678");
buff.setLength(3);
System.out.println(buff);
```

```
123
```

# java.lang.StringBuffer. subSequence

Метод subSequence() возвращает новую последовательность символов, которая является подпоследовательностью этой последовательности

```
StringBuffer buff = new StringBuffer("admininstrator");


CharSequence cSeq;

// returns the specified subSequence
cSeq = buff.subSequence(0,5);

// print the subsequence
System.out.println(cSeq);
```

**admin**

# java.lang.StringBuffer. substring

Метод **substring** (int start) возвращает новую строку, содержащую подпоследовательность символов, содержащихся в данной последовательности символов. Подстрока начинается с указанного индекса и продолжается до конца этой последовательности.

```
StringBuffer buff = new StringBuffer("***ABC");

System.out.println(buff.substring(3));
```

**ABC**

# java.lang.StringBuffer. toString

Метод **toString**() класса StringBuffer-это встроенный метод, используемый для возврата строки, представляющей данные, содержащиеся в объекте StringBuffer.

```
StringBuffer buff = new StringBuffer("ABC");
System.out.println(buff.toString());
```

# java.lang.StringBuffer. trimToSize

Метод trimToSize () класса StringBuffer-это встроенный метод, используемый для **ограничения емкости**, используемой для символьной последовательности объекта StringBuffer. Если буфер, используемый объектом StringBuffer, больше, чем необходимо для хранения его текущей последовательности символов, то этот метод вызывается для изменения размера объекта StringBuffer для преобразования этого объекта в более эффективное пространство. Вызов этого метода **может**, но не обязательно, **повлиять** на значение, возвращаемое последующим вызовом метода **capacity** ().

```
StringBuffer str = new StringBuffer("ABC");
System.out.println(str.capacity());
str.trimToSize();
System.out.println(str.capacity());
```

**19**

**3**

# Спасибо за внимание!