

# **String: mutable, immutable, StringBuilder, StringBuffer, StringTokenizer**

---

- ? 1. Что такое mutable/immutable объекты и зачем они
- ? 2. Все классы, связанные со строками, что они делают и все их методы
- ? 3. Внутреннее устройство String, метод substring
- ? 4. Поиск, получение, удаление подстроки в String
- ? 5. Что можно делать с помощью класса StringBuilder, StringBuffer
- ? 6. Регулярные выражения, примеры
- ? 7. StringTokenizer, String.replace, String.split
- ? 8. Задачи



# 1. Что такое mutable/imutable объекты и зачем они

---

- ? Объекты, которые после создания можно изменить, называются изменяемыми или **mutable**. Если объекты после создания изменить нельзя, называются неизменяемыми или **immutable**.
- ? Два свойства, которые характерны практически для всех immutable объектов:
- ? 1) **Неизменяемые объекты можно реализовать значительно проще, чем изменяемые.**
- ? 2) **Неизменяемые объекты можно свободно использовать одновременно из разных нитей.**



- 
- ? - А если мне нужно что-то поменять в таком объекте? Что вообще можно сделать с неизменяемым объектом?
  - ? - Обычно `immutable` классы содержат различные методы, которые «как-бы» меняют объект, но вместо изменения самого объекта эти методы просто создают новый объект и возвращают его.
  - ? **String s = "moscow";  
String s2 = s.toUpperCase();**
  - ? Класс `String` – это `immutable` класс . `s` содержит строку «moscow», а `s2` – «MOSCOW» т.е **содержит новую строку**, которая идентична первой
- 
- 

## 2. Все классы, связанные со строками, что они делают и все их методы

---

- ? String отвечает за неизменяемую строку(immutable)
- ? **StringBuilder** – за изменяемую(mutable)
- ? StringBuffer – это копия StringBilder, но все методы которого объявлены **synchronized**
- ? Разве тут строка не изменяется?

Пример кода	Что происходит на самом деле
<pre>String s = "cat"; s = s + "-" + s;</pre>	<pre>String s = "cat"; StringBuilder s2 = new StringBuilder(s); s2.append("-"); s2.append(s); s = s2.toString();</pre>



Метод(ы)	Пример(ы)
<p><b>int length();</b></p> <p>Метод <b>length</b> возвращает длину строки – количество в ней символов.</p>	<pre>String s = "Good news everyone!"; int n = s.length(); int n = "Good news everyone!".length();</pre>
<p><b>char charAt(int index)</b></p> <p>Метод <b>charAt</b> возвращает символ строки по его номеру. Нумерация символов начинается с 0.</p>	<pre>String s = "Good news everyone!"; char n = s.charAt(5); char n = "Good news everyone!".charAt(5);</pre>
<p><b>char[]toCharArray ()</b></p> <p>Метод <b>toCharArray</b> возвращает массив всех символов строки.</p>	<pre>String s = "Good news everyone!"; for(char c: s.toCharArray()) {     System.out.println(c); }</pre>



Метод(ы)	Пример(ы)
<p><b>boolean equals (Object o)</b></p> <p>Метод equals проверяет – совпадают ли строки</p>	<pre>String s = "cat"; boolean test1 = s.equals("cat");//true boolean test2 = s.equals("Cat");//false boolean test3 = s.equals("c"+"a"+"t");</pre>
<p><b>boolean equalsIgnoreCase (String str)</b></p> <p>метод equalsIgnoreCase – совпадают ли строки, игнорируя регистр букв</p>	<pre>String s = "cat"; boolean test1 = s.equalsIgnoreCase("cat"); boolean test2 = s.equalsIgnoreCase("Cat"); boolean test3 = s.equalsIgnoreCase("cAT");</pre>



Метод(ы)	Пример(ы)
<b>String toUpperCase()</b> Метод <b>toUpperCase</b> возвращает копию строки, все символы которой – большие.	<pre>String s = "Good news everyone!"; s = s.toUpperCase();</pre> <b>Результат:</b> <pre>s == "GOOD NEWS EVERYONE!";</pre>
<b>String toLowerCase()</b> Метод <b>toLowerCase</b> возвращает копию строки, все символы которой – маленькие.	<pre>String s = "Good news everyone!"; s = s.toLowerCase();</pre> <b>Результат:</b> <pre>s == "good news everyone!";</pre>
<b>String trim()</b> Метод <b>trim</b> возвращает копию строки, без «пустых» символов в начале и конце.	<pre>String s = "Good news everyone!"; s = s.trim();</pre>



### 3. Внутреннее устройство String, метод substring

---

Метод(ы)	Пример(ы)
<b>String substring(int beginIndex, int endIndex)</b>	<pre>String s = "Good news everyone!"; s = s.substring(1,6);</pre>
	<b>Результат:</b> <pre>s == "ood n";</pre>
<b>String substring(int beginIndex)</b>	<pre>String s = "Good news everyone!"; s = s.substring(1);</pre>

- ? Метод **substring** возвращает часть строки
  - ? Первый вариант возвращает подстроку, заданную начальным и конечным номерами символов. **Последний символ при этом не входит!**
  - ? Второй вариант – от переданного номера и до конца строки.
- 



Получение подстроки	Что хранится внутри подстроки
<code>String s = "мама";</code>	Что хранится в s: <code>char[] value = {'m','a','m','a'};</code> <code>offset = 0;</code> <code>count = 4;</code>
<code>String s2 = s.substring(1);</code>	Что хранится в s2: <code>char[] value = {'a','m','a'};</code> <code>offset = 1;</code> <code>count = 3;</code>
<code>String s3 = s.substring(1, 3);</code>	Что хранится в s3: <code>char[] value = {'a','m'};</code> <code>offset = 1;</code> <code>count = 2;</code>



## 4. Поиск, получение, удаление подстроки в String

Метод(ы)	Пример(ы)
<code>int indexOf(String str)</code>	<pre>String s = "Good news everyone!"; int index = s.indexOf("ne");</pre>
<code>indexOf</code> ищет в строке указанную строку с начала строки. Если найдена – возвращает номер ее первого символа, если нет - возвращает -1	<b>Результат:</b> <code>index == 5</code>
<code>int indexOf(String str, int from)</code>	<pre>String s = "Good news everyone!"; int index = s.indexOf("ne",7);</pre>
<code>indexOf</code> ищет в строке указанную строку начиная с какого-то номера. Если строка найдена – возвращает номер ее первого символа, если нет - возвращает -1	<b>Результат:</b> <code>index == 16</code>



Метод(ы)	Пример(ы)
<b>int lastIndexOf(String str)</b>	<pre>String s = "Good news everyone!"; int index = s.lastIndexOf("ne");</pre>
<b>lastIndexOf</b> ищет указанную строку в строке с самого конца строки. Если строка найдена –возвращает номер ее первого символа, если не найдена - возвращает -1.	<b>Результат:</b> index == 16
<b>int lastIndexOf(String str,int from)</b>	<pre>String s = "Good news everyone!"; int index = s.lastIndexOf("ne",15);</pre>
<b>lastIndexOf</b> ищет указанную строку в строке с конца, начиная с какого-то номера . Если строка найдена – возвращает номер ее первого символа, если не найдена - возвращает -1.	<b>Результат:</b> ?



Метод(ы)	Пример(ы)
<code>String replace(char oldChar, char newChar)</code>	<pre>String s = "Good news everyone!"; String s2 = s.replace('o', 'a');</pre>
Метод <b>replace</b> заменяет все вхождения определенного символа на другой.	<b>Результат:</b> <pre>s2 == "Gaad news everyane!";</pre>
<code>String replaceAll(String regex, String replacement)</code>	<pre>String s = "Good news everyone!"; String s2 = s.replaceAll("ne", "_");</pre>
Метод <b>replaceAll</b> заменяет все вхождения одной подстроки на другую.	<b>Результат:</b> <pre>s2 == "Good _ws everyo_!";</pre>
<code>String replaceFirst(String regex, String replacement)</code>	<pre>String s = "Good news everyone!"; String s2 = s.replaceFirst("ne", "_");</pre>
Метод <b>replaceFirst</b> заменяет первое вхождение переданной подстроки на заданную подстроку.	<b>Результат:</b> <pre>s2 == "Good _ws everyone!";</pre>



## 5. Что можно делать с помощью класса **StringBuilder**, **StringBuffer**

---

- ? **1) У меня есть обычная строка, я хочу сделать ее изменяемой.**

```
String s = "Bender";  
StringBuilder s2 = new StringBuilder(s);
```

- ? **2) Я хочу добавить символ к текущей «изменяемой строке»?**

```
String s = "Bender";  
StringBuilder s2 = new StringBuilder(s);  
s2.append("!");
```

- ? **3) А как преобразовать **StringBuilder** обратно в строку?**

```
String s = "Bender";  
StringBuilder s2 = new StringBuilder(s);  
s2.append("!");  
s = s2.toString();
```

---

---

? **4) А если мне нужно удалить символ?**

```
String s = "Bender";  
StringBuilder s2 = new StringBuilder(s);  
s2.deleteCharAt(2); //останется "Beder"
```

? **5) Я хочу заменить часть строки на другую?**

```
String s = "Bender";  
StringBuilder s2 = new StringBuilder(s);  
s2.replace (3, 5, "_DE_"); //будет "Ben_DE_r"
```

? **6) Мне нужно развернуть строку задом наперед?**

```
String s = "Bender";  
StringBuilder s2 = new StringBuilder(s);  
s2.reverse(); //будет "redneB";
```

---



## 6. Регулярные выражения

Шаблон	Описание	Примеры
.	Один любой символ	1
\d	Любая цифра	7
\D	Любая нецифра	C
\s	Пробел, перенос строки, символ табуляции	''
\S	Что угодно, кроме пробела, табуляции, переноса строки	f
[a-z]	Любая буква от a до z	z
[0-9]	Любая цифра от 0 до 9.	8
\w	Любая буква	c
\W	Любая небуква	_

Шаблон	Описание	Примеры
[a-d]?	Символы a-d встречаются 0..1 раз	a, b, c, d
[b-d,z]+	Символы b,c,d,z встречаются 1.. ∞ раз	b, bcdcdbdbdbdbzzzzbbzbz b, zbz
[1,7-9]*	Символы 1,7,8,9 встречаются 0.. ∞ раз	1, 7, 9, 9777, 111199
1{5}	Символ 1 встречается 5 раз	11111
[1,2,a,b]{2}	Символы 1,2,a,b встречаются 2 раза	11, 12, 1a, ab, 2b, bb, 22
[a,0]{2,3}	Символы a,0 встречаются 2..3 раз	aa, a0,00,0a, aaa,000, a00,0a0, a0a



Шаблон	Описание	Примеры
$A?$	Символ $A$ встречается 0..1 раз	$A$
$B^+$	Символ $B$ встречается 1.. $\infty$ раз	$BBBB$
$C^*$	Символ $C$ встречается 0.. $\infty$ раз	$CCC$
$D\{n\}$	Символ $D$ встречается $n$ раз	$DDDD$ , для шаблона $D\{4\}$
$E\{n,\}$	Символ $E$ встречается $n.. \infty$ раз	$EEEEEEEE$ , для шаблона $E\{2,\}$
$F\{n,m\}$	Символ $F$ встречается $n..m$ раз	$EEEE$ , для шаблона $E\{2,4\}$



# Регулярные выражения, примеры

---

- ? «**^**» - означает, что подстрока обязана включать начало строки.
- ? «**\$**» - означает, что подстрока обязана включать конец строки.
- ? В регулярных выражениях символы «**[ ] \ / ^ \$ . | ? \* + ( ) { }**» имеют специальное значение. Их еще называют «управляющие символы». Поэтому просто так их использовать в строке нельзя.
- ? Их необходимо экранировать. Для этого, используется символ «**\**».



Шаблон	Строка и найденные подстроки, совпадающие с шаблоном
<code>a{3}</code>	<b>aaa a aaa a aaa</b>
<code>a{3}\$</code>	aaa a aaa a <b>aaa</b>
<code>^a{3}</code>	<b>aaa</b> a aaa a aaa
<code>^a{3}\$</code>	aaa a aaa a aaa
<code>\?{3}</code>	aaa <b>???</b> aaa a



## 7. String.replace, String.split

Метод(ы)	Пример(ы)
boolean <b>matches</b> (String regex)	<pre>String s = "Good news everyone!"; Boolean test = s.matches("news\\.*");</pre>
Позволяет проверить, совпадает ли строка с шаблоном, заданным регулярным выражением	<b>Результат:</b> false (строка не начинается со слова news)
String[ ] <b>split</b> (String regex)	<pre>String s = "Good news everyone!"; String[ ] ss = s.split("ne"); System.out.println(Arrays.toString(ss) );</pre>
Разбивает строку на части, принимает маску подстроки-разделителя	<b>Результат (будет массив из трех строк):</b> [Good , ws everyo, !] "Good ", "ws everyo", "!";

---

Метод(ы)	Пример(ы)
<code>String <b>replaceAll</b>(String regex, String replacement)</code>	<pre>String s = "Good news everyone!"; String s2 = s.<b>replaceAll</b> ("e\\.","EX");</pre>
Заменяет все вхождения одной подстроки на другую	<b>Результат:</b> <pre>s2 == "Good nEXs EXEXyonEX";</pre>
<code>String <b>replaceFirst</b>(String regex, String replacement)</code>	<pre>String s = "Good news everyone!"; String s2 = s.<b>replaceFirst</b>("e\\.","EX");</pre>
Заменяет первое вхождение переданной подстроки на заданную подстроку	<b>Результат:</b> <pre>s2 == "Good nEXs everyone!";</pre>

---



## 7.StringTokenizer(еще один способ разбиения строки на части )

---

- ? Этот класс не использует регулярные выражения, вместо этого в него просто передается строка, состоящая из символов-разделителей. Преимущества этого подхода в том, что он не разбивает сразу всю строку на кусочки, а идет от начала к концу.
- ? Класс состоит из конструктора и двух методов. В конструктор нужно передать строку, которую мы разбиваем на части, и строку – набор символов, используемых для деления.
- ? Метод `nextToken` возвращает очередной токен – подстроку.
- ? Метод `hasMoreTokens()` возвращает `true`, если еще остались не отданные подстроки.



Метод(ы)	Пример(ы)
<b>boolean hasMoreTokens()</b>  <b>String nextToken()</b>	<pre>String s = "Good news everyone!";  StringTokenizer tokenizer =     new StringTokenizer(s,"ne"); while (tokenizer.hasMoreTokens()) {     String token = tokenizer.nextToken();     System.out.println(token); }</pre> <p><b>Вывод на экран будет таким:</b> Good ws v ry !</p>

## Задача 1 (com.simbirsoft.lesson00.task00)

---

### ? **Найти подстроку**

Метод `getPartOfString()` должен возвращать подстроку начиная с символа после 1-го пробела и до конца слова, которое следует после 4-го пробела.

Пример: "Курсы Simbirsoft - лучший способ прокачать уровень знаний Java."

Результат: "Simbirsoft - лучший способ "

На некорректные данные бросить исключение `TooShortStringException` (сделать исключением).



## Задача 2

---

### ? Между табуляциями

Метод `getPartOfString()` должен возвращать подстроку между первой и второй табуляцией.

На некорректные данные бросить исключение `TooShortStringException`.



# Задача 3

---

## **StringTokenizer**

Используя StringTokenizer разделить query на части по разделителю delimiter.

Пример:

```
getTokens("level22.lesson | 3.task0 |", ".") == {"level22",  
"lesson | 3", "task0 |"}
```



## ASCII (American Standard Code for Information Interchange)

Dec	Hex	Char									
0	0	NUL	32	20	(sp)	64	40	@	96	60	`
1	1	SOH	33	21	!	65	41	A	97	61	a
2	2	STX	34	22	"	66	42	B	98	62	b
3	3	ETX	35	23	#	67	43	C	99	63	c
4	4	EOT	36	24	\$	68	44	D	100	64	d
5	5	ENQ	37	25	%	69	45	E	101	65	e
6	6	ACK	38	26	&	70	46	F	102	66	f
7	7	BEL	39	27	'	71	47	G	103	67	g
8	8	BS	40	28	(	72	48	H	104	68	h
9	9	TAB	41	29	)	73	49	I	105	69	i
10	A	LF	42	2A	*	74	4A	J	106	6A	j
11	B	VT	43	2B	+	75	4B	K	107	6B	k
12	C	FF	44	2C	,	76	4C	L	108	6C	l
13	D	CR	45	2D	-	77	4D	M	109	6D	m
14	E	SO	46	2E	.	78	4E	N	110	6E	n
15	F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

```
package com.javarush.test.level18.lesson10.bonus0;
```

---

## Шифровка

Задача: Придумать механизм шифровки/дешифровки

Программа запускается с одним из следующих наборов параметров:

-e fileName outputFileName

-d fileName outputFileName

где

fileName - имя файла, который необходимо зашифровать/расшифровать

outputFileName - имя файла, куда необходимо записать результат шифрования/дешифрования

-e - ключ указывает, что необходимо зашифровать данные

-d - ключ указывает, что необходимо расшифровать данные

---



---

---

