

**String**

# Intro

# Problem

- Большая часть информации, которая используется в программном обеспечении является текстовой.
- Как работать с текстом в Java?

# **Solution**

- **String**

# Class **String**

# Class **String**

- **String** из пакета `java.lang`
- **String** **неизменяемый (immutable)**, т.е. после создания, его содержимое не может изменяться.

```
String str = "Hello Java!";
```

# How create **String** objects?

```
String str1 = "Java";  
String str2 = new String();  
String str3 = new String("JVM");  
String str4 = new String(new char[] {'h', 'e', 'l', 'l', 'o'});  
String str5 = new String(new char[] {'w', 'e', 'l', 'c', 'o', 'm', 'e'}, 3,  
  
System.out.println(str1); // 'Java'  
System.out.println(str2); //  
System.out.println(str3); // 'JVM'  
System.out.println(str4); // 'hello'  
System.out.println(str5); // 'come'
```

## Operator **+** for strings

Для *объектов* и *литералов* типа **String** определена одна операция **+** (**конкатенация**), которая объединяет две строки.

```
String str1 = "Java";  
String str2 = "Hello";  
String str3 = str1 + " " + str2;  
  
System.out.println(str3); // Hello Java
```



# **String:** main methods

## length(): int

```
String str1 = "Java";  
System.out.println(str1.length()); // 4
```

## toCharArray(): char[]

```
String str1 = new String(new char[] {'h', 'e', 'l', 'l', 'o'});  
char[] helloArray = str1.toCharArray();
```

```
String s = ""; // строка не указывает на объект  
if (s.length() == 0) {  
    System.out.println("String is empty");  
}
```

## **isEmpty(): boolean**

```
public boolean isEmpty() {  
    return value.length == 0;  
}
```

## isEmpty(): boolean

```
String s = null; // строка не указывает на объект
if (s.isEmpty()) { // NullPointerException
    System.out.println("String is empty");
}
```

```
String s = null; // строка не указывает на объект
if (s != null && s.isEmpty()) {
    System.out.println("String is empty");
}
```

## concat(String): String

```
String str1 = "Java";  
String str2 = "Hello";  
str2 = str2.concat(str1); // HelloJava
```

## join(CharSequence, CharSequence...): String

```
String str1 = "Java";  
String str2 = "Hello";  
String str3 = String.join(" ", str2, str1); // Hello Java
```

## charAt(int): char

```
String str = "Java";  
char c = str.charAt(2);  
  
System.out.println(c); // v
```



## getChars(int, int, char[], int): void

```
String str = "Hello world!";  
int srcBegin = 6;  
int srcEnd = 11;  
char[] dst = new char[end - start];  
int dstBegin = 0;  
str.getChars(srcBegin, srcEnd, dst, dstBegin);  
  
System.out.println(dst); // world
```

## **equals(Object): boolean**

```
String str1 = "Hello";  
String str2 = "hello";  
  
System.out.println(str1.equals(str2)); // false
```

## **equalsIgnoreCase(Object): boolean**

```
String str1 = "Hello";  
String str2 = "hello";  
  
System.out.println(str1.equalsIgnoreCase(str2)); // true
```

## regionMatches(int, String, int, int): boolean

```
regionMatches(int toffset, String other, int ooffset, int len): boolean  
regionMatches(boolean ignoreCase, int toffset, String other, int ooffset, int len): boolean
```

## **regionMatches(int, String, int, int): boolean**

```
String str1 = "Hello world";  
String str2 = "I work";  
boolean result = str1.regionMatches(6, str2, 2, 3);  
  
System.out.println(result); // true
```

## **compareTo(String): int** и **compareToIgnoreCase(String): int**

```
String str1 = "hello";  
String str2 = "world";  
String str3 = "hell";  
  
System.out.println(str1.compareTo(str2)); // -15 -> str1 меньше чем str2  
System.out.println(str1.compareTo(str3)); // 1 -> str1 больше чем str3
```

**indexOf(String): int** **lastIndexOf(String): int**

```
String str = "Hello world";  
int index1 = str.indexOf('l'); // 2  
int index2 = str.indexOf("wo"); // 6  
int index3 = str.lastIndexOf('l'); // 9
```

**startsWith(String): boolean** **||** **endsWith(String): boolean**

```
String str = "myfile.exe";  
boolean start = str.startsWith("my"); // true  
boolean end = str.endsWith("exe"); // true
```



## replace(CharSequence, CharSequence)

```
String str = "Hello world";  
String replStr1 = str.replace('l', 'd'); // Heddo wordd  
String replStr2 = str.replace("Hello", "Bye"); // Bye world
```

## trim(): String

```
String str = "  hello world  ";  
str = str.trim(); // "hello world"
```

## **substring(int): String and substring(int, int): String**

```
String str = "Hello world";  
String substr1 = str.substring(6); // "world"  
String substr2 = str.substring(3,5); // "lo"
```

## toLowerCase(): String **||** toUpperCase(): String

```
String str = "Hello World";  
System.out.println(str.toLowerCase()); // hello world  
System.out.println(str.toUpperCase()); // HELLO WORLD
```

## **split(String): String[] and split(String, int): String[]**

```
String text = "FIFA will never regret it";  
String[] words = text.split(" ");  
for (String word : words) {  
    System.out.println(word);  
}
```

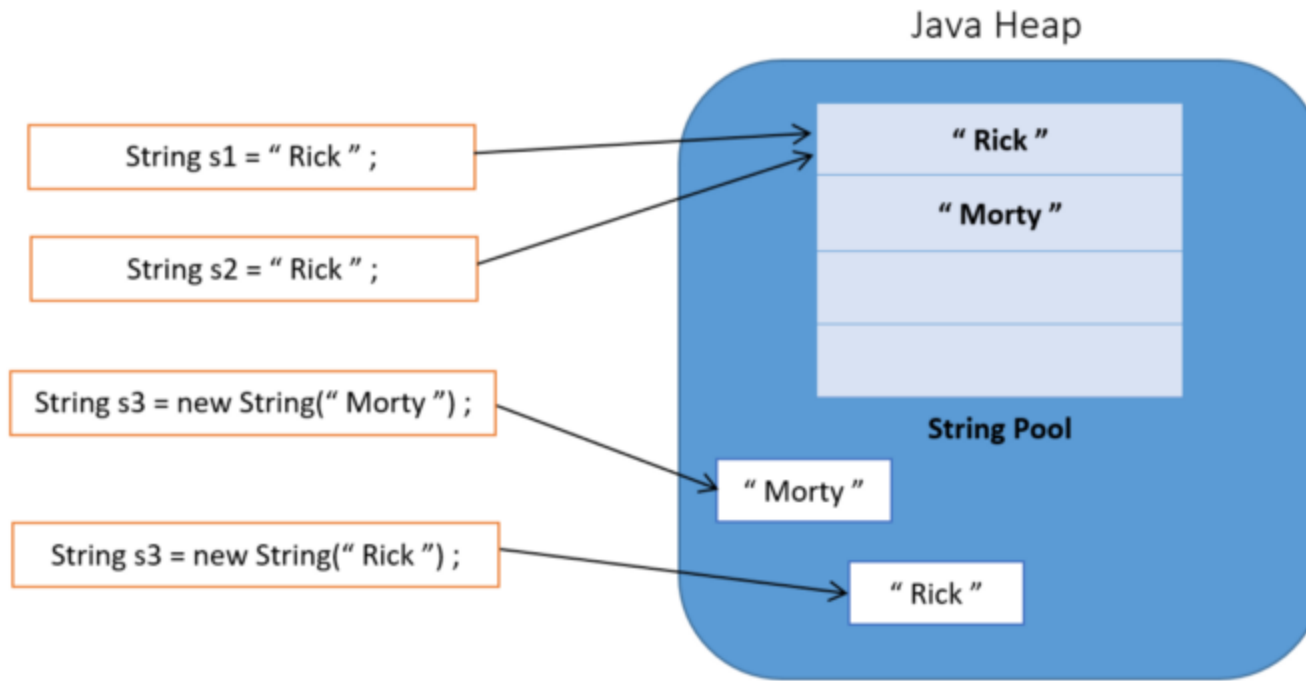
**split(String): String[]**

```
FIFA  
will  
never  
regret  
it
```

# String Pool

# String Pool

**String Pool (Пул строк)** — это набор строк, который хранится в **Heap**.





# String Pool

- При создании объекта через оператор **new** строка не помещается в **String Pool**.
- Для того чтобы поместить строку в **String Pool** используется метод **intern()**.

# StringBuffer и StringBuilder

# Immutable **String**

Класс **String** **immutable** (не изменяемый).

```
String str = "Hello";  
str += " Java";
```

Код приведенный выше, приводит к тому, что создается новый объект, и содержимое обеих исходных строк в него копируется.

# StringBuffer и StringBuilder

- Эту проблему решают объекты типа **StringBuilder** или **StringBuffer**.
- Оба класса позволяют менять содержимое находящихся в них строк.

```
String str = "Hello";  
StringBuilder strBuilder = new StringBuilder(str);  
strBuilder.append(" Java");
```

# StringBuffer и StringBuilder

- Класс **StringBuilder** - *NOT thread safe* (потоко-НЕбезопасный), но быстрый
- Класс **StringBuffer** - *thread safe* (потоко-безопасный), но медленный

## **StringBuffer** constructors

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

## **StringBuilder** constructors

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

# StringBuffer и StringBuilder

```
String str = "Java";  
StringBuffer strBuffer = new StringBuffer(str);  
System.out.println("Емкость: " + strBuffer.capacity()); // 20  
strBuffer.ensureCapacity(32);  
System.out.println("Емкость: " + strBuffer.capacity()); // 42  
System.out.println("Длина: " + strBuffer.length()); // 4
```



## **charAt(int): char и setCharAt(int, char): void**

```
StringBuffer strBuffer = new StringBuffer("Java");  
char c = strBuffer.charAt(0); // J  
System.out.println(c);  
strBuffer.setCharAt(0, 'c');  
System.out.println(strBuffer.toString()); // cava
```

## getChars(int, int, char[], int): void

```
StringBuffer strBuffer = new StringBuffer("world");  
int startIndex = 1;  
int endIndex = 4;  
char[] buffer = new char[endIndex - startIndex];  
strBuffer.getChars(startIndex, endIndex, buffer, 0);  
System.out.println(buffer); // orl
```

## append(\*): StringBuffer

```
StringBuffer strBuffer = new StringBuffer("hello");  
strBuffer.append(" world");  
System.out.println(strBuffer.toString()); // hello world
```

## insert(int, \*): StringBuffer

```
StringBuffer strBuffer = new StringBuffer("word");  
  
strBuffer.insert(3, 'l');  
System.out.println(strBuffer.toString()); // world  
  
strBuffer.insert(0, "s");  
System.out.println(strBuffer.toString()); // sworld
```

## **delete(int, int): StringBuffer**

## **deleteCharAt(int): StringBuffer**

```
StringBuffer strBuffer = new StringBuffer("assembler");  
strBuffer.delete(0,2);  
System.out.println(strBuffer.toString()); // sembler  
  
strBuffer.deleteCharAt(6);  
System.out.println(strBuffer.toString()); // semble
```

## **substring(int): String и substring(int, int): String**

```
StringBuffer strBuffer = new StringBuffer("hello java!");  
String str1 = strBuffer.substring(6); // обрезка строки с 6 символа до конца  
System.out.println(str1); //java!  
  
String str2 = strBuffer.substring(3, 9); // обрезка строки с 3 по 9 символ  
System.out.println(str2); //lo jav
```

## setLength(int): void

```
StringBuffer strBuffer = new StringBuffer("hello");  
strBuffer.setLength(10);  
System.out.println(strBuffer.toString()); // "hello      "  
  
strBuffer.setLength(4);  
System.out.println(strBuffer.toString()); // "hell"
```

## replace(int, int, String): StringBuffer

```
StringBuffer strBuffer = new StringBuffer("hello world!");  
strBuffer.replace(6, 11, "java");  
System.out.println(strBuffer.toString()); // hello java!
```



## reverse(): StringBuffer

```
StringBuffer strBuffer = new StringBuffer("assembler");  
strBuffer.reverse();  
System.out.println(strBuffer.toString()); // relbmessa
```