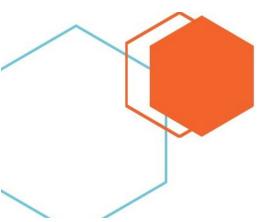
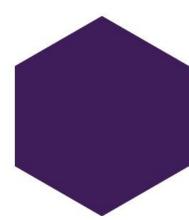




Аннотация: этот документ является практическим руководством к Week 1 Day 3 в Coding Bootcamp.

Академия Ковалевского





• • •

Теория	3
Задания: обновленный StdString	4
Новый StdString (теперь банановый)	4
StdString(char[] base)	5
StdString()	5
StdString(StdString stdString)	5
StdString toAsciiLowerCase()	6
StdString toAsciiUpperCase()	6
StdString subString(int from, int to)	6
StdString concat(StdString that)	7
StdString[] split(char separator)	7
StdString trim()	7
StdString removeCharacter(char toRemove)	8



• • •

1 Теория

В данном занятии рекомендуется ознакомится с такими топиками как:

- Наследование (видео)
- Ключевое слово super (<u>пример статьи</u>)
- Аргументы переменной длины (пример статьи)

Напоминаем, что ссылки тут на внешние источники просто примеры, крайне рекомендуем самостоятельно поискать материалы по интересующим темам что бы все в группе смотрели разные источники по теме.



• •

3адания: обновленный StdString

Новый StdString (теперь банановый)

Чтобы добавить новые возможности в наш класс мы создадим класс наследник вот с такой сигнатурой:

```
package com.kovalevskyi.academy.codingbootcamp.week1.day3;
import com.kovalevskyi.academy.codingbootcamp.week1.day0.StringUtils;
public class StdString extends
com.kovalevskyi.academy.codingbootcamp.week1.day2.StdString {
public StdString(char[] base) {
  // TODO
}
public StdString() {
  // TODO
public StdString(StdString that) {
  // TODO
public StdString toAsciiLowerCase() {
  // TODO
}
public StdString toAsciiUpperCase() {
  // TODO
public StdString subString(int from, int to) {
  // TODO
}
public StdString concat(StdString... that) {
  // TODO
}
```

• •

```
public StdString[] split(char separator) {
    // TODO
}

public StdString trim() {
    // TODO
}

public StdString removeCharacter(char toRemove) {
    // TODO
}
```

ВАЖНО! Путь к классу родителя должен соответствовать реальному размещению файла класса (com.kovalevskyi.academy.codingbootcamp.week1.day2.StdString).

А теперь рассмотрим их все по порядку.

StdString(char[] base)

```
public StdString(char[] base) {
  // TODO
}
```

Конструктор, который создает новуй строку на базе переданного массива символов.

StdString()

```
public StdString() {
   // TODO
}
```

Пустой конструктор, который создает пустую строку.

StdString(StdString stdString)

```
public StdString(StdString stdString) {
  // TODO
}
```

Конструктор копирования, который создает строку идентичной той, что передана на вход.



StdString toAsciiLowerCase()

```
public StdString toAsciiLowerCase() {
   // TODO
}
```

Создает копию строки, в которой все символы нижнего регистра. Например:

```
"CaT12" => "cat12"
" asdT2" => " asdt2"
```

Если символ не ascii, то метод должен кинуть исключение IllegalArgumentException.

StdString toAsciiUpperCase()

```
public StdString toAsciiUpperCase() {
    // TODO
}
```

То же, что и предыдущий, только для большого регистра.

StdString subString(int from, int to)

```
public StdString subString(int from, int to) {
   // TODO
}
```

Возвращает подстроку, начиная с заданного индекса (включительно) по заданный индекс (не включительно). Например:

```
"ased 79s", 1 5 => "sed"
"cat-dog-123", 2, 3 => "t"
```

В случае, если какой-либо из индексов выходит за пределы строки метод кидает IndexOutOfBoundsException. В случае, если аргументы неверные (например, начальный индекс больше конечного) метод кидает IllegalArgumentException



StdString concat(StdString... that)

```
public StdString concat(StdString... that) {
   // TODO
}
```

Метод создает новую строку, в которую входят все символы текущей строки, а также все символы каждой строки, которая была передана на вход. Например:

"thisStr", "thatStr123", "thatStr222" => "thisStrthatStr123thatStr222"

StdString[] split(char separator)

```
public StdString[] split(char separator) {
   // TODO
}
```

Метод делит текущую строку на подстроки в соответствии с заданным делителем и возвращает массив этих строк. Например:

```
"cat trim dog", ' ' => "cat", "trim" "dog"

"one!str!two ! three", '!' => "one", "str", "two ", " three"

"cat dog ", ' ' => "cat", "dog", "", ""

"cat dog ", '!' => "cat dog "
```

StdString trim()

```
public StdString trim() {
   // TODO
}
```

Метод создает новую строку, которая равна текущей строке, но только без пробелов в начале и в конце строки (если они там вообще есть). Например:

```
" hi "=> "hi"
"hi" => "hi"
" 1 2" => "1 2"
" "=> ""
```



• • •

StdString removeCharacter(char toRemove)

```
public StdString removeCharacter(char toRemove) {
   // TODO
}
```

Метод создает новую строку, которая равна текущей, однако без определенного символа, который передан на вход. Например:

```
"new,1,2,3"; '=> "new123"
" 1 2", ''=> "12"
"asdf123", '!' => "asdf123"
```

