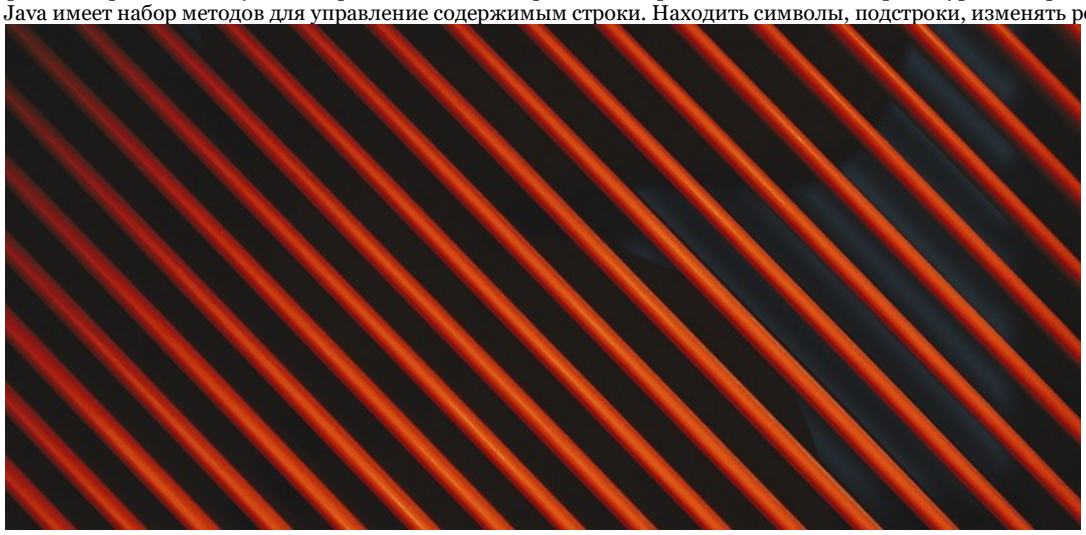
Управление строками, функции для работы со строками в Java

articles

- 15 декабря 2015
- 32779 views

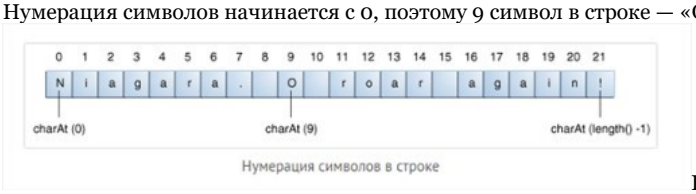
В этом уроке мы продолжим изучение строк в Java. Основы работы со строками можно посмотреть в уроке «Строки в Java». Класс `String` в Java имеет набор методов для управление содержимым строки. Находить символы, подстроки, изменять регистр и другие задачи. 

Получение символов и подстрок

Вы можете получить символ, находящийся на определенной позиции в строке, вызвав метод `charAt()`. Индекс первого символа в строке — 0, последнего — `length()` -1. Следующий код возвращает 9 символ строки.

```
String anotherPalindrome = "Niagara. O roar again!";
char aChar = anotherPalindrome.charAt(9);
```

Нумерация символов начинается с 0, поэтому 9 символ в строке — «O».



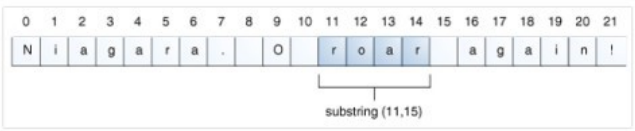
Нумерация символов в строке

Если вам необходимо получить не один символ, а часть строки, можно использовать метод `substring`. Метод `substring` имеет два варианта:

Метод	Описание
`String substring(int beginIndex, int endIndex)`	Возвращает подстроку данной строки, начиная с символа с индексом beginIndex, заканчивая endIndex – 1.
`String substring(int beginIndex)`	Возвращает подстроку данной строки, начиная с символа под номером beginIndex и до конца строки.

Следующий код вернет подстроку строки, начиная с 11 символа, но не включая 15, получится слово «roar»

```
String anotherPalindrome = "Niagara. O roar again!";
String roar = anotherPalindrome.substring(11, 15);
```



Подстрока Java

Другие методы для управления строками

В таблице приведены некоторые методы для работы со строками и их описание.

Метод	Описание
`String[] split(String regex)` `String[] split(String regex, int limit)`	Ищет совпадения в строке согласно заданному регулярному выражению и разбивает строку на массив. Необязательный аргумент limit задает максимальный размер возвращаемого массива.
`CharSequence subSequence(int beginIndex, int endIndex)`	Возвращает последовательность символов, начиная с `beginIndex`, заканчивая `endIndex - 1`.
`String trim()`	Возвращает строку, в которой удалены лишние пробелы в начале строки и в конце.
`String toLowerCase()` `String toUpperCase()`	Возвращает копию строки, символы которой переведены в нижний или верхний регистр. Если преобразований не требуется возвращается оригинальная строка.

Поиск символов и подстрок

Рассмотрим еще несколько методов для поиска символов и подстрок. Класс `String` содержит методы, которые возвращают позицию символа или подстроки в строке: `indexOf()` и `lastIndexOf()`. Методы `indexOf()` осуществляют поиск с начала строки, `lastIndexOf()` - с конца. Если данные методы не нашли совпадений, они возвращают -1. Также класс `String` содержит метод `contains`, который возвращает *true*, если заданная последовательность символов содержится в строке. Используйте этот метод, если необходимо узнать о существовании подстроки в строке, а её позиция не важна. В следующей таблице описываются методы `indexOf()` и `lastIndexOf()`.

Метод	Описание
`int indexOf(int ch)` `int lastIndexOf(int ch)`	Возвращает индекс первого(последнего) вхождения символа в строке.
`int indexOf(int ch, int fromIndex)` `int lastIndexOf(int ch, int fromIndex)`	Возвращает индекс первого(последнего) вхождения символа в строке, начиная поиск с указанного индекса.
`int indexOf(String str)` `int lastIndexOf(String str)`	Возвращает индекс первого(последнего) вхождения подстроки в строке.
`int indexOf(String str, int fromIndex)` `int lastIndexOf(String str, int fromIndex)`	Возвращает индекс первого(последнего) вхождения подстроки в строке, начиная поиск с указанного индекса.
`boolean contains(CharSequence s)`	Возвращает true, если заданная последовательность символов содержится в строке.

`CharSequence` — это интерфейс, который реализует класс `String`, поэтому вы можете передавать строки в метод `contains()`.

Изменение строк. Замена символов и подстрок

Класс `String` имеет несколько методов для вставки символов и подстрок в строку. В таблице описаны методы для замены найденных символов и подстрок.

Метод	Описание
`String replace(char oldChar, char newChar)`	Возвращает новую строку, в которой все oldChar заменены на newChar.
`String replace(CharSequence target, CharSequence replacement)`	Заменяет все вхождения подстроки target на строку replacement.
`String replaceAll(String regex, String replacement)`	Заменяет все подстроки, которые описывает заданное регулярное выражение на replacement
`String replaceFirst(String regex, String replacement)`	Заменяет только первую подходящую подстроку.

Пример

Следующий класс `Filename` показывает пример использования методов `lastIndexOf()` и `substring()` для использования разных частей строки с именем файла.

```
public class Filename {
    private String fullPath;
    private char pathSeparator,
                 extensionSeparator;

    public Filename(String str, char sep, char ext) {
        fullPath = str;
        pathSeparator = sep;
        extensionSeparator = ext;
    }

    public String extension() {
        int dot = fullPath.lastIndexOf(extensionSeparator);
        return fullPath.substring(dot + 1);
    }

    // получение имени файла без расширения
    public String filename() {
        int dot = fullPath.lastIndexOf(extensionSeparator);
        int sep = fullPath.lastIndexOf(pathSeparator);
        return fullPath.substring(sep + 1, dot);
    }

    public String path() {
        int sep = fullPath.lastIndexOf(pathSeparator);
        return fullPath.substring(0, sep);
    }
```

Теперь рассмотрим программу, которая использует класс `Filename`:

```
public class FilenameDemo {
    public static void main(String[] args) {
        final String FPATH = "/home/user/index.html";
        Filename myHomePage = new Filename(FPATH, '/', '.');
        System.out.println("Extension = " + myHomePage.extension());
        System.out.println("Filename = " + myHomePage.filename());
        System.out.println("Path = " + myHomePage.path());
    }
}
```

Программа выведет:

```
Extension = html
Filename = index
Path = /home/user
```

Метод `extension` использует метод `lastIndexOf` для определения последнего вхождения ".". Метод `substring` использует это значение для нахождения расширения файла. Ссылка на первоисточник: Управление строками, функции для работы со строками в Java Управление строками, функции для работы со строками в Java. Часть 2