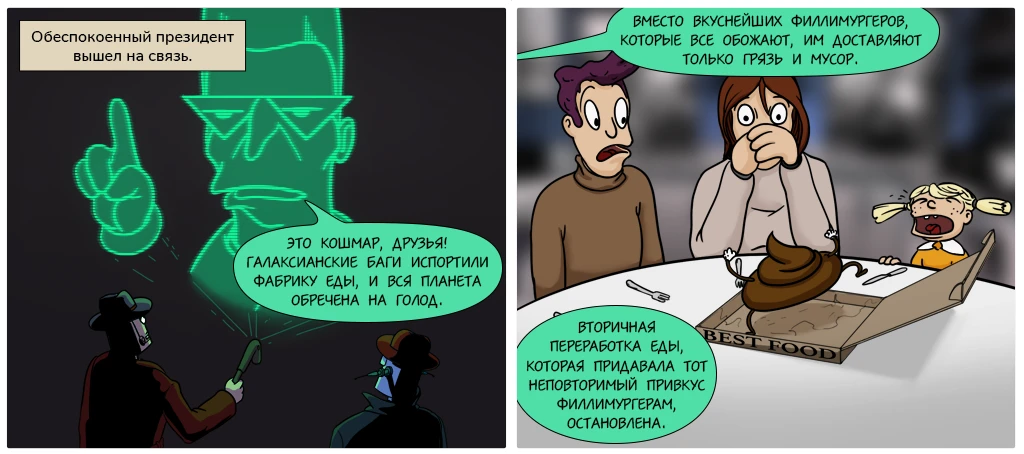
**Примеры работы со строками**

[Java Syntax Pro](https://javarush.ru/quests/QUEST_JAVA_SYNTAX_PRO)

[Уровень 2](https://javarush.ru/quests/lectures?quest=QUEST_JAVA_SYNTAX_PRO&level=2), Лекция 3





**1. Конкатенация — склеивание строк**

Есть очень интересная и простая вещь, которую можно делать со строками в Java — их можно склеивать. Такая операция называется конкатенация. Ну или чтобы легче запомнилось — Кон-Котэ-Нация. Хотя часто ее называют «склеивание строк» или даже просто «сложение строк».

Чтобы склеить две строки, нужно использовать знак +. Это очень просто:

"значение1" + "значение2"

Конкатенация двух строк

Примеры:

|  |  |
| --- | --- |
| **Команда** | **Примечание** |
| String name = "Аня" + "Аня"; | name содержит строку АняАня |
| String city = "New York" + "Аня"; | city содержит строку New YorkАня |
| String message = "Hello! " + "Аня"; | message содержит строку Hello! Аня |

Ну и, конечно, одновременно можно складывать много строк, а также складывать строки и переменные.

Примеры:

|  |  |
| --- | --- |
| **Команда** | **Примечание** |
| String name = "Аня";  String city = "New York";  String message = "Hello!" + city + name + city; | name содержит строку Аня city содержит строку New York message содержит строку Hello!New YorkАняNew York |

В последнем примере видно, что текст в переменной message плохо читается, т.к. в нем не хватает пробелов. Чтобы задать один пробел или несколько, нужно просто написать их в коде и обернуть в двойные кавычки. Это проще, чем кажется:

" "

Строка, содержащая один пробел

Кстати, если между кавычками совсем не оставить места (написать две двойные кавычки подряд), получится так называемая «пустая строка»:

""

Пустая строка

С одной стороны, она вроде бы и есть, а с другой, при ее выводе на экран ничего не выводится. И при склейке с другими строками ничего не происходит. Эдакий аналог нуля, только для строк.

**2. Преобразование к строковому типу**

Как уже говорилось выше, разработчики Java сделали так, что абсолютно все переменные, объекты, выражения в Java можно преобразовать к типу String.

Более того, это автоматически происходит, когда мы складываем тип String с каким-нибудь другим типом. Примеры:

|  |  |
| --- | --- |
| **Команда** | **Примечание** |
| int a = 5;  String name = "Аня" + a; | name содержит строку Аня5 |
| int a = 5;  String city = a + "New York" + a; | city содержит строку 5New York5 |
| int number = 10;  String code = "Yo";  String message = "Hello! " + number + code; | message содержит строку Hello! 10Yo |

Во всех трех примерах мы спокойно складывали переменные типа int и String, и в результате всегда получался тип String.

Арифметические операции с типом String проводить нельзя. Даже если строка целиком состоит из цифр.

Примеры:

|  |  |
| --- | --- |
| **Команда** | **Примечание** |
| int a = 5;  String name = "1" + a; | name содержит строку 15 |
| int a = 5;  String city = a + "9" + a; | city содержит строку 595 |
| int number = 10;  String code = "10";  String message = "" + number + code; | message содержит строку 1010 |

Операция сложения выполняется слева направо, так что результат может быть несколько неожиданным. Пример:

|  |  |
| --- | --- |
| **Команда** | **Примечание** |
| int a = 5;  String name = a + a + "1" + a; | name содержит строку 1015 |

Порядок выполнения: ((a + a) + "1") + a

**3. Преобразование строки в число**

Чтобы преобразовать число в строку в Java, достаточно сложить его и пустую строку:

String str = "" + число;

Преобразование числа в строку

А вот что делать, если нужно преобразовать строку в число? Ну, любую строку в число преобразовать нельзя. Но вот если строка состоит только из цифр, то все-таки можно. Для этого есть специальный метод у класса Integer.

Выглядит эта команда так:

int x = Integer.parseInt(строка);

Где int x – это объявление целочисленной переменной x, а строка — это число заданное в виде строки (строка состоящая из цифр).

Примеры:

|  |  |
| --- | --- |
| **Команда** | **Примечание** |
| String str = "123";  int number = Integer.parseInt(str); | number содержит число 123; |
| int number = Integer.parseInt("321"); | number содержит число 321 |
| int number = Integer.parseInt("321" + 0); | number содержит число 3210 |
| int number = "321"; | Не скомпилируется: переменная типа int, а значение типа String |

**4. Некоторые методы работы со строками**

Ну и напоследок хотелось бы рассказать о нескольких методах типа String.

**Метод length()**

Метод length() позволяет узнать длину строки – сколько в ней символов.

Примеры:

|  |  |
| --- | --- |
| **Команда** | **Примечание** |
| String name = "Rome";  int count = name.length(); | count содержит значение 4 |
| int count = "".length(); | count содержит значение 0 |
| String name = "Rom";  int count = (name + 12).length(); | count содержит значение 5 |

Да, методы можно вызывать у всего, что имеет тип String, даже у выражения:

(name + 12).length()

Вызов метода length() у выражения типа String

**Метод toLowerCase()**

Метод toLowerCase() позволяет преобразовать все символы строки в маленькие (строчные):

Примеры:

|  |  |
| --- | --- |
| **Команда** | **Примечание** |
| String name = "Rom";  String name2 = name.toLowerCase(); | name2 содержит строку rom |
| String name = "".toLowerCase(); | name содержит пустую строку |
| String name = "ROM123";  String name2 = name.toLowerCase(); | name2 содержит строку rom123 |

**Метод toUpperCase()**

Метод toUpperCase() позволяет преобразовать все символы строки в большие (заглавные):

Примеры:

|  |  |
| --- | --- |
| **Команда** | **Примечание** |
| String name = "Rom";  String name2 = name.toUpperCase(); | name2 содержит строку ROM |
| String name = "rom123";  String name2 = name.toUpperCase(); | name2 содержит строку ROM123 |

