3\_5 сравнение ссылок

**1. Сравнения**

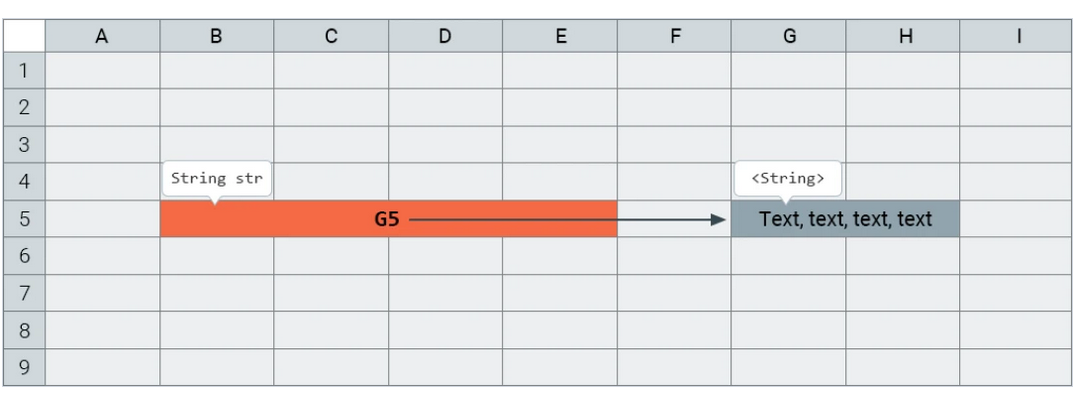
Очень часто программисту нужно сравнивать различные переменные между собой. И, как вы уже успели убедиться, не все так однозначно.

Целые числа сравнивать очень легко — просто используйте == и все. Чтобы сравнить вещественные числа, вам уже придется сравнивать их разность (вернее, модуль разности) с каким-нибудь очень маленьким числом.

Сравнение строк еще сложнее. А всё потому, что, во-первых, строки — это объекты, а во-вторых, в зависимости от ситуации программисту часто хочется, чтобы сравнение строк проходило чуть-чуть иначе (учитывая или не учитывая определенные факторы).

**2. Расположение строк в памяти**

Как вы уже успели увидеть, строки в памяти хранятся не так, как целые и вещественные числа:



Для хранения строк используется **два блока памяти**: один блок хранит сам текст (его размер зависит от размера текста), а второй (размером 4 байта) хранит адрес первого блока.

Хотя опытный программист в такой ситуации скажет что-то вроде «Переменная str типа String хранит ссылку на объект типа String

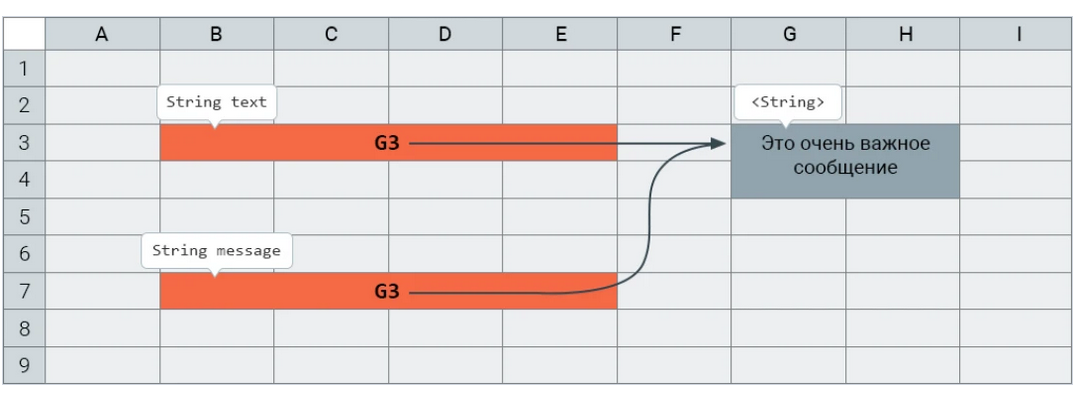
**3. Присваивание ссылок на строки**

Выгода такого подхода становится очевидной, если вам нужно присвоить одной строковой переменной другую строковую переменную. Пример:

String text = "Это очень важное сообщение";

String message = text;

А вот что в результате будет в памяти:



В результате этой операции присваивания объект String так и останется, где был, а в переменную messageскопируется только его адрес (ссылка на объект).

**4. Работа со ссылками и объектами**

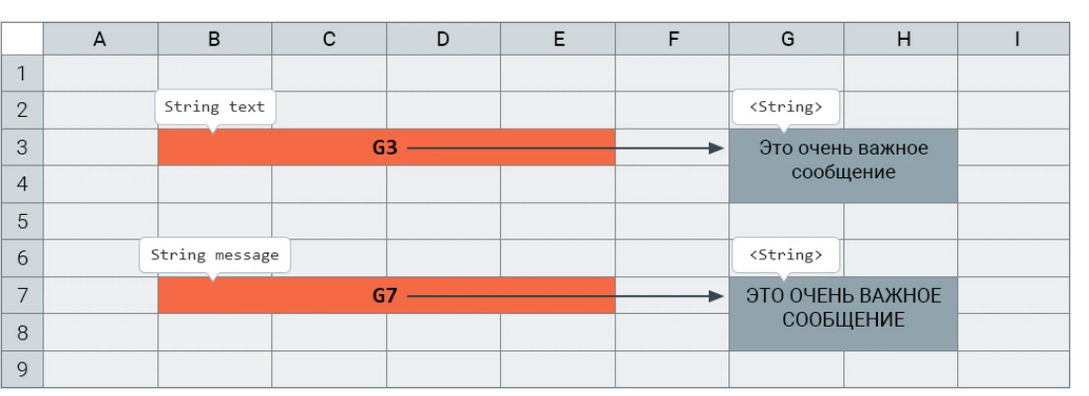
А вот если вы решите преобразовать строку к верхнему регистру (заглавные буквы), Java-машина сделает все правильно: у вас будут два объекта типа String, и переменные text и message будут хранить ссылки: каждая на свой объект.

Пример:

String text = "Это очень важное сообщение";

String message = text.toUpperCase();

А вот что в результате будет в памяти:



Обращаю ваше внимание, что метод toUpperCase() не меняет ту строку, у которой он был вызван. Вместо этого он создает новую строку (новый объект) и возвращает ссылку на него.

Или еще более интересный пример. Скажем, вы решили передать строку в объект типа Scanner (чтобы он читал значения из нее).

Пример:

String text = "10 20 40 80";

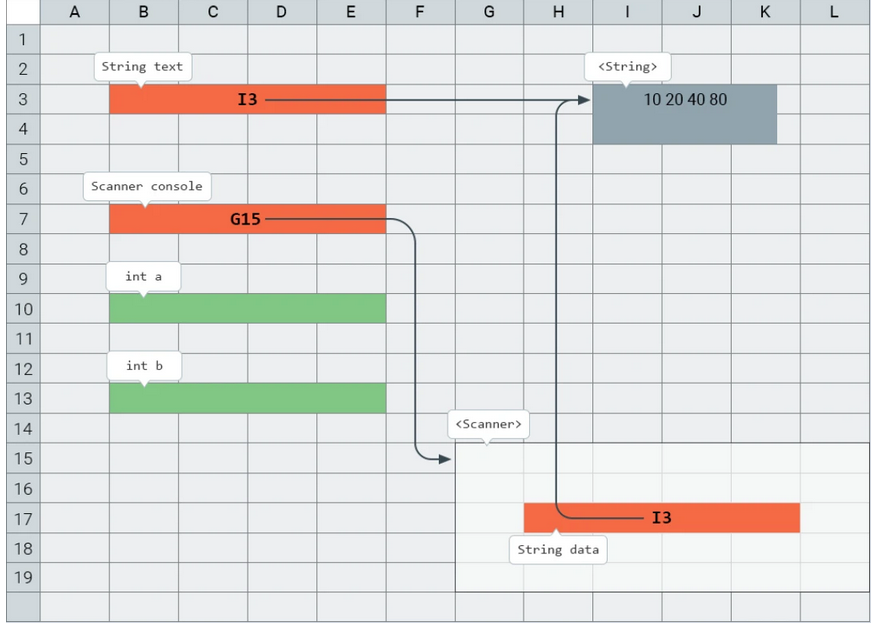
Scanner console = new Scanner(text);

int a = console.nextInt();

int b = console.nextInt();

Подробнее о работе класса Scanner вы можете узнать [по ссылке](https://javarush.ru/groups/posts/klass-scanner).

Вот как это все будет храниться в памяти:



При этом объект типа String как был в памяти в единственном экземпляре, так там и хранится — везде передаются и хранятся только ссылки на него.

**5. Сравнение ссылок на объекты типа String**

Ну и наконец мы дошли до самого интересного — сравнения строк.

Для сравнения строковых переменных можно использовать два оператора: == (равно) и != (не равно). Операторы «больше», «меньше», «больше либо равно» использовать нельзя — компилятор не допустит.

Но есть интересный нюанс: что у нас хранится в строковых переменных? Правильно: адреса (ссылки) на объекты. Вот эти самые адреса сравниваться и будут:

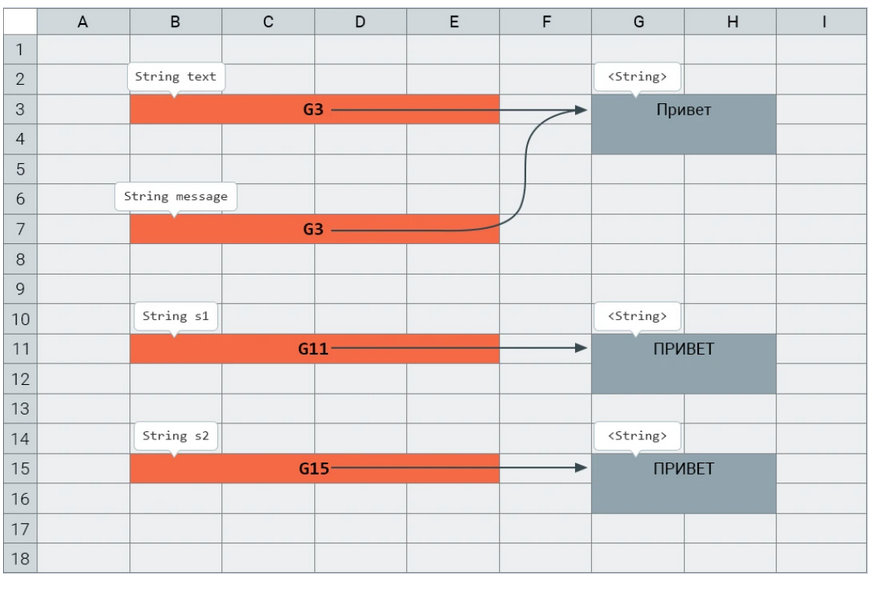
String text = "Привет";

String message = text;

String s1 = text.toUpperCase();

String s2 = text.toUpperCase();

Вот что будет в памяти:



Переменные message и text хранят адрес (ссылку) одного и того же объекта. А вот переменные s1 и s2 хранят ссылки на очень похожие объекты, но все-таки не на один и тот же объект.

И если вы сравните в коде эти 4 переменные, получите вот такой результат:

|  |  |
| --- | --- |
| **Код** | **Вывод на экран** |
| String text = "Привет";  String message = text;  String s1 = text.toUpperCase();  String s2 = text.toUpperCase();  System.out.println(text == message);  System.out.println(text == s1);  System.out.println(s1 == s2); | true // адреса равны  false // адреса разные  false // адреса разные |