String, StringBuilder и StringBuffer классы работы со строкой

При работе с текстовым данными, Java предоставляет вам 3 класса **String**, **StringBuffer** и **StringBuilder**.

При работе с большими данными вам стоит использовать **StringBuffer**или  **StringBuilder**чтобы достичь лучшую эффективность. Стандартно эти 3 класса имеют много общего.

* **String** не изменяется (*immutable*), это понятие будет детально описано в статье. Он не позволяет иметь подкласс.
* **StringBuffer**, **StringBuilder**могут быть изменены (*mutable*)

**StringBuilder и**  **StringBuffer**похожи, лишь отличаются в ситуациях связанные с использованием многопоточности (Multi Thread).

* При обработке тектса со многими потоками(Thread) вам следует использовать **StringBuffer**, чтобы предотвратить конфликт между потоками.
* При обработке текста с одним потоком стоит использовать **StringBuilder**.

Если сравнить скорость обработки, то самый лучший это  **StringBuilder**, затем идет  **StringBuffer**и последний это  **String**.

В java, **String** это особенный класс, причиной является то, что он используется постоянно в программе, поэтому он требует производительность и гибкость. Это причина, почему **String** имеет свойства объекта и примитивный вид (primitive).

**Примитивный вид:**

**String literal** хранится в стеке (stack),

Объекты  **String** хранятся в Heap.

**String literal** требует меньше пространства для хранения, и дешевле при обработке.

Вы можете создать **string literal** (строковый литерал):

* String literal = "Hello World";

Вы можете использовать оператор “+” чтобы соединить 2 string, этот оператор уже знаком и используется для данных примитивного вида как  **int, float, double**.   
**String literal** хранятся в общем пуле (common pool).

Два **string literal** имеющие одно содержание, используют общее место в памяти в стеке, это помогает сэкономить память *(см.пример дальше)*.

**Свойство объекта:**

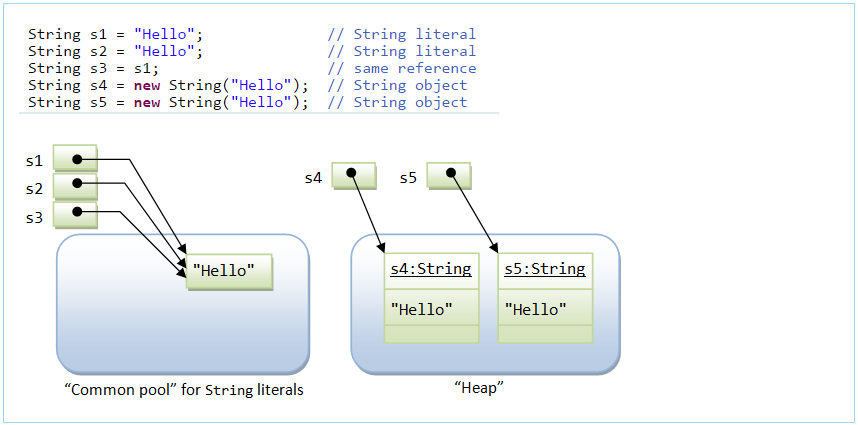
Так как  **String** является классом, он может быть создан используя оператор  **new**.

* String object = new String("Hello World");

Объекты  **String** хранятся в Heap, требует управление сложности и стоимости пространства хранения в памяти. Два объекта **String** с одинаковым содержанием хранятся на 2 разных областях в Heap.

3.2- String Literal vs. String Object

Как упомянуто, существует два способа построить строку (String): неявное строительство путем назначения строкового литерала ( *String literal*) или явно создать объект  **String** через оператор  **new** и составляющие  **String**. Например,



“Stack”

**String literal** с одинаковым содержанием разделяют одно место хранения в общем пуле (***common pool***) в **Stack**. При этом Объекты **String**хранятся в  **Heap**, и не разделяют место хранения включая эти 2 объекта string с одинаковым содержанием.

**equals() vs ==**

Метод  ***equals()*** используется для сравнения 2-х объектов, с **String** это означает сравнение содержания 2-х string. Для ссылочных видов (reference), оператор == значит сравнение адреса области хранения объекта. Посмотрим пример:

?

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | String s1 = "Hello";              // String literal  String s2 = "Hello";              // String literal  String s3 = s1;                   // same reference  String s4 = new String("Hello");  // String object  String s5 = new String("Hello");  // String object    s1 == s1;         // true, same pointer  s1 == s2;         // true, s1 and s2 share storage in common pool  s1 == s3;         // true, s3 is assigned same pointer as s1  s1 == s4;         // false, different pointers  s4 == s5;         // false, different pointers in heap    s1.equals(s3);    // true, same contents  s1.equals(s4);    // true, same contents  s4.equals(s5);    // true, same contents |

На самом деле, вам следует использовать  ***String literal***, вместо оператора  **new**. Это ускоряет вашу программу.

