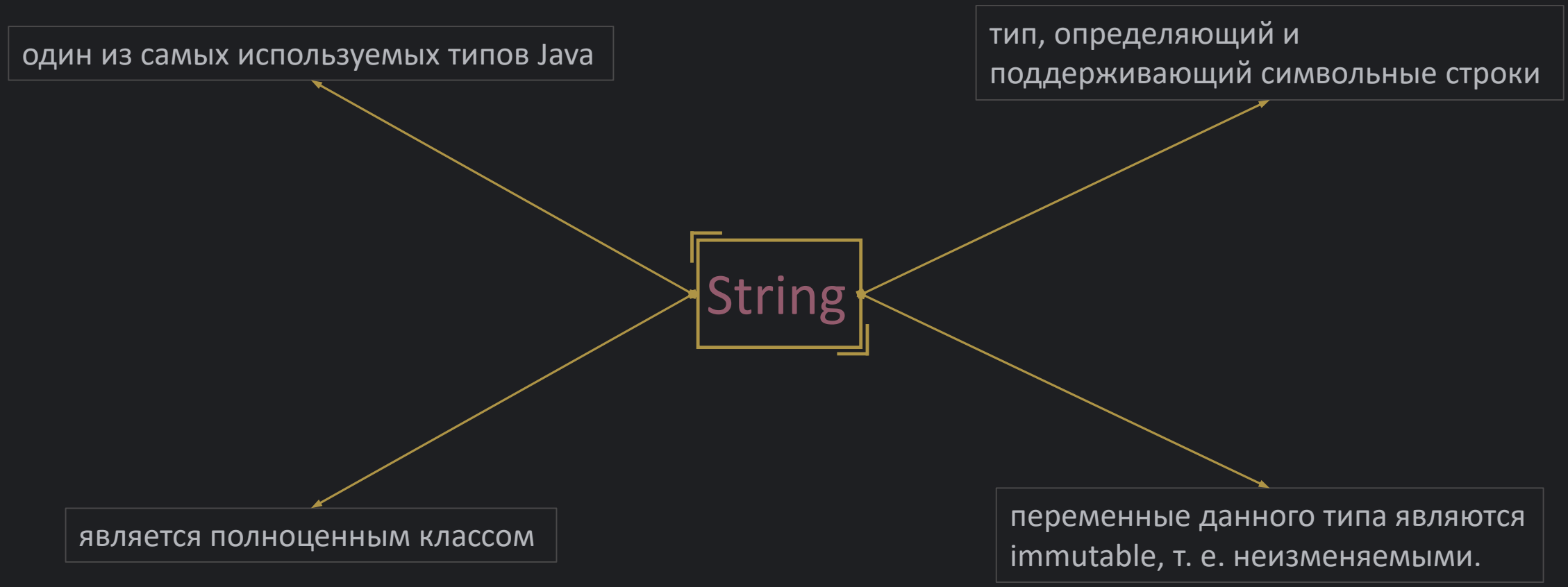
Java: String

Тип String


один из самых используемых типов Java
тип, определяющий и поддерживающий символьные строки
String
является полноценным классом
переменные данного типа являются immutable, т. е. неизменяемыми.

Тип String

Объявление переменной

```
String str1;
String str2, str3, str4;
```

Инициализация переменной

```
str1 = "Hello, World";
str2 = "Hello, World";

str3 = new String( original: "Hello, World");

char[] chars = {'H', 'e', 'l', 'l', 'o',',','W','o','r','l','d'};
str4 = new String(chars);
```

Конкатенация

```
"значение1" + "значение2"
```

```
String str1 = "Иван " + "Иванов";
```

Иван Иванов

```
String lastname = "Иванов";
String str2 = "Иван " + lastname;
```

Иван Иванов

```
String str3 = "Привет! " + "Иван " + "Иванов";
```

Привет! Иван Иванов

Преобразование к строковому типу

```
int a = 5;
String str1 = "Номер " + a;
```

Номер 5

```
int b = 12;
String str2 = b + " лет";
```

12 лет

```
double number = 10.23;
String str3 = "Hello! " + number;
```

Hello! 10.23

```
String str4 = a + b + " - сумма";
```

17 - сумма

Преобразование строки в число

```
Integer num1 = Integer.valueOf( s: "123");
```

num1 содержит число 123

```
int num2 = Integer.parseInt( s: "456");
```

num2 содержит число 456

```
double num3 = Double.parseDouble( s: "123.12");
```

num3 содержит число 123.12

```
String str = "123";
int num4 = Integer.parseInt(str);
```

num4 содержит число 123

```
int num5 = Integer.parseInt( s: "321" + 0);
```

num5 содержит число 3210

```
int num6= "321";
```

Ошибка компиляции

Работа с кодировкой

Код

```
byte[] bytes = new byte[3];
bytes[0] = 97;
bytes[1] = 98;
bytes[2] = 99;
System.out.println("Байты: " + Arrays.toString(bytes));

String str = new String(bytes, charsetName: "UTF-8");
System.out.println("Строка: " + str);

byte[] bytes2 = str.getBytes( charsetName: "UTF-8");
System.out.println("Байты снова: " + Arrays.toString(bytes2));
```

Результат

```
Байты: [97, 98, 99]
Строка: abc
Байты снова: [97, 98, 99]
```

Charset	Description
US-ASCII	Seven-bit ASCII, a.k.a. ISO646-US, a.k.a. the Basic Latin block of the Unicode character set
ISO-8859-1	ISO Latin Alphabet No. 1, a.k.a. ISO-LATIN-1
UTF-8	Eight-bit UCS Transformation Format
UTF-16BE	Sixteen-bit UCS Transformation Format, big-endian byte order
UTF-16LE	Sixteen-bit UCS Transformation Format, little-endian byte order
UTF-16	Sixteen-bit UCS Transformation Format, byte order identified by an optional byte-order mark

Методы работы со строками

Метод length() – возвращает длину строки

```
String str = "Hello";
int num = str.length();
```

num содержит число 5

```
int num2 = "Hello".length();
```

num2 содержит число 5

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

```
String ABAP = "ABAP";
String abap = ABAP.toLowerCase();
```

abap содержит строку «abap»

```
String Java = "Java";
String java = Java.toLowerCase();
```

java содержит строку «java»

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

Сравнение строк

boolean equals(String str) – Проверяет равенство строк

```
String str1 = "abap";
String str2 = "abap";
String str3 = "abap!";

boolean res1 = str1.equals(str2);
boolean res2 = str1.equals(str3);
```

res1 содержит число true
res2 содержит число false

Сравнение строк

`boolean equalsIgnoreCase(String str)`	- Сравнивает строки, игнорируя регистр (размер) букв
`int compareTo(String str)`	- Сравнивает строки лексикографически. Возвращает 0, если строки равны. Число меньше нуля, если текущая строка меньше строки-параметра. Число больше нуля, если текущая строка больше строки-параметра
`boolean compareToIgnoreCase(String str)`	- Сравнивает строки лексикографически, игнорирует регистр. Возвращает 0, если строки равны. Число меньше нуля, если текущая строка меньше строки-параметра. Число больше нуля, если текущая строка больше строки-параметра
`boolean regionMatches(int toffset, String str, int offset, int len)`	- Сравнивает части строк
`boolean startsWith(String prefix)`	- Проверяет, что текущая строка начинается со строки prefix
`boolean endsWith(String suffix)`	- Проверяет, что текущая строка заканчивается на строку suffix

Поиск подстрок

int indexOf(String str)	- Ищет строку str в текущей строке. Возвращает индекс первого символа встретившейся строки.
int indexOf(String str, int index)	- Ищет строку str в текущей строке, пропустив index первых символов. Возвращает индекс найденного вхождения.
int lastIndexOf(String str)	- Ищет строку str в текущей строке с конца. Возвращает индекс первого вхождения.
int lastIndexOf(String str, int index)	- Ищет строку str в текущей строке с конца, пропустив index первых символов.
boolean matches(String regex)	- Проверяет, что текущая строка совпадает с шаблоном, заданным регулярным выражением.

Создание подстрок

`String substring(int beginIndex, int endIndex)`	- Возвращает подстроку, заданную интервалом символов beginIndex..endIndex.
`String repeat(int count)`	- Повторяет текущую строку count раз
`String replace(char oldChar, char newChar)`	- Возвращает новую строку: заменяет символ oldChar на символ newChar
`String replaceFirst(String regex, String replacement)`	- Заменяет в текущей строке подстроку, заданную регулярным выражением.
`String replaceAll(String regex, String replacement)`	- Заменяет в текущей строке все подстроки, совпадающие с регулярным выражением.
`String trim()`	- Удаляет все пробелы в начале и конце строки.
`String[] split(String regex)`	- Разделяет строку на массив подстрок по разделителю regex.

Класс StringTokenizer

Класс StringTokenizer разделяет строку на подстроки по указанному разделителю.
Строка сразу не разбивается на части полностью.
Класс перебирает части строки последовательно и предоставляет очередную часть по запросу.

Методы:

StringTokenizer(String str, String delim)	- Конструктор. На вход передаются строка для разделения и разделитель
String nextToken()	- Возвращает следующую подстроку
boolean hasMoreTokens()	- Проверяет, есть ли еще подстроки.

Код

```
String str = "Добро пожаловать в Java!";

StringTokenizer tokenizer = new StringTokenizer(str, delim: " ");
while (tokenizer.hasMoreTokens())
{
    String token = tokenizer.nextToken();
    System.out.println(token);
}
```

Результат

```
Добро
пожаловать
в
Java!
```

Метод String.format()

Дано:

```
String name = "Петя";
int age = 20;
int weight = 70;
int height = 180;
```

Требуется вывести строку:
Пользователь = {имя: Петя, возраст: 20 лет, рост: 180 см., вес: 70 кг.}

Вариант 1.

```
System.out.println(
        "Пользователь = {имя: " + name + ", возраст: "
                + age + " лет, рост: " + height + " см., вес: " + weight + " кг.}");
```

Вариант 2.

```
System.out.println(
        String.format(
                "Пользователь = {имя: %s, возраст: %d лет, рост: %d см., вес: %d кг.}",
                name, age, height, weight));
```

Метод String.format()

Статический метод format() позволяет задать шаблон объединения строки с данными.

String имя = String.format(шаблон, параметры);

```
String str = String.format(
        "Age=%d, Name=%s", age, name);
```

str содержит строку «Age=20, Name=Петя»

Параметры:

%s	String
%d	целое число: byte, short, int, long
%f	вещественное число: float, double
%b	boolean
%c	char
%t	Date
%%	Символ %

Указание порядкового номера параметра – %1$s

Код

```
String s = String.format(
        "a=%3$d, b=%2$d, c=%d", 1, 2, 3);
```

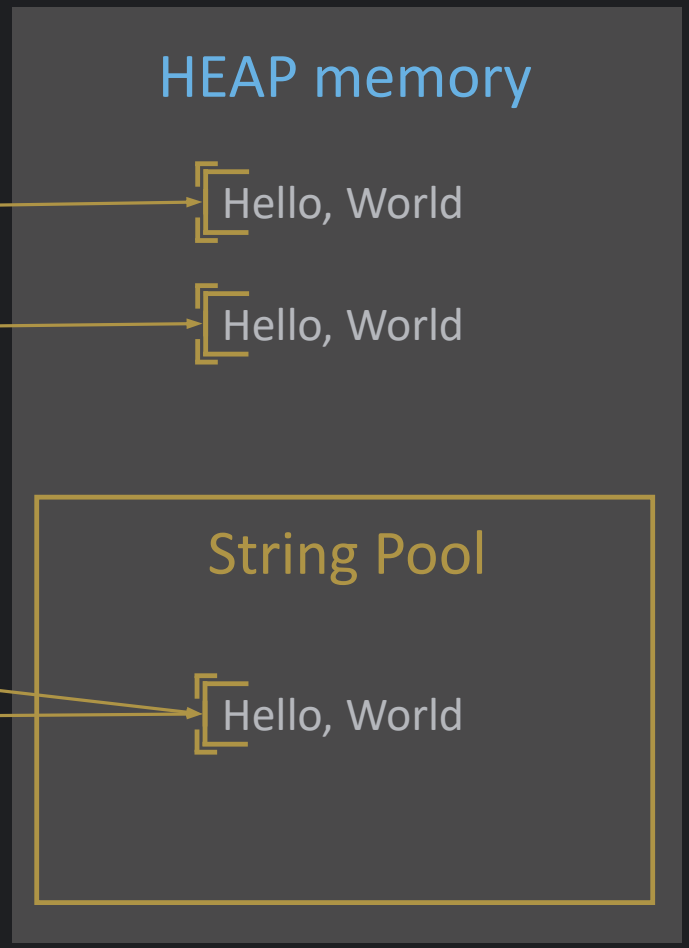
Результат

```
a=3, b=2, c=1
```

String в памяти

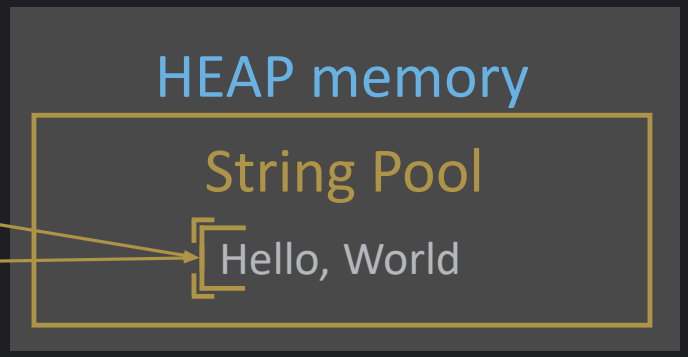
```
str3 = new String( original: "Hello, World");
str4 = new String(chars);

str1 = "Hello, World";
str2 = "Hello, World";
```



String в памяти

```
str1 = "Hello, World";
str2 = "Hello, World";
```



Код:

```
3  public class Main {
4      public static void main(String[] args) {
5
6          String str1;
7          String str2;
8
9          str1 = "Hello, World";
10         str2 = "Hello, World";
11
12         System.out.println("Строки равны? " + (str1 == str2 ));
13     }
14 }
```

Результат:

```
Строки равны? true
```

Обе переменные ссылаются на один и тот же объект

Сравнивает ссылки

String в памяти

Код:

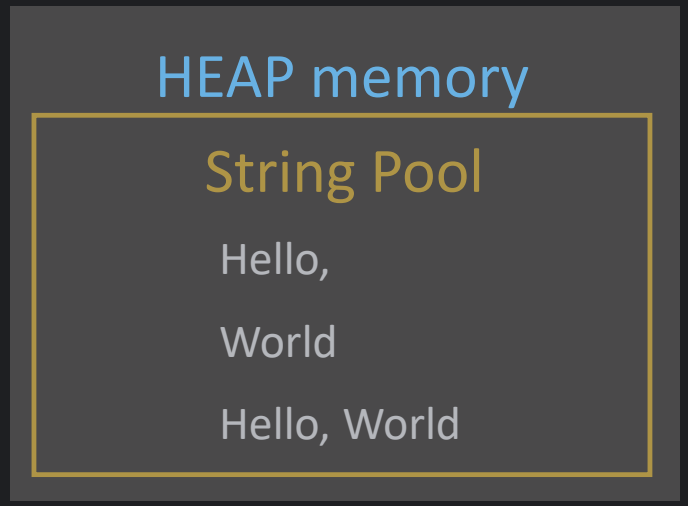
```
3  public class Main {
4      public static void main(String[] args) {
5
6          String str1, str2, str3, str4;
7
8          str1 = "Hello, World";
9          str2 = "Hello, World";
10         str3 = new String( original: "Hello, World");
11         str4 = new String( original: "Hello, World");
12
13         System.out.println("Строки 1 и 2 равны? " + (str1 == str2 ));
14         System.out.println("Строки 2 и 3 равны? " + (str2 == str3 ));
15         System.out.println("Строки 3 и 4 равны? " + (str3 == str4 ));
16     }
17 }
```

Результат:

```
Строки 1 и 2 равны? true
Строки 2 и 3 равны? false
Строки 3 и 4 равны? false
```

String в памяти

```
String str1 = "Hello, " + "World";
```



String в памяти

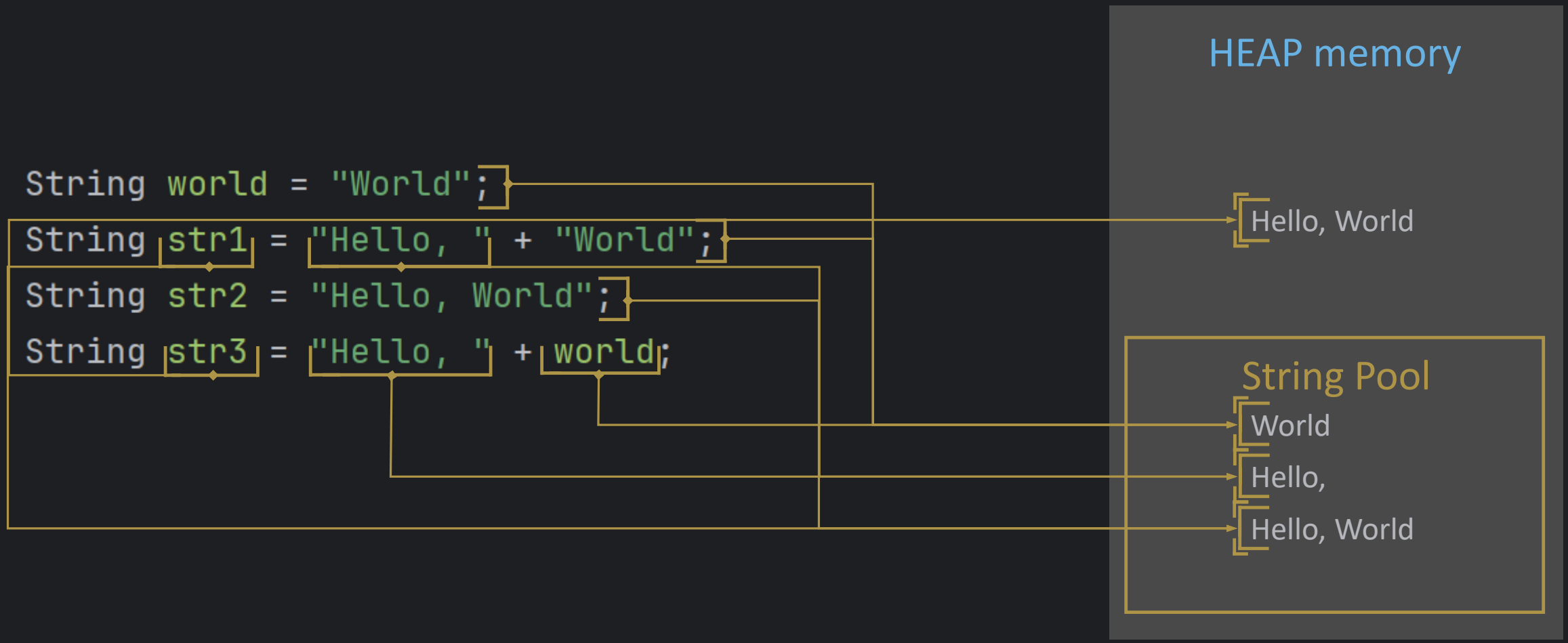
Код:

```
3  public class Main {
4      public static void main(String[] args) {
5
6          String world = "World";
7          String str1 = "Hello, " + "World";
8          String str2 = "Hello, World";
9          String str3 = "Hello, " + world;
10
11         System.out.println("Строки 1 и 2 равны? " + (str1 == str2 ));
12         System.out.println("Строки 2 и 3 равны? " + (str2 == str3 ));
13     }
14 }
```

Результат:

```
Строки 1 и 2 равны? true
Строки 2 и 3 равны? false
```

String в памяти



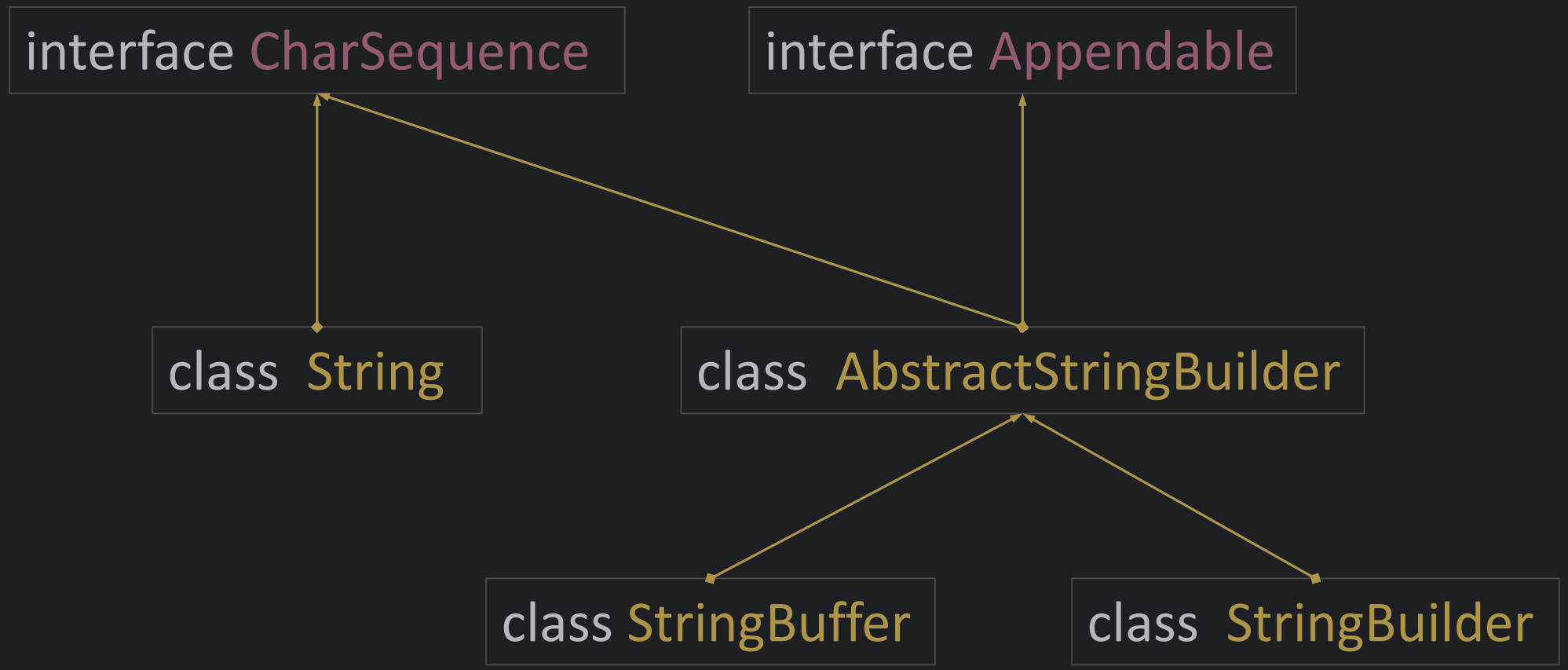
StringBuilder

String
- объекты неизменяемы
- любая операция изменения строки приводит к созданию новой строки
- частые изменения строк сказываются на производительности приложения.

Для решения этой проблемы в Java были добавлены классы StringBuffer и StringBuilder.

- представляют собой изменяемую строку
- оптимизирована работа с памятью
- класс StringBuffer – потокобезопасен, но работает медленнее
- класс StringBuilder – работает быстрее, но не подходит для многопоточной обработки

StringBuilder



Конструкторы StringBuilder

StringBuilder()	- Инициализация пустой строкой
StringBuilder(String str)	- Инициализация строкой из параметра str
StringBuilder(CharSequence chars)	- Инициализация с помощью объекта, реализующего интерфейс CharSequence
StringBuilder(int capacity)	- Инициализация с указанием начального объема памяти через параметр capacity

Управление выделением памяти

`int capacity()`	- Возвращает количество символов, для которых зарезервирована память
`void ensureCapacity(int minimumCapacity)`	- Изменить минимальный объем памяти. `minimumCapacity` – количество символов, под которое нужно выделить память.
`int length()`	- Фактическая длина строки

Код

```
3  public class Main {
4      public static void main(String[] args) {
5          String str = "Jabap";
6          StringBuilder builder = new StringBuilder(str);
7          System.out.println("Емкость: " + builder.capacity());
8          builder.ensureCapacity( minimumCapacity: 32);
9          System.out.println("Емкость: " + builder.capacity());
10         System.out.println("Длина: " + builder.length());
11     }
12 }
```

Результат

```
Емкость: 21
Емкость: 44
Длина: 5
```

Методы StringBuilder

StringBuilder append(Object obj)	- Преобразует переданный объект в строку и добавляет к текущей строке
StringBuilder insert(int index, Object obj)	- Преобразует переданный объект в строку и вставляет в текущую строку со смещением index
StringBuilder replace(int start, int end, String str)	- Заменяет часть строки, заданную интервалом start..end на переданную строку
StringBuilder deleteCharAt(int index)	- Удаляет из строки символ под номером index
StringBuilder delete(int start, int end)	- Удаляет из строки символы, заданные интервалом
int indexOf(String str, int index)	- Ищет подстроку в текущей строке
int lastIndexOf(String str, int index)	- Ищет подстроку в текущей строке с конца
char charAt(int index)	- Возвращает символ строки по его индексу
String substring(int start, int end)	- Возвращает подстроку, заданную интервалом
StringBuilder reverse()	- Разворачивает строку задом наперед.
void setCharAt(int index, char ch)	- Изменяет символ строки, заданный индексом на переданный

Сравнение скорости работы

```
private static final int NUM_STRINGS = 300000;
```

Код

```
private static String stringConcatenation() {
    String string = "";
    for (int i = 0; i < NUM_STRINGS; i++) {
        string += getSubstring(i);
    }
    return string;
}
```

Результат

```
String - 189597мс.
```

Код

```
private static String builderConcatenation() {
    StringBuilder builder = new StringBuilder();
    for (int i = 0; i < NUM_STRINGS; i++) {
        builder.append(getSubstring(i));
    }
    return builder.toString();
}
```

Результат

```
StringBuilder - 15мс.
```

Задачи для самостоятельной работы

1. Напишите программу, которая заменяет все пробелы в строке на символ подчеркивания (_).
Пример:
Ввод: «В абапе строки проще».
Вывод: «В_абапе_строки_проще».

2. Напишите программу, которая подсчитывает количество вхождений каждого символа в строке.
Пример:
Ввод: «abap».
Вывод: «a – 2, b – 1, p -1».

3. Напишите программу, которая удаляет повторяющиеся символы из строки.
Пример:
Ввод: «aabbcc».
Вывод: «abc».

4. Напишите программу, которая принимает строку с именем и возрастом, а затем форматирует её в виде: «Имя: [имя], Возраст: [возраст]».
Пример:
Ввод: «Петя 25».
Вывод: «Имя: Петя, Возраст: 25».

5. Напишите программу, которая подсчитывает количество слов в тексте.
Пример:
Ввод: «В абапе строки проще».
Вывод: «Количество слов: 4».

6. Напишите программу, которая удаляет все HTML-теги из строки.
Пример:
Ввод: « Hello ».
Вывод: «Hello».