**STRINGS**

1. Создать объект класса String можно с помощью ключевого слова *new*, либо при использовании конструктора, в котором можно передавать массив байтов, символов, цифр или присваивать строковое значение.

**Строковый литерал –** набор символов, помещенных в двойные кавычки.

**Упрощенное создание объекта String –** прямое создание объекта String с присваиванием строкового литерала.

1. Объект типа String является неизменяемым.

Каждый раз при изменении состояния объекта String создается новый объект.

Наследоваться от класса String нельзя, так как класс объявлен как *final*.

Строковые переменные используются как ключи/индексы, поэтому они неизменяемы, это используется для предотвращения изменения значения объекта, при условии, что на него ссылаются в разных частях кода.

1. Кодировка — это соответствие между визуальными символами и числами.

UTF-8, UTF-16, ASCII, Windows-1251.

String использует кодировку UTF-16, массив byte[] позволяет изменять кодировку.

1. Пул литералов – набор литералов, хранящийся в памяти java heap.

Строка заносится в пул при создании объекта класса String с помощью ключевого слова *new.*

При вызове метода intern() возвращается ссылка на строку из пула,

если такая строка уже существует.

Пул литералов в Java 7 хранится в куче (heap). В Java 6

пулы литералов хранились в PermGen — Permanent Generation –

область памяти в JVM, предназначенная для хранения описания классов Java и некоторых дополнительных данных.

1. Объекты класса StringBuffer и StringBuilder является изменяемыми, в отличии от объекта класса String.

StringBuffer является потокобезопасным.

Сравнение можно реализовать через методы .toString, которые приводят объект StringBuffer(StringBuilder) к типу String, а далее .equals, который сравнивает строки по значению

Sb1.toString.equals(sb2.toString)

Также сравнение объектов StringBuilder можно реализовать через метод .compareTo, значение которого должно равняться нулю

Sb1.compareTo(sb2) == 0

1. Unicode - стандарт кодирования символов, включающий в себя знаки почти всех письменных языков мира.
2. Кодовые единицы типа char представляют кодовые точки Unicode в кодировке UTF-16. Методы класса String для работы: codePointCount(int beginIndex, int endIndex), offsetByCodePoint(0, i), codePointAt(i).

**REGEX**

1. Регулярное выражение – строка-шаблон, позволяющая работать со строками.

Метасимволы регулярного выражения – специальные символы, имеющие определенные значения

\\b – граница слова, [\\s](file:///\\s) – пробел, ^ - начало строки, $ - конец строки, . – любой символ, [\\w](file:///\\w) – любой буквенно-цифровой элемент и т.д.

Квантификаторы – символы, обозначающие количество символов

\*- 0 или больше, + - 1 или больше, ? – один или ни одного

Специальные обозначения начала строки (^) и ее конца ($)

1. Pattern, Matcher.

Пакет – java.util.regex

String text = “Hello world!”; //строка, в которой будет

Pattern pattern = Pattern.compile(“\w{4,}”); //создание шаблона для анализа текста

Matcher matcher = pattern.matcher(text); //создание объекта класса Matcher с указание шаблона и текста для анализа

While (matcher.find()) System.out.println(matcher.group()); //метод, который выводит в консоль все совпадения в тексте согласно шаблону

1. Группа – пронумерованная часть регулярного выражения, обозначенная в круглых скобках

Нумеруются группы слева направо от 1, если одна группа входит в состав другой, то внешняя группа будет предшествовать по номеру внутренней.

Группа 0 относится ко всему выражению

Pattern email = Pattern.compile("\\w+@(gmail|yandex)\\.(com|ru)"); // группа 1 – группа указания домена 2го уровня почты, группа 2 – указание домена 1го уровня почты, группа 0 – все выражение целиком.