Вопросы и ответы по теме
Strings and basics of text processing

Strings

1. Как создать объект класса String, какие конструкторы класса String вы знаете? Что такое строковой литерал? Объясните, что значит «упрощенное создание объекта String»?

Основные конструкторы: String() – создает пустую строку; String(char[] value) – создает строку из массива символов; String(byte[] bytes) – создает строку из массива байт, преобразуя байты в символы в соответствии с кодировкой по умолчанию.

По общим правилам создания объектов объект класса String создается так String str = new String("какая-то строка"). Но как правило, при создании данного объекта используется упрощенный вариант String str = "какая-то строка".

Строковые литералы — это набор символов, заключенных в двойные кавычки.

2. Можно ли изменить состояние объекта типа String? Что происходит при попытке изменения состояния объекта типа String? Как вы думаете, почему строковые объекты Immuteble?

Объекты String неизменяемые. При попытке изменения состояния объекта типа String, каждый раз возвращается новый объект.

Строковые объекты сделали Immuteble: в целях безопасности (строки активно используются при загрузке классов, в сетевых соединениях, соединениях с базой данных); в многопоточности отсутствуют сложности в синхронизации; для поддержки пула строк; вычисление хэшкода один раз.

3. Объясните, что такое кодировка? Какие кодировки вы знаете? Как создать строки в различной кодировки?

Кодировка это просто таблица, где каждой букве соответствует число. Поэтому кодировка имеет смысл только в том случае, когда массив чисел byte[] нам нужно превратить в строку и нужно понимать с какой буквой ассоциировать каждое число. Если короче, то кодировкой называется набор символов и соответствующий им набор кодов.

Существует много кодировок, основные из них, которые часто используются это: UTF-8, UTF-16, Unicode, KOI-8 в Windows-1251.

Получить список всех кодировок, с которыми Java может работать, можно при использовании специального статистического метода availableCharsets() (SortedMap<String,Charset> charsets = Charset.availableCharsets();) Этот метод возвращает набор пар (имя кодировки, объект описывающий кодировку).

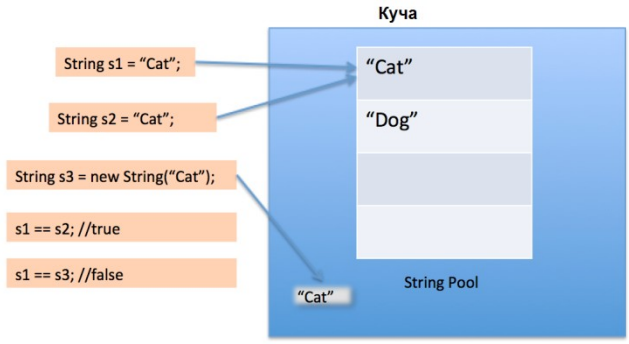
Создаем строку кодировки UTF-8:

```
String s = "Какой-то текст";
byte[] buffer = s.getBytes("UTF-8");
String str3 = new String(buffer3, "UTF-8");
```

4. Что такое пул литералов? Как строки заносятся в пул литералов? Как занести строку в пул литералов и как получить ссылку на строку, хранящуюся в пуле литералов? Где хранятся (в каком типе памяти) пул литералов в Java 1.6 и Java 1.7?

Пул строк (**String Pool**) — это множество строк в кучи (Java Heap Memory). Сам строковый пул возможен только потому, что строки в Java неизменные. Также пул строк позволяет сохранить память в Java Runtime, хотя это и требует больше времени на создание самой строки.

Когда мы используем двойные кавычки, чтобы создать новую строку, то первым делом идет поиск строки с таким же значением в пуле строк. Если java такую строку нашла, то возвращает ссылку, в противном случае создается новая строка в пуле, а затем возвращается ссылка. Однако использование оператора new заставляет класс String создать новый объект String. После этого можем использовать метод intern(), чтобы поместить этот объект в пул строк или обратиться к другому объекту из пула строк, который имеет такое же значение.



5. В чем отличие объектов классов StringBuilder и StringBuffer от объектов класса String? Какой из этих классов потокобезопасный? Как необходимо сравнивать на равенство объекты классов StringBuilder и StringBuffer и почему?

Строка является неизменной и финализированной в Java, поэтому все наши манипуляции со строкой всегда будут создавать новую строку. Манипуляции со строками ресурсоемкие, поэтому Java обеспечивает два полезных класса для манипуляций со строками – StringBuffer и StringBuilder. StringBuffer и StringBuilder являются изменяемыми классами. Операции с StringBuffer потокобезопасны и синхронизированы, а методы StringBuilder не потокобезопасны. Поэтому когда несколько нитей работают с одной строкой, мы должны использовать StringBuffer, но в однопоточном окужении мы должны использовать StringBuilder. StringBuilder более производительный, чем StringBuffer, поскольку не обременен синронизацией.

Внутри класса String метод equals переопределён, поэтому можно сравнивать строки по значению с помощью **equals**, а в классе StringBuilder и StringBuffer нет. Поэтому по значению проверяют так:

sb.toString().equals(sb2.toString()),т.е. приводят StringBuilder (StringBuffer) к сторке (метод toString) и вызывают метод equals.

6. Что такое Unicode?

Unicode – это стандарт кодирования символов, включающий в себя знаки почти всех письменных языков мира. Unicode постулирует чёткое разграничение между символами, их представлением в компьютере и их отображением на устройстве вывода.

7. Какие методы класса String используются для работы с кодовыми точками? Как вы думаете, когда следует их использовать?

В языке Java строки реализованы как последовательности значений типа **char**. Тип char позволяет задавать кодовые единицы, представляющие кодовые точки Unicode в кодировке UTF-16. Наиболее часто используемые символы Unicode представляются одной кодовой единицей. Дополнительные символы задаются парами кодовых единиц.

Метод length() возвращает количество кодовых единиц для данной строки в кодировке UTF-16.

Чтобы определить реальную длину, представляющую собой число кодовых точек, надо использовать следующий вызов: int cpCount = greeting.codePointCount(0, greeting.length());

Метод s.charAt(n) возвращает кодовую единицу в позиции n, где n находится в интервале от 0 до s.length() - 1.

Метод codePointAt() определяет, является ли кодовая единица первой или второй частью дополнительного символа, и всегда возвращает корректный результат.

```
String emojiString = "";
emojiString.codePoints().count()); //1
emojiString.chars().count()); //2
```

Regular Expression

1. Расскажи, что представляет собой регулярное выражение? Что такое метасимволы регулярного выражения? Какие вы знаете классы символов регулярных выражений? Что такое квантификаторы? Какие логические операторы регулярных выражений вы знаете? Что значит «якорь» для регулярного выражения?

Регулярное выражение – это шаблон для поиска строки в тексте. В Java исходным представлением этого шаблона всегда является строка, то есть объект класса String. Однако не любая строка может быть скомпилирована в регулярное выражение, а только та, которая соответствует правилам написания регулярного выражения – синтаксису, определенному в спецификации языка.

Для написания регулярного выражения используются буквенные и цифровые символы, а также метасимволы – символы, имеющие специальное значение в синтаксисе регулярных выражений. Например:

String regex=«\\d{3}»; // шаблон строки из трех цифровых символов;

Синтаксис регулярных выражений основан на использовании символов <([{\^-=$!|]})?*+.>, которые можно комбинировать с буквенными символами. В зависимости от роли их можно разделить на следующие группы:

1. Метасимволы для поиска совпадений границ строк или текста;
2. Метасимволы для поиска символьных классов;
3. Метасимволы для поиска символов редактирования текста;
4. Метасимволы для группировки символов;
5. Метасимволы для обозначения количества символов – квантификаторы. Квантификатор всегда следует после символа или группы символов.

Квантификаторы - это спецсимволы, которые ставятся после класса. Они устанавливают повторение класса. При этом класс находит не один символ, а несколько. Практически он находит строку. Каджый символ в этой строке должен соответствовать классу.

Чаще всего используется квантификатор "+", который устанавливает любое количество повторений. Он работает так: если класс находит подходящий символ, он переходит к следующему символу. Если и он соответствует классу, то он добавляется в найденную подстроку. Затем проверяется следующий символ и так до тех пор, пока проверка не дойдёт до символа, не соответствующего классу. Тогда класс заканчивает работу и выполняется следующая часть паттерна.

Особенностью квантификаторов является возможность использования их в разных режимах: жадном, сверхжадном и ленивом. Сверхжадный режим включается добавлением символа «+» после квантификатора, а ленивый – символа «?». Например:

"A.+a" // жадный режим

"A.++a" // сверхжадный режим

"A.+?a" // ленивый режим

Классы символов:

Конструкция	Описание
[abc]	a, b, или c (простой класс)
[^abc]	Любой символ, кроме a, b и c (отрицание)
[a-zA-Z]	от a до z, или от A до Z, включительно (диапазон)
[a-d[m-p]]	от a до d, или от m до p: [a-dm-p] (объединение)
[a-z&&[def]]	d, e, или f (пересечение)
[a-z&&[^bc]]	от a до z, исключая b и c: [ad-z] (вычитание)
[a-z&&[^m-p]]	от a до z, и не от m до p: [a-lq-z] (вычитание)

java.util.regex.Pattern содержит несколько предопределённых классов символов:

Конструкция	Описание
.	Любой символ (может совпадать и с символами конца строки)
\d	Цифра: [0-9]
\D	Не цифра: [^0-9]
\s	Пробельный символ: [\t\n\x0B\f\r]
\S	Непробельный символ: [^\s]
\w	Символ слова (английская буква, подчёркивание или цифра): [a-zA-Z_0-9]
\W	Несловарный символ: [^\w]

Якорь – определяет позицию шаблона в строке текста:

^ – якорь, определяющий начало строки;

$ – якорь, определяющий конец строки.

2. Какие java-классы работают с регулярными выражениями? В каком пакете они расположены? Приведите пример анализа текста с помощью регулярного выражения и поясните код примера.

Некоторые методы класса String принимают регулярные выражения и используют их для выполнения операций над строками.

Для разделения строки на подстроки применяется метод **split()**. В качестве параметра он может принимать регулярное выражение, которое представляет критерий разделения строки.

Например, разделим предложение на слова:

```
String text = "FIFA will never regret it";
String[] words = text.split("\\s*(\\s|,|!|\\.)\\s*");
```

Еще один метод класса String - matches() принимает регулярное выражение и возвращает true, если строка соответствует этому выражению. Иначе возвращает false.

Например, проверим, соответствует ли строка номеру телефона:

```
String input = "+12343454556";
boolean result = input.matches("(\\+*)\\d{11}");
```

Класс Pattern

Большая часть функциональности по работе с регулярными выражениями в Java сосредоточена в пакете java.util.regex.

Само регулярное выражение представляет шаблон для поиска совпадений в строке. Для задания подобного шаблона и поиска подстрок в строке, которые удовлетворяют данному шаблону, в Java определены классы Pattern и Matcher.

Для простого поиска соответствий в классе Pattern определен статический метод boolean matches(String pattern, CharSequence input). Данный метод

возвращает true, если последовательность символов input полностью соответствует шаблону строки pattern:

```
String input = "Hello";
        boolean found = Pattern.matches("Hello", input);
        if(found)
           System.out.println("Найдено");
        else
           System.out.println("Не найдено");
```

Класс Matcher

Рассмотрим основные методы класса Matcher:

boolean matches(): возвращает true, если вся строка совпадает с шаблоном

boolean find(): возвращает true, если в строке есть подстрока, которая совпадает с шаблоном, и переходит к этой подстроке

String group(): возвращает подстроку, которая совпала с шаблоном в результате вызова метода find. Если совпадение отсутствует, то метод **генерирует исключение IllegalStateException.**

int start(): возвращает индекс текущего совпадения

int end(): возвращает индекс следующего совпадения после текущего

String replaceAll(String str): заменяет все найденные совпадения подстрокой str и возвращает измененную строку с учетом замен

Используем класс Matcher. Для этого вначале надо создать объект Pattern с помощью статического метода compile(), который позволяет установить шаблон:

Pattern pattern = Pattern.compile("Hello");

В качестве шаблона выступает строка "Hello". Метод compile() возвращает объект Pattern, который мы затем можем использовать в программе.

В классе Pattern также определен метод matcher(String input), который в качестве параметра принимает строку, где надо проводить поиск, и возвращает объект Matcher:

```
String input = "Hello world! Hello Java!";
Pattern pattern = Pattern.compile("hello");
Matcher matcher = pattern.matcher(input);
```

3. Что такое группы в регулярных выражениях? Как нумеруются группы? Что представляет собой группа номер 0(ноль)? Приведите пример с использованием групп регулярного выражения.

Группы нумеруются путем подсчета их открывающих скобок слева направо. В выражении ((A) (B (C))), например, существует четыре такие группы: 1-я ((A)(Б(C))) 2-я (A) 3-я (B (C)) 4-я (C). Пример: (([a-z]+)\s*([^>]*)).

Нулевая группа всегда обозначает все выражение.