**Strings**

1. **Как создать объект класса String, какие конструкторы класса String вы знаете? Что такое строковый литерал? Объясните, что значит “упрощенное создание объекта String”?**

* Самый простой способ создания строки (присвоив ему значение строкового литерала) выглядит так:

String greeting = "ввв ввв ввввв!";

В данном случае «ввв ввв ввввв» — это строковый литерал (т.е. константа) состоящая из последовательности символов, заключенных в кавычки. Всякий раз, когда компилятор сталкивается со строковым литералом, он создает объект типа String со значением. Как и любой другой объект, строку можно создать, используя ключевое слово new.

Можно создать массив строк:

String[] cats = {"Васька", "Барсик", "Мурзик"};

Можно создать пустой объект класса String:

String str = new String();

Можно создать строку через массив символов:

сhar[] chars = { 'c', 'a', 't' };

String str = new String(chars);

**упрощенное создание объекта String?????**

1. **Можно ли изменить состояние объекта типа String? Что происходит при попытке изменения состояния объекта типа String? Можно ли наследоваться от класса String?**

**Как вы думаете, почему строковые объекты immutable??**

* Класс String — immutable класс, то есть его объекты не могут быть изменены после создания. Любые операции над объектом String, результатом которых должен быть объект класса String, приведут к созданию нового объекта.

Класс String в Java — это final класс, который не может иметь потомков.

Благодаря своей неизменности, объекты класса String являются потокобезопасными и могут быть использованы в многопоточной среде. Если строка будет изменяемой, тогда это станет серьезной угрозой безопасности приложения.

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

* Кодировкой называют способ представления в памяти компьютера цифр, буков и всех остальных знаков. Каждой букве или каждому символу соответствует какое-то число. Кодировкой называется набор символов и соответствующий им набор кодов. Именно поэтому тип char считается не только символьным, но и числовым типом.

Существуют кодировки UTF-8, UTF-16, Windows-1251, KOI8-R,… и UNIcode.

На основе строки можно создать массив байт в любой известной Java кодировке:

String s = "Good news everyone!";

byte[] buffer = s.getBytes("Windows-1251");

или

String s = "Good news everyone!";

Charset koi8 = Charset.forName("KOI8-R");

byte[] buffer = s.getBytes(koi8);

Преобразовать набор байт в строку:

byte[] buffer = new byte[1000];

inputStream.read(buffer);

String s = new String(buffer, "Windows-1251");

1. **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 хранится(в каком типе памяти) пул литералов в Java 1.6 и Java 1.7?**

* Пул литералов – это набор строк, который хранится в памяти heap (Java1.7) и perm (Java 1.6). Пул литералов\строк возможен исключительно благодаря неизменяемости строк в Java. Когда мы используем двойные кавычки для создания строки, сначала ищется строка в пуле с таким же значением, если находится, то просто возвращается ссылка, иначе создается новая строка в пуле, а затем возвращается ссылка. Пул строк помогает экономить большой объем памяти, но с другой стороны создание строки занимает больше времени.

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

* StringBuilder и StringBuffer – это аналог класса String, только изменяемый. Разница в том, что StringBuffer потокобезопасен, и все его методы синхронизированы, а StringBuilder — нет. Это единственная особенность. StringBuilder в Java работает быстрее StringBuffer’а благодаря несинхронизированности методов.

String str = new String("карамба");  
StringBuilder str1 = new StringBuilder(str);  
StringBuilder str2 = new StringBuilder(str);  
String t1 = str1.toString();  
String t2 = str2.toString();  
  
System.*out*.println(t1.equals(t2));

1. **Что такое Unicode?**

* Это стандарт, а не кодировка — сам по себе Юникод не определяет, как символы будут сохранятся на жестком диске или передаваться по сети. Он лишь определяет связь между символом и некоторым числом, а формат, согласно с которым эти числа будут превращаться в байты, определяется Юникод-кодировками (например, UTF-8 или UTF-16).

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

* codePointCount - возвращает количество кодовых единиц для данной строки в кодировке UTF-16.

1. int cpCount = str.codePointCount(0, str.length());

* codePointAt() - возвращает кодовую единицу в позиции n, где n находится в интервале от 0 до s.length() – 1,
* int cp = str.codePointAt(3);

**Regular Expressions**

1. **Расскажите, 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 (regular expression/regex/regexp) — это строка, которая является шаблоном (pattern), описывающим некий набор строк. Шаблон определяет, какие именно строки относятся к набору. Шаблон состоит из литералов и метасимволов — символов со специальным, а не буквальным значением. Поиск по шаблону –— это поиск по тексту для нахождения совпадений, то есть строк, соответствующих шаблону регулярного выражения.

Существуют простые классы символов, диапазонные классы, инверсию, объединение, пересечение и вычитание классов.

Простой класс символов состоит из размещенных бок о бок символов и задает соответствие только этим символам. Например, класс [abc] соответствует символам a,b и c.

Инвертированный класс символов начинается с метасимвола ^ и задает соответствие только тем символам, которые в нём не содержатся. Например, класс [^abc] соответствует всем символам, кроме a,b и c.

Диапазонный класс символов состоит из двух символов, разделенных символом дефиса (-). Все символы, начиная с символа слева от дефиса и заканчивая символом справа, относятся к диапазону. Например, диапазон [a-z] соответствует всем латинским буквам в нижнем регистре.

Объединение классов символов состоит из нескольких вложенных классов символов и соответствует всем входящим в результирующее объединение символам. Например, класс [a-d[m-p]] соответствует символам от a до d и от m до p.

Пересечение классов символов состоит из символов, общих для всех вложенных классов и соответствует только общим символам. Например, класс [a-z&&[d-f]] соответствует символам d, e и f.

Вычитание классов символов состоит из всех символов, кроме тех, которые содержатся во вложенных классах символов, и соответствует только этим остальным символам. Например, класс [a-z&&[^m-p]] соответствует символам от a до l и от q до z:

Некоторые классы символов встречаются в регулярных выражениях достаточно часто, чтобы оправдать использование сокращенных обозначений.

\d: Цифра. Эквивалентно [0-9].

\D: Нецифровой символ. Эквивалентно [^0-9].

\s: Пробельный символ. Эквивалентно [ \t\n\x0B\f\r].

\S: Не пробельный символ. Эквивалентно [^\s].

\w: Словообразующий символ. Эквивалентно [a-zA-Z\_0-9].

\W: Не словообразующий символ. Эквивалентно [^\w].

Иногда бывает нужно выполнить сопоставление с шаблоном в начале строки, на границе слов, в конце текста и т.д. Сделать это можно с помощью одного из граничных сопоставителей:

^: Начало строки;

$: Конец строки;

\b: Граница слова;

\B: Граница псевдослова;

\A: Начало текста;

\G: Конец предыдущего совпадения;

\Z: Конец текста, не считая итогового разделителя строк (если таковой присутствует);

\z: Конец текста

Квантификатор – это конструкция регулярного выражения, явно или неявно связывающая шаблон с числовым значением. Этот числовое значение определяет, сколько раз искать шаблон. Квантификаторы делятся на жадные, ленивые и сверхжадные:

Жадный квантификатор (?, \* или +) предназначен для поиска самого длинного совпадения.

Сверхжадный квантификатор (?+, \*+ или ++) аналогичен жадному, за исключением того, что сверхжадный квантификатор выполняет только одну попытку найти самое длинное совпадение, в то время как жадный может выполнять несколько попыток.

Логический оператор ИЛИ часто используется в регулярных выражениях

X|Y Либо X, либо Y.

У символов каретки ^ и доллара $ есть специальные значения в регулярных выражениях. Они называются «якоря».

Каретка ^ означает совпадение с началом текста, а доллар $ – с концом. (^Привет пока$) - точное совпадение (начинается и заканчивается как Привет пока).

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

* - Pattern, Matcher и PatternSyntaxException – три класса, составляющих API Regex. Каждый из них предоставляет методы, позволяющие использовать регулярные выражения в коде. Все они находятся в пакете пакета java.util.regex.
* Pattern - скомпилированное представление регулярного выражения. Для компиляции нужно вызвать статический метод compile.
* Matcher - движок, который производит операцию сравнения (match).
* PatternSyntaxException - указывает на синтаксическую ошибку в выражении**.**

Pattern pattern1 = Pattern.*compile* ("[x**-**z]+");//Поиск будет происходить от x до z включительно.  
//Поиск будет происходить для совпадений более одного раза и только по символам нижнего регистра.  
  
Matcher matcher1 = pattern1.matcher ("x y z 1 2 3 4 ");  
System.*out*.println (matcher1.find()); //Поиск любого совпадения с шаблоном.  
//Выводится значение true, так как в строке есть символы шаблона.

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

* Группа - это пронумерованная часть регулярного выражения. Например, в следующем выражении
* (\d+)zzz
* присутствуют две группы. Группа 0 всегда относится ко всему выражению, а группа 1 - к подвыражению, начинающемуся с открывающей круглой скобки "(" и заканчивающемуся закрывающей круглой скобкой ")".
* String str00 = "Достала учеба! Ничерта не понимаю. Перечитываю по 1000 раз! я - тупень!";  
    
  Pattern pattern = Pattern.*compile*("(.\*?)(\\d+)(.\*)"); // 3 группы для поиска.  
    
  Matcher m = pattern.matcher(str00);  
  if (m.find()) {  
   System.*out*.println(m.group(0)); // Сначало выражение жадно читает абсолютно все символы, коих может быть от 0 до беск.  
   System.*out*.println(m.group(1)); // 1 группа возвращает значения всех символов от 0 до беск. до действия условия 2 группы т.е. до цифр  
   System.*out*.println(m.group(2)); // Выдает цифры от 1 до беск.  
   System.*out*.println(m.group(3)); // выдает все символы после условия второй группы т.е. все после 1000.  
  }