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

Строка представляет собой последовательность символов. Для работы со строками в Java определен класс String, который предоставляет ряд методов для манипуляции строками. Физически объект String представляет собой ссылку на область в памяти, в которой размещены символы.

Строковые литералы — это набор символов, заключенных в двойные кавычки. Данный тип используется так же часто, как и числовые литералы. В строке также могут находится служебные символы, которые необходимо экранировать (так называемые escape-последовательности).

Для создания новой строки мы можем использовать один из конструкторов класса String, либо напрямую присвоить строку в двойных кавычках:

```
String str1 = "Java";
String str2 = new String(); // пустая строка
String str3 = new String(new char[] {'h', 'e', 'l', 'l', 'o'});
String str4 = new String(new char[]{'w', 'e', 'l', 'c', 'o', 'm', 'e'}, 3, 4);//3 -начальный индекс, 4 - кол-во символов
```

2. 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?

Нельзя изменить. Нет возможности его поменять. Наследоваться нельзя. Для безопасности.

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

Кодировка – соответствие определенному коду определенного символа. UTF-8, ASCII. Использовать конструкторы, принимающие аргументом кодировку.

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 хранится(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). Мы знаем, что String — особый класс в java, с помощью которого мы можем создавать строковые объекты.

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?

Все строковые классы – final (следовательно от них нельзя унаследоваться).

String

Строка — объект, что представляет последовательность символов. Для создания и манипулирования строками Java платформа предоставляет общедоступный финальный (не может иметь подклассов) класс java.lang.String. Данный класс является неизменяемым (immutable) — созданный объект класса String не может быть изменен.

StringBuffer

Строки являются неизменными, поэтому частая их модификация приводит к созданию новых объектов, что в свою очередь расходует драгоценную память. Для решения этой проблемы был создан класс java.lang.StringBuffer, который позволяет более эффективно работать над модификацией строки. Класс является mutable, то есть изменяемым — используйте его, если Вы хотите изменять содержимое строки. StringBuffer может быть использован в многопоточных средах, так как необходимые методы являются синхронизированными.

StringBuilder

StringBuilder — класс, что представляет изменяемую последовательность символов. Класс был введен в Java 5 и имеет полностью идентичный API с StringBuffer. Единственное отличие — StringBuilder не синхронизирован. Это означает, что его использование в многопоточных средах нежелательно. Следовательно, если вы работаете с многопоточностью, вам идеально подходит StringBuffer, иначе используйте StringBuilder, который работает намного быстрее в большинстве реализаций.

Краткий ответ :

String неизменяемый, StringBuilder и StringBuffer изменяемые. StringBuffer потокобезопасен и синхронизирован, а StringBuilder нет, но он быстрее. Нужно преобразовать StringBuffer и StringBuilder в String, а потом сравнивать, т.к. у них не переопределен метод equals().

6. Что такое Unicode?

Это стандарт кодирования символов, включающий почти все письменные символы.

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. Ниже приведен пример использования данного метода:

```
String greeting = "Hello";
int n = greeting.length(); // Значение n равно 5.
```

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.

Ниже приведены примеры вызова данного метода:

```
char first = greeting.charAt(0); // первый символ - 'H'
char last = greeting.charAt(4); // последний символ - 'o'
```

Для получения i-й кодовой точки надо использовать приведенные ниже выражения:

```
int index = greeting.offsetByCodePoints(0, i);
int cp = greeting.codePointAt(index);
```

Java подсчитывает кодовые единицы в строках специфическим образом: первая кодовая единица в строке расположена в позиции 0. Это соглашение пришло из языка C, где по техническим причинами подсчет позиций начинается с 0.

Эта причина давно ушла в прошлое, а неудобство осталось. Однако этому соглашению следует настолько много программистов, что проектировщики Java решили сохранить его.

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?

Регулярные выражениия (англ. regular expressions) — формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов (символов-джокеров, англ. wildcard characters). Для поиска используется строка-образец (англ. pattern, по-русски её часто называют «шаблоном», «маской»), состоящая из символов и метасимволов и задающая правило поиска. Для манипуляций с текстом дополнительно задаётся строка замены, которая также может содержать в себе специальные символы.
Подробнее:
http://website-lab.ru/article/regexp/shpargalka_po_regulyarnyim_vyirajeniyam/
https://javarush.ru/groups/posts/regulyarnye-vyrazheniya-v-java

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.

С регулярными выражениями работают классы java.util.regex.Matcher, java.util.regex.Pattern.

```
Pattern p = Pattern.compile("ab");
Matcher m = p.matcher("abasdfabasdfabdsfaab");
int count = 0;
while(m.find()) {
   count++;
}
```

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 представляют собой несколько регулярных выражений, каждое из которых заключено в круглые скобки. Нумерация групп идёт с нуля, группа номер 0 это то же регулярное выражение которое содержится во всех группах не разделённое на группы.

```
String input = "java8"; // определяем анализируемую строку
String regex = "([a-z]*)([0-9]*)"; // определяем паттерны по группам, в данном случае группа 1 ищет все буквы a-z, а группа 2 - цифры
Pattern pattern = Pattern.compile(regex); // создаем объект типа Pattern
```

```
Matcher matcher = pattern.matcher(input); // Создаем Matcher  и задаем анализируемую строку
if(matcher.matches()) { // если есть совпадения с паттерном
		System.out.println("main group: " + matcher.group(0));// печатаем результаты по группам
		System.out.println("group 1: " + matcher.group(1));
		System.out.println("group 2: " + matcher.group(2));
}
else {
		System.out.println("nothing");
}
```