**Java.lang.String – представляет строку в формате UTF-16. Является immutable объектом. Константные выражения с типом String всегда интернированы чтобы разделять уникальное значение между собой. Объект String можно интернировать использую String.intern().**

**Строковый литерал – ссылка на объект String(интернированный объект String).**

**Строки которые были concat(“+”) на стадии compile – интернированы, тогда как строки concat(“+”) на стадии run time – нет.**

* **String(char [] str[byte[] b, StringBuilder, StringBuffer, String]) – конструкторы**
* **contains (CharSequence s) –**
* **getBytes**
* **indexOf(string) - индекс первого вхождения**
* **char charAt(index)**
* **intern ()**
* **Strieam<String> lines () – возвращает стрим стрингов**
* **Replace (target, replacement)**
* **split(“srring”)**
* **String substring (begin, end)**
* **trim ()**
* **getChars(begin, end, [] buffer, beginBuffer)**
* **String.valueOf(примитивы) – чтобы получить строковое представление**

**Java.lang.StringBuilder – mutable последовательность символов, без гарантии synchronization, быстрее чем StringBuffer потому что отсутствует синхронизация.**

**StringBuildes([String]) – для реверса можно использовать**

**append() -**

**insert() - sb.append(“x”) has same effect as sb.insert(sb.lenght(), “x”)**

**Delete()**

**getChars(begin, end, [] buffer, beginBuffer)**

**reverse() –**

**length() –**

**toString – возвращает строку**

**java.lang.StringBuffer – thread-safe аналог StringBuilder**

**Object :**

* **Clone() – создает копию объекта. Класс должен реализовывать интерфейс Cloneable, иначе CloneNotSupportedException. Not deep copy.**
* **Equals() – если не переопределять, то (==).**

**Запись по все примитивы кроме long, double происходит атомарно.**

**С long и double это не работает, потому что некоторые машины не могу обработать 64-бита сразу.**

**Если установить переменную как volataile, то запись будет атомарна.**

**Java happens-before:**

* **Освобождение мьютекса *happens before* происходит раньше захвата этого же монитора другим потоком.**
* **Метод Thread.start() *happens before* Thread.run().**
* **Завершение метода run() *happens before* выход из метода join().**
* **Запись в volatile переменную *happens-before* чтению из той же переменной.**

**Можно получить синхронизацию над объектом Class выполнив статический synchronized метод.**

* **Finalize () – вызывается сборщиком мусора. Когда будет вызван непонятно. В новых версиях деприкейтид.**
* **getClass () – переопределять нельзя. Возвращает объект Class**
* **hashCode () – если не переопределять, то возвращает memory address объекта в виде hexadecimal числа. По определению, если 2 объекта equals, то их hashCode обязаны быть одинаковыми. Т.е переопределять вместе. Обратно неверно.**
* **toString() -**

**Instanceof – A instanceof B, если B Не null, и существует cast не выдающий ClassCastException при (B) A – касте А к B.**

**Collectors:**

**toList ()**

**joining ()**

**joining (StringDelimiter)**

**Из массива в ArrayList: Arrays.asList()**