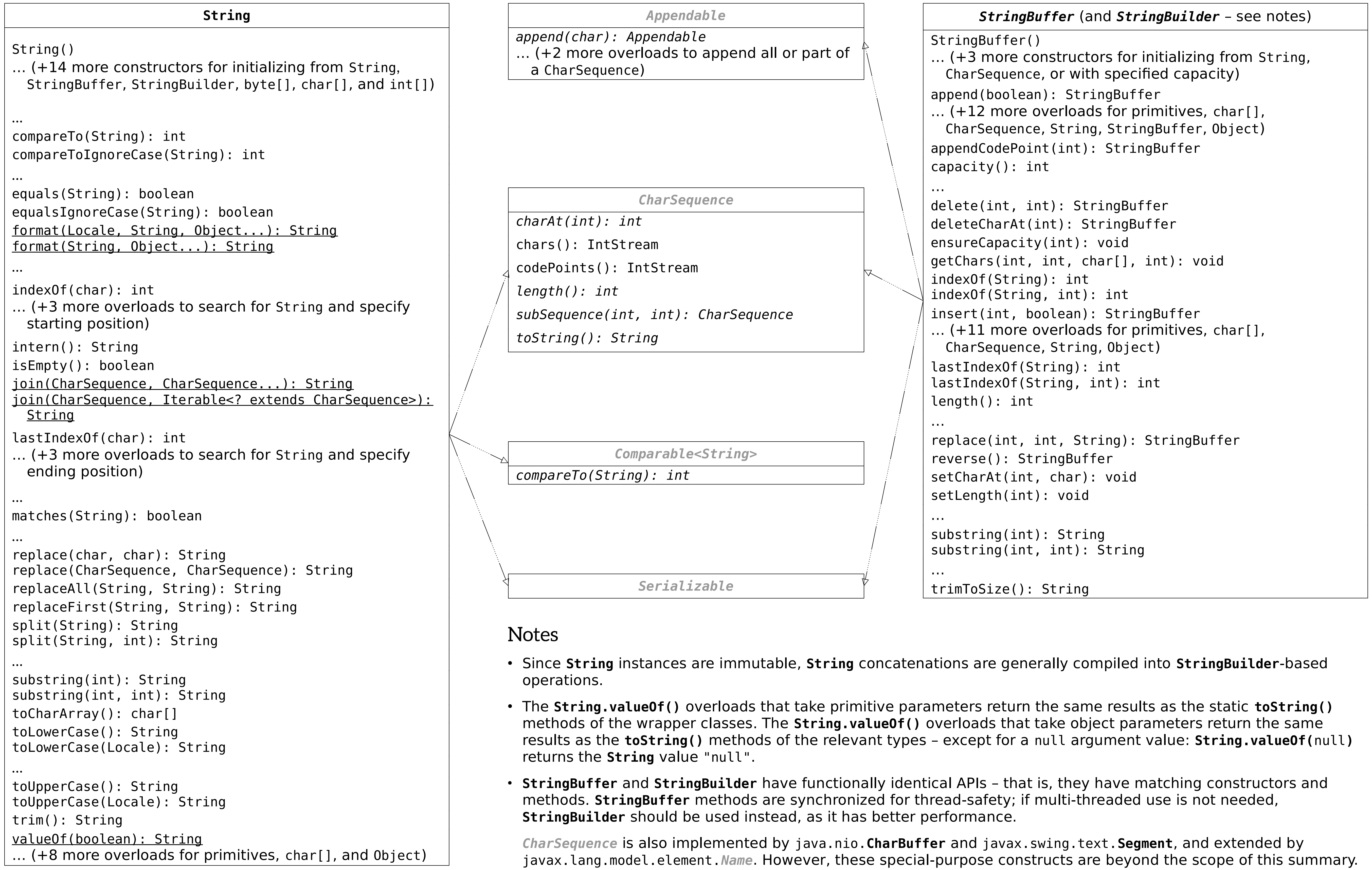
Java Strings

Key `String`-related *`interfaces`*, **`classes`**, and `methods` in `java.lang`

DEEP DIVE Coding Bootcamps



String

```
String()
… (+14 more constructors for initializing from String,
  StringBuffer, StringBuilder, byte[], char[], and int[])

…
compareTo(String): int
compareToIgnoreCase(String): int
…
equals(String): boolean
equalsIgnoreCase(String): boolean
format(Locale, String, Object...): String
format(String, Object...): String
…
indexOf(char): int
… (+3 more overloads to search for String and specify
  starting position)
intern(): String
isEmpty(): boolean
join(CharSequence, CharSequence...): String
join(CharSequence, Iterable<? extends CharSequence>):
  String
lastIndexOf(char): int
… (+3 more overloads to search for String and specify
  ending position)
…
matches(String): boolean
…
replace(char, char): String
replace(CharSequence, CharSequence): String
replaceAll(String, String): String
replaceFirst(String, String): String
split(String): String
split(String, int): String
…
substring(int): String
substring(int, int): String
toCharArray(): char[]
toLowerCase(): String
toLowerCase(Locale): String
…
toUpperCase(): String
toUpperCase(Locale): String
trim(): String
valueOf(boolean): String
… (+8 more overloads for primitives, char[], and Object)
```

Appendable

```
append(char): Appendable
… (+2 more overloads to append all or part of
  a CharSequence)
```

CharSequence

```
charAt(int): int
chars(): IntStream
codePoints(): IntStream
length(): int
subSequence(int, int): CharSequence
toString(): String
```

Comparable<String>

```
compareTo(String): int
```

Serializable

StringBuffer (and ***StringBuilder*** – see notes)

```
StringBuffer()
… (+3 more constructors for initializing from String,
  CharSequence, or with specified capacity)
append(boolean): StringBuffer
… (+12 more overloads for primitives, char[],
  CharSequence, String, StringBuffer, Object)
appendCodePoint(int): StringBuffer
capacity(): int
…
delete(int, int): StringBuffer
deleteCharAt(int): StringBuffer
ensureCapacity(int): void
getChars(int, int, char[], int): void
indexOf(String): int
indexOf(String, int): int
insert(int, boolean): StringBuffer
… (+11 more overloads for primitives, char[],
  CharSequence, String, Object)
lastIndexOf(String): int
lastIndexOf(String, int): int
length(): int
…
replace(int, int, String): StringBuffer
reverse(): StringBuffer
setCharAt(int, char): void
setLength(int): void
…
substring(int): String
substring(int, int): String
…
trimToSize(): String
```

Notes

- Since **`String`** instances are immutable, **`String`** concatenations are generally compiled into **`StringBuilder`**-based operations.
- The **`String.valueOf()`** overloads that take primitive parameters return the same results as the static **`toString()`** methods of the wrapper classes. The **`String.valueOf()`** overloads that take object parameters return the same results as the **`toString()`** methods of the relevant types – except for a `null` argument value: **`String.valueOf(`**`null`**`)`** returns the **`String`** value `"null"`.
- **`StringBuffer`** and **`StringBuilder`** have functionally identical APIs – that is, they have matching constructors and methods. **`StringBuffer`** methods are synchronized for thread-safety; if multi-threaded use is not needed, **`StringBuilder`** should be used instead, as it has better performance.

  `CharSequence` is also implemented by `java.nio.`**`CharBuffer`** and `javax.swing.text.`**`Segment`**, and extended by `javax.lang.model.element.`*`Name`*. However, these special-purpose constructs are beyond the scope of this summary.