



Programming in Java – Strings



Paolo Vercesi
Technical Program Manager

Agenda



Introduction

Character sets and encodings

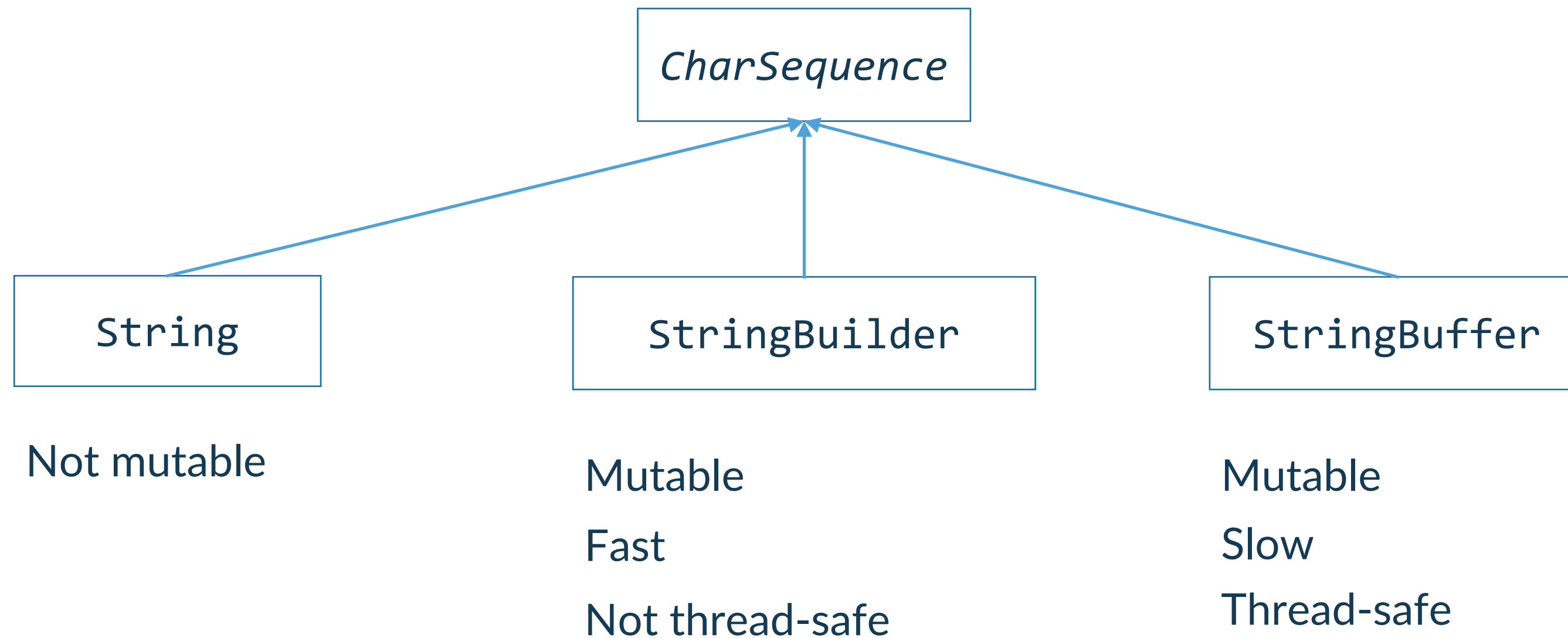
The String API



Introduction



Strings in Java



Non mutable Strings might seem a bit odd at the beginning.

StringBuilder and StringBuffer provide **exactly** the same methods, the only difference is in thread-safety and speed. They could be considered a sort of mutable-String or at least mutable-CharSequence.



String constructors

Constructor	Description
String()	Initializes a newly created String object so that it represents an empty character sequence.
String(byte[], ...)	Many constructors converting bytes into characters. They work well for characters in the US-ASCII character set, otherwise pay attention to the encoding.
String(char[] value, ...)	A couple of constructors to create Strings from character arrays.
String(int[] codePoints, int offset, int count)	Allocates a new String that contains characters from a subarray of the Unicode code point array argument.
String(String original)	Initializes a newly created String object so that it represents the same sequence of characters as the argument; in other words, the newly created string is a copy of the argument string.
String(StringBuffer buffer)	Allocates a new string that contains the sequence of characters currently contained in the string builder argument.
String(StringBuilder builder)	Allocates a new string that contains the sequence of characters currently contained in the string builder argument.



The CharSequence interface

Modifier and Type	Method	Description
char	<code>charAt(int index)</code>	Returns the char value at the specified index.
default <code>IntStream</code>	<code>chars()</code>	Returns a stream of int zero-extending the char values from this sequence.
default <code>IntStream</code>	<code>codePoints()</code>	Returns a stream of code point values from this sequence.
static int	<code>compare(CharSequence cs1, CharSequence cs2)</code>	Compares two CharSequence instances lexicographically.
default boolean	<code>isEmpty()</code>	Returns true if this character sequence is empty.
int	<code>length()</code>	Returns the length of this character sequence.
<code>CharSequence</code>	<code>subSequence(int start, int end)</code>	Returns a CharSequence that is a subsequence of this sequence.
<code>String</code>	<code>toString()</code>	Returns a string containing the characters in this sequence in the same order as this sequence.

<https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/CharSequence.html>



Take aways

- ❑ Java Strings are not mutable
- ❑ StringBuilder and StringBuffer are a efficient mutable strings
- ❑ StringBuffer is thread-safe, String Builder isn't thread-safe

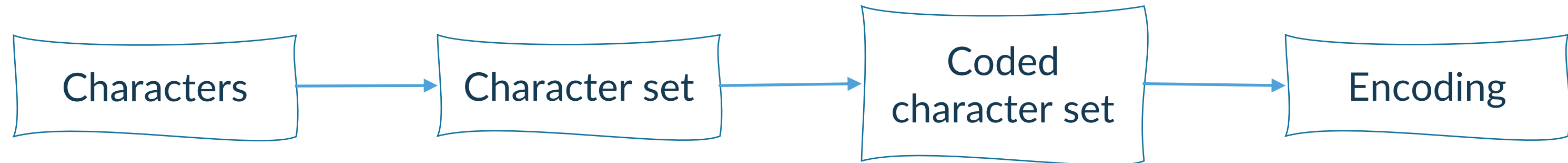




Character sets and encoding



Character sets and encoding



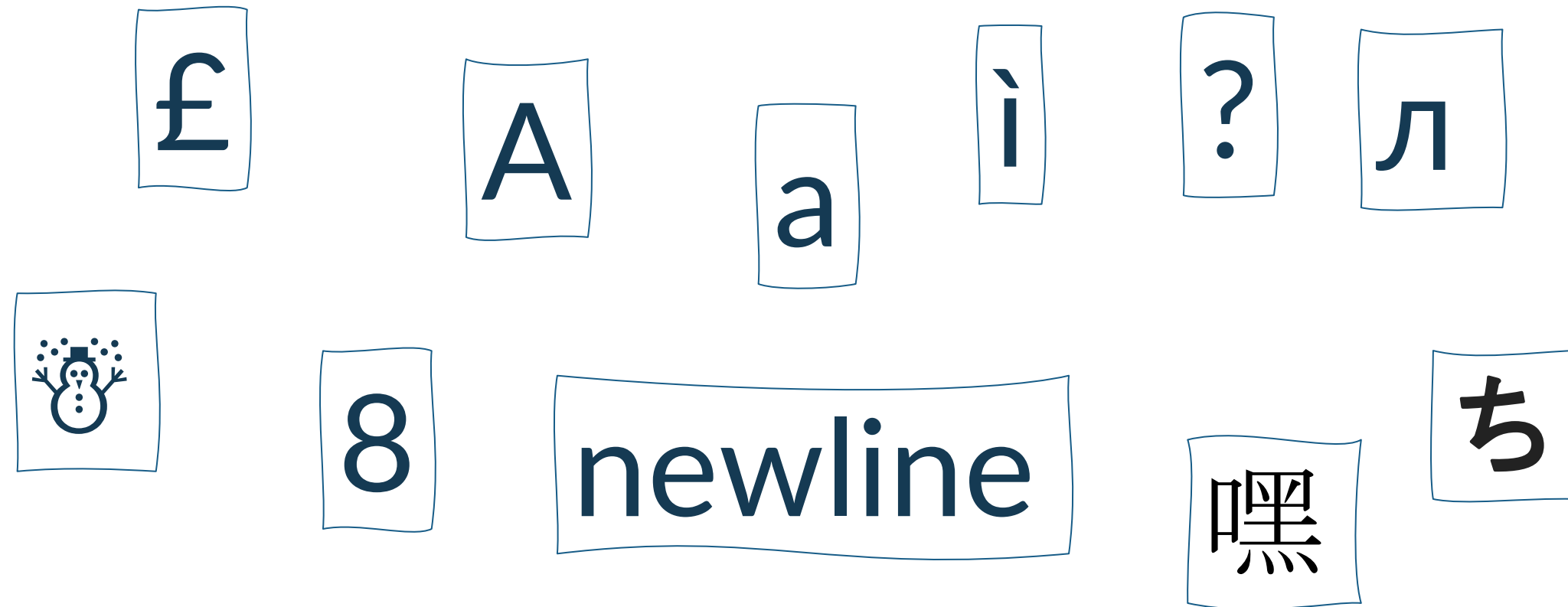
To know everything about character sets and encodings:

<https://www.joelonsoftware.com/2003/10/08/the-absolute-minimum-every-software-developer-absolutely-positively-must-know-about-unicode-and-character-sets-no-excuses/>



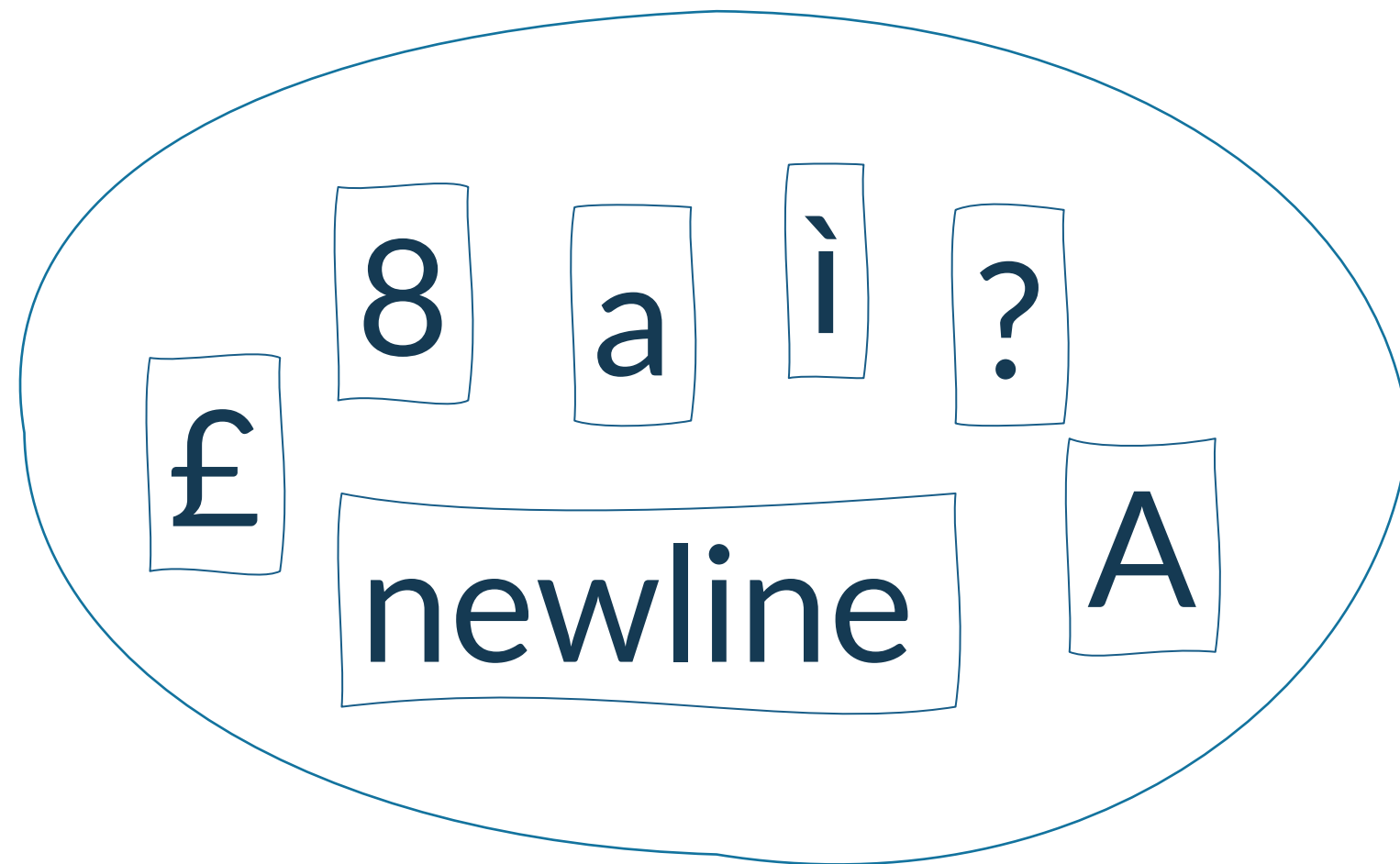
Character

A *character* is a minimal unit of text that has semantic value.



Character set

A *character set* is a collection of characters that might be used by multiple languages. For example, the Latin character set is used by English and most European languages, though the Greek character set is used only by the Greek language.



Unicode terminology – Coded character set

A *coded character set* is a character set where each character is assigned a unique number (code point).

US-ASCII

Code point	Character
0	NUL
1	SOH
...	...
65	A
66	B
67	C
...	...
126	~
127	DEL

Windows-1252/ISO-8859-1

Code point	Character
0	NUL
1	SOH
...	...
65	A
66	B
67	C
...	...
254	þ
255	ÿ

Windows-1250

Code point	Character
0	NUL
1	SOH
...	...
65	A
66	B
67	C
...	...
254	ţ
255	·

Windows-1252 and ISO-8859-1 are not the same character set, but they differ for some code points assigned to control codes. For HTML5 they can be considered the same <https://www.w3.org/TR/encoding/>



Unicode

The Unicode standard defines 144,697 characters and their respective code points. This character set is called **Universal Coded Character Set (UCS, Unicode)**.

Positions 0 through 255 of UCS and Unicode are the same as in **ISO-8859-1**.

Positions 0 through 127 of UCS are the same as in **US-ASCII**.

Positions 0 through 65535 of UCS (**Basic Multilingual Plan**) cover all the commonly used languages



How do we **encode** this information in computers?



Encodings

1 byte is enough to encode the whole US-ASCII and ISO-8859-1 character sets.

For characters sets with more than 256 characters with need to use **multibyte encodings**.

Generally, a character sets define its own encoding and so the term charset is used to refer to both the character set and the encoding. E.g., HTTP and HTML define a charset parameter and attribute, respectively, to define the combination character set/encoding.

UCS is currently the most important character sets and it has multiple encodings, so this character set is represented by the name of the encoding, UTF-8, UTF-16, or UTF-32.

A	Ω	語	𐄎	UTF-32
00000041	000003A9	00008A9E	00010384	
A	Ω	語	𐄎	UTF-16
0041	03A9	8A9E	D800 DF84	
A	Ω	語	𐄎	UTF-8
41	CE A9	E8 AA 9E	F0 90 8E 84	



Java characters

The Java primitive type char uses 16 bit to represent characters.

So, the char type is not able to represents all Unicode characters. Indeed, Java internally represents text in 16-bits **code units** using UTF-16.

The char type does not represent characters but code units, this is relevant only when we are using a language outside the **Basic Multilanguage Plane**. E.g., the cuneiform language, Phoenician, etc.



Encodings supported by Java

Every implementation of the Java platform is required to support the following standard charsets. Usually, every implementation supports **many more charsets**.

Charset	Description
US-ASCII	Seven-bit ASCII, a.k.a. ISO646-US, a.k.a. the Basic Latin block of the Unicode character set
ISO-8859-1	ISO Latin Alphabet No. 1, a.k.a. ISO-LATIN-1
UTF-8	Eight-bit UCS Transformation Format
UTF-16BE	Sixteen-bit UCS Transformation Format, big-endian byte order
UTF-16LE	Sixteen-bit UCS Transformation Format, little-endian byte order
UTF-16	Sixteen-bit UCS Transformation Format, byte order identified by an optional byte-order mark

If in doubt, create documents in **UTF-8**.

What about reading documents in unknown encodings?



Example

Constructor to create a String from a sequence of bytes

```
String(byte[] bytes, String charsetName)  
String(byte[] bytes, Charset charset)
```

Methods to encode a String into a sequence of bytes

```
byte[] getBytes(String charsetName)  
byte[] getBytes(Charset charset)
```



Literals

```
String s1 = "abc";
String s2 = "@ or \u00A9 are the same";
String s3 = "A multiline\nString!";
String s4 = "I\tlove\ttabs";
String s5 = "Windows loves \\backslashes\\";
String s6 = "I hate to \"escape\" double quotes";
String s7 = ""
    We can use "text blocks"
    since Java 15"";
String s8 = "\ud800\udf84 two characters, one codepoint"
String s9 = "\ud800\udf85"
```



There Is no Such Thing
as Plain Text





The String API



Length of a String

Length of a string, the number of Unicode **code units** in the string.

```
string.length()
```

Number of Unicode **code points** in the string.

```
string.codePoints().count()
```

Test if the string is empty.

```
public boolean isEmpty()
```

Test if the string contains blank characters only.

```
public boolean isBlank()
```



String concatenation and conversion

When we concatenate a primitive type to a String, Java invokes one the `String.valueOf()` methods.

```
public static String valueOf(boolean b);
public static String valueOf(byte b);
public static String valueOf(char c);
public static String valueOf(short s);
public static String valueOf(integer i);
public static String valueOf(float f);
public static String valueOf(double d);
```

When we concatenate an object reference to a String, Java invokes the `String.valueOf(Object)` method.

```
public static String valueOf(Object obj) {
    return (obj == null) ? "null" : obj.toString();
}
```

What happens if we invoke `toString()` on a null reference?

```
Object o1 = null;
System.out.println(o1.toString());
```



Character extraction

```
public char charAt(int index)
```

```
public int codePointAt(int index)
```

```
public String substring(int beginIndex)
```

```
public String substring(int beginIndex, int endIndex)
```

```
public CharSequence subSequence(int beginIndex, int endIndex)
```

```
public Stream<String> lines()
```



String comparison

```
public boolean equals(Object anObject)
```

```
public boolean contentEquals(StringBuffer sb)
```

```
public boolean contentEquals(CharSequence cs)
```

```
public boolean equalsIgnoreCase(String anotherString)
```

```
public int compareTo(String anotherString)
```

```
public int compareToIgnoreCase(String str)
```

```
public boolean regionMatches(int toffset, String other, int ooffset, int len)
```

```
public boolean regionMatches(boolean ignoreCase, int toffset,  
                             String other, int ooffset, int len)
```

```
public boolean startsWith(String prefix, int toffset)
```

```
public boolean startsWith(String prefix)
```

```
public boolean endsWith(String suffix)
```

```
public boolean contains(CharSequence s)
```



Searching strings

```
public int indexOf(int ch)
public int indexOf(int ch, int fromIndex)
public int lastIndexOf(int ch)
public int lastIndexOf(int ch, int fromIndex)
public int indexOf(String str)
public int indexOf(String str, int fromIndex)
public int lastIndexOf(String str)
public int lastIndexOf(String str, int fromIndex)
```



Modifying a String

```
public String concat(String str)
public String replace(char oldChar, char newChar)
public String replace(CharSequence target, CharSequence replacement)
public String toLowerCase(Locale locale)
public String toLowerCase()
public String toUpperCase(Locale locale)
public String toUpperCase()
public String trim()
public String strip()
public String stripLeading()
public String stripTrailing()
public String indent(int n)
public String stripIndent()
public String translateEscapes()
public String repeat(int count)
```



Regular expressions

```
public boolean matches(String regex)
```

```
public String replaceFirst(String regex, String replacement)
```

```
public String replaceAll(String regex, String replacement)
```



Splitting and joining Strings

```
public String[] split(String regex, int limit)
```

```
public String[] split(String regex)
```

```
public static String join(CharSequence delimiter, CharSequence... elements)
```

```
public static String join(CharSequence delimiter,  
Iterable<? extends CharSequence> elements)
```



Format

```
public static String format(String format, Object... args)
public static String format(Locale l, String format, Object... args)
public String formatted(Object... args)
```





Thank you!

esteco.com

