Java's API documentation

Java™ Platform, Standard Edition 8
API Specification

Knowing your Java classes

A Java programmer should

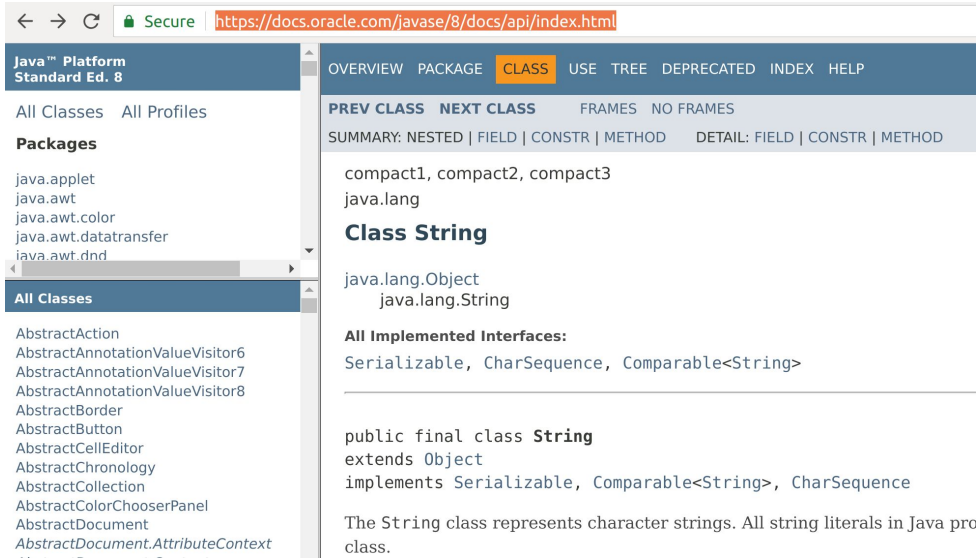
- know about the standard Java API
  - Not by heart, but how to find the documentation and how to read it
- have an idea about how classes in the API are organized in packages
  - Not all packages and classes, there are tons of them!
  - But some basic packages and classes are really good to know about

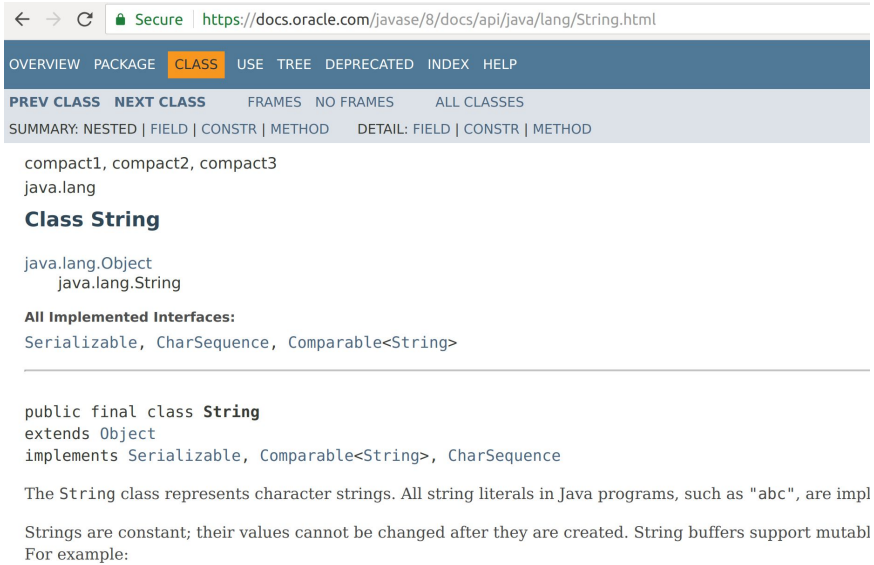
The Java API classes are organized in packages

- https://docs.oracle.com/javase/8/docs/api/index.html
- Related classes are logically grouped in packages, some good-to-know packages include:
- java.lang - Provides classes that are fundamental to the design of the Java programming language.
- java.io - Provides for system input and output through data streams, serialization and the file system.
- java.nio.file - Defines interfaces and classes for the Java virtual machine to access files, file attributes, and file systems.
- java.util - Contains the collections framework, legacy collection classes, event model, date and time facilities, internationalization, and miscellaneous utility classes (a string tokenizer, a random-number generator, and a bit array).
- java.text - Provides classes and interfaces for handling text, dates, numbers, and messages in a manner independent of natural languages.

The API docs come in two flavors

- One with frames (yey! 1990's, we love you!)
- One without frames





A look at java.lang.String

Strings are so common in programming that Java has placed them in the java.lang package. All classes there can be used directly in your code.

In order to use classes from other packages, you need to “import” the classes.

Here’s the link to the java.lang.String documentation page

A look at java.lang.String

The documentation page has four sections:

- The class declaration (`public final class String extends Object implements Serializable, Comparable<String>, CharSequence`), and a description (often with code examples)
- A section with the “fields” (variables in the class)
- The constructors of the class
- The methods of the class

A look at java.lang.String - description

public final class **String**
extends Object
implements Serializable, Comparable<String>, CharSequence

The `String` class represents character strings. All string literals in Java programs, such as "abc", are implemented as instances of this class.

Strings are constant; their values cannot be changed after they are created. String buffers support mutable strings. Because String objects are immutable they can be shared. For example:

```
String str = "abc";
```

is equivalent to:

```
char data[] = {'a', 'b', 'c'};
String str = new String(data);
```

Here are some more examples of how strings can be used:

```
System.out.println("abc");
String cde = "cde";
System.out.println("abc" + cde);
String c = "abc".substring(2,3);
String d = cde.substring(1, 2);
```

…

A look at java.lang.String - fields

Field Summary

Fields

Modifier and Type	Field and Description
`static Comparator<String>`	`CASE_INSENSITIVE_ORDER`

A `Comparator` that orders `String` objects as by `compareToIgnoreCase`.

A look at java.lang.String - constructors

Constructor Summary

Constructors

Constructor and Description

`String()`

Initializes a newly created String object so that it represents an empty character sequence.

`String(byte[] bytes)`

Constructs a new String by decoding the specified array of bytes using the platform's default charset.

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

Constructs a new String by decoding the specified array of bytes using the specified charset.

`String(byte[] ascii, int hibyte)`

Deprecated.

This method does not properly convert bytes into characters. As of JDK 1.1, the preferred way to do this is via the String constructors that take a Charset, charset name, or that use the platform's default charset.

...

...

A look at java.lang.String - methods

First, a list with links to show various kinds of methods (more on this later in the course/book):

Method Summary

All Methods Static Methods Instance Methods Concrete Methods Deprecated Methods

A look at java.lang.String - methods

Next, the list of methods (here showing instance methods:

Modifier and Type	Method and Description
`char`	`charAt(int index)` Returns the char value at the specified index.
`int`	`codePointAt(int index)` Returns the character (Unicode code point) at the specified index.
`int`	`codePointBefore(int index)` Returns the character (Unicode code point) before the specified index.
`int`	`codePointCount(int beginIndex, int endIndex)` Returns the number of Unicode code points in the specified text range of this String.
`int`	`compareTo(String anotherString)` Compares two strings lexicographically.
`int`	`compareToIgnoreCase(String str)` Compares two strings lexicographically, ignoring case differences.

Clicking on a method name takes you down to the longer description

A look at java.lang.String - methods - charAt()

Longer description of charAt():

`charAt`

`public char charAt(int index)`

Returns the char value at the specified index. An index ranges from 0 to length() - 1. The first char value of the sequence is at index 0, the next at index 1, and so on, as for array indexing.

If the char value specified by the index is a surrogate, the surrogate value is returned.

Specified by:

`charAt` in interface `CharSequence`

Parameters:

`index` - the index of the char value.

Returns:

the char value at the specified index of this string. The first char value is at index 0.

Throws:

`IndexOutOfBoundsException` - if the index argument is negative or not less than the length of this string.

A look at java.lang.String - methods - charAt()

Using the information from the description and the “signature”

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

we can tell that a String object has an instance method called charAt() which takes an int argument and gives us back the char (character) at the specified index (starting at 0):

```
String name = "Langefors";
char ch = name.charAt(5); // 'f'
```

Next

Next, see the live videos about the API and do the short exercises.

Don’t skip this part. Knowing your way around the API documentation is essential if you want to succeed as either a student or professional programmer!