

# **Progettazione del Software**

**Giuseppe De Giacomo, Paolo Liberatore,  
Massimo Mecella**

## **Esercitazione 2**



# API Java

- Con API (Application Programming Interface) si indicano tutta una serie di procedure che sono a disposizione del programmatore.
- In particolare le API Java mostrano tutte le classi offerte dal linguaggio e descrivono i metodi a disposizione in ogni classe.
  - Anche dette le librerie Java
- Le API per Java 8 sono disponibili all'URL <https://docs.oracle.com/javase/8/docs/api/>



# API Java

- Per ogni classe le API riportano e descrivono in dettaglio superclassi, sottoclassi, funzionalità ad alto livello della classe, campi, costruttori, metodi.
- In questa esercitazione ci concentreremo soprattutto sui metodi



# API Java

https://docs.oracle.com/javase/7/docs/api/ api java

Java™ Platform Standard Ed. 7

Overview Package Class Use Tree Deprecated Index Help

Prev Next Frames No Frames

## Java™ Platform, Standard Edition 7 API Specification

This document is the API specification for the Java™ Platform, Standard Edition.

See: Description

### Packages

Package	Description
<a href="#">java.applet</a>	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
<a href="#">java.awt</a>	Contains all of the classes for creating user interfaces and for painting graphics and images.
<a href="#">java.awt.color</a>	Provides classes for color spaces.
<a href="#">java.awt.datatransfer</a>	Provides interfaces and classes for transferring data between and within applications.
<a href="#">java.awt.dnd</a>	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
<a href="#">java.awt.event</a>	Provides interfaces and classes for dealing with different types of events fired by AWT components.
<a href="#">java.awt.font</a>	Provides classes and interface relating to fonts.
<a href="#">java.awt.geom</a>	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
<a href="#">java.awt.im</a>	Provides classes and interfaces for the input method framework.
<a href="#">java.awt.im.spi</a>	Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.
<a href="#">java.awt.image</a>	Provides classes for creating and modifying images.
<a href="#">java.awt.image.renderable</a>	Provides classes and interfaces for producing rendering-independent images.
<a href="#">java.awt.print</a>	Provides classes and interfaces for a general printing API.
<a href="#">java.beans</a>	Contains classes related to developing <i>beans</i> – components based on the JavaBeans™ architecture.
<a href="#">java.beans.beancontext</a>	Provides classes and interfaces relating to bean context.
<a href="#">java.io</a>	Provides for system input and output through data streams, serialization and the file system.
<a href="#">java.lang</a>	Provides classes that are fundamental to the design of the Java programming language.
<a href="#">java.lang.annotation</a>	Provides library support for the Java programming language annotation facility.
<a href="#">java.lang.instrument</a>	Provides services that allow Java programming language agents to instrument programs running on the JVM.

AbstractAction  
AbstractAnnotationValueVisitor6  
AbstractAnnotationValueVisitor7  
AbstractBorder  
AbstractButton  
AbstractCellEditor  
AbstractCollection  
AbstractColorChooserPanel  
AbstractDocument  
AbstractDocument.AttributeContext  
AbstractDocument.Content  
AbstractDocument.ElementEdit  
AbstractElementVisitor6  
AbstractElementVisitor7  
AbstractExecutorService  
AbstractInterruptibleChannel  
AbstractLayoutCache  
AbstractLayoutCache.NodeDimensions  
AbstractList  
AbstractListModel  
AbstractMap  
AbstractMap.SimpleEntry  
AbstractMap.SimpleImmutableEntry  
AbstractMarshalerImpl  
AbstractMethodError  
AbstractOwnableSynchronizer  
AbstractPreferences  
AbstractProcessor  
AbstractQueue  
AbstractQueuedLongSynchronizer  
AbstractQueuedSynchronizer



# API Java

## Esempio: la classe String

The screenshot shows the Oracle Java API documentation for the `String` class. The browser address bar shows `https://docs.oracle.com/javase/7/docs/api/` with a search for `api java`. The left sidebar lists various Java packages, with `String` highlighted under `java.lang`. The main content area is titled `Class String` and includes the following information:

- Overview** | **Package** | **Class** | **Use** | **Tree** | **Deprecated** | **Index** | **Help**
- Prev Class** | **Next Class** | **Frames** | **No Frames**
- Summary:** [Nested](#) | [Field](#) | [Constr](#) | [Method](#) | **Detail:** [Field](#) | [Constr](#) | [Method](#)

**java.lang**

### Class String

java.lang.Object  
java.lang.String

**All Implemented Interfaces:**  
Serializable, CharSequence, Comparable<String>

```
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);
```

The class `String` includes methods for examining individual characters of the sequence, for comparing strings, for searching strings, for extracting substrings, and for creating a copy of a string with all characters translated to uppercase or to lowercase. Case mapping is based on the Unicode Standard version specified by the `Character` class.

The Java language provides special support for the string concatenation operator (`+`), and for conversion of other objects to strings. String concatenation is implemented through the `StringBuilder` (or `StringBuffer`) class and its `append` method. String conversions are implemented through the method `toString`, defined by `Object` and inherited by all classes in Java. For additional information on string concatenation and conversion, see Gosling, Joy, and Steele, *The Java Language Specification*.



# API Java

## Esempio: la classe String

← ⓘ 🔒 https://docs.oracle.com/javase/7/docs/api/ ↻ 🔍 api java ☆ 📁 🔄 ⬇️ 🏠 🔴

java.awt.image.Renderable  
java.awt.print  
java.beans  
java.beans.beancontext  
java.io  
java.lang  
java.lang.annotation  
java.lang.instrument  
java.lang.invoke  
java.lang.management  
java.lang.ref  
java.lang.reflect  
java.math  
java.net  
java.nio

StatementEvent  
StatementEventListener  
StAXResult  
StAXSource  
Streamable  
StreamableValue  
StreamCorruptedException  
StreamFilter  
StreamHandler  
StreamPrintService  
StreamPrintServiceFactory  
StreamReaderDelegate  
StreamResult  
StreamSource  
StreamTokenizer  
StrictMath  
**String**  
StringBuffer  
StringBufferInputStream  
StringBuilder  
StringCharacterIterator  
StringContent  
StringHolder  
StringIndexOutOfBoundsException  
StringMonitor  
StringMonitorMBean  
StringNameHelper  
StringReader  
StringRefAddr  
StringSelection  
StringSeqHelper  
StringSeqHolder  
StringTokenizer  
StringValueExp

### Field Summary

Fields	
Modifier and Type	Field and Description
static Comparator<String>	CASE_INSENSITIVE_ORDER A Comparator that orders <code>String</code> objects as by <code>compareToIgnoreCase</code> .

### Constructor Summary

Constructors
Constructor and Description
<code>String()</code> Initializes a newly created <code>String</code> object so that it represents an empty character sequence.
<code>String(byte[] bytes)</code> Constructs a new <code>String</code> by decoding the specified array of bytes using the platform's default charset.
<code>String(byte[] bytes, Charset charset)</code> Constructs a new <code>String</code> by decoding the specified array of bytes using the specified <b>charset</b> .
<code>String(byte[] ascii, int hibyte)</code> <b>Deprecated.</b> <i>This method does not properly convert bytes into characters. As of JDK 1.1, the preferred way to do this is via the <code>String</code> constructors that take a <code>charset</code>, <code>charset</code> name, or that use the platform's default charset.</i>
<code>String(byte[] bytes, int offset, int length)</code> Constructs a new <code>String</code> by decoding the specified subarray of bytes using the platform's default charset.
<code>String(byte[] bytes, int offset, int length, Charset charset)</code> Constructs a new <code>String</code> by decoding the specified subarray of bytes using the specified <b>charset</b> .
<code>String(byte[] ascii, int hibyte, int offset, int count)</code> <b>Deprecated.</b> <i>This method does not properly convert bytes into characters. As of JDK 1.1, the preferred way to do this is via the <code>String</code> constructors that take a <code>charset</code>, <code>charset</code> name, or that use the platform's default charset.</i>
<code>String(byte[] bytes, int offset, int length, String charsetName)</code> Constructs a new <code>String</code> by decoding the specified subarray of bytes using the specified charset.
<code>String(byte[] bytes, String charsetName)</code> Constructs a new <code>String</code> by decoding the specified array of bytes using the specified <b>charset</b> .
<code>String(char[] value)</code> Allocates a new <code>String</code> so that it represents the sequence of characters currently contained in the character array argument.
<code>String(char[] value, int offset, int count)</code>



# API Java

## Esempio: la classe String

Browser address bar: <https://docs.oracle.com/javase/7/docs/api/> Search:

- java.awt.image.Renderable
- java.awt.print
- java.beans
- java.beans.beancontext
- java.io
- java.lang
- java.lang.annotation
- java.lang.instrument
- java.lang.invoke
- java.lang.management
- java.lang.ref
- java.lang.reflect
- java.math
- java.net
- java.nio
- StatementEvent
- StatementEventListener
- StAXResult
- StAXSource
- Streamable
- StreamableValue
- StreamCorruptedException
- StreamFilter
- StreamHandler
- StreamPrintService
- StreamPrintServiceFactory
- StreamReaderDelegate
- StreamResult
- StreamSource
- StreamTokenizer
- StrictMath
- String**
- StringBuffer
- StringBufferInputStream
- StringBuilder
- StringCharacterIterator
- StringContent
- StringHolder
- StringIndexOutOfBoundsException
- StringMonitor
- StringMonitorMBean
- StringNameHelper
- StringReader
- StringRefAddr
- StringSelection
- StringSeqHelper
- StringSeqHolder
- StringTokenizer
- StringValueExp

### Method Summary

Methods	
Modifier and Type	Method and Description
char	<code>charAt(int index)</code> Returns the char value at the specified index.
int	<code>codePointAt(int index)</code> Returns the character (Unicode code point) at the specified index.
int	<code>codePointBefore(int index)</code> Returns the character (Unicode code point) before the specified index.
int	<code>codePointCount(int beginIndex, int endIndex)</code> Returns the number of Unicode code points in the specified text range of this string.
int	<code>compareTo(String anotherString)</code> Compares two strings lexicographically.
int	<code>compareToIgnoreCase(String str)</code> Compares two strings lexicographically, ignoring case differences.
String	<code>concat(String str)</code> Concatenates the specified string to the end of this string.
boolean	<code>contains(CharSequence s)</code> Returns true if and only if this string contains the specified sequence of char values.
boolean	<code>contentEquals(CharSequence cs)</code> Compares this string to the specified CharSequence.
boolean	<code>contentEquals(StringBuffer sb)</code> Compares this string to the specified StringBuffer.
static String	<code>copyValueOf(char[] data)</code> Returns a String that represents the character sequence in the array specified.
static String	<code>copyValueOf(char[] data, int offset, int count)</code> Returns a String that represents the character sequence in the array specified.
boolean	<code>endsWith(String suffix)</code> Tests if this string ends with the specified suffix.
boolean	<code>equals(Object anObject)</code> Compares this string to the specified object.
boolean	<code>equalsIgnoreCase(String anotherString)</code> Compares this String to another String, ignoring case considerations.
static String	<code>format(Locale l, String format, Object... args)</code> Returns a formatted string using the specified locale, format string, and arguments.
static String	<code>format(String format, Object... args)</code> Returns a formatted string using the specified format string and arguments.



# Intro

Nel risolvere gli esercizi

si faccia riferimento ai seguenti metodi della classe String:

- *toUpperCase()* per trasformare tutti i caratteri di una stringa in maiuscolo
- *substring(i,j)* per estrarre una sottostringa
- *contains(x)* per verificare che una stringa contenga la stringa x
- *length()* per ottenere la lunghezza della stringa
- *charAt(i)* per ottenere l'i-esimo carattere della stringa

Per utilizzare i metodi della classe JOptionPane, questa va importata dal package javax.swing

Per utilizzare i metodi della classe Scanner, questa va importata dal package java.util





# Esercizio 1

Scrivere un programma Java che date due stringhe in input stampi *true* se una stringa è contenuta nell'altra, *false* altrimenti.

Implementare due varianti in cui le stringhe sono:

- Lette in input dalla console
- Lette in input da finestra di dialogo (JOptionPane)



## Esercizio 2

Scrivere un programma Java che data una stringa interamente in minuscolo, stampi la stringa stessa con le lettere nelle posizioni dispari in maiuscolo.

*Esempio: input = programmazione ; output= PrOgRaMmAzIoNe*



# Esercizio 3

Scrivere un programma Java che dato un intero rappresentante un anno, indichi se questo è bisestile.

Un anno è bisestile se è divisibile per 4 ma non per 100 oppure per 400.



# Esercizio 4

Scrivere un programma Java che date due stringhe, stampi il rapporto tra le lunghezze delle stringhe stesse.

*Esempio: input1 = "lessie", input2 = "rex" il risultato deve essere 2,  
input1 = "rex", input2 = "lessie" il risultato deve essere 0.5*



# Esercizio 5

Scrivere un programma Java che chieda in input all'utente due stringhe, e stampi solo la stringa che abbia la somma dei codici ASCII delle proprie lettere maggiore e il valore di tale somma.

*Esempio: se le stringhe passate sono "aaa" e "aab" il programma deve stampare "aab 292"*



## Esercizio 6

Scrivere un programma Java che calcoli l'area di triangoli. Iterativamente il programma deve chiedere in input all'utente due valori numerici rappresentanti la base e l'altezza del triangolo corrente (leggendolo o da console o da JOptionPane) e deve scrivere il risultato in una finestra di output.

Il programma continua a chiedere valori e ad eseguire il calcolo finché non viene inserito un numero negativo.

