

UD4: Desarrollo de una aplicación basada en POO. Utilización de clases predefinidas.

**IES LOS SAUCES – BENAVENTE
CFGS DESARROLLO DE APLICACIONES WEB
PROGRAMACIÓN**

API

Al instalar el JDK, además del compilador y la máquina virtual de Java también tendremos la API (Application Programming Interface) de Java.

La API de Java es un conjunto de clases para efectuar toda clase de tareas necesarias dentro de una aplicación.

La API está organizada en paquetes, donde cada paquete contiene un conjunto de clases.

Para poder utilizar la API deberemos consultar la documentación de la misma.

<https://docs.oracle.com/en/java/javase/21/docs/api/index.html>

API

OVERVIEW MODULE PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

Java SE 11 & JDK 11

ALL CLASSES

SEARCH: Search

Java® Platform, Standard Edition & Java Development Kit Version 11 API Specification

This document is divided into two sections:

Java SE

The Java Platform, Standard Edition (Java SE) APIs define the core Java platform for general-purpose computing. These APIs are in modules whose names start with `java`.

JDK

The Java Development Kit (JDK) APIs are specific to the JDK and will not necessarily be available in all implementations of the Java SE Platform. These APIs are in modules whose names start with `jdk`.

Se organiza en módulos (desde Java 9)

All Modules	Java SE	JDK	Other Modules
Module	Description		
<code>java.base</code>	Defines the foundational APIs of the Java SE Platform.		
<code>java.compiler</code>	Defines the Language Model, Annotation Processing, and Java Compiler APIs.		
<code>java.datatransfer</code>	Defines the API for transferring data between and within applications.		
<code>java.desktop</code>	Defines the AWT and Swing user interface toolkits, plus APIs for accessibility, audio, imaging, printing, and JavaBeans.		
<code>java.instrument</code>	Defines services that allow agents to instrument programs running on the JVM.		
<code>java.logging</code>	Defines the Java Logging API.		
<code>java.management</code>	Defines the Java Management Extensions (JMX) API.		
<code>java.management.rmi</code>	Defines the RMI connector for the Java Management Extensions (JMX) Remote API.		
<code>java.naming</code>	Defines the Java Naming and Directory Interface (JNDI) API.		
<code>java.net.http</code>	Defines the HTTP Client and WebSocket APIs.		
<code>java.prefs</code>	Defines the Preferences API.		
<code>java.rmi</code>	Defines the Remote Method Invocation (RMI) API.		
<code>java.scripting</code>	Defines the Scripting API.		
<code>java.se</code>	Defines the API of the Java SE Platform.		

API

OVERVIEW MODULE PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

Java SE 11 & JDK 11

ALL CLASSES

SEARCH:

MODULE: DESCRIPTION | MODULES | PACKAGES | SERVICES

Module java.base

Defines the foundational APIs of the Java SE Platform.

Providers:

The JDK implementation of this module provides an implementation of the `jrt` file system provider to enumerate and read the class and resource files in a run-time image. The `jrt` file system can be created by calling `FileSystems.newFileSystem(URI.create("jrt:/"))`.

Tool Guides:

java launcher, keytool

Module Graph:

java.base

Since:
9

Packages

Exports

Package	Description
java.io	Provides for system input and output through data streams, serialization and the file system.
java.lang	Provides classes that are fundamental to the design of the Java programming language.
java.lang.annotation	Provides library support for the Java programming language annotation facility.
java.lang.invoke	The <code>java.lang.invoke</code> package provides low-level primitives for interacting with the Java Virtual Machine.
java.lang.module	Classes to support module descriptors and creating configurations of modules by means of resolution and service binding.
java.lang.ref	Provides reference-object classes, which support a limited degree of interaction with the garbage collector.
java.lang.reflect	Provides classes and interfaces for obtaining reflective information about classes and objects.
java.math	Provides classes for performing arbitrary-precision integer arithmetic (<code>BigInteger</code>) and arbitrary-precision decimal arithmetic (<code>BigDecimal</code>).

Los módulos contienen
paquetes

Module java.base

Package java.lang

Provides classes that are fundamental to the design of

Frequently it is necessary to represent a value of primitive type that can be stored in a variable of reference type. The class `Void` represents the type `void`.

The class `Math` provides commonly used mathematical functions such as sine, cosine, and square root. The classes `String`, `StringBuffer`, and `StringBuilder` similarly provide commonly used operations on character strings.

Classes `ClassLoader`, `Process`, `ProcessBuilder`, `Runtime`, `SecurityManager`, and `System` provide "system operations" that manage the dynamic loading of classes, creation of external processes, host environment inquiries such as the time of day, and enforcement of security policies.

Class `Throwable` encompasses objects that may be thrown by the `throw` statement. Subclasses of `Throwable` represent errors and exceptions.

Character Encodings

The specification of the `java.nio.charset.Charset` class describes the naming conventions for character encodings as well as the set of standard encodings that must be supported by every implementation of the Java platform.

Since:
1.0

Interface Summary

Interface	Description
Appendable	An object to which char sequences and values can be appended.
AutoCloseable	An object that may hold resources (such as file or socket handles) until it is closed.
CharSequence	A <code>CharSequence</code> is a readable sequence of char values.
Cloneable	A class implements the <code>Cloneable</code> interface to indicate to the <code>Object.clone()</code> method that it is legal for that method to make a field-for-field copy of instances of that class.
Comparable<T>	This interface imposes a total ordering on the objects of each class that implements it.
Iterable<T>	Implementing this interface allows an object to be the target of the enhanced <code>for</code> statement (sometimes called the "for-each loop" statement).
ProcessHandle	<code>ProcessHandle</code> identifies and provides control of native processes.
ProcessHandle.Info	Information snapshot about the process.
Readable	A <code>Readable</code> is a source of characters.
Runnable	The <code>Runnable</code> interface should be implemented by any class whose instances are intended to be executed by a thread.
StackWalker.StackFrame	A <code>StackFrame</code> object represents a method invocation returned by <code>StackWalker</code> .
System.Logger	<code>System.Logger</code> instances log messages that will be routed to the underlying logging framework the <code>LoggerFinder</code> uses.
Thread.UncaughtExceptionHandler	Interface for handlers invoked when a <code>Thread</code> abruptly terminates due to an uncaught exception.

Class Summary

Class	Description
Boolean	The <code>Boolean</code> class wraps a value of the primitive type <code>boolean</code> in an object.
Byte	The <code>Byte</code> class wraps a value of primitive type <code>byte</code> in an object.
Character	The <code>Character</code> class wraps a value of the primitive type <code>char</code> in an object.
Character.Subset	Instances of this class represent particular subsets of the Unicode character set.
Character.UnicodeBlock	A family of character subsets representing the character blocks in the Unicode specification.
Class<T>	Instances of the class <code>Class</code> represent classes and interfaces in a running Java application.

Cada paquete contiene, entre otros elementos, un conjunto de clases

API

OVERVIEW MODULE PACKAGE **CLASS** USE TREE DEPRECATED INDEX HELP

Java SE 11 & JDK 11

ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

SEARCH:

Module java.base
Package 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. For additional information on string concatenation and conversion, see *The Java™ Language Specification*.

Nombre de la clase

Descripción de la clase

15.18.1 String Concatenation Operator +

Field Summary

Atributos

Fields

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

Constructor Summary

Constructores

Constructors

Constructor	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[] ascii, int hibyte)	Deprecated. This method does not properly convert bytes into characters.
String(byte[] bytes, int offset, int length)	Constructs a new String by decoding the specified subarray of bytes using the platform's default charset.
String(byte[] ascii, int hibyte, int offset, int count)	Deprecated. This method does not properly convert bytes into characters.
String(byte[] bytes, int offset, int length, String charsetName)	Constructs a new String by decoding the specified subarray of bytes using the specified charset.
String(byte[] bytes, int offset, int length, Charset charset)	Constructs a new String by decoding the specified subarray of bytes using the specified charset.
String(byte[] bytes, String charsetName)	Constructs a new String by decoding the specified array of bytes using the specified charset.
String(byte[] bytes, Charset charset)	Constructs a new String by decoding the specified array of bytes using the specified charset.
String(char[] value)	Allocates a new String so that it represents the sequence of characters currently contained in the character array argument.

API

OVERVIEW MODULE PACKAGE **CLASS** USE TREE DEPRECATED INDEX HELP

Java SE 11 & JDK 11

ALL CLASSES

SEARCH:

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Method Summary

Métodos

All Methods	Static Methods	Instance Methods	Concrete Methods	Deprecated Methods
Modifier and Type	Method	Description		
char	<code>charAt(int index)</code>	Returns the <code>char</code> value at the specified index.		
<code>IntStream</code>	<code>chars()</code>	Returns a stream of <code>int</code> zero-extending the <code>char</code> values from this sequence.		
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 <code>String</code> .		
<code>IntStream</code>	<code>codePoints()</code>	Returns a stream of code point values from this sequence.		
int	<code>compareTo(String anotherString)</code>	Compares two strings lexicographically.		
int	<code>compareToIgnoreCase(String str)</code>	Compares two strings lexicographically, ignoring case differences.		
<code>String</code>	<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 <code>CharSequence</code> .		
boolean	<code>contentEquals(StringBuffer sb)</code>	Compares this string to the specified <code>StringBuffer</code> .		
static <code>String</code>	<code>copyValueOf(char[] data)</code>	Equivalent to <code>valueOf(char[])</code> .		
static <code>String</code>	<code>copyValueOf(char[] data, int offset, int count)</code>	Equivalent to <code>valueOf(char[], int, int)</code> .		
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 <code>String</code> to another <code>String</code> , ignoring case considerations.		
static <code>String</code>	<code>format(String format, Object... args)</code>	Returns a formatted string using the specified format string and arguments.		

API

Field Detail

Tipo

Nombre

CASE_INSENSITIVE_ORDER

```
public static final Comparator<String> CASE_INSENSITIVE_ORDER
```

A Comparator that orders String objects as by `compareToIgnoreCase`. This comparator is serializable.

Note that this Comparator does *not* take locale into account, and will result in an unsatisfactory ordering for certain locales. The `Collator` class provides locale-sensitive comparison.

Since:

1.2

See Also:

`Collator`

Modificadores

API

Constructor Detail

String

```
public String()
```

Initializes a newly created String object so that are immutable.

Modificador de acceso

String

```
public String(String original)
```

Initializes a newly created String object so that copy of the argument string. Unless an explicit c

Parameters:

original - A String

Parámetros

API

Modificadores

charAt

Tipo devuelto

Nombre

public **char** **charAt**(**int** **index**)

Returns the `char` value at the specified index. An index ranges from 0 to `length() - 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.

Parámetros

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 `len`

Package java.lang (Module java.base)

Contiene las clases fundamentales. No es necesario importarlo.

Object

Class

System

String

StringBuilder

Math

Character

Integer

Float

Boolean

Object

Clase que es la superclase de todas las clases.

Todos los objetos implementan los métodos de esta clase.

public final Class<?> getClass()

Devuelve la clase de un objeto

public boolean equals(Object obj)

Indica si un objeto es igual a otro

protected Object clone()

Devuelve una copia del objeto

public String toString()

Devuelve una representación del objeto en forma de cadena de caracteres.

Class

Clase que representa las clases que se está ejecutando en la aplicación, hay una instancia de Class por cada clase cargada.

public boolean isInstance(Object obj)

Devuelve true si el argumento es no nulo y se puede convertir al tipo de referencia que representa este objeto Class

public String getName()

Devuelve el nombre de la entidad representado por este objeto de clase, como una cadena

static Class<?> forName(String className)

Devuelve el objeto Class asociado con la clase o interfaz con el nombre de cadena dado.

System

Clase que proporciona acceso a los recursos del sistema.

public static final PrintStream out

Salida estándar

public static final InputStream in

Entrada estándar

public static final PrintStream err

Salida de error estándar

public static Console console()

Devuelve el único objeto Console asociado a la máquina virtual de Java

public static long currentTimeMillis()

Devuelve la diferencia, medida en milisegundos, entre el momento actual y la medianoche del 1 de enero de 1970

public static String getProperty(String key)

Obtiene la propiedad del sistema indicada por la clave especificada

public static void exit(int status)

Termina la ejecución de la máquina virtual de Java

String

Clase que representa una cadena de caracteres.

Todas las cadenas de caracteres en java se implementan como una instancia de la clase String.

Los objetos de la clase String son constantes, sus valores no se pueden modificar después de haber sido creados.

public char charAt(int index)

Devuelve el valor del carácter especificado por el índice

public int compareTo(String anotherString)

Compara dos String lexicográficamente. Devuelve 0 si la cadena argumento es igual a esta cadena, un valor menor que 0 si esta cadena es menor que el argumento y un valor mayor que 0 si esta cadena es mayor que el argumento

public boolean equals(Object anObject)

Devuelve true si el parámetro representa un String equivalente a este

String

public int compareToIgnoreCase(String str)

Compara 2 String lexicográficamente, ignorando mayúsculas y minúsculas. Devuelve 0 si la cadena argumento es igual a esta cadena, un valor menor que 0 si esta cadena es menor que el argumento y un valor mayor que 0 si esta cadena es mayor que el argumento

public boolean startsWith(String prefix)

Comprueba si esta cadena comienza con el prefijo especificado

public boolean endsWith(String suffix)

Comprueba si esta cadena termina con el sufijo especificado

public int indexOf(String str)

Devuelve el índice de la primera aparición de la subcadena especificada o -1 si la subcadena no aparece

public int lastIndexOf(String str)

Devuelve el índice de la última aparición de la subcadena especificada o -1 si la subcadena no aparece

String

<i>public int length()</i>	Devuelve la longitud de esta cadena
<i>public String substring(int beginIndex, int endIndex)</i>	Devuelve una subcadena de esta cadena. La subcadena comienza en beginIndex y se extiende hasta endIndex-1
<i>public String concat(String str)</i>	Concatena la cadena especificada al final de esta cadena
<i>public String replace(char oldChar, char newChar)</i>	Devuelve una cadena que resulta de reemplazar todas las apariciones de oldChar en esta cadena con newchar
<i>public String toLowerCase()</i>	Convierte todos los caracteres de esta cadena a minúsculas
<i>public String toUpperCase()</i>	Convierte todos los caracteres de esta cadena a mayúsculas
<i>public String trim()</i>	Devuelve una cadena cuyo valor es esta cadena con los espacios en blanco iniciales y finales eliminados
<i>public static String valueOf(int i)</i>	Devuelve la representación de cadena del argumento

String

Métodos añadidos en Java 11

public boolean isBlank()

Devuelve **true** si la cadena está vacía o contiene solo espacios en blanco; de lo contrario, es **false**

public String strip()

Devuelve una cadena cuyo valor es esta cadena, con todos los espacios en blanco iniciales y finales eliminados

public String stripLeading()

Devuelve una cadena cuyo valor es esta cadena, con todos los espacios en blanco iniciales eliminados

public String stripTrailing()

Devuelve una cadena cuyo valor es esta cadena, con todos los espacios en blanco finales eliminados

public String repeat(int count)

Devuelve una cadena cuyo valor es la concatenación de esta cadena repetidas veces

StringBuilder

Clase que representa una cadena de caracteres que se puede modificar.

<i>public char charAt(int index)</i>	Devuelve el valor del carácter especificado por el índice
<i>public int length()</i>	Devuelve la longitud
<i>public StringBuilder append(String str)</i>	Añade la cadena especificada en el parámetro
<i>public StringBuilder insert(int offset, String str)</i>	Inserta la cadena en esta secuencia de caracteres
<i>public StringBuilder reverse()</i>	Devuelve esta cadena dada la vuelta
<i>public String substring(int start, int end)</i>	Devuelve una nueva cadena que contiene una subsecuencia de caracteres contenidos en esta secuencia.

Math

Contiene métodos para realizar operaciones numéricas.

<i>public static final double E</i>	Constante que representa el número e
<i>public static final double PI</i>	Constante que representa el número pi
<i>public static double sqrt(double a)</i>	Devuelve la raíz cuadrada del argumento
<i>public static double pow(double a, double b)</i>	Devuelve el valor del primer argumento elevado a la potencia del segundo argumento
<i>public static int round(float a)</i>	Devuelve el entero más cercano al argumento
<i>public static int abs(int a)</i>	Devuelve el valor absoluto del argumento
<i>public static int max(int a, int b)</i>	Devuelve el mayor de los dos argumentos
<i>public static int min(int a, int b)</i>	Devuelve el menor de los dos argumentos

Character

Clase envoltorio del tipo primitivo char.

<i>public static boolean isLowerCase(char ch)</i>	Determina si el argumento es un carácter minúscula
<i>public static boolean isUpperCase(char ch)</i>	Determina si el argumento es un carácter mayúscula
<i>public static boolean isDigit(char ch)</i>	Determina si el argumento es un dígito
<i>public static boolean isLetter(char ch)</i>	Determina si el carácter especificado es una letra
<i>public static boolean isWhiteSpace(char ch)</i>	Determina si el carácter especificado es un espacio en blanco según Java
<i>public static char toLowerCase(char ch)</i>	Convierte el argumento a minúscula
<i>public static char toUpperCase(char ch)</i>	Convierte el argumento a mayúscula

Integer

Clase envoltorio del tipo primitivo **int**.

<i>public static final int MIN_VALUE</i>	Constante que contiene el valor mínimo que puede tener un int
<i>public static final int MAX_VALUE</i>	Constante que contiene el valor máximo que puede tener un int
<i>public static String toString(int i)</i>	Devuelve un objeto String que representa el entero especificado
<i>public static String toBinaryString(int i)</i>	Devuelve una cadena representando el entero recibido en base 2
<i>public static String toHexString(int i)</i>	Devuelve una cadena representando el entero recibido en base 16
<i>public static Integer valueOf(String s)</i>	Devuelve un objeto Integer que contiene el valor de la cadena especificada
<i>public static int parseInt(String s)</i>	Devuelve el valor entero representado por el argumento

Float

Clase envoltorio del tipo primitivo float.

<i>public static final float MIN_VALUE</i>	Constante que contiene el valor mínimo que puede tener un float
<i>public static final float MAX_VALUE</i>	Constante que contiene el valor máximo que puede tener un float
<i>public static String toString(float f)</i>	Devuelve un objeto String que representa el float especificado
<i>public static Float valueOf(String s)</i>	Devuelve un objeto Float que representa el valor de la cadena especificada
<i>public static float parseFloat(String s)</i>	Devuelve el valor float representado por el argumento

Boolean

Clase envoltorio del tipo primitivo boolean.

<i>public static final Boolean FALSE</i>	Objeto Boolean correspondiente al valor primitivo FALSE
<i>public static final Boolean TRUE</i>	Objeto Boolean correspondiente al valor primitivo TRUE
<i>public static String toString(boolean b)</i>	Devuelve un objeto String que representa el boolean especificado
<i>public static Boolean valueOf(String s)</i>	Devuelve un objeto Boolean que representa el valor de la cadena especificada
<i>public static boolean parseBoolean(String s)</i>	Devuelve el valor boolean representado por el argumento

Package java.util (Module java.base)

Contiene una miscelánea de clases útiles.

Scanner

Random

StringTokenizer

StringJoiner

Locale

TimeZone

Scanner

Rompe la entrada en tokens utilizando un patrón delimitador, que por defecto coincide con el espacio en blanco. Los tokens resultantes pueden ser convertidos en valores de diferentes tipos.

<i>public int nextInt()</i>	Analiza el siguiente token de la entrada como un int
<i>public float nextFloat()</i>	Analiza el siguiente token de la entrada como un float
<i>public boolean hasNextInt()</i>	Devuelve true si el siguiente token de la entrada se puede interpretar como un valor entero
<i>public String nextLine()</i>	Devuelve el resto de la línea
<i>public String next()</i>	Devuelve el siguiente token completo de la entrada
<i>public Scanner useDelimiter(String pattern)</i>	Establece el patrón de delimitación de este escáner a un modelo construido a partir del atributo pattern
<i>public Scanner reset()</i>	Restablece el escáner

Random

Una instancia de esta clase se utiliza para generar números pseudoaleatorios.

<i>public int nextInt()</i>	Devuelve el siguiente número entero de forma aleatoria
<i>public int nextInt(int bound)</i>	Devuelve un número comprendido entre 0 (incluido) y el parámetro bound (excluido) de forma aleatoria
<i>public float nextFloat()</i>	Devuelve un float comprendido entre 0.0f (incluido) y 1.0f (excluido) de forma aleatoria

StringTokenizer

Permite romper una cadena de caracteres en tokens.

public boolean hasMoreTokens()

Comprueba si hay más tokens

public String nextToken()

Devuelve el siguiente token

public int countTokens()

Devuelve el número de tokens

StringJoiner

Permite unir varias cadenas separadas por un delimitador.

<i>public StringJoiner add(CharSequence cadena)</i>	Añade la cadena especificada
<i>public int length()</i>	Devuelve el número de caracteres
<i>public StringJoiner setEmptyValue(CharSequence cadena)</i>	Establece la cadena que se devolverá cuando esté vacío

Locale

Representa una región geográfica, política o cultural específica.

<i>public static Locale getDefault()</i>	Devuelve la configuración regional predeterminada para esta instancia de la máquina virtual Java.
<i>public static void setDefault(Locale newLocale)</i>	Establece la configuración regional predeterminada para esta instancia de la máquina virtual Java.
<i>public String getLanguage()</i>	Devuelve el código de idioma
<i>public String getCountry()</i>	Devuelve el código de país/región

TimeZone

Representa una zona horaria.

<i>public static TimeZone getDefault()</i>	Devuelve la zona horaria predeterminada de la máquina virtual Java.
<i>public static void setDefault(TimeZone zone)</i>	Establece la zona horaria predeterminada de la máquina virtual Java.
<i>public static TimeZone getTimeZone(String ID)</i>	Devuelve la zona horaria para el ID especificado
<i>public static TimeZone getTimeZone(ZoneId zoneId)</i>	Devuelve la zona horaria para el zoneId especificado
<i>public ZoneId toZoneId()</i>	Convierte este objeto TimeZone en un ZoneId.

Package java.time (Module java.base)

Contiene clases relacionadas con fecha y hora. A partir de Java 8.

LocalDate

LocalTime

LocalDateTime

ZoneId

ZonedDateTime

ZoneOffset

OffsetDateTime

Instant

Period

Duration

LocalDate

Una fecha sin zona horaria en el sistema de calendario ISO-8601, como 2007-12-03.

<i>public static LocalDate now()</i>	Devuelve una instancia de LocalDate con la fecha actual
<i>public static LocalDate now(ZoneId zone)</i>	Devuelve una instancia de LocalDate con la fecha actual del sistema en la zona horaria especificada
<i>public static LocalDate of(int year, int month, int dayOfMonth)</i>	Devuelve una instancia de LocalDate con los datos pasados en los argumentos

LocalTime

Una hora sin zona horaria en el sistema de calendario ISO-8601, como 10:15:30.

<i>public static LocalTime now()</i>	Devuelve una instancia de LocalTime con la hora actual
<i>public static LocalTime now(ZoneId zone)</i>	Devuelve una instancia de LocalTime con la hora actual del sistema en la zona horaria especificada
<i>public static LocalTime of(int hour, int minute, int second)</i>	Devuelve una instancia de LocalTime con los datos pasados en los argumentos

LocalDateTime

Una fecha y hora sin una zona horaria en el sistema de calendario ISO-8601, como 2007-12-03T10: 15: 30.

```
public static LocalDateTime now()
```

Devuelve una instancia de LocalDateTime con la fecha y hora de la zona horaria predeterminada

```
public static LocalDateTime now(ZoneId zone)
```

Devuelve una instancia de LocalDateTime con la fecha y hora de la zona horaria especificada

ZoneId

Un ID de zona horaria, como Europe/Paris.

```
public static ZoneId systemDefault()
```

Devuelve la zona horaria predeterminada del sistema

```
public static ZoneId of(String zoneId)
```

Devuelve una instancia de ZoneId a partir de un ID, lo que garantiza que el ID sea válido y esté disponible para su uso.

ZonedDateTime

Una fecha y hora con una zona horaria en el sistema de calendario ISO-8601, como 2007-12-03T10:15:30+01:00 Europa / París.

<i>public static ZonedDateTime now()</i>	Devuelve la fecha y hora actual del reloj del sistema en la zona horaria predeterminada.
<i>public static ZonedDateTime now(ZoneId zone)</i>	Devuelve la fecha y hora actual del reloj del sistema en la zona horaria especificada.
<i>public static ZonedDateTime of(LocalDateTime localDateTime, ZoneId zone)</i>	Devuelve una instancia de ZonedDateTime de una fecha y hora local.
<i>public static ZonedDateTime ofInstant(Instant instant, ZoneId zone)</i>	Devuelve una instancia de ZonedDateTime de un Instant.
<i>public LocalDateTime toLocalDateTime()</i>	Devuelve una instancia de LocalDateTime.

ZoneOffset

Un desplazamiento de zona horaria de Greenwich/ UTC, como +02: 00.

public static ZoneOffset of(String offsetId)

Devuelve una instancia de ZoneOffset usando el ID.

public static ZoneOffset ofHours(int hours)

Devuelve una instancia de ZoneOffset usando un desplazamiento en horas.

OffsetDateTime

Una fecha y hora con un desfase de UTC / Greenwich en el sistema de calendario ISO-8601, como 2007-12-03T10: 15: 30 + 01: 00.

Se pretende que ZonedDateTime o Instant se utilicen para modelar datos en aplicaciones más sencillas. Esta clase puede usarse cuando se modelan conceptos de fecha y hora con más detalle, o cuando se comunica con una base de datos o en un protocolo de red.

<i>public static OffsetDateTime now()</i>	Devuelve la fecha y hora actual del reloj del sistema en la zona horaria predeterminada.
<i>public static OffsetDateTime now(ZoneId zone)</i>	Devuelve la fecha y hora actual del reloj del sistema en la zona horaria especificada.
<i>public static OffsetDateTime of(LocalDateTime dateTime, ZoneOffset offset)</i>	Devuelve una instancia de OffsetDateTime a partir de una fecha y hora y un desplazamiento

Instant

Un momento en la línea de tiempo en UTC (Coordinated Universal Time).

Podría usarse para registrar marcas de tiempo de eventos en la aplicación.

```
public static Instant now()
```

Devuelve una instancia de `Instant` con el momento actual del reloj del sistema

```
public ZonedDateTime atZone(ZoneId zone)
```

Devuelve una instancia de `ZonedDateTime` combinando este instante con una zona horaria

Period

Una cantidad de tiempo basada en la fecha en el sistema de calendario ISO-8601, como '2 años, 3 meses y 4 días'.

Se utiliza para modificar valores de una fecha u obtener la diferencia entre dos fechas.

```
public static Period between(LocalDate  
startDateInclusive, LocalDate endDateExclusive)
```

Devuelve un período que consiste en el número de años, meses y días entre dos fechas.

```
public static Period of(int years, int months,  
int days)
```

Devuelve un período que representa un número de años, meses y días.

Duration

La clase Duración representa un intervalo de tiempo en segundos o nanosegundos y es más adecuada para manejar períodos de tiempo más cortos, en casos que requieren más precisión.

public static Duration between(Temporal startInclusive, Temporal endExclusive)

Devuelve una instancia de Duration que representa la duración entre tiempos (Instant, LocalDateTime, OffsetDateTime, ZonedDateTime,..)

public static Duration of(long amount, TemporalUnit unit)

Devuelve una instancia de Duration que representa una cantidad en la unidad especificada (las unidades más utilizadas se definen en ChronoUnit).

Package java.io (Module java.base)

Contiene las clases relacionadas con entrada y salida.

BufferedReader

Console

File

BufferedReader

Permite leer caracteres de una entrada.

<code>public int read()</code>	Lee un carácter de la entrada
<code>public String readLine()</code>	Lee una línea de texto de la entrada

Console

Para la entrada y salida de datos por la línea de comandos.

<i>public String readLine()</i>	Lee una línea de texto desde la consola
<i>public char[] readPassword()</i>	Lee una password desde la consola
<i>public Console printf(String format, Object... args)</i>	Escribe una cadena en la consola con el formato especificado

Sintaxis del formato:

***%[argument_index\$][flags][width]
[.precision]conversion***

File

Representa un fichero o un directorio del sistema de archivos.

<code>public boolean exists()</code>	Comprueba si el archivo o directorio existe
<code>public String getName()</code>	Devuelve el nombre del archivo o directorio
<code>public boolean isDirectory()</code>	Comprueba si se trata de un directorio
<code>public boolean isFile()</code>	Comprueba si se trata de un archivo
<code>public boolean canRead()</code>	Comprueba si el archivo se puede leer
<code>public boolean canWrite()</code>	Comprueba si el archivo se puede modificar

Module java.desktop

Contiene entre otros los paquetes:

java.awt **Contiene clases útiles para crear interfaces gráficas**

javax.swing **Amplia y mejora el paquete java.awt**

JOptionPane

JOptionPane

Permite mostrar un cuadro de diálogo estándar para solicitar o mostrar información al usuario.

```
public static void showMessageDialog(Component parentComponent, Object message, String title, int messageType)
```

```
public static String showInputDialog(Component parentComponent, Object message, String title, int messageType)
```

```
public static int showConfirmDialog(Component parentComponent, Object message, String title, int optionType, int messageType)
```

Valores para el atributo *messageType*: **Valores para el atributo *optionType*:**

JOptionPane.ERROR_MESSAGE

JOptionPane.YES_NO_OPTION

JOptionPane.INFORMATION_MESSAGE

JOptionPane.YES_NO_CANCEL_OPTION

JOptionPane.WARNING_MESSAGE

JOptionPane.OK_CANCEL_OPTION

JOptionPane.QUESTION_MESSAGE

JOptionPane.PLAIN_MESSAGE

Javadoc

Utilidad para la generación de documentación de API's en formato HTML a partir de código fuente Java.

La mayoría de los IDE's utilizan javadoc para generar automáticamente la documentación de clases.

En la documentación deberemos incluir:

nombre de la clase, descripción, versión y autores.

Por cada constructor y método:

nombre, tipo de retorno, nombres y tipos de parámetros, descripción general, descripción de parámetros y descripción del retorno.

La documentación para javadoc debe ir encerrada entre los símbolos de comentario `/` y `*/`.**

La ubicación del comentario indica a javadoc que es lo documentado.

Si el comentario aparece antes de la declaración de clase se considerará un comentario de clase.

Si el comentario aparece antes de un constructor o un método se considerará un comentario de constructor o de método.

Javadoc

Para generar API's con Javadoc han de usarse etiquetas HTML o ciertas palabras reservadas precedidas del carácter @.

@author nombre del autor

@deprecated indica que el método o la clase es obsoleto

@param definición de un parámetro de un método

@return informa de lo que devuelve un método

@see asocia con otro método o clase

@throws excepción lanzada por el método

@version versión del método o clase

@since versión desde la que existe el método

Javadoc

Par indicar palabras claves y nombres de la API:

`{@code nombre}`

Para añadir enlaces en los nombres de la API:

`{@link enlace}`

Para generar la documentación de una clase:

`javadoc Clase.java`

Javadoc

```
/**
 * The {@code Integer} class wraps a value of the primitive type
 * {@code int} in an object. An object of type {@code Integer}
 * contains a single field whose type is {@code int}.
 *
 * <p>In addition, this class provides several methods for converting
 * an {@code int} to a {@code String} and a {@code String} to an
 * {@code int}, as well as other constants and methods useful when
 * dealing with an {@code int}.
 *
 * <p>Implementation note: The implementations of the "bit twiddling"
 * methods (such as {@link #highestOneBit(int) highestOneBit} and
 * {@link #numberOfTrailingZeros(int) numberOfTrailingZeros}) are
 * based on material from Henry S. Warren, Jr.'s <i>Hacker's
 * Delight</i>, (Addison Wesley, 2002).
 *
 * @author Lee Boynton
 * @author Arthur van Hoff
 * @author Josh Bloch
 * @author Joseph D. Darcy
 * @since 1.0
 */
public final class Integer extends Number implements Comparable<Integer> {
```

Javadoc

```
public final class Integer  
extends Number  
implements Comparable<Integer>
```

The `Integer` class wraps a value of the primitive type `int` in an object. An object of type `Integer` contains a single field whose type is `int`.

In addition, this class provides several methods for converting an `int` to a `String` and a `String` to an `int`, as well as other constants and methods.

Implementation note: The implementations of the "bit twiddling" methods (such as `highestOneBit` and `numberOfTrailingZeros`) are based on the algorithms in *Computer Systems: A Programmer's Perspective* (Addison Wesley, 2002).

Since:

1.0

Javadoc

```
/**
 * A constant holding the minimum value an {@code int} can
 * have,  $-2^{31}$ .
 */
@Native public static final int    MIN_VALUE = 0x80000000;

/**
 * A constant holding the maximum value an {@code int} can
 * have,  $2^{31}-1$ .
 */
@Native public static final int    MAX_VALUE = 0x7fffffff;
```


MIN_VALUE

```
@Native  
public static final int MIN_VALUE
```

A constant holding the minimum value an `int` can have, -2^{31} .

See Also:

[Constant Field Values](#)

MAX_VALUE

```
@Native  
public static final int MAX_VALUE
```

A constant holding the maximum value an `int` can have, $2^{31}-1$.

See Also:

[Constant Field Values](#)

Javadoc

```
/**
 * Parses the string argument as a signed decimal integer. The
 * characters in the string must all be decimal digits, except
 * that the first character may be an ASCII minus sign {@code '-'}
 * ({@code '\u005Cu002D'}) to indicate a negative value or an
 * ASCII plus sign {@code '+'} ({@code '\u005Cu002B'}) to
 * indicate a positive value. The resulting integer value is
 * returned, exactly as if the argument and the radix 10 were
 * given as arguments to the {@link #parseInt(java.lang.String,
 * int)} method.
 *
 * @param s      a {@code String} containing the {@code int}
 *                representation to be parsed
 * @return       the integer value represented by the argument in decimal.
 * @exception   NumberFormatException if the string does not contain a
 *                parsable integer.
 */
public static int parseInt(String s) throws NumberFormatException {
```

parseInt

`public static int parseInt(String s) throws NumberFormatException`

Parses the string argument as a signed decimal integer. The characters in the string indicate a negative value or an ASCII plus sign '+' ('\u002B') to indicate a positive value. See the Javadoc for the `parseInt(java.lang.String, int)` method.

Parameters:

`s` - a `String` containing the `int` representation to be parsed

Returns:

the integer value represented by the argument in decimal.

Throws:

`NumberFormatException` - if the string does not contain a parsable integer.