

INGENIERÍA MECATRÓNICA



DI_CERO

DIEGO CERVANTES RODRÍGUEZ

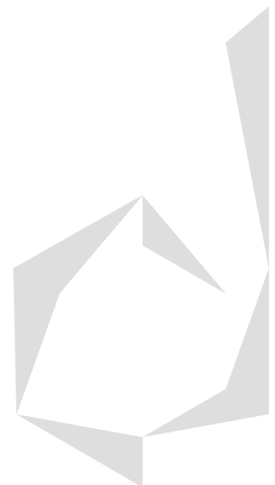
JAVA 8

APACHE NETBEANS IDE 12.5

Clases Predefinidas en Java

Contenido

Clases Predefinidas en Java.....	2
Clase Predefinida Math.....	2
Refundiciones.....	5
Clase Predefinida String.....	6
Clase Predefinida Scanner.....	8
Clase Predefinida JOptionPane.....	11
Clase Predefinida Integer.....	13
Clase Predefinida Double.....	14
Referencias:	15



Clases Predefinidas en Java

Veremos los usos de algunas clases predefinidas en Java como la clase Math, String, Array y Scanner.



Para poder ver la documentación de las clases predefinidas debo meterme a la API y en el lado izquierdo de la pantalla donde dice All Classes estarán enlistadas todas las clases en orden alfabético.

Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	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.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
java.awt.im	Provides classes and interfaces for the input method framework.
java.awt.im.spi	Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.
java.awt.image	Provides classes for creating and modifying images.
java.awt.image.renderable	Provides classes and interfaces for producing rendering-independent images.
java.awt.print	Provides classes and interfaces for a general printing API.
java.beans	Contains classes related to developing beans -- components based on the JavaBeans™ architecture.
java.beans.beancontext	Provides classes and interfaces relating to bean context.

Clase Predefinida Math

La clase Math tiene muchos métodos para hacer cálculos matemáticos, en este documento veremos algunos, pero si queremos consultarlos a fondo debemos dirigirnos a la API, buscar la clase Math en

donde dice All Classes (las clases están enlistadas en orden alfabético) y dar clic en la clase que quiero ver. Esta clase pertenece al paquete principal java.lang, por lo que no debe ser importado.

The screenshot shows the Java Platform Standard Ed. 7 IDE. On the left, the 'All Classes' list is visible, with 'Math' highlighted. The main area displays the 'Class Math' page, which includes the package 'java.lang', the class 'Math', and its inheritance from 'Object'. The description states that the class contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions. It also mentions that unlike some of the numeric methods of class `StrictMath`, all implementations of the equivalent functions of class `Math` are not defined to return the bit-for-bit same results. This relaxation permits better-performing implementations where strict reproducibility is not required.

CONSTRUCTOR: La clase Math no tiene ningún método constructor.

MÉTODOS: Dentro de las clases predefinidas los métodos son los que puedo usar para que ejecuten alguna acción dentro de nuestro código. Los métodos están enlistados en orden alfabético.

The screenshot shows the Java Platform Standard Ed. 7 IDE. On the left, the 'All Classes' list is visible, with 'Math' highlighted. The main area displays the 'Method Summary' page for the class `Math`. The table lists various static methods, including `abs`, `acos`, `asin`, `atan`, `atan2`, `cbrt`, `ceil`, `copySign`, and `cos`.

Modifier and Type	Method and Description
static double	<code>abs(double a)</code> Returns the absolute value of a double value.
static float	<code>abs(float a)</code> Returns the absolute value of a float value.
static int	<code>abs(int a)</code> Returns the absolute value of an int value.
static long	<code>abs(long a)</code> Returns the absolute value of a long value.
static double	<code>acos(double a)</code> Returns the arc cosine of a value; the returned angle is in the range 0.0 through pi.
static double	<code>asin(double a)</code> Returns the arc sine of a value; the returned angle is in the range -pi/2 through pi/2.
static double	<code>atan(double a)</code> Returns the arc tangent of a value; the returned angle is in the range -pi/2 through pi/2.
static double	<code>atan2(double y, double x)</code> Returns the angle theta from the conversion of rectangular coordinates (x, y) to polar coordinates (r, theta).
static double	<code>cbrt(double a)</code> Returns the cube root of a double value.
static double	<code>ceil(double a)</code> Returns the smallest (closest to negative infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer.
static double	<code>copySign(double magnitude, double sign)</code> Returns the first floating-point argument with the sign of the second floating-point argument.
static float	<code>copySign(float magnitude, float sign)</code> Returns the first floating-point argument with the sign of the second floating-point argument.
static double	<code>cos(double a)</code> Returns the trigonometric cosine of an angle.

- **Math.sqrt(numero):** Con este método obtenemos la raíz cuadrada de un número.

`static double`

`sqrt(double a)`

Returns the correctly rounded positive square root of a double value.

En la documentación de la API podemos ver que el método recibe como parámetro un ángulo de tipo double y regresa otro número de tipo double. Es un método estático.

Podemos ver que en la descripción del **método** se dice que es **estático** (static), esto implica que para poder usarlo no se debe crear ningún objeto de la clase Math, simplemente se debe usar el nombre de la clase delante del método:

nombreDeLaClase.metodo_estático;

Si por cualquier razón quiero saber más del método puedo dar clic en el nombre del método para que me mande a otra página que me dé más información.

sqrt

```
public static double sqrt(double a)
```

Returns the correctly rounded positive square root of a double value. Special cases:

- If the argument is NaN or less than zero, then the result is NaN.
- If the argument is positive infinity, then the result is positive infinity.
- If the argument is positive zero or negative zero, then the result is the same as the argument.

Otherwise, the result is the double value closest to the true mathematical square root of the argument value.

Parameters:

a - a value.

Returns:

the positive square root of a. If the argument is NaN or less than zero, the result is NaN.

- **Math.pow(base, exponente):** Para obtener un número base elevado a cierta potencia. Tanto la base como el exponente deben ser números decimales tipo double.

```
static double
```

```
pow(double a, double b)
```

Returns the value of the first argument raised to the power of the second argument.

En la documentación de la API podemos ver que el método recibe como parámetro dos números de tipo double y regresa otro número de tipo double. Es un método estático.

- **Math.sin(angulo):** Se usa para obtener el seno de cierto ángulo dado en radianes.

```
static double
```

```
sin(double a)
```

Returns the trigonometric sine of an angle.

En la documentación de la API podemos ver que el método recibe como parámetro un ángulo dado en radianes y de tipo double, como resultado regresa otro número de tipo double. Es un método estático.

- **Math.cos(angulo):** Se usa para obtener el coseno de cierto ángulo dado en radianes.

```
static double
```

```
cos(double a)
```

Returns the trigonometric cosine of an angle.

En la documentación de la API podemos ver que el método recibe como parámetro un ángulo dado en radianes y de tipo double, como resultado regresa otro número de tipo double. Es un método estático.

- **Math.atan(angulo):** Se usa para obtener el arco tangente de cierto número, la tangente inversa lo que regresará es un ángulo.

```
static double
```

```
atan(double a)
```

Returns the arc tangent of a value; the returned angle is in the range $-\pi/2$ through $\pi/2$.

En la documentación podemos ver que el método recibe como parámetro un número de tipo double y regresa otro número de tipo double. Es un método estático.

- **Math.round(número_decimal)**: Se usa para redondear un número decimal.

<code>static long</code>	<code>round(double a)</code> Returns the closest <code>long</code> to the argument, with ties rounding up.
<code>static int</code>	<code>round(float a)</code> Returns the closest <code>int</code> to the argument, with ties rounding up.

En la documentación podemos ver que hay dos métodos `round`, usaremos uno u otro dependiendo del tipo de dato que pongamos como su parámetro, un parámetro de tipo `double` regresará un número redondeado de tipo `long` y un parámetro de tipo `float` regresará un número redondeado de tipo `int`. Ambos son métodos estáticos.

- **Math.random(número_decimal)**: Se usa para que el programa cree un número aleatorio entre el 0 y 1.

<code>static double</code>	<code>random()</code> Returns a <code>double</code> value with a positive sign, greater than or equal to 0.0 and less than 1.0.
----------------------------	--

En la documentación podemos ver que el método no recibe nada como parámetro y retorna un número de tipo `double`. Es un método estático.

CONSTANTES: En las clases predefinidas también podemos encontrar constantes ya guardadas como por ejemplo `pi`, la constante de Euler, entre otras. Dentro de la documentación podremos encontrar datos adicionales.

- **Math.PI**: Constante de clase con el número π , en la documentación de abajo se indica que esta constante es de tipo `double`.
- **Math.E**: Número de Euler, en la documentación de abajo se indica que esta constante también es de tipo `double`.

Field Summary

Fields	
Modifier and Type	Field and Description
<code>static double</code>	<code>E</code> The <code>double</code> value that is closer than any other to e , the base of the natural logarithms.
<code>static double</code>	<code>PI</code> The <code>double</code> value that is closer than any other to π , the ratio of the circumference of a circle to its diameter.

Ambas son constantes estáticas, esto implica que para usarlas solo se debe usar el nombre de la clase.

Refundiciones

Esto no tiene que ver con la clase `Math` pero es de gran uso porque se utiliza para transformar un tipo de dato numérico en otro diferente.

`tipo_primitivo_1 variable = valor;`

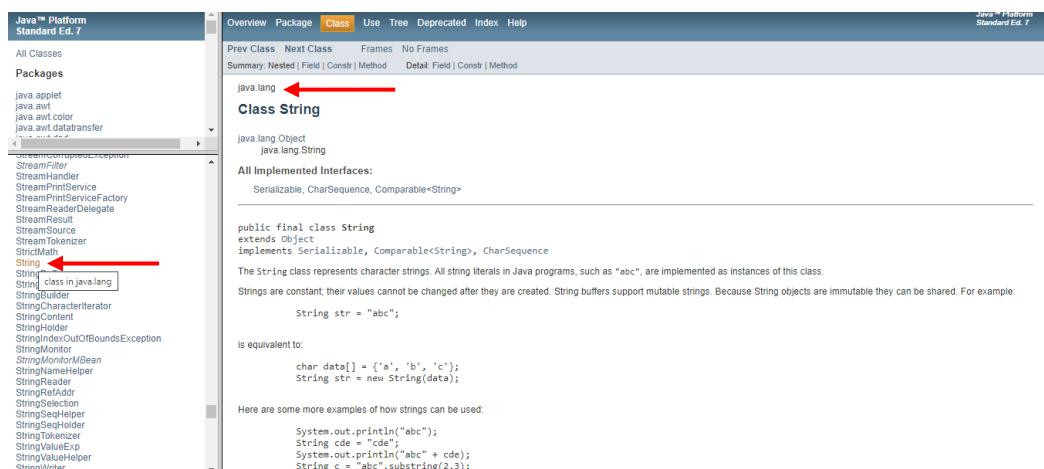
`tipo_primitivo_2 nombre_de_la_variable_transformada = (tipo_primitivo_2) variable;`

Ejemplo: El método round regresará un tipo de dato long si es que su parámetro es de tipo double, por medio de la refundición este dato puede ser cambiado a tipo int. El problema con hacer esto es que, si la variable num1 es un número muy grande, debido a que el tipo de dato int permite un número menor de datos, se perderá información.

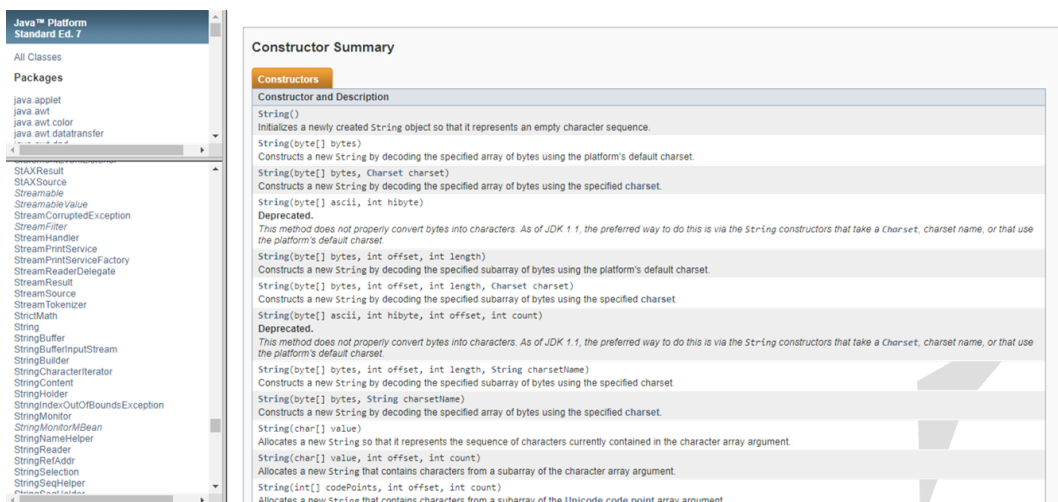
```
int raiz=(int)Math.round(num1);
```

Clase Predefinida String

En Java las palabras no son un tipo de dato, son más bien un objeto de la clase predefinida String, esto se maneja así para que con los distintos métodos de esta clase predefinida pueda manipular o analizar cada palabra que cree. **Esta clase pertenece al paquete principal java.lang, por lo que no debe ser importado.**



CONSTRUCTOR: Recordemos que el método constructor es aquel que tiene el mismo nombre de la clase. La clase String si tiene métodos constructores (que se diferencian entre sí por el tipo de parámetro que recibe cada uno, debido a que todos tienen el mismo nombre) aunque para los métodos que veremos no usaremos ninguno.



Java™ Platform
Standard Ed. 7

All Classes

Packages

java.applet

java.awt

java.awt.color

java.awt.datatransfer

java.awt.dnd

java.io

java.lang

java.lang.annotation

java.lang.invoke

java.lang.management

java.lang.module

java.lang.ref

java.lang.reflect

java.lang.runtime

java.lang.security

java.lang.string

java.lang.system

java.lang.thread

java.lang.util

java.net

java.nio

java.security

java.sql

java.time

java.util

java.util.concurrent

java.util.logging

java.util.regex

java.util.zip

javax.swing

javax.swing.text

javax.swing.text.html

javax.swing.text.rtf

javax.xml

javax.xml.parsers

javax.xml.xpath

javax.xml.ws

javax.xml.ws.handler

javax.xml.ws.soap

javax.xml.ws.soap.http

javax.xml.ws.soap.mex

javax.xml.ws.soap.mex.binding

javax.xml.ws.soap.mex.binding.binding

javax.xml.ws.soap.mex.binding.binding.binding

javax.xml.ws.soap.mex.binding.binding.binding.binding

javax.xml.ws.soap.mex.binding.binding.binding.binding.binding

javax.xml.ws.soap.mex.binding.binding.binding.binding.binding.binding

javax.xml.ws.soap.mex.binding.binding.binding.binding.binding.binding.binding

javax.xml.ws.soap.mex.binding.binding.binding.binding.binding.binding.binding.binding

javax.xml.ws.soap.mex.binding.binding.binding.binding.binding.binding.binding.binding.binding

javax.xml.ws.soap.mex.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding

javax.xml.ws.soap.mex.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding

javax.xml.ws.soap.mex.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding

javax.xml.ws.soap.mex.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding

javax.xml.ws.soap.mex.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding

javax.xml.ws.soap.mex.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding

javax.xml.ws.soap.mex.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding

javax.xml.ws.soap.mex.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding

javax.xml.ws.soap.mex.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding

javax.xml.ws.soap.mex.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding

javax.xml.ws.soap.mex.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding

javax.xml.ws.soap.mex.binding

javax.xml.ws.soap.mex.binding

javax.xml.ws.soap.mex.binding

javax.xml.ws.soap.mex.binding

javax.xml.ws.soap.mex.binding

javax.xml.ws.soap.mex.binding

javax.xml.ws.soap.mex.binding

javax.xml.ws.soap.mex.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding

javax.xml.ws.soap.mex.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding.binding

Method Summary

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.
String	concat(String str) Concatenates the specified string to the end of this string.
boolean	contains(CharSequence s) Returns true if and only if this string contains the specified sequence of char values.
boolean	contentEquals(CharSequence cs) Compares this string to the specified CharSequence.
boolean	contentEquals(StringBuffer sb) Compares this string to the specified StringBuffer.
static String	copyValueOf(char[] data) Returns a String that represents the character sequence in the array specified.
static String	copyValueOf(char[] data, int offset, int count) Returns a String that represents the character sequence in the array specified.
boolean	endsWith(String suffix) Tests if this string ends with the specified suffix.

- | | |
|-----|------------------------------------|
| int | length() |
| | Returns the length of this string. |

char	charAt(int index) Returns the char value at the specified index.
------	---

En la documentación de la API podemos ver que el método recibe como parámetro un número de tipo `int` que le indique al método la ubicación de la letra que quiero extraer del objeto `String` y regresa un tipo de dato `char` (caracter o letra), al igual que en el caso anterior, el método se aplica por medio de poner un punto después del nombre del objeto `String` al que lo quiera aplicar. Es un método NO estático.

- **`.substring(posición1, posición2)`**: Se usa para extraer solo una parte de la palabra (objeto `String`) a la que se le esté aplicando el método, existen dos métodos con el mismo nombre y se aplicará uno u otro dependiendo del número de parámetros que le introduzca. Las posiciones se empiezan a contar desde cero.

<code>String</code>	<code>substring(int beginIndex)</code> Returns a new string that is a substring of this string.
<code>String</code>	<code>substring(int beginIndex, int endIndex)</code> Returns a new string that is a substring of this string.

En la documentación de la API podemos ver que el método recibe como parámetro la posición de inicio o fin desde donde quiero extraer una parte de esa palabra, ambas posiciones son de tipo `int`. Si se usa una sola posición de la palabra se extraerá desde ahí hasta el final de la palabra y si se pone el inicio y fin se extraerá solo de ese rango. Ambos son métodos NO estáticos.

- **`.equals(objeto_String)`**: Este método se usa para ver si dos palabras (osea objetos `String`) son iguales tomando en cuenta mayúsculas y minúsculas.

<code>boolean</code>	<code>equals(Object anObject)</code> Compares this string to the specified object.
----------------------	---

En la documentación podemos ver que el método recibe como parámetro un objeto de la clase `String` y regresa un tipo de dato booleano `true` o `false`. Es un método NO estático.

- **`.equalsIgnoreCase(objeto_String)`**: Este método se usa para ver si dos palabras (osea objetos `String`) son iguales sin tomar en cuenta mayúsculas o minúsculas.

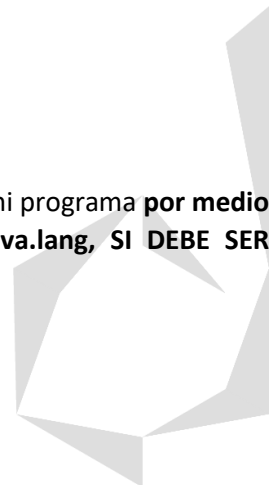
<code>boolean</code>	<code>equalsIgnoreCase(String anotherString)</code> Compares this String to another String, ignoring case considerations.
----------------------	--

En la documentación podemos ver que el método recibe como parámetro un objeto de la clase `String` y regresa un tipo de dato booleano `true` o `false`.

Es un método NO estático.

Clase Predefinida Scanner

Esta clase predefinida es muy importante porque me permite **introducir datos** a mi programa **por medio de la consola** del sistema. Esta clase **NO pertenece al paquete principal `java.lang`**, **SI DEBE SER IMPORTADO**. Pertenece al paquete `java.util`.



CONSTRUCTOR: Recordemos que el método constructor es aquel que tiene el mismo nombre de la clase. La clase Scanner tiene varios métodos constructores (que se diferencian entre sí por el tipo o número de parámetros que recibe cada uno, debido a que todos tienen el mismo nombre), dependiendo de los parámetros que le demos al constructor se construirá un objeto o instancia de la clase Scanner diferente o que tendrá un estado inicial dado.

- **Scanner(elemento_que_quiero_usar_para_introducir_datos):** Con este constructor se creará un objeto de tipo Scanner que va a proporcionar al programa un valor procedente del elemento que yo le indique como parámetro, este elemento usualmente es la consola o una parte de la interfaz gráfica.

Scanner(File source)
Constructs a new Scanner that produces values scanned from the specified file.

Scanner

```
public Scanner(File source)
    throws FileNotFoundException
```

Constructs a new Scanner that produces values scanned from the specified file. Bytes from the file are converted into characters using the underlying platform's default charset.

Parameters:

source - A file to be scanned

Throws:

FileNotFoundException - if source is not found

En el objeto Scanner que se cree se almacenará lo que el usuario introduzca por medio de lo que hayamos indicado en el parámetro del constructor.

- **System.in:** Se usa para especificar en el parámetro del constructor que la consola es el elemento por el cual queremos introducir datos a nuestro programa, osea que System.in lo debo poner como el File Source del constructor.

MÉTODOS: Los métodos de la clase Scanner sirven para indicarle al programa qué tipo de dato voy a introducir.

Modifier and Type	Method and Description
void	close() Closes this scanner.
Pattern	delimiter() Returns the Pattern this Scanner is currently using to match delimiters.
String	findInline(Pattern pattern) Attempts to find the next occurrence of the specified pattern ignoring delimiters.
String	findInline(String pattern) Attempts to find the next occurrence of a pattern constructed from the specified string, ignoring delimiters.
String	findWithinHorizon(Pattern pattern, int horizon) Attempts to find the next occurrence of the specified pattern.
String	findWithinHorizon(String pattern, int horizon) Attempts to find the next occurrence of a pattern constructed from the specified string, ignoring delimiters.
boolean	hasNext() Returns true if this scanner has another token in its input.
boolean	hasNext(Pattern pattern) Returns true if the next complete token matches the specified pattern.
boolean	hasNext(String pattern) Returns true if the next token matches the pattern constructed from the specified string.
boolean	hasNextBigDecimal() Returns true if the next token in this scanner's input can be interpreted as a BigDecimal using the nextBigDecimal() method.
boolean	hasNextBigInteger() Returns true if the next token in this scanner's input can be interpreted as a BigInteger in the default radix using the nextBigInteger() method.
boolean	hasNextBigInteger(int radix) Returns true if the next token in this scanner's input can be interpreted as a BigInteger in the specified radix using the nextBigInteger() method.
boolean	hasNextBoolean() Returns true if the next token in this scanner's input can be interpreted as a boolean value using a case insensitive pattern created from

- **.nextLine():** Con este método puedo introducir texto a mi programa.

String **nextLine()**
Advances this scanner past the current line and returns the input that was skipped.

En la documentación de la API podemos ver que el método no recibe como parámetro nada, pero se debe aplicar a un objeto de la clase String por medio de un punto y regresa un número de tipo int. Es un método NO estático.

La mayoría de los métodos de la clase Scanner NO son estáticos, esto implica que para usarlos debo crear un objeto de la clase Scanner y por medio de éste aplicar el método, para crear un objeto de cualquier clase debo usar alguno de sus constructores, recordando que cualquiera de los constructores siempre va a tener el mismo nombre que la clase, se diferenciarán entre ellos por el tipo de dato que reciben como parámetro o por el número de parámetros que reciben:

Nombre_clase nombre_objeto = new Constructor(parametro_1, parametro_2, ...);

En específico, la sintaxis por la cual se crea una instancia u objeto de la clase Scanner o cualquier otra que no sea la clase String es la siguiente:

```
Scanner nombre_objeto_Scanner = new Scanner(parametro_del_constructor);
```

El objeto o instancia de la clase predefinida se debe usar para aplicar un método NO estático.

```
nombre_objeto_Scanner.metodo_no_estático;
```

- **.nextInt()**: Con este método puedo introducir números enteros a mi programa.

int	nextInt() Scans the next token of the input as an int.
int	nextInt(int radix) Scans the next token of the input as an int.

En la documentación de la API podemos ver que el método recibe como parámetro un número de tipo int que le indique al método la ubicación de la letra que quiero extraer del objeto String y regresa un tipo de dato char (caracter o letra), al igual que en el caso anterior, el método se aplica por medio de poner un punto después del nombre del objeto String al que lo quiera aplicar.

Ambos son métodos **NO** estáticos.

- **.nextDouble()**: Con este método puedo introducir números decimales a mi programa.

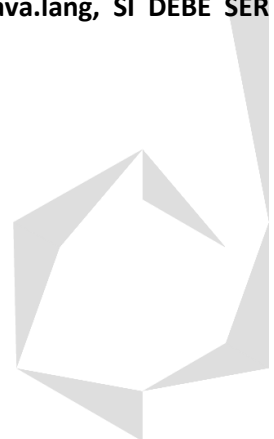
double	nextDouble() Scans the next token of the input as a double.
--------	--

En la documentación de la API podemos ver que el método recibe como parámetro la posición de inicio o fin desde donde quiero extraer una parte de esa palabra, ambas posiciones son de tipo int. Si se usa una sola posición de la palabra se extraerá desde ahí hasta el final de la palabra y si se pone el inicio y fin se extraerá solo de ese rango.

Es un método **NO** estático.

Clase Predefinida JOptionPane

Esta clase predefinida es muy importante porque me permite **introducir datos** a mi programa **por medio de una ventana emergente**. Esta clase **NO pertenece al paquete principal java.lang, SI DEBE SER IMPORTADO**. Pertenece al paquete java.swing.



Java™ Platform
Standard Ed. 7

All Classes

Packages

- java.applet
- java.awt
- java.awt.color
- java.awt.datatransfer
- java.awt.dnd
- java.awt.font
- java.awt.image
- java.awt.im
- java.awt.print
- java.beans
- java.io
- java.lang
- java.lang.annotation
- java.lang.invoke
- java.lang.management
- java.lang.module
- java.lang.ref
- java.lang.reflect
- java.lang.runtime
- java.lang.security
- java.lang.util
- java.math
- java.net
- java.nio
- java.rmi
- java.security
- java.security.cert
- java.security.interfaces
- java.sql
- java.text
- java.time
- java.util
- java.util.concurrent
- java.util.logging
- java.util.regex
- java.util.zip
- javax.accessibility
- javax.annotation
- javax.crypto
- javax.imageio
- javax.jmx
- javax.json
- javax.lang.model
- javax.management
- javax.net
- javax.net.ssl
- javax.naming
- javax.print
- javax.script
- javax.security.auth
- javax.security.interfaces
- javax.swing
- javax.swing.plaf
- javax.swing.text
- javax.swing.undo
- javax.xml
- javax.xml.crypto
- javax.xml.datatype
- javax.xml.parsers
- javax.xml.xpath

Overview **Package** Class Use Tree Depreciated Index Help

Prev Class Next Class Frames No Frames

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

javax.swing

Class JOptionPane

java.lang.Object
 java.awt.Component
 java.awt.Container
 javax.swing.JComponent
 javax.swing.JOptionPane

All Implemented Interfaces:
 ImageObserver, MenuContainer, Serializable, Accessible

```
public class JOptionPane
extends JComponent
implements Accessible
```

JOptionPane makes it easy to pop up a standard dialog box that prompts users for a value or informs them of something. For information about using JOptionPane, see [How to Make Dialogs](#), a section in *The Java Tutorial*.

While the JOptionPane class may appear complex because of the large number of methods, almost all uses of this class are one-line calls to one of the static `showXXXDialog` methods shown below:

Method Name	Description
<code>showConfirmDialog</code>	Asks a confirming question, like yes/no/cancel.
<code>showInputDialog</code>	Prompt for some input.
<code>showMessageDialog</code>	Tell the user about something that has happened.
<code>showOptionDialog</code>	The Grand Unification of the above three.

Each of these methods also comes in a `showInternalXXX` flavor, which uses an internal frame to hold the dialog box (see `JInternalFrame`). Multiple convenience methods have also been defined – overloaded versions of the basic methods that use different parameter lists.

Java™ Platform
Standard Ed. 7

All Classes

Packages

java.applet
java.awt
java.awt.color
java.awt.datatransfer

javax.swing

Method Summary

Modifier and Type	Method and Description
JDialog	createDialog(Component parentComponent, String title) Creates and returns a new JDialog wrapping this centered on the parentComponent in the parentComponent's frame.
JDialog	createDialog(String title) Creates and returns a new parentless JDialog with the specified title.
JInternalFrame	createInternalFrame(Component parentComponent, String title) Creates and returns an instance of JInternalFrame.
AccessibleContext	getAccessibleContext() Returns the AccessibleContext associated with this JOptionPane.
static JDesktopPane	getDesktopPaneForComponent(Component parentComponent) Returns the specified component's desktop pane.
static Frame	getFrameForComponent(Component parentComponent) Returns the specified component's Frame.
Icon	getIcon() Returns the icon this pane displays.
Object	getInitialSelectionValue() Returns the input value that is displayed as initially selected to the user.
Object	getInitialValue() Returns the initial value.
Object	getInputValue() Returns the value the user has input, if wantsInput is true.
int	getMaxCharactersPerLineCount() Returns the maximum number of characters to place on a line in a message.
Object	getMessage() Returns the message-object this pane displays.
int	getMessageType() Returns the message type.
Object[]	getOptions() Returns the choices the user can make.

static String	<code>showInputDialog(Component parentComponent, Object message)</code> Shows a question-message dialog requesting input from the user parented to parentComponent.
static String	<code>showInputDialog(Component parentComponent, Object message, Object initialSelectionValue)</code> Shows a question-message dialog requesting input from the user and parented to parentComponent.
static String	<code>showInputDialog(Component parentComponent, Object message, String title, int messageType)</code> Shows a dialog requesting input from the user parented to parentComponent with the dialog having the title title and message type messageType.
static Object	<code>showInputDialog(Component parentComponent, Object message, String title, int messageType, Icon icon, Object[] selectionValues, Object initialSelectionValue)</code> Prompts the user for input in a blocking dialog where the initial selection, possible selections, and all other options can be specified.
static String	<code>showInputDialog(Object message)</code> Shows a question-message dialog requesting input from the user.
static String	<code>showInputDialog(Object message, Object initialSelectionValue)</code> Shows a question-message dialog requesting input from the user, with the input value initialized to initialSelectionValue.

En la documentación de la API podemos ver que el método no recibe como parámetro nada, pero se debe aplicar a un objeto de la clase `String` por medio de un punto y regresa un número de tipo `int`. Todos son métodos estáticos.

En particular el método que aplicaremos más de todos los enlistados es el siguiente, ya que recibe como parámetro un objeto cualquiera que se mostrará como mensaje en la ventana emergente y retornará un `String`.

```
static String showInputDialog(Object message)
Shows a question-message dialog requesting input from the user.
```

Podemos ver que en la descripción del **método** se dice que es **estático** (static), esto implica que para poder usarlo no se debe crear ningún objeto de la clase `JOptionPane`, simplemente se debe usar el nombre de la clase delante del método:

```
nombreDeLaClase.metodo_estático;
```

Clase Predefinida Integer

Esta clase se suele usar como complemento de la clase `JOptionPane` y sirve principalmente para convertir cualquier tipo de dato en un objeto `Integer`, que es lo mismo a referirse a un tipo de dato int (osea entero). Pertenece al paquete principal `java.lang` por lo que no debe ser importado.

The screenshot shows the Java Platform Standard Ed. 7 API documentation for the `Class Integer`. The left sidebar lists various Java classes, with `Integer` highlighted. The main content area shows the class hierarchy: `java.lang.Object` and `java.lang.Number` are parents, and `java.lang.Integer` is the current class. It lists implemented interfaces: `Serializable` and `Comparable<Integer>`. The class declaration is shown: `public final class Integer extends Number implements Comparable<Integer>`. A description states: "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`." It also mentions that the class provides methods for converting an `int` to a `String` and a `String` to an `int`. An implementation note mentions "bit twiddling" methods based on material from Henry S. Warren, Jr.'s *Hacker's Delight*. The "Since" section indicates it was introduced in JDK 1.0. The "See Also" section lists `Serialized Form`. At the bottom, there is a "Field Summary" section with a "Fields" tab.

MÉTODOS: Los métodos de la clase Integer son los que hacen la conversión de String a int.

Modifier and Type	Method and Description
static int	bitCount(int i) Returns the number of one-bits in the two's complement binary representation of the specified int value.
byte	byteValue() Returns the value of this Integer as a byte.
static int	compare(int x, int y) Compares two int values numerically.
int	compareTo(Integer anotherInteger) Compares two Integer objects numerically.
static Integer	decode(String nm) Decodes a String into an Integer.
double	doubleValue() Returns the value of this Integer as a double.
boolean	equals(Object obj) Compares this object to the specified object.
float	floatValue() Returns the value of this Integer as a float.
static Integer	getInteger(String nm) Determines the integer value of the system property with the specified name.
static Integer	getInteger(String nm, int val) Determines the integer value of the system property with the specified name.
static Integer	getInteger(String nm, Integer val) Returns the integer value of the system property with the specified name.
int	hashCode() Returns a hash code for this Integer.
static int	highestOneBit(int i) Returns an int value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.
int	intValue() Returns the value of this Integer as an int.

- **Integer.parseInt():** Con este método convierto un String a un entero (int).

```
static int      parseInt(String s)
                Parses the string argument as a signed decimal integer.

static int      parseInt(String s, int radix)
                Parses the string argument as a signed integer in the radix specified by the second argument.
```

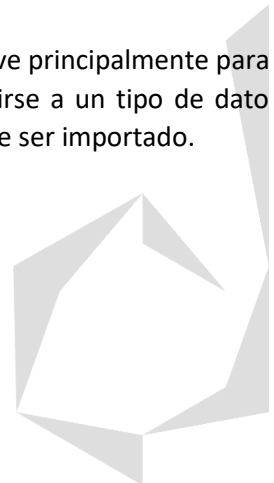
En la documentación de la API podemos ver que el método recibe como parámetro al objeto String y en el segundo argumento se puede indicar el sistema numérico que equivale al número recibido, este puede ser binario, hexadecimal, decimal, octal, etc. y respetará el signo con el que viene para retornar un dato tipo int (entero). El método es estático.

Podemos ver que en la descripción del **método** se dice que es **estático** (static), esto implica que para poder usarlo no se debe crear ningún objeto de la clase Integer, simplemente se debe usar el nombre de la clase delante del método:

nombreDeLaClase.metodo_estático;

Clase Predefinida Double

Esta clase también se suele usar como complemento de la clase JOptionPane y sirve principalmente para convertir cualquier tipo de dato en un objeto Double, que es lo mismo a referirse a un tipo de dato double (osea decimal). Pertenece al paquete principal **java.lang** por lo que no debe ser importado.



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 Double

java.lang.Object
java.lang.Number
java.lang.Double

All Implemented Interfaces:
Serializable, Comparable<Double>

public final class Double
extends Number
implements Comparable<Double>

The Double class wraps a value of the primitive type double in an object. An object of type Double contains a single field whose type is double.

In addition, this class provides several methods for converting a double to a String and a String to a double, as well as other constants and methods useful when dealing with a double.

Since:
JDK1.0

See Also:
Serialized Form

Field Summary

Fields

MÉTODOS: Los métodos de la clase Double son los que hacen la conversión de String a double.

Method Summary

Methods

Modifier and Type	Method and Description
byte	byteValue() Returns the value of this Double as a byte (by casting to a byte).
static int	compare(double d1, double d2) Compares the two specified double values.
int	compareTo(Double anotherDouble) Compares two Double objects numerically.
static long	doubleToLongBits(double value) Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout.
static long	doubleToRawLongBits(double value) Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout, preserving Not-a-Number (NaN) values.
double	doubleValue() Returns the double value of this Double object.
boolean	equals(Object obj) Compares this object against the specified object.
float	floatValue() Returns the float value of this Double object.
int	hashCode() Returns a hash code for this Double object.

- **Double.parseDouble():** Con este método convierto un String a un entero (int).

```
static double parseDouble(String s)
```

Returns a new double initialized to the value represented by the specified String, as performed by the valueOf method of class Double.

En la documentación de la API podemos ver que el método recibe como parámetro al objeto String y retorna un tipo de dato double. El método es estático.

Podemos ver que en la descripción del **método** se dice que es **estático** (static), esto implica que para poder usarlo no se debe crear ningún objeto de la clase Integer, simplemente se debe usar el nombre de la clase delante del método:

nombreDeLaClase.metodo_estático;

Referencias:

Pildoras Informáticas, "Curso Java", 2023 [Online], Available:
<https://www.youtube.com/playlist?list=PLU8oAlHdN5BktAXdEVCLUYzvDyqRQJ2Ik>