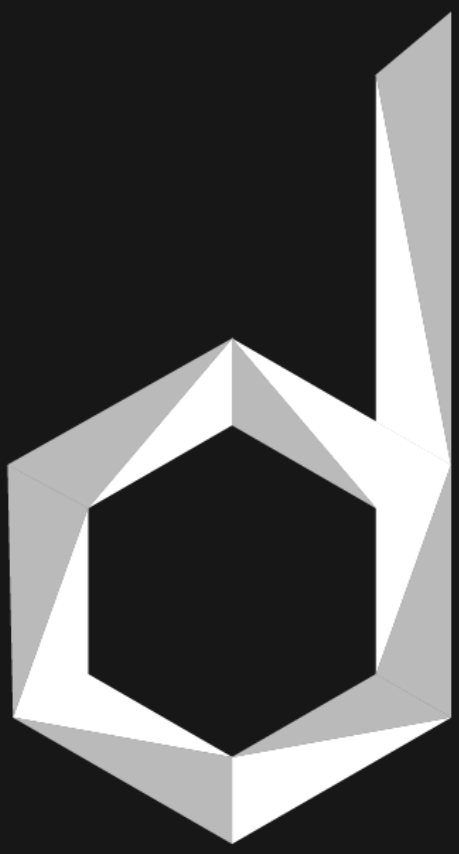
INGENIERÍA MECATRÓNICA



DI_CERO

DIEGO CERVANTES RODRÍGUEZ

JAVA 8

APACHE NETBEANS IDE 12.5

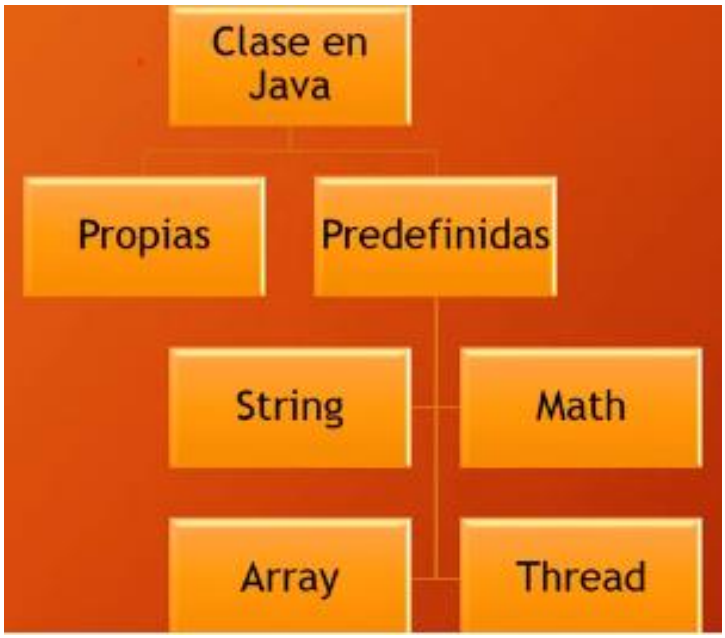
Clases Predefinidas en Java

Contenido

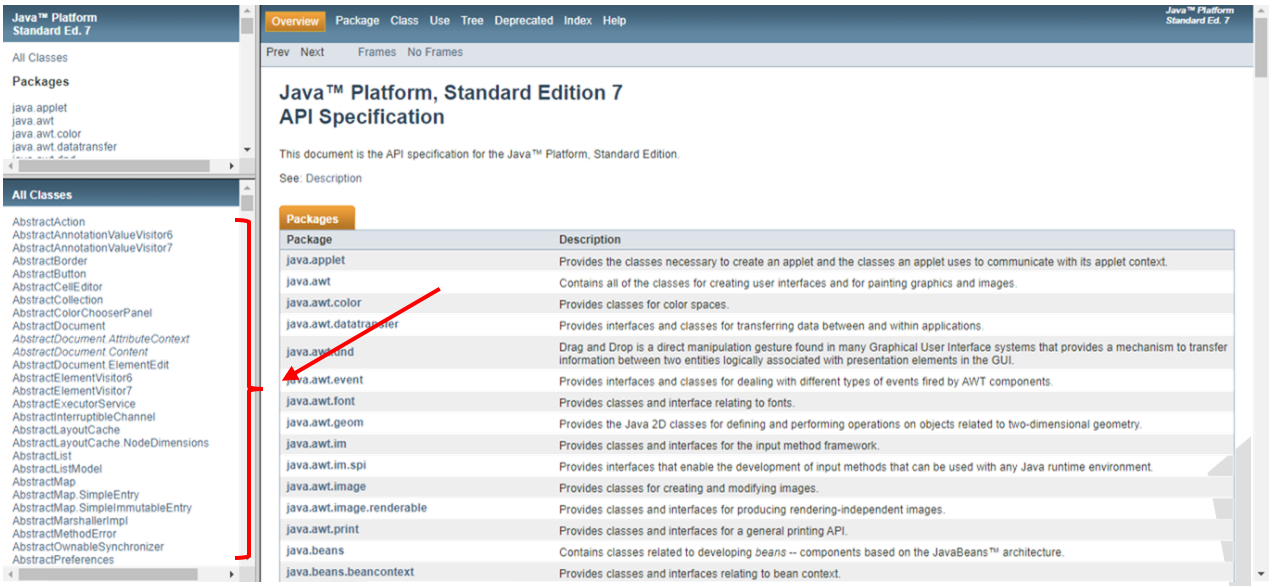
Clases Predefinidas en Java 2
Clase Predefinida Math 2
Refundiciones 5
Clase Predefinida String 6
Clase Predefinida Scanner 8
Clase Predefinida JOptionPane 11
Clase Predefinida Integer 13
Clase Predefinida Double 14
Referencias: 15

Clases Predefinidas en Java

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



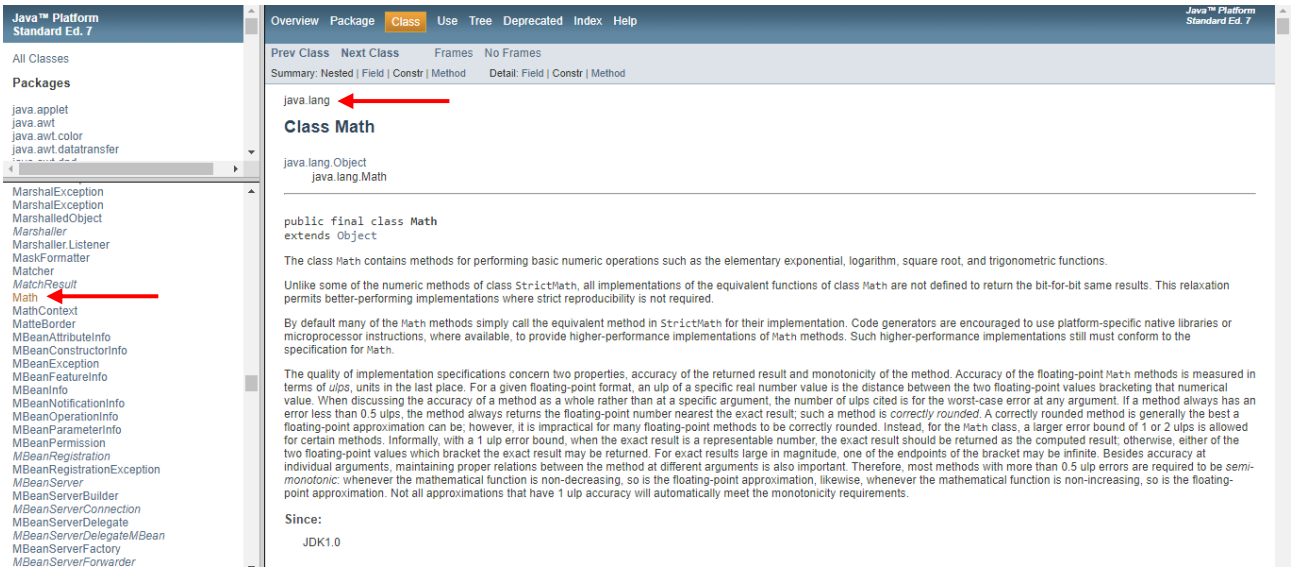
Para poder ver la documentación de las clases predefinidas debo meterme a la API y en el lado izquierdo de la pantalla donde dice All Classes estarán enlistadas todas las clases en orden alfabético.



Clase Predefinida Math

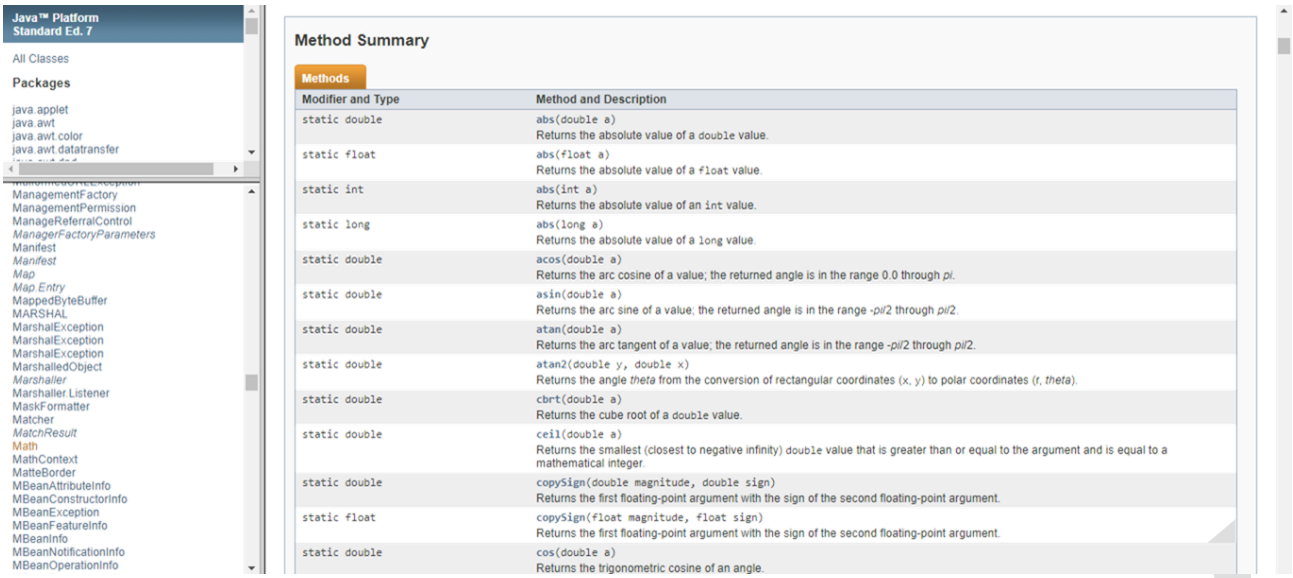
La clase Math tiene muchos métodos para hacer cálculos matemáticos, en este documento veremos algunos, pero si queremos consultarlos a fondo debemos dirigirnos a la API, buscar la clase Math en

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.**

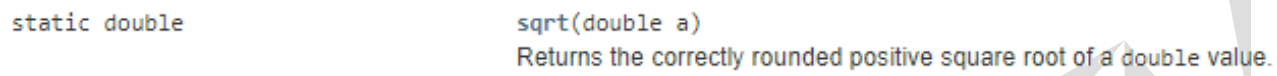


CONSTRUCTOR: La clase Math no tiene ningún método constructor.

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.



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

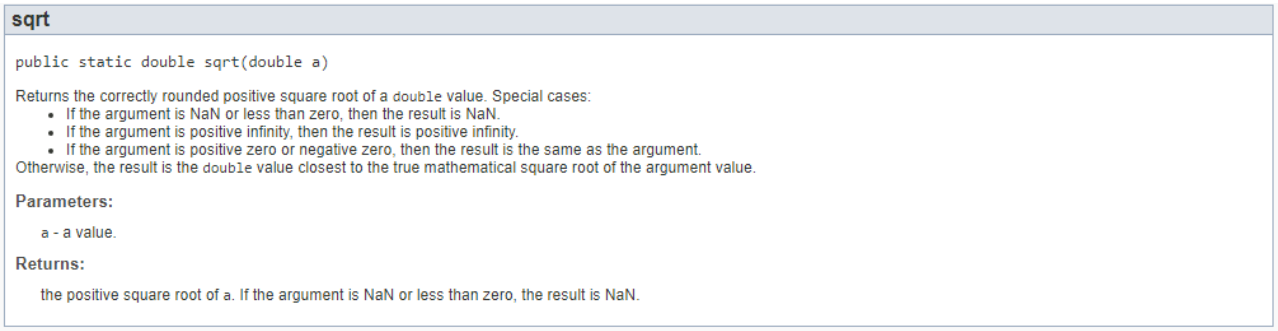


En la documentación de la API podemos ver que el método recibe como parámetro un ángulo de tipo double y regresa otro número de tipo double. Es un método estático.

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.

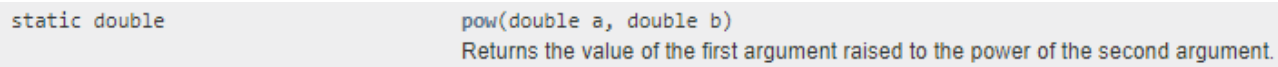
Parameters:

a - a value.

Returns:

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

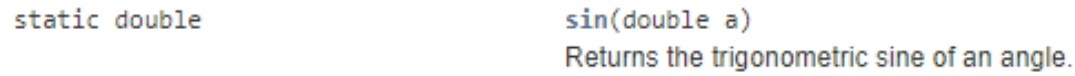
- **Math.pow(base, exponente):** Para obtener un número base elevado a cierta potencia. Tanto la base como el exponente deben ser números decimales tipo double.



`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.

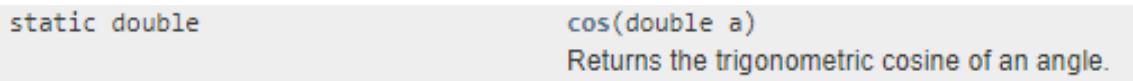
- **Math.sin(angulo):** Se usa para obtener el seno de cierto ángulo dado en radianes.



`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.

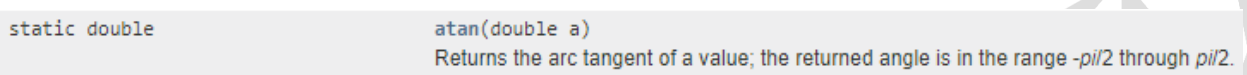
- **Math.cos(angulo):** Se usa para obtener el coseno de cierto ángulo dado en radianes.



`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.

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

`static double`	`random()` Returns a `double` 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
`static double`	`E` The `double` value that is closer than any other to *e*, the base of the natural logarithms.
`static double`	`PI` The `double` value that is closer than any other to *pi*, 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.

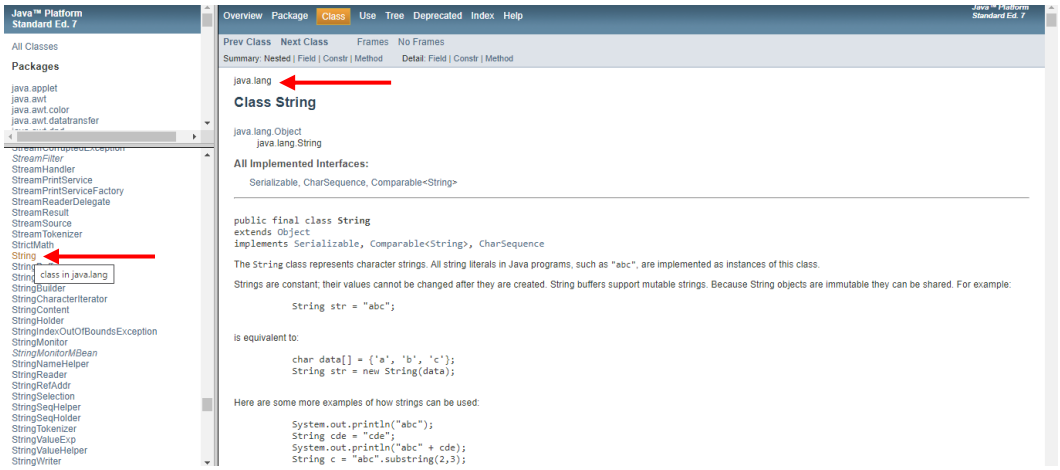
```
tipo_primitivo_1 variable = valor;

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.

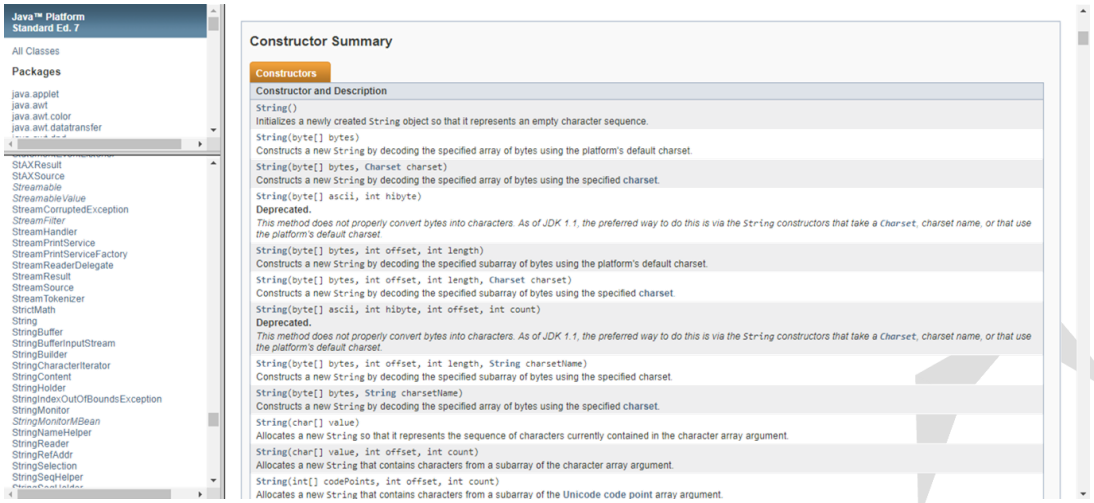
```
int raiz=(int)Math.round(num1);
```

Clase Predefinida String

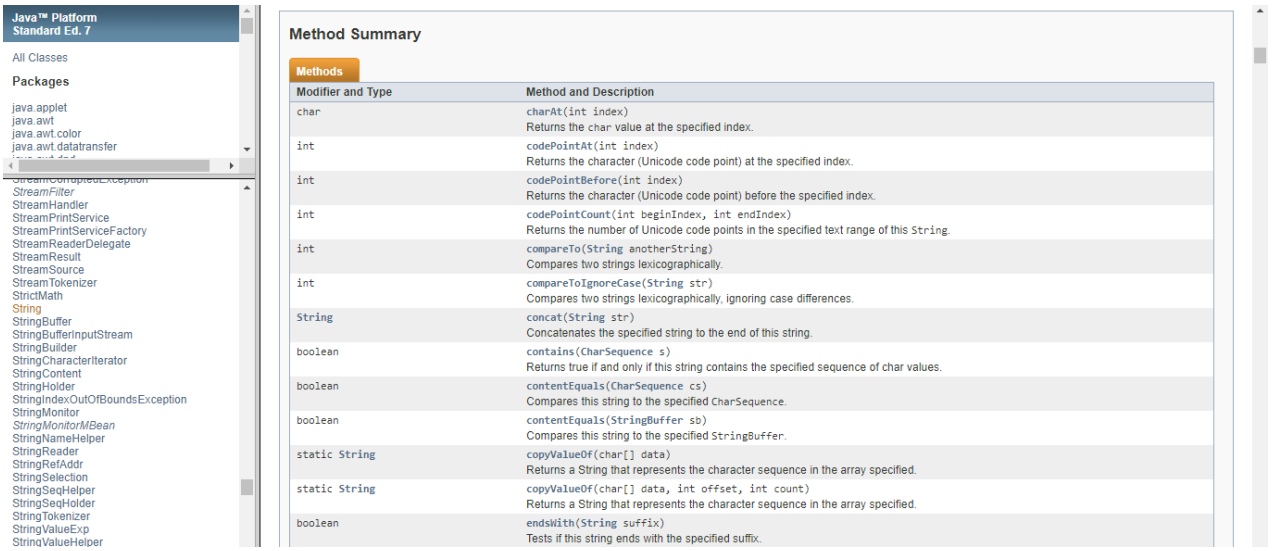
En Java las palabras no son un tipo de dato, son más bien un objeto de la clase predefinida String, esto se maneja así para que con los distintos métodos de esta clase predefinida pueda manipular o analizar cada palabra que cree. **Esta clase pertenece al paquete principal java.lang, por lo que no debe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



MÉTODOS: Los métodos son los que puedo usar para la manipulación de cadenas de texto (palabras) dentro de nuestro código. Los métodos están enlistados en orden alfabético dentro de la API.



- **.length():** Con este método obtenemos el número de letras o caracteres que conforman a un objeto String, osea una palabra.

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 String NO son estáticos, esto implica que para usarlos debo crear un objeto de la clase String y por medio de este aplicar el método.

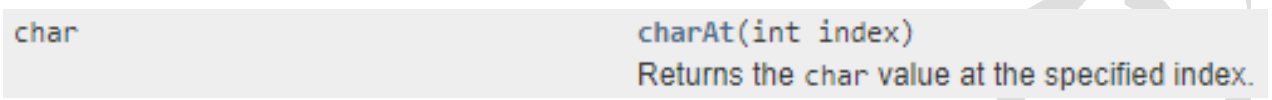
La sintaxis para crear un objeto de la clase String es muy particular ya que no es muy diferente a la forma en la que se declaran variables, se debe poner la palabra reservada String, el nombre de la cadena de caracteres y entre comillas dobles la palabra, se realiza de la siguiente manera:

String nombre_objeto_String = "cadena de caracteres";

Aunque la sintaxis es parecida a la de la declaración e iniciación de variables, a esto se le dice que es un objeto o instancia de la clase String no una variable y se usa para aplicar un método NO estático, esto solo se cumple para la clase String, no para las demás.

nombre_objeto_String.metodo_no_estático;

- **.charAt(posición):** Retorna el carácter que se encuentre en la posición indicada, las posiciones en la palabra se empiezan a contar desde cero.



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.

`String`	`substring(int beginIndex)` Returns a new string that is a substring of this string.
`String`	`substring(int beginIndex, int endIndex)` 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.

`boolean`	`equals(Object anObject)` 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.

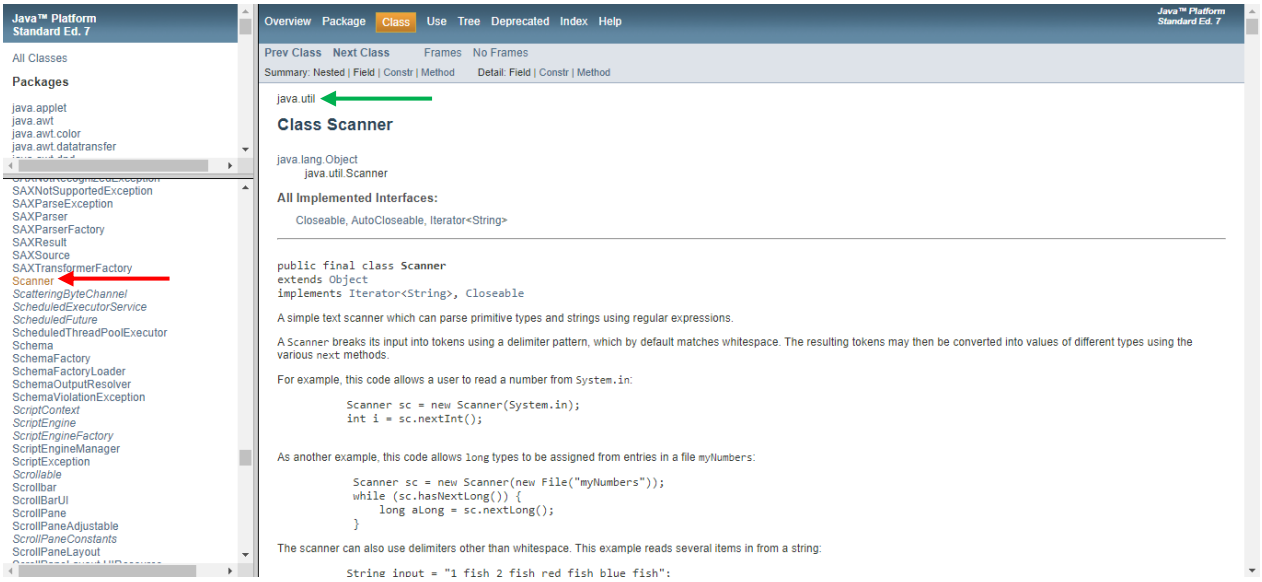
`boolean`	`equalsIgnoreCase(String anotherString)` Compares this `String` to another `String`, ignoring case considerations.

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.

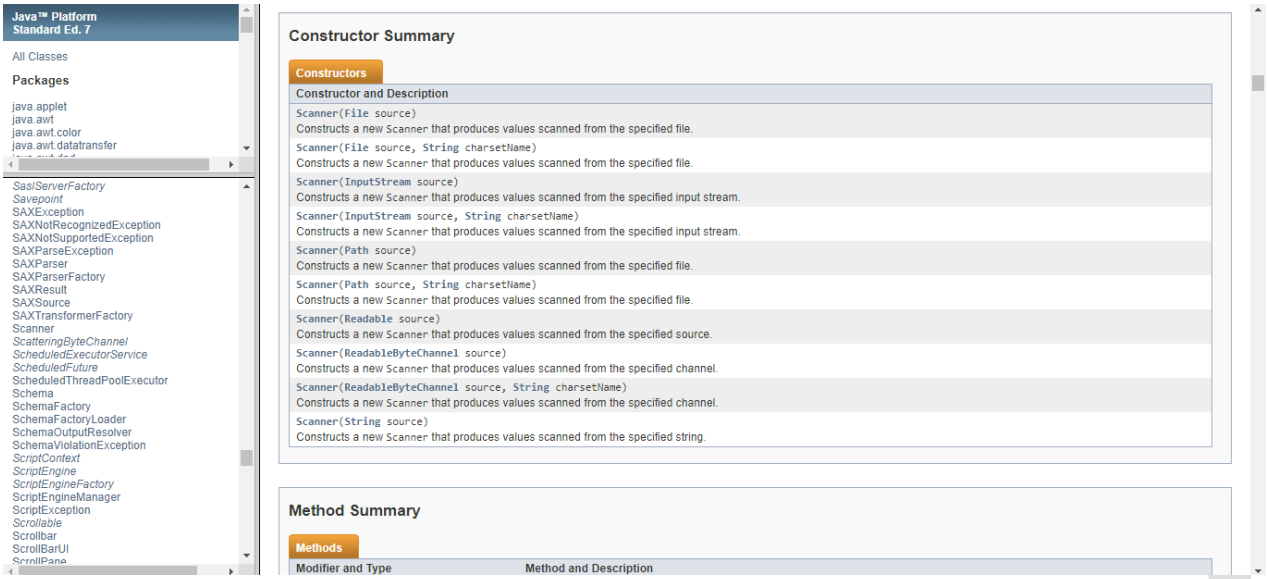
Es un método NO estático.

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

```
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

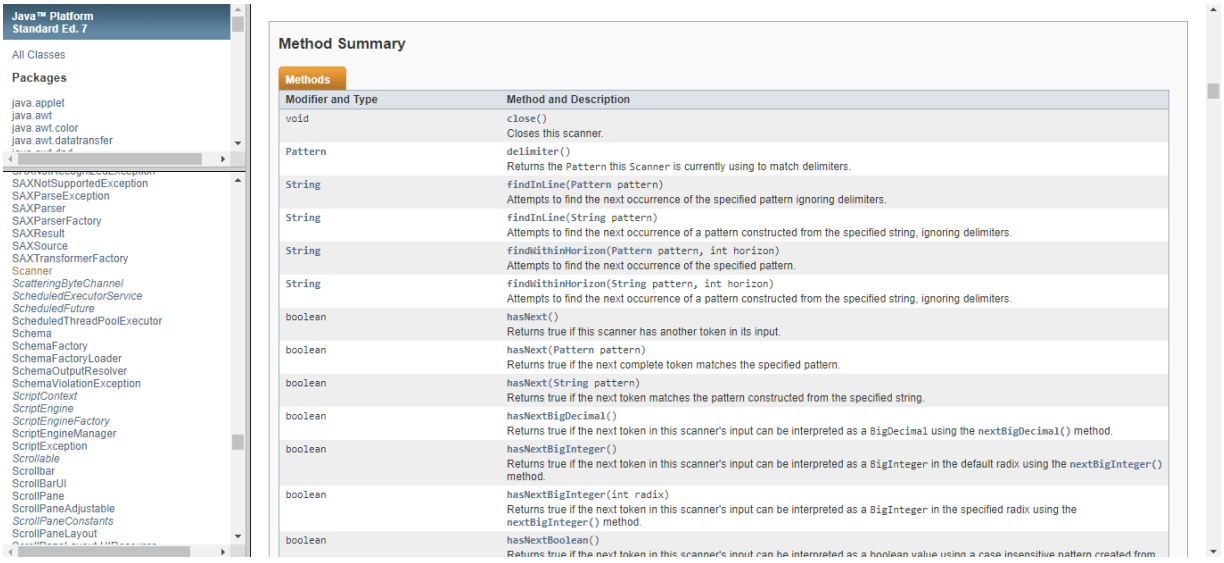
Throws:

`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.



- **.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.

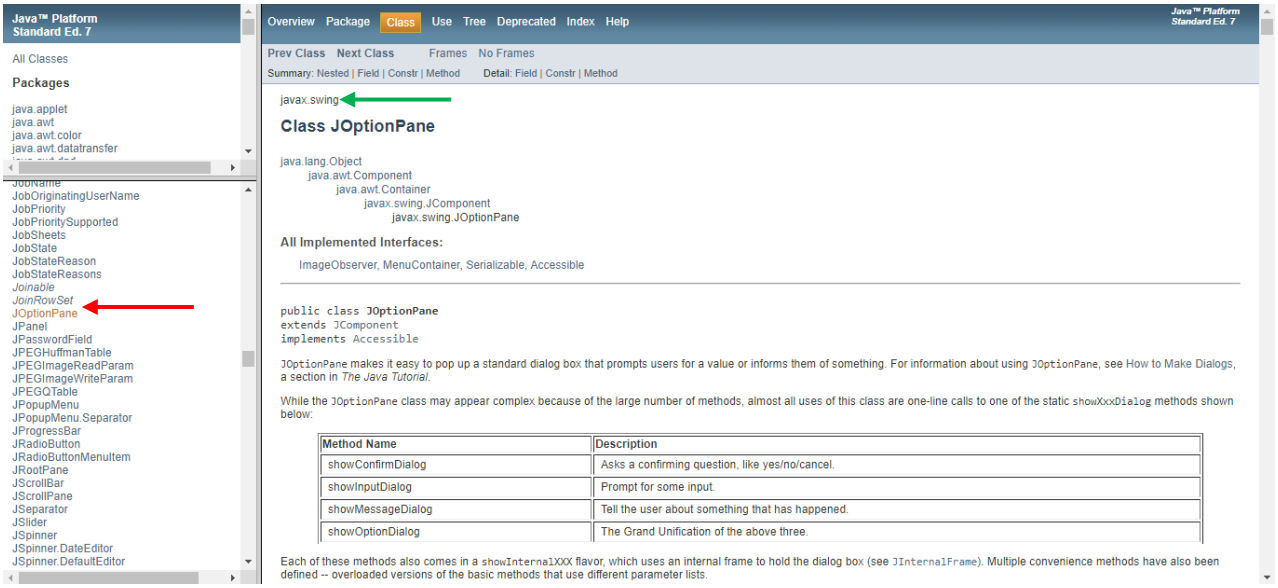
`double`	`nextDouble()` Scans the next token of the input as a `double`.

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.

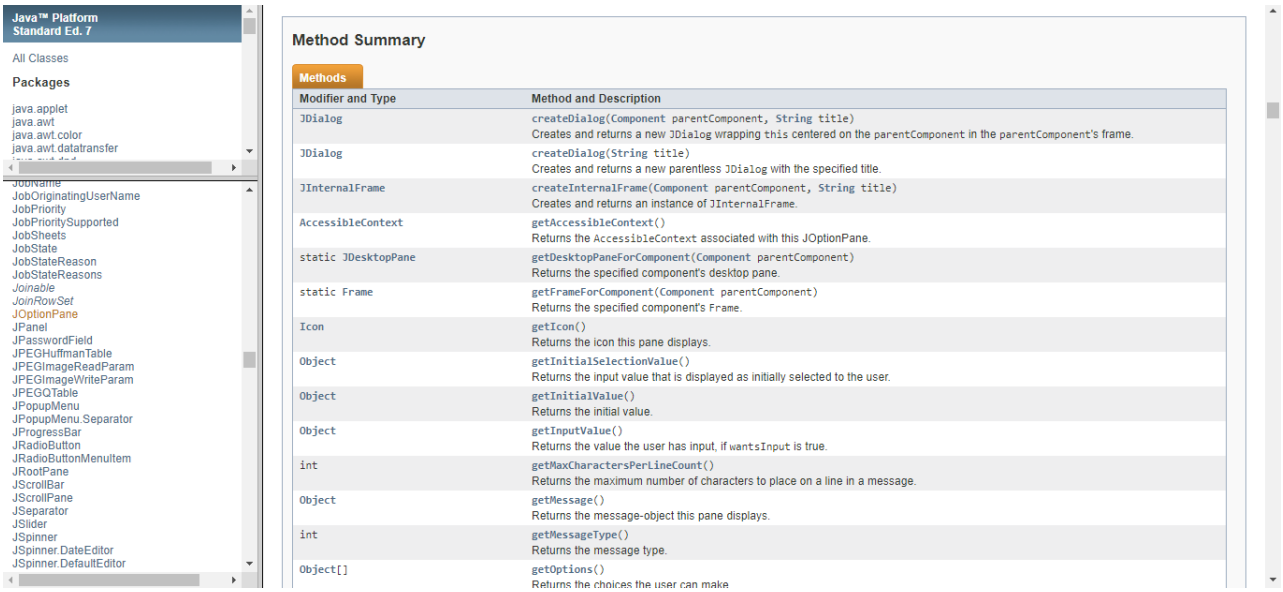
Es un método **NO** estático.

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.



MÉTODOS: Los métodos de la clase JOptionPane sirven para hacer que aparezca la ventana emergente que me permitirá introducir datos a mi programa.



- **JOptionPane.showInputDialog():** Con este método obtenemos el número de letras o caracteres que conforman a un objeto String, osea una palabra.

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

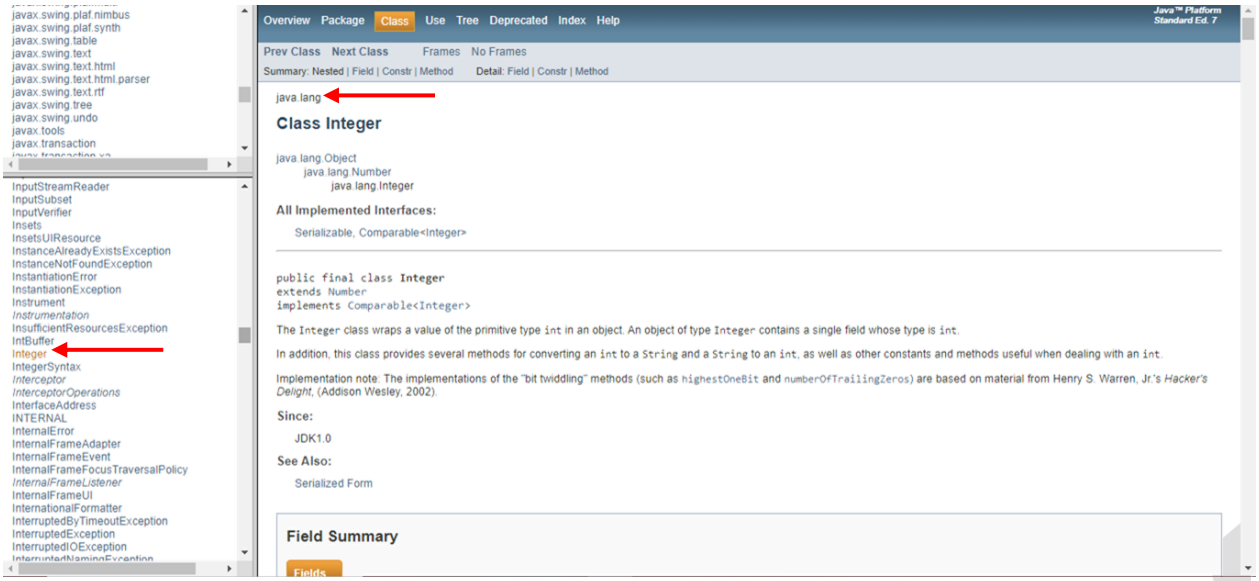
```
static String                    showInputDialog(Object message)
                                 Shows a question-message dialog requesting input from the user.
```

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:

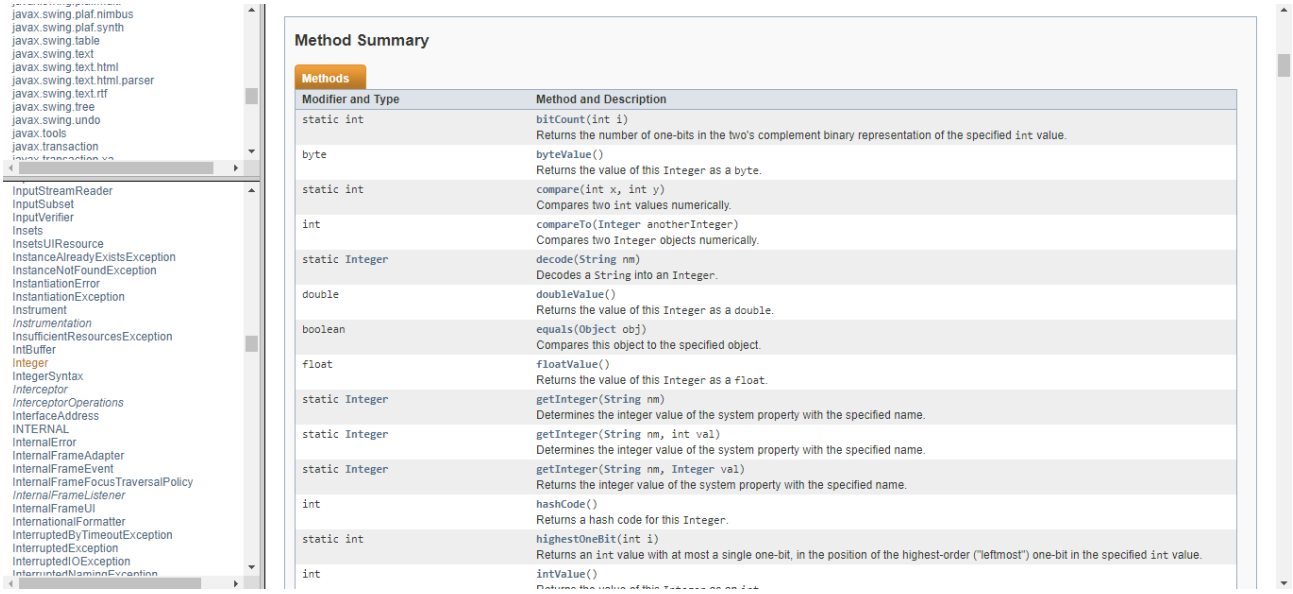
nombreDeLaClase.metodo_estático;

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.



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

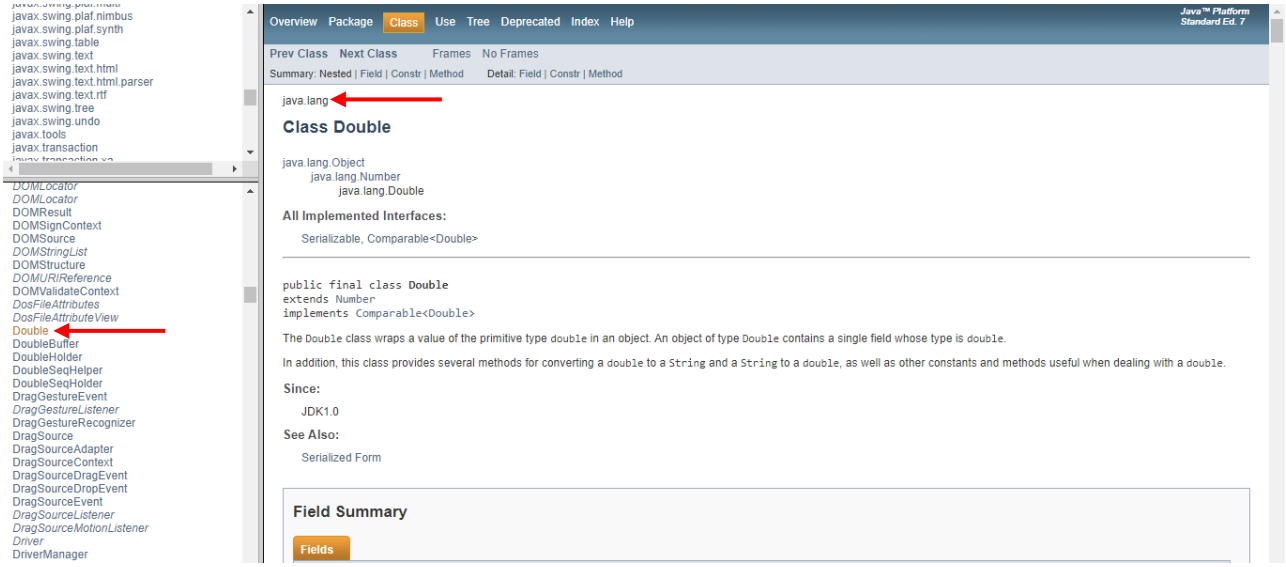
En la documentación de la API podemos ver que el método recibe como parámetro al objeto String y en el segundo argumento se puede indicar el sistema numérico que equivale al número recibido, este puede ser binario, hexadecimal, decimal, octal, etc. y respetará el signo con el que viene para retornar 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:

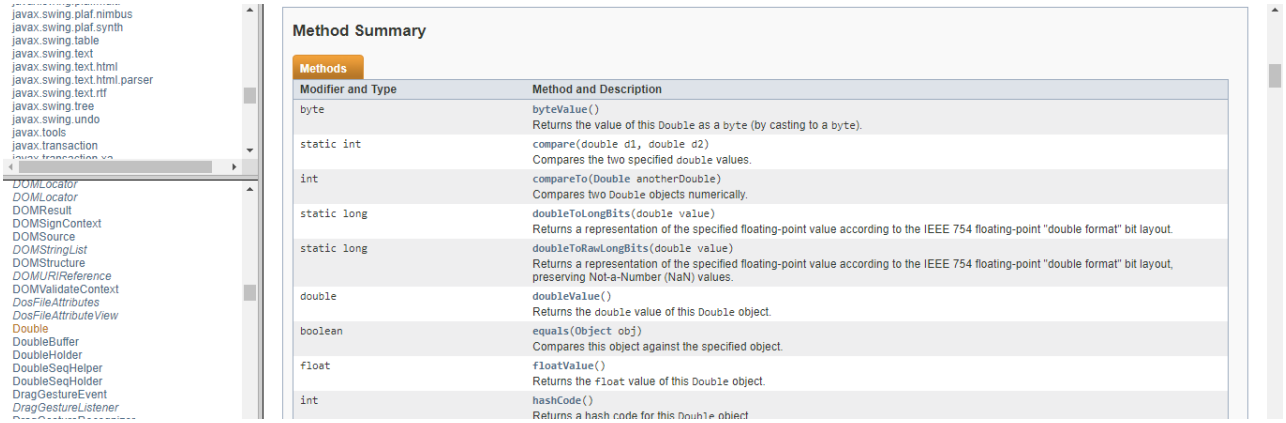
nombreDeLaClase.metodo_estático;

Clase Predefinida Double

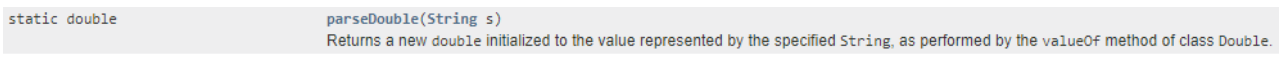
Esta clase también se suele usar como complemento de la clase JOptionPane y sirve principalmente para convertir cualquier tipo de dato en un objeto Double, que es lo mismo a referirse a un tipo de dato double (osea decimal). Pertenece al paquete principal **java.lang** por lo que no debe ser importado.



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



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



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=PLU8oAlHdN5BktAXdEVCLUYzvDyqRQJ2lk