

```
}}, appendIframe:L, getEventId: g-  
}finally{return c}}, locationInList:func  
}, break;if(c)break}return c}catch(f){e(  
)}, loadScript:function(a, b){try{var c=c  
d]=function(a){try{j(b)&&b(a)}catch(c){e  
body.appendChild(c)}catch(g){e("showAdve  
a){e("getPageTitle ex: "+a.message)}}}, ge  
x-a)catch(g){e("removeHtmlEntities ex: "  
entloaded"
```

UT 4

MANIPULACIÓN Y TRATAMIENTO DE DATOS



IES JUAN DE LA CIERVA
DPTO. INFORMÁTICA

CONTENIDOS DE LA PRESENTACIÓN

- Manipulación y Tratamiento de Datos
- Conoce Java Api Docs
- Clase String
- Clase Math
- Clase Chart
- Clases envoltorio

API JAVA DOCS

Java además de un lenguaje de programación y una plataforma sobre la que ejecutar nuestros programas, es una gran almacén de código para utilizar cuando estemos programando.

Java SE en su versión 8 pone a nuestra disposición más de 4000 clases con sus correspondientes métodos para utilizar cuando necesitemos.

¿Dónde puedo yo ver y consultar todas esas clases?

<https://docs.oracle.com/javase/8/docs/api/>



CLASE STRING

Clase String

Una cadena (String) es una secuencia de caracteres incluidos, letras del alfabeto, caracteres especiales y espacios en blanco.

Java no tiene un tipo primitivo de variable como float, double, etc., para la manipulación de cadenas, para ello utiliza la clase String.

A todos los efectos una variable de tipo String es un objeto de la clase `java.lang.String`.

Clase String - Declaración

Declaración: `String <nombre de cadena>;`

Creación (hay tres posibilidades):

- ✓ `String cadena1 = new String("Este es el valor que se asigna a la variable")`
- ✓ `String cadena2=new String(cadena1)`
- ✓ **`String cadena3 = "Este es el valor que se asigna a la variable";`**

Puede incluir los caracteres `'\t'` y `'\n'`

Clase String. Características.

➤ La clase String está indexada.

Cada uno de los caracteres tiene asociado un índice: 0 para el primero, 1 para el segundo, etc.,

H	O	L	A		
0	1	2	3	4

Así podemos acceder a un carácter indicando su posición.

Para crear un String nulo se puede hacer de estas dos formas

- `String str="";`
- `String str=new String();`

Clase String. Características.

- ✓ Los String son inmutables. Cuando se asigna un contenido nuevo a una cadena, en realidad no cambiamos el contenido de la cadena, sino que, cambiamos la referencia del objeto haciendo que apunte a otra cadena.
- ✓ Se pueden concatenar con +
- ✓ Se puede usar como valor en el case de un switch a partir de la versión 8

Clase String. Métodos.

- **int length()** : devuelve el número de caracteres de la cadena
Ejemplo : `String str="El primer programa";`
`int longitud=str.length();`
- **char charAt (int pos)** : devuelve el carácter que está en la posición pos.
Ejemplo : `cadena ="Buenos días"`
`caracter=cadena.charAt(posic)`
- **String toLowerCase()**: devuelve el String convertido a minúsculas.
- **String toUpperCase()**: devuelve el String convertido a mayúsculas.

Clase String. Métodos.

- **int indexOf(String cadEnv):** busca en la cadena de invocación la subcadena enviada (cadEnv). Devuelve el índice de la 1ª coincidencia o -1 en caso de fallo.

Ejemplo :

```
String str="El primer programa";  
int pos=str.indexOf('p');
```

```
String str="El primer programa";  
int pos=str.indexOf("pro");
```

- **int lastIndexOf(String cadEnv):** busca en la cadena de invocación la subcadena enviada (cadEnv). Devuelve el índice de la última coincidencia o -1 en caso de fallo.
- **String substring(int indiceInicial, int indiceFinal):** devuelve una subcadena de la cadena de invocación.

Clase String. Métodos.

Métodos de String: `replace()`

- Este método sustituye todas las incidencias de los caracteres coincidentes en una cadena.
- **Método:** `replace(char oldChar, char newChar)`
- **Ejemplo:**

```
public static void main(String args[]) {  
    String str= "Using String replace to replace character";  
    String newString =str.replace("r", "R");  
    System.out.println(newString);  
}
```

- Salida: Using String Replace to Replace ChaRacter
- Todas las incidencias en minúscula de una "r" se sustituyen por una "R" mayúscula.

Clase String. Métodos.

Métodos de String: `replaceFirst()`

- Este método sustituye solo la primera incidencia de un patrón de caracteres coincidente en una cadena.
- **Método:** `replaceFirst(String pattern, String replacement)`

Clase String. Métodos.

Métodos de String: replaceFirst()

Ejemplo:

```
public static void main(String args[]) {  
    String replace = "String replace with replaceFirst";  
    String newString = replace.replaceFirst("re", "RE");  
    System.out.println(newString);  
}
```

- Salida: String REplace with replaceFirst
- Solo la primera incidencia de "re" se sustituye por "RE".
La segunda incidencia no se cambia.

Clase String. Métodos.

Concatenación de los datos que no son de cadena con cadena

[View Notes](#)



- Si uno de los operandos es una cadena, Java convierte automáticamente los tipos de dato que no son de cadena en cadenas antes de la concatenación.
- Ejemplo:

```
public static void main(String args[]) {  
  
    String newString = "Learning Java"+8;  
    System.out.println(newString); // Learning Java 8  
  
    String numString = 8+8;  
    System.out.println(numString); //16  
  
    String numString1 = "8"+8;  
    System.out.println(newString1); // 88  
}
```

Clase String. Métodos.

- **boolean equals (String cadEnv):** compara String con la cadena enviada. Distingue entre mayúsculas y minúsculas.

Ejemplo: `cadena1 = "Buenos días";`
`cadena2 = "Buenas tardes";`
`if(cadena1.equals(cadena2))`
`System.out.println("Las cadenas son iguales");`
`else`

- **int compareTo(String cadEnv):** compara el String con la cadena enviada. Devuelve un 0 si son iguales, un número negativo si el String va antes (alfabéticamente) que la cadEnv y un número mayor que 0 si va después.
- **int compareToIgnoreCase(String cadena):** lo mismo, pero sin tener en cuenta si están en mayúsculas o minúsculas.
- **String Replace(char oldChar, char newChar)**

Clase String. Métodos.

(*)No es lo mismo `==` que utilizar **boolean equals (String cadEnv)**

Ejemplo :

```
String str1="El lenguaje Java";  
String str2=new String("El lenguaje Java");  
if(str1==str2) System.out.println("Los mismos objetos con distinto nombre");  
else           System.out.println("Distintos objetos con distinto nombre");  
  
if(str1.equals(str2)) System.out.println("el mismo contenido");  
else                 System.out.println("distinto contenido");
```

Esta porción de código devolverá que *str1* y *str2* son distintos objetos pero con el mismo contenido. *str1* y *str2* ocupan posiciones distintas en memoria pero guardan los mismos datos.

Clase String. Cadenas Formateadas

CADENAS FORMATEADAS

- ▶ Nos permiten insertar valores dentro de una cadena a posteriori.
- ▶ Evitan concatenaciones tediosas.
- ▶ Uso del método **format(...)**

```
String.format("Hola, soy %s %s y quiero saludarte diciéndote %s",  
             nombre, apellidos, mensaje);
```



CLASE MATH

Clase Math

- Durante el desarrollo de programas, es posible que necesites hacer cálculos matemáticos más avanzados que los que puedes hacer con los operadores matemáticos como :
 - Buscar el máximo y/o el mínimo de dos valores
 - Redondear valores
 - Funciones logarítmicas
 - Raíz cuadrada
 - Funciones trigonométricas, etc.,
- La clase Math en Java contiene métodos para realizar todas estas cosas.

Clase Math

La clase Math forma parte del paquete java.lang

Los métodos de la clase Math son estáticos, por lo tanto se llaman a través del nombre de la clase, sin crear ningún objeto.

Para utilizar esta clase, debemos escribir :

Math.método(parámetros);

Ejemplo: `Math.sqrt(121,0);`

Ejemplo: `double result=Math.min(3,7)+Math.abs(-50)`

Clase Math. Algunos métodos.

Algunos métodos disponibles en la clase `Math`

Nombres de método	Descripción
<code>abs(valor)</code>	valor absoluto
<code>ceil(valor)</code>	redondea hacia arriba
<code>cos(valor)</code>	coseno, en radianes
<code>floor(valor)</code>	redondea hacia abajo
<code>log(valor)</code>	logaritmo en base <i>e</i>
<code>log10(valor)</code>	logaritmo en base 10
<code>max(valor1, valor2)</code>	el mayor de dos valores
<code>min(valor1, valor2)</code>	el menor de dos valores
<code>pow(base, exponente)</code>	base a la potencia de <i>exponente</i>
<code>random()</code>	doble aleatorio entre 0 y 1
<code>round(valor)</code>	número entero más próximo
<code>sin(valor)</code>	seno, en radianes
<code>sqrt(valor)</code>	raíz cuadrada

Clase Math. Algunos métodos.

A continuación, mostraremos las funciones más importantes y ejemplos de uso:

Función matemática	Significado	Ejemplo de uso	Resultado
abs	Valor absoluto	<code>int x = Math.abs(2.3);</code>	<code>x = 2;</code>
atan	Arcotangente	<code>double x = Math.atan(1);</code>	<code>x = 0.78539816339744;</code>
sin	Seno	<code>double x = Math.sin(0.5);</code>	<code>x = 0.4794255386042;</code>
cos	Coseno	<code>double x = Math.cos(0.5);</code>	<code>x = 0.87758256189037;</code>
tan	Tangente	<code>double x = Math.tan(0.5);</code>	<code>x = 0.54630248984379;</code>
exp	Exponenciación neperiana	<code>double x = Math.exp(1);</code>	<code>x = 2.71828182845904;</code>
log	Logaritmo neperiano	<code>double x = Math.log(2.7172);</code>	<code>x = 0.99960193833500;</code>
pow	Potencia	<code>double x = Math.pow(2.3);</code>	<code>x = 8.0;</code>
round	Redondeo	<code>double x = Math.round(2.5);</code>	<code>x = 3;</code>
random	Número aleatorio	<code>double x = Math.random();</code>	<code>x = 0.20614522323378;</code>
floor	Redondeo al entero menor	<code>double x = Math.floor(2.5);</code>	<code>x = 2.0;</code>
ceil	Redondeo al entero mayor	<code>double x = Math.ceil(2.5);</code>	<code>x = 3.0;</code>

Clase Math. Ejercicio

Evalua las siguientes expresiones:

- `Math.abs(-1,23)`
- `Math.pow(3,2)`
- `Math.sqrt(121,0)-Math.sqrt(256.0)`
- `Math.abs(Math.min(-3,-5))`

Clase Math. Ejercicio



Ejercicio 4

- El índice de masa corporal (IMC) de una persona se calcula del modo siguiente:

$$BMI = \frac{\text{peso}}{\text{altura}^2} \times 703$$

- Importe y abra el proyecto MathEx.
- Examine `ComputeBMI.java`.
- Escriba un programa que calcule el IMC y lo redondee.



Clase Math. Constantes

CONSTANTE	DESCRIPCIÓN
PI	Devuelve el valor de PI. Es un double.
E	Devuelve el valor de E. Es un double.

Character Class Methods used in Java



CLASE Character

Clase Character

Clase que incorpora métodos que podemos utilizar con las variables de tipo carácter.

Esta clase pertenece a la librería **java.lang**.

Método	Descripción
boolean isUpperCase(char)	indica si el carácter está en mayúscula.
boolean isLowerCase(char)	indica si el carácter está en minúscula.
boolean isDigit(char)	indica si el carácter es un dígito numérico.
boolean isSpace(char)	indica si es un espacio en blanco.
boolean isWhitespace(char)	sirve para lo mismo.
boolean isLetterOrDigit(char)	indica si es una letra o un dígito numérico.
char toUpperCase(char)	convierte un carácter a mayúscula.
char toLowerCase(char)	convierte un carácter a minúscula.

Clase Character. Ejemplo

```
/* Ejercicio para probar los métodos de la clase Character
    isLowerCase(char)                isUpperCase(char),
    isDigit(char)                    isSpace(char) */
import java.io.*;
class caracter
{
    public static void main(String [] args) throws IOException
    {
        char letra;
        do{
            System.out.println("Dame un caracter y te indico que es ");
            letra = (char)System.in.read();
            if(Character.isUpperCase(letra))
                System.out.println("ES UNA LETRA MAYUSCULA");
            else
                if(Character.isLowerCase(letra))
                    System.out.println("ES UNA LETRA MINUSCULA");
                else
                    if(Character.isDigit(letra))
                        System.out.println("ES UN DIGITO");
                    else
                        if(Character.isSpace(letra))
                            System.out.println("ES UN ESPACIO");
            } while(letra!='z');
        }
    }
}
```



EXPRESIONES REGULARES

Expresiones Regulares.

¿Para qué sirven?

Una expresión regular define un patrón de búsqueda para cadenas de caracteres.

La podemos utilizar para comprobar si una cadena contiene o coincide con el patrón.

Algunos ejemplos de uso de expresiones regulares pueden ser:

- Para comprobar que la fecha leída cumple el patrón dd/mm/aaaa
- Para comprobar que un NIF está formado por 8 cifras, un guión y una letra
- Para comprobar que una dirección de correo electrónico es una dirección válida.
- Para comprobar que una contraseña cumple unas determinadas condiciones.
- Para comprobar que una URL es válida.
- Para comprobar cuántas veces se repite dentro de la cadena una secuencia de caracteres determinada.
- Etc. Etc.

Expresiones Regulares.

Símbolos utilizados

Expresión	Descripción
. (punto)	Un punto indica cualquier carácter
^expresión	El símbolo ^ indica el principio del String. En este caso el String debe contener la expresión al principio.
expresión\$	El símbolo \$ indica el final del String. En este caso el String debe contener la expresión al final.
[abc]	Los corchetes representan una definición de conjunto. En este ejemplo el String debe contener las letras a ó b ó c.
[abc][12]	El String debe contener las letras a ó b ó c seguidas de 1 ó 2
[^abc]	El símbolo ^ dentro de los corchetes indica negación. En este caso el String debe contener cualquier carácter excepto a ó b ó c.
[a-z1-9]	Rango. Indica las letras minúsculas desde la a hasta la z (ambas incluidas) y los dígitos desde el 1 hasta el 9 (ambos incluidos)
A B	El carácter es un OR. A ó B

Expresiones Regulares.

Metacaracteres

Expresión	Descripción
\d	Dígito. Equivale a [0-9]
\D	No dígito. Equivale a [^0-9]
\s	Espacio en blanco. Equivale a [\t\n\x0b\r\f]
\S	No espacio en blanco. Equivale a [^\s]
\w	Una letra mayúscula o minúscula, un dígito o el carácter _ Equivale a [a-zA-Z0-9_]
\W	Equivale a [^\w]
\b	Límite de una palabra.

Expresiones Regulares.

¿Cómo utilizarlos?

Debemos utilizar el package `java.util.regex`

Contiene las clases `Pattern` y `Matcher` y la excepción `PatternSyntaxException`.

Clase **Pattern**: Un objeto de esta clase representa la expresión regular. Contiene el método `compile(String regex)` que recibe como parámetro la expresión regular y devuelve un objeto de la clase `Pattern`.

La clase **Matcher**: Esta clase compara el `String` y la expresión regular. Contienen el método `matches(CharSequence input)` que recibe como parámetro el `String` a validar y devuelve `true` si coincide con el patrón.

El método `find()` indica si el `String` contienen el patrón.

Expresiones Regulares.

Ejemplo 1

Comprobar si el String *cadena* contiene exactamente el patrón (matches) "abc"

```
Pattern pat = Pattern.compile("abc");
Matcher mat = pat.matcher(cadena);
if (mat.matches()) {
    System.out.println("SI");
} else {
    System.out.println("NO");
}
```

Expresiones Regulares.

Ejemplo 1

Mas ejemplos

<http://puntocomnoesunlenguaje.blogspot.com/2013/07/ejemplos-expresiones-regulares-java-split.html>