educación IT
CODE YOUR FUTURE

Introducción a Java

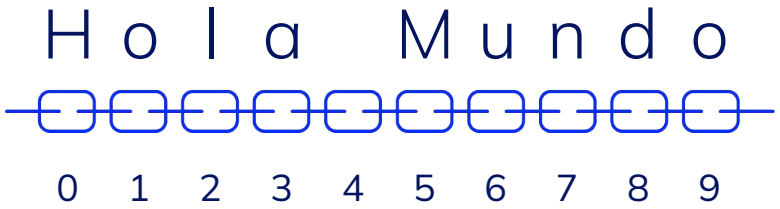
Módulo 2

Clase *String*

String y qué podemos hacer

Anteriormente habíamos conocidos los *datos primitivos* en Java y el tipo de dato `string` que nos ayuda representar una cadena de caracteres. Ahora bien, los `string` no son un tipo de dato primitivo sino un ***objeto***.

Por ahora, solo diremos que los objetos son **un tipo de dato más complejo, que posee atributos (variables y/o constantes) y métodos que nos serán de gran utilidad a lo largo del desarrollo del software**.



Métodos

Tipo	Método	Descripción
Char	**charAt(int index)**	Devuelve el valor de carácter en el índice especificado.
String	**concat(String str)**	Concatena la cadena especificada al final de esta cadena.
Boolean	**contains(CharSequence s)**	Devuelve verdadero si y solo si esta cadena contiene la secuencia especificada de valores de caracteres.
Boolean	**endsWith(String suffix)**	Prueba si esta cadena termina con el sufijo especificado.
Boolean	**equals(Object anObject)**	Compara esta cadena con el objeto especificado.
Boolean	**equalsIgnoreCase(String anotherString)**	Compara esta cadena con otra cadena, ignorando las consideraciones de caso.
Static String	**format(String format, Object... args)**	Devuelve una cadena formateada utilizando la cadena de formato y los argumentos especificados.
Int	**indexOf(int ch)**	Devuelve el índice dentro de esta cadena de la primera aparición del carácter especificado.

Tipo	Método	Descripción
Boolean	**isEmpty()**	Devuelve verdadero si, y solo si, *length()* es 0
Static String	**join(CharSequence delimiter, CharSequence... elements)**	Devuelve una nueva cadena compuesta por copias de los elementos *CharSequence* unidas con una copia del delimitador especificado.
Int	**lastIndexOf(int ch)**	Devuelve el índice dentro de esta cadena de la última aparición del carácter especificado.
Int	**length()**	Devuelve la longitud de esta cadena.
String	**replace(char oldChar, char newChar)**	Devuelve una cadena resultante de reemplazar todas las apariciones de *oldChar* en esta cadena por *newChar.*
String	**toLowerCase()**	Convierte todos los caracteres de esta cadena a minúsculas utilizando las reglas de la configuración regional predeterminada.
String	**toUpperCase()**	Convierte todos los caracteres de esta cadena a mayúsculas utilizando las reglas de la configuración regional predeterminada.

Tipo	Método	Descripción
Static String	**valueOf(objeto o dato primitivo b)**	Devuelve la representación de cadena del argumento enviado.
Static String	**trim()**	Devuelve una cadena cuyo valor es esta cadena, con cualquier espacio en blanco inicial y final eliminado.
String	**substring(int beginIndex)**	Devuelve una cadena que es una subcadena de esta cadena.
Char[]	**substring(int beginIndex, int endIndex)**	Devuelve una cadena que es una subcadena de esta cadena.
String	**toLowerCase()**	Convierte todos los caracteres de esta cadena a minúsculas utilizando las reglas de la configuración regional predeterminada.
String	**toUpperCase()**	Convierte todos los caracteres de esta cadena a mayúsculas utilizando las reglas de la configuración regional predeterminada.
Static String	**valueOf(objeto o dato primitivo b)**	Devuelve la representación de cadena del argumento enviado.

¿Por qué usar el método *equals*?

En Java, los **operadores relacionales** comparan *bit* a *bit* y en los objetos (que profundizaremos más adelante) se compara no el valor, sino la posición de memoria, por lo que si comparamos dos cadenas de texto introducidas por el teclado con el operador relación `==`, nos devolverá *`false`.*

Java nos entrega un método **`equals`** que solventa este problema.

educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!