TÉCNICAS DE PROGRAMACIÓN (JAVA)

Presentación 04 – Clase String y Clase Scanner

CFP Nº8 FORMACION PROFESIONAL SMATA

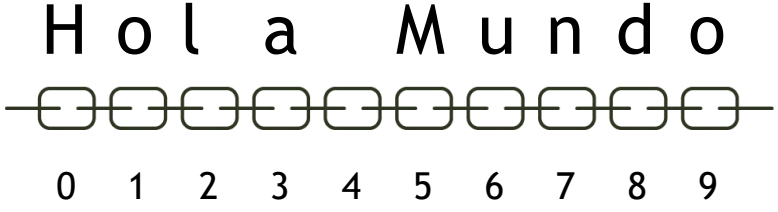
Contenido:

- String
- Métodos de la clase String
- Comparación de String
- Librerías
- Clase Scanner

String

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 de la Clase String

Tipo de retorno	Método	Descripción
char	charAt(int index)	Devuelve el valor del 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 del caso.
String	format(String format, Object... args)	Devuelve una cadena formateada utilizando la cadena de formato y argumentos especificados.
Int	indexOf(int ch)	Devuelve el índice dentro de esta cadena de la primera aparición del carácter especificado.

Métodos de la Clase String

Tipo de retorno	Método	Descripción
boolean	isEmpty()	Devuelve verdadero si, y solo si, length() es 0
String	join(CharSequence delimiter, CharSequence... elements)	Devuelve una nueva cadena compuesta por copias de los elemento 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.

Métodos de la Clase String

Tipo de retorno	Método	Descripción
String	valueOf(objeto o dato primitivo b)	Devuelve la representación de cadena del argumento enviado.
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.
String	substring(int beginIndex, int endIndex)	Devuelve una cadena que es una subcadena de esta cadena.
String	toLowerCase(Locale locale)	Convierte todos los caracteres de esta cadena a minúsculas utilizando las reglas de la configuración regional predeterminada.
String	toUpperCase(Locale to Lowercase)	Convierte todos los caracteres de esta cadena a mayúsculas utilizando las reglas de la configuración regional predeterminada.
String	valueOf(objeto o dato primitivo b)	Devuelve la representación de cadena del argumento enviado.

Comparación de String

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.

Librerías

En Java, como en muchos otros lenguajes de programación, existe el concepto de librería: una clase o conjunto de ellas que posee métodos y atributos (variables y/o constantes) que nos permiten reutilizar código para un propósito particular.

Para recurrir a una librería interna o externa en Java debemos utilizar la palabra reservada **import** seguida del nombre de la clase y el paquete donde se encuentra.

```
//bibliotecas importadas
import java.util.Scanner;
```

Librerías

En la imagen de la siguiente pantalla podemos apreciar que nuestros proyectos tienen un apartado **JRE System Library** que contiene las librerías que ofrece Java.

Algo particular que pasa en Java es que hay clases que no necesitan importarse, como cuando utilizamos **System.out.print**. Esto se debe a que esta clase se encuentra dentro del paquete **java.lang** y no necesitamos importarla.

En cambio, con las otras utilidades o librerías debemos hacerlo siempre.

Presentación 04 – Clase String y Clase Scanner

Scanner

Hay una utilidad que nos ayuda a capturar los datos de la consola, para ello debemos importar la biblioteca java.util.Scanner.

```
// paquete al que pertenece la clase
package identificadorPaquete;

// bibliotecas importadas
import java.util.Scanner;

// declaracion de la clase
public class IdentificadorClase {
```

Para poder obtener los datos ingresados por el usuario debemos crear este elemento con la siguiente estructura:

```
// la clase scanner para capturar valores del teclado
Scanner identificador = new Scanner(System.in);
```

Scanner

Tipo de Dato	Método
Boolean	`nextBoolean()`
Byte	`nextByte()`
Double	`nextDouble()`
Float	`nextFloat()`
Int	`nextInt()`
Long	`nextLong()`
Short	`nextShort()`
String	`next()`
String	`nextLine()`
N/A	`close()`

Cada vez que usemos uno de los métodos de la clase *Scanner*, la consola quedará esperando que el usuario ingrese una información. Es recomendable mostrar por pantalla qué dato debe ingresar:

```
// ejemplo de como capturar un numero entero
System.out.println("Ingrese el numero entero");

int identificadorEntero = identificador.nextInt();
```

Scanner

Tipo de Dato	Método
Boolean	`nextBoolean()`
Byte	`nextByte()`
Double	`nextDouble()`
Float	`nextFloat()`
Int	`nextInt()`
Long	`nextLong()`
Short	`nextShort()`
String	`next()`
String	`nextLine()`
N/A	`close()`

Cada vez que usemos uno de los métodos de la clase *Scanner*, la consola quedará esperando que el usuario ingrese una información. Es recomendable mostrar por pantalla qué dato debe ingresar:

```
// ejemplo de como capturar un numero entero
System.out.println("Ingrese el numero entero");

int identificadorEntero = identificador.nextInt();
```