Programación

Clase String

Las cadenas de texto en Java son **Objetos**. Parece que tienen un funcionamiento similar a los tipos primitivos, pero a diferencia de estos tienen una serie de métodos. Además es un tipo de objeto con algunos privilegios.

Un objeto String representa una cadena de caracteres, y puede tener 0 ó más caracteres. Su representación literal se rodea con comillas dobles: `"asd 123"`.

Paquete java.lang

Este paquete es el encargado de contener algunas de las clases más usadas. La característica de las clases contenidas en este paquete es que no necesitan ser importadas `import` para ser usadas. Todo proyecto java importa este paquete completo por defecto.

Crear una instancia de String

Crear un objeto se una clase, se conoce como *instanciar*:

```
//Instanciación rápida
String cadena1 = "primer programa";

//Instanciación lenta
String cadena2 = new String("otro programa");
```

Métodos de la clase String

Los Strings son objetos, por lo que además de un estado (valor) tienen unas operaciones (métodos).

Definición de método

Los métodos (o funciones) son un tipo de instrucción que puede devolver (dar como resultado) un valor de un tipo de datos, y además pueden recibir varios valores para funcionar. Los valores que reciben los métodos se conocen como ***parámetros***, y se escriben entre los paréntesis finales de la llamada al método método.

A continuación, vemos algunos de los métodos de la clase String más usados:

Nombre	Función	Parámetros que recibe	Valor que devuelve
length()	Indica la longitud en caracteres del String	No recibe nada	Devuelve un *int* con la cantidad de caracteres
charAt()	Obtiene el caracter (*char*) en la posición indicada	Un *int* indicando la posición (Empieza en 0)	El *char* de esa posición
equals()	Indica si dos Strings son iguales	Recibe otro *String*	true si son iguales

Nombre	**Función**	**Parámetros que recibe**	**Valor que devuelve**
equalsIgnoreCase()	Indica si dos Strings son iguales sin atender a maýusculas-minúsculas	otro *String*	true si son iguales
contains()	Indica si una cadena contiene a otra pasada por parámetro	otro String	true si la contiene, false si no
compareTo()	Comprueba si una cadena es mayor, igual o menor a otra, alfabéticamente (tabla ascii)	otro *String*	Un *int* <0, 0, ó >0 si el parámetro es menor, igual o mayor a la cadena.
substring()	Obtiene una subcadena con el rango de caracteres desde inicio a fin	Dos *ints*, inicio y fin, indicando el rango de la subcadena	una subcadena *String* con el rango de caracteres desde inicio hasta fin
substring()	Obtiene una subcadena *String* desde el valor inicio, hasta el final de la cadena	un solo int de inicio	subcadena *String* desde el caracter indicando en inicio, hasta el final de la cadena
valueOf()	Convierte el valor recibido a un *String*	Un elemento de cualquier tipo primitivo	*String* con la representación
replaceAll()	Reemplaza la aparición de una serie de caracteres por otros	Dos Strings con los caracteres a reemplazar, y el valor del reemplazo	*String* con el resultado del reemplazo
indexOf()	Busca el valor recibido e indica la posición donde lo encuentra	*String* ó *char* con el valor	*int* con la posición en la que lo encuentra ó -1 si no lo encuentra
indexOf()	Busca el valor recibido a partir de una posicion dada e indica la posición donde lo encuentra	(2 parámetros) *String* ó un *char* con el valor y un entero con la posición de inicio de la búsqueda	*int* con la posición en la que lo encuentra ó -1 si no lo encuentra
startsWith()	Indica si una cadena empieza por otra cadena recibida por parámetro	un String	true si la cadena comienza por la cadena recibida por parámetro
lastIndexOf()	Devuelve la posición de la última aparición de un valor	*String* ó *char* con el valor a buscar	Un int con la posición, ó -1 si no se encuentra
endsWith()	Indica si una cadena termina por otra cadena recibida por parámetro	un String	true si la cadena termina por la cadena recibida por parámetro
toLowerCase()	Convierte el String a mínúsculas	Nada	El *String* en minúsculas
toUpperCase()	Convierte el String a mayúsculas	Nada	El *String* en mayúsculas
split()	Crea un *array* resultado de dividir el String por el valor separador	*String* con el valor separador	Un array con tantas celdas, como divisiones tenga el *String* por el valor separador indicado
toCharArray()	Obtiene un array de caracteres	Nada	Un array con los caracteres del String

Para acceder a los métodos de un String, ponemos el caracter . detrás de la variable o el literal String.

```
String cadena = "Hola Mundo";

//Método para mostrar la longitud con un int
System.out.println( cadena.length() ); //Muestra el int 10

//Método para mostrar el caracter 4 de la cadena
System.out.println( cadena.charAt(3) ); //Muestra el caracter 'a'
```

Puedo consultar todos los métodos de la clase String: https://docs.oracle.com/javase/8/docs/api/java/lang/String.html [https://docs.oracle.com/javase/8/docs/api/java/lang/String.html]

Operador de concatenación

El operador + permite unir cadenas:

```
String cadena1 = "Mi";
String cadena2 = " primer";
String cadena3 = cadena1 + cadena2 + " programa";

System.out.println(cadena3); -> "Mi primer programa"
```

Siempre que concatenemos distintos tipos de datos con un String, da como resultado otro String:

```
int numero = 99;
double decimal = -35.14;

String cadena = numero + decimal + "";
//El resultado no es una suma, sino una concatenacion: "99-35.14"
```

Conversiones entre String y tipos primitivos

No se puede hacer un casting directo entre tipos de datos que no tiene nada que ver (cadenas de caracteres, y tipos primitivos). Debemos utilizar métodos que nos ayuden a hacer esas conversiones.

Convertir de String a tipo primitivo

Para poder convertir desde un valor contenido en un String al tipo primitivo de ese valor, usaremos los métodos de una serie de clases:

```
String cadenaEntero = "123";
int entero = Integer.parseInt(cadenaEntero);

boolean valor = Boolean.parseBoolean("true");

double decimal = Double.parseDouble("23.45");
```

Estas clases son conocidas como clases envoltorio, ya que se usan para envolver a tipos primitivos. Existe una clase por cada uno de los 8 tipos primitivos de datos.

Convertir de tipo primitivo a String

Para convertir de un tipo primitivo a String tenemos dos formas:

```
int numero = 46;
String cadenaNumerica = String.valueOf(numero);

double decimal = 34.75;
String cadenaDecimal = String.valueOf(decimal);
```

```
//El resultado de concatenar cualquier valor con un String, es otro String
String cadenaNumerica = numero + "";
```

Privilegios de la clase String

La clase String es inmutable: esto significa que su valor no se puede modificar. Cada vez que cambiamos el valor de una variable String, estamos construyendo un nuevo objeto String de forma transparente.

También permite una asignación directa de un valor literal para crear un objeto String:

```
String cadena = “hola”;
```

La sentencia anterior está construyendo un nuevo objeto (instancia)

El otro aspecto de la inmutabilidad se refleja en el siguiente ejemplo:

```
String cadena1 = new String("texto");
String cadena2 = new String("texto");

cadena1 == cadena2 //false, son objetos diferentes
cadena1.equals(cadena2) //true, contienen lo mismo

cadena1 = "otroTexto";
cadena2 = "otroTexto";

cadena1 == cadena2 //true, solo se ha creado un objeto
cadena1.equals(cadena2) //true, son el mismo objeto
```

Los Strings son objetos y se comparan con su método equals.

Cada vez que damos un valor literal a un String, primero se busca en el *String pool [http://www.myjavazone.com/2014/08/string-constant-pool-piscina-de.html]*. Si ya existe, no se crea un nuevo objeto y se usa el que ya existe.

Clase String
de fvaldeon