
ORACLE
Academy

Java Foundations

4-3

La clase String

ORACLE Academy

Objetivos

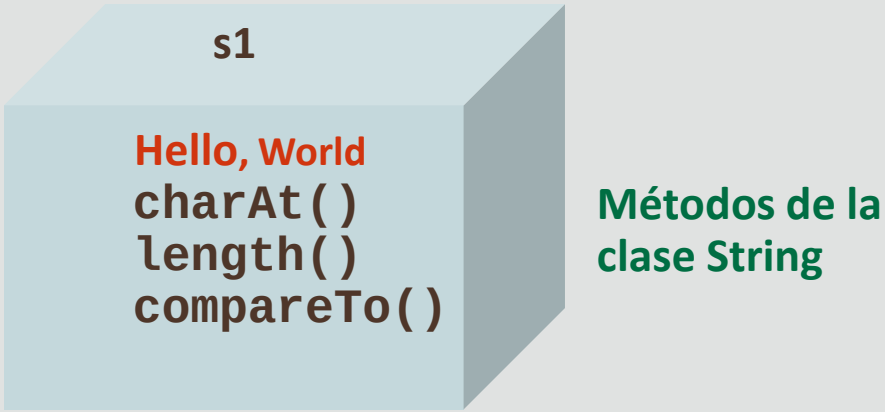
- En esta lección se abordan los siguientes objetivos:
  - Localizar la clase String en la documentación de la API de Java
  - Comprender los métodos de la clase String
  - Comparar lexicográficamente dos objetos String
  - Buscar la ubicación de una subcadena en un objeto String
  - Extraer una subcadena de un objeto String

¿Qué es una cadena?

- Una cadena es una secuencia de caracteres, incluidos letras del alfabeto, caracteres especiales y espacios en blanco
- Por ejemplo:
  - "¿Cómo estás?" es una cadena que contiene letras, espacios en blanco y un carácter especial ("?")
- En Java, las cadenas no son un tipo de dato primitivo
- En su lugar, son objetos de la clase String.

Representación de las cadenas en Java

- En Java, las cadenas son objetos de la clase denominada java.lang.String
- Ejemplo:
  - String s1= “Hello, World”;



Representación de las cadenas en Java

- Una cadena en java es más abstracta
- Es decir, no se supone que conoce su estructura interna, lo que hace que sea más fácil de utilizar
- Sus métodos permiten a un programador realizar operaciones en ella

Uso de la clase String

- La clase String:
  - Es una de las muchas clases incluidas en las bibliotecas de clases Java
  - Forma parte de java.lang.package
  - Proporciona la capacidad de mantener una secuencia de caracteres de datos
- Utilizará la clase String frecuentemente en sus programas
- Por lo tanto, es importante comprender algunas de las características especiales de las cadenas en Java

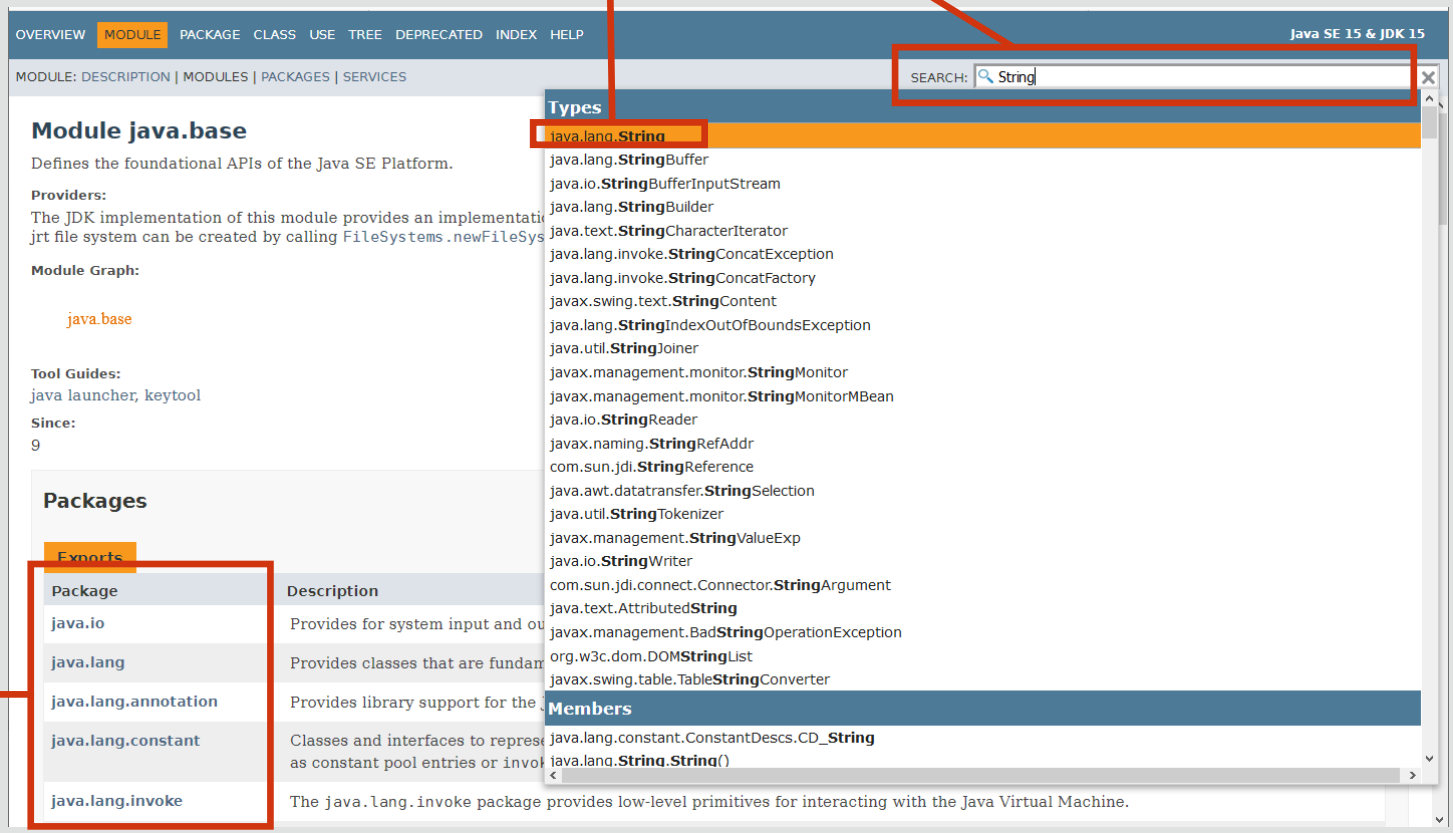
Documentación de la clase String

- Puede acceder a la documentación de la clase String de Java desde aquí:
  - https://docs.oracle.com/en/java/javase/17/docs/api/java.base/module-summary.html

Documentación de la plataforma Java SE 17 de la clase String

Busque un paquete aquí. Escriba String en el cuadro de búsqueda y, de los tipos mostrados, seleccione java.lang.String.

Desplácese hacia abajo y seleccione los paquetes aquí



ORACLE Academy

Documentación de la clase String: Resumen del Método

- `public int charAt(String str)`

Tipo de retorno del método

Nombre del método

Tipo de dato del parámetro que se debe transferir al método

Method Summary

All Methods | Static Methods | Instance Methods | Concrete Methods | Deprecated Methods

Modifier and Type	Method	Description
char	charAt(int index)	Returns the char value at the specified index.
IntStream	chars()	Returns a stream of int zero-extending the char values from this sequence.
int	codePointAt(int index)	Returns the character (Unicode code point) at the specified index.
int	codePointBefore(int index)	Returns the character (Unicode code point) before the specified index.
int	codePointCount(int beginIndex, int endIndex)	Returns the number of Unicode code points in the specified text range of this String.
IntStream	codePoints()	Returns a stream of code point values from this sequence.
int	compareTo(String anotherString)	Compares two strings lexicographically.
int	compareToIgnoreCase(String str)	Compares two strings lexicographically, ignoring case differences.
String	concat(String str)	Concatenates the specified string to the end of this string.

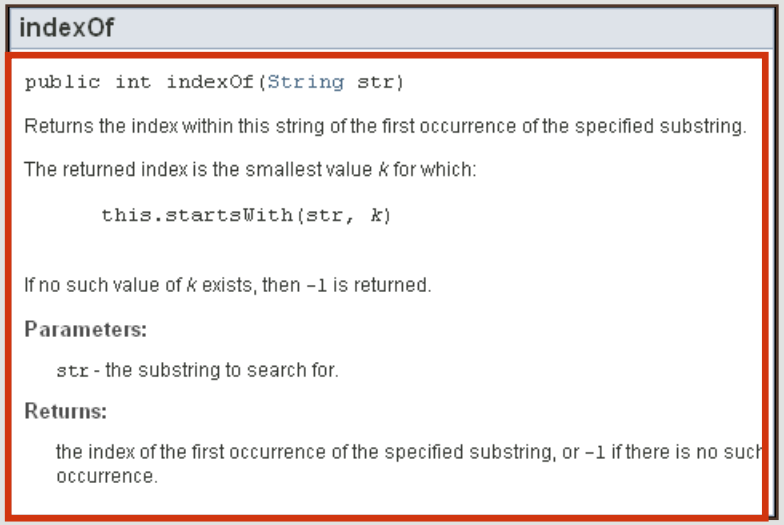
Documentación de la clase String: Detalles del método

Haga clic aquí para obtener la descripción detallada del método

int	indexOf(String str)
int	indexOf(String str, int fromIndex)

Descripción detallada del método indexOf()

Se muestran más detalles sobre los parámetros y el valor de retorno en la lista de métodos



indexOf

public int indexOf(String str)

Returns the index within this string of the first occurrence of the specified substring.

The returned index is the smallest value *k* for which:

this.startsWith(str, *k*)

If no such value of *k* exists, then -1 is returned.

Parameters:

str - the substring to search for.

Returns:

the index of the first occurrence of the specified substring, or -1 if there is no such occurrence.

Métodos de String: length

- Puede calcular la longitud de una cadena mediante el método length definido en la clase String:
  - Método: name.length()
  - Devuelve la longitud, o el número de caracteres, en nombre como un valor entero
- Ejemplo:

```
String name = "Mike.W";

System.out.println(name.length()); //6
```

Acceso a cada carácter en una cadena

- Puede acceder a cada carácter en una cadena a través de su índice numérico
- El primer carácter de la cadena está en el índice 0, el siguiente está en el índice 1 y así sucesivamente
- Por ejemplo:
- `String str= "Hello, World";`

H	e	l	l	o	,		W	o	r	l	d
0	1	2	3	4	5	6	7	8	9	10	11

  - str tiene de 0 a 11 índices, es decir, entre 0 y str.length()-1

Métodos de String: indexOf()

- Cada carácter de una cadena tiene un índice
- Puede recuperar el valor de índice de un carácter de la cadena mediante el método indexOf:

Método	Descripción
`str.indexOf(char c)`	Devuelve el valor de índice de la primera incidencia de c en la cadena str.
`s1.indexOf(char c,int beginIdx)`	Devuelve el valor de índice de la primera incidencia de c en la cadena s1, empezando por beginIdx hasta el final de la cadena.

Métodos de String: indexOf()

```
public static void main(String args[]){
        String phoneNum = "404-543-2345";
        int idx1 = phoneNum.indexOf('-');
        System.out.println("index of first dash: "+ idx1); //3
        int idx2 = phoneNum.indexOf('-’, idx1+1);
        System.out.println("second dash idx: "+ idx2); // 7
}//end method main
```

Métodos de String: charAt

- Devuelve el carácter de la cadena ubicado en el índice pasado como parámetro
- Método: str.charAt(int index)

```
String str = "Susan";
System.out.println(str.charAt(0));  //S
System.out.println(str.charAt(3));  //a
```

Métodos de String: substring()

- Puede extraer una subcadena de una cadena determinada
- Java ofrece dos métodos para esta operación:

Método	Descripción
`str.substring(int beginIdx)`	Devuelve la subcadena de beginIdx hasta el final de la cadena.
`str.substring(int beginIdx, int endIdx)`	Devuelve la subcadena de beginIdx hasta, pero sin incluir, endIdx.

Métodos de String: substring()

```
public static void main(String args[]){
    String greeting = "Hello, World!";
    String sub = greeting.substring(0, 5); → "Hello"
    String w = greeting.substring(7, 11); → "Worl"
    String tail = greeting.substring(7); → "World!"
}//end method main
```

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);
}//end method main
```

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

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)

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);
}//end method main
```

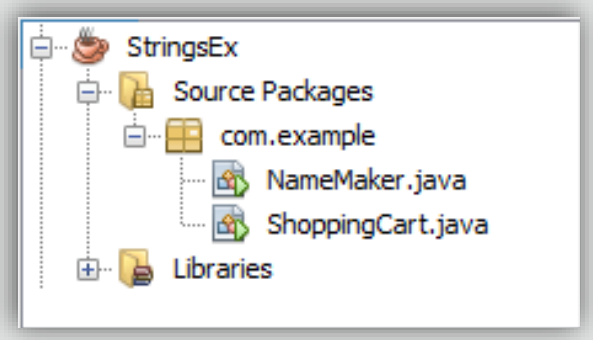
- Salida:
  - String REplace with replaceFirst
- Solo la primera incidencia de "re" se sustituye por "RE"
- La segunda incidencia no se cambia

Ejercicio 1, parte 1

- Cree un nuevo proyecto y agréguele los archivos `ShoppingCart.java` y `NameMaker.java`
- Examine `ShoppingCart.java`
- Realice lo siguiente:
  - Utilice el método indexOf para obtener el índice del carácter de espacio (" ") en custName
  - Asígnelo a spaceIdx
  - Utilice el método substring y spaceIdx para obtener la primera parte del nombre custName
  - Asígnelo a firstName e imprima firstName

Ejercicio 1, parte 2

- Puede que observe que este proyecto tiene dos archivos .java con métodos main
  - Puede parecer una contradicción porque dijimos que nunca se escribe más de un método main



- En ocasiones, los programadores lo hacen al probar pequeños bits de código y desean mantener todos sus archivos perfectamente en un proyecto
  - Lamentablemente, al pulsar Run en su IDE siempre se ejecuta el mismo archivo y nunca los demás
  - Tendrá que hacer clic con el botón derecho en el archivo alternativo que desea ejecutar Aparecerá un menú con una opción para ejecutar ese archivo

Declaración y creación de una cadena

- Puede instanciar las cadenas de dos maneras:
- Literales de cadena:
  - Asigne directamente un literal de cadena a una referencia de cadena


```
String hisName = "Fred Smith";
```

- Operador new: Similar a cualquier otra clase No se suelen utilizar y no se recomiendan

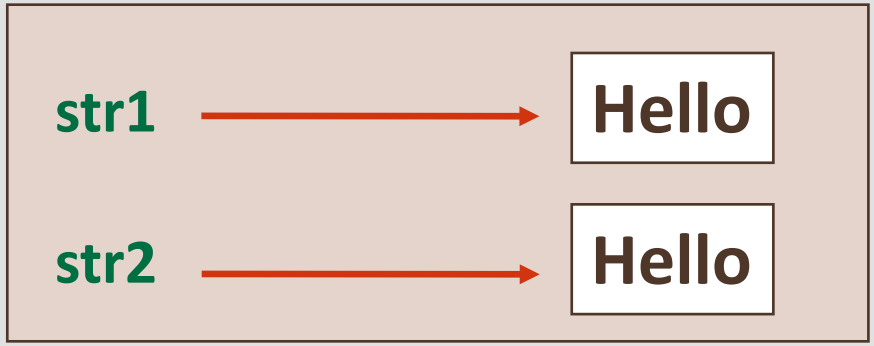
```
String herName = new String("Anne Smith");
```

Las cadenas son inmutables

- Un objeto String es inmutable; es decir, después de crear el objeto String, su valor no se puede cambiar
- Como las cadenas son inmutables, Java puede procesarlas de forma muy eficaz
  - Tenga en cuenta lo siguiente:

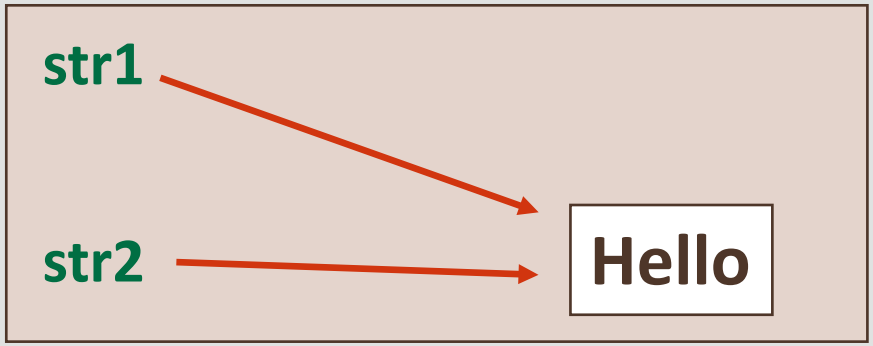
```
String str1 = "Hello";
String str2 = "Hello";
```

  - Esperamos esto...



Las cadenas son inmutables

- Pero esto es lo que ocurre...



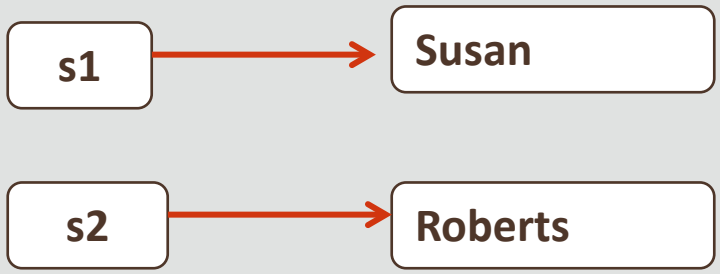
- El sistema de tiempo de ejecución de Java sabe que las dos cadenas son idénticas y asigna la misma ubicación de memoria para los dos objetos

Concatenación de cadenas

- En Java, la concatenación de cadenas forma una nueva cadena que es la combinación de varias cadenas
- Puede concatenar las cadenas de dos maneras en Java:
  - Operador de concatenación de cadenas **+**
  - Método **concat()**

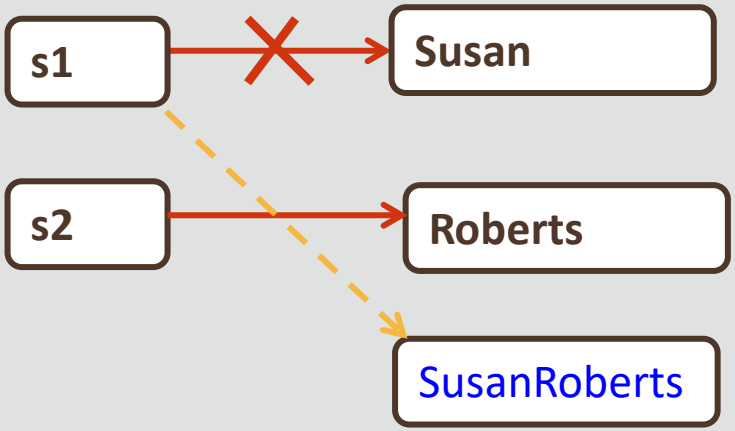
Uso del operador + (antes de la concatenación)

```
public static void main(String args[]) {
    String s1 = "Susan";
    String s2 = "Roberts";
}//end method main
```



Mediante el operador + (después de la concatenación)

```
public static void main(String args[]) {
    String s1 = "Susan";
    String s2 = "Roberts";
    S1 = s1 + s2;
    System.out.println(s1);
}//end method main
```



Concatenación de los datos que no son de cadena con cadena

- 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:

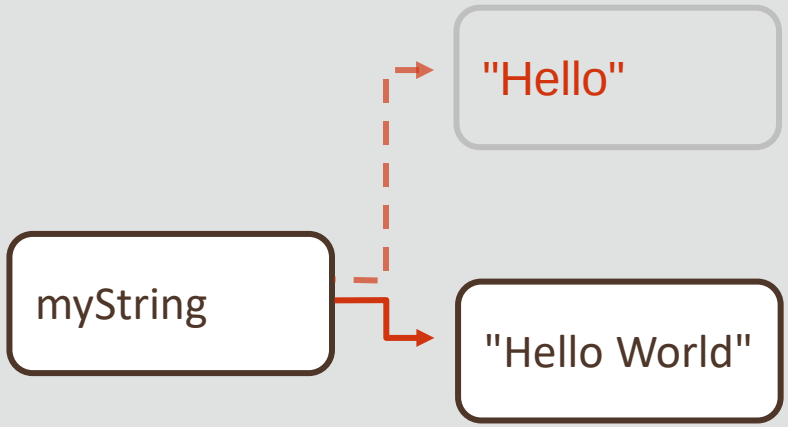
```
public static void main(String args[]) {
String newString = "Learning Java" + 17;
    System.out.println(newString);              //Learning Java 17

    System.out.println("Total : " + 17 + 17);   //Total: 1717
    System.out.println("Total : " + (17 + 17)); //Total: 34

    String numString1 = "17" + 17;
    System.out.println(numString1);             //1717
}//end method main
```

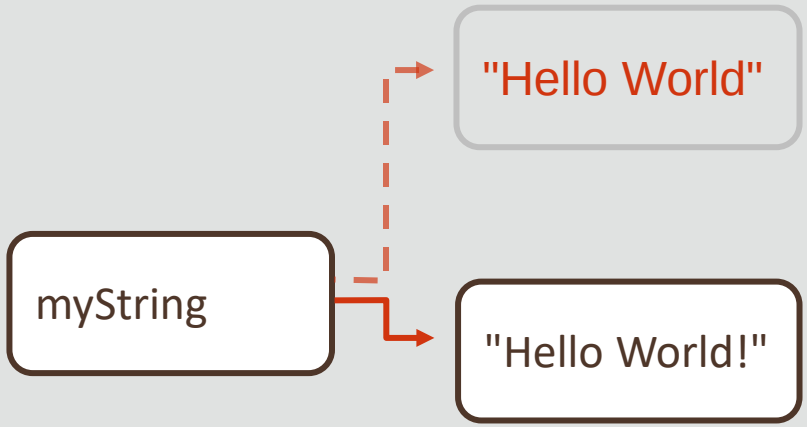
Uso del método concat() (antes de la concatenación)

```
String myString = "Hello";
myString = myString.concat(" World");
```



Uso del método concat() (después de la concatenación)

```
String myString = "Hello";
myString = myString.concat(" World");
myString = myString + "!"
```



Ejercicio 2

- Abra el proyecto que ha creado en el ejercicio 1
- Examine `NameMaker.java`
- Realice lo siguiente:
  - Declare las variables de cadena: firstName, middleName, lastName y fullName
  - Indique a los usuarios que introduzcan su nombre de pila, segundo nombre, apellidos y lea los nombres desde el teclado
  - Defina y muestre fullName como firstName+a blank char+middleName+a blank char+lastName

Ejercicio 2

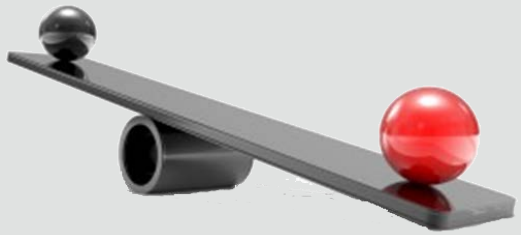
- ¿Qué cree que es preferible en este caso?
- Es decir, ¿el operador de concatenación de cadenas o el método concat()?

¿Cuál es la mejor forma de concatenar cadenas?

- Como se observa en el ejercicio anterior:
- Operador +:
  - Puede trabajar entre una cadena y una cadena, el valor del tipo de dato char, int, double o float
  - Convierte el valor a su representación de cadena antes de la concatenación
- Método concat():
  - Solo se puede llamar en cadenas
  - Comprueba la compatibilidad del tipo de dato y se produce un error de tiempo de compilación si no coinciden

¿Cómo se comparan los objetos String?

- Puede comparar dos objetos String mediante el método compareTo
- Este método compara según el orden lexicográfico de cadenas. Las comparaciones lexicográficas son similares al orden que encontramos en un diccionario
- Las cadenas se comparan carácter por carácter hasta que se determina su orden o hasta que demuestran que son idénticas
- Sintaxis: **s1.compareTo(s2)**
- Devuelve un valor entero que indica el orden de las dos cadenas



Valor devuelto por compareTo()

- El valor entero devuelto por el método compareTo() se puede interpretar como se indica a continuación:
  - Devuelve < 0 cuando la cadena que llama al método está lexicográficamente en primer lugar
  - Devuelve == 0 cuando las dos cadenas son lexicográficamente equivalentes
  - Devuelve > 0 cuando el parámetro transferido al método está lexicográficamente en primer lugar

Uso del método compareTo

- Observe algunos ejemplos:
  - "computer".compareTo("comparison")
    - Devuelve un entero > 0 porque el parámetro "comparison" está lexicográficamente en primer lugar
  - "cab".compareTo("car")
    - Devuelve un entero < 0 porque la cadena "cab" que llama al método está lexicográficamente en primer lugar
  - "car".compareTo("car")
    - Devuelve un entero igual a 0 porque ambos son lexicográficamente equivalentes

Uso del método compareTo: Ejemplo

- Vamos a escribir un programa para comparar nombres mediante el método compareTo:

```
public static void main(String[] args) {

    String s1 = "Susan";
    String s2 = "Susan";
    String s3 = "Robert";

    //Returns 0 because s1 is identical to s2
    System.out.println(s1.compareTo(s2)); //Output is 0

    //Returns >0 because ‘S’ follows‘R’
    System.out.println(s1.compareTo(s3)); // Output is 1

    //Returns <0 because ‘R’ precedes ‘S’
    System.out.println(s3.compareTo(s1)); // Output is -1
}//end method main
```

Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Localizar la clase String en la documentación de la API de Java
  - Comprender los métodos de la clase String
  - Comparar lexicográficamente dos objetos String
  - Buscar la ubicación de una subcadena en un objeto String
  - Extraer una subcadena de un objeto String


ORACLE
Academy