La clase String

https://docs.oracle.com/javase/9/docs/api/java/lang/String.html

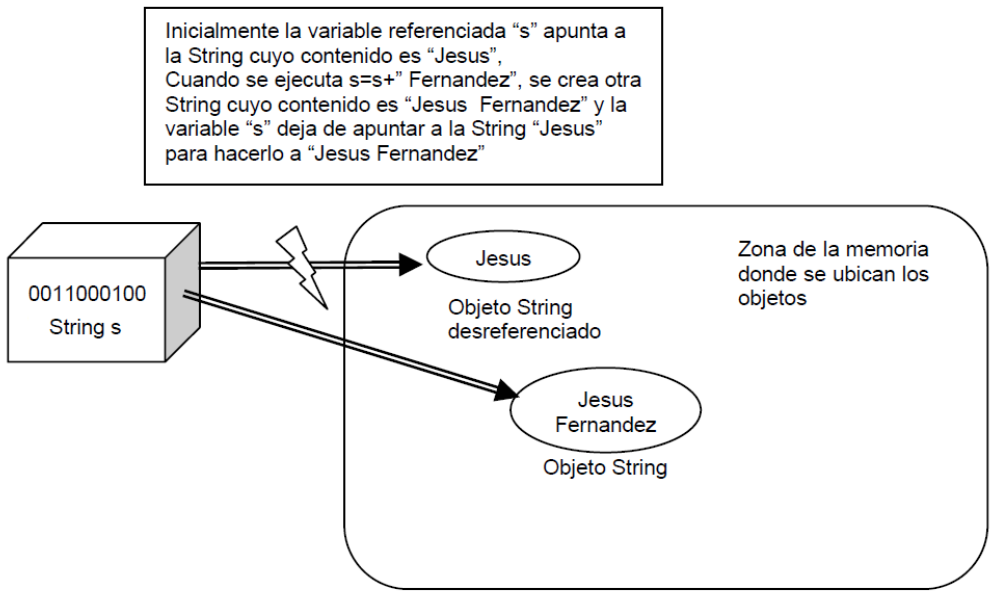
Una String es una variable referenciada asociada a un objeto de la clase java.lang.String. Se emplea para almacenar cadenas de caracteres.

Las cadenas tienen una característica que las diferencia del resto de objetos: son inmutables, es decir, cuando se intenta modificarlas, por ejemplo al aplicarles un método, no se modifican sino que se crea otra cadena nueva.

Ejemplo 1 (inmutabilidad): se observa que siendo una String una variable referenciada no se emplea constructor para crearla sino que se siguen las mismas pautas que en la variables primitivas. Cuando se vean los constructores de String se explicará el motivo.

```
//Creación de un objeto String al que apunta la variable referenciada s.
String s="Jesus";
//Parece que se modifica la String s añadiéndole " Fernandez",
//pero, en realidad, lo que ocurre es que se crea una nueva String
//con nuevo contenido: "Jesus Fernandez".
//Además, la variable referenciada "s" deja de apuntar a la String
//cuyo contenido es "Jesus" para hacerlo a la nueva.
//Se ha desreferenciado y vaga por la memoria sin rumbo ni destino
s=s+" Fernandez";

//Por consola: Jesus Fernandez
System.out.println(s);
```



Además de lo anterior las Strings tienen dos propiedades muy importantes:

- Una **String está indexada**, es decir, cada uno de sus caracteres tiene asociado un índice: 0 para el primero, 1 para el segundo, etc.
- La cadena de caracteres almacenada por una String siempre se escribe entre comillas dobles.

CONSTRUCTORES:

String(): construye un objeto de la clase String sin inicializar.

Ejemplo:

```
String cadena=new String();
cadena="Me llamo Jesus";
```

Se crea un objeto de la clase String sin inicializar, al que se asigna la referencia o nombre, cadena. Luego, se inicializa con "Me llamo Jesus".

String(String texto): contruye una String con otra que se le pasa al argumento del constructor.

Sin embargo, cuando se trabaja con Strings no suele emplearse ningún constructor sino que se hace como si fuera una variable primitiva (propiedad particular de String). Así, si se desea crear una String y mostrarla por consola, el código habitual sería:

```
String cadena="Esto es una cadena de texto";
System.out.println(cadena);
```

MÉTODOS PRINCIPALES: para poder aplicar estos métodos es necesario crear un objeto String.

- **int length():** devuelve la longitud de la String, incluyendo espacios en blanco.

  Ejemplo:

  ```
  String s="cucu";
  int longitud=s.length();
  System.out.println(longitud):
  ```

  Por consola:

  4

- **int indexOf(String str, int indice):** devuelve el índice en el que aparece por primera vez la String del primer argumento en la que se aplica el método, a partir del índice especificado en el segundo argumento. Recordar que una String está indexada. Si el índice a partir del que se inicia la búsqueda no existe o la String no aparece, devuelve –1. MUY USADO.

  Ejemplo:

```
String str="expedicion";

System.out.println(str.indexOf("x",str.length()));
```

Por consola:

–1 porque la búsqueda se inicia a partir de un índice que no existe ya que el índice mayor es la (longitud de la

String) –1.

• **int indexOf(char ch):** devuelve la posición en la que aparece por primera vez el carácter que se le pasa al argumento. Se observa que el nombre de este método es igual al anterior aunque su número de argumentos es distinto además de su tipo. A esto se le llama sobrecarga de métodos: mismo nombre pero distinto número de argumentos o distinto tipo de argumentos o distinto orden.

• **String replace (char viejoChar, char nuevoChar):** cambia el carácter asociado al primer argumento por el que se le pasa al segundo, de la String sobre la que se aplica el método generando una nueva. La String sobre la que se aplica el método no cambia, simplemente se crea otra nueva.

Ejemplo:

```
String s="cucu";

String s1=s.replace('u','o');

System.out.pritnln(s);

System.out.pritnln(s1);
```

Por consola:

cucu

coco

- **String toLowerCase():** genera una nueva String convirtiendo todos los caracteres de la String sobre la que se aplica el método, en minúsculas.
- **String toUpperCase():** genera una nueva String convirtiendo todos los caracteres de la String sobre la que se aplica el método, en mayúsculas.
- **boolean equals(String str):** investiga si dos String tienen los mismos caracteres y en el mismo orden. Si es así devuelve true y si no false. MUY USADO.
- **boolean equalsIgnoreCase(String str):** investiga si dos String tienen los mismos caracteres y en el mismo orden sin tener en cuenta las mayúsculas. Si es así devuelve true y si no false.
- **boolean startsWith(String str):** devuelve true si la String sobre la que se aplica comienza por la del argumento; false si esto no ocurre.

- **boolean startsWith(String str, int indice):** devuelve true si la String sobre la que se aplica comienza por la del argumento a partir de un determinado índice asociado al segundo argumento; false si esto no ocurre.
- **boolean endsWith(String str):** devuelve true si la String sobre la que se aplica acaba en la del argumento; false si esto no ocurre.
- **String trim():** devuelve una String en base a la que se le pasa al argumento, pero sin espacios en blanco al principio ni al final. No elimina los espacios en blanco situados entre las palabras.

```
Ejemplo:
String str=" hola que tal ";
System.out.println(str.length());
String strBis=str.trim();
System.out.println(strBis.length());
Por consola:
14
12
```

• **String substring(int indiceIni, int indiceFin):** devuelve una String obtenida a partir del índice inicial incluido y del índice final excluido, es como un intervalo semiabierto [indiceIni, indiceFin). Lanza una excepción del tipo IndexOutOfBoundsException si el índice final sobrepasa la longitud de la String.

```
Ejemplo:
String str="cucurrucucupaloma";
System.out.println(str.substring(4,9));
Por consola:
rrucu
```

• char charAt (int indice): devuelve el carácter asociado a la posición que se le pasa como argumento de la String sobre la que se aplica el método. Si el índice no existe se lanza una StringIndexOutOfBoundsException que hereda de IndexOutOfBoundsException.

IMPORTANTE: los métodos anteriores que devuelven una String NO MODIFICAN la cadenasobre la que se aplican; lo que hacen es crear otra nueva, en base a la que se emplea para aplicar el método.

Otra cuestión que se va a introducir y cuyo uso se verá más adelante es la siguiente: si se quiere convertir a String un objeto de cualquier clase, puede emplearse el método "void toString()" de java.lang.Object. Ojo pues este método sólo es aplicable a objetos, NO a variables primitivas.

Ejercicios

1. Realizar programa en el cúal se defina una cadena de caracteres y mostrar esa cadena por pantalla con la primera letra en mayúsculas y las demás en minúsculas.

2. Crea un programa Java que defina una cadena de caracteres y los muestre con la posición de sus caracteres invertida

3. Crea un programa Java que lea una cadena de teclado y la imprima al revés

4. Determina si una cadena leída por teclado es o no un palíndromo

5. Crea un programa Java que lea por teclado y muestre un mensaje con el número de veces que ha aparecido la letra A y la letra E.

6. Realizar un programa que haga lo siguiente:

   1) Pida al usuario un número entero comprendido entre [10,25] y repita la solicitud en caso de que el número no pertenezca al intervalo.

   2) Una vez introducido el número el programa debe detectar si es divisible por 5 y mostrar el mensaje adecuado.

   3) Luego, el programa pedirá al usuario su nombre para mostrarle lo siguiente:

      Número de caracteres del nombre

      Carácter inicial y final

      Nombre e mayúsculas

      Indicación de si la lentra ñ o la Ñ forman parte de su nombre

   NOTA 1: las excepciones deben gestionarse con la cláusula “throws IOException”.

   NOTA 2: la letra ñ es el carácter 164 de la tabla Unicode, la Ñ el 165.

7. Verificar si la cadena de texto almacenada en la String nif, es un NIF correcto o no. Si lo es, se mostrará por consola su parte numérica; si no lo es se mostrará el mensaje "NIF no valido". Se tendrá en cuenta lo siguiente: Los NIFs tienen 8 dígitos y, a continuación, una letra (no importa que sea mayúscula o minúscula).

8. Realizar una clase, que permita cargar una dirección de mail en el constructor, luego en otro método mostrar un mensaje si contiene el caracter '@'.

9. Desarrollar un programa que solicite la carga de una clave. La clase debe tener dos métodos uno para la carga y otro que muestre si la clave es la correcta (la clave a comparar es "123abc")

10. Codifique un programa que permita cargar una oración por teclado, luego mostrar cada palabra ingresada en una línea distinta.

```
Por ejemplo, si cargo:
  Talavera de la Reina

Deberá  aparecer:

Talavera
de
la
Reina
```