La clase String

La clase String

Introducción.

Para Java las cadenas de texto son **objetos** especiales. Los textos deben manejarse creando objetos de tipo **String**.
Ejemplo:

```
String texto1 = “¡Prueba de texto!”;
```

Las cadenas pueden ocupar varias líneas utilizando el operador de **concatenación “+”.**

```
String texto2 =”Este es un texto que ocupa “ +
“varias líneas, no obstante se puede “+
“perfectamente encadenar”;
```

También se pueden crear objetos String sin utilizar constantes entrecomilladas, usando otros constructores:

```
char[] palabra = {‘P’,’a’,’l’,’b’,’r’,’a’};//Array de char
String cadena = new String(palabra);
byte[] datos = {97,98,99};
String codificada = new String (datos, “8859_1”);
```

En el último ejemplo la cadena codificada se crea desde un array de tipo byte que contiene números que serán interpretados como códigos Unicode. Al asignar, el valor 8859_1 indica la tabla de códigos a utilizar.

comparación entre objetos String.

Los objetos String no pueden compararse directamente con los operadores de comparación. En su lugar se deben utilizar estos métodos:

- **cadena1.equals(cadena2)**. El resultado es true si la cadena1 es igual a la cadena2. Ambas cadenas son variables de tipo String.
- **cadena1.equalsIgnoreCase(cadena2)**. Como la anterior, pero en este caso no se tienen en cuenta mayúsculas y minúsculas.
- **s1.compareTo(s2)**. Compara ambas cadenas, considerando el orden alfabético.

  Si la primera cadena es mayor en orden alfabético que la segunda devuelve 1, si son iguales devuelve 0 y si es la segunda la mayor devuelve -1. Hay que tener en cuenta que el orden no es el del alfabeto español, sino que usa la tabla ASCII, en esa tabla la letra ñ es mucho mayor que la o.
- **s1.compareToIgnoreCase(s2**). Igual que la anterior, sólo que además ignora las mayúsculas (disponible desde Java 1.2)

String.valueOf.

Este método pertenece no sólo a la clase String, sino a otras clases y siempre es un método que convierte valores de una clase a otra. En el caso de los objetos String, permite convertir valores que no son de cadena a forma de cadena. Ejemplos:

```
String numero = String.valueOf(1234);
String fecha = String.valueOf(new Date());
```

En el ejemplo se observa que este método pertenece a la clase String directamente, no hay que utilizar el nombre del objeto creado (como se verá más adelante, es un método estático).

Recuerda

Las variables de tipo String son **objetos** y String es una **clase**. Tiene métodos que nos permiten manejar y manupular el contenido del objeto (equals, valueOf,...).

Dos objetos de tipo String **no se pueden comparar utlizando los operadores relacionales** ==, <=, ... se deben utilizar los métodos equals, equalsIgnoreCase, CompareTo o compareToIgnoreCase.

Para **convertir** un dato (de cualquier tipo) a String utilizaremos el método **String.valueOf().**

Por ejemplo:

```
String str1="El lenguaje Java";
String str2=new String("El lenguaje Java");
if(str1==str2){

        System.out.println("Los mismos objetos");

        }

        else{

        System.out.println("Distintos objetos");

}
if(str1.equals(str2)){

        System.out.println("El mismo contenido");

        } String str1="El lenguaje Java";
        String str2=str1;
        System.out.prinln("Son el mimso objeto "+(str1==str2);

        else{

        System.out.println("Distinto contenido");

}
```

Esta porción de código devolverá que *str1* y *str2* son dstintos objetos pero con el mismo contenido. *str1* y *str2* ocupan posiciones distintas en memoria pero guradan los mismos datos.

Cambiemos la segunda sentencia y escribamos

```
String str1="El lenguaje Java";

String str2=str1;

System.out.prinln("Son el mimso objeto "+(str1==str2));
```

Los objetos str1 y str2 guardan la misma referencia al objeto de la clase *String* creado. La expresión *(str1==str2)* devolverá **true**.

Métodos más utilizados

Son métodos que poseen las propios objetos de tipo String. Para utilizarlos basta con poner el nombre del método y sus parámetros después del nombre del objeto String.

objetoString.método(argumentos)

length(). Permite devolver la longitud de una cadena (el número de caracteres de la cadena):

```
String texto1=”Prueba”;
System.out.println(texto1.length());   //Escribe 6
```

concatenar cadenas. Se puede hacer de dos formas, utilizando el método **concat** o con el operador **+.**

```
String s1=”Buenos ”, s2=”días”, s3, s4;
s3 = s1 + s2;
s4 = s1.concat(s2);
```

charAt(). Devuelve un carácter de la cadena. El carácter a devolver se indica por su posición (***el primer carácter es la posición 0***) Si la posición es negativa o sobrepasa el tamaño de la cadena, ocurre un error de ejecución, una excepción tipo **IndexOutOfBounds-Exception (recuerda este tipo de error, se repetirá muchas veces).**

Sintaxis: strObj. charAt(index)

```
String s1=”Prueba”;
char c1=s1.charAt(2); //c1 valdrá ‘u’
```

substring(). Da como resultado una porción del texto de la cadena. La porción se toma desde una posición inicial hasta una posición final (sin incluir esa posición final). Si las posiciones indicadas no son válidas ocurre una excepción de tipo IndexOutOfBounds-Exception. Se empieza a contar desde la posición cero.

Sintaxis:
public String substring(int beginIndex)
public String substring(int beginIndex, int endIndex)

```
String s = "Manual Web";
s.substring(7); // Nos devuelve "Web".
```

indexOf(). Devuelve la primera posición en la que aparece un determinado texto en la cadena. En el caso de que la cadena buscada no se encuentre, devuelve -1. El texto a buscar puede ser char o String.

Sintaxis:

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

```
String s1=”Quería decirte que quiero que te vayas”;
System.out.println(s1.indexOf(“que”)); //Da 15
Se puede buscar desde una determinada posición. En el ejemplo anterior:
System.out.println(s1.indexOf(“que”,16)); //Ahora da 26
```

lastIndexOf(). Devuelve la última posición en la que aparece un determinado texto en la cadena. Es casi idéntica a la anterior, sólo que busca desde el final.

Sintaxis:

public int lastIndexOf(String str)
public int lastIndexOf(String str, int fromIndex)
public int lastIndexOf(int ch)
public int lastIndexOf(int ch, int fromIndex)

```
String s1="Quería decirte que quiero que te vayas";
System.out.println(s1.lastIndexOf("que"); //Da 26
```

También permite comenzar a buscar desde una determinada posición.

endsWith(). Devuelve true si la cadena termina con un determinado texto.

Sintaxis: public boolean endsWith(String suffix)

```
String s1="Quería decirte que quiero que te vayas";
System.out.println(s1.endsWith("vayas"); //Da true
```

startsWith(). Devuelve true si la cadena empieza con un determinado texto.

Sintaxis

public boolean startsWith(String prefix, int toffset)
public boolean startsWith(String prefix)

replace. Cambia todas las apariciones de un carácter por otro en el texto que se indique y lo almacena como resultado. El texto original no se cambia, por lo que hay que asignar el resultado de replace a un String para almacenar el texto cambiado:

Sintaxis: public String replace(char caracter_antiguo, char caracter_nuevo)

```
String s1="Mariposa";
System.out.println(s1.replace('a','e'));  //Da Meripose
System.out.println(s1);                   //Sigue valiendo Mariposa
```

replaceAll(). Modifica en un texto cada entrada de una cadena por otra y devuelve el resultado. El primer parámetro es el texto que se busca (que puede ser una expresión regular), el segundo parámetro es el texto con el que se reemplaza el buscado. La cadena original no se modifica.

Sintaxis: public String replaceAll(String regex, String replacement)

```
String s1="Cazar armadillos";
System.out.println(s1.replaceAll("ar","er"));   //resultado:  Cazer ermadillos
System.out.println(s1);//Sigue valiendo Cazar armadilos
```

toUpperCase(). Devuelve la versión en mayúsculas de la cadena.
toLowerCase(). Devuelve la versión en minúsculas de la cadena.
toCharArray(). Obtiene un array de caracteres a partir de una cadena.

Lista de métodos

lista completa de métodos.

Método	Descripción
char charAt(int index)	Proporciona el carácter que está en la posición dada por el entero index.
int compareTo(string s)	Compara las dos cadenas. Devuelve un valor menor que cero si la cadena s es mayor que la original, devuelve 0 si son iguales y devuelve un valor mayor que cero si s es menor que la original.
int compareToIgnoreCase(string s)	Compara dos cadenas, pero no tiene e cuenta si el texto es mayúsculas o no.
String concat(String s)	Añade la cadena s a la cadena original.
String copyValueOf(char[] data)	Produce un objeto String que es igual al array de caracteres data.
boolean endsWith(String s)	Devuelve true si la cadena termina con el texto s
boolean equals(String s)	Compara ambas cadenas, devuelve true si son iguales
boolean equalsIgnoreCase(String s)	Compara ambas cadenas sin tener en cuenta las mayúsculas y las minúsculas.
byte[] getBytes()	Devuelve un array de caracteres que toma a partir de la cadena de texto
void getBytes(int srcBegin, int srcEnd, char[] dest, int dstBegin);	Almacena el contenido de la cadena en el array de caracteres dest. Toma los caracteres desde la posición srcBegin hasta la posición srcEnd y les copia en el array desde la posición dstBegin
int indexOf(String s)	Devuelve la posición en la cadena del texto s
int indexOf(String s, int primeraPos)	Devuelve la posición en la cadena del texto s, empezando a buscar desde la posición PrimeraPos
int lastIndexOf(String s)	Devuelve la última posición en la cadena del texto s
int lastIndexOf(String s, int primeraPos)	Devuelve la última posición en la cadena del texto s, empezando a buscar desde la posición PrimeraPos
int length()	Devuelve la longitud de la cadena
String replace(char carAnterior, char ncarNuevo)	Devuelve una cadena idéntica al original pero que ha cambiando los caracteres iguales a carAnterior por carNuevo
String replaceFirst(String str1, String str2)	Cambia la primera aparición de la cadena str1 por la cadena str2
String replaceFirst(String str1, String str2)	Cambia la primera aparición de la cadena uno por la cadena dos
String replaceAll(String str1, String str2)	Cambia la todas las apariciones de la cadena uno por la cadena dos
String startsWith(String s)	Devuelve true si la cadena comienza con el texto s.
String substring(int primeraPos, int segundaPos)	Devuelve el texto que va desde primeraPos a segunaPos.
char[] toCharArray()	Devuelve un array de caracteres a partir de la cadena dada
String toLowerCase()	Convierte la cadena a minúsculas

String toLowerCase(Locale local)	Lo mismo pero siguiendo las instrucciones del argumento local
String toUpperCase()	Convierte la cadena a mayúsculas
String toUpperCase(Locale local)	Lo mismo pero siguiendo las instrucciones del argumento local
String trim()	Elimina los blancos que tenga la cadena tanto por delante como por detrás
static String valueOf(tipo elemento)	Devuelve la cadena que representa el valor elemento. Si elemento es booleano, por ejemplo devolvería una cadena con el valor true o false

Ejercicios

1.- Solitar una cadena por teclado y mostrar por pantalla la cantidad de vocales que tiene.

Ejemplo:
Entrada: cad = "Hola tu"
Salida: La cantidad de vocales es 3

2.- Pedir cadena cadena por teclado e invertir la misma y mostrar por pantalla.

Ejemplo:
Entrada: "casa blanca"
Salida: "acnalb asac"

3.- Pedir una cadena por teclado (cad) y un carácter (car), y calcular cuántas veces se repite el carácter car en la cadena cad,

Por ejemplo:
Entrada: cad = "casa blanca", car = 'a'
Salida: El caracter 'a' se repite 4 veces

4.- Ingresar una cadena y determinar cuantas palabras se encuentran en la cadena. Cada palabra se separa por medio de un espacio en blanco.

5.- Leer una frase por teclado y encontrar la palabra de mayor longitud. El programa debe imprimir tanto la palabra como el número de caracteres de la misma.

6.- Escriba un programa que calcule la frecuencia de aparición de las vocales de un texto porcionado por el usuario. Esta solución se debe presentar en forma de histograma, por ejmplo:

a 15 **********
e 8 ********

7.- Un grupo de inteligencia militar desea codificar los mensajes secretos de tal forma que no puedan ser interpretados con una lectura directa, para lo cual han establecido las siguientes reglas:

a) Todo mensaje debe tener sus letras en mayúsculas.
b) Reemplazar cada letra por la que sigue según abecedario, excepto Z que se deberá reemplazar con la letra A.
c) reemplazar cada dígito encontrado por el siguiente numero excepto el 9 que deberá ser reemplazado por el 0.

El programa pedirá un texto y mostraré su codificación.

EJERCICIOS VOLUNTARIOS

1.- Pedir un número telefónico en formato de cadena y luego lo convierta de la siguiente manera:

Número Telefónico: 5256284000
Nueva Cadena: (52)-5-6284000

2.- Hacer un programa que pida como datos dos cadenas de caracteres y forme una tercera cadena intercalando los caracteres de las palabras de las cadenas recibidas.

3.- Programa que pida un texto e indique el porcentaje de palabras que tienen menos de 5 caracteres y el porcentaje de palabras con 5 o más caracteres.

4.- Programa que lea una frase y a continuación visualice cada palabra de la frase en columnas, seguida del número de letras que

tiene cada palabra.