STRING Y MATH

MÓDULO: Programación en Java

ÍNDICE DE CONTENIDOS

1.OBJETIVOS

- Utilizar el objeto String para trabajar con datos de tipo cadena de texto.
- Conocer los métodos más importantes del objeto String, además de saber como aplicarlos.
- Aprender a utilizar el objeto Math.
- Trabajar con los diferentes métodos del objeto Math.

2. STRING

Vamos a utilizar clases importantes en el lenguaje Java y a crear objetos de dichas clases. Empezaremos por la clase *String* una de las más importantes del lenguaje Java.

Un String en Java representa una **cadena de caracteres no modificable**.

Todos los literales de la forma "*cualquier texto*", es decir, literales entre comillas dobles, que aparecen en un programa java se implementan como objetos de la clase String.

CONSTRUCTOR	DESCRIPCIÓN

String()	Constructor por defecto. El nuevo String toma el valor ""
	String s = new String(); //crea el string s vacio.
	Equivale a: String s = "";

String(String s)	Crea un nuevo String, copiando el que recibe como parámetro. String s = "hola"; String s1 = new String(s); //crea el String s1 y le copia el contenido de s
String(char[] v)	Crea un String y le asigna como valor los caracteres contenidos en el array recibido como parámetro. char [] a = {'a', 'b', 'c', 'd', 'e'}; String s = new String(a); //crea String s con valor "abcde"
String(char[] v, int pos, int n)	Crea un String y le asigna como valor los n caracteres contenidos en el array recibido como parámetro, a partir de la posición pos. char [] a = {'a', 'b', 'c', 'd', 'e'}; String s = new String(a, 1, 3); //crea String s con valor "bcd";

2. STRING

CREAR UN STRING

Se puede **crear un String** de varias formas, entre ellas:

- Utilizando una **cadena de caracteres** entre comillas:

String s1 = "abcdef";

- Utilizando **operador de concatenación +**con dos o más objetos String:

String s2 = s1 + "ghij"; //s2 contiene "abcdefghij"
String s3 = s1 + s2 + "klm"; //s3 contiene " abcdefabcdefghijklm"
Además la clase String proporciona varios **constructores**, entre ellos:

2. STRING
2.1 Métodos

La clase String proporciona métodos para el tratamiento de las cadenas de caracteres: acceso a caracteres individuales, buscar y extraer una subcadena, copiar cadenas, convertir cadenas a mayúsculas o minúsculas, etc.

METODO	SIGNIFICADO
boolean equals(String s1)	El método equals retorna true si el contenido de caracteres del parámetro s1 es exactamente igual a la cadena de caracteres del objeto que llama al método equals.
boolean equalsIgnoreCase(String s1)	El funcionamiento es casi exactamente igual que el método equals con la diferencia que no tiene en cuenta mayúsculas y minúsculas (si comparamos 'Ana' y 'ana' el método equalsIgnoreCase retorna true)
int compareTo(String s1)	Este método retorna un 0 si el contenido de s1 es exactamente igual al String contenido por el objeto que llama al método compareTo. Retorna un valor >0 si el contenido del String que llama al método compareTo es mayor alfabéticamente al parámetro s1.
char charAt(int pos)	Retorna un carácter del String, llega al método la posición del carácter a extraer.

2. STRING
2.1 Métodos

Los objetos String no son modificables.

Por lo tanto, los métodos que actúan sobre un String con la intención de modificarlo lo que hacen es crear un nuevo String a partir del original y devolverlo modificado.

Por ejemplo: Una operación como convertir a mayúsculas o minúsculas un String no lo modificará sino que creará y devolverá un nuevo String con el resultado de la operación.

El **recolector de basura** es el encargado de eliminar de forma automática los objetos a los que ya no hace referencia ninguna variable.

int length()	Retorna la cantidad de caracteres almacenados en el String
String substring(int pos1,int pos2)	Retorna un substring a partir de la posición indicada en el parámetro pos1 hasta la posición pos2 sin incluir dicha posición.
int indexOf(String s1)	Retorna -1 si el String que le pasamos como parámetro no está contenida en la cadena del objeto que llama al método. En caso que se encuentre contenido el String s1 retorna la posición donde comienza a repetirse.
String toUpperCase()	Retorna un String con el contenido convertido todo a mayúsculas
String toLowerCase()	Retorna un String con el contenido convertido todo a minúsculas.

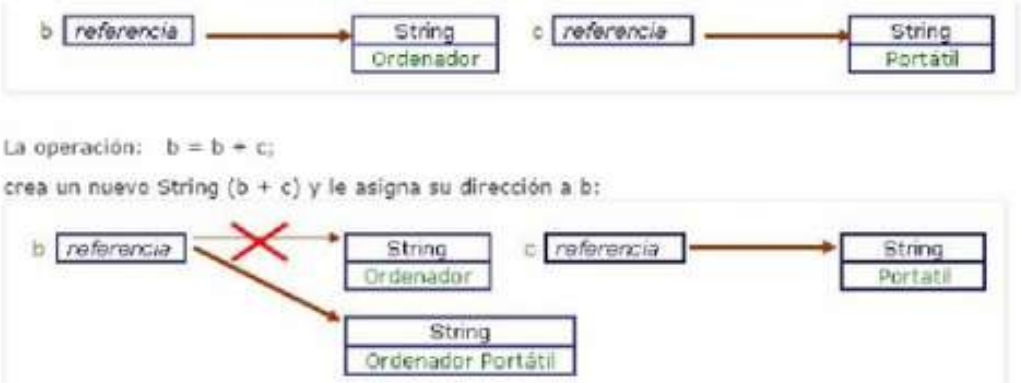
2. STRING
2.1 Métodos

EL OPERADOR DE CONCATENACIÓN +

- La clase proporciona el operador + (concatenación) para unir dos o más String.
- El resultado de aplicar este operador es un nuevo String concatenación de los otros.

Por ejemplo:

String b = "Ordenador";
String c = " Portátil";



Los String son constantes; sus valores no se pueden cambiar después de su creación.

2. STRING

2.1 Métodos

```
import java.util.Scanner;
public class Cadena1 {
public static void main(String[] args) {
        Scanner teclado=new Scanner(System.in);
        String cad1;
        String cad2;
        System.out.print("Ingrese la primera cadena:");
        cad1=teclado.nextLine();
        System.out.print("Ingrese la segunda cadena:");
        cad2=teclado.nextLine();
        if (cad1.equals(cad2)==true) {
                System.out.println(cad1+" es exactamente igual a "+cad2);
        } else {
                System.out.println(cad1+" no es exactamente igual a "+cad2);
        }
        if (cad1.equalsIgnoreCase(cad2)==true) {
                System.out.println(cad1+" es igual a "+cad2+
                " sin tener en cuenta mayúsculas/minúsculas");
        } else {
                System.out.println(cad1+" no es igual a "+cad2+
                " sin tener en cuenta mayúsculas/minúsculas");
        }
        if (cad1.compareTo(cad2)==0) {
                System.out.println(cad1+" es exactamente igual a "+cad2);
        } else {
                if (cad1.compareTo(cad2)>0) {
                        System.out.println(cad1+ " es mayor
                        alfabéticamente que "+cad2);
        } else
                        System.out.println(cad2+ " es mayor
                        alfabéticamente que "+cad1);
        }
```

```
        System.out.println("El primer carácter de "+cad1+
                                                " es "+cad1.charAt(0));
        System.out.println("La longitud de la cadena "+cad1+" es "+
                                                cad1.length());
        System.out.println("Los primeros tres caracteres de "+cad1+" son  "+
                                                cad1.substring(0,3));
        int posi=cad1.indexOf(cad2);
        if (posi==-1) {
                System.out.println(cad2+" no está contenido en "+cad1);
        } else {
                System.out.println(cad2+" está contenido en "+cad1+
                " a partir de la posición "+posi);
        }
System.out.println(cad1+ " convertido a mayúsculas es "+cad1.toUpperCase());
System.out.println(cad1+ " convertido a minúsculas es "+cad1.toLowerCase());
    }
}
```

2. STRING

2.1 Métodos

Ejemplo:

Crear dos objetos persona que nos digan si los dos apellidos de cada persona son iguales. Realizarlo con constructores para inicializar los objetos personas . Para comparar los apellidos utilizaremos el método comparar()

```
package curso2018;
import java.util.Scanner;

public class Apellidos {

    Scanner teclado=new Scanner(System.in);
    String   apellido1;
    String   apellido2;
    String   nombre;

public Apellidos(String apl,String apel){
    apellido1=apl;
    apellido2=apel;
}
public Apellidos(String nomb,String apl,String apel){
    nombre=nomb;
    apellido1=apl;
    apellido2=apel;

}
public  void comparar(){

    if (apellido1.equals(apellido2)==true) {
        System.out.println(apellido1+" es exactamente igual a "+apellido2);
        System.out.print(nombre+"\t");
        System.out.print(apellido1+"\t");
        System.out.println(apellido2);
        } else {
        System.out.println(apellido1+" no es exactamente igual a "+apellido2);

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


Apellidos persona1 =new Apellidos("sanchez","perez");
Apellidos persona2 =new Apellidos("Pilar","Sanchez","Sanchez");
        persona1.comparar();
        persona2.comparar();
}
```

NO

3. STRINGBUFFER Y STRINGBUILDER

La clase String es una clase no modificable. Esto quiere decir que cuando se modifica un String se crea un nuevo objeto String modificado a partir del original y el recolector de basura es el encargado de eliminar de la memoria el String original.

Java proporciona la clase StringBuffer y a partir de Java 5 la clase StringBuilder para trabajar con cadenas de caracteres sobre las que vamos a realizar modificaciones frecuentes de su contenido.

CONSTRUCTOR	DESCRIPCIÓN
StringBuilder ()	Crea un StringBuilder vacío. StringBuilder sb = new StringBuilder ();
StringBuilder(int n)	Crea un StringBuilder vacío con capacidad para n caracteres.
StringBuilder(String s);	Crea un StringBuilder y le asigna el contenido del String s. String s = "ejemplo"; StringBuilder sb = new StringBuilder (s);

NO

3. STRINGBUFFER Y STRINGBUILDER

La **diferencia entre StringBuffer y StringBuilder** es que los métodos de StringBuffer están sincronizados y los de StringBuilder no lo están. Por este motivo StringBuilder ofrece mejor rendimiento que StringBuffer y la utilizaremos cuando la aplicación tenga un solo hilo de ejecución.

En general decidiremos cuando usar String, StringBuilder o StringBuffer según lo siguiente:

- Usaremos **String** si la cadena de caracteres no va a cambiar.
- Usaremos **StringBuilder** si la cadena de caracteres puede cambiar y solamente tenemos un hilo de ejecución.
- Usaremos **StringBuffer** si la cadena de caracteres puede cambiar y tenemos varios hilos de ejecución.

CONSTRUCTORES DE LA CLASE STRINGBUILDER

La clase StringBuilder proporcionan varios **constructores**.

NO

3. STRINGBUFFER Y STRINGBUILDER

DIGI TECH
Centro Oficial FP Digital & Tech

Retorno	Método	Descripción
StringBuilder	append(...)	Añade al final del StringBuilder a la que se aplica, un String o la representación en forma de String de un dato asociado a una variable primitiva
int	length()	Devuelve el número de caracteres del StringBuilder
StringBuilder	reverse()	Invierte el orden de los caracteres del StringBuilder
void	setCharAt(int indice,char ch)	Cambia el carácter indicado en el primer argumento por el carácter que se le pasa en el segundo
char	charAt(int indice)	Devuelve el carácter asociado a la posición que se le indica en el argumento
void	setLength(int nuevaLongitud)	Modifica la longitud. La nueva longitud no puede ser menor
String	toString()	Convierte un StringBuilder en un String
StringBuilder	insert(int indiceIni,String cadena)	Añade la cadena del segundo argumento a partir de la posición indicada en el primero
StringBuilder	delete(int indiceIni,int indiceFin)	Borra la cadena de caracteres incluidos entre los dos índices indicados en los argumentos
StringBuilder	deleteChar(int indice)	Borra el carácter indicado en el índice
StringBuilder	replace(int indiceIni, int indiceFin,String str)	Reemplaza los caracteres comprendidos entre los dos índices por la cadena que se le pasa en el argumento
int	indexOf (String str)	Analiza los caracteres de la cadena y encuentra el primer índice que coincide con el valor deseado
String	subString(int indiceIni,int indiceFin)	Devuelve una cadena comprendida entre los dos índices

3. STRINGBUFFER Y STRINGBUILDER

NO

Ejemplo
Realizar un programa que realice lo siguiente:

a) Crear un StringBuilder con la cadena "Hola Caracola" y mostrarla por consola.

b) Mostrar por consola su longitud.

c) Partiendo de la cadena anterior y usando los métodos de StringBuilder modificar la cadena para que pase a ser "Hay Caracolas" y mostrarla.

d) Partiendo de la cadena anterior y usando los métodos de StringBuilder modificar la cadena para que pase a ser "Hay 5000 Caracolas" y mostrarla.

e) Partiendo de la cadena anterior y usando los métodos de StringBuilder modificar la cadena para que pase a ser "Hay 5000 Caracolas en el mar" y mostrarla.

```
package cadena;

public class cadenaStringBuffer {


              public static void main(String[] args) {

             //a) Crear un StringBuilder con la cadena "Hola
Caracola" y mostrarla por consola.
               StringBuilder s = new StringBuilder("hola Caracola");
               System.out.println(s);

             //b)Mostrar por consola su  longitud.
               System.out.println ("Longitud de la cadena:
"+s.length()+".");

             /*c)Modificar la cadena para que pase a ser "Hay
Caracolas" y mostrarla*/
               System.out.println (s.replace(s.indexOf("ola
Caracola"), s.indexOf(" Caracola"),"ay").append("s"));
           //c)Otra forma
             s.delete(0, s.indexOf("Caracola"));
             s.insert(s.indexOf("Caracola"),"hay ");
             s.append("s");
             System.out.println(s);
   /*d) Modificar la cadena para que pase a ser "Hay 5000 Caracolas" y
mostrarla. */
             s.insert(s.indexOf("Caracolas"),"5000 ");
               System.out.println(s);

             /*e) Partiendo de la cadena anterior y usando los
métodos de StringBuilder modificar la cadena para que pase a ser "Hay
5000 Caracolas en el mar" y mostrarla*/
               s.append(" en el Mar");
               System.out.println(s);
              }
}
```

4. CLASE MATH

Se echan de menos operadores matemáticos más potentes en Java. Por ello se ha incluido una clase especial llamada Math dentro del paquete java.lang.

Para poder utilizar esta clase, se debe incluir esta instrucción:
import java.lang.Math;

Si no, se invocan siempre de la siguiente manera:

Math.funcion(argumentos)
Math posee dos constantes, que son:

CONSTANTE	SIGNIFICADO
final static double E	El número e (2, 7182818245...)
final static double PI	El número Π (3,14159265...)

4. CLASE MATH

4.1 Métodos

Esta clase posee métodos muy interesantes para realizar cálculos matemáticos complejos:

METODO	SIGNIFICADO
`double ceil(double x)`	Redondea x al entero mayor siguiente
`double floor(double x)`	Redondea *x* al entero menor siguiente
`int round(double x)`	Redondea *x* de forma clásica
`double random()`	Número aleatorio [0.0, 1.0).
`tiponúmero abs(tiponúmero x)`	Devuelve el valor absoluto de *x*.
`double sqrt(double x)`	Calcula la raíz cuadrada de *x*
`double pow(double x, double y)`	Calcula x^y
`tiponúmero min(tiponúmero x,tiponúmero y)`	Devuelve el menor valor de *x* o *y*
`tiponúmero max(tiponúmero x,tiponúmero y)`	Devuelve el mayor valor de *x* o *y*

4. CLASE MATH

Simple lanzamiento de dados, y que señale el dado ganador.

```
public static void main(String args[]){
        System.out.println("Se presentan los participantes.
        A continuación, hará su lanzamiento el jugador 1");
        double aux1= (Math.random()*6)+0.5;
        double aux2= (Math.random()*6)+0.5;
        double intento1= Math.round(aux1);
        double intento2= Math.round(aux2);
        int jugador1= (int)intento1;
        int jugador2= (int)intento2;
        System.out.println("El jugador 1 ha sacado: "+jugador1+
                        " y el jugador 2 ha sacado"+jugador2);
        int ganador = Math.max(jugador1, jugador2);
        System.out.println("Quien sacó "+ganador+" es el ganador");
}
```

La fórmula general para obtener números aleatorios entre *num1* y *num2*(ambos inclusive) es:

```
int numAleatorio=(int)Math.floor(Math.random()*(num2-num1+1)+num1);
```

RESUMEN DE LA UNIDAD

- Los objetos String en Java son inmutables, lo que significa que una vez creados, su valor no puede ser cambiado; cualquier operación que modifique una cadena crea un nuevo objeto String en lugar de alterar el existente.
- La clase String proporciona una amplia variedad de métodos para manipular y operar sobre cadenas de texto, incluyendo operaciones como comparación, búsqueda, extracción de subcadenas, reemplazo y conversión de mayúsculas a minúsculas, entre otras.
- El objeto Math en Java proporciona métodos estáticos para realizar operaciones matemáticas comunes como cálculos trigonométricos (sin, cos, tan), exponenciales (exp, log), funciones de redondeo (round, ceil, floor), y operaciones básicas como max, min, entre otros
- Los métodos en Math están diseñados para proporcionar resultados precisos y consistentes según los estándares definidos por la especificación Java, facilitando operaciones matemáticas avanzadas sin la necesidad de implementar funciones personalizadas.

BIBLIOGRAFÍA/WEBGRAFÍA

- Thierry Richard. "Los fundamentos del lenguaje Java".
- Vozmediano,A.M. "Java para novatos".
- Jimenez, Alfonso."Aprender a programar en Java".
- Gervás, Luc. "Aprender los fundamentos del lenguaje Java"

RECURSOS DE INTERÉS

- **Utilización de cadenas de texto en Java:** https://www.youtube.com/watch?v=YMyJLs22uY4
- **Método Java:** https://l1nq.com/452jJ
- **Math:** https://l1nq.com/jFW15
- **Clase Math:** https://encr.pw/SJDE4