

Resolución de Problemas y Algoritmos

Cadenas de Caracteres

Facultad de Informática
Universidad Nacional del Comahue

2021

- 1 Cadenas de Caracteres
- 2 Declaración
- 3 Posiciones en una Cadena de Caracteres
- 4 Concatenación de Cadenas de Caracteres
- 5 Tipos de Recorridos
- 6 Clases, objetos y métodos
- 7 Métodos asociados a Cadenas de Caracteres
- 8 Comparación de Cadenas de Caracteres

- 1 Cadenas de Caracteres
- 2 Declaración
- 3 Posiciones en una Cadena de Caracteres
- 4 Concatenación de Cadenas de Caracteres
- 5 Tipos de Recorridos
- 6 Clases, objetos y métodos
- 7 Métodos asociados a Cadenas de Caracteres
- 8 Comparación de Cadenas de Caracteres

Cadenas de Caracteres.

- Una **cadena de caracteres** es una secuencia de caracteres que se representa como un ítem único.
- Una **cadena de caracteres** se declara con la palabra TEXTO en pseudocódigo, y con String en JAVA.
- Una **cadena de caracteres** tiene una notación especial, está rodeada de comillas dobles. Notación valida para pseudocódigo y Java.
- Ya hemos usado constantes de tipo cadena de caracteres ó String. Netbeans suele mostrar en color naranja todas las cadenas de caracteres. Ej:

"Ingrese **s** para **si** y **n** para **no**"

- Hoy veremos que operaciones podemos realizar con una cadena de caracteres.
- Tambien veremos como tratar con cadenas de caracteres recorriendo cada uno de sus caracteres en la secuencia que representa una cadena de este tipo.

Temario

- 1 Cadenas de Caracteres
- 2 **Declaración**
- 3 Posiciones en una Cadena de Caracteres
- 4 Concatenación de Cadenas de Caracteres
- 5 Tipos de Recorridos
- 6 Clases, objetos y métodos
- 7 Métodos asociados a Cadenas de Caracteres
- 8 Comparación de Cadenas de Caracteres

Pseudocódigo: Declaración en Pseudocódigo

```
1  ALGORITMO principal() RETORNA VACIO
2      TEXTO cadena
3      ESCRIBIR(" Ingrese una cadena de caracteres:")
4      LEER(cadena)
5      ESCRIBIR(cadena)
6  FIN ALGORITMO principal
```

Declaración en JAVA de una cadena de Caracteres

```
1 public static void main(String[] args) {  
2     // ... comentario ...  
3     String cadena;  
4     System.out.println("Ingresa una cadena de caracteres:");  
5     cadena = TecladoIn.readLine();  
6     System.out.print(cadena);  
7 }
```

Cadenas de Caracteres en JAVA.

Más sobre Declaración en JAVA

- Declaración de una cadena de Caracteres.
- String es una clase. Notar que su nombre comienza con mayúscula a diferencia de todos los tipos de datos primitivos de JAVA. String no es un tipo primitivo.

```
String saludo;  
saludo = "Hola";
```

ó

```
String saludo = "Hola";
```

ó

```
String saludo = new String("Hola");
```



Cadenas de Caracteres en JAVA.

Mostrar su contenido y su longitud

```
String saludo;  
saludo = "Hola";
```

ó

```
String saludo = "Hola";
```

- Para imprimir:

```
System.out.println(saludo);
```

- La longitud de la cadena saludo es 4. En pseudocódigo la longitud de una cadena se obtiene con longitud(saludo) y en Java saludo.length().

```
System.out.println(saludo.length());
```

Temario

- 1 Cadenas de Caracteres
- 2 Declaración
- 3 Posiciones en una Cadena de Caracteres**
- 4 Concatenación de Cadenas de Caracteres
- 5 Tipos de Recorridos
- 6 Clases, objetos y métodos
- 7 Métodos asociados a Cadenas de Caracteres
- 8 Comparación de Cadenas de Caracteres

Cadenas de Caracteres

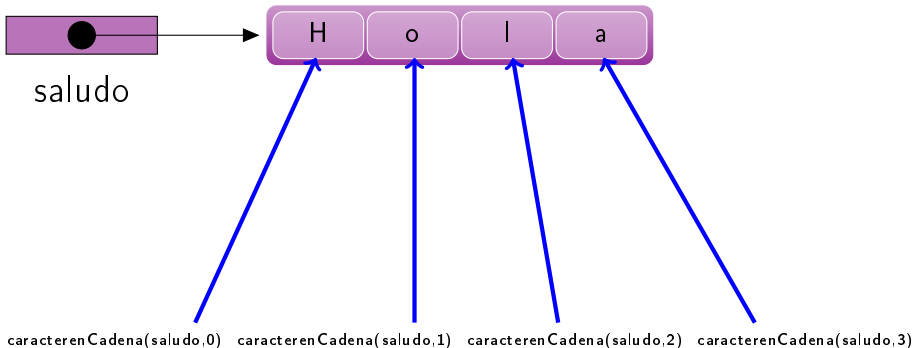
Posiciones en un String

- Las posiciones dentro de un String comienzan en 0.
- Ejemplo: En la cadena "Java es divertido" la 'J' esta en la posición 0

Cadenas de Caracteres

Posiciones dentro de la cadena

```
TEXTO saludo  
saludo <— "Hola"
```



Cadenas de Caracteres

Posiciones en un String

- Las posiciones dentro de un String comienzan en 0.
- Ejemplo: En la cadena "Java es divertido" la 'J' esta en la posición 0
- Otro ejemplo:

```
TEXTO cartel  
cartel ← "Bienvenido_a_Java"  
ESCRIBIR(caracterenPosicion(cartel,4))  
ESCRIBIR(subcadena(cartel,3,9))
```

```
String cartel = "Bienvenido_a_Java";  
System.out.println(cartel.charAt(4));  
System.out.println(cartel.substring(3,9));
```

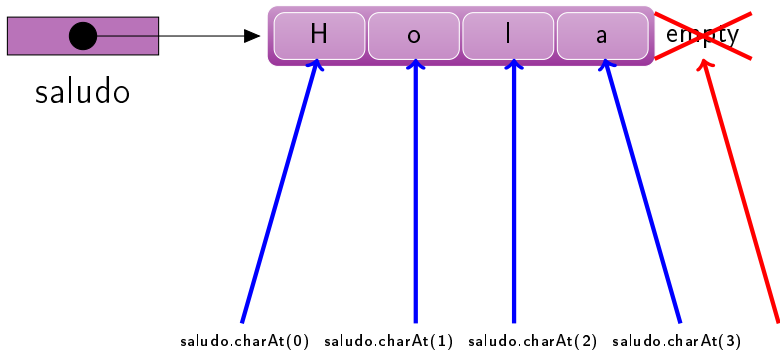
Un recuadro amarillo que representa una cadena de caracteres. Contiene el texto "v" en la primera línea y "nvenid" en la segunda línea, lo que sugiere una visualización de los caracteres de una cadena como "v" y "nvenid" (posiblemente parte de "Bienvenido").

v
nvenid

Cadenas de Caracteres

Posiciones dentro de la cadena

```
String saludo;  
saludo = "Hola";
```



Fuera de Rango

Temario

- 1 Cadenas de Caracteres
- 2 Declaración
- 3 Posiciones en una Cadena de Caracteres
- 4 Concatenación de Cadenas de Caracteres**
- 5 Tipos de Recorridos
- 6 Clases, objetos y métodos
- 7 Métodos asociados a Cadenas de Caracteres
- 8 Comparación de Cadenas de Caracteres

Cadenas de Caracteres.

Concatenación de Strings en Pseudocódigo

- Dos Strings se concatenan usando el módulo concatenar(cadena1,cadena2)

```
TEXTO saludo <— "Hola"  
TEXTO cadena  
cadena <— concatenar(saludo , " operador")  
ESCRIBIR (cadena)
```

Hola operador

- Puede concatenar cualquier cantidad de Strings.
- Un string y un número entero tambien pueden concatenarse de la misma manera:

```
TEXTO cadena  
cadena <— concatenar ("La temperatura es ", 72)  
ESCRIBIR (cadena)
```

La temperatura es 72

Cadenas de Caracteres.

Concatenación de Strings en JAVA

- Dos Strings se concatenan usando el operador +

```
String saludo = "Hola";  
String cadena;  
cadena = saludo + " operador";  
System.out.println(cadena);
```

Hola operador

- La tercera linea también se puede escribir así:

```
cadena = saludo.concat(" operador");
```

- Puede concatenar cualquier cantidad de Strings.
- Un string y un número entero tambien pueden concatenarse de la misma manera:

```
String cadena;  
cadena = "La temperatura es " + 72;  
System.out.println(cadena);
```

La temperatura es 72

Pseudocódigo

```
1  ALGORITMO principal() RETORNA VACIO
2      TEXTO cadena
3      ENTERO i
4      ESCRIBIR("Ingrese una cadena de caracteres:")
5      LEER(cadena)
6      ESCRIBIR("Sus caracteres son:")
7      PARA i <— 0 HASTA longitud(cadena)-1 PASO 1 HACER
8          ESCRIBIR("posicion " + i + ":" +
9                  caracterEnPosicion(cadena, i))
10     FIN PARA
11 FIN ALGORITMO principal
```

cadena	i	salida
		"Ingrese una cadena de caracteres:"
"casa"		
		"Sus caracteres son: "
	0	posicion 0: 'c'
	1	posicion 1: 'a'
	2	posicion 2: 's'
	3	posicion 3: 'a'
	4	

Pseudocódigo

```
1 public static void main(String[] args) {
2     String cadena;
3     int i;
4     System.out.println("Ingrese una cadena de caracteres:");
5     cadena = Teclado.readLine();
6     System.out.println("Sus caracteres son:");
7     for (i=0; i<cadena.length(); i++){
8         System.out.println("posicion" + i + ": "
9                             + cadena.charAt(i));
10    }
11 }
```

cadena	i	salida
"casa"	0	"Ingrese una cadena de caracteres:"
	1	"Sus caracteres son:"
	2	posicion 0: 'c'
	3	posicion 1: 'a'
	4	posicion 2: 's'
		posicion 3: 'a'

Pseudocódigo

```
1  ALGORITMO principal() RETORNA VACIO
2      TEXTO cadena
3      ESCRIBIR ("Ingrese una cadena de caracteres:")
4      LEER(cadena)
5      ESCRIBIR ("Su longitud es:")
6      ESCRIBIR (longitud(cadena))
7      ESCRIBIR ("La cadena invertida es:")
8      ESCRIBIR (invertir(cadena))
9  FIN ALGORITMO principal
10
11 MODULO invertir(TEXTO cad) RETORNA TEXTO
12     ENTERO i
13     TEXTO res
14     res ← ""
15     PARA i ← 0 HASTA longitud(cad)-1 PASO 1 HACER
16         res ← caracterenPosicion(cad, i) + res
17     FIN PARA
18     RETORNA res
19 FIN MODULO invertir
```

Pseudocódigo

```
1 ALGORITMO principal() RETORNA VACIO
2     TEXTO cadena
3     ESCRIBIR ("Ingrese una cadena de caracteres:")
4     LEER(cadena)
5     ESCRIBIR ("Su longitud es: ")
6     ESCRIBIR (longitud(cadena))
7     ESCRIBIR ("La cadena invertida es: ")
8     ESCRIBIR (invertir(cadena))
9 FIN ALGORITMO principal
```

cadena	salida
"casa"	"Ingrese una cadena de caracteres:"
	"Su longitud es: "
	4
	"La cadena invertida es: "

cad	i	res	valor retornado

Pseudocódigo

```
1 MODULO invertir(TEXTO cad) RETORNA TEXTO
2     ENTERO i
3     TEXTO res
4     res <— ""
5     PARA i <— 0 HASTA longitud(cad)-1 PASO 1 HACER
6         res <— caracterenPosicion(cad, i) + res
7     FIN PARA
8     RETORNA res
9 FIN MODULO invertir
```

cadena	salida
"casa"	"Ingrese una cadena de caracteres:"
	"Su longitud es: "
	4
	"La cadena invertida es: "

cad	i	res	valor retornado
"casa"			

Pseudocódigo

```
1 MODULO invertir(TEXTO cad) RETORNA TEXTO
2     ENTERO i
3     TEXTO res
4     res <— ""
5     PARA i <— 0 HASTA longitud(cad)-1 PASO 1 HACER
6         res <— caracterenPosicion(cad, i) + res
7     FIN PARA
8     RETORNA res
9 FIN MODULO invertir
```

cadena	salida
"casa"	"Ingrese una cadena de caracteres:"
	"Su longitud es: "
	4
	"La cadena invertida es: "

cad	i	res	valor retornado
"casa"		""	
	0	'c' + ""	

Pseudocódigo

```
1  MODULO invertir(TEXTO cad) RETORNA TEXTO
2      ENTERO i
3      TEXTO res
4      res <— ""
5      PARA i <— 0 HASTA longitud(cad)-1 PASO 1 HACER
6          res <— caracterenPosicion(cad, i) + res
7      FIN PARA
8      RETORNA res
9  FIN MODULO invertir
```

cadena	salida
"casa"	"Ingrese una cadena de caracteres:"
	"Su longitud es: "
	4
	"La cadena invertida es: "

cad	i	res	valor retornado
"casa"		""	
	0	"c"	
	1	"a"+"c"	

Pseudocódigo

```
1  MODULO invertir(TEXTO cad) RETORNA TEXTO
2      ENTERO i
3      TEXTO res
4      res <— ""
5      PARA i <— 0 HASTA longitud(cad)-1 PASO 1 HACER
6          res <— caracterenPosicion(cad, i) + res
7      FIN PARA
8      RETORNA res
9  FIN MODULO invertir
```

cadena	salida
"casa"	"Ingrese una cadena de caracteres:"
	"Su longitud es: "
	4
	"La cadena invertida es: "

cad	i	res	valor retornado
"casa"		""	
	0	"c"	
	1	"ac"	
	2	"s"+"ac"	

Pseudocódigo

```
1  MODULO invertir(TEXTOS cad) RETORNA TEXTOS
2      ENTERO i
3      TEXTOS res
4      res <— ""
5      PARA i <— 0 HASTA longitud(cad)-1 PASO 1 HACER
6          res <— caracterenPosicion(cad, i) + res
7      FIN PARA
8      RETORNA res
9  FIN MODULO invertir
```

cadena	salida
"casa"	"Ingrese una cadena de caracteres:"
	"Su longitud es: "
	4
	"La cadena invertida es: "

cad	i	res	valor retornado
"casa"		""	
	0	"c"	
	1	"ac"	
	2	"sac"	
	3	"a"+"sac"	

Pseudocódigo

```
1 MODULO invertir(TEXT0 cad) RETORNA TEXTO
2     ENTERO i
3     TEXTO res
4     res <— ""
5     PARA i <— 0 HASTA longitud(cad)-1 PASO 1 HACER
6         res <— caracterenPosicion(cad, i) + res
7     FIN PARA
8     RETONAR res
9 FIN MODULO invertir
```

cadena	salida
"casa"	"Ingrese una cadena de caracteres:"
	"Su longitud es: "
	4
	"La cadena invertida es: "
	"asac"

cad	i	res	valor retornado
"casa"		""	"asac"
	0	"c"	
	1	"ac"	
	2	"sac"	
	3	"asac"	
	4		



```
1 public static String invertir(String cad) {  
2     // ...  
3     int i;  
4     String res = "";  
5     for (i = 0; i < cad.length(); i++) {  
6         res = cad.charAt(i) + res;  
7     }  
8     return res;  
9 }
```

Pseudocódigo

```
1  ALGORITMO principal() RETORNA VACIO
2      TEXTO cadena
3      CARACTER car
4      ESCRIBIR("Ingrese una cadena de caracteres: ")
5      LEER(cadena)
6      ESCRIBIR("Contaremos cantidad de ocurrencias de un caracte")
7      ESCRIBIR("Ingrese un caracter: ");
8      LEER(car)
9      ESCRIBIR("La cantidad de ocurrencias es: ");
10     ESCRIBIR(cantOcurrencias(cadena, car));
11  FIN ALGORITMO principal
12
13  MODULO cantOcurrencias(TEXTO cad, CARACTER x) RETORNA ENTERO
14      ENTERO i, cont
15      cont <— 0
16      PARA i <— 0 HASTA longitud(cad)-1 PASO 1 HACER
17          SI (caracterEnPosicion(cad, i)= x) ENTONCES
18              cont <— cont + 1
19          FIN SI
20      FIN PARA
21      RETORNAR cont
22  FIN MODULO
```



```
1      public static int cantOcuurrencias(String cad, char x) {  
2          // ...ejemplo de un recorrido total  
3          int i, cont;  
4          cont = 0;  
5          for (i = 0; i < cad.length(); i++) {  
6              if (cad.charAt(i) == x )  
7                  cont++;  
8          }  
9          return cont;  
10     }
```

Pseudocódigo

```
1  ALGORITMO principal() RETORNA VACIO
2      TEXTO cadena
3      CARACTER car
4      ESCRIBIR("Ingresa una cadena de caracteres: ")
5      LEER(cadena)
6      ESCRIBIR("Ingresa un caracter:")
7      LEER(car)
8      ESCRIBIR("La cantidad de vocales es: ")
9      ESCRIBIR(cantVocales(cadena))
10 FIN ALGORITMO principal
11 MODULO cantVocales(TEXTO cad) RETORNA ENTERO
12     RETORNAR cantOcurrencias(cad, 'a') +
13         cantOcurrencias(cad, 'A') +
14         cantOcurrencias(cad, 'e') +
15         cantOcurrencias(cad, 'E') +
16         cantOcurrencias(cad, 'i') +
17         cantOcurrencias(cad, 'I') +
18         cantOcurrencias(cad, 'o') +
19         cantOcurrencias(cad, 'O') +
20         cantOcurrencias(cad, 'u') +
21         cantOcurrencias(cad, 'U')
```

Cadenas de Caracteres



```
1 public static int cantVocales(String cad) {  
2     // ... ejemplo de un recorrido total  
3     return cantOcuurrencias(cad, 'a') +  
4         cantOcuurrencias(cad, 'A') +  
5         cantOcuurrencias(cad, 'e') +  
6         cantOcuurrencias(cad, 'E') +  
7         cantOcuurrencias(cad, 'i') +  
8         cantOcuurrencias(cad, 'I') +  
9         cantOcuurrencias(cad, 'o') +  
10        cantOcuurrencias(cad, 'O') +  
11        cantOcuurrencias(cad, 'u');  
12        cantOcuurrencias(cad, 'U') +  
13    }
```


Pseudocódigo

```
1  MODULO cantVocales2 (TEXTO cad) RETORNA ENTERO
2      ENTERO i, cont
3      cont ← 0
4      PARA i ← 0 HASTA longitud(cad)-1 PASO 1 HACER
5          SI (caracterEnPosicion(cad, i)= 'a') OR
6              (caracterEnPosicion(cad, i)= 'A') OR
7              (caracterEnPosicion(cad, i)= 'e') OR
8              (caracterEnPosicion(cad, i)= 'E') OR
9              (caracterEnPosicion(cad, i)= 'i') OR
10             (caracterEnPosicion(cad, i)= 'I') OR
11             (caracterEnPosicion(cad, i)= 'O') OR
12             (caracterEnPosicion(cad, i)= 'o') OR
13             (caracterEnPosicion(cad, i)= 'U') OR
14             (caracterEnPosicion(cad, i)= 'u') OR ENTONCES
15                 cont ← cont + 1
16             FIN SI
17     FIN PARA
18     RETORNAR cont
19 FIN MODULO cantVocales2
```

Temario

- 1 Cadenas de Caracteres
- 2 Declaración
- 3 Posiciones en una Cadena de Caracteres
- 4 Concatenación de Cadenas de Caracteres
- 5 Tipos de Recorridos**
- 6 Clases, objetos y métodos
- 7 Métodos asociados a Cadenas de Caracteres
- 8 Comparación de Cadenas de Caracteres

IMPORTANTE: Diferentes tipos de Recorridos.

- **Recorridos Totales o Exhaustivos:** Se recorre la cadena desde el principio al final, o al revés. El recorrido es completo. Utilizamos una estructura Repetitiva PARA.
- **Recorridos parciales:** Se recorre la cadena hasta que se cumpla una condición. En el peor de los casos recorreremos en forma completa si la condición no se cumple. Utilizamos una estructura repetitiva MIENTRAS ó REPETIR HASTA.

Pseudocódigo

```
1 ALGORITMO principal() RETORNA VACIO
2     TEXTO cadena
3     CARACTER car
4     ESCRIBIR("Ingrese una cadena de caracteres: ")
5     LEER(cadena)
6     ESCRIBIR("Ingrese un caracter: ")
7     LEER(car)
8     ESCRIBIR("Verificar existencia del caracter: ")
9     ESCRIBIR(verifExistencia(cadena, car))
10 FIN ALGORITMO principal
```

Pseudocódigo

```
1  MODULO verifExistencia (TEXTO cad, CARACTER x) RETORNA LOGICO
2      ENTERO i
3      LOGICO encontrado
4      encontrado ← FALSO
5      i ← 0
6      MIENTRAS (i < longitud(cad) AND encontrado = FALSO) HACER
7          SI caracterEnPosicion(cad, i) = x ENTONCES
8              encontrado ← VERDADERO
9          FIN SI
10         i ← i + 1
11     FIN MIENTRAS
12     RETORNAR encontrado
13 FIN MODULO verifExistencia
```



```
1 public static boolean verifExistencia(String cad, char x) {  
2     // ... ejemplo de un recorrido total  
3     int i= 0;  
4     boolean encontrado = false;  
5     while (i < cad.length() && (encontrado== false)) {  
6         if (cad.charAt(i) == x )  
7             encontrado = true;  
8         i++;  
9     }  
10    return encontrado;  
11 }
```

Temario

- 1 Cadenas de Caracteres
- 2 Declaración
- 3 Posiciones en una Cadena de Caracteres
- 4 Concatenación de Cadenas de Caracteres
- 5 Tipos de Recorridos
- 6 Clases, objetos y métodos**
- 7 Métodos asociados a Cadenas de Caracteres
- 8 Comparación de Cadenas de Caracteres

Cadenas de Caracteres. Operaciones sobre Strings

- Una **clase** es un tipo usado para producir objetos.
- Un **objeto** es una entidad que almacena datos y puede tomar acciones definidas por **métodos**.
- Un objeto de la clase **String** almacena datos que consisten en una secuencia de caracteres.
- Las operaciones sobre objetos de la clase **String** se ejecutan invocando los metodos correspondientes

Temario

- 1 Cadenas de Caracteres
- 2 Declaración
- 3 Posiciones en una Cadena de Caracteres
- 4 Concatenación de Cadenas de Caracteres
- 5 Tipos de Recorridos
- 6 Clases, objetos y métodos
- 7 Métodos asociados a Cadenas de Caracteres**
- 8 Comparación de Cadenas de Caracteres

Cadenas de Caracteres. Operaciones sobre Strings

Método	Descripción	Ejemplo
<code>longitud(cadena)</code> <code>cadena.length()</code>	Retorna la longitud del objeto String	<pre>String saludo="Hola!"; saludo.length();</pre> <p>salida por pantalla: 5</p>
<code>igual(cad1, cad2)</code> <code>cad1.equals(cad2)</code>	Retorna true si el contenido del objeto <i>cad1</i> y de <i>cad2</i> son iguales, sino retorna false.	<pre>String saludo= TecladoIn.readLine(); if (saludo.equals("Hola")) System.out.println ("Saludo informal");</pre>
<code>igualIgnoMayusc(cad1,cad2)</code> <code>cad1.equalsIgnoreCase(cad2)</code>	Retorna true si el contenido del objeto <i>cad1</i> y de <i>cad1</i> son iguales, considerando una letra en mayúscula y minúscula como iguales, sino retorna false.	<pre>String s1="mary"; s1.equalsIgnoreCase("Mary")</pre> <p>retorna true</p>
<code>aMinuscula(cad1)</code> <code>cad1.toLowerCase()</code>	Retorna un String con los mismos caracteres pero convertidos en minúscula	<pre>String s1="Hola_Mary"; System.out.print(s1.toLowerCase())</pre> <p>retorna "hola mary"</p>

Cadenas de Caracteres. Operaciones sobre Strings

Método	Descripción	Ejemplo
<code>aMayuscula(cad1)</code> <code>cad1.toUpperCase()</code>	Retorna un String con los mismos caracteres pero convertidos en mayúscula	<pre>String s1 = "Hola Mary"; System.out.print(s1.toUpperCase())</pre> retorna "HOLA MARY"
<code>remove(cad)</code> <code>cad.trim()</code>	Retorna un String con los mismos caracteres que el objeto <i>cad</i> , pero remueve los espacios por delante y detrás	<pre>String pausa = " Hmmm "; pausa.trim() — retorna "Hmmm"</pre>
<code>caracterenPosicion(cad,pos)</code> <code>cad.charAt(pos)</code>	Retorna el carácter que en el objeto <i>cad</i> se encuentra en la posición <i>pos</i> . La posición se cuenta 0, 1, 2, etc.	<pre>String saludo = "Hola!"; saludo.charAt(0) — retorna "H"; saludo.charAt(3) — retorna "a";</pre>
<code>subcadena(cad,ini)</code> <code>cad.substring(ini)</code>	Retorna el sub-string del objeto <i>cad</i> desde la posición <i>ini</i> hasta el final. Se cuenta 0,1,etc.	<pre>String prueba = "AbcdefG"; prueba.substring(2); — retorna "cdefG"</pre>

Cadenas de Caracteres. Operaciones sobre Strings

Método	Descripción	Ejemplo
<code>subcadena(cad,ini,fin)</code> <code>cad.substring(ini,fin)</code>	Retorna el sub-string del objeto <i>cad</i> desde la posición <i>ini</i> hasta la posición <i>fin</i> , pero sin incluirla. Se cuenta 0,1,etc	<pre>String prueba = "AbcdefG"; prueba.substring(2,5) — retorna "cde"</pre>
<code>indiceDe(cad1,cad2)</code> <code>cad1.indexOf(cad2)</code>	Retorna la posición de la primer ocurrencia de <i>cad2</i> dentro del objeto <i>cad1</i> . Las posiciones se cuentan 0,1,2,etc. Retorna -1 si <i>cad2</i> no existe dentro del String <i>cad1</i> .	<pre>String saludo = "Hola Mary"; saludo.indexOf("Mary") — retorna 5 saludo.indexOf("Sally") — retorna -1</pre>
<code>indiceDe(cad1,cad2,ini)</code> <code>cad1.indexOf(cad2,ini)</code>	Retorna la posición de la primer ocurrencia de <i>cad2</i> dentro del String <i>cad1</i> , a partir de la posición <i>ini</i> . Las posiciones se cuentan 0,1,2, etc. Retorna -1 si <i>cad2</i> no existe dentro del String <i>cad1</i> .	<pre>String saludo = "Hola Mary"; saludo.indexOf("Mary",1) — retorna 5 saludo.indexOf("Mary",8) — retorna -1</pre>

Cadenas de Caracteres. Operaciones sobre Strings

Método	Descripción	Ejemplo
<code>ultimoIndice(<i>cad1</i>,<i>cad2</i>)</code> <code>cad1.lastIndexOf(<i>cad2</i>)</code>	Retorna la posición de la última ocurrencia de <i>cad2</i> dentro del String <i>cad1</i> . Las posiciones se cuentan 0,1,2,etc. Retorna -1 si <i>cad2</i> no existe dentro del String <i>cad1</i> .	<pre>String s1 = "Mary, Mary, Mary"; s1.lastIndexOf("Mary") —retorna 6</pre>
<code>compareA(<i>cad1</i>, <i>cad2</i>)</code> <code>cad1.compareTo(<i>cad2</i>)</code>	Compara el string <i>cad1</i> con <i>cad2</i> de acuerdo a su orden lexicográfico (que es equivalente al alfabético si las letras están todas en minúsculas o mayúsculas). Si el objeto <i>cad1</i> es menor que <i>cad2</i> devuelve un valor negativo, si son iguales devuelve 0 y si no devuelve un número positivo.	<pre>String entrada = "aventura"; entrada.compareTo("zoo") retorna un valor negativo entrada.compareTo("aventura") retorna 0 entrada.compareTo("abajo") retorna un valor positivo</pre>

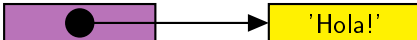
Temario

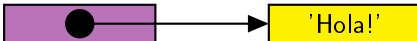
- 1 Cadenas de Caracteres
- 2 Declaración
- 3 Posiciones en una Cadena de Caracteres
- 4 Concatenación de Cadenas de Caracteres
- 5 Tipos de Recorridos
- 6 Clases, objetos y métodos
- 7 Métodos asociados a Cadenas de Caracteres
- 8 Comparación de Cadenas de Caracteres

Cadenas de Caracteres

Comparando el Contenido de dos cadenas

- Para comparar **contenido** de Strings hay que utilizar el método **equals**.

s1 

s2 

`igual(s1,s2)`

`s1.equals(s2)`



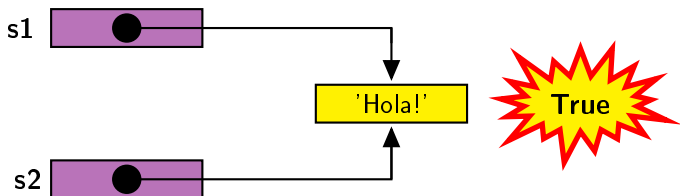
Cadenas de Caracteres

Comparación de Strings

El operador `==` no es apropiado para determinar si dos strings tienen el mismo valor

```
String s1, s2;  
s1 = "Hola!";  
s2 = s1;
```

- ❶ `(s1 == s2)` es true si `s1` y `s2` están en la misma locación de memoria



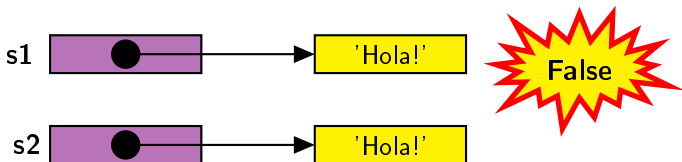
Cadenas de Caracteres

Comparación de Strings

El operador `==` no es apropiado para determinar si dos strings tienen el mismo valor

```
String s1, s2;  
s1 = "Hola!";  
s2 = "Hola!";
```

❶ `(s1 == s2)` es false si `s1` y `s2` no están en la misma locación de memoria



Comparación de Objetos y Caracteres

Ojo con las comparaciones!

```
1 String str1="El lenguaje Java";
2 String str2=str1;
3 System.out.println("String1 ->"+str1);
4 System.out.println("String2 ->"+str2);
5 System.out.println("¿Es el mismo objeto?>"+(str2==str1));
6
7 str2=new String(str1);
8 System.out.println("String1 ->"+str1);
9 System.out.println("String2 ->"+str2);
10 System.out.println("¿Es el mismo objeto?>"+(str2==str1));
11 System.out.println("¿El mismo valor?>"+(str2.equals(str1)));
```

String1 -> El lenguaje Java

String2 -> El lenguaje Java

¿Es el mismo objeto? > true

String1 -> El lenguaje Java

String2 -> El lenguaje Java

¿Es el mismo objeto? > false

¿El mismo valor? > true

Cadenas de Caracteres

```
1 public class StringEqualityDemo
2 {
3     public static void main(String [] args)
4     {
5         String s1, s2;
6         System.out.println("Ingresa dos líneas de texto:");
7         s1 = TecladoLn.readLine();
8         s2 = TecladoLn.readLine();
9         if (s1.equals(s2))
10            System.out.println("Las dos líneas son iguales.");
11        else
12            System.out.println("Las dos líneas NO son iguales.");
13        if (s2.equals(s1))
14            System.out.println("Las dos líneas son iguales.");
15        else
16            System.out.println("Las dos líneas NO son iguales.");
17        if (s1.equalsIgnoreCase(s2))
18            System.out.println("Son iguales si ignoramos mayúsculas.");
19        else
20            System.out.println("No son iguales ignorando mayúsculas.");
21    }}
```

Cadenas de Caracteres

Ejemplo de Comparación de Strings

La salida por pantalla es la siguiente:

Ingrese dos lineas de texto:

Java no es Cafe

Java no es CAFE

s1: Java no es Cafe

s2: Java no es CAFE

Las dos lineas NO son iguales.

Las dos lineas NO son iguales.

Pero las lineas son iguales ignorando el tipo de letra.

Cadenas de Caracteres

Ejemplo de Comparación de Strings

Java usa orden lexicográfico

El **orden lexicográfico** es similar al orden alfabético pero basado en el orden de los caracteres en ASCII/Unicode.

- Los dígitos van antes que todas las letras.
- Las letras mayúsculas van antes que todas las minúsculas.
- El carácter blanco va antes que los dígitos y las letras.

Cadenas de Caracteres

Comparación de Strings

- El método `compareTo` se debe utilizar combinado con el metodo `toUpperCase` o `toLowerCase`.

```
1 String s1 = "HoLa";  
2 String s2 = "hola";  
3 if (s1.toLowerCase().compareTo(s2.toLowerCase()) == 0)  
4     System.out.println("Iguales!");
```

Cadenas de Caracteres

Comparación de Strings. Ejemplo

```
1
2 public class StringDemo {
3 public static void main(String [] args) {
4 String sentencia = "Procesamiento de texto es duro!";
5 int posicion;
6 posicion = sentencia.indexOf("duro");
7 System.out.println(sentencia);
8 System.out.println("012345678901234567890123");
9 System.out.println("La palabra \"duro\" comienza en el
10 indice "+posicion);
11 sentencia = sentencia.substring(0, posicion) + "facil!";
12 System.out.println("El string cambiado es: "+sentencia);
13 }}
```

La salida por pantalla es la siguiente:

```
Procesamiento de texto es duro!
012345678901234567890123
La palabra "duro" comienza en el indice 26
El string cambiado es: Procesamiento de texto es facil!
```



Cátedra Desarrollo de Algoritmos.

Apunte sobre Strings

Facultad de Informática, Universidad del Comahue, 2021.



Sun

Code Convention for the JAVA Programming Language

Sun Microsystems, Inc. 1997