

# Cadenas de Caracteres en Java

## *String*

Elizabeth León Guzmán, Ph.D.

`eleonguz@unal.edu.co`

Jonatan Gómez Perdomo, Ph. D.

`jgomezpe@unal.edu.co`

Arles Rodríguez, Ph.D.

`aerodriguezp@unal.edu.co`

Camilo Cubides, Ph.D. (c)

`eccubidesg@unal.edu.co`

Carlos Andres Sierra, M.Sc.

`casierrav@unal.edu.co`

Research Group on Data Mining – Grupo de Investigación en Minería de Datos – (Midas)

Research Group on Artificial Life – Grupo de Investigación en Vida Artificial – (Alife)

Computer and System Department

Engineering School

Universidad Nacional de Colombia

# Agenda

## 1 Introducción

## 2 Cadenas

- Definición
- Creación de cadenas

## 3 Uso de cadenas

- Lectura
- Comparación
- Contención
- Concatenación
- Longitud
- Extracción de caracteres

## 4 Problemas



# Caracter

## Definición

Un **caracter** es el elemento mínimo de información usado para representar, controlar, transmitir y visualizar datos.

## Definición

Al conjunto de caracteres usados con este fin se le llama **Esquema de codificación**. Los esquemas de codificación en general usan un número de bits o bytes fijos para codificar cada caracter



# Esquemas de Codificación - ASCII

ASCII es el Código Estadounidense Estándar para el Intercambio de Información (*American Standard Code for Information Interchange*)

- En su versión original usa 7 bits, definiendo 128 caracteres.
- En la versión extendida usa 8 bits (esto es 1 byte), definiendo 256 caracteres.
- Es la base de los archivos de texto plano (o sin formato)
- Es el esquema base para la escritura de programas en casi todos los lenguajes de programación (incluido **Java**).



# Esquemas de Codificación - ASCII

Imagen tomada de <https://elcodigoascii.com.ar/>

## Caracteres ASCII de control

00	NULL	(carácter nulo)
01	SOH	(inicio encabezado)
02	STX	(inicio texto)
03	ETX	(fin de texto)
04	EOT	(fin transmisión)
05	ENQ	(consulta)
06	ACK	(reconocimiento)
07	BEL	(timbre)
08	BS	(retroceso)
09	HT	(tab horizontal)
10	LF	(nueva línea)
11	VT	(tab vertical)
12	FF	(nueva página)
13	CR	(retorno de carro)
14	SO	(desplaza afuera)
15	SI	(desplaza adentro)
16	DLE	(esc.vínculo datos)
17	DC1	(control disp. 1)
18	DC2	(control disp. 2)
19	DC3	(control disp. 3)
20	DC4	(control disp. 4)
21	NAK	(conf. negativa)
22	SYN	(inactividad sinc)
23	ETB	(fin bloque trans)
24	CAN	(cancelar)
25	EM	(fin del medio)
26	SUB	(sustitución)
27	ESC	(escape)
28	FS	(sep. archivos)
29	GS	(sep. grupos)
30	RS	(sep. registros)
31	US	(sep. unidades)
127	DEL	(suprimir)

## Caracteres ASCII imprimibles

32	espacio	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	\$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

## ASCII extendido (Página de código 437)

128	Ç	160	á	192	Ł	224	Ó
129	ü	161	í	193	ł	225	ô
130	é	162	ó	194	Ł	226	ò
131	à	163	ú	195	ł	227	ó
132	ä	164	ñ	196	—	228	ô
133	å	165	Ñ	197	+	229	ö
134	ä	166	ª	198	ä	230	µ
135	ç	167	º	199	À	231	þ
136	ê	168	¿	200	É	232	þ
137	ë	169	®	201	Ê	233	ù
138	è	170	¬	202	Ë	234	û
139	ï	171	½	203	Ë	235	ü
140	ì	172	¼	204	Ë	236	ý
141	í	173	⅓	205	=	237	ÿ
142	Ä	174	¼	206	≠	238	—
143	Å	175	»	207	≠	239	·
144	É	176	⌘	208	ø	240	±
145	æ	177	⌘	209	ø	241	±
146	Æ	178	⌘	210	Ê	242	≠
147	ó	179	⌘	211	Ê	243	¼
148	ö	180	⌘	212	Ê	244	¼
149	ò	181	À	213	í	245	\$
150	ú	182	À	214	í	246	+
151	û	183	À	215	í	247	°
152	ÿ	184	©	216	í	248	°
153	Ö	185	Y	217	í	249	°
154	Ü	186	Y	218	í	250	°
155	ø	187	Y	219	í	251	°
156	£	188	Y	220	í	252	°
157	ø	189	¢	221	í	253	°
158	x	190	¥	222	í	254	°
159	f	191	γ	223	í	255	nbsp



# Esquemas de Codificación - Unicode

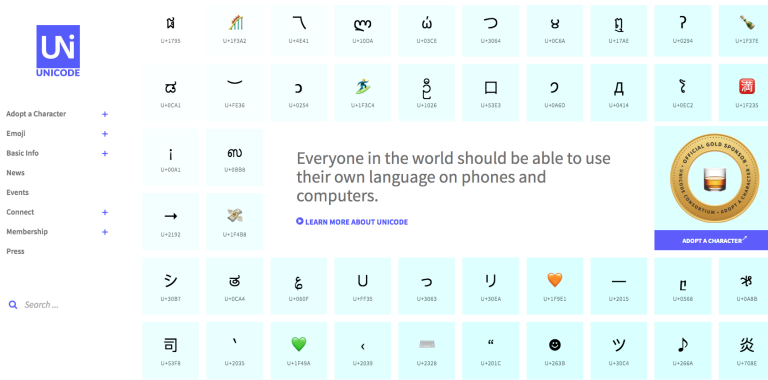
Esquema de codificación cuyo objetivo es dar a cada caracter usado por cada uno de los lenguajes humanos su propio código, es decir, permitir la "internacionalización" de la computación.

- **UTF-8.** Definido por ocho (8) bits (un byte). Toma como base el ASCII, ANSI de Windows y el ISO-8859-1. Muy usado en HTML.
- **UTF-16.** Definido por 16 bits (2 bytes). Usa una representación de longitud variable que permite su optimización en procesos de codificación a texto (usando un subconjunto de ASCII o UTF-8).
- **UTF-32.** Definido por 32 bits (4 bytes). Es el más simple pues usa una representación de longitud fija.



## Esquemas de Codificación - Unicode

Captura de la página <https://home.unicode.org/>



# Usando caracteres en un programa

Dado que **Java** usa ASCII para la escritura de sus programas, se cuenta con un esquema de representación para indicar que se usarán los mismos. El caracter a usar se delimita por el caracter ' o por el caracter " (llamado *escape*) de caracteres tanto de control o Unicode.

- 'A' Se refiere al caracter A
- '3' Se refiere al caracter 3
- '"' Se refiere al caracter "





# Usando caracteres en un programa

Cuando se requieren caracteres especiales, de control o de unicode, se puede utilizar la secuencia de *escape* apropiada.

- `\n`: Una nueva línea
- `\t`: Un tabulado
- `\"`: Una comilla doble
- `'`: Una comilla simple
- `\\`: El caracter `\`



# Agenda

## 1 Introducción

## 2 Cadenas

- Definición
- Creación de cadenas

## 3 Uso de cadenas

- Lectura
- Comparación
- Contención
- Concatenación
- Longitud
- Extracción de caracteres

## 4 Problemas



# Agenda

## 1 Introducción

## 2 Cadenas

- Definición
- Creación de cadenas

## 3 Uso de cadenas

- Lectura
- Comparación
- Contención
- Concatenación
- Longitud
- Extracción de caracteres

## 4 Problemas



# Definición

En programación, una cadena (o String) es una **secuencia ordenada** de **caracteres**, los cuales pueden ser letras, números, signos o incluso símbolos.

## Ejemplo

Aquí varios ejemplos de cadenas o secuencias de caracteres:

- "Perro"
- "La programación es genial!"
- "Mi número es +12 34 56 78 9"
- "12"



# Clase String

- Java nos ofrece la clase incorporada `String`, que encapsula las funcionalidades principales de las cadenas de texto para su creación y manipulación.  
Dicha clase es la representación de un arreglo de caracteres **immutable**, lo que indica que no puede cambiar una vez sea inicializada.
- Así mismo existe una clase llamada `StringBuffer`, que se utiliza para crear cadenas que puedan ser modificadas después de haber sido creadas. Sin embargo su uso implica más uso de memoria y por tanto es **recomendable** usar `String` siempre que sea apropiado.



# Agenda

## 1 Introducción

## 2 Cadenas

- Definición
- Creación de cadenas

## 3 Uso de cadenas

- Lectura
- Comparación
- Contención
- Concatenación
- Longitud
- Extracción de caracteres

## 4 Problemas



# Creación de cadenas I

En Java, las cadenas de texto se definen entre *comillas dobles* " ". El contenido de las cadenas de texto serán caracteres, los cuales están restringidos por el estándar de codificación **Unicode**.

Unicode proporciona la codificación de los textos de forma consistente entre todos los lenguajes, facilitando el intercambio de archivos de texto internacionalmente.

De esta forma, un archivo de texto escrito con el alfabeto Latinoamericano podrá ser codificado correctamente desde el alfabeto Chino.



# Creación de cadenas II

Unicode hace esto posible definiendo un identificador numérico a cada caracter (llamado *Code Point*) que usa como referencia para su correcta interpretación.

Un *Code Point* (o punto de código) es el número con el cual se identifica un carácter en el estándar Unicode, que tiene como formato

U+XXXX

Donde XXXX son de cuatro a seis dígitos en el sistema hexadecimal.

## Ejemplo

Unicode le asigna 65 a la letra 'a' latina minúscula. El punto de código equivalente sería

"\u0041"





# Creación de cadenas IV

En Java está implementada la codificación **Unicode** dentro de la clase `String`, por lo que se podrá hacer uso de la misma fácilmente.

La forma de crear cadenas de texto en Java es la siguiente

- Primero se indica el tipo de objeto que se va a instanciar, en este caso sería

```
String
```

- Luego se define un nombre para el objeto que se está creando

```
String miCadena
```

- Por último se inicializa el objeto con el valor del `String`. Es importante recordar que las cadenas de texto van entre comillas dobles.

```
String miCadena = "¡Hola Mundo!";
```



# Creación de cadenas V

Otra forma de crear cadenas de texto en Java es a través del *constructor* de la clase.

Para esto se debe inicializar el objeto como cualquier objeto en Java, usando la palabra reservada `new`.

```
String miCadena = new String("¡Hola Mundo!");
```



# Agenda

## 1 Introducción

## 2 Cadenas

- Definición
- Creación de cadenas

## 3 Uso de cadenas

- Lectura
- Comparación
- Contención
- Concatenación
- Longitud
- Extracción de caracteres

## 4 Problemas



# Agenda

## 1 Introducción

## 2 Cadenas

- Definición
- Creación de cadenas

## 3 Uso de cadenas

- Lectura
- Comparación
- Contención
- Concatenación
- Longitud
- Extracción de caracteres

## 4 Problemas



# Lectura I

Para leer el valor de una cadena de texto se utiliza la clase [Scanner](#)

## Ejemplo

Lee de consola y la lectura la asigna a un String, que luego se imprime

```
import java.util.Scanner;
public class Lectura {
    public static void main(String[] args) {
        // Declarar el objeto e inicializa
        Scanner sc = new Scanner(System.in);
        // lee una cadena
        String miCadena = sc.nextLine();
        // Imprime la cadena leída
        System.out.println(miCadena);
    }
}
```



# Agenda

## 1 Introducción

## 2 Cadenas

- Definición
- Creación de cadenas

## 3 Uso de cadenas

- Lectura
- **Comparación**
- Contendencia
- Concatenación
- Longitud
- Extracción de caracteres

## 4 Problemas



# Comparación de Cadenas I

Como las cadenas de texto no son de tipo primitivo, determinar si las cadenas `str1` y `str2` son iguales se usa: `str1.equals(str2)`

Para determinar si dos cadenas son iguales ignorando sensibilidad mayúsculas o minúsculas, se usa: `str1.equalsIgnoreCase(str2)`

## Ejemplo

```
public class ComparacionCadenas {  
    public static void main(String args[]) {  
        String a = "hola";  
        String b = "hola";  
        String c = "HOLA";  
        System.out.println(a.equals(b));  
        System.out.println(a.equals(c));  
        System.out.println(a.equalsIgnoreCase(c));  
    }  
}
```



# Comparación de Cadenas II

## Ejemplo

Lee dos cadenas y las compara

```
import java.util.Scanner;
public class LecturaComparacion {
    public static void main(String[] args) {
        // Declarar el objeto e inicializa
        Scanner sc = new Scanner(System.in);
        // lee cadena 1
        String miCadena1 = sc.nextLine();
        // lee cadena 2
        String miCadena2 = sc.nextLine();
        // Imprime true o false
        System.out.println(miCadena1.equals(miCadena2));
    }
}
```





# Agenda

## 1 Introducción

## 2 Cadenas

- Definición
- Creación de cadenas

## 3 Uso de cadenas

- Lectura
- Comparación
- **Contenencia**
- Concatenación
- Longitud
- Extracción de caracteres

## 4 Problemas



# Contenencia de una cadena

Para determinar si una cadena `str1` se encuentra dentro de otra `str2` se utiliza:

```
str1.contains(str2)
```

Este devuelve el valor de *true* o *false*.

## Ejemplo

```
String a = "hola Jorge";  
String b = "hola";  
System.out.println(b.contains(a));
```

La salida del programa es: *true*



# Agenda

## 1 Introducción

## 2 Cadenas

- Definición
- Creación de cadenas

## 3 Uso de cadenas

- Lectura
- Comparación
- Contención
- **Concatenación**
- Longitud
- Extracción de caracteres

## 4 Problemas



# Concatenación

La concatenación de cadenas es la operación mediante la cual se unen múltiples subcadenas de texto para formar una sola.

En Java la operación concatenación está definida por el operador suma +. Como nota se debe recordar que el caracter espacio " " importa, y por tanto si no lo incluimos tendremos palabras unidas tal como se muestra a continuación.

## Ejemplos

- `"Hola" + "Mundo"` produce `"HolaMundo"`
- Si la variable `miCadena` tiene el valor `"¡Hola mundo!"`, la operación `miCadena + "123"` produce `"¡Hola Mundo!123"`
- `"Cadenas de" + " " + "texto"` produce `"Cadenas de texto"`
- `"Hola " + "Mundo"` produce `"Hola Mundo"`



# Concatenación

## Ejemplo

```
String a = "Hola";  
String b = "Mundo";  
System.out.println("¡" + a + " " + b + "!");
```

La salida del programa es: *¡Hola Mundo!*



# Agenda

## 1 Introducción

## 2 Cadenas

- Definición
- Creación de cadenas

## 3 Uso de cadenas

- Lectura
- Comparación
- Contención
- Concatenación
- Longitud
- Extracción de caracteres

## 4 Problemas



# Longitud de Cadenas

Se puede saber exactamente cuántos caracteres tiene una cadena de texto mediante el método `length()` de la clase `String`.

```
str1.length()
```

Este método retorna un entero igual a la cantidad de caracteres que tiene la cadena de texto.

## Ejemplos

- `"Hola Mundo".length() = 10`
- `"1234567".length() = 7`
- `miCadena = "¡Hola Mundo!"`  
`miCadena.length() = 12`
- `"".length() = 0`



# Agenda

## 1 Introducción

## 2 Cadenas

- Definición
- Creación de cadenas

## 3 Uso de cadenas

- Lectura
- Comparación
- Contención
- Concatenación
- Longitud
- Extracción de caracteres

## 4 Problemas





# Extracción de caracteres I

Se puede extraer un único carácter de una cadena de texto mediante el método `charAt()` de la clase `String`. Se debe pasar como parámetro el índice (iniciando en 0) dentro de la cadena, y retornará el caracter en esa posición.

```
cadena.charAt(pos)
```

## Ejemplo

- `"Hola Mundo!".charAt(0) = "H"`
- `"1234567".charAt(6) = "7"`

## Ejemplo

```
String s = "Programar es genial!";  
System.out.println(s.charAt(10));
```

El programa muestra como salida: e



# Extracción de caracteres II

Es posible extraer parte de una cadena utilizando el método `substring(inicio, fin)` de la clase `String`.

El método retornará parte de la cadena que se encuentra entre `inicio` y `fin`.

```
cadena.substring(inicio, fin)
```

## Ejemplo

```
String s = "Programar es genial!";  
System.out.println(s.substring(10, s.length()));
```

El programa muestra como salida: *es genial!*



# Extracción de caracteres III

Es posible dividir una cadena dado una subcadena (por la cual se desea separar) utilizando `split(str1)` de la clase `String`. El método retorna un arreglo de `Strings`.

```
cadena.split(str2)
```

## Ejemplo

Para leer el nombre de una persona, la edad y la estatura separados por espacio, se puede utilizar el siguiente código usando el espacio como la cadena que separa:

```
String s = "Juan 30 1.68";  
String[] line = s.split(" ");  
String nombre = line[0];  
int edad = Integer.parseInt(line[1]);  
double estatura = Double.parseDouble(line[2]);  
System.out.println("nombre: " + nombre);  
System.out.println("edad: " + edad);  
System.out.println("estatura: " + estatura + "m");
```



# Extracción de caracteres IV

## Ejemplo

Es posible leer una cadena de texto y procesarla. Si la cadena está separada por ejemplo por & y el formato del texto es nombre&edad&estatura

```
Scanner sc = new Scanner(System.in);
String s = sc.nextLine();
String[] line = s.split("&");
String nombre = line[0];
int edad = Integer.parseInt(line[1]);
double estatura = Double.parseDouble(line[2]);
System.out.println("nombre: " + nombre);
System.out.println("edad: " + edad);
System.out.println("estatura: " + estatura + "m");
```



# Extracción de caracteres V

Si el usuario digita Ana María&40&1.65, la salida será:

```
nombre: Ana María
```

```
edad: 40
```

```
estatura: 1.65m
```



# Información Adicional

Podrá encontrar documentación adicional de String en: <https://docs.oracle.com/javase/7/docs/api/java/lang/String.html>



# Agenda

## 1 Introducción

## 2 Cadenas

- Definición
- Creación de cadenas

## 3 Uso de cadenas

- Lectura
- Comparación
- Contención
- Concatenación
- Longitud
- Extracción de caracteres

## 4 Problemas



# Problemas varios I

## Problemas

- 1 Elabore un programa que dada una letra cuente cuantas ocurrencias de esta letra hay.
- 2 Elabore un programa que dada una cadena diga si todos los símbolos de la cadena son letras.
- 3 Elabore un programa que dada una cadena cuente las consonantes en dicha cadena.





# Problemas varios II

## Problemas

- ④ Desarrollar un algoritmo que retorne un valor booleano que indique si dos cadenas son iguales, esto es, que tienen la misma longitud con los mismos símbolos en la mismas posiciones.
- ⑤ Desarrollar un algoritmo que permita concatenar dos cadenas (colocar la segunda inmediatamente después de la primera), hay que tener en cuenta que la cadena resultante tendrá un tamaño mayor que cualquiera de la cadenas operandos si son distintas de la cadena vacía (la cadena resultante debe quedar guardada en una variable aparte).
- ⑥ Desarrollar un algoritmo que invierta una cadena de caracteres (la cadena invertida debe quedar guardada en una variable aparte).



# Problemas varios III

## Problemas

- 4 Desarrollar un algoritmo que determine si una cadena de caracteres es palíndromo. Una cadena se dice palíndromo si al invertirla es igual a ella misma. Ejemplos:
- “ala” es palíndromo
  - “anita lava la tina” No es palíndromo, pues al invertirla con espacios no es exactamente igual a la original.
  - “los estudiantes de programación leyeron toda la guía” no es palíndromo.
  - “robas ese sabor” es palíndromo

