

Programación en Java

Inicio

Elementos de un programa informático

- Programa y lenguajes de programación
- El lenguaje Java
- Variables, sentencias y paquetes
- Tipos de datos
- Datos primitivos
- Datos no primitivos
- String
- Operadores y expresiones
- Comentarios en Java
- Constantes y literales
- Entrada y salida de información por consola
- Introducción a la programación orientada a objetos

Estructuras básicas de control

Rekursividad

Programación orientada a objetos POO

- Arrays
- POO avanzada
- Colecciones

Programación avanzada

- Interfaces de usuario
- Entrada/Salida (I/O) de la información
- Persistencia de la información

Atajos de teclado para IntelliJ

About me

String en Java

Un String es un tipo de dato no primitivo que, 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.

A diferencia de muchos objetos vimos que un String se puede crear sin la palabra new.

```
String text = "hola";
```

Creación de String

Se puede crear un String de varias formas.

```
String texto = "Severo Ochoa";

//Utilizando new
String texto2 = new String("Severo Ochoa");

//Utilizando el operador concatenación +
String s2 = text + " 2021";           //s2 contiene "Severo Ochoa 2021"
```

El operador 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, si tenemos dos String b y c:

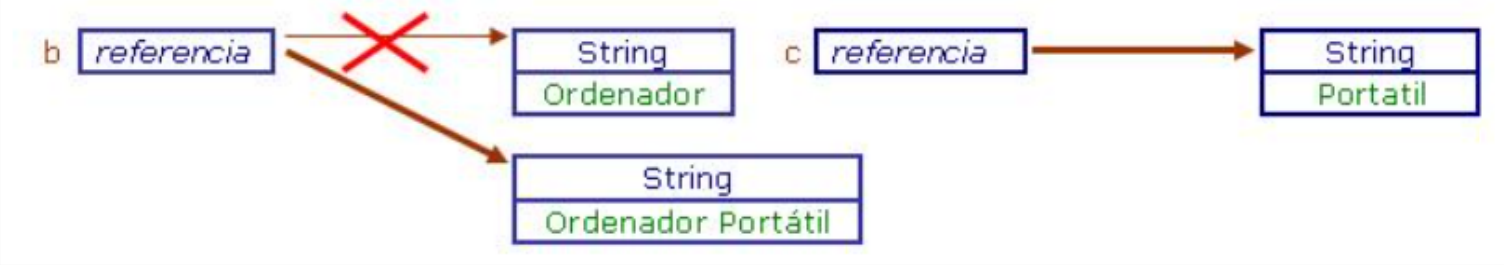
```
String b = "Ordenador";
String c = " Portátil";
```



La operación

```
b = b + c;
```

Creas un nuevo String que se incluye en el String Pool:



Índices

Cada uno de los caracteres que forman un String son del tipo primitivo **char**. Los caracteres de un string están numerados internamente con índices empezando desde el cero:

```
String name = "Ultimate";
```

index	0	1	2	3	4	5	6	7
character	U	l	t	i	m	a	t	e

El primer carácter tiene índice 0 y el último tiene la longitud del string menos 1.

Métodos de la clase String

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.

MÉTODO	DESCRIPCIÓN
length()	Devuelve la longitud de la cadena
indexOf('carácter')	Devuelve la posición de la primera aparición de <i>carácter</i> dentro del String. Devuelve -1 si no lo encuentra.
lastIndexOf('carácter')	Devuelve la posición de la última aparición de <i>carácter</i> dentro del String. Devuelve -1 si no lo encuentra.
charAt(n)	Devuelve el carácter que está en la posición n
substring(n1,n2)	Devuelve la subcadena desde la posición n1 hasta n2 - 1
toUpperCase()	Devuelve la cadena convertida a mayúsculas
toLowerCase()	Devuelve la cadena convertida a minúsculas
equals(otroString)	Compara dos cadenas y devuelve true si son iguales
equalsIgnoreCase(otroString)	Igual que equals pero sin considerar mayúsculas y minúsculas
compareTo(OtroString)	Devuelve 0 si las dos cadenas son iguales. <0 si la primera es alfabéticamente menor que la segunda ó >0 si la primera es alfabéticamente mayor que la segunda.
compareToIgnoreCase(OtroString)	Igual que compareTo pero sin considerar mayúsculas y minúsculas.
valueOf(N)	Convierte el valor N a String. N puede ser de cualquier tipo.

Para acceder a alguno de los métodos siguientes utilizamos la notación "."

```
String texto = "Clase";
```

Tabla de contenidos

Creación de String

El operador concatenación

Índices

Métodos de la clase String

Comparar Strings

char dentro de String

char


Diferencias entre char y String

```
int longitud = texto.length(); //devuelve 5

// index      012345678901
String s1 = "Stuart Reges";
String s2 = "Marty Stepp";

System.out.println(s1.length());           // 12
System.out.println(s1.indexOf("e"));       // 8
System.out.println(s1.substring(7, 10));   // "Reg"

String s3 = s2.substring(1, 7);
System.out.println(s3.toLowerCase());     // "arty s"
```

 Tip

Para más información consulta la documentación oficial de la clase [String](#)

Comparar Strings

Los operadores relacionales como == o < > **NO** se utilizan para comparar Strings, aunque el código compile no es correcto, ya que == compara objetos, y devolvería falso aunque dos strings tuvieran el mismo texto puesto que son objetos diferentes.

Para comparar strings utilizamos el método *equals*.

```
String name = "Patri";

if (name.equals("Patri")) {
    System.out.println("Coincide.");
}
```

La siguiente tabla muestra los métodos que se utilizan para comparar Strings.

Method	Description
equals (str)	whether two strings contain the same characters
equalsIgnoreCase (str)	whether two strings contain the same characters, ignoring upper vs. lower case
startsWith (str)	whether one contains other's characters at start
endsWith (str)	whether one contains other's characters at end
contains (str)	whether the given string is found within this one

char dentro de String

Como se ha comentado, un String está compuesto de caracteres tipo char.

Para acceder a los caracteres dentro de un String usamos el método **charAt**.

Se puede usar la concatenación + para concatenar char con String.

```
String food = "cookie";
char firstLetter = food.charAt(0); // 'c'
System.out.println(firstLetter + " is for " + food);
```

También podemos recorrer el String con un bucle for e imprimir cada uno de los caracteres que lo forman.

```
String major = "CSE";
for (int i = 0; i < major.length(); i++) {
    char c = major.charAt(i);
    System.out.println(c);
}
```

```
OUTPUT
C
S
E
```

char

A todos los valores char se les asigna un número internamente por el ordenador, son los llamados valores ASCII. Por ejemplo:

```
el carácter 'A' es 65 en código ASCII

el carácter 'a' es 97 en código ASCII
```

Mezclar tipos de datos char e int automáticamente cause una conversión en entero. Por ejemplo:

```
'a' + 10 --> devuelve 107.
```

Para convertir un entero en su equivalente a carácter (char) haríamos:

```
(char) ('a' + 2) --> devuelve 'c'.
```

Diferencias entre char y String

- String es un objeto, por tanto, contiene métodos.
- char es un tipo de dato primitivo, no puedes llamar a métodos con él.
- String utiliza comillas dobles.
- char utiliza comillas simples.
- No se puede comparar un String usando operadores relacionales.
- Si se puede comparar un char usando operadores relacionales: 'a' < 'b', 'X' == 'X', ...