


Programació



UT3.4 Tipus de dades compostos.
Tractament d'Strings



Tractament d'Strings

- Com ja sabem, un String és una seqüència (cadena) de caràcters.

H	o	l	a		m	ó	n	!
---	---	---	---	--	---	---	---	---

- A partir d'ara, quan ens referim a un tipus de dada compost, rebrà el nom de "**classe**" (a excepció de l'array).
- També farem servir el nom d'"**objecte**" per a referir-nos al valor assignat a una variable d'un tipus de dada compost.

Tractament d'Strings.

Exemple de classe i objecte

```
String missatge = "Hola món!";
```

- En l'exemple de dalt veiem com s'ha definit una variable de la classe (tipus de dades compost) **String**.
- La variable "missatge" guardarà un objecte (un valor d'un tipus de dades compost) de la classe **String**.

Construcció d'Strings

- Un String es construirà en el moment que li assignem (amb l'operador d'assignació '=') qualsevol cadena de text.
- Tota cadena de text haurà d'estar definida entre una parella de dobles cometes

```
String missatge = "Hola món!";
```


Construcció d'Strings

Altres alternatives per a crear i inicialitzar un String:

```
String message = new String("Welcome to Java");
```

```
char[] charArray = {'G', 'o', 'o', 'd', ' ', 'D', 'a', 'y'};  
String message = new String(charArray);
```


Manipulació d'Strings

- A diferència dels tipus de dades primitius, **qualsevol objecte** emmagatzemat en una variable pertanyent a 1 classe **no es pot manipular directament mitjançant operadors** (*suma, resta...*)
- La manipulació d'Strings es fa sempre **mitjançant la invocació del que anomenem MÈTODES**
- Qualsevol mètode obtindrà informació que es basarà en l'String consultat però **MAI** el modificarà, és a dir, les variables String són **IMMUTABLES**.

Manipulació de strings

Llavors, què fa això?

```
String s = "Java";  
s = "HTML";
```

Manipulació de strings

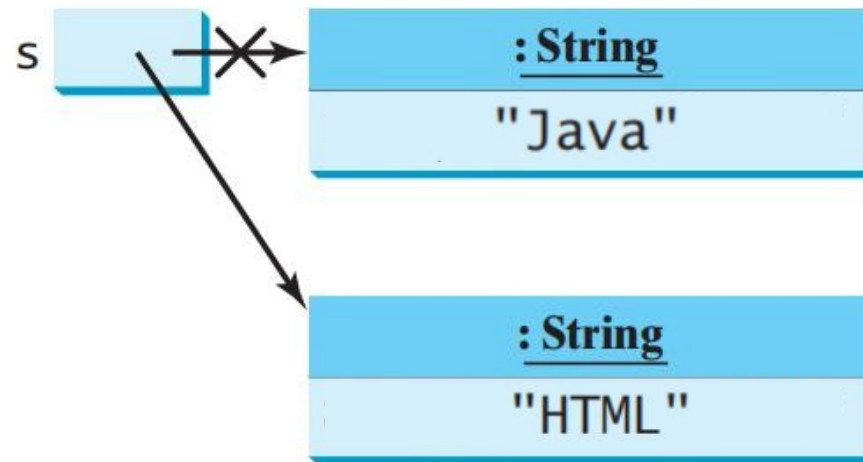
No hi ha que oblidar que Java treballa sempre amb **REFERÈNCIES** a dades quan són dades de tipus compost (Arrays, Strings ...)

després de executar `String s = "Java";`



no pot canviar

després de executar `s = "HTML";`



Aquesta informació ara queda sense referència

Alguns mètodes d'String

<i>Method</i>	<i>Description</i>
length()	Returns the number of characters in this string.
charAt(index)	Returns the character at the specified index from this string.
concat(s1)	Returns a new string that concatenates this string with string s1.
toUpperCase()	Returns a new string with all letters in uppercase.
toLowerCase()	Returns a new string with all letters in lowercase
trim()	Returns a new string with whitespace characters trimmed on both sides.

De quin tipus són els valors retornats per cada mètode?

Obtenció de la longitud d'un String.

Mètode length()

```
public class Precedencia {  
    public static void main (String[] args) {  
        //El text té una mida de 22 caràcters.  
        String text = "Hi havia una vegada...";  
        //ordre: 1) mètode, 2) multiplicació, 3) resta.  
        int dobleMidaMenysUn = 2 * text.length() - 1;  
        System.out.println(dobleMidaMenysUn);  
    }  
}
```

- La invocació del mètode té **SEMPRE** la màxima precedència a l'hora d'avaluar l'expressió.
- La invocació es realitzarà utilitzant el operador **.** (punt)

Mètode charAt d'String

```
String missatge = "Hola món!";
```

missatge conté un objecte de la classe String



0	1	2	3	4	5	6	7	8
H	o	l	a		m	ó	n	!

```
char c = missatge.charAt(3);
```

c

'a'

S'avalua com al caràcter 'a'

Pràctica 3.10

- Crea un programa que donada una cadena de text introduïda per teclat, mostre esta mateixa cadena escrita a l'inrevés.

PISTA: Cal utilizar bucles i el mètode charAt

Pràctica 3.11

- Crea un programa que a partir d'una cadena de caràcters introduïda per teclat, mostre la mateixa cadena però amb tots els seus caràcters en majúscules (és obligatori utilitzar el mètode `charAt`)

PISTA: has utilitzar la taula ASCII

Concatenació de cadenes

- Com ja has vist als temes anteriors, la concatenació es realitza usant l'operador +. Este operador és en realitat una invocació al mètode concat.
- Alternativament es pot invocar a este mètode i tindrem el mateix resultat.

```
String s3 = s1.concat(s2);
```

```
String s3 = s1 + s2;
```


Pregunta 1

Quina eixida s'obté per pantalla?

```
public class ConcatMethod {  
    public static void main(String[] args) {  
        String str1 = "Learn";  
        String str2 = " Java Programming";  
        String str3 = " at tutorialgateway.org";  
  
        String str4 = str1.concat(" JAVA");  
        String str5 = str1.concat(str2);  
        String str6 = str1.concat(str2).concat(str3);  
  
        String str7 = str1 + str2;  
  
        System.out.println(str4);  
        System.out.println(str5);  
        System.out.println(str6);  
        System.out.println(str7);  
    }  
}
```


Pràctica 3.12

- Crea un programa que llija 5 paraules (podran ser introduïdes per teclat en una o en diverses línies).

PISTA: recorda que per a llegir Strings existeixen els mètodes `next()` i `nextLine()` d'`Scanner`. Recordes la diferència entre ells?

Mètodes enfocats a la comparació

<i>Method</i>	<i>Description</i>
<code>equals(s1)</code>	Returns true if this string is equal to string s1.
<code>equalsIgnoreCase(s1)</code>	Returns true if this string is equal to string s1; it is case insensitive.
<code>compareTo(s1)</code>	Returns an integer greater than 0, equal to 0, or less than 0 to indicate whether this string is greater than, equal to, or less than s1.
<code>compareToIgnoreCase(s1)</code>	Same as <code>compareTo</code> except that the comparison is case insensitive.
<code>startsWith(prefix)</code>	Returns true if this string starts with the specified prefix.
<code>endsWith(suffix)</code>	Returns true if this string ends with the specified suffix.
<code>contains(s1)</code>	Returns true if s1 is a substring in this string.

De quin tipus són els valors retornats per cada mètode?

Pregunta 2

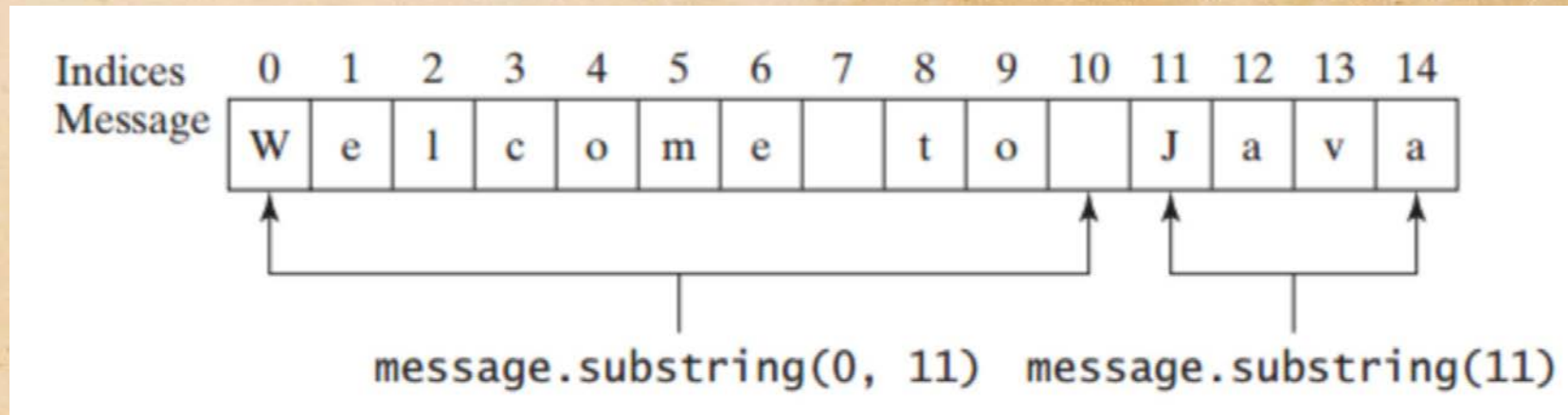
Què estem comparant en cada cas?

```
if (string1 == string2)
    System.out.println("string1 and string2 are the same object");
else
    System.out.println("string1 and string2 are different objects");
```

```
if (string1.equals(string2))
    System.out.println("string1 and string2 have the same contents");
else
    System.out.println("string1 and string2 are not equal");
```


Mètodes enfocats a l'obtenció de subcadenaes d'un String

Method	Description
<code>substring(beginIndex)</code>	Returns this string's substring that begins with the character at the specified <code>beginIndex</code> and extends to the end of the string, as shown in Figure 4.2.
<code>substring(beginIndex, endIndex)</code>	Returns this string's substring that begins at the specified <code>beginIndex</code> and extends to the character at index <code>endIndex - 1</code> , as shown in Figure 4.2. Note that the character at <code>endIndex</code> is not part of the substring.



Mètodes enfocats a trobar un caràcter o subcadena en un String

<i>Method</i>	<i>Description</i>
<code>index(ch)</code>	Returns the index of the first occurrence of <code>ch</code> in the string. Returns -1 if not matched.
<code>indexOf(ch, fromIndex)</code>	Returns the index of the first occurrence of <code>ch</code> after <code>fromIndex</code> in the string. Returns -1 if not matched.
<code>indexOf(s)</code>	Returns the index of the first occurrence of string <code>s</code> in this string. Returns -1 if not matched.
<code>indexOf(s, fromIndex)</code>	Returns the index of the first occurrence of string <code>s</code> in this string after <code>fromIndex</code> . Returns -1 if not matched.
<code>lastIndexOf(ch)</code>	Returns the index of the last occurrence of <code>ch</code> in the string. Returns -1 if not matched.
<code>lastIndexOf(ch, fromIndex)</code>	Returns the index of the last occurrence of <code>ch</code> before <code>fromIndex</code> in this string. Returns -1 if not matched.
<code>lastIndexOf(s)</code>	Returns the index of the last occurrence of string <code>s</code> . Returns -1 if not matched.
<code>lastIndexOf(s, fromIndex)</code>	Returns the index of the last occurrence of string <code>s</code> before <code>fromIndex</code> . Returns -1 if not matched.

Pràctica 3.11

- Realitza un programa que:
 - Demane a l'usuari introduir un text (pot contindre espais).
 - Després pregunte a l'usuari per un caràcter a buscar en la frase anterior.
 - El programa haurà mostrar:
 - Si el caràcter existeix dins de la frase, la posició de la primera ocurrència i l'última en la frase.
 - Si no existeix, es mostrarà un missatge al respecte.

Pràctica 3.12

- Crea un programa que donada una paraula secreta, demane a l'usuari que l'endevine.

Pràctica 3.13

- Crea un programa que donat un array d'Strings, l'ordene fent ús de qualsevol dels mètodes de comparació vistos anteriorment.

Mètode per dividir un String en subcadenaes (substrings)

```
public String[] split(String regex)
```

Splits this string around matches of the given regular expression.

This method works as if by invoking the two-argument `split` method with the given expression and a limit argument of zero. Trailing empty strings are therefore not included in the resulting array.

The string `"boo:and:foo"`, for example, yields the following results with these expressions:

Regex

`:`

`0`

Result

`{ "boo", "and", "foo" }`

`{ "b", "", ":and:f" }`

- El mètode **split** ens retorna un array d'Strings a partir de l'String original i un delimitador.
- El delimitador s'exclou del resultat
- El delimitador s'admet encara que estiga repetit

Conversió d'String a dada numèrica i viceversa

- `String.valueOf(valor)` → retorna la dada "valor" transformant-la en tipus String.
- I al contrari:

<code>Byte.parseByte(text)</code>	retorna →	<code>byte</code>
<code>Integer.parseInt(text)</code>	→	<code>int</code>
<code>Double.parseDouble(text)</code>	→	<code>double</code>
<code>Float.parseFloat(text)</code>	→	<code>float</code>
<code>Long.parseLong(text)</code>	→	<code>long</code>
<code>Short.parseShort(text)</code>	→	<code>short</code>



Si text NO pot ser convertit apareixerà un error (Exception).

→ Retornen la dada **text** de tipus String al valor corresponent en el tipus de dada primitiu (byte, int, double...)

API d'String

La biblioteca de mètodes de string la pots trobar a:

<https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/String.html>

OVERVIEW MODULE PACKAGE **CLASS** USE TREE PREVIEW NEW DEPRECATED INDEX HELP Java SE 17 & JDK 17

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH:

Module java.base
Package java.lang

Class String

java.lang.Object
java.lang.String

All Implemented Interfaces:
Serializable, CharSequence, Comparable<String>, Constable, ConstantDesc

public final class **String**
extends Object
implements Serializable, Comparable<String>, CharSequence, Constable, ConstantDesc

The String class represents character strings. All string literals in Java programs, such as "abc", are implemented as instances of this class.

Strings are constant: their values cannot be changed after they are created. String buffers support mutable strings. Because String objects are immutable they can be shared. For example:

```
String str = "abc";
```

is equivalent to:

```
char data[] = {'a', 'b', 'c'};  
String str = new String(data);
```

Here are some more examples of how strings can be used:

```
System.out.println("abc");  
String cde = "cde";  
System.out.println("abc" + cde);  
String c = "abc".substring(2, 3);  
String d = cde.substring(1, 2);
```

The class String includes methods for examining individual characters of the sequence, for comparing strings, for searching strings, for extracting substrings, and for creating a copy of a string with all characters translated to uppercase or to lowercase. Case mapping is based on the Unicode Standard version specified by the Character class.

The Java language provides special support for the string concatenation operator (+), and for conversion of other objects to strings. For additional information on string concatenation and conversion, see *The Java Language Specification*.

Unless otherwise noted, passing a null argument to a constructor or method in this class will cause a `NullPointerException` to be thrown.