

## Caratteri e Stringhe

Unità didattica 4

Armando Tacchella

Fondamenti di Informatica

## Caratteri

- ◆ In Java i singoli caratteri sono rappresentati utilizzando il tipo di dato **char** (tipo **primitivo** come int, float, boolean, ...)
- ◆ Le costanti letterali di tipo char sono caratteri racchiusi tra apici singoli, per esempio, 'a', 'X', e '5'
- ◆ Un carattere è codificato nella memoria del calcolatore con un codice numerico (ad es. il codice numerico per 'A' è 65)
- ◆ Una schema di codifica per i caratteri molto diffuso è il codice **ASCII** (American Standard Code for Information Interchange), basato sull'alfabeto anglosassone
- ◆ Java utilizza un'estensione conservativa del codice ASCII chiamata Unicode (Unicode Worldwide Character Standard), pensata per gestire alfabeti diversi da quello anglosassone

## Tabella ASCII (standard)

	0	1	2	3	4	5	6	7	8	9
0	nul	soh	stx	etx	ext	enq	ack	bel	bs	ht
10	lf	vt	ff	cr	so	si	del	dc2	dc3	
20	cd	nak	syn	stb	can	em	sub	esc	fs	gs
30	rs	us	sp	!	"	#	\$	%	&	'
40	(	)	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[	\	]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{	}		~	del		

Il carattere 'O' ha codice 79 (riga 70 + colonna 9 = 79).

## Gestione di caratteri

```
char ch1;  
char ch2 = 'X';
```

Dichiarazione ed inizializzazione

```
uscitaDati.print("Codice ASCII di X: ");  
uscitaDati.println((int) 'X');
```

Conversione tra int e char.

```
uscitaDati.print("Codice ASCII 88: ");  
uscitaDati.println((char) 88);
```

```
'A' < 'c'
```

true perchè il codice ASCII di 'A' è 65 e il codice di 'c' è 99.

## Stringhe

- ◆ Una **stringa** è una sequenza di caratteri alfanumerici trattata come un **singolo valore**
- ◆ La classe predefinita **String** è utilizzata per rappresentare le stringhe in Java.
- ◆ Le **costanti letterali** della classe String sono sequenze di caratteri racchiuse tra doppi apici, ad es. "Ciao"
- ◆ Abbiamo utilizzato costanti letterali della classe String per visualizzare testi con **OutputBox**:

```
OutputBox uscitaDati = new OutputBox(...);  
uscitaDati.print("Un messaggio");
```

## Le stringhe sono oggetti in Java

- ◆ Dato che String è una classe dobbiamo creare **oggetti String** per rappresentare testi in Java
- ◆ Come tutti gli oggetti, sono necessarie una dichiarazione e una creazione per usare uno String
- ◆ Java consente di trattare la **creazione** di un oggetto String in maniera **analoga** ai tipi primitivi

```
String nome1;  
nome1 = new String( "Latte" );
```

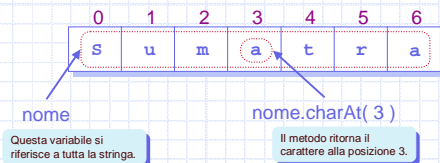
Queste istruzioni sono equivalenti.

```
String nome1;  
nome1 = "Latte";
```

## Accesso ai caratteri di una stringa

- ◆ I singoli caratteri in un oggetto String sono accessibili con il metodo **charAt**

```
String nome = "Sumatra";  
char inNome = nome.charAt(3);
```



## Lunghezza di una stringa

- ◆ Il numero di caratteri in una stringa si ottiene con il metodo **length**

```
String nome = "Sumatra",  
      str1 = "one",  
      str2 = "",  
      str3;
```

```
nome.length() → 7  
str1.length() → 3  
str2.length() → 0  
str3.length() → Errore!
```

str3 non  
corrisponde ad  
alcun oggetto.

## Concatenazione di stringhe

- ◆ Mediante l'operatore **+** applicato a due oggetti String è possibile ottenerne un terzo che risulta dalla **concatenazione delle sequenze** contenute nei primi due
- ◆ Esempio di concatenazione:

```
String prima = "Torno ";  
String seconda = "subito!";  
String terza = prima + seconda;
```

- ◆ Possiamo utilizzare il **+** per concatenare diversi messaggi in una sola istruzione print (println) di un OutputBox

```
uscitaDati.print("Ciao, " + nome + " come stai?");
```

## Esempio: conteggio di vocali

```
String nome = richiestaDati.getString("Nome?");  
int numeroVocali = 0;  
for (int i = 0; i < nome.length(); i++) {  
    char letter = nome.charAt(i);  
    if (letter == 'a' || letter == 'A' ||  
        letter == 'e' || letter == 'E' ||  
        letter == 'i' || letter == 'I' ||  
        letter == 'o' || letter == 'O' ||  
        letter == 'u' || letter == 'U') {  
        numeroVocali++;  
    }  
}  
uscitaDati.print("il tuo nome ha " +  
    numeroVocali + " vocali");
```

Queste istruzioni  
contano il numero di  
vocali nella stringa  
in ingresso (nome).

## Esempio: conteggio di parole (1)

```
String frase = richiestaDati.getString("Dammi una frase:");  
int nc = frase.length();  
int indice = 0, numeroParole = 0;  
  
while (indice < nc) {  
    // ignora gli spazi bianchi  
    while (frase.charAt(indice) == ' ') {  
        indice++;  
    }  
    // trova la fine della parola  
    while (frase.charAt(indice) != ' ') {  
        indice++;  
    }  
    // un'altra parola trovata, incremento del contatore  
    numeroParole++;  
}
```

**Problema:**  
Il ciclo più interno potrebbe  
assegnare ad indice un  
valore che eccede il  
massimo consentito (nc)

## Esempio: conteggio di parole (2)

```
String frase = richiestaDati.getString("Dammi una frase:");  
int nc = frase.length();  
int indice = 0, numeroParole = 0;  
  
while (indice < nc) {  
    // ignora gli spazi bianchi  
    while ((indice < nc) && (frase.charAt(indice) == ' ')) {  
        indice++;  
    }  
    // trova la fine della parola  
    while ((indice < nc) && (frase.charAt(indice) != ' ')) {  
        indice++;  
    }  
    // un'altra parola trovata, incremento del contatore  
    numeroParole++;  
}
```

**Problema:**  
wordCount è maggiore di  
uno rispetto al conteggio  
corretto se la frase finisce  
con uno o più spazi.

## Uguaglianza fra stringhe

- ◆ Per confrontare il contenuto di due oggetti String x e y
  - **x.equals(y)** confronta il contenuto per uguaglianza stretta, ossia x.equals(y) è true soltanto se il contenuto di x e y è identico
  - **x.equalsIgnoreCase(y)** confronta il contenuto ignorando la differenza tra maiuscole e minuscole
  - **x.compareTo(y)** restituisce 0 se x.equals(y) è true, un numero minore di 0 se x precede y, e un numero maggiore di 0 altrimenti

```
String x = "Alfa", y = "alfa", z = "beta";
boolean r;
int c;
r = x.equals(y);           // r == false
r = x.equalsIgnoreCase(y); // r == true
c = y.compareTo(z);        // c < 0
c = x.compareTo(y);        // c < 0
```

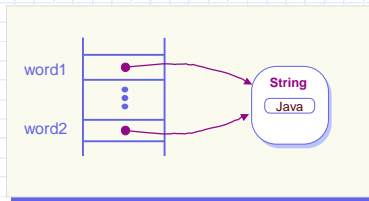
## Esempio: confronto tra oggetti String

```
int javaNum = 0;
boolean ripeti = true;
String parola;

while ( ripeti ) {
    parola = inData.getString("Prossima parola:");
    if ( parola.equals("STOP") ) {
        ripeti = false;
    } else if ( parola.equalsIgnoreCase("Java") ) {
        javaNum++;
    }
}
```

Continua a leggere le parole e conta quante volte la parola Java appare nel testo in ingresso.

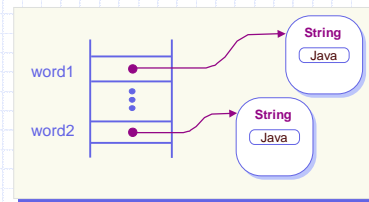
## Uguaglianza (==) vs. equals (1)



```
word1 == word2 → true
word1.equals( word2 ) → true
```

word1 e word2 puntano allo stesso oggetto.

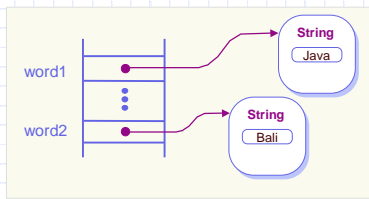
## Uguaglianza (==) vs. equals (2)



```
word1 == word2 → false
word1.equals( word2 ) → true
```

word1 e word2 puntano ad oggetti diversi con uguale contenuto.

## Uguaglianza (==) vs. equals (3)



```
word1 == word2 → false
word1.equals( word2 ) → false
```

word1 e word2 puntano ad oggetti diversi con diversi contenuti.

## Altri metodi della classe String

Metodo	Significato
substring	<b>x.substring(n,m)</b> restituisce una stringa corrispondente alla porzione di x dalla posizione n alla posizione m String s = "Viva Java!"; String t = substring(1,3); // contiene "iva"
trim	<b>x.trim()</b> restituisce una stringa con lo stesso contenuto di x privato di eventuali spazi all'inizio e alla fine
startsWith	<b>x.startsWith(y)</b> restituisce true se x comincia con y String s = "Viva Java!"; t = "Viva"; boolean r = s.startsWith(t);
endsWith	<b>x.endsWith(y)</b> restituisce true se x termina con y String s = "Viva Java!"; t = "Java!"; boolean r = s.endsWith(t);
indexOf	<b>x.indexOf(c)</b> restituisce la posizione di c in x e <b>x.indexOf(c,n)</b> restituisce la posizione di c in x cominciando la ricerca dalla posizione n; entrambe restituiscono -1 se c non è presente in x (c di tipo char)

## Classe StringBuffer

- ◆ In Java non è possibile modificare dinamicamente il contenuto di un oggetto String
- ◆ I metodi come trim e substring non modificano il contenuto iniziale dell'oggetto a cui sono applicati, e restituiscono un nuovo oggetto come risultato
- ◆ Java adotta questa restrizione per poter allocare efficientemente gli oggetti String in memoria
- ◆ Se si desidera modificare dinamicamente un testo è necessario usare la classe **StringBuffer**

```
StringBuffer x = new StringBuffer();
StringBuffer x =
    new StringBuffer("Contenuto iniziale");
```

## Metodi di StringBuffer

Metodo	Significato
charAt	<b>x.charAt(n)</b> restituisce il carattere in posizione n (analogo alla classe String)
length	<b>x.length()</b> restituisce il numero di caratteri in x (analogo alla classe String)
setCharAt	<b>x.setCharAt(n, c)</b> imposta il carattere in posizione n al valore c (c è di tipo char)
deleteCharAt	<b>x.deleteCharAt(n,)</b> cancella il carattere in posizione n
append	<b>x.append(y)</b> concatena a x il contenuto di y
reverse	<b>x.reverse()</b> rovescia il contenuto di x
insert	<b>x.insert(n,y)</b> inserisce y all'interno di x a partire dalla posizione n
delete	<b>x.delete(n,m)</b> cancella dalla posizione n alla posizione m
toString	<b>y = x.toString()</b> restituisce il contenuto di x sotto forma di String (y è un oggetto String)

## Utilizzo di StringBuffer (1)

Rimpiazza tutte le vocali dell'oggetto frase con 'X'

```
String frase = richiestaDati.getString("Frase:");
StringBuffer temp = new StringBuffer(frase);

for (int i = 0; i < temp.length(); i++) {
    char c = temp.charAt(i);

    if ( c == 'a' || c == 'A' || c == 'e' || c == 'E' ||
        c == 'i' || c == 'I' || c == 'o' || c == 'O' ||
        c == 'u' || c == 'U' ) {
        temp.setCharAt(i, 'X');
    }
}

uscitaDati.print(temp);
```

## Utilizzo di StringBuffer (2)

Costruisce una frase con le sole parole di lunghezza pari

```
String parola;
StringBuffer temp = new StringBuffer("");
boolean ripeti = true;

while ( ripeti ) {
    parola = inDati.getString("Prossima parola:");
    if ( parola.equals("STOP") ) {
        ripeti = false;
    } else if ( parola.length() % 2 == 0 ) {
        temp.append(parola + " ");
    }
}
```

Aggiunge la parola e uno spazio a tempStringBuffer.

## StringBuffer vs. String

- ◆ **StringBuffer** consente di modificare il contenuto con i metodi **setCharAt**, **insert**, ecc.
- ◆ **String** non consente di modificare il contenuto: i metodi trim, +, substring, ecc. creano un nuovo oggetto String
- ◆ Non è possibile utilizzare uno **StringBuffer** per **input di testi**
- ◆ Gli oggetti String sono utilizzati per input di testi
- ◆ **StringBuffer** è la **sceita ideale** per rappresentare un testo quando è necessario **manipolarne il contenuto**
- ◆ **String** è la **sceita obbligata** per le **costanti letterali** e per i testi che non devono essere manipolati frequentemente
- ◆ **StringBuffer** è **flessibile** ma poco efficiente in termini di occupazione di memoria
- ◆ **String** è **poco flessibile** ma efficiente in termini di occupazione di memoria

## Conversione da String a tipi primitivi (1)

- ◆ La classe String mette a disposizione il metodo **valueOf** che consente di **convertire** un tipo primitivo in un testo
- ◆ Esempi (tipo primitivo ⇒ String)
 

```
String x = String.valueOf(40.5);
String y = String.valueOf('c');
String z = String.valueOf(true);
```
- ◆ Per convertire un oggetto String che sappiamo contenere un **valore significativo** corrispondente a un tipo primitivo (diverso da char) si utilizza una **classe wrapper**
- ◆ Esempi (String ⇒ tipo primitivo)
 

```
double d = Double.parseDouble(x);
boolean b = Boolean.getBoolean(z);
char c = y.charAt(0);
```

### Conversione da String a tipi primitivi (2)

- ◆ Le **classi wrapper** sono predefinite e corrispondono ai tipi primitivi: **Integer**, **Long**, **Float**, **Double** e **Boolean**
- ◆ Si possono **creare istanze** delle classi wrapper, ad es.  

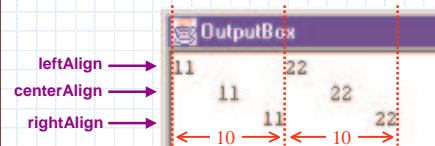
```
String x = "37";
Integer iObj = new Integer(x);
```
- ◆ Ogni classe wrapper **T** corrisponde ad un tipo primitivo **t** ed ha un metodo **parseT** o **getT** che consente di convertire un oggetto String in una variabile di **t**, ad es.  

```
int i = Integer.parseInt(x);
boolean b = Boolean.getBoolean(z);
```
- ◆ In **alternativa** si possono usare gli oggetti wrapper, ad es.  

```
int i = iObj.intValue();
float f = iObj.floatValue();
```

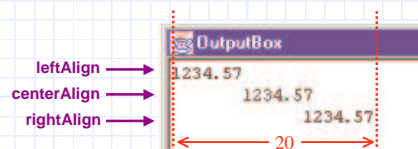
### Impaginazione con Format e OutputBox (1)

```
int x = 11, y = 22;
outputBox.println( Format.leftAlign ( 10, x ) +
                  Format.leftAlign ( 10, y ) );
outputBox.println( Format.centerAlign( 10, x ) +
                  Format.centerAlign( 10, y ) );
outputBox.println( Format.rightAlign ( 10, x ) +
                  Format.rightAlign ( 10, y ) );
```



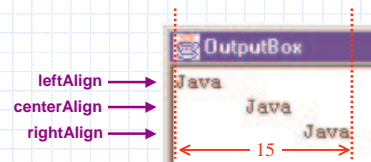
### Impaginazione con Format e OutputBox (2)

```
double w = 1234.5678;
outputBox.println( Format.leftAlign ( 20, 2, w ) );
outputBox.println( Format.centerAlign( 20, 2, w ) );
outputBox.println( Format.rightAlign ( 20, 2, w ) );
```



### Impaginazione con Format e OutputBox (3)

```
String s = "Java";
outputBox.println( Format.leftAlign ( 15, s ) );
outputBox.println( Format.centerAlign( 15, s ) );
outputBox.println( Format.rightAlign ( 15, s ) );
```



### Classe ResponseBox

- ◆ La classe **ResponseBox** può essere utilizzata per sollecitare una particolare risposta (affermativa o negativa) dall'utente

```
MainWindow finestra = new MainWindow( );
ResponseBox yesNoBox
    = new ResponseBox( finestra );
yesNoBox.prompt( "Do you love Java?" );
```



### Gestire la selezione con ResponseBox

```
int selection = yesNoBox.prompt("Premi un bottone");

switch (selection) {
    case ResponseBox.YES:
        messageBox.show("Hai premuto Yes");
        break;
    case ResponseBox.NO:
        messageBox.show("Hai premuto No");
        break;
}
```

## Esempio di utilizzo di ResponseBox

```
choice = yesNoBox.prompt
    ("Vuoi iniziare ad eseguire i calcoli?");

while (choice == ResponseBox.YES) {

    //qui vengono eseguiti i calcoli

    choice = yesNoBox.prompt
        ("Altri calcoli? ");
}
```

## Classe ListBox

- ◆ La classe ListBox fornisce una lista di cui l'utente può selezionare una sola voce
- ◆ ListBox evita che l'utente possa compiere scelte non valide ed elimina il controllo a posteriori su eventuali errori
- ◆ Creazione di un oggetto ListBox

```
MainWindow finestra = new MainWindow( );
ListBox colorList =
    new ListBox(finestra, "Select Color" );
```

Possiamo creare colorList come  
... = new ListBox(finestra);  
se non necessitiamo di un titolo.

## Aggiunta elementi ad un ListBox

- ◆ Il metodo addItem consente di aggiungere voci

```
colorList.addItem("Magenta");
colorList.addItem("Cyan");
colorList.addItem("Red");
colorList.addItem("Blue");
colorList.addItem("Green");
```

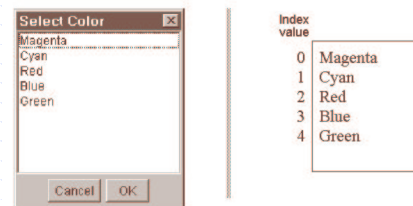
Questi elementi verranno  
aggiunti alla lista  
nell'ordine in cui addItem  
è invocato.

- ◆ Invocando il metodo

```
selection = colorList.getSelectedIndex();
```

viene visualizzato il ListBox per consentire all'utente la scelta della voce desiderata

## Relazione tra posizione e voce in ListBox



Il metodo `getSelectedIndex` ritorna l'indice della voce prescelta

## Metodi di ListBox

Metodo	Argomenti	Descrizione
ListBox (costruttore)	MainWindow	Crea un oggetto <b>ListBox</b>
addItem	String	Aggiunge una voce alla lista; le voci sono aggiunte a partire dall'alto; la voce più in alto ha indice pari a 0
getSelectedIndex	(nessuno)	Restituisce l'indice della voce selezionata
Costante		Descrizione
NO_SELECTION		Questo valore viene restituito da <code>getSelectedIndex</code> quando l'utente preme OK senza aver scelto nulla
CANCEL		<code>x.append(y)</code> Questo valore viene restituito da <code>getSelectedIndex</code> quando l'utente preme Cancel

## Gestire la scelta dell'utente con un ListBox

```
int selection =
    colorList.getSelectedIndex();
String output;

if (selection == ListBox.NO_SELECTION) {
    output = "Non hai scelto nulla!";
} else if (selection == ListBox.CANCEL) {
    output = "Hai cancellato l'operazione";
} else if (selection == 0) {
    output = "Hai scelto magenta";
} else if (selection == 1) {
    ...
} else {
    output = "Hai scelto verde";
}
```

L'uso delle costanti  
NO\_SELECTION e  
CANCEL rende il codice  
più leggibile. Rende  
anche più facile  
modificare il codice.

### Esercitazione – Caratteri e Stringhe

- ◆ Modificare il programma per il conteggio delle parole in modo che esegua un conteggio corretto anche in presenza di spazi al termine della frase; cercare una soluzione con e senza il metodo trim()
- ◆ Modificare il programma per il conteggio delle parole in modo che esegua anche un conteggio dei caratteri alfabetici (A-Z e a-z)
- ◆ Scrivere un programma che, dato un testo, ricerchi all'interno dello stesso il numero di occorrenze di un dato carattere
- ◆ Scrivere lo stesso programma utilizzando il metodo indexOf della classe String; suggerimento: servono entrambe le versioni di indexOf, come nell'esempio seguente

```
char c = 'a';  
String x = "Java";  
int prima = x.indexOf(c); // prima == 1  
int seconda = x.indexOf(c, prima + 1); // seconda == 3
```

### Esercitazione – Stringhe e classi

- ◆ Scrivere una classe istanziabile per eseguire calcoli tra numeri binari rappresentati come oggetti String
- ◆ Le operazioni sono somma, sottrazione, divisione e prodotto
- ◆ Il calcolatore ha la possibilità di memorizzare l'ultima operazione
- ◆ Suggerimento (1): convertire i numeri binari in interi decimali, eseguire l'operazione e riconvertire il risultato
- ◆ Suggerimento (2): concentrarsi, in prima istanza, sui due metodi long daBinario(String x) e String aBinario(long x)
- ◆ Esempio di utilizzo:

```
CalcolatoreBinario calc =  
    new CalcolatoreBinario();  
String a = "1010"; // a corrisponde a 10  
String b = "101"; // b corrisponde a 5  
String c =  
    calc.somma(a,b); // "1111" corrisponde a 15
```

### Unità didattica 4 – Argomenti svolti

- ◆ Manipolazione di testi utilizzando
  - il tipo primitivo **char** per rappresentare singoli caratteri
  - la classe predefinita **String** per rappresentare sequenze di caratteri
  - la classe predefinita **StringBuffer** per consentire la manipolazione dinamica di sequenze di caratteri
- ◆ Conversione di oggetti String in variabili di tipo predefinito con l'utilizzo delle classi wrapper
- ◆ Impaginazione di testi utilizzando la classe **Format**
- ◆ Utilizzo di **ResponseBox** e **ListBox** del package **javabook**