String e StringBuilder

- le classi String e StringBuilder del package java.lang
- La classe String ha lo scopo di rappresentare stringhe (sequenze) di caratteri che non devono essere modificate dopo essere state costruite (oggetti immutabili)
- La classe StringBuilder ha lo scopo di rappresentare stringhe (sequenze) di caratteri che possono essere modificate dopo essere state costruite

Definizione di variabili

- tipo nome; oppure
- tipo nome1,..., nomeN ;
- String nome;
- StringBuilder risultato;
- Dopo la definizione esiste solo il riferimento, non un oggetto di tipo nome, null, String !

null

- Il valore speciale null è il valore iniziale di default per qualunque variabile di tipo strutturato.
- indica che il riferimento è nullo e non c'e' nessun oggetto riferito
- nome non è un oggetto di tipo String è solo un riferimento utilizzabile per accedere ad un oggetto String

Operatore new

- L'operatore new NomeClasse crea un nuovo oggetto con le proprietà definite in NomeClasse (istanza dellaclasse) e ritorna il riferimento ad esso
- L'operatore new dà luogo all'invocazione di un metodo costruttore passandogli gli argomenti necessari
- Il costruttore invocato deve essere di una classe uguale o "compatibile" con la definizione della variabile
- Ogni classe può avere più costruttori che si differenziano per la lista degli argomenti

Costruttori

- La scelta del costruttore da invocare avviene tramite gli argomenti attuali che vengono passati
- New di un oggetto String

```
String saluto;
saluto = new String("Ciao ciao");
```

- L'operatore new può essere usato al momento della definizione

```
String saluto = new String("Ciao ciao");
```

- Solo per la classe String , in quanto di uso molto comune, Java offre la forma compatta

```
String s = "Ciao ciao";
```

Una particolarità di String

- Usare esplicitamente new oppure la forma abbreviata per inizializzare un oggetto String non è esattamente la stessa cosa
- Se si usa esplicitamente new, la Java Virtual Machine crea oggetti distinti anche se di contenuto uguale
- Se non si usa esplicitamente new, la Java Virtual Machine evita di creare oggetti distinti ma dal contenuto uguale

I più importanti metodi di cui sono dotati gli oggetti di tipo String.

Tipo restituito	**Metodi e parametri**	**Descrizione**
int	charAt(int i)	Restituisce il carattere alla posizione i.
boolean	endsWith(String s)	Restituisce true se l'oggetto di invocazione termina con la sottostringa s.
boolean	equals(String s)	Restituisce true quando l'oggetto di invocazione e s rappresentano la medesima sequenza.
int	indexOf(char c)	Restituisce la prima posizione del carattere c, oppure -1 nel caso tale carattere non faccia parte della stringa.
int	indexOf(char c, int i)	Come il precedente, con la differenza che la ricerca del carattere c comincia dalla posizione i.
int	indexOf(String s)	Restituisce la prima posizione della sottostringa s, oppure -1 nel caso tale sottostringa non compaia nell'oggetto di invocazione.
int	indexOf(String s, int i)	Come il precedente, con la differenza che la ricerca della sottostringa s prende piede dalla posizione i.
int	length()	Restituisce la lunghezza della stringa.
String	replace(char c1, char c2)	Restituisce una nuova stringa, ottenuta dall'oggetto di invocazione sostituendo il carattere c2 ad ogni occorrenza del carattere c1.
boolean	starsWith(String s)	Restituisce true se l'oggetto di invocazione inizia con la sottostringa s.
String	toLowerCase()	Restituisce una nuova stringa, ottenuta traslando verso il minuscolo ogni carattere dell'oggetto di invocazione.

Tipo restituito	Metodi e parametri	Descrizione
String	toUpperCase()	Restituisce una nuova stringa, ottenuta traslando verso il maiuscolo ogni carattere dell'oggetto di invocazione.
String	trim()	Restituisce una nuova stringa, ottenuta dall'oggetto di invocazione eliminando gli spazi che precedono il primo carattere significativo e quelli che seguono l'ultimo. In pratica, " ciao ".trim() restituisce "ciao".

- Le stringhe in Java sono oggetti.
- La particolarità di questa classe è quella di essere l'unica classe che è possibile istanziare come se fosse un tipo di dato primitivo.

```
int compareTo(String other)
```

Esegue una comparazione lessicale. Ritorna un intero:

- < 0 se la stringa corrente è minore della stringa other
- = 0 se le due stringhe sono identiche
- > 0 se la stringa corrente è maggiore di other

```
int indexOf(int ch)
```

Restituisce l'indice del carattere specificato

```
int lastIndexOf(int ch)
```

E' come indexOf() ma viene restituito l'indice dell'ultima occorrenza trovata

```
int length()
```

Restituisce il numero di caratteri di cui è costituita la stringa corrente

```
String replace(char oldChar, char newChar)
```

Restituisce una nuova stringa, dove tutte le occorrenze di oldChar sono rimpiazzate con newChar

```
String substring(int startIndex)
```

Restituisce una sottostringa della stringa corrente, composta dai caratteri che partono dall'indice startIndex alla fine

```
String substring(int startIndex, int number)
```

Restituisce una sottostringa della stringa corrente, composta dal numero number di caratteri che partono dall'indice startIndex

```
String toLowerCase()
```

Restituisce una nuova stringa equivalente a quella corrente ma con tutti i caratteri minuscoli

```
String toUpperCase()
```

Restituisce una nuova stringa equivalente a quella corrente ma con tutti i caratteri maiuscoli

Package java.util

Il package java.util contiene una serie di classi utili come il framework "Collections" per gestire collezioni eterogenee di ogni tipo, il modello a eventi, classi per la gestione facilitata delle date e degli orari, classi per la gestione dell'internazionalizzazione e tante altre utilità come un separatore di stringhe (StringTokenizer), un generatore di numeri casuali ecc.

StringTokenizer

La classe StringTokenizer permette l'estrazione di sottostringhe

- StringTokenizer (String str, String delim)
- Costruisce un estrattore di token per la stringa str
- delim e' il delimitatore ricercato tra i token estratti
- La classe StringTokenizer mette quindi a disposizione metodi per la gestione dei token
    - public boolean hasMoreTokens()
    - public String nextToken()

Esempio

// il numero di token e' noto: nome, eta', reddito String str; StringTokenizer st = new StringTokenizer(str," "); // Anche: StringTokenizer st = // new StringTokenizer (str); while (st.hasMoreTokens()){ String token = st.nextToken(); ... Integer.parseInt(token) ... // int eta = Integer.parseInt (st.nextToken ()); // double reddito = Double.parseDouble // (st.nextToken ()); }

Classe StringTokenizer

Spesso risulta necessario manipolare dei token di testo. Una semplice classe che permette di separare i contenuti di una stringa in più parti, chiamate token, è la classe **StringTokenizer**.

Questa classe si utilizza solitamente per estrarre le parole di una stringa. L'utilizzo di base è estremamente semplice, occorrono: una stringa da "navigare", cioè da cui estrarre i token un delimitatore, che serve per identificare i token Un token è, quindi, la sequenza massima di caratteri consecutivi che non sono delimitatori.

CREARE OGGETTO STRINGTOKENIZER

Occorre creare in prima istanza l'oggetto StringTokenizer, usando il costruttore dell'omonima classe.

- Il costruttore può accettare da 1 a 3 parametri: la stringa da cui estrarre i token il delimitatore, che può essere: esplicito [st2 – st3] di default " \t\n\r\f" (notare che il primo delimitatore e uno spazio) [st1] un booleano che, se settato a true, considera token anche gli stessi delimitatori

```
StringTokenizer st1 = new StringTokenizer("Stringa da dividere");
StringTokenizer st2 = new StringTokenizer("Stringa sezionata", ";");
StringTokenizer st3 = new StringTokenizer("Ciao Mamma", "a", true);
```

Per scandire l'intero testo si può usare un ciclo while con all'interno l'invocazione del metodo hasMoreTokens() che ritorna true se sono presenti altri token, altrimenti false. Per stampare il token appena recuperato si può invocare il metodo nextToken() sull'oggetto StringTokenizer.

```
StringTokenizer st = new StringTokenizer("Stringa da dividere");
while (st.hasMoreTokens()) {
    // Due metodi per fare la stessa cosa
System.out.println(st.nextToken());
System.out.println(st.nextElement().toString());
}
```

Costruttori pubblici:

Costruttore	**definizione**
StringTokenizer(String str)	Costruisce uno StringTokenizer per la stringa str, che come delimitatori usa i caratteri "\t\n\r\f".
StringTokenizer(String str, String delim)	Costruisce uno StringTokenizer per la stringa str, che come delimitatori usa i caratteri contenuti nella stringa delim.
StringTokenizer(String str, String delim, boolean returnDelims)	Costruisce uno StringTokenizer per la stringa str, che come delimitatori usa i caratteri contenuti nella stringa delim. Se returnDelims ﷺ true, i caratteri divisori verranno restituiti come token.

Metodi pubblici:

Metodo	Definizione
int countTokens()	Restituisce il numero dei token elaborati.
boolean hasMoreElements()	Come il successivo hasMoreTokens().
boolean hasMoreTokens()	Restituisce true se ci sono ancora dei token da considerare.
Object nextElement()	Restituisce il token successivo, sotto forma di Object.
String nextToken()	Restituisce il token successivo, sotto forma di String.
String nextToken(String delim)	Imposta una nuova serie di caratteri delimitatori, quindi restituisce il token successivo.

Classe StringBuffer

Un oggetto String

- NON è modificabile
- Una volta creato non possiamo aggiungere, eliminare, modificare caratteri (i metodi visti creano nuove stringhe)
- Tale restrizione è dovuta a ragioni di efficienza

Le considerazione precedenti non sono vere per la classe StringBuffer

Esempio

`StringBuffer myStringBuffer = new stringBuffer ("stringa modificabile");

myStringBuffer.setCharAt (8, 'M'); // Trasforma in "stringa Modificabile"`

Si usa raramente

Un oggetto StringBuffer non può essere utilizzato per operazioni di I/O

```
System.out.println (myStringBuffer.toString());
```

Metodi:

- Aggiunta di caratteri `myStringBuffer.append ("aggiunta");`
- insert
- delete
- reverse

I più importanti metodi di cui sono dotati gli oggetti di tipo StringBuffer.

Tipo restituito	Metodi e parametri	Descrizione

Tipo restituito	Metodi e parametri	Descrizione
StringBuffer	append(boolean b)	Aggiunge il valore b in coda alla stringa. Modifica l'oggetto di invocazione, ed in più restituisce un riferimento allo stesso StringBuffer.
StringBuffer	append(char c)	Aggiunge il carattere c in coda alla stringa. Modifica l'oggetto di invocazione, ed in più restituisce un riferimento allo stesso StringBuffer.
StringBuffer	append(char[] c)	Aggiunge i caratteri contenuti nell'array in coda alla stringa. Modifica l'oggetto di invocazione, ed in più restituisce un riferimento allo stesso StringBuffer.
StringBuffer	append(double d)	Aggiunge il valore di d in coda alla stringa. Modifica l'oggetto di invocazione, ed in più restituisce un riferimento allo stesso StringBuffer.
StringBuffer	append(float f)	Aggiunge il valore di f in coda alla stringa. Modifica l'oggetto di invocazione, ed in più restituisce un riferimento allo stesso StringBuffer.
StringBuffer	append(int i)	Aggiunge il valore di i in coda alla stringa. Modifica l'oggetto di invocazione, ed in più restituisce un riferimento allo stesso StringBuffer.
StringBuffer	append(long l)	Aggiunge il valore di l in coda alla stringa. Modifica l'oggetto di invocazione, ed in più restituisce un riferimento allo stesso StringBuffer.
StringBuffer	append(Object obj)	Aggiunge il valore di obj.String() in coda alla stringa. Modifica l'oggetto di invocazione, ed in più restituisce un riferimento allo stesso StringBuffer.
StringBuffer	append(String s)	Aggiunge s in coda alla stringa. Modifica l'oggetto di invocazione, ed in più restituisce un riferimento allo stesso StringBuffer.
StringBuffer	append(StringBuffer s)	Aggiunge s in coda alla stringa. Modifica l'oggetto di invocazione, ed in più restituisce un riferimento allo stesso StringBuffer.
char	chatAt(int i)	Restituisce il carattere alla posizione i.
StringBuffer	delete(int start, int end)	Rimuove tutti i caratteri dall'indice start (incluso) all'indice end (escluso). Modifica l'oggetto di invocazione, ed in più restituisce un riferimento allo stesso StringBuffer.

Tipo restituito	Metodi e parametri	Descrizione
StringBuffer	deleteCharAt(int i)	Rimuove il carattere alla posizione i. Modifica l'oggetto di invocazione, ed in più restituisce un riferimento allo stesso StringBuffer.
int	indexOf(String s)	Restituisce la prima posizione della sottostringa s, oppure -1 nel caso tale sottostringa non compaia nell'oggetto di invocazione.
int	indexOfString(String s, int i)	Come il precedente, con la differenza che la ricerca della sottostringa s prende prende piede dalla posizione i.
StringBuffer	insert(int offset, boolean b)	Aggiunge il valore di b alla stringa, inserendolo alla posizione offset. Modifica l'oggetto di invocazione, ed in più restituisce un riferimento allo stesso StringBuffer.
StringBuffer	insert(int offset, char c)	Aggiunge il carattere c alla stringa, inserendolo alla posizione offset. Modifica l'oggetto di invocazione, ed in più restituisce un riferimento allo stesso StringBuffer.
StringBuffer	insert(int offset, char[] c)	Aggiunge i caratteri contenuti nell'array alla stringa, inserendoli alla posizione offset. Modifica l'oggetto di invocazione, ed in più restituisce un riferimento allo stesso StringBuffer.
StringBuffer	insert(int offset, double d)	Aggiunge il valore di d alla stringa, inserendolo alla posizione offset. Modifica l'oggetto di invocazione, ed in più restituisce un riferimento allo stesso StringBuffer.
StringBuffer	insert(int offset, float f)	Aggiunge il valore di f alla stringa, inserendolo alla posizione offset. Modifica l'oggetto di invocazione, ed in più restituisce un riferimento allo stesso StringBuffer.
StringBuffer	insert(int offset, int i)	Aggiunge il valore di i alla stringa, inserendolo alla posizione offset. Modifica l'oggetto di invocazione, ed in più restituisce un riferimento allo stesso StringBuffer.
StringBuffer	insert(int offset, long l)	Aggiunge il valore di l alla stringa, inserendolo alla posizione offset. Modifica l'oggetto di invocazione, ed in più restituisce un riferimento allo stesso StringBuffer.
StringBuffer	insert(int offset, Object obj)	Aggiunge il valore di obj.toString() in coda alla stringa. Modifica l'oggetto di invocazione, ed in più restituisce un riferimento allo stesso StringBuffer.
StringBuffer	insert(int offset, String s)	Aggiunge s in coda alla stringa. Modifica l'oggetto di invocazione, ed in più restituisce un riferimento allo stesso StringBuffer.

Tipo restituito	Metodi e parametri	Descrizione
StringBuffer	insert(int offset, StringBuffer s)	Aggiunge s in coda alla stringa. Modifica l'oggetto di invocazione, ed in più restituisce un riferimento allo stesso StringBuffer.
int	length()	Restituisce la dimensione della stringa.
StringBuffer	setCharAt(int i, char c)	Cambia in c il carattere alla posizione i.
void String toString()	Restituisce un oggetto String con il medesimo contenuto dello StringBuffer di invocazione.	