Stringhe

Stringa come oggetto

sequenza finita di elementi sintatticamente concatenati in una frase

Stringhe di caratteri in Java: l'oggetto String

In Java le stringhe sono oggetti appartenenti alla classe String

Una stringa in Java rappresenta uno specifico valore e come tale è *immodificabile*:

una String non è un contenitore

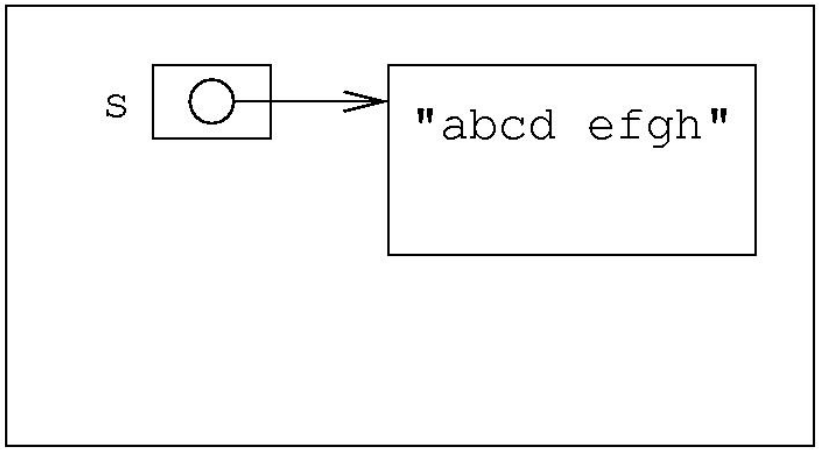
- se occorre, esistono altre classi, ad esempio StringBuilder o StringBuffer

Per un confronto di performances

Un oggetto di tipo String è una stringa costante

String s;

s = “abcd efgh”;



memoria

Varibile: conterrà il riferimento all’oggetto, NON l’oggetto

```
String frase = “Questa è una stringa”;
```

Tipo di dato: riferimento a String
Notate l’iniziale maiuscola

Oggetto String

Stringhe di caratteri in Java: l'oggetto String

COSTANTI `String`

- Le costanti `String` possono essere denotate nel modo usuale:

  `"ciao"` `"mondo\n"`

- Quando si scrive una costante `String` tra virgolette, *viene creato implicitamente un nuovo oggetto di classe `String`,* inizializzato a tale valore.
- Una costante `String` *non può eccedere la riga*: quindi, dovendo scrivere stringhe più lunghe, conviene *spezzarle e concatenarle con +.*

L'operatore + denota la concatenazione fatta a *tempo di compilazione*

→ *senza inefficienze*

La creazione di un oggetto String

La creazione di un oggetto String può essere:

- **Esplicita:**
  attraverso il costruttore *polimorfo* String() … ne esistono 11
  Es: String s = new String( );
  String s = new String(“parola”);

- **Implicita:**
  quando il compilatore incontra una serie di caratteri delimitati da virgolette, crea un oggetto di tipo String a cui assegna il valore della stringa individuata
  Es: String s = “parola”; *Il equivale a*
  String s = new String(“parola”);

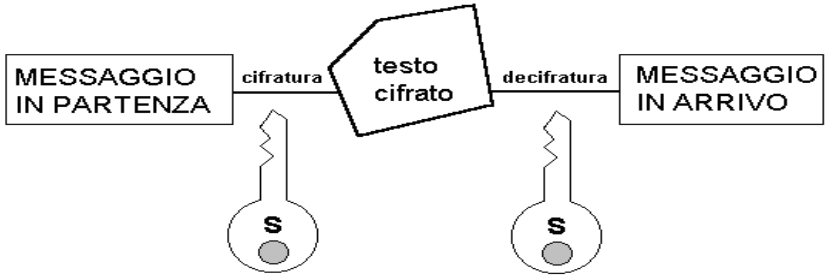
Oggetto String: insieme di caratteri

Un oggetto String può essere analizzato carattere per carattere

```
                import
          java.io.PrintStream;
       public class ASmileyFace {
    public static void main(String[]
a) { PrintStream out = System.out ;
out.print(     "This"      + " is " +
"both a "        +         "reform"+
"atted J"    +  "av"  +     "a prog"+
"ram, an"        +         "d a sm"+
"i".concat(   "ley " +    "face. Th"+
"is "    + "shows how to " +    "f"+
  "orm"     + "at Jav" +      "a "+
    "usi"                   +"n"+
       "g whitespace only.")
                } ;) }
```

Oggetto String: insieme di caratteri

Si pensi ad una applicazione capace di **anagrammare** una parola



Oppure si voglia realizzarne una capace di cifrare un messaggio segreto, usando il *codice di Cesare con s =3*

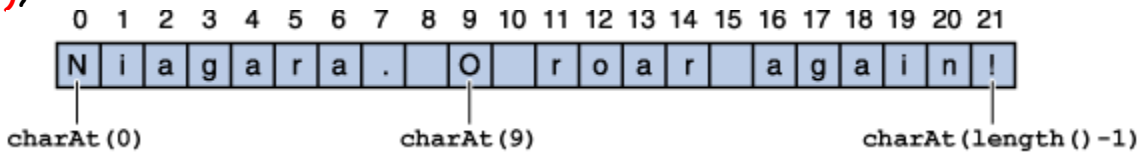
Chiaro	AUGURIDIBUONCOMPLEANNO
Spostamento	3
Cifra Pulisci	
Cifrato	DXJXULGLEXRQFRPSOHDQQR

Storia della crittografia

Stringhe di caratteri in Java: l'oggetto String

- Per *selezionare il carattere i-esimo* si usa il metodo **charAt()**:

```
String s = "Niagara. O roar again!";
char ch = s.charAt(9);
```




```
// la variabile ch conterrà il carattere O
```

- La classe String definisce *decine* di metodi:
  si veda la documentazione (Java 2 API) e i tutorial

API

Tutorial

La classe String: alcuni metodi principali

In riferimento a: **String s = "**parola**";**

- **length()** ritorna la lunghezza della stringa

  int len = s.length(); // len vale **6**

- **indexOf(char c)** ritorna l'indice della prima occorrenza del carattere indicato

  int i = s.indexOf('o'); // i vale **3** a partire da 0

- **lastIndexOf(char c)** ritorna l'indice dell'ultima occorrenza del carattere indicato

  int i = s.indexOf('a'); // i vale **5** a partire da 0

- **substring(int i, int k)** ritorna una sottostringa con indice da i a k-1

  String sub = s.substring(2,4); // sub vale "**ro**"

Metodi di confronto tra stringhe

- metodo **equals()** che restituisce un valore *boolean*
  true se sono uguali le due stringhe confrontate

```
if (str1.equals (str2))
```

anche metodo equalsIgnoreCase() ignorando maiuscole e minuscole

- metodo **compareTo()** che restituisce un valore *intero*

```
if (str1.compareTo (str2) == -1 )  //  -1 se alfabeticamente str1<str2
if (str1.compareTo (str2) == 0 )  //  0 se uguali
if (str1.compareTo (str2) == 1 )  //  1 se alfabeticamente str1>str2
```

anche metodo compareToIgnoreCase()
ignorando maiuscole e minuscole

Metodi di conversione

Conversione implicita: prevale tipo che occupa più memoria

Es: char ch = 'a'; // carattere Unicode a 16 bit

```
System.out.println("Carattere: " + ch);
```

Converte ch in stringa e lo concatena alla frase.

Conversione esplicita: metodo **valueOf()**, ereditato da classe Object, per convertire variabili di tipo numerico in String

Es: int num = 10 ; // intero a 32 bit
String s = String.valueOf(num);

Il metodo `toString()`

- Tutte le classi Java definiscono un metodo **`toString()`**, ereditato dalla classe Object, che produce un oggetto String a partire da un oggetto della classe; *questo consente di "stampare" facilmente qualunque oggetto di qualunque classe*
- È responsabilità del progettista definire un metodo toString() che produca una stringa "*significativa*"
- Quello di default, infatti, stampa un identificativo alfanumerico dell'oggetto

Il metodo toString()

Con riferimento all'uso di una progettata classe Counter:

```
Counter c = new Counter( );
System.out.println(c);
```

Usa il metodo toString() predefinito di Counter → Stampa un identificativo dell'oggetto c.

```
Counter@4abc9
```

- Potete **ridefinire esplicitamente il metodo `toString()` della classe `Counter`**, *facendogli stampare cio che preferite*

Ad esempio:

```
public class Counter {
 ... int val= 10;
 public String toString(){
   return "Counter di valore " + val;
 }
}
```

ora stampa:

```
Counter di valore 10
```