Classe String:

- le stringhe sono istanze della classe String
- due costruttori
  - senza parametri, per la stringa vuota

    `String e = new String ();`
  - con un parametro di tipo array di caratteri

    `char elem[] = {'h', 'e', 'l', 'l', 'o'};`

    `String s = new String (elem);`

`String e = "";`

`String s = "hello";`

Le costanti stringa sono oggetti della classe String

le stringhe non terminano con un carattere speciale

`String s;`

Costruttore di copia:

`String r = new String (s);` ← crea una copia della String s

`String t = new String ("abc");` ← crea una copia della String "abc"

Operatore:

unico concesso è + per la concatenazione

Tutti gli oggetti hanno un metodo toString di default

lunghezza

Se s è un oggetto di tipo String

`s.length()`

e ritorna la lunghezza della stringa s

```
String e = "hello";
System.out.println(e.length()); // 5
```

accesso di un carattere di una stringa

accesso all'i-esimo carattere di una stringa

```
s.charAt(i);
```

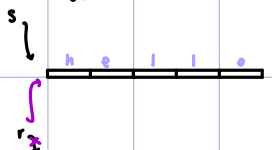
ritorna il carattere i, se fuori dai limiti lancia un'eccezione

non è possibile ritornare un risultato reference return.

le stringhe sono oggetti immutabili

Assegnamento: avviene fra reference e non fra oggetti

```
String s = "hello";
String r;
r = s;
```



condividono lo stesso oggetto ma quello rimane immutabile

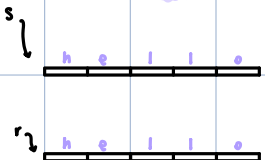
confronto fra stringhe

anche == si applica alla reference

```
s == r
```

true se puntano allo stesso oggetto (reference) = Array

Se voglio confrontare due oggetti di tipo String, esiste un metodo **equals** (in realtà tutte le classi hanno questo metodo)




```
String s = "hello";
String r = "hello";
s.equals(r)  // true
if (s == "hello")
```

N.B. se ci sono tanti oggetti equals li elide e fa reference tutti allo stesso

IndexOf

Se s è un oggetto tipo String e c un carattere

```
s.IndexOf(c)
```

ritorna l'indice della prima occorrenza di c in s, oppure -1 se c non è presente nella stringa s

Substring

- Se s è un oggetto di tipo String e init e end interi da 0 a s.length() - 1

  s.substring (init, end)

  ritorna la sottostringa s a partire da init fino a end-1

- se end è omesso

  s.substring (init)

  ritorna la sottostringa di s a partire da init fino a s.length(-1) - 1

- se gli indici sono fuori dai limiti della stringa : eccezione

1)

```
import java.util.Scanner;
public class Maiuscola {
        public static void main (String[] args) {
        String s;
        Scanner sc = new Scanner (System.In);
        s = sc.nextLine ();
        sc.close ();
        System.out.println( "Stringa letta: " + s );
        char c = Character.toUpperCase (s.charAt(0));
        Strin r = c + s.substring (1);
        System.out.println( " Stringa convertita : "+ r );
        }
}
```

2) CString in java

```
public class StringSet {
    private String [] s;
    public StringSet (){
        this.s = new String [0]; }
    public void add (String x) {
        if (!contains (x)){
            String newSet = new String [s.length +1];
            for (int i =0; i < s.length; i ++) {
                newSet [i] = s[i]}
            newSet [s.length] = x;
            s = newSet;
            }
        }
    public boolean contains (String x) {
        for (int i =0; i < s.length; i ++){
            if (s[i].equals (x)) {
                return true;}
            return false;
    public int size () {
        return s.length;}
    public StringSet union (StringSet other) {
        StringSet result = new StringSet ();
            for (int i =0; i < size (); i ++) {
                result.add (s[i]); }
```

```
            for (int i =0; i < other.size(); i ++) {
                result.add ( other.s[i]); }
        return  result;
    public    boolean   isSubSetOf  (StringSet  other){
        if (other.size() >= size()) {
            return false; }
        for (int i=0; i< size(); i++){
            if (!other. contains(s[i])) {
                return  false; }
            }
        return  true;
}
    public  StringSet  intersection (StringSet   other) {
        StringSet  result = new  StringSet();
        for  (int i=0; i < size(); i ++) {
            if ( other. contain (s[i])){
                result. add (s[i]); }
            }
        return  result; }
    public   String   getString() {
        String  r = " { ";
        for  (int i = 0;  i < size(); i ++){
            r= r + s[i]; }
```

```
            if (i< size()-1) {

                r = r + " , ";}

            r= r+ " } ";

            return  r; }

}
```