Stringhe in Java

Concatenation

La concatenazione è l'unione di due o più stringhe.

Esempi di Sintassi delle Stringhe
Abbiamo due stringhe `str1 = "Rock" e str2 = "Star"`
Se uniamo queste due stringhe, dovremmo ottenere un risultato come `str3 = "RockStar"`.
Ci sono due metodi per concatenare le stringhe in Java: il primo utilizza il metodo concat della classe String e il secondo utilizza l'operatore aritmetico +. Entrambi producono lo stesso output.

```java
public class Sample_String{

  public static void main(String[] args){
    String str1 = "Rock";
    String str2 = "Star";

    // Metodo 1: Utilizzando concat
    String str3 = str1.concat(str2);
    System.out.println(str3);

    // Metodo 2: Utilizzando l'operatore "+"
    String str4 = str1 + str2;
    System.out.println(str4);
  }

}
```

length

Per ottenere la lunghezza di una stringa, utilizziamo il metodo length().

```
public class Sample_String{

  public static void main(String[] args){
    String str_Sample = "RockStar";

    // Lunghezza di una stringa
    System.out.println("Lunghezza della stringa: " + str_Sample.length());
  }
}
```

indexOf

Per ottenere l'indice di un carattere o di una sottostringa, utilizziamo il metodo indexOf().

```
public class Sample_String{

  public static void main(String[] args){
    String str_Sample = "RockStar";

    // Carattere in posizione
    System.out.println("Carattere in posizione 5: " + str_Sample.charAt(5));

    // Indice di un dato carattere
    System.out.println("Indice del carattere 'S': " + str_Sample.indexOf('S'));
  }
}
```

charAt

Similmente al precedente, dato l'indice di un carattere, possiamo ottenere il carattere corrispondente.

```
public class Sample_String{

  public static void main(String[] args){
    String str_Sample = "RockStar";

    // Carattere in posizione
    System.out.println("Carattere in posizione 5: " + str_Sample.charAt(5));
  }
}
```

CompareTo

Ci sono due metodi per confrontare le stringhe in Java: il primo è il metodo `compareTo()` e il secondo è il metodo `compareToIgnoreCase()`.

`compareToIgnoreCase` si utilizza per confrontare due stringhe ignorando maiuscole e minuscole.
`compareTo` si utilizza per confrontare due stringhe considerando maiuscole e minuscole.

Il risultato è un intero che può essere:

- 0: se le stringhe sono uguali
- > 0: se la stringa è maggiore della stringa specificata
- < 0: se la stringa è minore della stringa specificata

```java
public class Sample_String{

  public static void main(String[] args){
    String str_Sample = "RockStar";

    // Confronta con una Stringa
    System.out.println("Confronta con 'ROCKSTAR': " + str_Sample.compareTo("rockstar"));

    // Confronta - Ignora il case
    System.out.println("Confronta con 'ROCKSTAR' - Ignora il case: " + str_Sample.compareToIgnoreCase("ROCKSTAR"));
  }
}
```

Risultato:

```
Confronta con 'ROCKSTAR': 32
Confronta con 'ROCKSTAR' - Ignora il case: 0
```

Contains

Il metodo `contains` è utilizzato per controllare se una stringa contiene una sequenza di caratteri specificata.

```
public class Sample_String{

  public static void main(String[] args){
    String str_Sample = "RockStar";

    System.out.println("Contiene la sequenza 'tar': " + str_Sample.contains("tar"));
  }
}
```

endsWith

Se al posto di controllare se una stringa contiene una sequenza di caratteri, vogliamo controllare se termina con una particolare sequenza, possiamo utilizzare il metodo `endsWith`.

```
public class Sample_String{

  public static void main(String[] args){  // Controlla se termina con una particolare sequenza
    String str_Sample = "RockStar";

    System.out.println("Termina con il carattere 'r': " + str_Sample.endsWith("r"));
  }
}
```

startsWith

Se al posto di controllare se una stringa termina con una particolare sequenza, vogliamo controllare se inizia con una particolare sequenza, possiamo utilizzare il metodo `startsWith`.

```
public class Sample_String{

  public static void main(String[] args){
    String str_Sample = "RockStar";

    // Controlla se inizia con una particolare sequenza
    System.out.println("Inizia con il carattere 'R': " + str_Sample.startsWith("R"));
  }
}
```

replaceAll e replaceFirst

Il metodo `replaceAll` è utilizzato per sostituire tutte le occorrenze di una stringa con un'altra stringa.
Il metodo `replaceFirst` è utilizzato per sostituire la prima occorrenza di una stringa con un'altra stringa.

La differenza tra `replace` e `replaceAll` è che `replaceAll` accetta espressioni regolari mentre `replace` accetta solo stringhe.

```
public class Sample_String{

  public static void main(String[] args){//Sostituisci Rock con la parola Trap
    String str_Sample = "RockStar";

    System.out.println("Sostituisci 'Rock' con 'Trap': " + str_Sample.replace("Rock", "Trap"));
  }
}
```

toLowerCase e toUpperCase

Il metodo `toLowerCase` è utilizzato per convertire una stringa in minuscolo, mentre il metodo `toUpperCase` è utilizzato per convertire una stringa in maiuscolo.

```
public class Sample_String{

  public static void main(String[] args){//Converti in minuscolo
    String str_Sample = "RockStar";

    System.out.println("Converti in minuscolo: " + str_Sample.toLowerCase());

    //Converti in maiuscolo
    System.out.println("Converti in maiuscolo: " + str_Sample.toUpperCase());
  }
}
```

trim

Il metodo `trim` è utilizzato per rimuovere gli spazi bianchi iniziali e finali da una stringa.

```
public class Sample_String{

  public static void main(String[] args){
    String str_Sample = "   RockStar   ";

    System.out.println("Prima del trim: " + str_Sample);
    System.out.println("Dopo il trim: " + str_Sample.trim());
  }
}
```

split

Il metodo `split` è utilizzato per dividere una stringa in sottostringhe utilizzando un delimitatore specificato.

```
public class Sample_String{

  public static void main(String[] args){
    String str_Sample = "I am a RockStar";

    // Dividi la stringa in sottostringhe
    String[] arr_Sample = str_Sample.split(" ");
    for (String a : arr_Sample)
      System.out.println(a);
  }
}
```