String, StingBuffer e StringBuilder

La prima cosa da sottolineare è che l'oggetto **String** è un **oggetto immutabile**, cioè, una volta creato, non può essere modificato. Di fatto, nel momento in cui modifichiamo un oggetto String, in Java, ne stiamo creando un'altra istanza, rimpiazzando quella precedente.

Gli stessi operatori di concatenazione (`x = x + "aaa"`) non fanno altro che fare l'override dei metodi presenti nella classe StringBuffer con la quale la classe String viene gestita per questioni di *performance*.

Proprio per questo motivo in ambiente *pre java 1.5* il modo più ovvio per gestire **catene di testo mutabili** era quello di usare direttamente la classe **StringBuffer** ed i relativi metodi per la gestione del testo (in particolare del metodo **append** e dei metodi **replace**).

La classe è una ***threadsafe***, il che consente di poter effettuare le operazioni sull'oggetto StringBuffer condividendolo tra diversi thread.

StringBuilder è identica a StringBuffer, stessi metodi stessa logica, unica differenza: ***non è threadsafe***. Le ***performance*** **migliorano** in maniera netta, in particolare considerando che molti programmi fanno un notevole uso delle stringhe e della loro modifica durante il ciclo di vita del software.

L'utilizzo, quindi, in applicazioni *enterprise*, in cui **non abbiamo da gestire** aspetti legati all'**accesso concorrente alla risorsa**, dovrebbe essere scontato a favore della nuova classe introdotta: **StringBuilder**

http://java.html.it/guide/lezione/3994/stringbuilder/

Test sull'uso delle Stringhe con lo stesso listing nella **misura del tempo** per inizializzare String con assegnamento, StringBuffer e StringBuilder con append() in ciclo dove il contenuto è l'indice da 0 a 15000

Tempo di esecuzione testString() 1656 millis.
Tempo di esecuzione testStringBuffer() 16 millis.
Tempo di esecuzione testStringBuilder() 0 millis.

Secondo lancio

Tempo di esecuzione testString() 1423 millis.
Tempo di esecuzione testStringBuffer() 2 millis.
Tempo di esecuzione testStringBuilder() 1 millis.

Terzo lancio

Tempo di esecuzione testString() 1386 millis.
Tempo di esecuzione testStringBuffer() 1 millis.
Tempo di esecuzione testStringBuilder() 2 millis.

Quarto lancio

Tempo di esecuzione testString() 1404 millis.
Tempo di esecuzione testStringBuffer() 2 millis.
Tempo di esecuzione testStringBuilder() 1 millis.

NB: nel caso di StringBuffer e StringBuilder la mancata ridefinizione del metodo **equals** provoca un comportamento inatteso.

Due oggetti di questo tipo infatti vengono ritenuti "uguali" dal metodo equals se e solo se vengono generati a partire da una stessa istanza di un oggetto String, prescindendo dal contenuto.

È pertanto necessario ridefinire tale metodo o usare un istanza della classe String.

Si ricordi che:

- per creare una stringa modificabile a partire da una String immodificabile, la sintassi è:

  StringBuilder *nomeStringBuilder* = new StringBuilder(*nomeString*);

- per convertire da StringBuider a String, si ricorre al metodo toString e la sintassi è:

  String *nomeString* = *nomeStringBuilder*.**toString()**;

listing

```
public class StringConfronto {

  /**
   * @param args
   */
  public static void main(String[] args) {
    long start=System.currentTimeMillis();
    testString();
    long end=System.currentTimeMillis();
    System.out.println("Tempo di esecuzione testString() "+(end-start)+" millis.");

    start=System.currentTimeMillis();
    testStringBuffer();
    end=System.currentTimeMillis();
    System.out.println("Tempo di esecuzione testStringBuffer() "+(end-start)+" millis.");

    start=System.currentTimeMillis();
    testStringBuilder();
    end=System.currentTimeMillis();
    System.out.println("Tempo di esecuzione testStringBuilder() "+(end-start)+" millis.");
  }

  private static void testString() {
    String x = "";
    for(int i=0;i<15000;i++){
          x+=i;          //operazione di append con incremento
    }
  }

  private static void testStringBuffer() {
    StringBuffer x = new StringBuffer("");
    for(int i=0;i<15000;i++){
          x.append(i);   //operazione di append con metodo
    }
  }

  private static void testStringBuilder() {
    StringBuilder x = new StringBuilder("");
    for(int i=0;i<15000;i++){
          x.append(i);  //operazione di append con metodo
    }
  }
}
```