Table of Contents

[1. Java String 2](#_Toc3788508)

[2. Expresii regulate 3](#_Toc3788509)

[3. Java StringBuilder, StringBuffer 3](#_Toc3788510)

[4. Clase immutable 4](#_Toc3788511)

[5. Clase abstracte 5](#_Toc3788512)

[6. Referințe 6](#_Toc3788513)

# Java String

Clasa String din java este un bun exemplu de clasa immutable. Este o clasă mai mult utilitară.

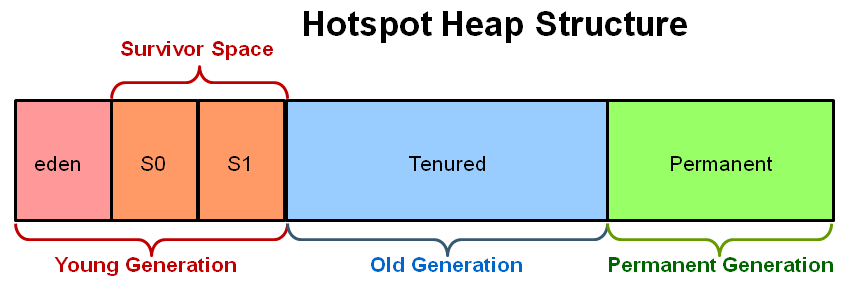
Orice operațiune se va executa asupra ei va avea ca finalitate un alt obiect de tip String.

*// declaring a String*  
String string = **"my first string"**;  
String otherString = **new** String(**"other string"**);  
String thirdString = string + **", "** + otherString;  
  
System.***out***.println(**"third: "** + thirdString);  
  
*// search for a string in a string***int** i = thirdString.indexOf(**", "**);  
System.***out***.println(**"find \", \" at position: "** + i);  
  
*// search for a char in a string, instead of double quote we have simple quotes*System.***out***.println(**"find ',' at position: "** + thirdString.indexOf(**','**));  
  
*// substring start from a positiion till the end*String forthString = thirdString.substring(i + 2);  
System.***out***.println(**"forthString: "** + forthString);  
  
*// begin, end - stop at a defined end*forthString = thirdString.substring(i + 2, i + 2 + 5);  
System.***out***.println(**"forthString: "** + forthString);  
  
*// begin, end - stop at a defined end after real end,  
// will result in an exception Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String index out of range*forthString = thirdString.substring(i + 2, i + 2 + 5 + 25);  
System.***out***.println(**"forthString: "** + forthString);  
  
*// String create from StringBuilder*StringBuilder stringBuilder = **new** StringBuilder();  
stringBuilder.append(**"un string urmat de un integer: "**).append(22).append(**", apoi de un caracter"**).append(**';'**);  
String fifthString = **new** String(stringBuilder);  
System.***out***.println(**"fifthString: "** + fifthString);  
*// or*System.***out***.println(**"stringBuilder: "** + stringBuilder.toString());  
  
*// the same will work for StringBuffer*

Pentru o mai eficientă utilizare a string-urilor, ele se păstrează într-un pool.

Clasele String instanțiate cu new nu ajung in pool.

*// new for new string's*String s1 = **"un string"**;  
String s2 = **"un string"**;  
String s3 = **new** String(**"un string"**);  
  
System.***out***.println(**"s1 == s2 :"**+(s1==s2));  
System.***out***.println(**"s1 == s3 :"**+(s1==s3));  
*// output  
// s1 == s2 :true  
// s1 == s3 :false*



JDK 7 în 8 Permanent a devenit Metadata

split – regular expresion

isEmpty – return 0 “”

indexOf – căutare un caracter sau un șir de string-uri, se poate menționa si de unde să înceapă căutarea

lastIndexOf – căutare un caracter sau un șir de string-uri, se poate menționa si de unde să înceapă căutarea

length - lungimea

matches – regular expresion, se poate verifica dacă string-ul, în întregime, respectă un anume pattern

toLowerCase

toUpperCase

substring

startsWith

endsWith

sursa - <https://docs.oracle.com/javase/8/docs/api/index.html?java/util/StringTokenizer.html>

# Expresii regulate

split

“ ” – spațiu

email

"^[A-Z0-9.\_%+-]+@[A-Z0-9.-]+\\.[A-Z]{2,6}$"

# Java StringBuilder, StringBuffer

Amândouă sunt clase utilitare ce vă ajută sa construiți obiecte String fără sa fragmentați memoria prea mult.

StringBuilder este mult mai rapid ca StringBuffer însă nu este sincronizat.

Operațiunea append este cea mai uzitată, permite adăugarea la sfărșit a altor obiecte. Se apeleaza metoda lor toString().

De asemenea pentru a obține un string se apelează toString.

Un număr de metode uzitate ar fi:

delete

substring

indexOf

lastIndexOf

replace

reverse

mai multe se pot găsi: <https://docs.oracle.com/javase/8/docs/api/java/lang/StringBuilder.html>

Comparație



# Clase immutable

Reguli pentru a implementa propria clasă immutable

1. Clasa trebuie sa fie final, să nu permită extinderea/moștenirea
2. Variabilele trebuie să fie private și final
3. Dacă în constructor primim ca parametru o variabilă mutabilă trebuie să realizăm o copie a ei
4. Nu se vor oferi metode setter
5. Dacă avem variabile mutabile ce sunt returnate prin getter, se va face o copie a lor și se va returna referința lor.

Proprietăți pe care le au clasele immutable

* Simplu de construit, testat și utilizat
* Sunt thread-safe , nu au probleme de sincronizare
* Nu necesită un constructor copy (nu ai de ce)
* Nu necesită clone ( nu ai de ce)

Joshua Bloch, în cartea Effective Java recomandă:

"Classes should be immutable unless there's a very good reason to make them mutable....If a class cannot be made immutable, limit its mutability as much as possible."

# Clase abstracte

Proprietăți:

* clasă abstractă se declară folosind keyword-ul abstract
* Poate avea și metode ne-abstracte
* Nu pot fi instanțiate
* Pot avea constructor
* Pot avea metode statice
* Pot avea metode final

Implementarea unei clase abstracte într-o clasă concretă

abstract class => class

* Pentru fiecare metodă abstractă se va realiza o implementare concretă, nu abstractă
* Implementarea metodelor va respecta aceleași principii ca pentru override

# Referințe

1. Java string class pool, <https://www.journaldev.com/797/what-is-java-string-pool>
2. Java memory area, <https://www.programcreek.com/2013/04/jvm-run-time-data-areas/>
3. Jvm memory 1.7, <https://dzone.com/articles/java-memory-architecture-model-garbage-collection>
4. Java StringBuilder at Oracle, <https://docs.oracle.com/javase/8/docs/api/java/lang/StringBuilder.html>
5. Java string oracle documentation, <https://docs.oracle.com/javase/8/docs/api/index.html?java/util/StringTokenizer.html>
6. Expresii regulate, <https://docs.oracle.com/javase/tutorial/essential/regex/>
7. Expresii regulate – construcție, <https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html>