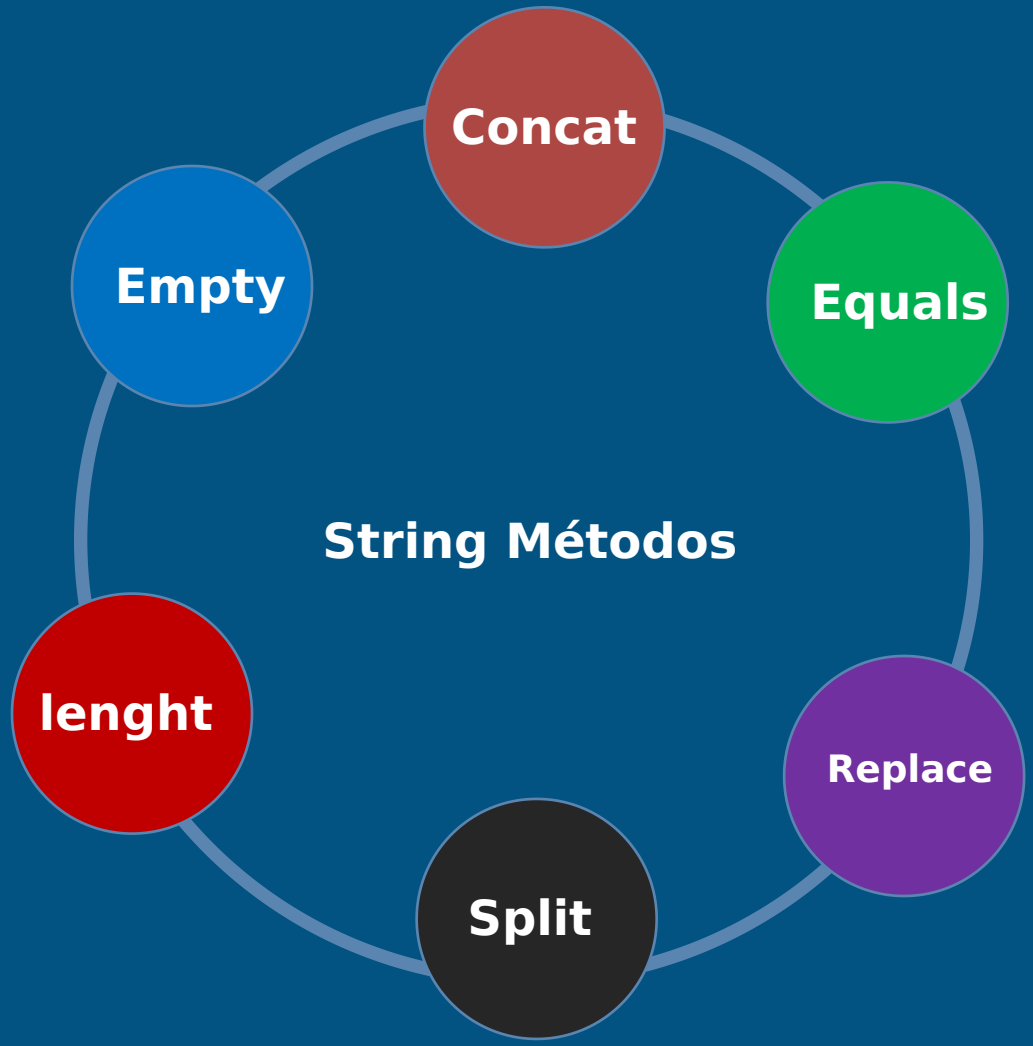
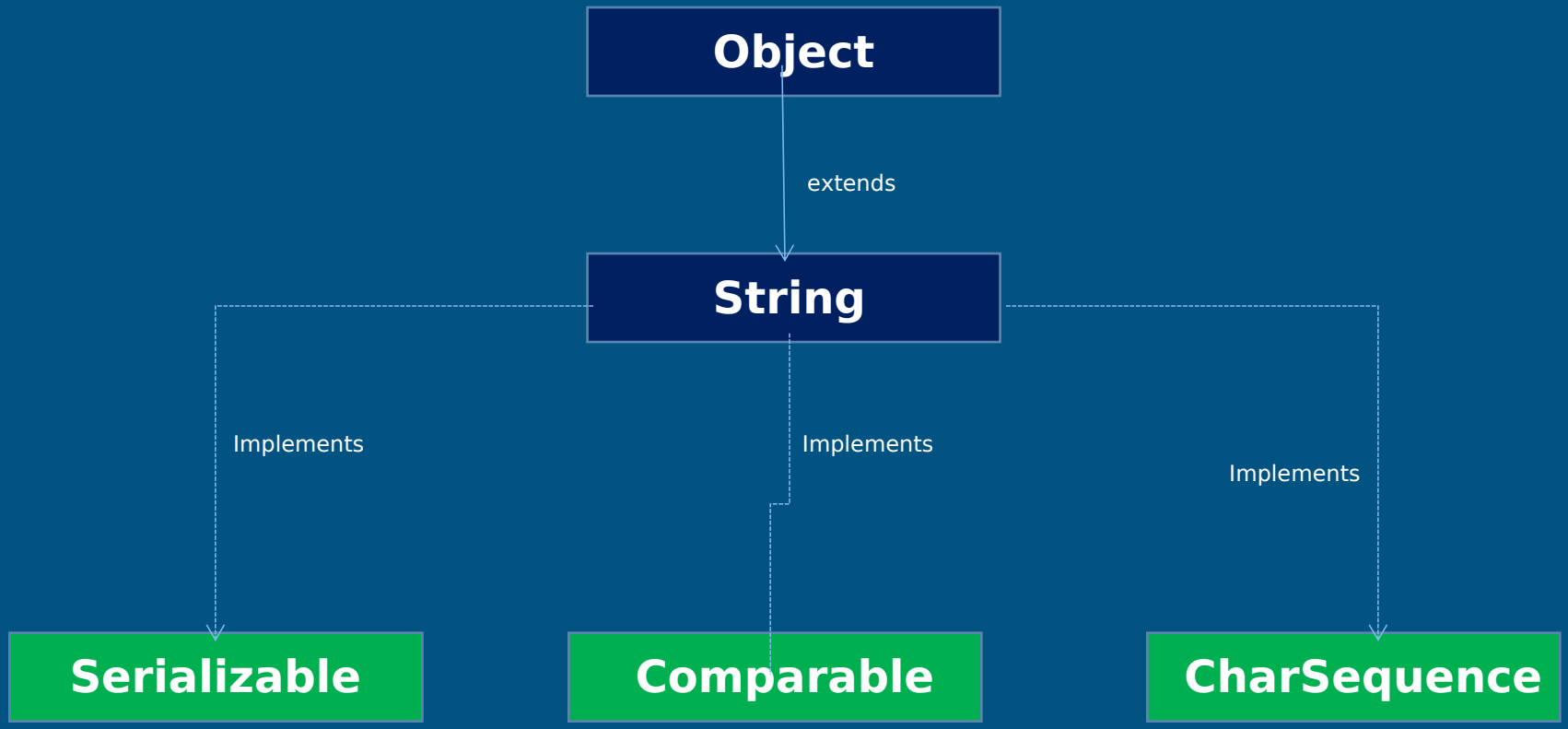


Java

String

Java String


Object
extends
String
Implements
Implements
Implements
Serializable
Comparable
CharSequence

Java String

Serializable

Serializable é uma interface de marcador (não possui membro e método de dados).
É usado para "marcar" as classes Java para que os objetos dessas classes possam obter um determinado recurso.
O Cloneable e o Remote também são interfaces de marcador.

Comparable

O método Java String compareTo () é usado para comparar duas seqüências lexicograficamente. Cada caractere de ambas as seqüências é convertido em um valor Unicode para comparação. ...
O resultado é positivo se a primeira string for lexicograficamente maior que a segunda, caso contrário, o resultado seria negativo.

CharSequence

Uma CharSequence é uma sequência legível de valores de char. Essa interface fornece acesso uniforme e somente leitura a muitos tipos diferentes de seqüências de caracteres.
Um valor de caractere representa um caractere no plano multilíngue básico (BMP) ou um substituto.

Java String

01 **Object**

String é um Objeto Java

02 **Characters**

Representa uma sequencia
de caracteres

03 **Class**

java.lang.String class
manipular e criar String

04 **Immutable**

String e imutável na
sua essencia

Java String

CharSequence

String

A classe String representa cadeias de caracteres. Todos os literais de cadeia de caracteres nos programas Java, como "abc", são implementados como instâncias dessa classe.

Strings são constantes; seus valores não podem ser alterados após serem criados.

StringBuffer

Uma sequência de caracteres mutável e segura para threads. Um buffer de string é como um String, mas pode ser modificado. A qualquer momento, ele contém uma sequência específica de caracteres, mas o comprimento e o conteúdo da sequência podem ser alterados por meio de determinadas chamadas de método.

StringBuilder

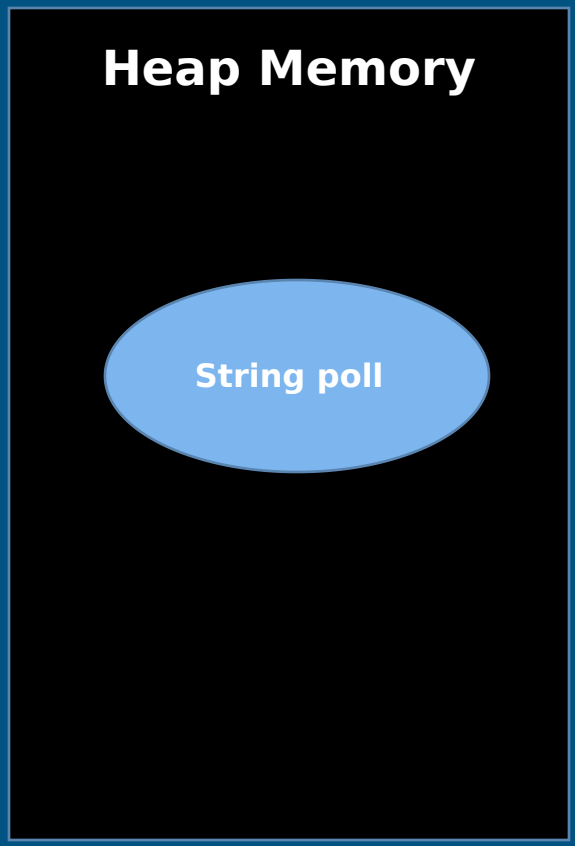
Uma sequência mutável de caracteres. Esta classe fornece uma API compatível com StringBuffer, mas sem garantia de sincronização. Essa classe foi projetada para ser usada como um substituto para o StringBuffer em locais onde o buffer da string estava sendo usado por um único encadeamento (como geralmente é o caso). Sempre que possível, é recomendável que essa classe seja usada preferencialmente ao StringBuffer, pois será mais rápida na maioria das implementações.

String Pool

Java String poll refere-se a uma coleções de String são armazenada em um Heap de Memoria

Os Objetos do tipo String são imutáveis na sua essência sendo o conceito visto dentro da figura ao lado.

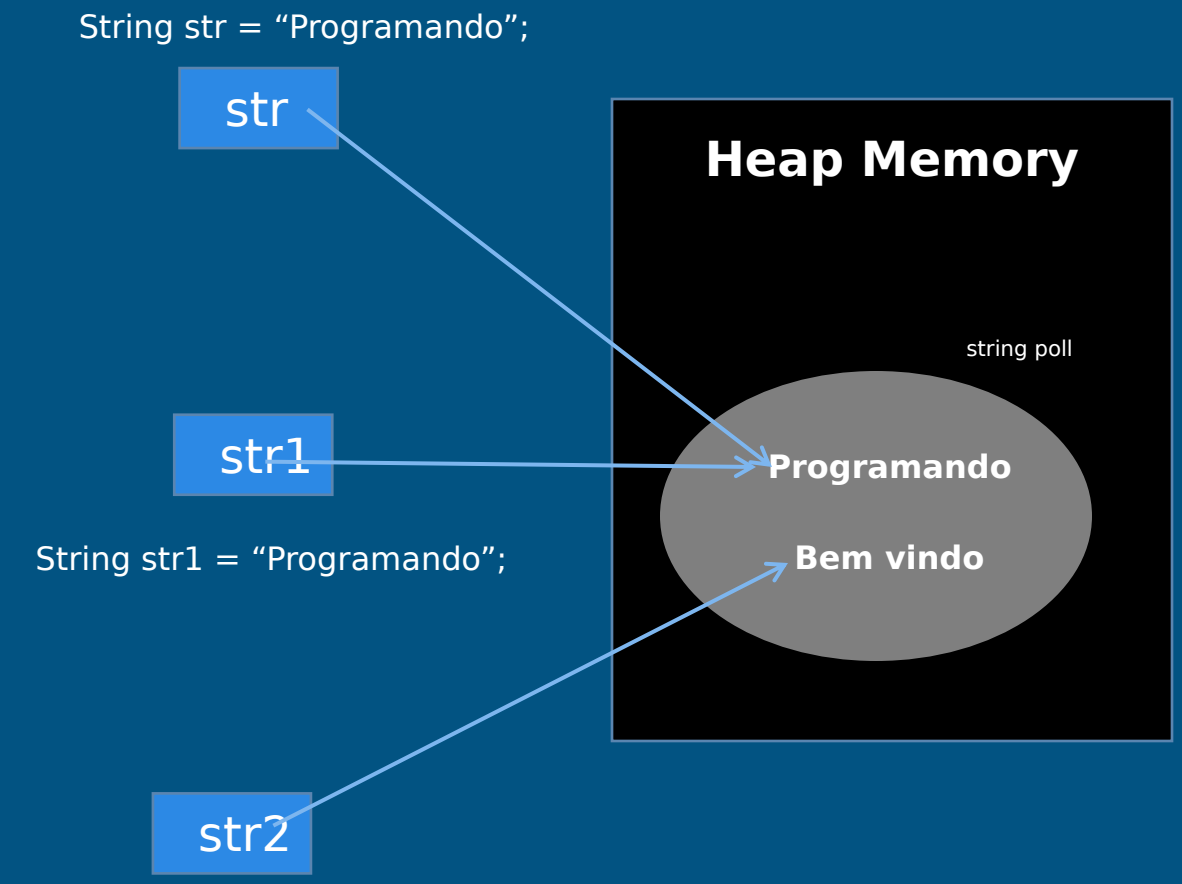
String Poll ajuda salvando o espaço em memoria para Java Runtime



Criando

String

literal

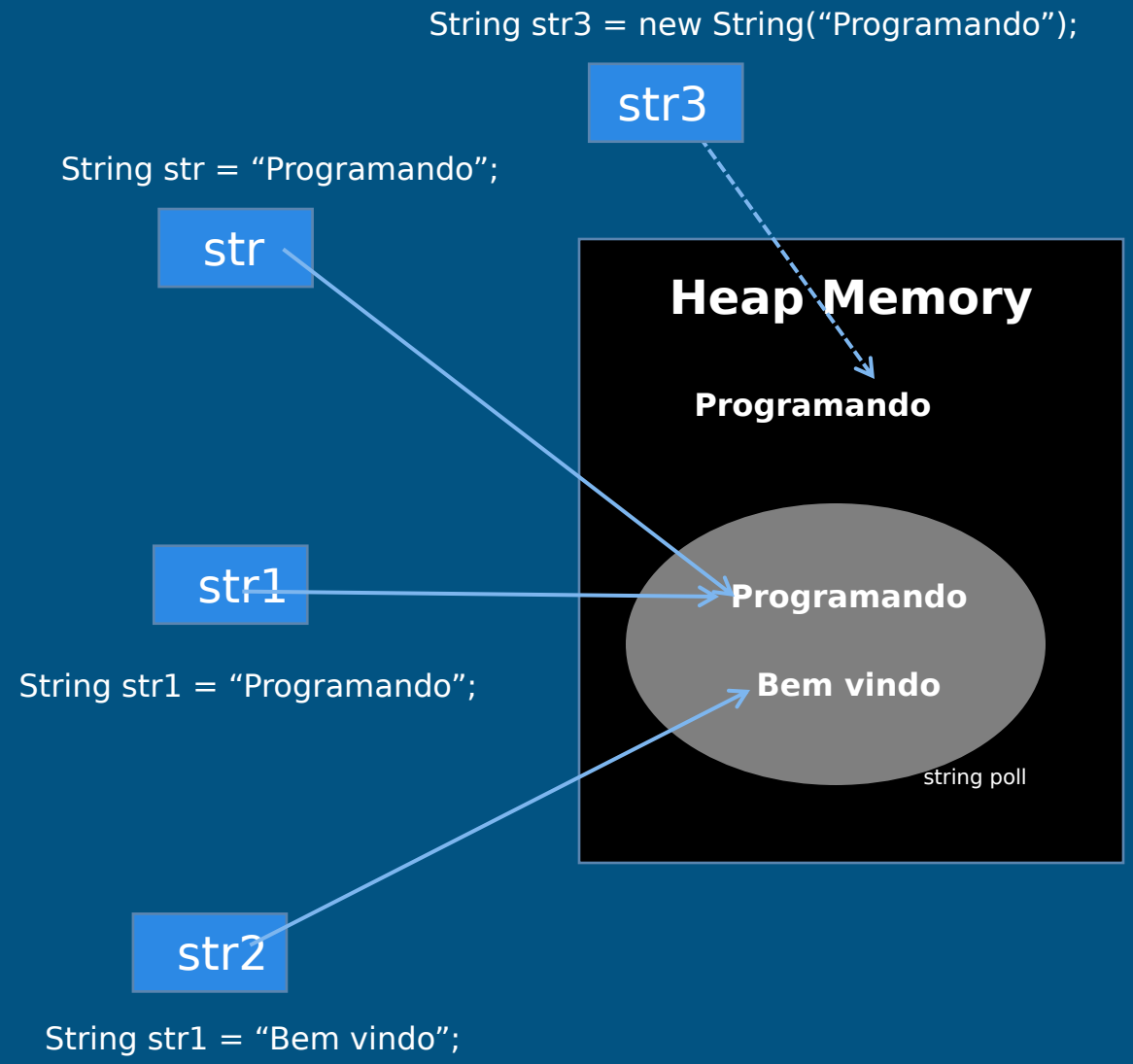


Antes de criar uma String literal e buscado uma String com o mesmo valor no poll de String se encontrar e retornado a referência do mesmo, caso não e criado uma nova String no poll e retornado essa referência

Criando

String

Keyword



String Métodos

Method	Description	Return
charAt()	Returns the character at the specified index (position)	char
codePointAt()	Returns the Unicode of the character at the specified index	int
codePointBefore()	Returns the Unicode of the character before the specified index	int
codePointCount()	Returns the Unicode in the specified text range of this String	int
compareTo()	Compares two strings lexicographically	int
compareToIgnoreCase()	Compares two strings lexicographically, ignoring case differences	int
concat()	Appends a string to the end of another string	boolean
contains()	Checks whether a string contains a sequence of characters	boolean
contentEquals()	Checks whether a string contains the exact same sequence of characters of the specified CharSequence or StringBuffer	boolean
copyValueOf()	Returns a String that represents the characters of the character array	String

Method	Description	Return
endsWith()	Checks whether a string ends with the specified character(s)	boolean
equals()	Compares two strings. Returns true if the strings are equal, and false if not	boolean
equalsIgnoreCase()	Compares two strings, ignoring case considerations	boolean
format()	Returns a formatted string using the specified locale, format string, and arguments	String
getBytes()	Encodes this String into a sequence of bytes using the named charset, storing the result into a new byte array	byte[]
getChars()	Copies characters from a string to an array of chars	void
hashCode()	Returns the hash code of a string	int
indexOf()	Returns the position of the first found occurrence of specified characters in a string	int
intern()	Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index	String
isEmpty()	Checks whether a string is empty or not	boolean

Method	Description	Return
lastIndexOf()	Returns the position of the last found occurrence of specified characters in a string	int
length()	Returns the length of a specified string	int
matches()	Searches a string for a match against a regular expression, and returns the matches	boolean
offsetByCodePoints()	Returns the index within this String that is offset from the given index by codePointOffset code points	int
regionMatches()	Tests if two string regions are equal	boolean
replace()	Searches a string for a specified value, and returns a new string where the specified values are replaced	String
replaceFirst()	Replaces the first occurrence of a substring that matches the given regular expression with the given replacement	String
replaceAll()	Replaces each substring of this string that matches the given regular expression with the given replacement	String
split()	Splits a string into an array of substrings	String []
istartsWith()	Checks whether a string starts with specified characters	boolean

Regex

Regex

\d{5}-\d{3}

O padrão de um CEP como 05432-001: 5 dígitos, um - (hífen) e mais 3 dígitos.
A sequência \d é um meta caractere, um curinga que casa com um dígito (0 a 9).
A sequência {5} é um quantificador: indica que o padrão precedente deve ser repetido 5 vezes, portanto \d{5} é o mesmo que \d\d\d\d\d.

[012]\d:[0-5]\d

Semelhante ao formato de horas e minutos, como 03:10 ou 23:59. A sequência entre colchetes [012] define um conjunto. Neste caso, o conjunto especifica que primeiro caractere deve ser 0, 1 ou 2.
Dentro dos [] o hífen indica uma faixa de caracteres, ou seja, [0-5] é uma forma abreviada para o conjunto [012345]; o conjunto que representa todos os dígitos, [0-9] é o mesmo que \d.
Note que esta expressão regular também aceita o texto 29:00 que não é uma hora válida

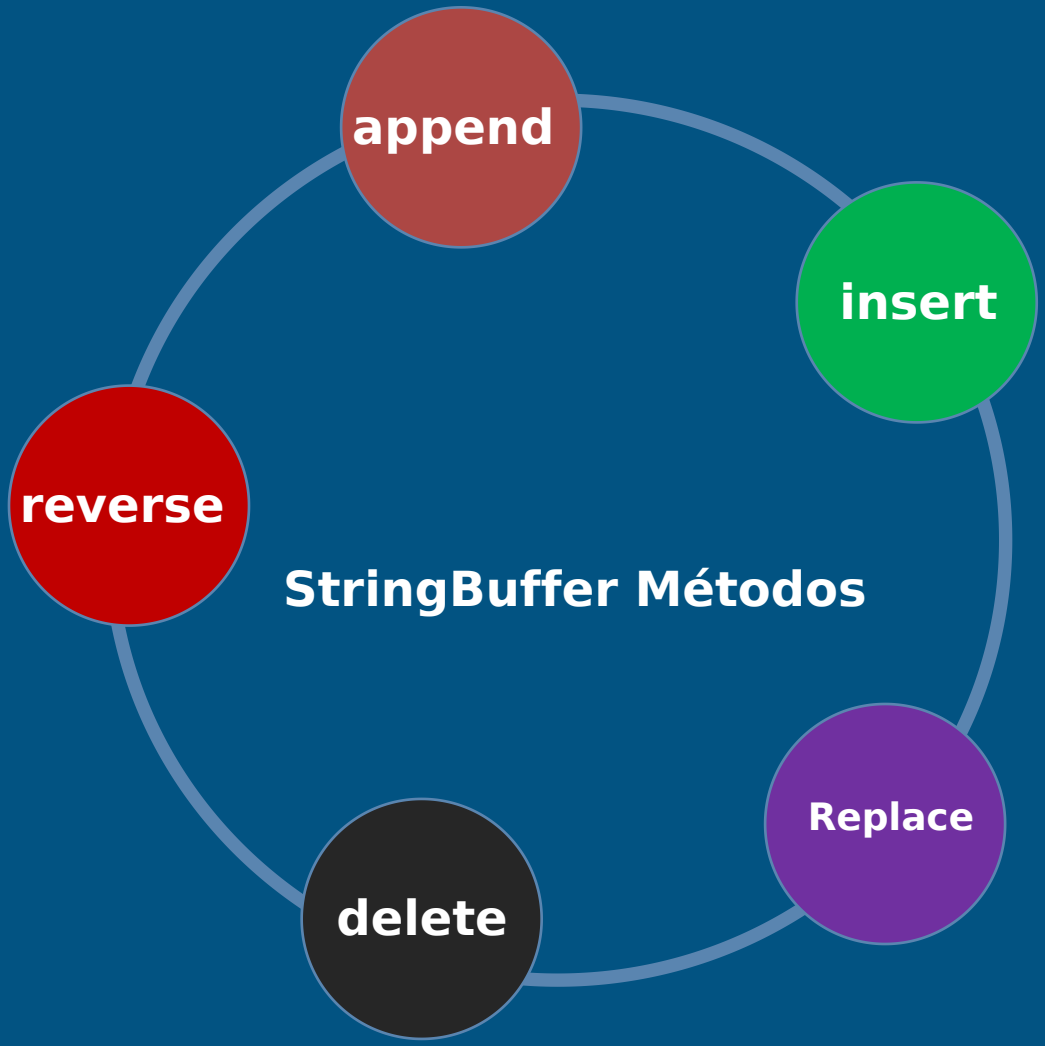
A-Z]{3}-\d{4}

É o padrão de uma placa de automóvel no Brasil: três letras de A a Z é seguidas de um - (hífen) s eguido de quatro dígitos, como CKD-4592.

teste: https://www.regexpal.com/

http://turing.com.br/material/regex/introducao.html acessado: 09/04/2020

Method	Description	Return
subSequence()	Returns a new character sequence that is a subsequence of this sequence	CharSequence
substring()	Extracts the characters from a string, beginning at a specified start position, and through the specified number of character	String
toCharArray()	Converts this string to a new character array	char[]
toLowerCase()	Converts a string to lower case letters	String
toString()	Returns the value of a String object	String
toUpperCase()	Converts a string to upper case letters	String
trim()	Removes whitespace from both ends of a string	String
valueOf()	Returns the primitive value of a String object	String



Java

StringBuffer

StringBuffer

Definição

StringBuffer é mutável significa que é possível alterar o valor do objeto.

O objeto criado por meio de StringBuffer é armazenado no heap.

StringBuffer possui os mesmos métodos que o StringBuilder, mas cada método no StringBuffer é sincronizado, ou seja, StringBuffer é seguro para threads.

Por esse motivo, ele não permite que dois threads acessem simultaneamente o mesmo método. Cada método pode ser acessado por um thread por vez.

Mas ser seguro para threads também tem desvantagens, pois o desempenho do StringBuffer ocorre devido à propriedade segura para threads. Portanto, StringBuilder é mais rápido que o StringBuffer ao chamar os mesmos métodos de cada classe.

StringBuffer Métodos

StringBuffer

Métodos

O StringBuffer e o StringBuilder têm os mesmos métodos (além da declaração do método sincronizado na classe StringBuilder).

Vamos ver alguns dos mais comuns:

- append()
- insert()
- replace()
- delete()
- reverse()

StringBuffer

Métodos

Uso

StringBuffer sb1 = new StringBuffer("Buffer no 1");

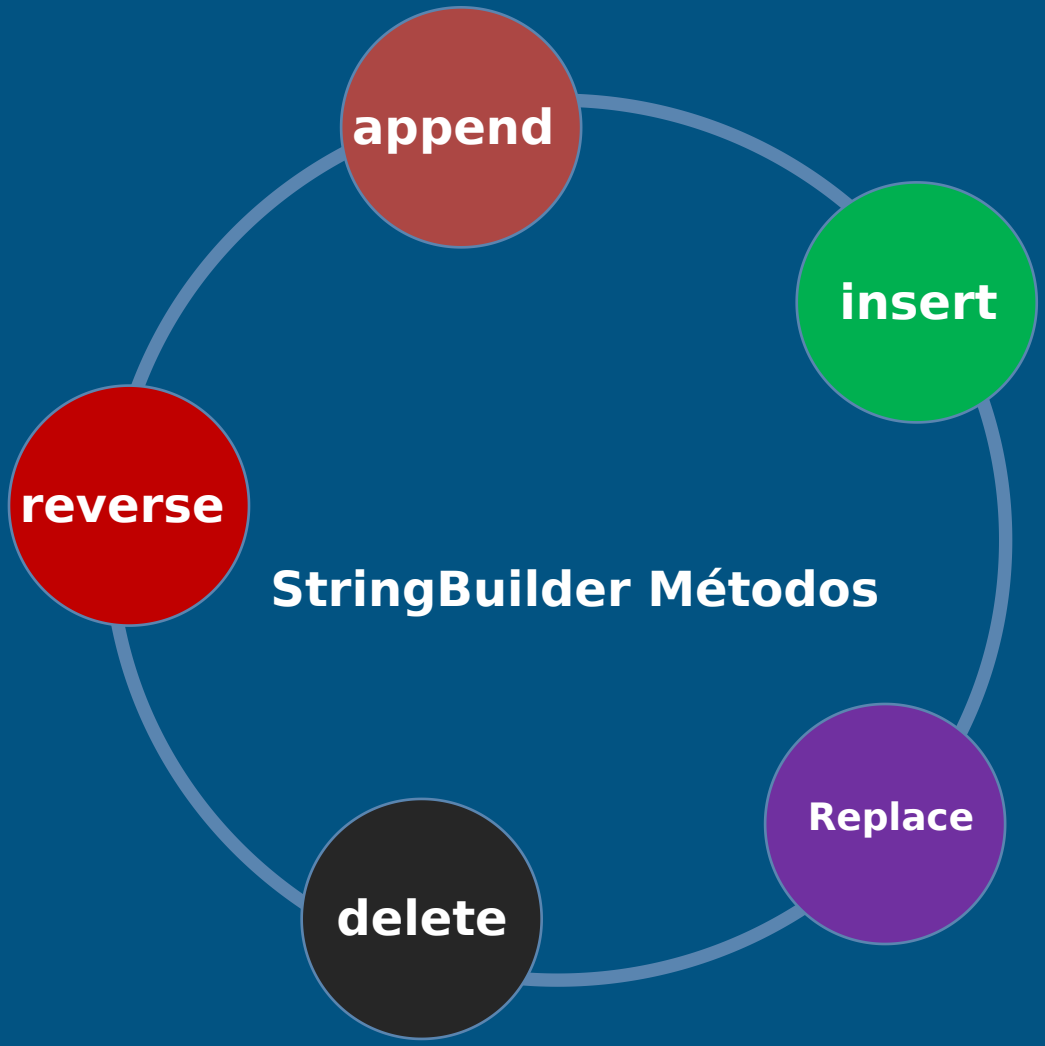
append()
System.out.println(sb1);
sb1.append(" - and this is appended!");
System.out.println(sb1);

insert()
sb1.insert(11, ", this is inserted");
System.out.println(sb1);

replace()
sb1.replace(7, 9, "Number");
System.out.println(sb1);

delete()
sb1.delete(7, 14);
System.out.println(sb1);

reverse()
sb1.reverse();
System.out.println(sb1);



StringBuilder

StringBuilder

StringBuilder é o mesmo que StringBuffer, ou seja, armazena o objeto na pilha e também pode ser modificado.

A principal diferença entre o StringBuffer e o StringBuilder é que o StringBuilder não é seguro para threads.

O StringBuilder é rápido, pois não é seguro para threads.

Definição

O uso do StringBuilder resultou em um tempo ~ 6000 vezes mais rápido que o String normal.

O que levaria o StringBuilder a concatenar em 1 segundo levaria o String 1.6 horas (se pudéssemos concatenar isso).

Atividades String

1. Faça um programa que, a partir de uma string digitada pelo usuário, imprima:
a)O número de caracteres da string.
b)A string com todas suas letras em maiúsculo.
c)O número de vogais da string.
d)Se a string digitada começa com “GOMES” (ignorando maiúsculas/minúsculas).
e)Se a string digitada termina com “SOUZA” (ignorando maiúsculas/minúsculas).
f)O número de dígitos (0 a 9) da string.
g)Se a string é um palíndromo ou não.

2.Escreva um programa que dado um valor numérico digitadopelo usuário (armazenado em uma variável inteira),
imprima cada um dos seus dígitos por extenso.Exemplo:
Entre o número: 1457
Resultado: um, quatro, cinco, sete

Atividades String

3. Escreva um programa que, a partir deum nome informado pelo usuário, exiba suas iniciais.

As iniciais são formadas pela primeira letra de cada nome, sendo que todas deverão aparecer em maiúsculas na saída do programa.
Note que os conectores e, do, da, dos, das, de, di, du não são considerados nomes e, portanto, não devem ser considerados para a obtenção das iniciais.
As iniciais devem ser impressas em maiúsculas, ainda que o nome seja entrado todo em minúsculas.
Exemplos:

Duana Barbosa Silva=> DBS
Amanda Reis => AR
Vitor da Silva => VS
Ana Carolina Pio => ACP

4.Faça um programa que, a partir de um texto digitadopelo usuário, conte o número de caracteres total e o número de palavras (palavra é definida por qualquer sequência de caracteres delimitada por espaços em branco) e exiba o resultado

Atividades String

5. Faça um programa que, a partir de um texto digitado pelo usuário, imprima o texto todos os espaços em branco adicionais encontrados, de modo que haja, no máximo, um espaço em branco separando as palavras presentes nesse texto.

6. [LOGIN E SENHA]

Escreva um programa que receba um login e uma senha de um usuário. O programa deverá checar se os dados digitados pelo usuário são iguais aos dados internos do programa, que são:

Login = Admin
Senha = Admin

Se os dados estiverem corretos, o programa deverá imprimir a frase BEM VINDO, caso contrário, deverá ser impressa Login e/ou Senha incorretos!.