JAVA

LINGUAGEM DE PROGRAMAÇÃO

Tipo de dados Strings

LINGUAGEM DE PROGRAMAÇÃO JAVA

String

A classe **String** inclui métodos para examinar caracteres individuais da sequência, comparar strings, pesquisar strings, extrair substrings e criar uma cópia de uma string com todos os caracteres traduzidos para letras maiúsculas ou minúsculas.

String

A classe **String** representa strings de caracteres. Todos os literais de string em programas Java, como "abc", são implementados como instâncias dessa classe.

- Java String literal é criado usando aspas duplas.

```
String linguagem = “java”;
```

String

Mas é possível também criar objetos Strings utilizando a palavra-chave new .

```
String linguagem = new String (“Java”);
```

String

Um objeto **String** é imutável, o que significa que o texto que ele carrega nunca é alterado.

Sempre que um texto precisa ser modificado é utilizado mais espaço em memória para que uma nova String seja criada contendo a nova versão dele.

String

Por que a imutabilidade de **String** é um recurso de design em Java:

- Segurança: Objetos imutáveis são inerentemente thread-safe, pois não podem ser modificados após a criação. Essa propriedade elimina problemas de sincronização em aplicativos multithread, tornando a manipulação de strings mais segura. Strings mutáveis podem apresentar riscos de segurança e criar comportamentos imprevisíveis de segurança da JVM.

String

- **Otimização de desempenho por meio de agrupamento de strings**: como as strings são imutáveis, o Java pode armazená-las em cache em uma região de memória especial chamada String Pool. Se a mesma string literal for usada em outro lugar no programa, ambas as referências apontarão para o mesmo objeto no pool. Isso economiza memória e aumenta o desempenho

String

- **Integridade e confiabilidade**: A imutabilidade garante que uma vez que um objeto string seja criado, ele não será alterado por nenhuma parte do código, intencionalmente ou não. Esse comportamento torna o código mais previsível e mantém a integridade dos dados.

Principais Métodos da Classe String

LINGUAGEM DE PROGRAMAÇÃO JAVA

String - Concatenação

Existem duas formas de unir duas ou mais sequências de caracteres:

- A mais comum dentre elas é utilizando o operador de adição (+)

```
String nomeCompleto = nome + sobrenome;
```

String - concat()

concat() -O método concat() acrescenta uma string ao final de outra.

```
String first = "Linguagem";
String second = "JAVA";
String full = first.concat(second);
System.out.println(full);
// Output: LinguagemJAVA
```

String - length()

Length() - Este método retorna o comprimento da string, ou seja, o número de caracteres que contém nela.

```
String name = "Java";
int len = name.length();
System.out.println("Length: " + len);
// Output: Length: 4
```

String - equals(valor)

O método equals() verifica se duas strings são iguais.

```
String one = "JAVA";
String two = "Programacao";
boolean isEqual = one.equals(two);
// false
```

String - toLowerCase() e toUpperCase():

Esses métodos convertem a string em minúsculas e maiúsculas, respectivamente.

toLowerCase = minúsculas

toUpperCase = maiúsculas

```
String aulajava = "Aula Java";
System.out.println(aulajava.toLowerCase());
// Output: aula java
System.out.println(aulajava.toUpperCase());
// Output: AULA JAVA
```

String - trim()

trim() - Gera um novo objeto string, removendo todos os caracteres de espaço em branco que aparecem **no início ou no fim** da string.

```
String name = "    Java      ";
System.out.println(name.trim());
// Output: Java
```

String - replace(ant, new)

replace() - Retorna um novo objeto contendo a string original com um trecho especificado substituído por outra expressão indicada. Esse método deixa a string original inalterada.

```
String nome = "aula java";
String nomeAlterado = nome.replace('a', 'A');
System.out.println(nomeAlterado);
//AulA jAvA
```

String - valueOf()

valueOf() - é um método estático da classe String, que não precisa de uma instância para ser invocado. Ele converte um tipo primitivo em um objeto do tipo String.

```
double numero = 102939939.939;
boolean booleano = true;
System.out.println("Valor: " + String.valueOf(numero));
System.out.println("Valor: " + String.valueOf(booleano));
```

String – substring()

substring() - Permite extrair substrings de strings e fornece 2 métodos substring para permitir que um novo objeto seja criado copiando parte de um objeto string existente.
Veremos a seguir as duas possibilidades:

STRING SUBSTRING (início)

Será passado por parâmetro um tipo inteiro, que especifica a partir de que caractere a cópia deve começar.

A substring retornada contém uma cópia dos caracteres desde esse índice até o último caractere na string.

```
String nome = "Java";
String substring = nome.substring(2);
System.out.println(substring);
// va
```

STRING SUBSTRING (início, fim)

Será enviado dois argumentos do tipo inteiro. A String retornada será composta pelo o primeiro argumento e se estende até o caractere anterior ao segundo argumento. Em outras palavras, você pode dizer que o primeiro argumento é inclusivo e o segundo argumento é exclusivo

```
String nome = "Java";
String substring = nome.substring(1,3);
System.out.println(substring);
// av
```

Boas Práticas Com Strings

- Use literais de string onde possível: As strings literais são gerenciadas no String Pool, permitindo o uso eficiente da memória. Compare usando equals() em vez de ==;
- Evite Concatenação em Loops: A concatenação de strings dentro de um loop pode levar a problemas de desempenho devido à criação de vários objetos. Utilize método concat();
- Use a formatação de string: Para criação de strings complexas, o String.format fornece uma abordagem mais legível;

```
String.format("Order %d: %s", orderId, orderName);
```

Boas Práticas Com Strings

- **Utilize métodos de string para limpeza e análise:** Faça uso de métodos como trim(), split(), toLowerCase(), toUpperCase(), etc., para manipular strings sem reinventar a roda.
- **Verificar sempre se está vazia:** Prefira isEmpty() do que length() Verifique se há Vazio na string

```
if (str.isEmpty())
```

```
true
```

- **Tenha cuidado com a codificação de caracteres:** Se estiver trabalhando com codificações de caracteres diferentes, fique atento à codificação usada, especialmente ao ler ou gravar strings;

Documentação de Strings

Utilize a documentação da Classe String para uso dos métodos:

https://docs.oracle.com/en/java/javase/20/docs/api/java.base/java/lang/String.html

Obrigado!

INTRODUÇÃO A LINGUAGEM DE PROGRAMAÇÃO JAVA