ORACLE

Academy

Java Foundations

4-3

A Classe String

ORACLE
Academy

Objetivos

- Esta lição abrange os seguintes objetivos:
  - Localizar a classe String na documentação da API Java
  - Entender os métodos da classe String
  - Comparar dois objetos String utilizando léxico
  - Encontrar a localização de uma substring em um objeto String
  - Extrair uma substring de um objeto String

Tópicos

- **Introdução a Strings**
- Documentação da Classe String
- Métodos da Classe String
- Concatenando Strings
- Comparando Strings

ORACLE Academy

JFo 4-3
A Classe String

O que É uma String?

- Uma string é uma sequência de caracteres que inclui letras do alfabeto, caracteres especiais e espaço em branco
- Por exemplo:
  - "Como você está?" é uma string que contém letras, espaço em branco e um caractere especial ('?')
- Em Java, as strings não são um tipo de dados primitivo
- Em vez disso, elas são objetos da classe String

Representando Strings em Java

- Em Java, as strings são objetos da classe denominada java.lang.String
- Exemplo:
  - String s1= “Hello, World”;

Representando Strings em Java

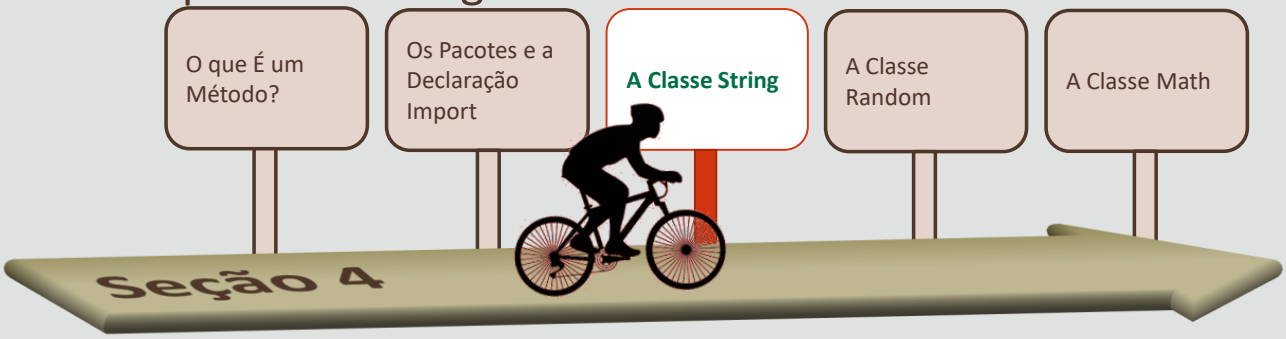
- Uma string em Java é mais abstrata
- Ou seja, você não precisa conhecer sua estrutura interna, o que facilita seu uso
- Seus métodos permitem que um programador execute operações nela

Usando a Classe String

- A classe String:
  - É uma das muitas classes incluídas nas bibliotecas de classes Java
  - É parte do java.lang.package
  - Permite a você manter uma sequência de caracteres de dados
- Você usará a classe String frequentemente em todos os seus programas
- Portanto, é importante entender algumas das características especiais das strings em Java

Tópicos

- Introdução a Strings
- **Documentação da Classe String**
- Métodos da Classe String
- Concatenando Strings
- Comparando Strings

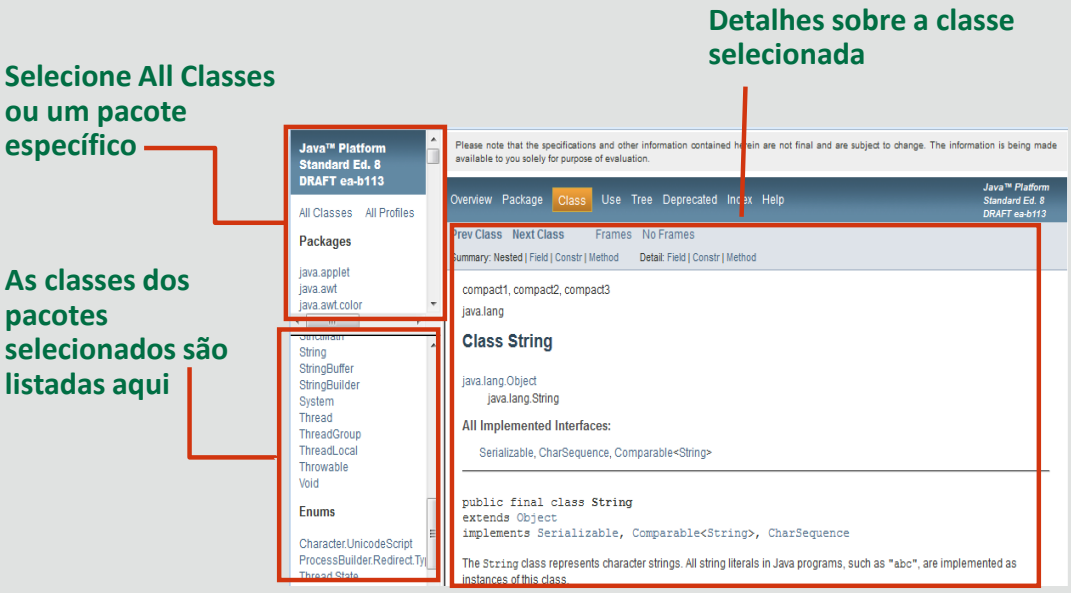


ORACLE Academy

Documentação da Classe String

- Você pode acessar a documentação da classe Java String em:
  - https://docs.oracle.com/javase/8/docs/api/

Documentação do Java Platform SE 8 da Classe String



ORACLE Academy

JFo 4-3
A Classe String

No screenshot, você pode ver os três painéis principais da página da Web.

O painel superior esquerdo permite que você selecione um pacote. As classes Java estão organizadas em pacotes, mas, se você não conhecer o pacote de uma classe específica, poderá selecionar Todas as Classes.

O painel inferior esquerdo fornece uma lista de classes em um pacote ou todas as classes que você selecionou. Neste painel, a classe String foi selecionada, e o painel principal à direta está preenchido com os detalhes da classe String. O painel principal contém muitas informações sobre a classe. Por isso, você precisa rolar para baixo para acessar as informações.

String: Resumo do Método

- `public int charAt(String str)`

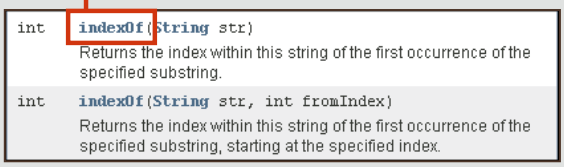
ORACLE Academy

Se você continuar rolando pelos detalhes da classe String, verá a lista de métodos (só um subconjunto pequeno dessa lista é mostrado aqui).

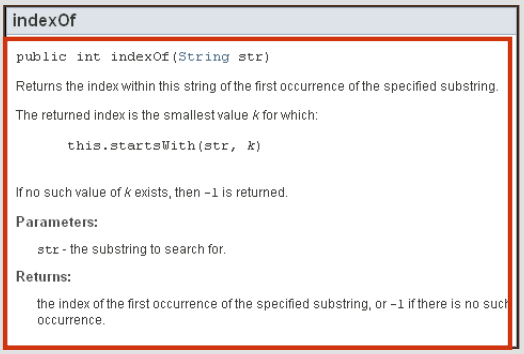
Essa lista mestre de métodos fornece os detalhes básicos do método. Nesse caso, você pode ver que o nome do método é charAt, seu tipo é char e isso requer que um parâmetro de índice (do tipo int) seja passado. Também existe uma descrição resumida que esse método retorna, o valor char em um índice específico na string. Para cada método, o nome do método e os tipos de parâmetros são vinculados como hiperlinks de modo que você possa obter mais detalhes.

String: Detalhe do Método

Clique aqui para obter uma descrição detalhada do método



Uma descrição detalhada do método indexOf()



Informações adicionais sobre parâmetros e o valor de retorno são mostradas na lista de métodos

ORACLE Academy

JFo 4-3
A Classe String

Para cada método, o nome do método e os tipos de parâmetros são vinculados como hiperlinks de modo que você possa obter mais detalhes. O exemplo aqui mostra uma descrição detalhada de um dos métodos `indexOf()` de `String`.

Tópicos

- Introdução a Strings
- Documentação da Classe String
- **Métodos da Classe String**
- Concatenando Strings
- Comparando Strings

ORACLE Academy

Métodos de String: length

- Você pode calcular o comprimento de uma string usando o método length definido na classe String:
- Método: name.length()
- Retorna o comprimento ou o número de caracteres no nome como um valor inteiro
- Exemplo:

```
String name = "Mike.W";
System.out.println(name.length()); //6
```

Acessando Cada Caractere em uma String

- Você pode acessar cada caractere em uma string por seu índice numérico
- O primeiro caractere da string está no índice 0, o seguinte está no índice 1 e assim por diante
- Por exemplo:
- `String str= "Hello, World";`

H	e	l	l	o	,		W	o	r	l	d
0	1	2	3	4	5	6	7	8	9	10	11

  - str tem de 0 a 11 índices; ou seja, entre 0 a str.length()-1

Métodos de String: indexOf()

- Cada caractere de uma string tem um índice
- Você pode recuperar o valor do índice de um caractere na string usando o método indexOf:

Método	Descrição
`str.indexOf(char c)`	Retorna o valor do índice da primeira ocorrência de c na String str
`s1.indexOf(char c, int beginIdx)`	Retorna o valor do índice da primeira ocorrência de c em String s1, começando em beginIdx até o fim da string

Métodos de String: indexOf()

```
public static void main(String args[]){
   String phoneNum = "404-543-2345";
   int idx1 = phoneNum.indexOf('-’);
   System.out.println("índice do primeiro hífen: "+ idx1); //3
   int idx2 = phoneNum.indexOf('-', 4);
   System.out.println("índice do segundo hífen: "+ idx2); // 7
}//fim do método main
```

Métodos de String: charAt

- Retorna o caractere da string localizada no índice passado como o parâmetro
- Método: str.charAt(int index)

```
String str = "Susan";
System.out.println(str.charAt(0));  //S
System.out.println(str.charAt(3));  //a
```

Métodos de String: substring()

- Você pode extrair uma substring de determinada string
- O Java fornece dois métodos para essa operação:

Método	Descrição
`str.substring(int beginIdx)`	Retorna a substring de beginIdx até o fim da string
`str.substring(int beginIdx, int endIdx)`	Retorna a substring de beginIdx até, mas não inclusive, endIdx

Métodos de String: substring()

```
public static void main(String args[]){
    String greeting = "Hello, World!";
    String sub = greeting.substring(0, 5); → "Hello"
    String w = greeting.substring(7, 11); → "Worl"
    String tail = greeting.substring(7); → "World!"
}//fim do método main
```

Métodos de String: replace()

- Este método substitui todas as ocorrências dos caracteres correspondentes em uma string
- Método: replace(char oldChar,char newChar)
- Exemplo:

```
public static void main(String args[]) {
String str = "Usando a String replace para substituir "
             + "caractere";
    String newString = str.replace("r", "R");
    System.out.println(newString);
}//fim do método main
```

  - Saída: Usando a String Replace para Substituir CaRacteRe
  - Todas as ocorrências de um "r" minúsculo são substituídas por um "R" maiúsculo

Métodos de String: replaceFirst()

- Este método só substitui a 1a. ocorrência do padrão de caracteres correspondentes em uma string
- Método: replaceFirst(String pattern, String replacement)

Métodos de String: replaceFirst()

- Exemplo:

```
public static void main(String args[]) {
    String replace = "String replace with replaceFirst";
    String newString = replace.replaceFirst("re", "RE");
    System.out.println(newString);
}//fim do método main
```

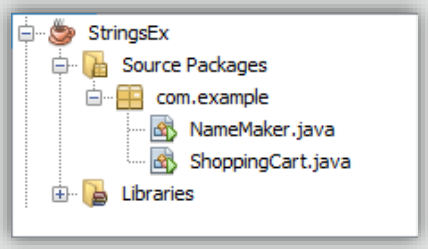
- Saída:
  - String REplace com replaceFirst
- Só a primeira ocorrência de "re" é substituída por "RE“
- A segunda ocorrência não é alterada

Exercício 1, Parte 1

- Importe e abra o projeto `StringsEx`
- Examine `ShoppingCart.java`
  - Faça o seguinte:
  - Use o método indexOf para obter o índice do caractere de espaço (" ") dentro de custName
  - Atribua-o a spaceIdx
  - Use o método da substring e spaceIdx para obter a parte do primeiro nome de custName
  - Atribua-o a firstName e imprima firstName

Exercício 1, Parte 2

- Você deve ter percebido que esse projeto tem dois arquivos
- Java com métodos main
  - Isso pode parecer uma contradição porque orientamos a nunca usar mais de um método main

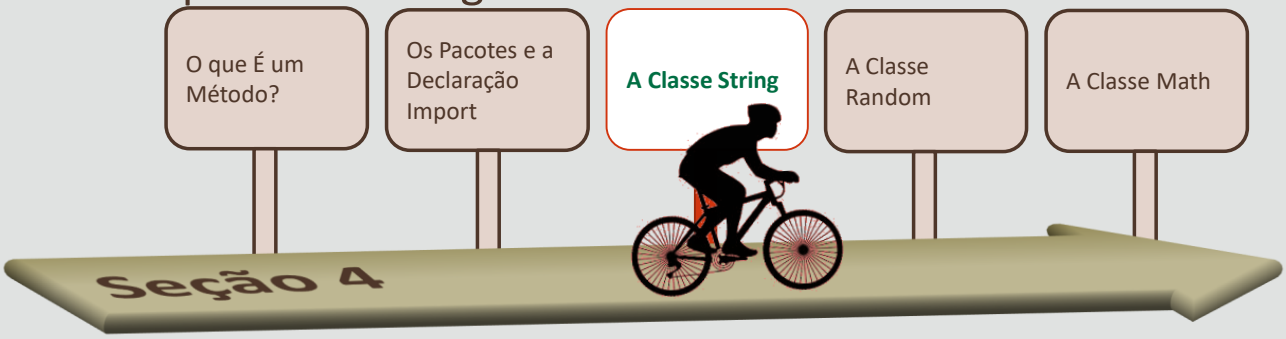


- Às vezes, os programadores fazem isso quando estão testando bits pequenos de código e desejam manter todos os arquivos organizados em um projeto
  - Infelizmente, pressionar run no NetBeans sempre executa o mesmo arquivo, e nunca os outros
  - Você precisará clicar com o botão direito do mouse no outro arquivo que deseja executar
  - Aparecerá um menu com uma opção para executar esse arquivo

Outra opção seria ativar o botão **Executar Arquivo** no NetBeans (ou pressionar Shift+F6). Se essa inconsistência for ruim, avise-nos e tentaremos corrigi-la na próxima versão do curso.

Tópicos

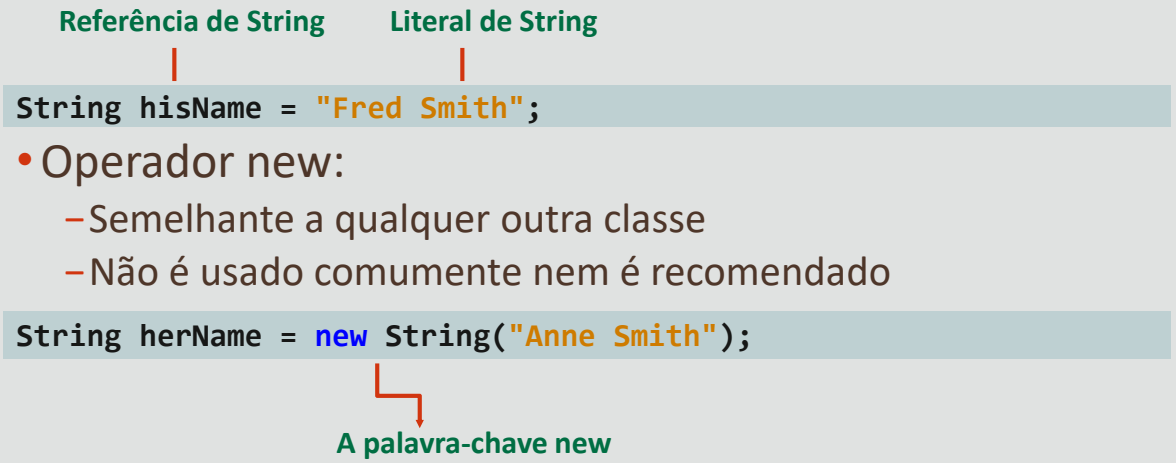
- Introdução a Strings
- Documentação da Classe String
- Métodos da Classe String
- **Concatenando Strings**
- Comparando Strings



ORACLE Academy

Declarando e Criando uma String

- Você pode instanciar strings de duas maneiras:
- Literais de string: Atribua diretamente uma literal de string a uma referência de string



- Operador new:
  - Semelhante a qualquer outra classe
  - Não é usado comumente nem é recomendado

ORACLE Academy

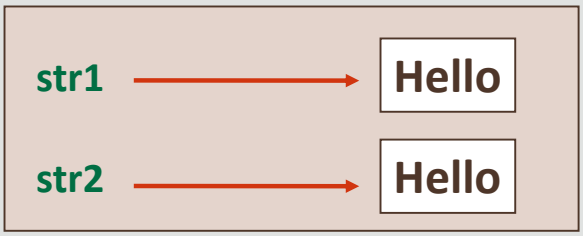
Embora você possa usar o operador `new` para criar uma string, não o utilize. Você entenderá por que mais adiante neste curso.

As Strings São Imutáveis

- Um objeto String é imutável; ou seja, depois que um objeto String é criado, seu valor não pode ser alterado
- Como as strings são imutáveis, o Java pode processá-las de maneira muito eficiente
  - Considere o seguinte:

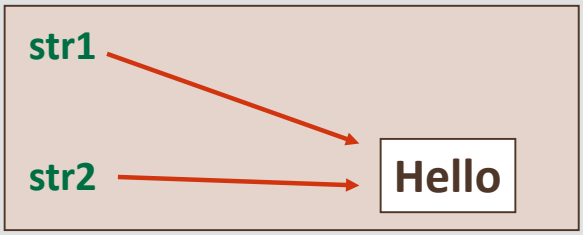
```
String str1 = "Hello";
String str2 = "Hello";
```

  - Esperamos isso...



As Strings São Imutáveis

- Mas isso é o que acontece…



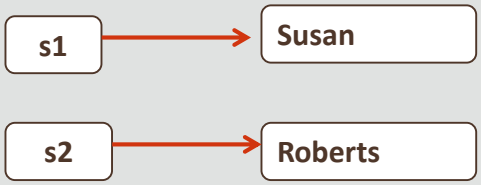
- O sistema de run-time Java sabe que as duas strings são idênticas e aloca o mesmo local de memória para os dois objetos

Concatenando Strings

- No Java, a concatenação de strings forma uma nova string que é a combinação de várias strings
- Você pode concatenar strings em Java de duas maneiras:
  - operador de concatenação de strings **+**
  - método **concat()**

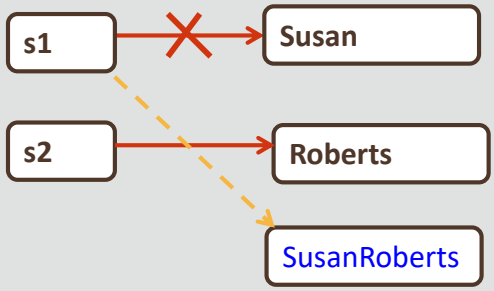
Usando o Operador + (Antes da Concatenação)

```
public static void main(String args[]) {
    String s1 = "Susan";
    String s2 = "Roberts";
}//fim do método main
```



Usando o Operador + (Depois da Concatenação)

```
public static void main(String args[]) {
    String s1 = "Susan";
    String s2 = "Roberts";
    S1 = s1 + s2;
    System.out.println(s1);
}//fim do método main
```



ORACLE Academy

Depois da operação de concatenação das strings, um novo objeto `String`, "SusanRoberts,", é criado e `s1` aponta para ele devido à propriedade imutável de Strings. Como não há referências à string, Susan é removida da memória.

Concatenando Dados Não String com String

- Se um dos operandos for uma string, o Java converterá os tipos de dados não string automaticamente em strings antes da concatenação
- Exemplo:

```
public static void main(String args[]) {
    String newString = "Aprendendo Java" + 8;
    System.out.println(newString); //Aprendendo Java 8

    String numString = 8 + 8;
    System.out.println(numString); //16

    String numString1 = "8" + 8;
    System.out.println(newString1); //88
}//fim do método main
```

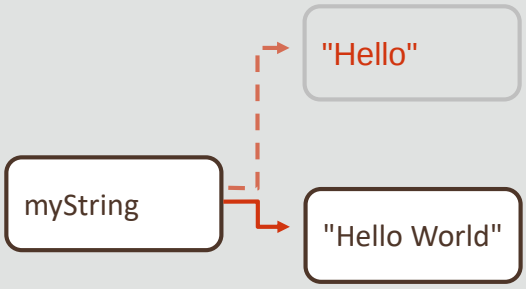

O exemplo do slide demonstra como concatenar uma literal de `String` e um número inteiro usando o operador +. O Java converte automaticamente tipos de dados não String em strings antes da concatenação.

Saída:

Aprendendo Java 8

Usando o Método concat() (Antes da Concatenação)

```
String myString = "Hello";
myString = myString.concat(" World");
```



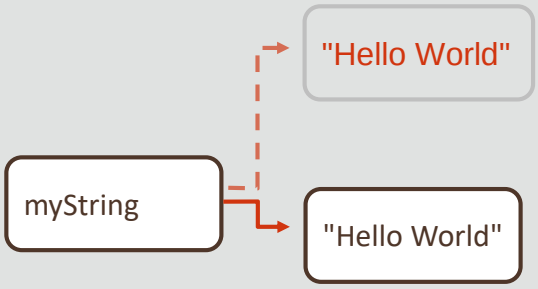
ORACLE Academy

Neste exemplo, a string "`World`" está sendo concatenada com a string original. O método `concat` é usado aqui, mas independentemente de você usá-lo ou de utilizar o operador de concatenação (+), um novo objeto `String` é criado, e a nova referência `String` aponta para esse novo objeto.

No diagrama, a referência `myString String` não se refere mais a "`Hello`" e será removida da memória.

Usando o Método concat() (Após a Concatenação)

```
String myString = "Hello";
myString = myString.concat(" World");
myString = myString + "!"
```



ORACLE Academy

Neste exemplo, depois de o método `concat` ser chamado, um novo objeto (HelloWorld) é criado e a referência a ele é atribuída a `myString`.

Por fim, ao concatenar outra string, dessa vez usando o operador de concatenação, a mesma coisa acontece novamente. É criado um novo objeto (HelloWorld!), e a referência a esse objeto é atribuída a `myString`.

Exercício 2

- Importe e abra o projeto `StringsEx`
- Examine `NameMaker.java`
- Faça o seguinte:
  - Declare variáveis de String: firstName, middleName, lastName e fullName
  - Solicite que usuários insiram os respectivos nomes, nomes do meio e sobrenomes e leiam os nomes no teclado
  - Defina e exiba fullName como firstName+a espaço em branco char+middleName+a espaço em branco char+lastName

Exercício 2

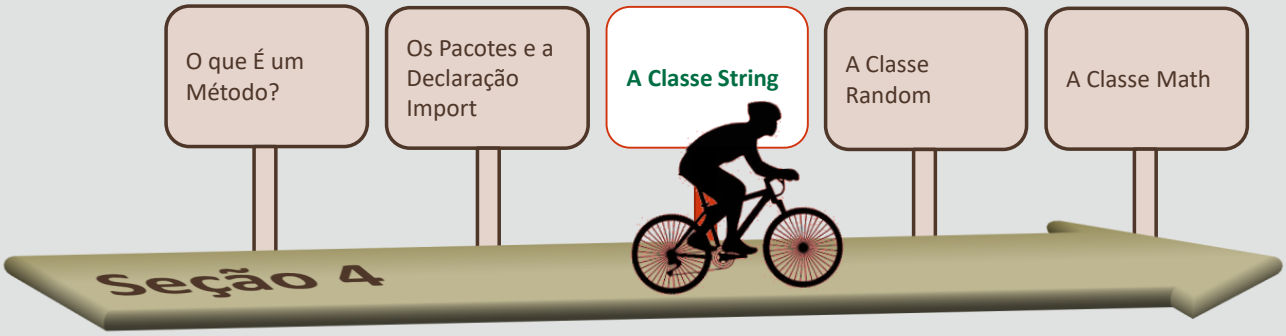
- O que você acha que é preferível para este cenário?
- Ou seja, o operador de concatenação de string ou o método concat()?

Qual é a Maneira Preferida para Concatenar Strings?

- Como você observou no exercício anterior:
- operador **+**:
  - Pode funcionar entre uma string e uma string ou um valor de tipo de dados char, int, double ou float
  - Converte o valor em sua representação de string antes da concatenação
- Método **concat()**:
  - Só pode ser chamado em strings
  - Verifica a compatibilidade dos tipos de dados
  - Será produzido um erro de tempo de compilação se não houver compatibilidade

Tópicos

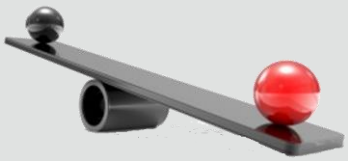
- Introdução a Strings
- Documentação da Classe String
- Métodos da Classe String
- Concatenando Strings
- **Comparando Strings**



ORACLE Academy

Como Você Compara Objetos String?

- Você pode comparar dois objetos String usando o método compareTo
- Esse método compara com base na ordem lexicográfica das strings
- As comparações lexicográficas são semelhantes à ordenação encontrada em um dicionário
- As strings são comparadas caractere por caractere até sua ordem ser determinada ou até provarem ser idênticas
- Sintaxe: s1.compareTo(s2)
- Retorna um valor inteiro que indica a ordem das duas strings



ORACLE Academy

As strings são comparadas lexicograficamente, e não alfabeticamente. As comparações lexicográficas são semelhantes à ordenação encontrada em um dicionário.

Valor Retornado por compareTo()

- O valor inteiro retornado pelo método compareTo() pode ser interpretado da seguinte maneira:
  - Retorna < 0 quando a string que está chamando o método é a primeira lexicograficamente
  - Retorna == 0 quando as duas strings são lexicograficamente equivalentes
  - Retorna > 0 quando o parâmetro passado para o método é o primeiro lexicograficamente

Usando o Método compareTo

- Vamos analisar alguns exemplos:
- "computer".compareTo("comparison")
  - Retorna um valor inteiro > 0 porque o parâmetro "comparison" é o primeiro lexicograficamente
- "cab".compareTo("car")
  - Retorna um valor inteiro < 0 porque a string “cab” que está chamando o método é a primeira lexicograficamente
- "car".compareTo("car")
  - Retorna um valor inteiro igual a 0 porque ambos são lexicograficamente equivalentes

Usando o Método compareTo: Exemplo

- Vamos escrever um programa para comparar nomes usando o método compareTo:

```
public static void main(String[] args) {

    String s1 = "Susan";
    String s2 = "Susan";
    String s3 = "Robert";

    //Retorna 0 porque s1 é idêntico a s2
    System.out.println(s1.compareTo(s2)); //Output is 0

    //Retorna > 0 porque 'S' vem depois de 'R'
    System.out.println(s1.compareTo(s3)); // Output is 1

    //Retorna < 0 porque 'R' vem antes de 'S'
    System.out.println(s3.compareTo(s1)); // Output is -1
}//fim do método main
```

Resumo

- Nesta lição, você deverá ter aprendido a:
  - Localizar a classe String na documentação da API Java
  - Entender os métodos da classe String
  - Comparar dois objetos String lexicograficamente
  - Encontrar a localização de uma substring em um objeto String
  - Extrair uma substring de um objeto String


ORACLE
Academy