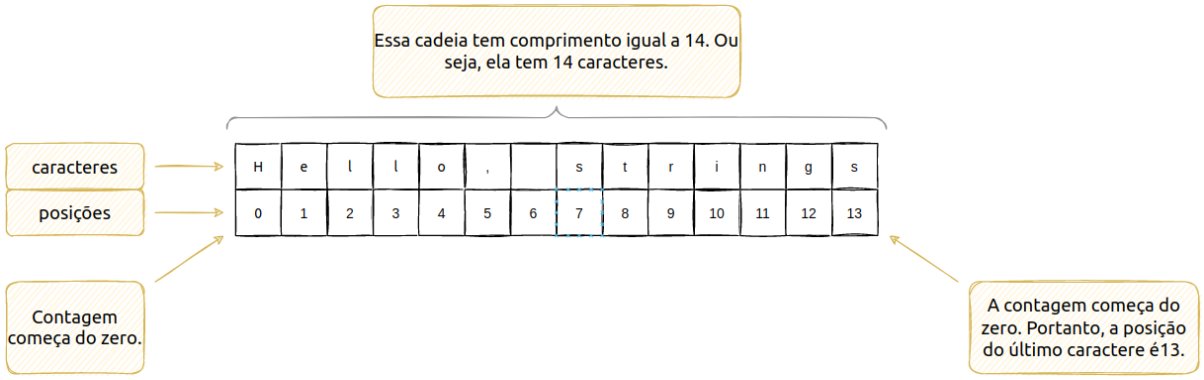
1 Introdução

Strings são sequências de caracteres. É comum utilizarmos as expressões “string”, “sequência de caracteres” e “cadeia de caracteres” como sinônimos. Em Java, Strings são objetos que manipulamos utilizando variáveis de referência. É importante observar que strings são **imutáveis**. Neste material, estudaremos sobre algumas das principais características das strings em Java, por meio da resolução de exercícios.

2 Desenvolvimento

Cada caractere pertencente a uma string possui uma “**posição**” dentro da string. As posições começam do número zero, como mostra a Figura 2.1. Também é comum chamarmos a posição de um caractere de **índice**.

Figura 2.1



É muito importante que o desenvolvedor Java conheça a documentação da plataforma, denominada **javadoc**. Ela pode ser encontrada no Link 2.1.

Link 2.1
https://docs.oracle.com/en/java/javase/17/docs/api/index.html

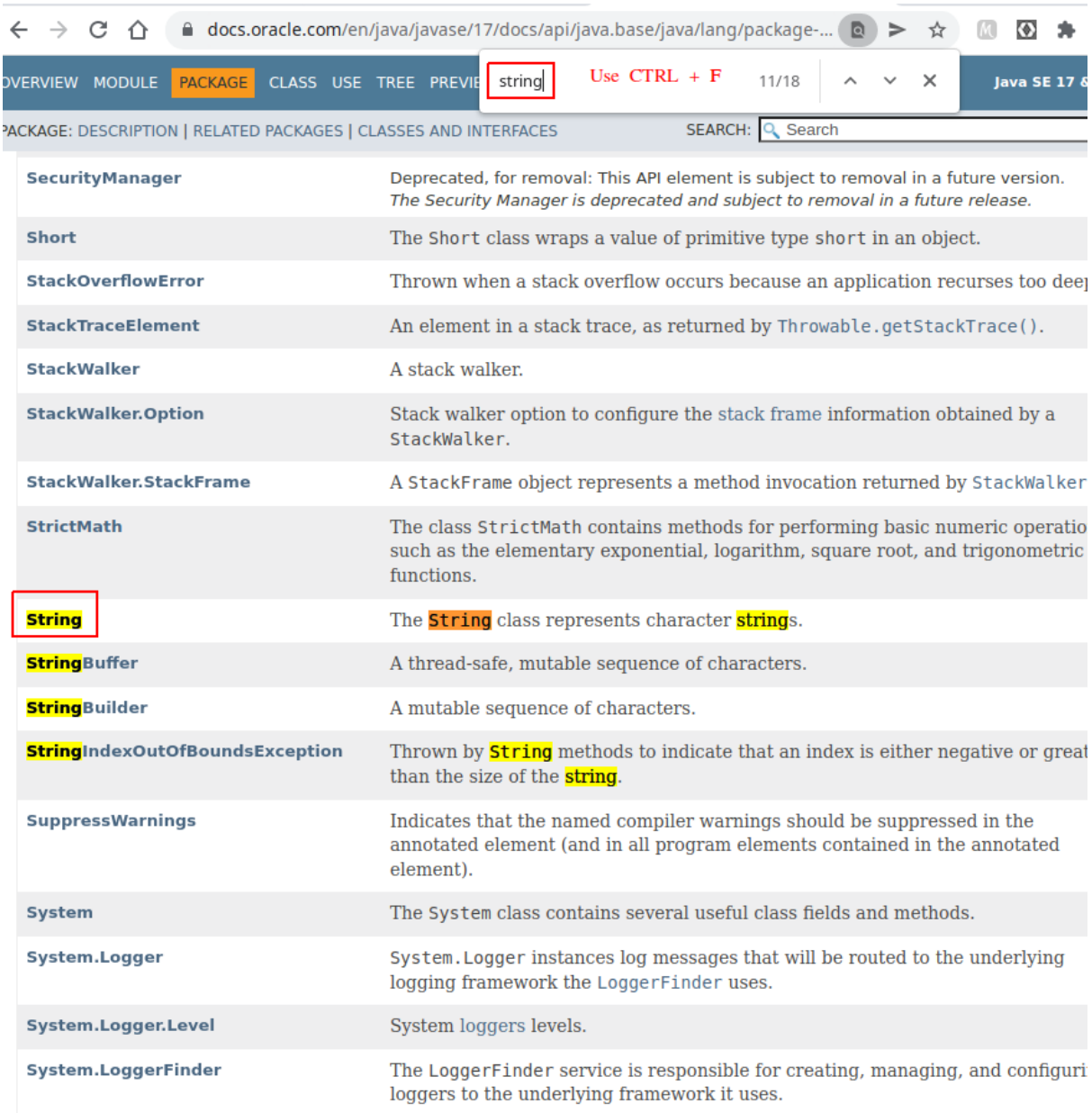
Todas as classes, interfaces, métodos etc. da especificação Java se encontram documentadas nesta página. Isso inclui a própria classe String.

Para encontrar a documentação da classe String, clique:

java.base >> java.lang

Na página resultante, use CTRL + F no seu navegador e busque por String. Veja a Figura 2.1. Clique em **String**.

Figura 2.1



Junto com seu professor, leia trechos da documentação da classe String e vasculhe seu conteúdo, dando especial atenção aos métodos que a classe possui, como

- charAt
- compareTo
- contains
- equals
- equalsIgnoreCase
- format
- isEmpty
- indexOf
- lastIndexOf
- length
- repeat
- replace
- startsWith
- substring
- toLowerCase
- toUpperCase

(Comprimento de strings: o método length) O Bloco de Código 2.1 mostra como calcular o número de caracteres de uma string.

Bloco de Código 2.1

```
import javax.swing.JOptionPane;
public class Strings{
   public static void main(String[] args) {
        // Verificar o comprimento de uma string
        String s = JOptionPane.showInputDialog("Digite uma
string");
        int comprimento = s.length();
        JOptionPane.showMessageDialog(null, s + " tem " +
comprimento + " caracteres.");
    }
}
```

(**Formatação de strings: o método format**) Concatenar strings utilizando o operador **+** é ineficiente computacionalmente e seu uso tende a comprometer a legibilidade do código. O método **format** da classe String pode auxiliar. A ideia é a seguinte:

- Especificamos um “template” que contém as partes “fixas” da string que desejamos
- O template possui códigos de formatação que serão substituídos por valores de interesse
- Especificamos quais valores serão utilizados como substitutos dos códigos de formatação, o que ocorre de acordo com a ordem.
- Cada tipo tem um código de formatação apropriado.

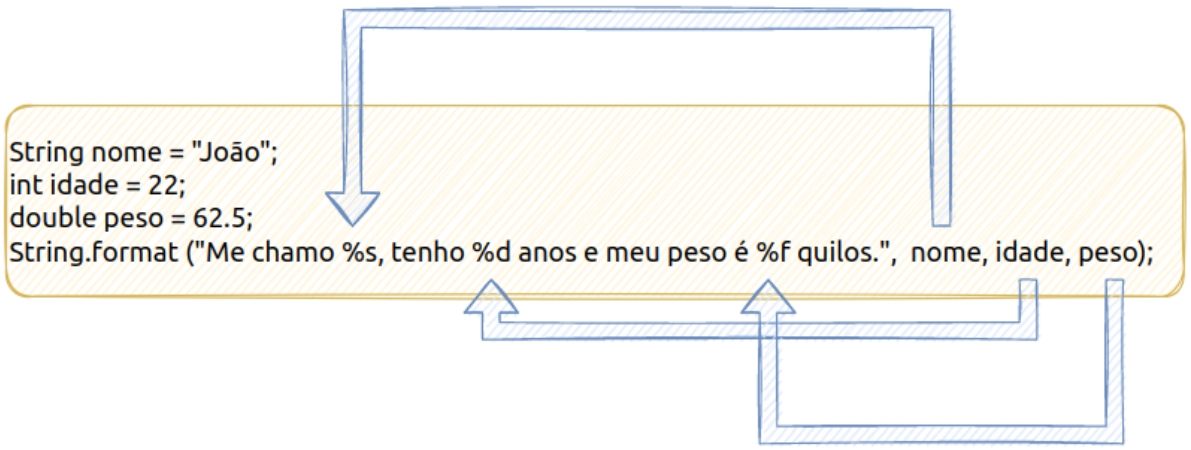
A Tabela 2.1 mostra alguns dos principais códigos de formatação associados a seus respectivos tipos.

Tabela 2.1

Tipo	**Código de formatação**
String	%s
double	%f
int	%d
char	%c

A Figura 2.2 ilustra a forma como a substituição de códigos de formatação pelos valores acontece.

Figura 2.2



O Bloco de Código 2.2 mostra um exemplo de uso do método format.

Bloco de Código 2.2

```
import javax.swing.JOptionPane;
public class Strings{
   public static void main(String[] args) {
        String nome = JOptionPane.showInputDialog("Qual o seu
nome?");
        int idade =
Integer.parseInt(JOptionPane.showInputDialog("Quantos anos você
tem?"));

        //concatenando com o operador +
        JOptionPane.showMessageDialog(null, "Oi, " + nome + ".
Você tem " + idade + " anos.");

        //"montando a string com o método format"
        String s = String.format ("Oi, %s. Você tem %d anos.",
nome, idade);
        JOptionPane.showMessageDialog(null, s);

   }
}
```

(Caractere na posição especificada: o método charAt) Podemos descobrir qual caractere se encontra numa determinada posição de uma string utilizando o método charAt, como mostra o Bloco de Código 2.3.

Bloco de Código 2.3

```
import javax.swing.JOptionPane;
public class Strings{
   public static void main(String[] args) {
        String teste = "Hello, strings";
        int posicao =
Integer.parseInt(JOptionPane.showInputDialog("Digite a posição
desejada"));
        String resultado = String.format("O caractere na posição
%d é o %c.", posicao, teste.charAt(posicao));
        JOptionPane.showMessageDialog(null, resultado);
   }
}
```

(Concatenação de strings: o método concat) Assim como o operador +, o método concat pode ser utilizado para concatenar strings, como mostra o Bloco de Código 2.4.

Bloco de Código 2.4

```
import javax.swing.JOptionPane;
public class Strings{
   public static void main(String[] args) {
        String nome = JOptionPane.showInputDialog("Qual o seu
nome?");
        String sobrenome = JOptionPane.showInputDialog ("Qual o
seu sobrenome?");
        String resultado = nome.concat(" ").concat(sobrenome);
        JOptionPane.showMessageDialog(null, resultado);
   }
}
```

(Conversão para caixa baixa e alta: os métodos toLowerCase e toUpperCase) O Bloco de Código 2.5 mostra como podemos converter uma string para que seja composta apenas letras minúsculas ou maiúsculas.

Bloco de Código 2.5

```
import javax.swing.JOptionPane;
public class Strings{
   public static void main(String[] args) {
        String teste = "HeLLo, WoooOrlllD";
        String minusculas = teste.toLowerCase();
        String maiusculas = teste.toUpperCase();
        JOptionPane.showMessageDialog(null, String.format
("Maiúsculas: %s\n Minúsculas: %s", maiusculas, minusculas));
    }
}
```

(Comparação por igualdade: os métodos equals e equalsIgnoreCase) Strings podem ser comparadas por igualdade de diferentes formas. Se a necessidade é verificar se duas strings são iguais diferenciando maiúsculas de minúsculas, usamos o método equals. Se a caixa não importar, usamos o método equalsIgnoreCase. Veja o Bloco de Código 2.6.

Bloco de Código 2.6

```
import javax.swing.JOptionPane;
public class Strings{
   public static void main(String[] args) {
        String s1 = JOptionPane.showInputDialog("Qual a primeira string?");
        String s2 = JOptionPane.showInputDialog("Qual a segunda string?");
        //comparação considerando maiúsculas e minúsculas: s é diferente de
S
        if (s1.equals(s2)){
            JOptionPane.showMessageDialog(null, "Considerando a caixa, as
strings são iguais");
        }
        else{
            JOptionPane.showMessageDialog(null, "Considerando a caixa, as
strings são diferentes");
        }
        //comparação considerando maiúsculas e minúsculas: s é igual a S
        if (s1.equalsIgnoreCase(s2)){
            JOptionPane.showMessageDialog(null, "Desconsiderando a caixa, as
strings são iguais");
        }
        else{
            JOptionPane.showMessageDialog(null, "Desconsiderando a caixa, as
strings são diferentes");
        }
    }
}
```

(Comparação lexicográfica: quem aparece primeiro no dicionário? O método compareTo) Com duas strings em mãos, podemos verificar qual delas aparece primeiro no dicionário. Isso pode ser feito utilizando-se o método **compareTo**. Se as duas strings são referenciadas por variáveis chamadas **s1** e **s2**, seu uso fica assim:

s1.compareTo(s2);

O método compareTo produz um número inteiro. Se s1 aparece antes de s2 no dicionário, ele é negativo. Se s2 aparece antes de s1 no dicionário, ele é positivo. Se as strings são iguais, o valor produzido por compareTo é igual a zero. Veja o Bloco de Código 2.7.

Bloco de Código 2.7

```
import javax.swing.JOptionPane;
public class Strings{
   public static void main(String[] args) {
        String s1 = JOptionPane.showInputDialog("Qual a primeira string?");
        String s2 = JOptionPane.showInputDialog("Qual a segunda string?");
        //considerando caixa alta e baixa
        int resultado = s1.compareTo(s2);
        if (resultado < 0) {
            JOptionPane.showMessageDialog(null, s1 + " vem antes de " + s2 + " no
dicionário");
        }
        else if (resultado > 0){
            JOptionPane.showMessageDialog(null, s2 + " vem antes de " + s1 + " no
dicionário");
        }
        else{
            JOptionPane.showMessageDialog(null, s2 + " e " + s1 + " são iguais");
        }

        //desconsiderando caixa alta e baixa
        resultado = s1.compareToIgnoreCase(s2);
        if (resultado < 0) {
            JOptionPane.showMessageDialog(null, s1 + " vem antes de " + s2 + " no
dicionário");
        }
        else if (resultado > 0){
            JOptionPane.showMessageDialog(null, s2 + " vem antes de " + s1 + " no
dicionário");
        }
        else{
            JOptionPane.showMessageDialog(null, s2 + " e " + s1 + " são iguais");
        }
    }
}
```

Nota. A comparação entre caracteres é baseada em seu código Unicode. Visite o Link 2.1 para ver os códigos Unicode dos principais caracteres.

Link 2.1
https://docs.google.com/spreadsheets/d/1mUDDFXK1yuWQnMQQLkxXm3ek-BeINh0RRDX4RY7__54/edit?usp=sharing

Exercícios

1. Ler uma única string que contém nome e sobrenome de uma pessoa. Exemplo: “João Silva”. Para este exemplo, o programa deve exibir: “Olá, João. Seu sobrenome é Silva”.

2. Ler uma string e verificar se ela representa uma senha válida. Para tal, ela deve ter as seguintes características:

- comprimento igual a 4
- primeiro símbolo igual a A ou a
- conter pelo menos um número ímpar

3. Escreva um programa que

- lê uma string **s**
- lê dois inteiros **a** e **b**
- Exibe a substring que começa na posição **a** e termina na posição **b**, incluindo a posição b.

  Exemplo: s = Hello World, a = 2, b = 4. Resultado: llo.

4. Escreva um programa que lê uma string composta por duas palavras separadas por um espaço em branco. Seu programa deve exibir o comprimento de cada palavra.