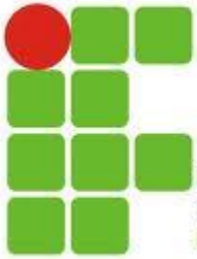


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

AULA 12

CLASSES DA API JAVA

Disciplina: Programação Orientada a Objetos
Professora: Alba Lopes
alba.lopes@ifrn.edu.br

Classes da API Java

- A maioria dos programas de computador que resolvem problemas do mundo real não são pequenos
- A experiência de programação mostrou que a melhor maneira de desenvolver e manter um programa grande é construí-lo através de pedaços pequenos ou *módulos*
- Em Java, a modularização é feita através de classes e métodos

Classes da API Java

- Vimos que é possível criar Classes utilizando a Linguagem Java
- Porém, a linguagem dispõe de diversas classes já criadas que podemos fazer uso delas
- A Java API (*Application Programming Interface* - ou Interface de Programação de Aplicações) oferece uma rica coleção de classes e métodos para:
  - realizar cálculos matemáticos comuns
  - manipular strings
  - verificação de erros
  - etc

Classes da API Java

- As classes predefinidas em Java são agrupadas em diretórios chamados de pacotes
- Coletivamente, esses pacotes são referidos como Java API
- Para utilizar classes já definidas e disponíveis em pacotes, utiliza-se a instrução **import**.
  - Nessa instrução, especifica-se a localização das classes que são usadas
  - **Exemplo:** ao utilizarmos a classe **Scanner** para ler valores digitados pelo usuário, escrevemos a instrução **import**:

```
import java.util.Scanner;
```

Classes da API Java

- Consulte o seguinte link para verificar os diversos pacotes e classes disponíveis na API de JAVA: http://docs.oracle.com/javase/6/docs/api/
- Ao construir grandes programas, sempre busque na API para verificar se há alguma classe disponível que ajude-o a resolver o seu problema.

Métodos em Java

- Em Java, métodos são invocados (ou chamados) escrevendo o nome do método seguido pela lista de argumentos (ou parâmetros).
- Em geral, para utilizar um método de uma classe, deve-se, primeiro instanciar um objeto do tipo da classe e então chamar o método que se deseja.
  - Isso não se aplica a métodos estáticos (veremos mais adiante o que são esses métodos)
- A sintaxe utilizada é:

```
<nome_do_objeto>.<nome_do_metodo>(<argumentos>)
```

Classe String

- String é uma classe JAVA que faz parte do pacote java.lang.String
- Os objetos da classe String são tratados como se fossem tipos primitivos (como **int**, **float**, **boolean**)
  - Por esse motivo não é necessário realizar o import quando se utiliza objetos do tipo String

Classe String

- As strings podem ser instanciadas de duas formas
  - Atra
  - Atra literal

```
String s = new String("nova String");
```

```
String s = "nova String";
```

- A classe String possui métodos que servem para realizar operações sobre as Strings, como por exemplo
  - Comparar strings
  - Procurar um caractere na string
  - Informar a quantidade de caracteres existentes
  - Criar uma nova string com todas as letras maiúsculas ou minúsculas, etc

Métodos da Classe String

- Método para **comparar**
  - Em Java, String são comparadas através do método **.equals(String s)**. O método **equals** requer que a String que se deseja comparar seja passada por parâmetro (argumento):

```
public static void main(String[] args) {
    String a = "teste";
    String b = "teste";

    if (a.equals(b)){
        System.out.println("String iguais!");
    }else{
        System.out.println("String diferentes!");
    }

}
```

Métodos da Classe String

- Método para **concatenar**
  - String podem ser concatenadas (juntar uma com a outra) através do método **.concat(String s)**

```
public static void main(String[] args) {
        String silaba1 = "ca";
        String silaba2 = "sa";
        String palavra = silaba1.concat(silaba2);
        System.out.println(palavra);

}
```

Métodos da Classe String

- Método para **concatenar**
  - O operador + pode ser utilizado ao invés do método **concat** para facilitar a construção dos programas.

```
public static void main(String[] args) {
        String silaba1 = "ca";
        String silaba2 = "sa";
        String palavra = silaba1 + silaba2;
        System.out.println(palavra);

}
```

Métodos da Classe String

- Método para **tamanho**
  - Retornar o tamanho da string: **.length().** Não necessita de nenhum argumento.

```
public static void main(String[] args) {
    String palavra = "moradia";
    int tamanho;
    tamanho = palavra.length();
    System.out.println("A quantidade de caracteres da palavra é: " + tamanho);
}
```

```
Saída - Exemplos String (run)
run:
A quantidade de caracteres da palavra é: 7
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

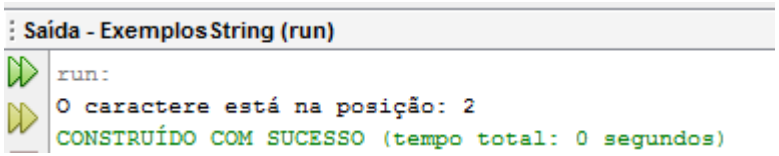
Métodos da Classe String

- Método para retornar caractere em determinado índice
  - Retorna o índice do caractere **c** passado por parâmetro: **indexOf(char c)**
  - As strings começam a contar do caractere 0:

m	o	r	a	d	i	a
0	1	2	3	4	5	6

```
String palavra = "moradia";
int posicao = palavra.indexOf('r');
System.out.println("O caractere está na posição: " + posicao);
```

  - Resultado



Métodos da Classe String

- Método para retornar caractere em determinado índice
  - Se o caractere buscado não existir na String, o valor **-1** é retornado

m	o	r	a	d	i	a
0	1	2	3	4	5	6

```
String palavra = "moradia";
int posicao = palavra.indexOf('x');
System.out.println("O caractere está na posição: " + posicao);
```

```
Saída - Exemplos String (run)
run:
O caractere está na posição: -1
CONSTRUÍDO COM SUCESSO (tempo total: 1 segundo)
```

Métodos da Classe String

- É possível utilizar o método indexOf para procurar não apenas caracteres, mas uma string também. O funcionamento é o mesmo:

indexOf(String s)

m	o	r	a	d	i	a
0	1	2	3	4	5	6

```
String palavra = "moradia";
int posicao = palavra.indexOf("dia");
System.out.println("A string está na posição: " + posicao);
```

```
Saída - ExemplosString (run)
run:
A string está na posição: 4
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Métodos da Classe String

- Retorna o caractere na posição indicada: **.charAt(int)**

```
13
14      public static void main(String[] args) {
15          String palavra = "moradia";
16          char caractere = palavra.charAt(4);
17          System.out.println("O caractere na posição 4 é: " + caractere);
18
```

Saída - ExemplosString (run) | Tarefas

```
run:
O caractere na posição 4 é: d
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Métodos da Classe String

- Outros métodos
  - **toLowerCase()**
    - Retorna nova String toda minúscula. Não necessita de argumento
  - **toUpperCase()**
    - Retorna nova String toda maiúscula. Não necessita de argumento.
  - **compareTo(String s)**
    - Compara duas strings lexográficamente (em ordem alfabética). Retorna: 0 se as strings forem iguais; valor maior do que 0 se a string for maior; valor menor que 0 a string for menor.
  - **compareToIgnoreCase(String s)**
    - Compara duas strings alfabeticamente ignorando maiúsculas e minúsculas.

Métodos da Classe String

- Outros métodos
  - **replace(char** caractere_antigo, **char** novo_caractere)
    - Retorna uma nova string substituindo todas as ocorrências do caractere_antigo pelo caractere_novo
  - **substring(int** inicio, **int** fim)
    - Retorna uma nova string que é parte da string original, delimitada pelos índices passados como parâmetro.

Exercícios

1. Crie um programa em Java que leia o login e a senha de um usuário. Caso o login seja igual a “ifrn” e a senha “escola”, deverá ser exibida a mensagem “Usuário autenticado no sistema”. Caso contrário, deverá ser exibida a mensagem “Falha na autenticação”. Ignore maiúsculas e minúsculas
2. Crie um programa em Java que leia uma frase e substitua todas os letras “a” por “b”.
3. Crie um programa em Java que leia uma frase e remova todos os caracteres de espaços da frase. Ex: a frase “O livro está em cima da mesa” deverá ficar como: “Olivroestáemcimadamesa”
4. Criar um algoritmo que, dado o nome de uma pessoa (Nome + Sobrenome), escreva apenas o sobrenome. Ex: “Alba Lopes” , escreve somente “Lopes”
5. Crie um programa em Java que leia duas palavras do usuário e, em seguida, e escreva-as em ordem alfabéticas.
6. Crie um programa em Java que simule a criação de um cadastro. Deverá ser informado o nome e o e-mail. O sistema só deve permitir realizar cadastro caso o e-mail digitado seja válido. Um e-mail é considerado válido caso possua os caracteres arroba (@) e ponto (.). Além disso, o nome anterior ao @ deve possuir no mínimo 2 caracteres.

Classe Math

- Os métodos da classe Math permitem ao programador realizar certos cálculos matemáticos comuns
- Eles não precisam instanciar nenhum objeto antes da utilização, pois são métodos **estáticos** da classe.
- Para utilizá-los, escreve-se:

```
<nome_da_classe>.<nome_do_metodo>(<argumentos>)
```

Classe Math

- **Exemplo:** calcular a raiz quadrada de um número
  - Utiliza-se o método **.sqrt(double d)** da classe Math.

```
public static void main(String [] args){
    double x = 9.0;
    double raiz = Math.sqrt(x);
    System.out.println("A raíz de " + x + " é: " + raiz);
}
```

- Os métodos estáticos da classe Math são sempre utilizados d

Math.<nome_do_metodo>(<argumentos>)

Classe Math

- Outros métodos

Método	Descrição	Exemplo
abs (double d)	Valor absoluto de x. Pode receber também como argumento um valor **float, int e long**	para x > 0, **abs(x)** é x para x = 0, **abs(x)** é 0 para x < 0, **abs(x)** é -x
ceil (double d)	Arrendonda x para o inteiro maior do que x.	**ceil(9.2)** é 10.0 **ceil(-9.8)** é -9.0
floor(double x)	Arredonda x para o inteiro menor do que x	**floor(9.2)** é 9.0 **floor(-9.8)** é -10.0
max(double x, double y)	Maior valor entre x e y. Pode receber também como argumento um valor **float, int e long**	**max (2.3, 12.7)** é 12.7 **max (-2.3, -12.7)** é -2.3

Classe Math

- Outros métodos

Método	Descrição	Exemplo
min(double x, double y)	Menor valor entre x e y. Pode receber também como argumento um valor **float, int e long**	**max (2.3, 12.7)** é 2.3 **max (-2.3, -12.7)** é -12.7
pow(double x, double y)	x elevedo à potência y (x^y)	pow(2.0, 7.0) é 128.0 pow(9.0, 2.0) é 81.0
sqrt(double x)	Raíz quadrada de x	sqrt(900.0) é 30.0 sqrt(9.0) é 3.0
random()	Gera um valor aleatório entre 0.0 e 1.0	

Classe Math

- **Exemplos:**

Referências

1. http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/String.html
2. http://docs.oracle.com/javase/6/docs/api/
3. DEITEL, H,M; DEITEL, P.J. “Java como programar”.