



**INSTITUTO FEDERAL DE  
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA**  
RIO GRANDE DO SUL  
Câmpus Feliz

# **Arrays de objetos, Strings, e atributos e métodos estáticos**

Prof. Moser Fagundes

Programação II  
Técnico em Informática

# Sumário

- Arrays de objetos
- Strings
- Atributos estáticos
- Métodos estáticos
- Exercícios

# Arrays de Objetos

Até o momento estudamos apenas arrays de tipos de **dados primitivos**, porém também podemos criar arrays de **objetos**.

Por exemplo, podemos criar arrays de:

- **Carros**
- **Pessoas**
- **Computadores**

# Exemplo

```
public class Carro {  
    private String marca;  
    private String modelo;  
  
    public Carro(String marca, String modelo) {  
        this.marca = marca;  
        this.modelo = modelo;  
    }  
  
    public String getMarca() {  
        return marca;  
    }  
  
    public String getModelo() {  
        return modelo;  
    }  
}
```

# Exemplo

```
public class Exemplo1 {  
  
    public static void main(String[] args) {  
  
        // Criamos o array de carros  
        Carro[] c = new Carro[2];  
  
        // Criamos os 2 carros do array  
        c[0] = new Carro("Ford", "Maverick");  
        c[1] = new Carro("Chevrolet", "Opala");  
  
        for (int i = 0; i < c.length; i++) {  
            System.out.println(c[i].getMarca());  
            System.out.println(c[i].getModelo());  
            System.out.println();  
        }  
    }  
}
```

# Exercício

*Construa um **array de cidades**, sendo que cada elemento do array é um objeto da classe **Cidade**, a qual deve ser criada por você.*

# Strings

Uma String é uma seqüência de caracteres que pode ser criada de 3 maneiras diferentes conforme podemos ver abaixo. Até agora, usamos apenas a primeira maneira.

```
public class TesteCria {
```

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

```
        String cidade = "Feliz ";
```

1ª maneira

```
        String estado = new String("RS ");
```

2ª maneira

```
        char letrasPais[] = {'B','r','a','s','i','l'};
```

```
        String pais = new String(letrasPais);
```

3ª maneira

```
        System.out.println(cidade + estado + pais);
```

```
    }
```

```
}
```

# equals e equalsIgnoreCase

Com o `equals` podemos comparar duas Strings. Se usarmos o `equalsIgnoreCase`, o programa ignora diferenças de minúsculas ou maiúsculas.

```
public class TesteEquals {  
    public static void main(String[] args) {  
        String cidade = "Feliz";  
  
        if(cidade.equals("Feliz")) {  
            System.out.println("Igual a Feliz - Case Sensitive");  
        }  
  
        if (cidade.equals("FELIZ")) {  
            System.out.println("Igual a FELIZ - Case Sensitive");  
        }  
  
        if (cidade.equalsIgnoreCase("FELIZ")) {  
            System.out.println("Igual a FELIZ - Ignore Case Sensitive");  
        }  
    }  
}
```



# startsWith e endsWith

Com o **startsWith** / **endsWith** podemos verificar se uma String começa / termina com uma determinada substring.

Estes métodos retornam **true** ou **false**.

```
public class TesteEndsWith {  
    public static void main(String[] args) {  
        String cidade = "IFRS-Campus-Feliz";  
        System.out.println(cidade.endsWith("eliz"));  
        System.out.println(cidade.endsWith("Fel"));  
        System.out.println(cidade.startsWith("IFRS-Cam"));  
        System.out.println(cidade.startsWith("RS-Campus"));  
        System.out.println(cidade.startsWith("Campus",5));  
    }  
}
```

# toCharArray

Este método converte uma String em um array de **char**, ou seja, uma String de 10 caracteres se converterá em um **char**[] de 10 elementos.

```
public class TesteToCharArray {  
  
    public static void main(String[] args) {  
  
        String cidade = "Feliz";  
        char[] cidadeCharArray = cidade.toCharArray();  
  
        for (int i = 0; i < cidadeCharArray.length; i++) {  
            System.out.println("Caracter: " + cidadeCharArray[i]);  
        }  
  
        for(char c : cidadeCharArray){  
            System.out.println("Caracter: " + c);  
        }  
    }  
}
```

# split

O método `split` cria um array de Strings com base no “caractere divisor” passado como argumento.

```
public class TesteSplit {  
  
    public static void main(String[] args) {  
  
        String teste = "IFRS-Campus-Feliz";  
        String[] splitArray = teste.split("-");  
  
        for (int i = 0; i < splitArray.length; i++) {  
            System.out.println("Elemento: " + splitArray[i]);  
        }  
    }  
}
```

# substring

O método `substring` retorna uma parte específica de uma determinada `String` delimitada pelo índice dos caracteres inicial (inclusive) e final (exclusivo).

```
public class TesteSubstring {  
    public static void main(String[] args) {  
        String teste = "Programacao em Java";  
  
        System.out.println(teste.substring(0, 5));  
        System.out.println(teste.substring(12, 14));  
        System.out.println(teste.substring(0, 11));  
    }  
}
```

# toLowerCase e toUpperCase

O método `toLowerCase` converte toda a String para caixa baixa e o `toUpperCase` faz o inverso, convertendo toda a string para caixa alta.

```
public class TesteLower {  
    public static void main(String[] args) {  
        String cidade = "IFRS-Campus-Feliz";  
        System.out.println(cidade);  
        System.out.println(cidade.toLowerCase());  
        System.out.println(cidade.toUpperCase());  
    }  
}
```

# trim

Este método remove espaços em branco no início e fim da String.

```
public class TesteTrim {  
    public static void main(String[] args) {  
        String cidade = "    IFRS-Campus-Feliz    ";  
        System.out.println(cidade);  
        System.out.println(cidade.trim());  
    }  
}
```

# format

Formatando **Strings** usando o format.

```
package exemplo2;

public class TesteFormat {
    public static void main(String[] args) {
        String t1 = String.format("%15s %15s %15s", "Coluna 1", "Coluna 2", "Coluna 3");
        String t2 = String.format("%15s %15s %15s", "Hello", "World", "Again");

        System.out.println(t1);
        System.out.println(t2);

        String t3 = String.format("%-15s %-15s %-15s", "Coluna 1", "Coluna 2", "Coluna 3");
        String t4 = String.format("%-15s %-15s %-15s", "Hello", "World", "Again");

        System.out.println(t3);
        System.out.println(t4);
    }
}
```

**%-15s** ← s indica que é String  
↑  
↑  
↑  
número de caracteres  
- à esquerda  
% Início da especificação de formato

# format

Formatando **números** usando o format.

```
package exemplo2;

public class TesteFormatNumero {

    public static void main(String[] args) {

        String t1 = String.format("%-15s %-15s", "Coluna 1", "Coluna 2");
        String t2 = String.format("%-15.3f %-15.8f", 666.23429837482, 9.99);

        System.out.println(t1);
        System.out.println(t2);
    }
}
```

número de casas decimais

← f indica que é número

← número de caracteres

← - à esquerda

% Início da especificação de formato

**%-15.3f**



# Pergunta

É possível usar um método especificado dentro de uma classe sem criar um objeto desta classe?

# Modificador `static`

- O `static` (ou modificador estático) muda o escopo de um método ou atributo.
  - Se adicionarmos o `static` na declaração de um atributo ou na assinatura de um método, ao invés do mesmo pertencer ao objeto, ele pertencerá à classe.

# Exemplo

O exemplo abaixo cria uma classe chamada **Contador** que tem um atributo estático público chamado **numero** que inicialmente tem valor 0.

```
package exemplos;  
  
public class Contador {  
    public static int numero = 0;  
}
```

# Exemplo

A classe abaixo muda o valor do atributo estático, sem que seja criado um objeto. Observe que são feitas várias mudanças e impressões do valor do atributo.

```
package exemplos;

public class TesteEstatico {
    public static void main(String[] args) {
        System.out.println(Contador.numero);
        Contador.numero++;
        System.out.println(Contador.numero);
        Contador.numero = 10;
        System.out.println(Contador.numero);
    }
}
```

# Exercício

*Modifique a classe **Contador** de modo que o atributo estático numero contenha o número de objetos já criados desta classe. Para isso, crie um construtor que atualize o valor de numero.*

*Teste o seu código criando múltiplos objetos da classe **Contador** e depois imprima o valor do atributo estático numero.*

# O que são métodos estáticos?

Um **método estático** é um método (função) da classe, ou seja, podemos chamar usando a classe como referência.

Um exemplo é a classe **Math** ( já estudada ):

```
Math.abs(x);    // Valor absoluto  
Math.ceil(x);  // Arredonda para cima
```

Método estático

Classe **Math**

# Exemplo

```
package exemplos;
```

```
public class Matematica {
```

```
    public static int soma(int a, int b) {  
        return a+b;  
    }
```



Método estático

# Exemplo

```
package exemplos;
```

```
public class TesteStatic {
```

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

```
        int soma1 = Matematica.soma(10,20);  
        int soma2 = Matematica.soma(999,1);
```

```
        System.out.println(soma1);
```

```
        System.out.println(soma2);
```

```
    }
```

```
}
```

Chamadas ao método estático de soma.



# **Exercício**

Lista de exercícios no Moodle.