

# Classes

João Batista Ribeiro  
[joao42ibatista@gmail.com](mailto:joao42ibatista@gmail.com)

09/11/2020

UFV-CRP

# Requisitos e Comunicação

## Requisitos – recomendado

- conhecimentos básicos de programação

## Aulas e comunicação

- aulas síncronas pelo *Google Meet*
- grupo de comunicação no *Slack*
- conteúdo “extra” (vídeos, links, apostilas disponibilizados pelo grupo no *Slack*)

- Covid-19 #Fique em casa #Fique seguro

# Aula de hoje ...

→ Introdução a Classes – POO

Conteúdo utilizado: <https://github.com/ryuuzaki42/concurso-ufv-prog-2020>

# Roteiro

- Programação – POO
- Classes e Objetos
- Atributos e métodos
- Construtores
- Referências

# Relembrando ...

## Programação

- escrita, teste e manutenção de um *software*

## Paradigmas de programação

- maneiras/formas de criar uma solução/programa
- ex.: procedural e **orientado a objetos**
- paradigma mais adequado, não o certo ou errado

# Programação Orientada a Objetos

Procura compor modelos de forma mais próxima às **interações existentes no mundo real**

- Tenta de aproximar o mundo virtual criando um modelo do mundo real

Define que objetos se comunicam através da troca de mensagens para promover a troca de serviços

# O que é um objeto?



- caneta, livro, animal, aluno, compromisso marcado, conta corrente
- É alguma coisa que pode ser descrita por suas **características**, **comportamento** e **estado atual**

Fonte: <https://www.estudopratico.com.br/nomes-de-coisas-e-objetos-de-escritorio-office-em-ingles/>

# Objeto



Objetos de um mesmo tipo têm um formato/molde em comum

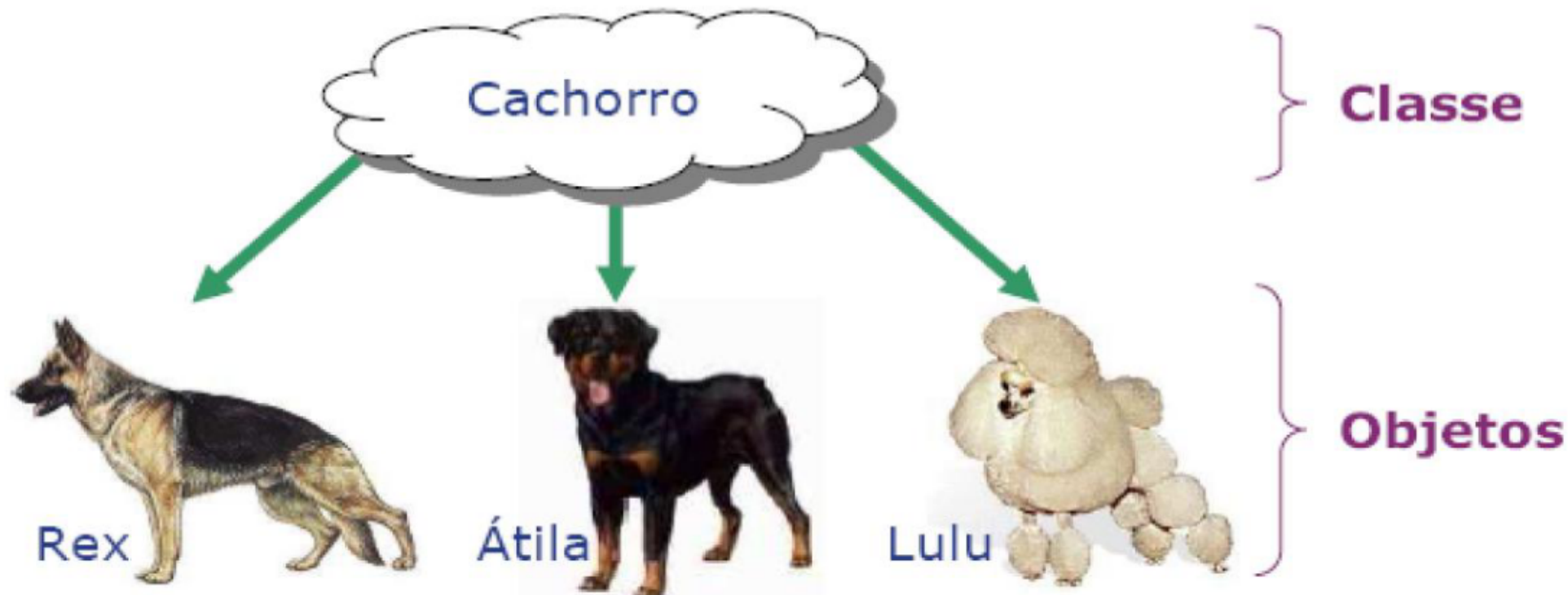
Esse formato ou classificação (classe) será nossa classe na POO



O objeto em si (ex.: caneta, carro) é um objeto na POO, uma instancia da classe “molde”



# Classe X Objetos



Classe é mais geral, e um objeto é mais específico

# Diferenças entre Classe e Objeto?

## Classe:

- é a definição do tipo
- representa um conjunto de objetos de mesmo tipo
- Classe = {obj1, obj2, obj3, ... objN}

## Objeto:

- é cada instância derivada da classe
- é um elemento do conjunto representado pela classe

# Definição de uma Classe

- 1) O que eu tenho? - **atributos** (características)
  - cor, modelo, carga (% de tinta); saldo, nº da conta
- 2) O que eu faço? - **métodos** (comportamentos)
  - escrever, tampar; saque, depósito
- 3) Como estou? - **estado**
  - escrevendo, 40 % de tinta; ativa, saldo de R\$ 500

# Atributos

→ características - propriedades

São as propriedades/características de um objeto, variáveis

Essas propriedades definem o estado de um objeto, podendo sofrer alterações

Todas informações sobre o objeto que julgar importante armazenar

- **nível de detalhes - abstração**

# Métodos

Definem os serviços/ações que podem ser solicitados a uma instância (objeto), **o comportamento de um objeto**

Exemplo: classe **ContaCorrente**, os métodos podem ser:

- `verificaSaldo`: retorna o saldo da conta
- `depositaValor`: deposita um valor especificado X na conta
- `retiraValor`: retira um valor especificado X da conta

# Classe

1º pensar e criar o molde – classe

```
classe Caneta {  
    cor: carácter  
    carga: inteiro ou real  
    ...  
    escrever()  
    tampar()  
    destampar()  
    ...  
}
```

Depois instanciar, classe → objeto

```
c1 = nova Caneta  
c1.cor = "azul"  
c1.carga = 50  
c1.tampar()  
  
c2 = nova Caneta  
c2.cor = "vermelho"  
c2.carga = 90  
c2.destampar()
```

# Linguagens de programação



- Linguagens POO, qual utilizar?
- Linguagem é uma ferramenta

# Classe Java – Caneta versão 1

```
public class CanetaVer1 {  
    String cor;  
    int carga;  
  
    void status() {  
        System.out.println("Cor: " + cor);  
        System.out.println("Carga: " + carga);  
    }  
}
```

## CamelCase

- class CanetaVer1
- string nomeCompleto

```
$ javac CanetaVer1.java  
$ javac CanetaMainVer1.java  
$ java CanetaMainVer1  
Status c1  
Cor: azul  
Carga: 90  
$
```

```
public class CanetaMainVer1 {  
    public static void main(String[] args) {  
        CanetaVer1 c1 = new CanetaVer1();  
        c1.cor = "azul";  
        c1.carga = 90;  
        System.out.println("Status c1");  
        c1.status();  
    }  
}
```



# Visibilidade - modificador de acesso

*public* (público) - Visível para todos

- permite acesso a qualquer código externo à classe
- facilidade de uso, mas com menor segurança
- analogia: acesso ao cofre do banco

• *private* (privado) - Visível dentro na classe

- proíbe qualquer acesso externo à própria classe
- maior segurança
- analogia: acesso através do atendente do banco

• *protected* – próximas aulas - herança

# Visibilidade - modificador de acesso

```
public class CanetaVer2 {  
    private String cor;  
    private int carga;  
    ...  
}
```

```
public class CanetaMainVer1 {  
    public static void main(String[] args) {  
        CanetaVer1 c1 = new CanetaVer1();  
        c1.cor = "azul";  
        c1.carga = 90;  
        System.out.println("Status c1");  
        c1.status();  
    }  
}
```

```
$ javac CanetaMain2.java
```

```
CanetaMain2.java:5: error: cor has private access in CanetaVer2
```

```
    c1.cor = "azul";
```

```
CanetaMain2.java:6: error: carga has private access in CanetaVer2
```

```
    c1.carga = 90;
```

```
2 errors
```

# Métodos especiais

## Métodos acessadores – get

- Retornam o valor de um atributo
- `getCor()`, `getTamanho()`, `getDataNascimento()`

## Métodos modificadores – set

- Modificam o valor de um atributo
- `setCor()`, `setTamanho()`, `setDataNascimento()`

**this** → usado para referenciar atributo da classe

# Caneta versão 2

```
public class CanetaVer2 {
    private String cor;
    private int carga;

    //getters
    public String getCor() {
        return this.cor;
    }
    public int getCarga() {
        return this.carga;
    }

    //setters
    public void setCor(String cor) {
        this.cor = cor;
    }
    public void setCarga(int carga) {
        this.carga = carga;
    }

    public void status() {
        System.out.println("Cor: " + this.cor);
        System.out.println("Carga: " + this.carga);
    }
}
```

```
public class CanetaMainVer2 {
    public static void main(String[] args) {
        CanetaVer2 c1 = new CanetaVer2();
        c1.setCor("azul");
        c1.setCarga(90);
        System.out.println("Status c1");
        c1.status();
    }
}
```

```
$ java CanetaMain2
Status c1
Cor: azul
Carga: 90
$
```

```
class ContaCorrente {  
    private float saldo;  
    ...  
  
    public void saque(float valor) {  
        System.out.println("Solicitado saque de valor R$ " + valor);  
        if (valor > 0) {  
            if (this.saldo >= valor) {  
                this.saldo -= valor;  
            } else  
                System.out.println("Saldo insuficiente");  
        } else {  
            System.out.println("valor do saque deve ser positivo");  
        }  
    }  
}
```

Altera o valor do atributo em apenas um método (parte de código)

- regra de negócio

# Construtores

- Permite fazer inicializações (definir o valor inicial) no objeto assim que ele é declarado com o *new* (instanciado)
- Construtor tem o mesmo nome da classe que pertence e com os parâmetros (atributos) que desejar inicializar

Por meio dos construtores podemos garantir que o código que eles contém será executado antes de qualquer outro código

```

public class CanetaVer3 {
    private String cor;
    private int carga;

    public CanetaVer3(){}

    public CanetaVer3(String cor, int carga) {
        this.cor = cor;
        this.carga = carga;
    }

    //getters
    public String getCor() {
        return this.cor;
    }
    public int getCarga() {
        return this.carga;
    }

    //setters
    public void setCor(String cor) {
        this.cor = cor;
    }
    public void setCarga(int carga) {
        this.carga = carga;
    }

    public void status() {
        System.out.println("Cor: " + this.cor);
        System.out.println("Carga: " + this.carga);
    }
}

```

```

public class CanetaMainVer3 {
    public static void main(String[] args) {
        CanetaVer3 c1 = new CanetaVer3();
        c1.setCor("azul");
        c1.setCarga(90);
        System.out.println("Status c1");
        c1.status();

        CanetaVer3 c2 = new CanetaVer3("verde", 55);
        System.out.println("Status c2");
        c2.status();
    }
}

```

```

$ java CanetaMainVer3
Status c1
Cor: azul
Carga: 90
Status c2
Cor: verde
Carga: 55
$

```

# Diferença entre construtor e método

- Construtores não podem retornar nenhum valor
- Construtores não podem ser chamados diretamente, somente quando a instância for inicializada com *new*
- Toda classe tem pelo menos um construtor, o construtor padrão (vazio, sem argumentos)
- Uma classe pode ter vários construtores



# Classe – esquema geral

```
public class <NomeDaClasse> {  
    // Atributos  
  
    // Construtores da classe  
  
    // Métodos get/set  
  
    // Métodos  
  
}
```

→ **Outros exemplos de código**

# Sugestão – Tarefinha - Extra

- Pesquise um pouco sobre os assuntos das próximas aulas  
→ próximo slide
- Confira as boas aulas/*playlists*:
  - POO - Curso em Vídeo  
[https://www.youtube.com/playlist?list=PLHz\\_AreHm4dkqe2aR0tQK74m8SFe-aGsY](https://www.youtube.com/playlist?list=PLHz_AreHm4dkqe2aR0tQK74m8SFe-aGsY)
  - POO - UNIVESP  
[https://www.youtube.com/playlist?list=PLxl8Can9yAHewZWSrIhpld71bk5N\\_W7W1](https://www.youtube.com/playlist?list=PLxl8Can9yAHewZWSrIhpld71bk5N_W7W1)

# Próximas aulas...

- Métodos e Passagem de Parâmetros
- Encapsulamento
- Herança
- Polimorfismo
- UML
- Tratamento de exceções
- Arquivo
- ...

# Principais Referências

- DEITEL, P.; DEITEL, H. M.; FURMANKIEWICZ, E.. **Java: como programar**. 8 ed. São Paulo: Pearson Prentice Hall, 2010
- Jaques, P. A. Apostila: **Programação Básica em Java**, 2017. Acesso 08/11/2020 <[http://professor.unisinos.br/pjaques/material/java\\_basico.pdf](http://professor.unisinos.br/pjaques/material/java_basico.pdf)>
- Caelum, Apostila: **Java e Orientação a Objetos**, Acesso 08/11/2020 <<https://www.caelum.com.br/apostila/apostila-java-orientacao-objetos.pdf>>
- DORÇA, F., **Aulas de Programação Orientada a Objetos**, Acesso: 08/11/2020 <<http://www.facom.ufu.br/~fabiano>>

# Alguma dúvidas?



**Obrigado pela atenção! :-)**