Universidade Federal de Viçosa, *campus* Rio Paranaíba
Instituto de Ciências Exatas e Tecnológicas
Sistemas de Informação
Programação, Algoritmos e Estruturas de Dados e afins

Classes

João Batista Ribeiro
joao42lbatista@gmail.com

Rio Paranaíba, Minas Gerais

Novembro, 2020

Requisitos, aulas e comunicação

- Conhecimentos básicos de Programação
- Aulas síncronas pelo *Google Meet*
- Comunicação por *e-mail* e grupo no *Slack*
- Conteúdo “extra” (vídeos, *links* e apostilas) disponibilizados no *Slack*

Covid-19 #Fique em casa #Fique seguro

Aula de hoje...

Introdução à Classes - POO

Conteúdo: https://github.com/ryuuzaki42/aula-classes-poo

Roteiro

- Programação e Programação Orientada a Objetos
- Classes e Objetos
- Atributos e Métodos
- Construtores

Relembrando...

Programação

- Escrita, teste e manutenção de um *software*

Paradigmas de programação

- Maneiras/formas de criar uma solução/programa
- Exemplo: procedural e **orientado a objetos**
- Paradigma mais adequado, não o certo ou errado

Programação Orientada a Objetos

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

- Tenta aproximar o mundo virtual criando um modelo do mundo real
- Escreve o código organizado em torno de objetos (conceitos), não de funções

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/

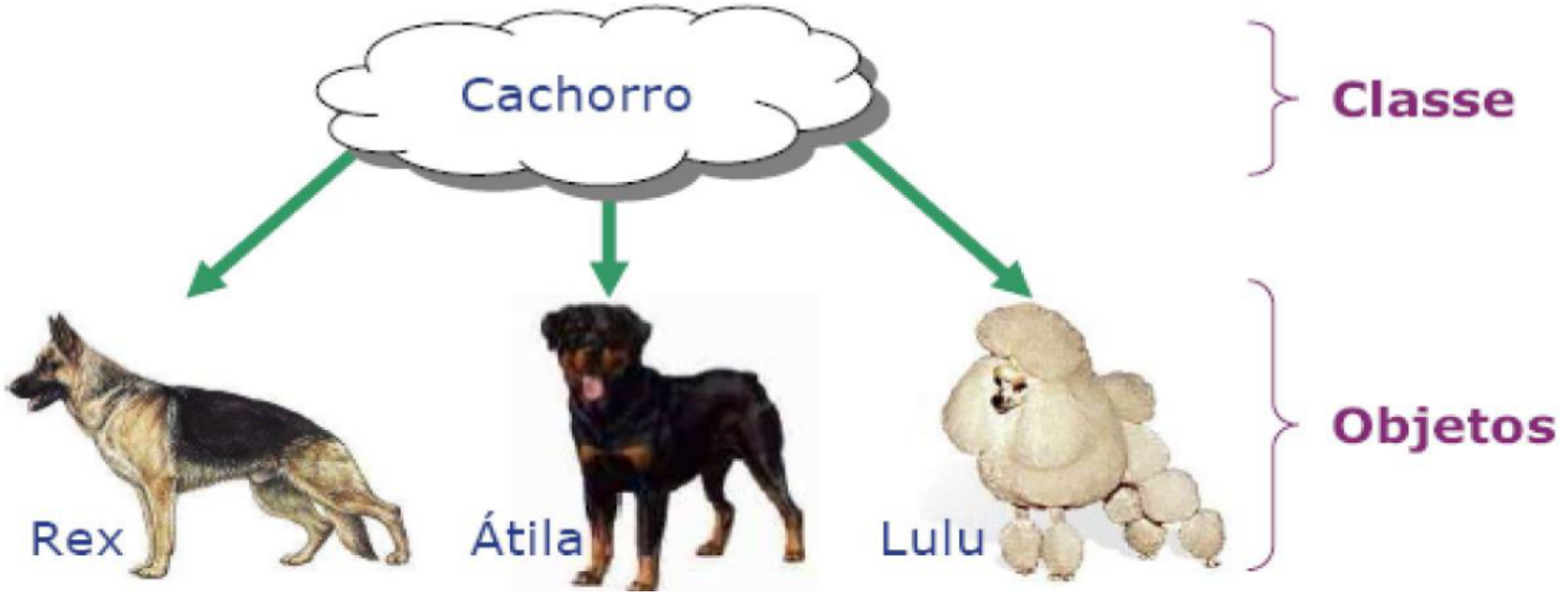
Objeto

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

Esse formato ou classificação (classe) é a nossa **classe** na POO

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

Classe x Objetos



- **Classe** é mais geral e **objeto** é mais específico

Diferenças

Classe

- É a definição do tipo
- Representa um conjunto de objetos de mesmo tipo
- Classe = {obj1, obj2, ... objN}

Objeto

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

Identificação de um objeto

- Distinção por sua própria existência
- Dois objetos são distintos mesmo que todos os seus atributos sejam iguais
  - Identificador único para cada objeto

Definição de uma classe

1) O que eu tenho? → características → **atributos**
   - cor, carga (% de tinta); saldo, nº da conta

2) O que eu faço? → comportamentos → **métodos**
   - escrever, tampar; saque, depósito

3) Como estou? → **estado** atual
   - escrevendo, 40 % de tinta; ativa, saldo de R$ 500

Atributos

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

- **abstração → nível de detalhes**

Métodos

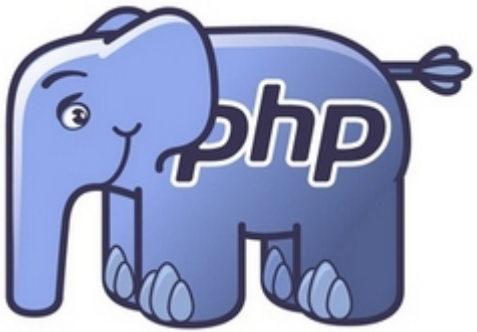
Definem os serviços/ações que podem ser solicitados a uma instância (objeto) → **comportamento de um objeto**

Não é possível executar um método de uma classe sem criar um objeto dela

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

- verificaSaldo: retorna o saldo da conta
- depositaValor: deposita um valor x na conta
- retiraValor: retira um valor x da conta

Linguagens de POO



- Qual utilizar?
- **Linguagem** é uma ferramenta!

Classes em Java

```
public class NomeClasse {
   tipo atributo1;
   tipo atributo2;
   ...
   retorno metodo1(parâmetros) {...}
   retorno metodo2(parâmetros) {...}
   ...
}
```

Tipo de dados mais comuns	
lógico	boolean
inteiro	int
real	float
caractere	char
Cadeia de caracteres	String

Retorno

- é o tipo de retorno
- **void** quando não há retorno

Classe Java – Caneta versão 1

```java
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
$
```

```java
public class CanetaMainVer1 {
    public static void main(String[] args) {
        CanetaVer2 c1 = new CanetaVer2();
        c1.cor = "azul";
        c1.carga = 90;
        System.out.println("Status c1");
        c1.status();
    }
}
```

Visibilidade de atributos

- *public*
  - permite acesso a qualquer código externo à classe
  - facilidade de uso, mas com menor segurança

    → analogia: acesso ao cofre do banco
- *private*
  - proíbe qualquer acesso externo à própria classe
  - maior segurança

    → analogia: acesso através do atendente do banco
- *protected* – próximas aulas: herança

```java
class ContaCorrente {
    private float saldo;
    ...

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

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

- **regra de negócio**

Visibilidade de atributos

```
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 CanetaMainVer1.java
CanetaMain2.java:4: error: cor has private access in CanetaVer2
        c1.cor = "azul";
CanetaMain2.java:5: 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 um atributo da classe

Caneta versão 2

```java
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);
    }
}
```

```java
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
$
```

Construtores

Permitem fazer **inicializações** no objeto assim que ele é declarado com o ***new*** (instanciado)

Devem ter o mesmo nome da classe que pertencem e com os parâmetros (atributos) que você desejar inicializar

Garantem que o código que contêm será executado antes de qualquer outro

Construtores x Métodos

Construtores não podem retornar nenhum valor

Construtores não podem ser chamados diretamente, somente quando o objeto é instanciado

Toda **classe** tem pelo menos um construtor, o **padrão** (vazio, sem argumentos)

```java
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);
    }
}
```

```java
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
$
```

O que vimos até aqui...

Classe: esquema geral

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

→ **Confira os outros exemplos de código**

Sugestões

→ Pesquise um pouco sobre os assuntos das **próximas aulas**:

- Passagem de parâmetros, Encapsulamento, Herança, Polimorfismo e UML

→ Confira as videoaulas/*playlists*:

- POO - Curso em Vídeo
  https://www.youtube.com/playlist?list=PLHz_AreHm4dkqe2aR0tQK74m8SFe-aGsY

- POO - UNIVESP
  https://www.youtube.com/playlist?list=PLxI8Can9yAHewZWSrlhpId71bk5N_W7W1

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>

Caelum. Apostila: **Java e Orientação a Objetos**, Acesso: 08/11/2020 <https://www.caelum.com.br/apostila/apostila-java-orientacao-objetos.pdf>

DORÇA, F., Notas de Aulas: **Programação Orientada a Objetos**, Acesso: 08/11/2020 <http://www.facom.ufu.br/~fabiano>

Obrigado pela atenção! :-)