

Técnica de Programação I



Curso Superior de Tecnologia em Desenvolvimento de
Software Multiplataforma

Aula 01

Prof. Claudio Benossi

Prof. Claudio Benossi

- Mestre em Tecnologia da Inteligência e Design Digital (PUC-SP)
- Especialista em Consultoria Web e E-Business (FATEC-SP)
- Bacharel em Ciências da computação (UNINOVE-SP)



Conhecendo a turma

Quem são vocês??





Plano de Aula

Plano de Aula

FATEC – Zona Leste

Disciplina – TÉCNICA DE PROGRAMAÇÃO I

PRESENCIAL - 80 Aulas



Plano de Aula

Competências Profissionais desenvolvidas neste componente

- ▶ Utilizar linguagens de programação orientada a objetos e raciocínio lógico adequados para resolução de situações problema e ou desenvolvimento de projetos diversos.

Plano de Aula

Competências Socioemocionais

- Agir com pensamento crítico voltado à resolução de situações-problema.
- Demonstrar capacidade de lidar com situações novas.
- Evidenciar iniciativa e flexibilidade para adaptar-se a novas dinâmicas.

Plano de Aula

Objetivos de Aprendizagem:

- Utilizar linguagem de programação, difundida no mercado, para codificação aplicando os conceitos de orientação a objetos.
- Abstração, encapsulamento, herança, polimorfismo. Relacionamento entre classes.
- Compreender e programar Tratamento de exceções.
- Criar Interfaces gráficas com usuário.
- Aplicar conceitos da Arquitetura Model-View-Controller.
- Conhecer frameworks de desenvolvimento front-end e back-end.
- Aplicar versionamento e documentação da aplicação

Plano de Aula

Ementa:

- Conceitos de orientação a objetos: Classes, Objeto, Encapsulamento, Herança, Polimorfismo.
- Princípios de padrões de projeto.
- Declaração de Classes e Objetos. Classe Abstrata.
- Métodos. Sobrecarga de Métodos.
- Conceitos de Herança múltipla.
- Modificadores de acesso.
- Construtores.
- Manipulação de Exceções.
- Conceitos e aplicações de arquitetura em Camadas.
- Uso de Interface Gráfica e Teste de Software.

Plano de Aula

Metodologia proposta:

- Aulas Expositivas.
- Aprendizagem Baseada em Projetos/Problemas.
- Gamificação.
- Coding Dojo.





Plano de Aula

Instrumentos de avaliação:

- ▶ **Avaliação Formativa:** Exercícios para prática. Análise e Resolução de Problemas acompanhado de rubrica de avaliação.
- ▶ **Avaliação Somativa:** Provas. Projetos. Avaliação em pares. Desafios de Programação e Trabalhos Interdisciplinares.

Plano de Aula

Bibliografia Básica:

- FURGERI, S. **Programação orientada a objetos: Conceitos e técnicas**. São Paulo: Erica. 2015.
- NASCIMENTO JR. O.S. **Introdução à Orientação a Objetos com C++ e Python: Uma abordagem prática**. São Paulo: Novatec, 2017
- SIERRA, K. BATES, B. **Use a Cabeça! Java. 2 ed.** São Paulo: O'Rilly, 2005.

Plano de Aula

Bibliografia Complementar:

- BHARGAVA, A. Y. **Entendendo Algoritmos: Um guia ilustrado para programadores e outros curiosos**. São Paulo: Novatec, 2019.
- KOPEC, D. **Problemas Clássicos de Ciência da Computação com Python**. São Paulo: Novatec, 2019.
- MARTIN, Robert C. **Código Limpo: Habilidades Práticas do Agile Software**. Rio de Janeiro: Alta Books, 2012.
- RAMALHO, L. **Python Fluente: Programação Clara, Concisa e Eficaz**. São Paulo: Novatec, 2015.
- SCHILD, H. **Java para Iniciantes: Crie, Compile e Execute Programas Java Rapidamente**. 6 ed. Porto Alegre: Bookman: 2015.
- SILVERMAN, R. E. **Git: guia prático**. São Paulo: Novatec, 2019.

1. Unidade

Introdução ao Programação Orientada a Objetos

Programação Orientada a Objetos

Paradigmas de Programação

Um paradigma de programação fornece (e determina) a visão que o programador possui sobre a estruturação e execução do programa;



Programação Orientada a Objetos

Paradigmas de Programação

Assim como ao resolver um problema podemos adotar uma entre variadas metodologias, ao criar um programa podemos adotar um determinado paradigma de programação para desenvolvê-lo.

Programação Orientada a Objetos

Paradigmas de Programação

Vamos entender a diferença entre **Programação Estruturada** e **Programação Orientada a Objetos**.



Programação Estruturada

- **Sequência:** Uma tarefa é executada após a outra, linearmente.
- **Decisão:** A partir de um teste lógico, determinado trecho de código é executado, ou não.

Programação Estruturada

- **Iteração:** A partir de um teste lógico, determinado trecho de código é repetido por um número finito de vezes.

Programação Estruturada

Vantagens

- É fácil de entender.
- Ainda muito usada em cursos introdutórios de programação.
- Execução mais rápida.

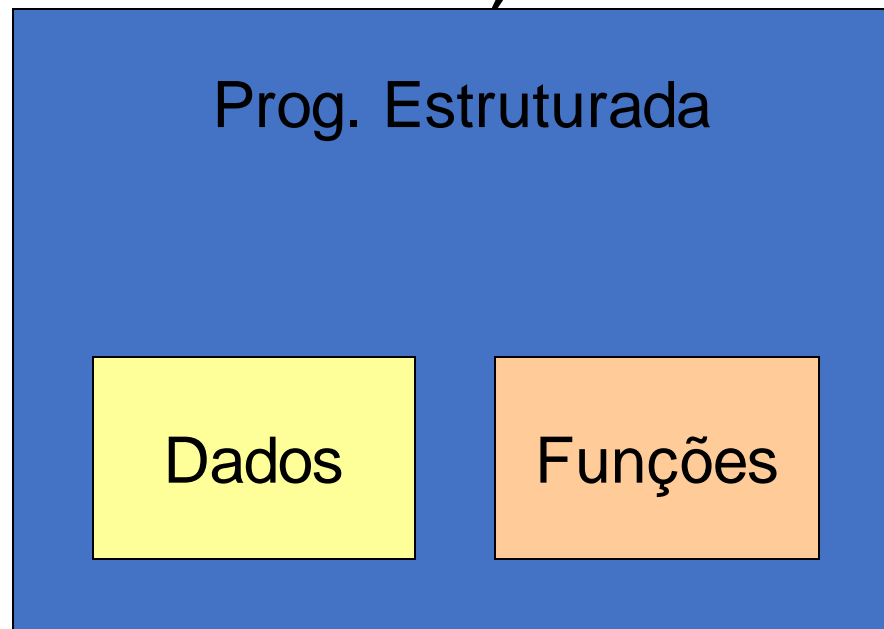
Programação Estruturada

Desvantagens

- Baixa reutilização de código
- Códigos confusos: Dados misturados com comportamento.

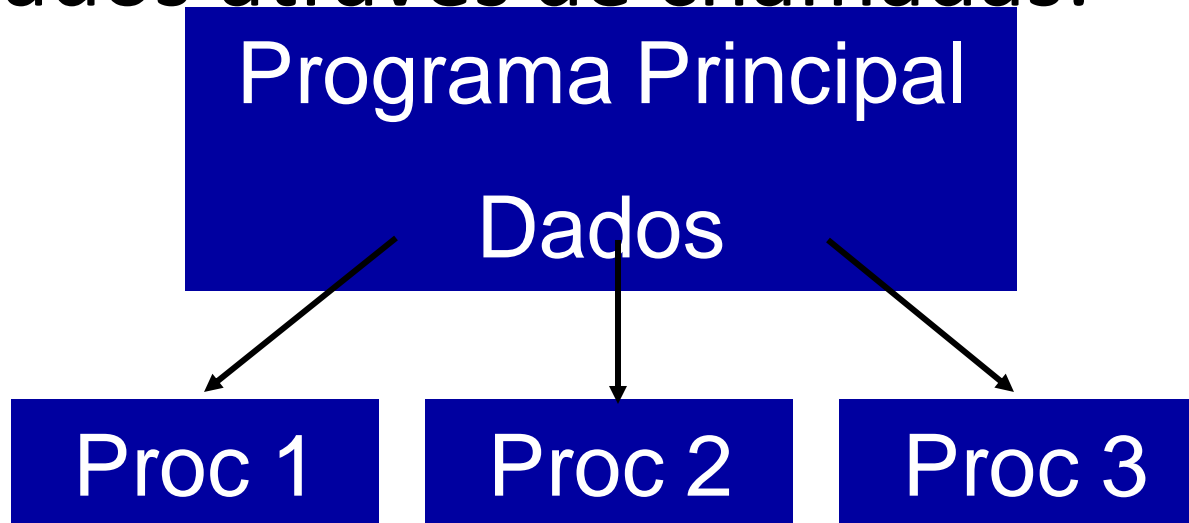
Programação Estruturada

Ênfase nos procedimentos, implementados em blocos estruturados, com comunicação entre procedimentos por passagem de dados;



Programação Estruturada

Na programação estruturada o código é composto por vários processos ligados através de chamadas.



Exemplo:

Linguagens estruturadas(**C, Pascal, Cobol, Basic, Clipper**).

Programação Orientada a Objetos

- Diminuiu a distância entre mundo real e solução computacional;
- Baseada em Classes e Objetos;
- Métodos e Atributos;
- Programação se dá pela **comunicação entre objetos**.

Programação Orientada a Objetos

Vantagens

- Melhor organização do código ;
- Bom reaproveitamento de código.

Programação Orientada a Objetos

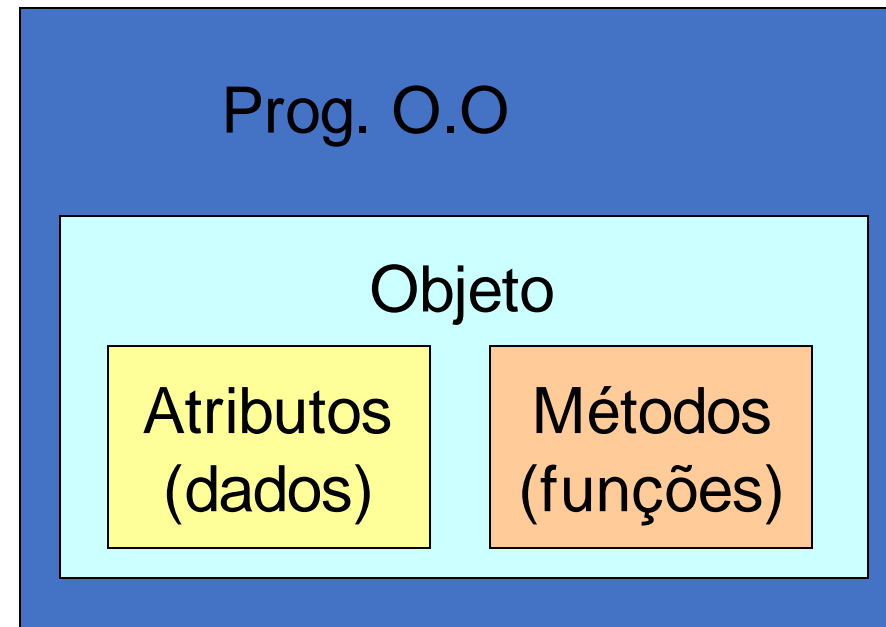
Desvantagens

- Mais difícil compreensão.

Programação Orientada a Objetos

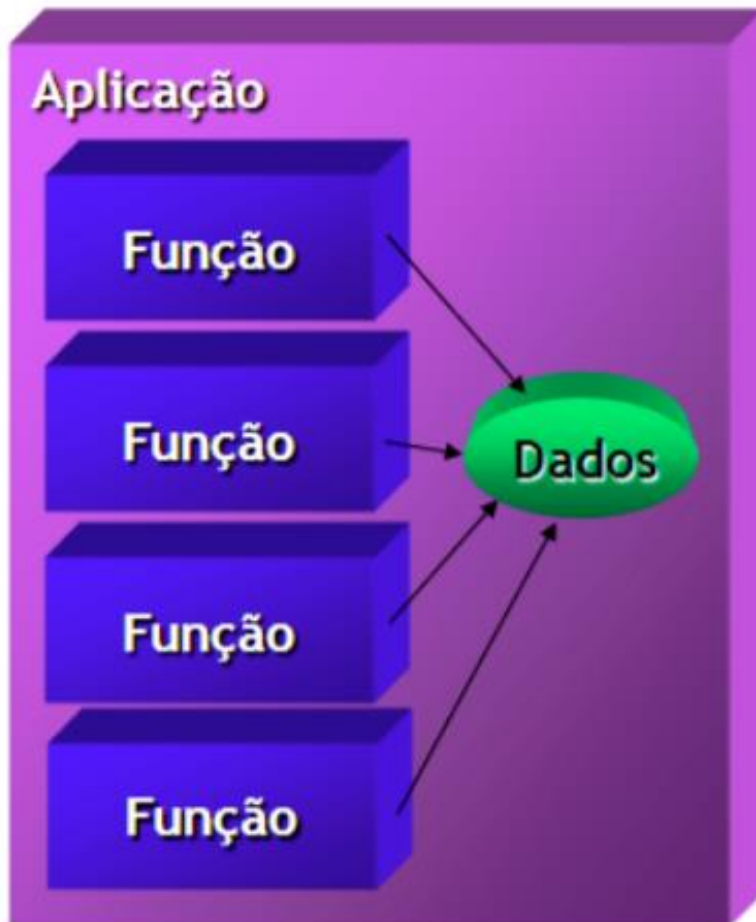
Dados e procedimentos fazem parte de um só elemento básico (objeto).

Os elementos básicos comunicam-se entre si por mensagens e tem ênfase nos dados e no agrupamentos dos mesmos.

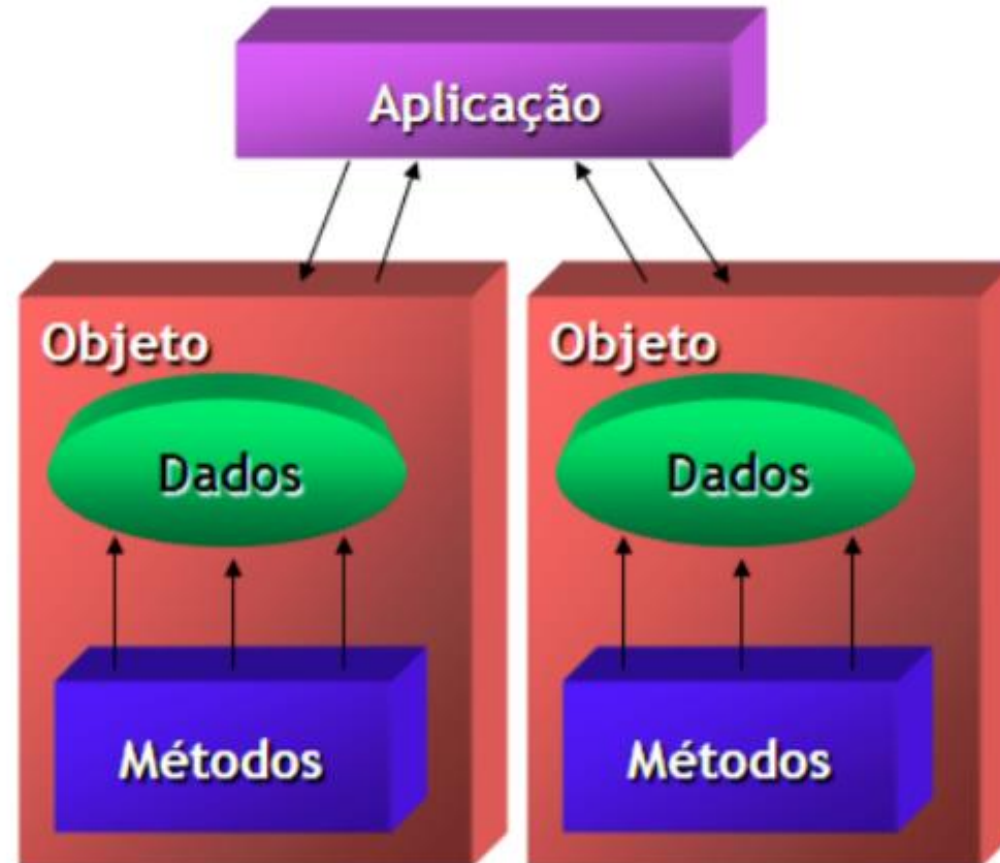


Programação Orientada a Objetos x Estruturada

Estruturada



Orientação a Objetos



Conceitos da Programação Orientada a Objetos

Abstração:

- Habilidade de se concentrar nos aspectos essenciais do sistema, ou um contexto qualquer, ignorando o que é supérfluo;

Conceitos da Programação Orientada a Objetos

Objeto:

- Representação computacional de algo do mundo real
 - Concreto
 - Abstrato

Conceitos da Programação Orientada a Objetos

Abstração

- Transformar aquilo que observamos realidade para a virtualidade.

Conceitos da Programação Orientada a Objetos

Objetos concretos

- Cão
- Moto
- Casa

Objetos abstratos

- Música
- Transação Bancária

Conceitos da Programação Orientada a Objetos

Modelo

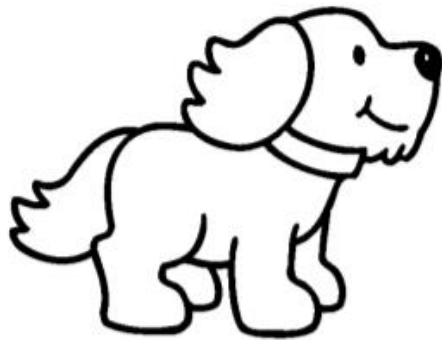
- Características + Comportamento

Estado - Atributos (Características)

Operações - Métodos (Comportamentos)

Conceitos da Programação Orientada a Objetos

Atributos e Métodos



▶ Atributos

- Raça: Poodle
- Nome: Rex
- Peso: 5 quilos

▶ Método

- Latir
- Comer
- Dormir



- Potência: 500cc
- Modelo: Honda
- Ano: 1998

- Acelerar
- Frear
- Abastecer

Conceitos da Programação Orientada a Objetos

Atividade

Cite 4 atributos de um aluno:

Cite 3 métodos de um aluno:



Por que usar Programação Orientada a Objetos

A metodologia Orientação a Objetos é **baseada** em “objetos do mundo real”, e por este motivo é mais intuitiva, pois oferece recursos como: **objetos e atributos, classes e membros, estruturas e componentes, ação e reação.**

Por que usar Programação Orientada a Objetos

Em um sistema orientado a objetos, os dados e todas as operações, que manipulam esses dados, são agrupados em estruturas chamadas de “**classes**”.

Por que usar Programação Orientada a Objetos

Os principais problemas do software hoje são:

- Diminuir o custo e o tempo da mudança;
- Aumentar a capacidade e facilidade de adaptação.

Por que usar Programação Orientada a Objetos


Objetos são especialmente bons para:


- Reduzir o tempo necessário para adaptar um sistema existente (reação mais rápida à mudanças no seu ambiente de negócio);
- Reduzir o esforço, a complexidade e os custos associados à mudança

Por que usar Programação Orientada a Objetos

O termo orientação a objetos significa organizar o mundo real como uma **coleção de objetos** que incorporam **estrutura de dados** e um **conjunto de operações** que manipulam estes dados.

Por que usar Programação Orientada a Objetos

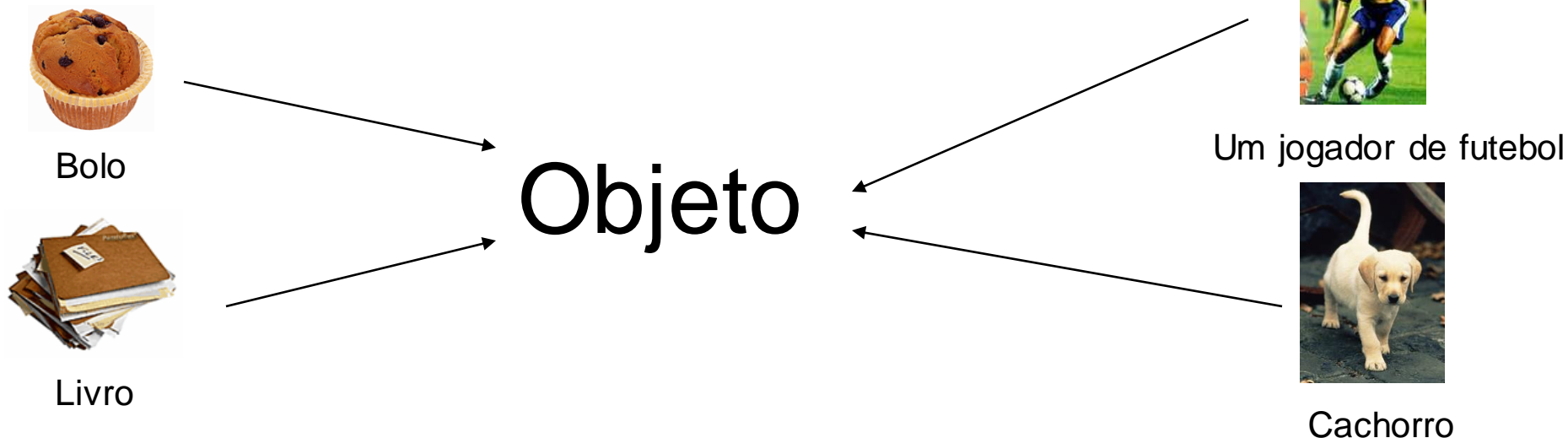
Classe: Pessoa	Propriedades	Comportamentos
	<p>Nome</p> <p>Profissão</p> <p>Data de Nascimento</p> <p>Peso</p> <p>Altura</p>	<p>Andar</p> <p>Correr</p> <p>Trabalhar</p> <p>Chorar</p> <p>Dançar</p>

Classe: Pássaro	Propriedades	Comportamentos
	<p>Espécie</p> <p>Cor das penas</p> <p>Tamanho</p> <p>Peso</p>	<p>Andar</p> <p>Correr</p> <p>Voar</p> <p>Pousar</p>

Por que usar Programação Orientada a Objetos

Lembrando que objetos são quaisquer coisas na natureza que possuam propriedades (características) e comportamentos (operações).

42



Por que usar Programação Orientada a Objetos

Um objeto tem dados e comportamento.

Atributos

Métodos



nome = Marina
cpf = 022.200.708-12
idade = 27

corre
dorme



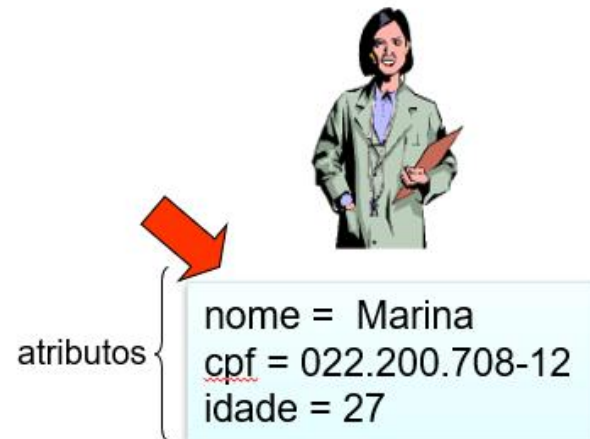
nome = Felipe
cpf = 039.217.908-22
idade = 42

corre
dorme

Atributos

São características presentes nos objetos;

- Os valores de todos os atributos é chamado de estado do objeto;
- Somente atributos que são de interesse do sistema devem ser considerados.





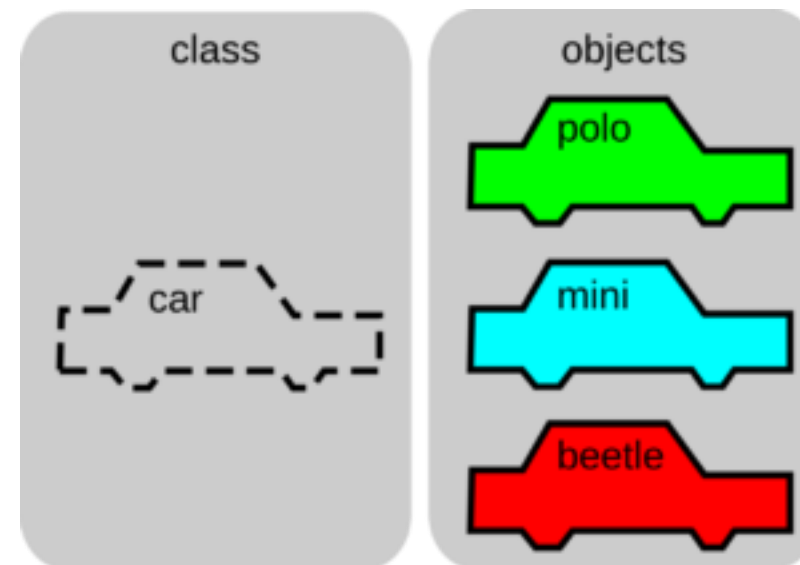
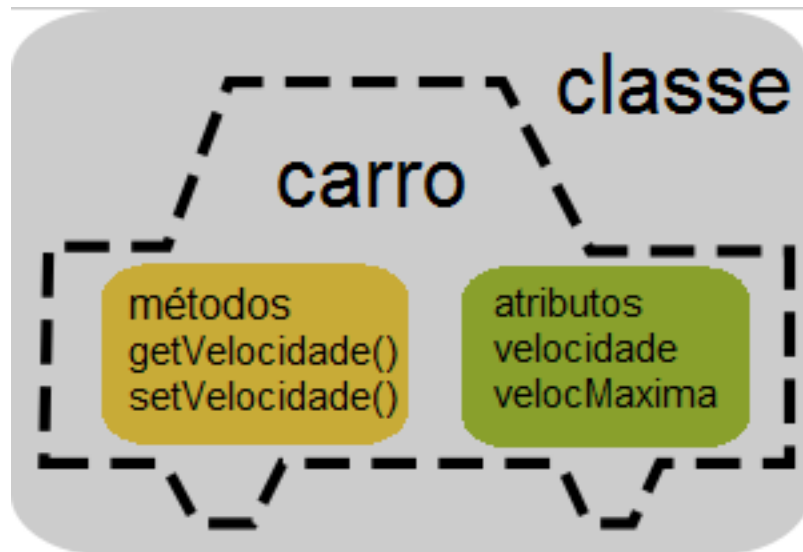
Atributos

São características presentes nos objetos;

- Para criarmos objetos em um sistema precisamos abstraí-lo criando uma classe que o represente.

Classes

As **Classes** e os **Objetos** possuem uma relação de dependência, pois não existe objeto sem a classe;





Classes

A **Classe** representa a **abstração das características comuns** mais relevantes (atributos e métodos) de um conjunto de objetos. A classe passa a ser um molde para se criar objetos do mesmo tipo.

Classes

*“Uma **classe é uma entidade** que descreve um conjunto de objetos com propriedades (atributos) e comportamentos (métodos) semelhantes e com relacionamentos comuns com outros objetos”*

Classes

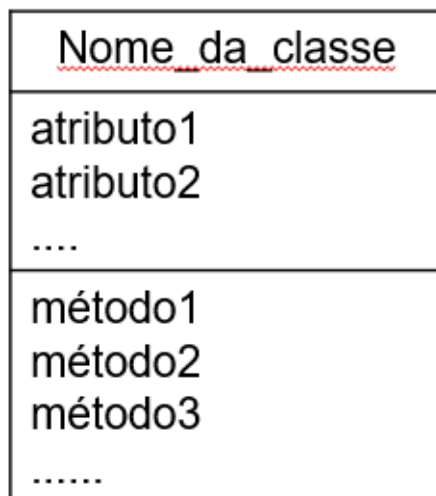
As classes são as partes mais importantes de qualquer sistema orientado a objetos.

Essas classes podem incluir abstrações que são parte do domínio do problema.



Classes

Graficamente, as classes são representadas por retângulos incluindo nome, atributos e métodos. Devem receber nomes representativos.

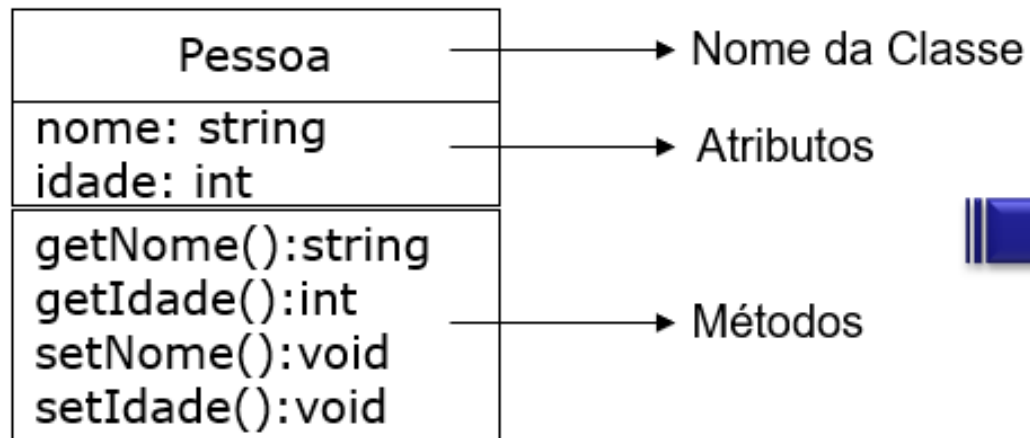


Representação em Diagrama
de Classe UML (Unified
Modelling Language)

Classes

Exemplo:

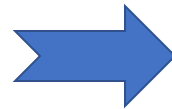
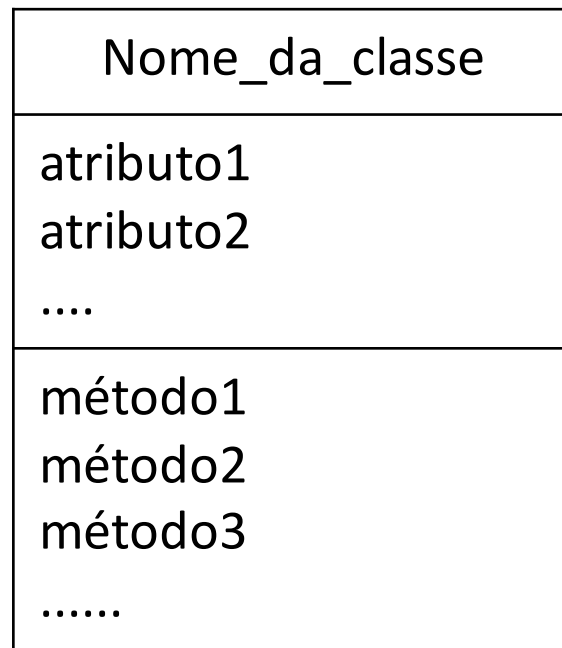
A classe Pessoa deverá ter atributos e métodos comuns



Representação em Diagrama de Classe UML (Unified Modelling Language)

Classes

Construindo classes em Java:



```
public class <Nome_da_classe> {
    <lista de atributos>
    <lista de métodos>
}
```



Um arquivo em Java precisa ser uma classe pública com o **mesmo nome do arquivo**

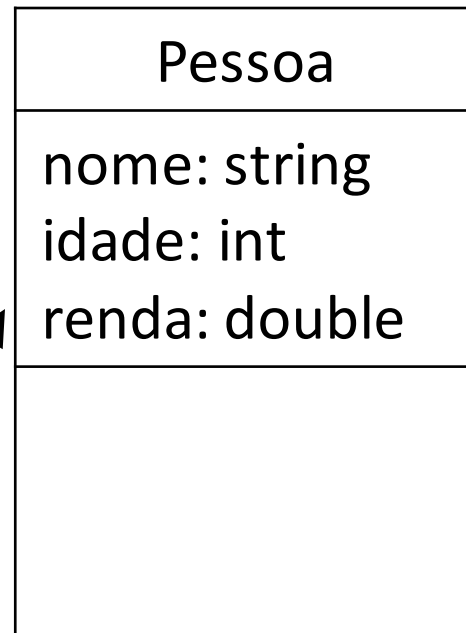
Modelo UML de classe

Classes

1 Pessoa



2

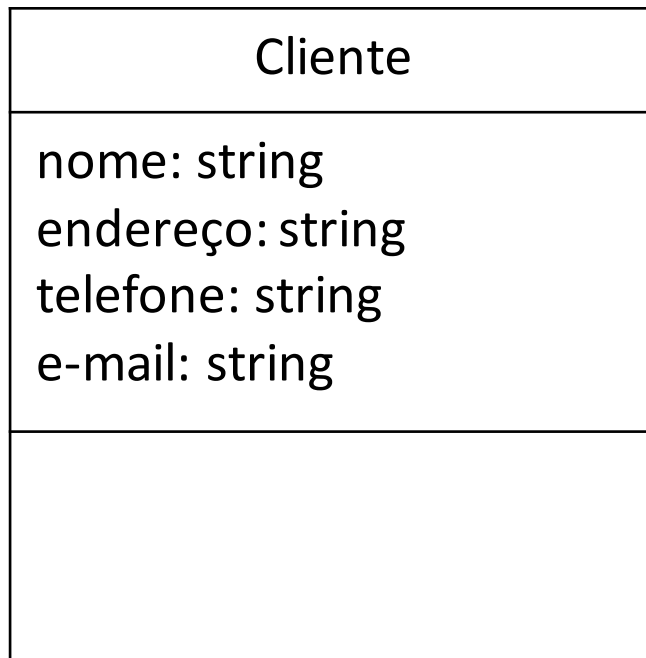


3

```
public class Pessoa {  
    //lista de atributos  
    String nome;  
    int idade;  
    double renda;  
  
    //lista de métodos  
  
}
```

Classes

Diagrama UML

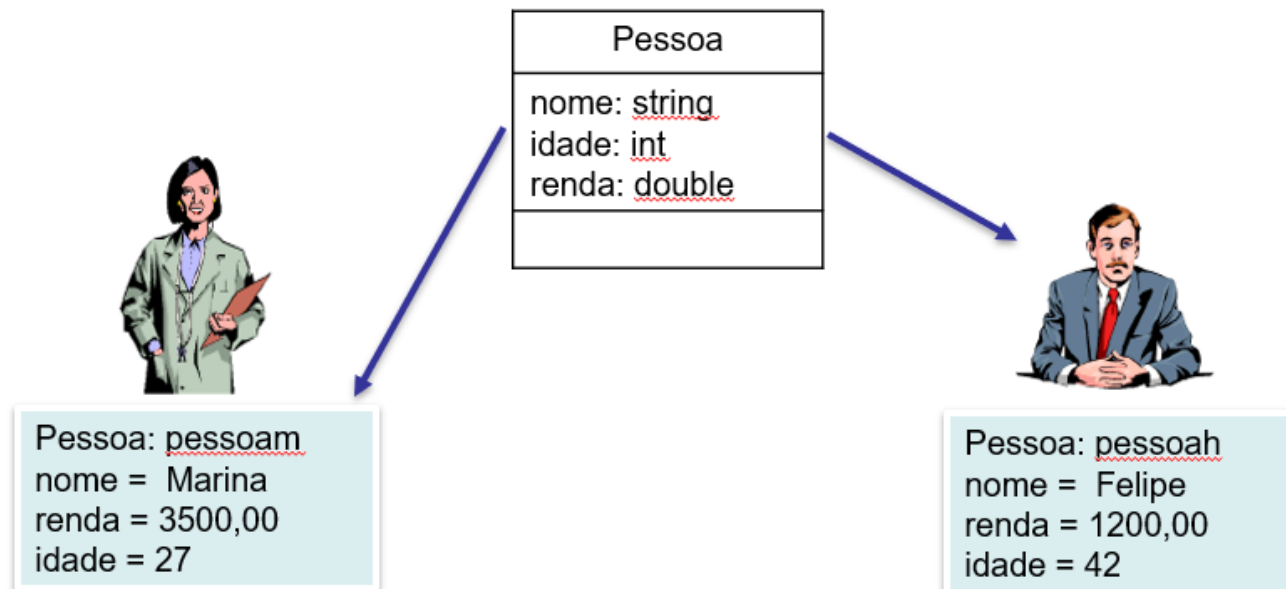


Classe em Java

```
public class Cliente {  
    //atributos  
    String nome;  
    String endereco;  
    String telefone;  
    String email;  
  
    //lista de métodos  
  
}
```

Instanciando um Objeto

A classe é um modelo para os objetos do sistema, desta forma após a definição da classe podemos instanciar (criar) objetos que serão do tipo da classe. Eles são chamados de instâncias.



Instanciando um Objeto

Classe (iniciais em maiúsculas)

```
public class Pessoa {  
    //lista de atributos  
    String nome;  
    int idade;  
    double renda;  
}
```

Atributos
(em letras minúsculas)

```
public class UsaClasses {  
  
    public static void main(String[] args) {  
        //Pessoa p = new Pessoa();  
        Pessoa p;  
        p = new Pessoa();  
        Pessoa p1 = new Pessoa();  
  
        p.nome = "Fulano";  
        p.idade = 25;  
        p.renda = 1000;  
  
        System.out.println("Nome: " + p.nome);  
        System.out.println("Idade: " + p.idade);  
        System.out.println("Renda: " + p.renda);  
  
        System.out.println("Nome: " + p1.nome);  
        System.out.println("Idade: " + p1.idade);  
        System.out.println("Renda: " + p1.renda);  
    }  
}
```

Objetos (em letras minúsculas)

```
public class UsaClasses {
```

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

```
        //Pessoa p = new Pessoa();
```

```
        Pessoa p;
```

```
        p = new Pessoa();
```

```
        Pessoa p1 = new Pessoa();
```

```
        p.nome = "Fulano";
```

```
        p.idade = 25;
```

```
        p.renda = 1000;
```

```
        System.out.println("Nome: " + p.nome);
```

```
        System.out.println("Idade: " + p.idade);
```

```
        System.out.println("Renda: " + p.renda);
```

```
        System.out.println("Nome: " + p1.nome);
```

```
        System.out.println("Idade: " + p1.idade);
```

```
        System.out.println("Renda: " + p1.renda);
```

```
    }
```

Método principal por onde o Java começa a execução do programa

Instanciando 2 objetos da classe Pessoa. Cada um com o seu identificador

Estamos criando o estado do objeto p1, acessando os seus atributos através de um ponto após o nome do objeto

Podemos também acessar os atributos e recuperar o estado dos objetos para imprimir na tela o seu conteúdo

OBS: Os arquivos de classe UsaClasses.java e Pessoa.java precisam estar na mesma pasta

Classes

Outros exemplos:

Triangulo
base: float altura: float

Data
dia: int mes: int ano: int

Curso
nome: string qtdealunos: int turma: string

Exercícios

01-) Crie um diagrama de classes UML para abstrair os atributos dos seguintes objetos:

- a) Eletrodoméstico
- b) Carro
- c) Caixa de Diálogo

Exercícios

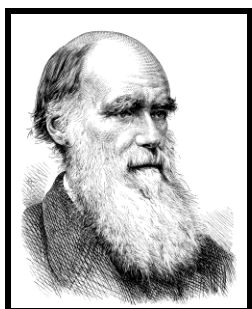
01-) Crie um diagrama de classes UML para abstrair os atributos dos seguintes objetos:

- a) Eletrodoméstico
- b) Carro
- c) Caixa de Diálogo

02-) Implemente as classes do exercício acima (cada uma em um arquivo).

01-) Crie uma classe com o método main e instancie um objeto de cada uma das classes acima, coloque valores nos atributos e os mostre na tela.

“Não é o mais forte
que sobrevive, nem o
mais inteligente, mas
o que melhor se
adapta as mudanças”



Charles Darwin

Obrigado!

Se precisar ...

Prof. Claudio Benossi

Claudio.benossi@fatec.sp.gov.br

