Programação Orientada à Objetos

POO - Conceitos Gerais

Paradigmas

- Programação Estruturada ou Procedural
  - Baseada em uma sequencia específica de passos que o software deve seguir para atingir um objetivo
  - Forte dependência de funções que podem depender de outras funções

- Programação Orientada a Objetos (POO ou OOP)
  - Estratégia em que elementos existentes em um software são representados como objetos com características e comportamentos
  - Estas características e comportamentos em comum podem ser agrupadas, facilitando sua coesão e manutenção

Vantagens & Desvantagens da POO

- Vantagens
  - Possibilidade de representação relacionada ao mundo real
  - Reutilização de código
  - Leitura e manutenção de código
  - Consistência e segurança

- Desvantagens
  - Desenvolvimento mais demorado
  - Complexidade maior na implementação

Pilares

1. Abstração
2. Encapsulamento
3. Herança
4. Polimorfismo

Abstração

Classes

- Agrupamentos de variáveis (propriedades) e funções (métodos)
- Funcionam como modelos para objetos
- Descrevem como um objeto deve ser

Objetos

- Também conhecidos como “instâncias”
- São criados (instanciados) a partir de uma Classe previamente definida
- Objetos podem utilizar todas as propriedades e métodos existentes em sua Classe

Propriedades

- Estruturas de dados que representam a Classe
- Também conhecidos como “atributos”
- Definem as características da Classe e dos objetos derivados dela

Cliente

```
+nome: string
+email: string
+senha: string
+telefones: array
```

Métodos

- Funções que determinam as “habilidades” da Classe
- Também conhecidos como “comportamentos”
- Definem as ações que a Classe e seus objetos derivados podem fazer

Cliente

```
+nome: string
+email: string
+senha: string
+telefones: array
+exibirDados()
```

Construtor

- Método especial que será “chamado” toda vez em que um objeto for **criado** a partir da classe onde o construtor for declarado

- Pode ser utilizado para inicializar propriedades e outros métodos no momento da criação de um objeto

Encapsulamento

Encapsulamento e Visibilidade

- Mecanismo para controlar a forma de acesso à propriedades e métodos de uma Classe/Objeto

- O acesso às propriedades de um objeto é feito e controlado através de ***getters*** e ***setters*** públicos, sem a necessidade de expor a estrutura interna

Encapsulamento e Visibilidade

Cliente

```
-nome: string
-email: string
-senha: string
-telefones: array
+exibirDados()
+setNome(nome:string)
+setEmail(email:string)
+setSenha(senha:string)
+setTelefones(telefones:array
+getNome(): string
+getEmail(): string
+getSenha(): string
+getTelefones(): array
```

Tipos de visibilidade:

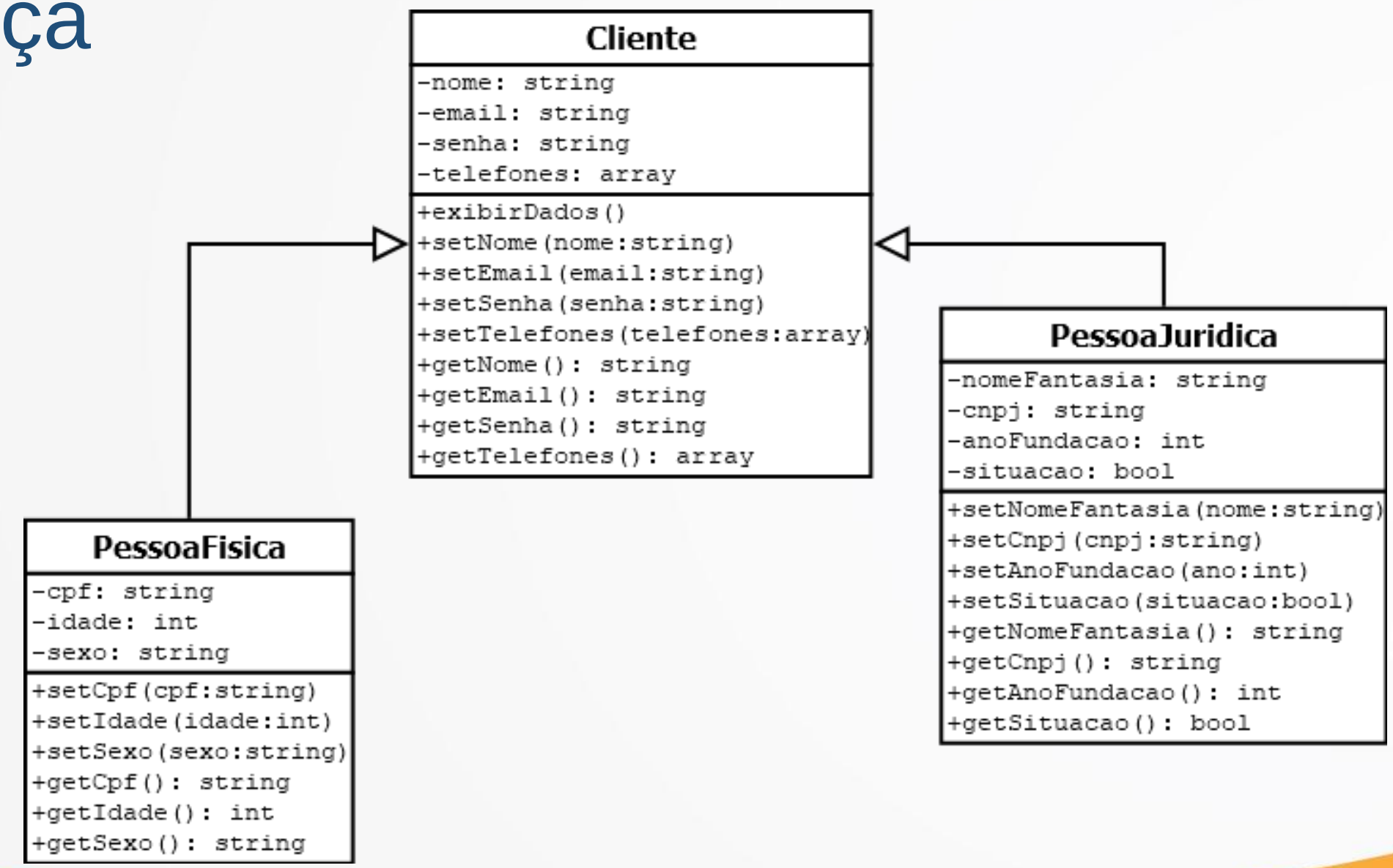
+público (padrão)
-privado
#protegido

Herança

Herança

- Mecanismo que possibilita a uma classe estender propriedades e métodos para outras classes.
- A classe de primeiro nível é chamada de “superclasse”
- As classes para as quais são estendidas funcionalidades da superclasse são conhecidas como “subclasses”

Herança


Cliente
-nome: string
-email: string
-senha: string
-telefones: array
+exibirDados()
+setNome(nome:string)
+setEmail(email:string)
+setSenha(senha:string)
+setTelefones(telefones:array)
+getNome(): string
+getEmail(): string
+getSenha(): string
+getTelefones(): array
PessoaFisica
-cpf: string
-idade: int
-sexo: string
+setCpf(cpf:string)
+setIdade(idade:int)
+setSexo(sexo:string)
+getCpf(): string
+getIdade(): int
+getSexo(): string
PessoaJuridica
-nomeFantasia: string
-cnpj: string
-anoFundacao: int
-situacao: bool
+setNomeFantasia(nome:string)
+setCnpj(cnpj:string)
+setAnoFundacao(ano:int)
+setSituacao(situacao:bool)
+getNomeFantasia(): string
+getCnpj(): string
+getAnoFundacao(): int
+getSituacao(): bool

Polimorfismo

Polimorfismo

- Mecanismo que possibilita comportamentos diferentes para um mesmo método

- Possibilidade de sobrescrever o comportamento de um método já definido

Polimorfismo

Exemplos