PROGRAMAÇÃO ORIENTADA A OBJETOS

POO

Introdução

Prof. Anderson Henrique

TECH
soluções em ti


Programação Estruturada
Dados Globais
Procedimento
Procedimento
Procedimento
Procedimento
...
Procedimento


Programação Orientada a Objetos
Dados Objeto
Método
...
Método
Dados Objeto
Método
...
Método
...
Dados Objeto
Método
...
Método

Programação Orientada a Objetos?

DIFICULDADES

- Complexidade no aprendizado em comparação com a programação estruturada
- Seus conceitos são de difícil compreensão

BENEFICIOS

- Mais fácil descrever o mundo real através dos objetos
- O encapsulamento facilita a manutenção do código
- Maior facilidade para reutilização de código

Conceitos de Orientação a Objetos

- POO – Programação Orientada a Objetos;
- Objetivo – Aproximar o mundo digital do mundo real;
- Como era? Programação Baixo Nível-Programação Linear-Programação Estruturada - Programação Modular - POO
- Quem criou? Alan Kay 1970 (Era formado em Matemática e Biologia);
- Programação Estruturada – Programação Modular (Dados Globais) -> Procedimentos | Dados de Objetos -> Métodos

Vantagens

- Confiável -> O isolamento entre as partes gera software seguro. Ao alterar uma parte, nenhuma outra é afetada;
- Oportuno -> Ao dividir tudo em partes, várias delas podem ser desenvolvidas em paralelo;
- Manutenível -> Atualizar um software é fácil. Uma pequena modificação vai beneficiar todas as partes que usarem o objeto;
- Extensível -> O software não é estático. Ele deve crescer para permanecer útil;
- Reutilizável -> Podemos usar objetos de um sistema que criamos em outro sistema futuro;

- Natural -> Mais fácil de entender. Você se preocupa mais na funcionalidade do que nos detalhes de implementação;

Classes

É uma estrutura estática utilizada para descrever objetos mediante atributos e métodos. A classe é um Modelo (Model) ou Template para criação desses objetos. Ela é representada pelo operador (class).

Atributos

São as características que definem o que o objeto é, os atributos são as variáveis da classe, poderão armazenar valores que serão manipulados através dos métodos da classe

Ex.: **atributo**: $Nome

Pode ser manipulada através do **método**: alterarNome($nome)

TECH
soluções em ti

Métodos

São as funcionalidades da classe, servem para manipular os dados de um ou mais atributos.

Os métodos podem ser declarados na classe ou podem ser declarados e implementados

Ex.: **function** Crescer($centimetros);

```
function Crescer($centimetros){
    aqui vem a implementação...
}
```

TECH
soluções em ti

Objeto

- Coisa material ou abstrata que pode ser percebida pelos sentidos e descrita por meio das suas **características**, **comportamentos** e **estado atual**.
- Objeto caneta (**Coisas que eu tenho** [Modelo, Cor, Ponta, Carga, Tampada], **Coisas que eu faço** [Escrever, Rabiscar, Pintar, Tampar, Destampar], **Como eu estou agora** [50% de Carga, Destampada, Escrevendo]).

```
Classe Caneta
        Atributos
                modelo: Caractere
                cor: Caractere
                ponta: Real
                carga: inteiro
                tampada: lógico
        Métodos
                Método rabiscar( )

                FimMétodo
                Método tampar( )

                FimMétodo
FimClasse
```

TECH
soluções em ti

Como podemos criar objeto?

Classe

c1 = new Caneta
c1.cor = “Azul”
c1.ponta = 0.5
c1.tampada = false

instanciar

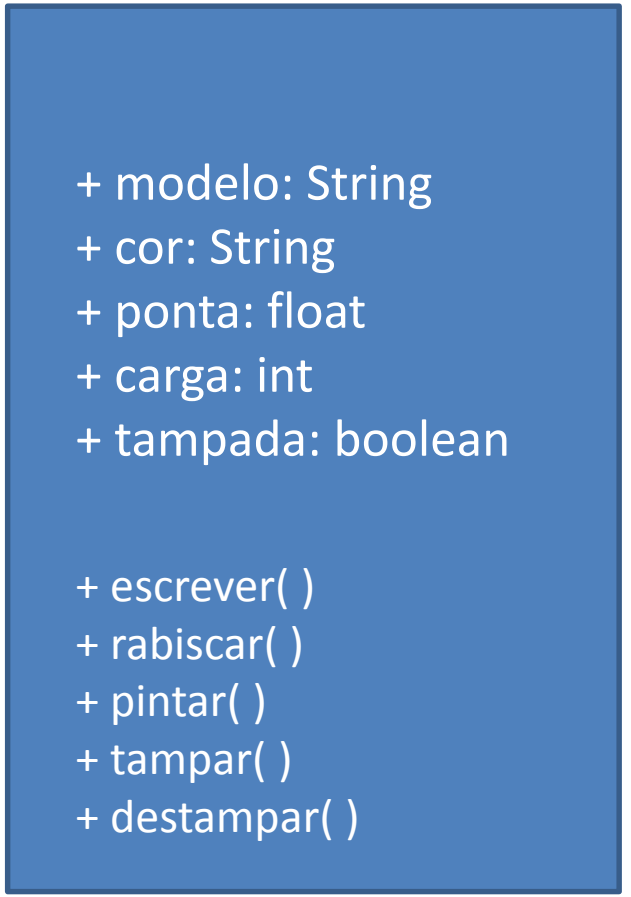
c2 = new Caneta
c2.cor = “Vermelho”
c2.ponta = 1.0
c2.tampada = true

Objeto

- **Classe** -> Define os atributos e métodos comuns que serão compartilhados por um objeto.
- **Objeto** -> É a instância de uma classe.

Linguagem de Modelagem Unificada (UML)

Não veremos de forma aprofundada sobre essa temática, no entanto, precisamos utilizar o Diagrama de Classes.



TECH
soluções em ti

Modelando a classe Pessoa

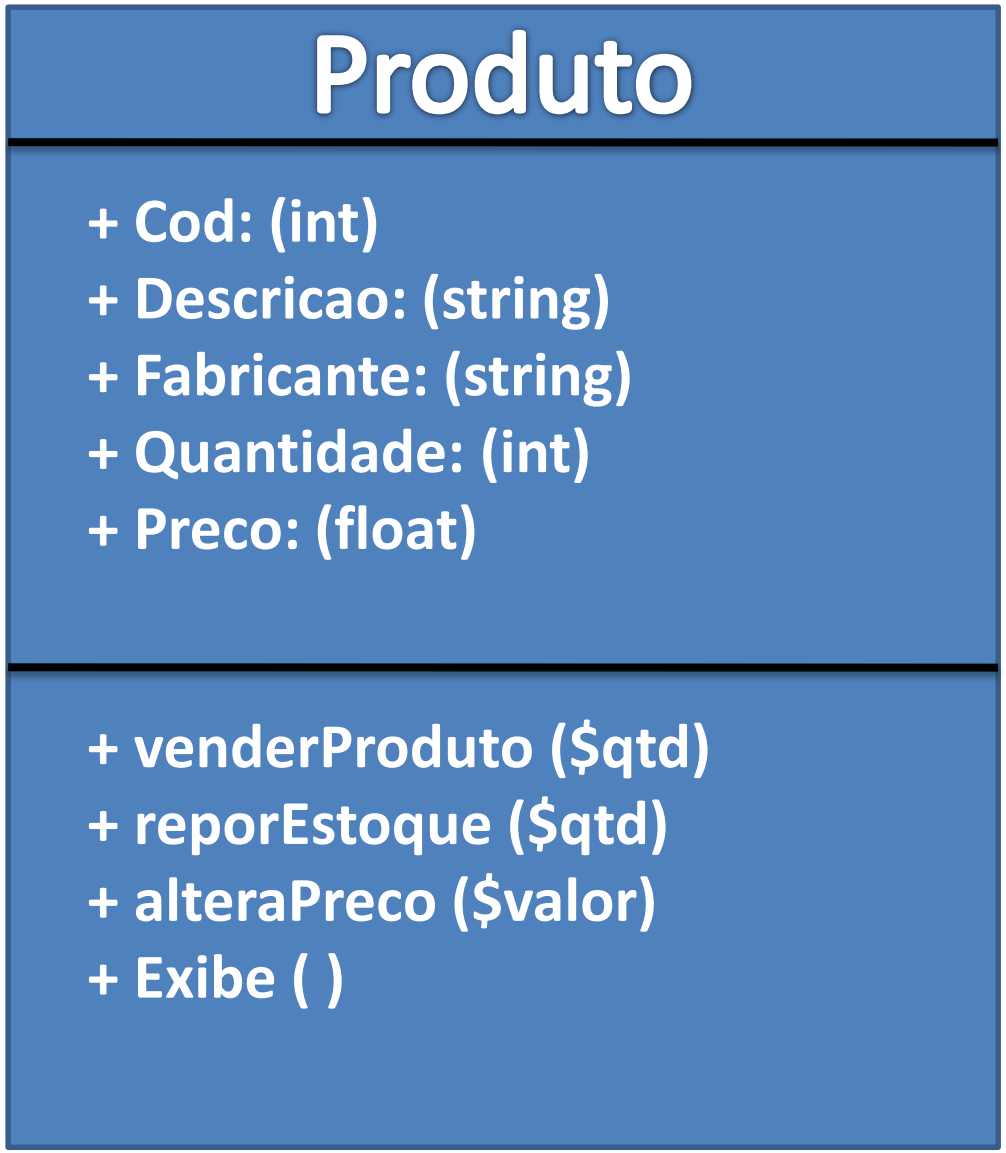
Nome da Classe

Propriedades

Métodos

TECH
soluções em ti

Modelando a classe Produto



→ Nome da Classe

→ Propriedades

→ Métodos

TECH
soluções em ti

Construtor

É um método especial utilizado para definir o comportamento inicial de um objeto, ou seja, o comportamento no momento da criação. É um método executado automaticamente.
Utiliza-se o operador __construct(), pode ser gerado no netbeans com o atalho (**ALT + Insert**)

TECH
soluções em ti

Destrutor

É um método especial executado automaticamente quando o objeto é desalocado da memória, quando atribuímos o valor **NULL** ao objeto, quando utilizamos a função **unset( )**, ou, em último caso, quando o programa é **finalizado**. Utilizado para finalizar conexões...

Classe SPL

Podemos utilizar a função spl_autoload_extensions() para carregar as extensões dos arquivos, require_once faz a requisição de um arquivo externo (precisa indicar o caminho do arquivo com o nome da classe e sua extensão), e por fim, utilizamos a função spl_autoload_register() para informar qual função deve ser utilizada para carregar as classes. Exemplo:

```
function myAutoload($className){
        $extension = spl_autoload_extensions();
        require_once (_DIR_ . '/' . $className . $extension);

 }

spl_autoload_extensions('.class.php');
spl_autoload_register('myAutoload');
```

SPL significa (**Standard PHP Library**) é a forma utilizada no PHP Moderno, a função mágica __autoload() foi depreciada deixando de funcionar.

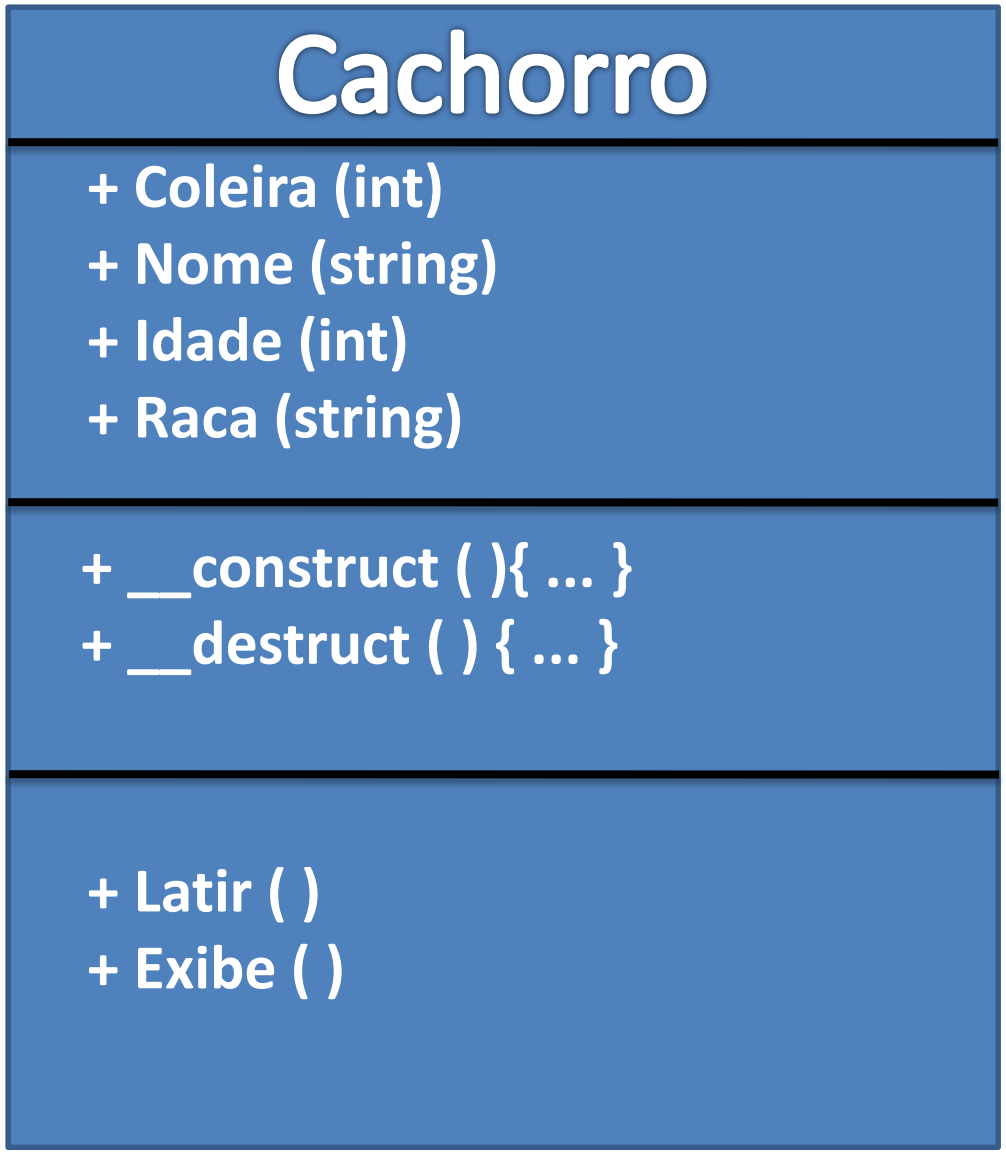
Utilizando uma classe:

```
Class Autoload{

        public function __construct(){
                spl_autoload_extensions('.class.php');
                spl_autoload_register(array($this, 'load'));
        }

        private function load($className){
                $extension = spl_autoload_extensions();
                require_once (__DIR__ . '/'. $className . $extension);
        }
}
```

Utilizando a classe Autoload:

```
$autoload = new Autoload();

$objectA = new ClassA();
$objectB = new ClassB();
```

Modelando a classe Cachorro



→ Nome da Classe

→ Propriedades

→ Construtor
Destrutor

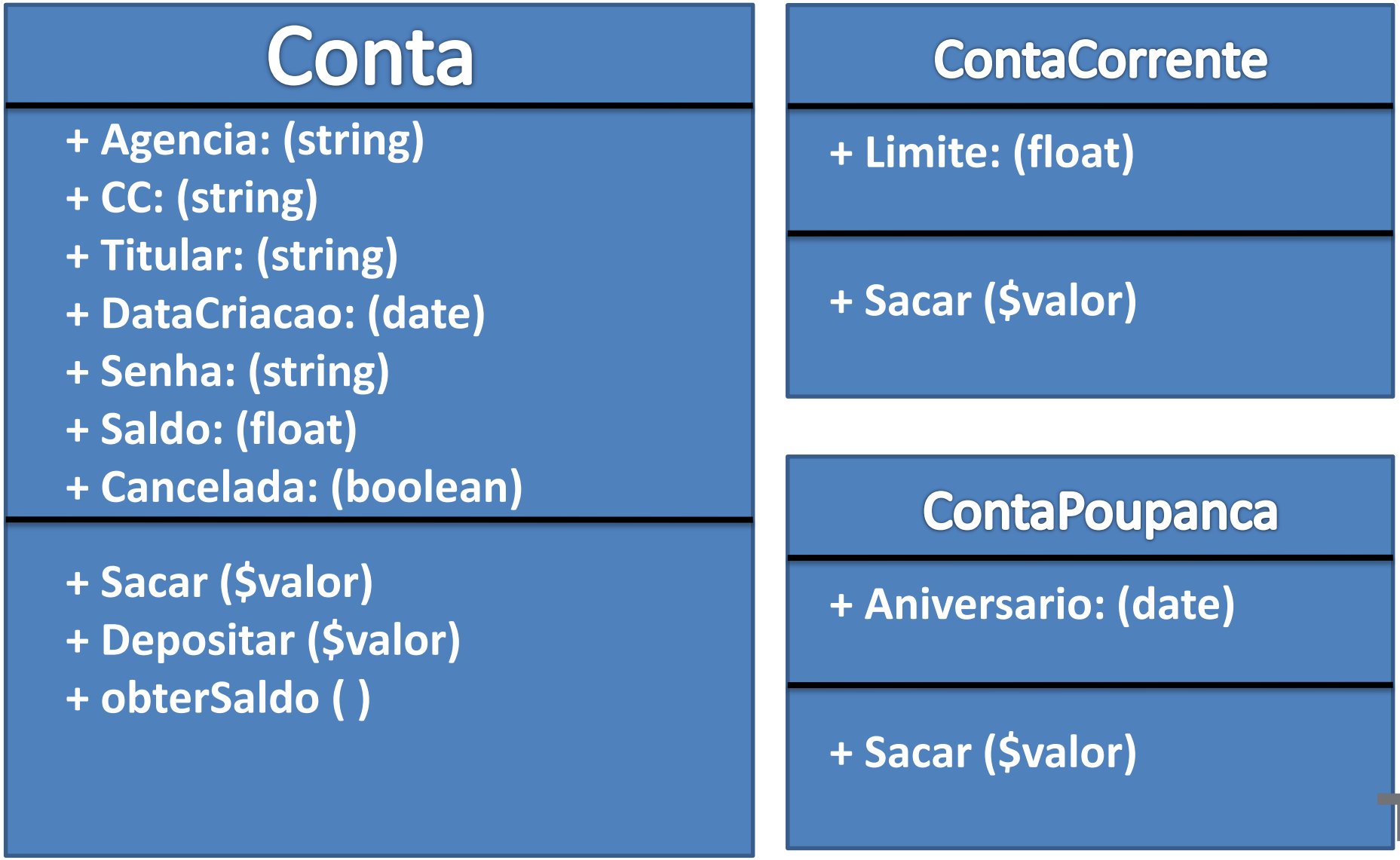
→ Métodos

TECH
soluções em ti

Herança

A possibilidade de reutilizar partes de código já definidas é o que nos dá maior agilidade no dia-a-dia, além de eliminar a necessidade de eventuais duplicações ou reescritas de código.
Os atributos e os métodos declarados na classe-pai são herdadas em todas as classes-filhas que dela se estendem

As classes-filha ContaCorrente e ContaPoupanca



TECH
soluções em ti

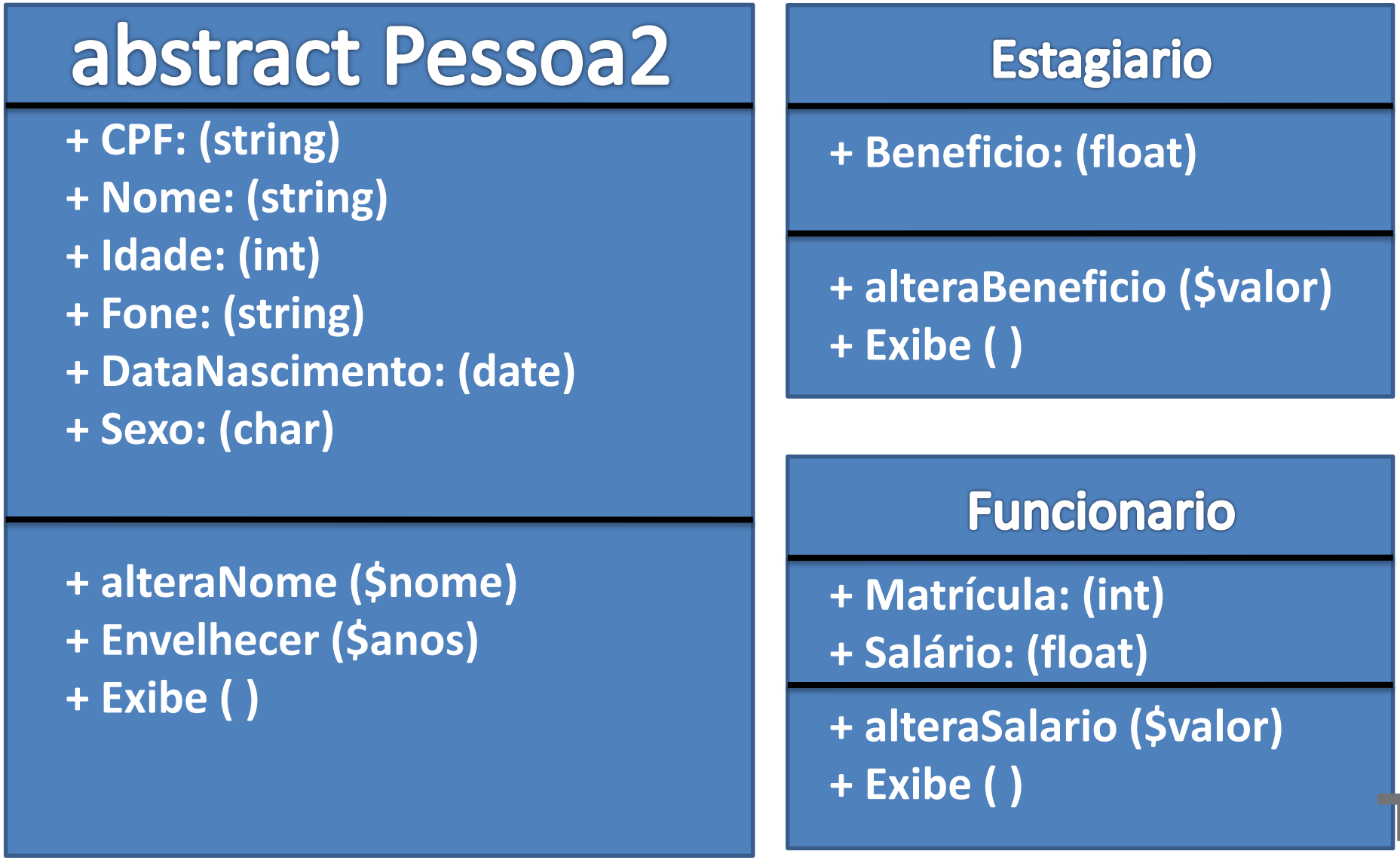
Polimorfismo

Classes derivadas de uma mesma classe-pai tenham métodos sobrescritos (overridindg), mas comportamentos diferentes, redefinidos em cada uma das classes-filha.
O método Sacar( ) na classe ContaPoupanca verifica somente se há saldo, na ContaCorrente verifica se está dentro do limite, além de debitar o imposto (CPMF)

Abstração

Classes estruturais, que na hierarquia de classes servem de base para outras. São classes que nunca serão instanciadas na forma de objetos, somente as suas filhas serão. Marcamos essas classes como abstratas (abstract). No exemplo da classe Conta, esta jamais poderá ser instanciada, então a marcamos como **abstract** class{ }

As classes-filha Funcionario e Estagiario

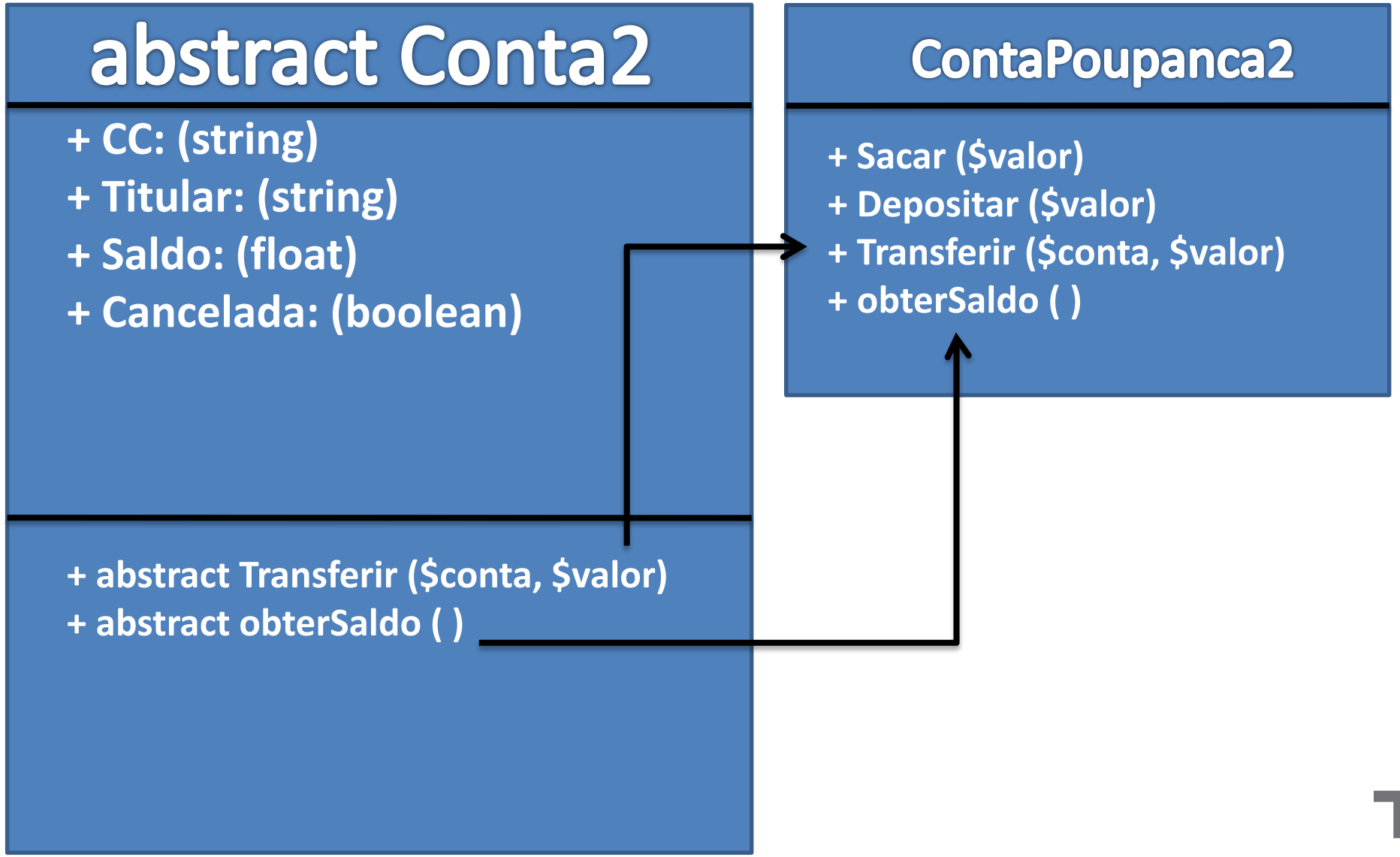


TECH
soluções em ti

Método abstrato

São métodos declarados na classe abstrata sendo somente implementados nas classes-filha. Métodos abstratos somente podem ser declarados em classe abstrata, sua implementação nas classes-filha se torna obrigatória.

A classe-filha ContaPoupanca2 estende da classe-pai Conta2



TECH
soluções em ti

Classe final

A classe final não pode ser uma superclasse, ou seja, não pode ser base em uma estrutura de herança. Se definirmos uma classe como final pelo operador final, ela não poderá mais ser especializada (herdada) em classes-filha.

Método final

Um método final não pode ser sobrescrito, ou seja, não pode ser redefinido nas classes-filha. Trabalhando com herança, os métodos definidos como finais não poderão sofrer o overriding.

Encapsulamento

Um mecanismo que provê proteção de acesso aos membros internos de um objeto. Lembre-se que uma classe possui responsabilidade sobre os atributos que contém.

Podemos representar o encapsulamento de acordo com a imagem a seguir:

Modificadores de Acesso

Indicam o nível de acesso aos componentes internos de uma classe (atributos e métodos).

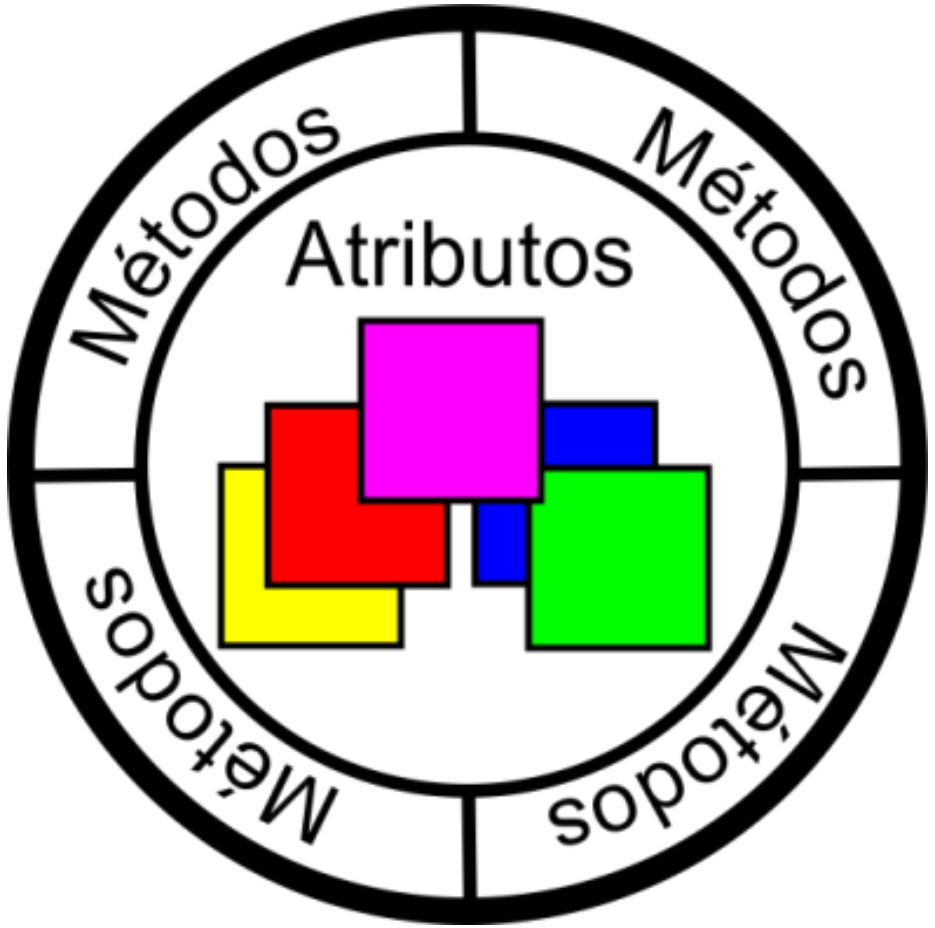
Na UML, estes modos de visibilidade são representados da seguinte forma:

+ public (público)

- private (privado)

protected (protegido)

Podemos definir a visibilidade dos atributos e dos métodos de um objeto. A visibilidade define a forma como essas atributos devem ser acessados



TECH
soluções em ti

Visibilidade	Descrição
Private (-)	Membros declarados como **private** somente podem ser acessados dentro da própria classe em que foram declarados.
Protected (#)	Membros declarados como **protected** somente podem ser acessados dentro da própria classe e a partir de classes descendentes (classes-filha), mas não poderão ser acessados a partir do programa que faz uso dessa classe.
Public (+)	Membros declarados como **public** poderão ser acessados livremente a partir da própria classe, a partir de classes descendentes e a partir do programa que faz uso dessa classe.

Membros da classe

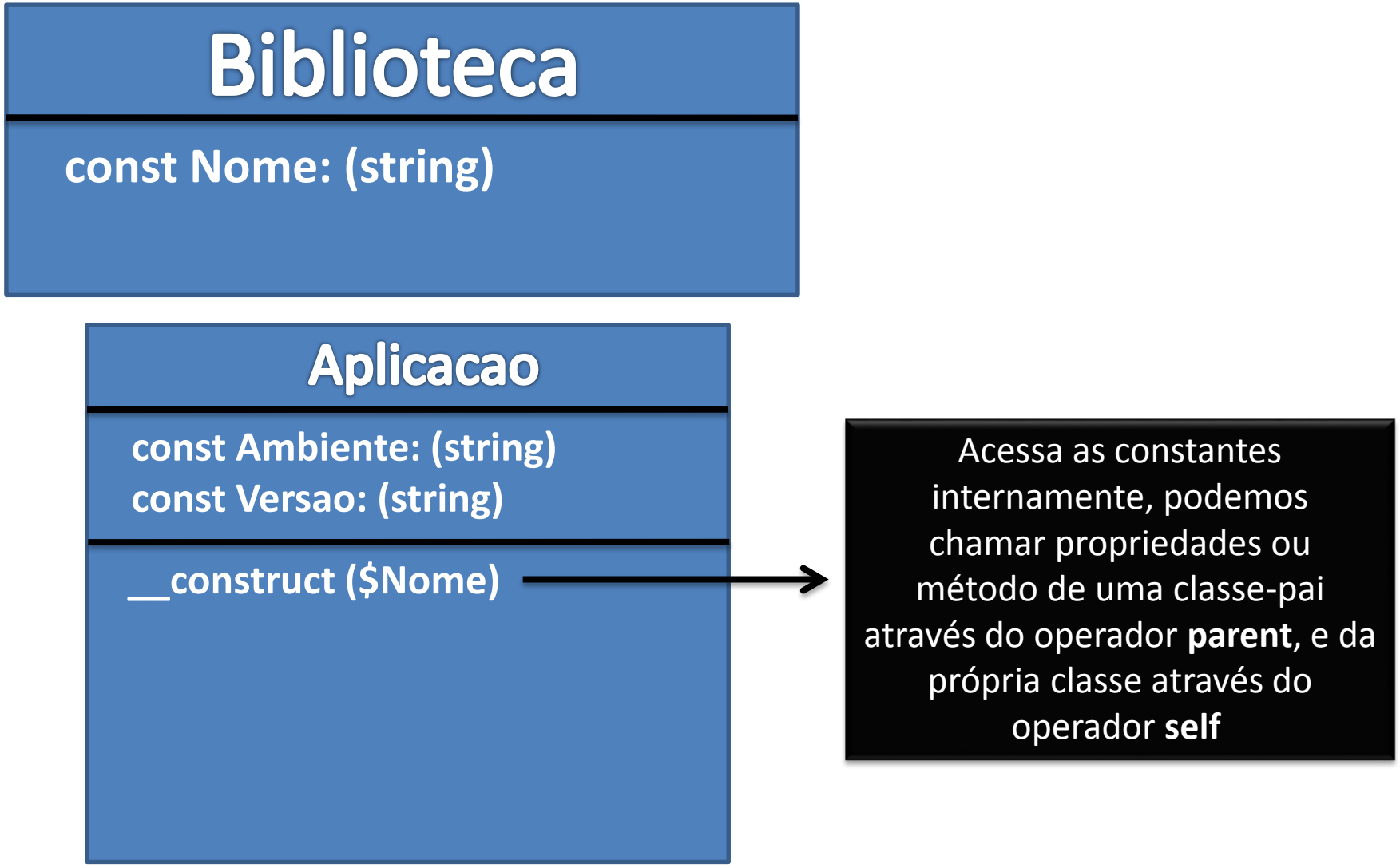
A classe é uma estrutura padrão para a criação de objetos. A classe permite que armazenemos valores nela de duas formas:

- Constantes de classe
- Propriedades estáticas

Constantes de classe

Declaramos uma constante de classe utilizando o operador (**const**), podemos acessá-la externamente pela sintaxe: **NomeDaClasse::NomeDaConstante** e internamente pela sintaxe: **self::NomeDaConstante**

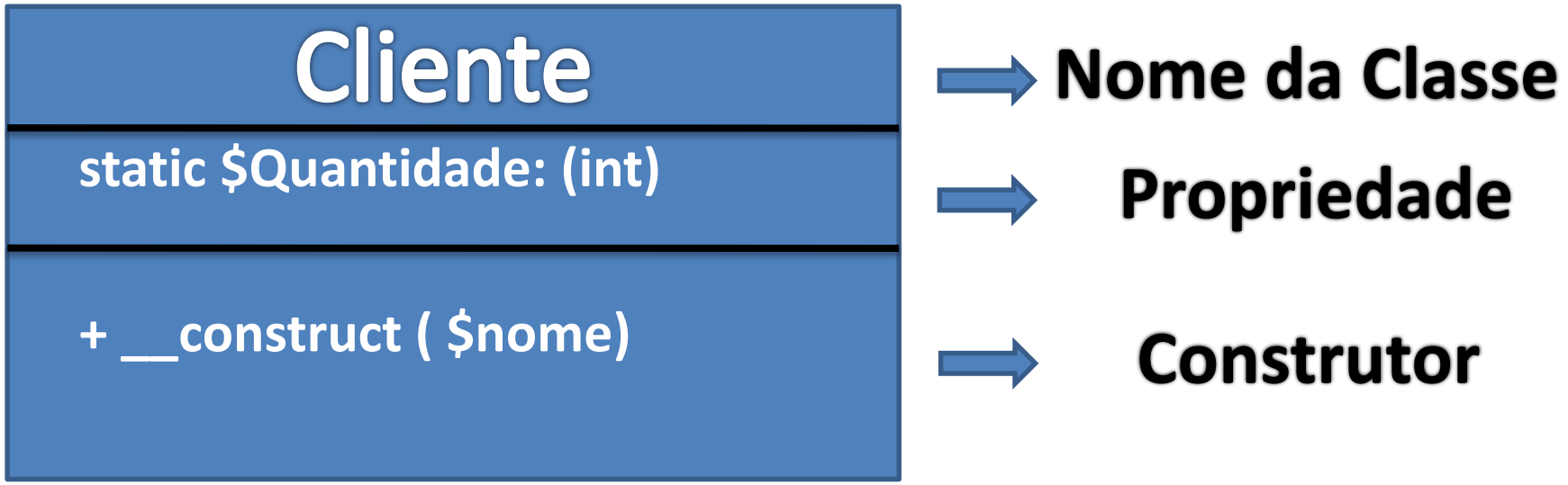
A classe-filha Aplicacao estende da classe-pai Biblioteca



Atributo estático

São atributos de um objeto. São dinâmicos como os atributos de um objeto, mas estão relacionados à classe. Como a classe é a estrutura comum a todos os objetos dela derivados, são compartilhados entre todos os objetos de uma mesma classe

Modelando a classe Cliente

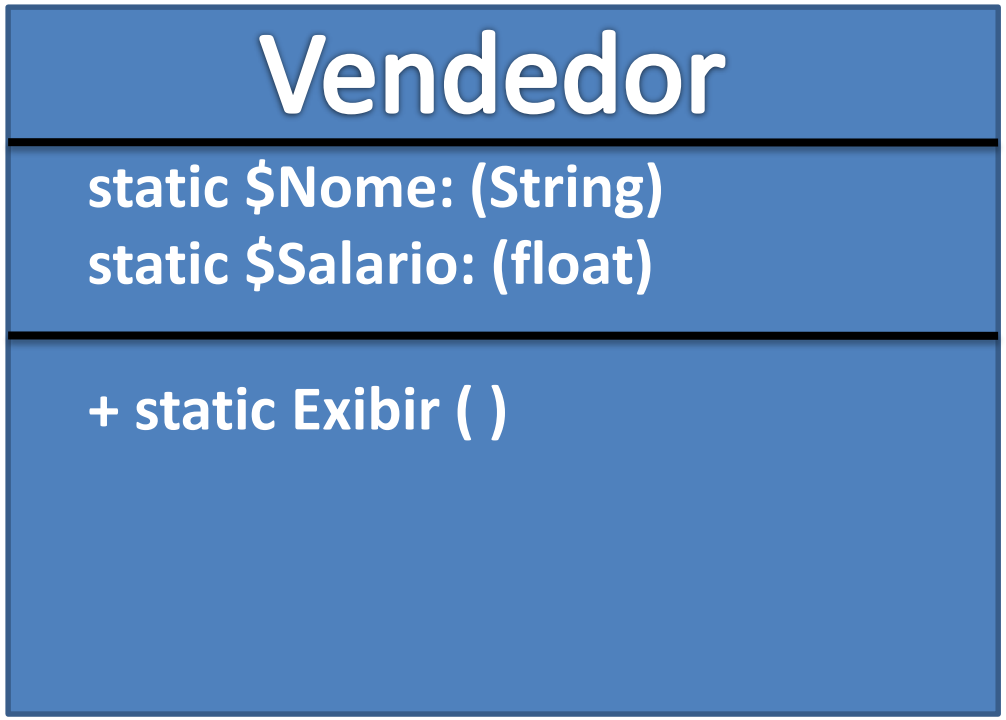


Método estático

Podem ser invocados diretamente da classe, sem a necessidade de instanciar um objeto para isso. São limitados a chamarem outros métodos estáticos da classe ou utilizar apenas propriedades estáticas. Para executar um método estático utilizamos a sintaxe:

NomeDaClasse::NomeDoMetodo

Modelando a classe Vendedor



→ Nome da Classe

→ Propriedades

→ Método

TECH
soluções em ti

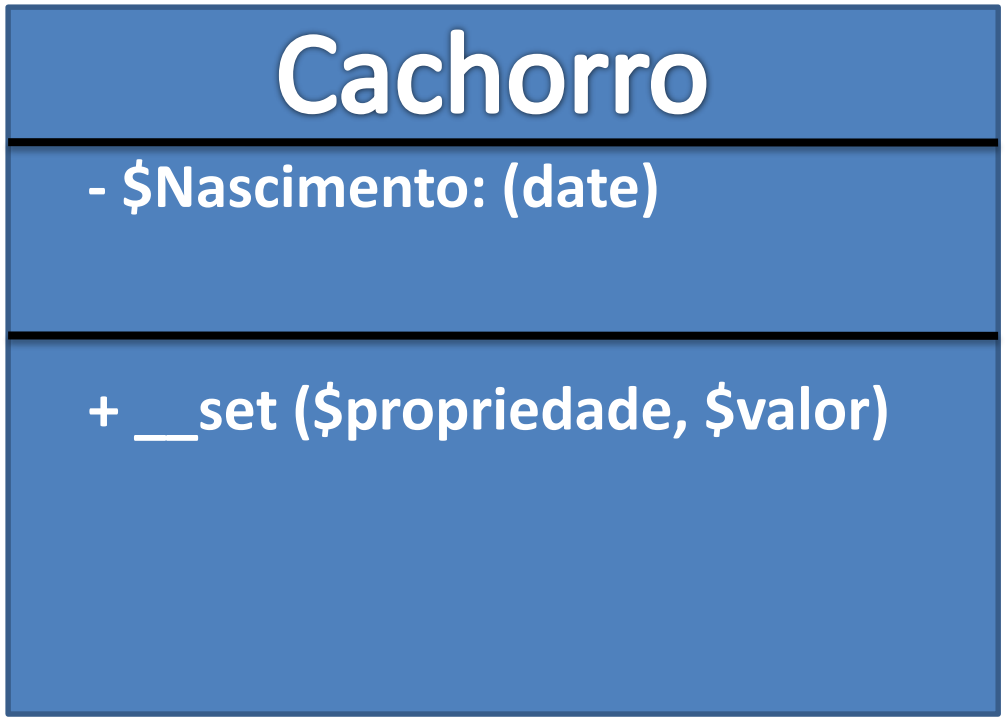
Intercepções

O PHP 5 introduziu o conceito de interceptação realizadas por objetos por meio dos métodos **set( )** e **get( )**, vistos a seguir:

Método set( )

Intercepta a atribuição de valores a propriedades de um objeto. Sempre que for atribuído um valor a uma propriedade do objeto, automaticamente esta atribuição passa pelo método set( ), o qual recebe o nome da propriedade e o valor a ser atribuído.

Ex.: **setNome**(**$nome**){}



O método set interceptará o valor atribuído, validando-o ou não

TECH
soluções em ti

Método get( )

Intercepta as requisições de propriedades do objeto. Sempre que for requisitada uma propriedade, automaticamente passará pelo método get( ), o qual recebe o nome da propriedade.

Ex.: **getNome**( ){ return; }

Método call( )

Intercepta a chamada a métodos. Sempre que for executado um método que não existir no objeto, automaticamente a execução será direcionada para ele, que recebe dois parâmetros, o nome do método requisitado e o parâmetro recebido, podendo decidir o procedimento a realizar

Método toString( )

Quando imprimimos objetos na tela, por meio de comandos como echo ou print, o PHP exibe no console o identificador interno do objeto. Para alterar esse comportamento, podemos definir o método __toString() para cada classe. Caso o método __toString() exista, no momento em que mandarmos exibir um objeto no console, o PHP irá imprimir o retorno desta função

Método __clone( )

O comportamento padrão do PHP quando atribuímos um objeto ao outro é criar uma referência entre os objetos. Dessa forma, teremos duas variáveis apontando para a mesma região da memória. Utilizamos o método __clone() quando queremos duplicar o objeto, o objeto resultante poderá ter os mesmos valores em suas propriedades, ou não.

Interfaces

A POO baseia-se fortemente na interação de classes e objetos. Um objeto deve conhecer quais são as funcionalidades que um outro objeto pode lhe fornecer. Na etapa de projeto do sistema, podemos definir conjuntos de métodos que determinadas classes do nosso sistema deverão implementar incondicionalmente. Tais conjuntos de métodos são as interfaces, as quais contêm a declaração de métodos de forma prototipada, sem qualquer implementação.

Toda classe que implementar uma interface deverá obrigatoriamente possuir os métodos predefinidos na interface, caso contrário, resultará em erro