Orientação a Objetos

- Abstração:

- Classes abstratas: São classes que estão na hierarquia de classes mas que servirão de base para outras. Elas nunca serão instanciadas; somente suas filhas.
- Conta3.class.php

Orientação a Objetos

- Abstração:

- Classes finais: uma classe final não pode ser uma superclasse, ou seja, não pode ser base em uma estrutura de herança.
- ContaPoupanca.class.php
- classe_final.php

Orientação a Objetos

- Métodos Abstratos:

Um método abstrato consiste na definição de uma assinatura na classe abstrata. Este método deverá conter uma implementação na classe-filha, mas não deve possuir implementação na classe em que ele é definido.

- Conta4.class.php
- abstrato.php

Orientação a Objetos

- Métodos Finais:

Um método final não pode ser subscrito, ou seja, não pode ser redefinido na classe-filha. Para marcar um método como final, basta utilizar o operador FINAL no início da sua declaração.

- ContaCorrente2.class.php
- metodo_final.php

Orientação a Objetos

- Encapsulamento: é um mecanismo que provê proteção de acesso aos membros internos de um objeto.

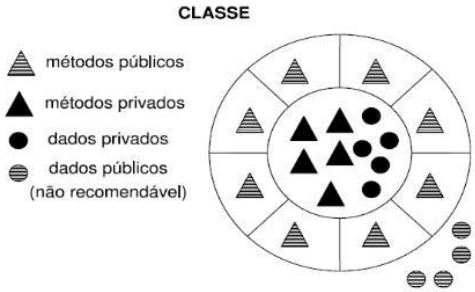


Figura 3.1: Encapsulamento: os métodos públicos podem ser empregados para proteger os dados privados.

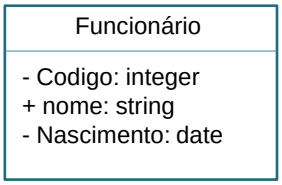
Orientação a Objetos

- Encapsulamento
  - Private: membros declarados como private somente podem ser acessados dentro da própria classe.
  - Protected: membros declarados como protected somente podem ser acessados dentro da própria classe em que foram declarados e a partir de classes descendentes, mas não poderão ser acessados a partir do programa que faz uso da classe.
  - Public: membros declarados como public poderão ser acessados livremente a partir da própria classe em que foram declarados, a partir de classes descendentes e a partir do próprio programa que faz uso dessa classe.

Orientação a Objetos

- Encapsulamento

Private

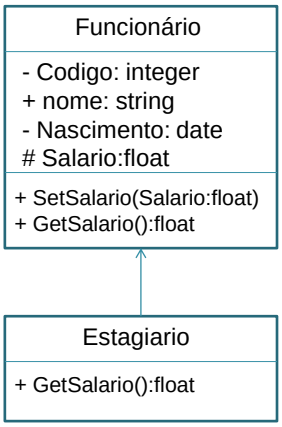


- Funcionario1.class.php
- private.php
- Funcionario2.class.php
- private2.php

Orientação a Objetos

- Encapsulamento

Protected



- Estagiario.class.php
- protected.php
- Funcionario3.class.php
- protected2.php

Orientação a Objetos

- Encapsulamento

Public

Public.php

Orientação a Objetos

- Membros de Classe:

Como visto anteriormente, a classe é uma estrutura-padrão para criação dos objetos. A classe permite que armazenemos valores nela de duas formas: constantes de classe e propriedades estáticas. Estes atributos são comuns a todos os objetos da mesma classe.

Orientação a Objetos

- Constantes:

No exemplo a seguir, veremos como se dá a declaração de uma constante pelo operador **const**, seu acesso de forma externa ao contextos da classe, pela sintaxe self::nomeDaConstante. O operador self representa a própria classe.

constantes.php

Orientação a Objetos

- Propriedades Estáticas:

Propriedades estáticas são atributos de uma classe; são dinâmicas como as propriedades de um objeto, mas estão reacionadas à classe. Como a classe é a estrutura comum a todos os objetos dela derivados, propriedades estáticas são compartilhadas entre todos os objetos de uma mesma classe.

propesta.php

Orientação a Objetos

- Métodos Estáticos:

Métodos estáticos podem ser invocados diretamente da classe, sem a necessidade de instanciar um objeto para isso. Eles não devem referenciar propriedades internas pelo operador $this, porque este operador é utilizado para referenciar instâncias da classe (objetos), mas não a própria classe; são limitados a chamarem outros métodos estáticos da classe ou utilizar apenas propriedades estáticas. Para executar um método estático basta utilizar a sintaxe NomeDaClasse::NomeDoMetodo().

metesta.php

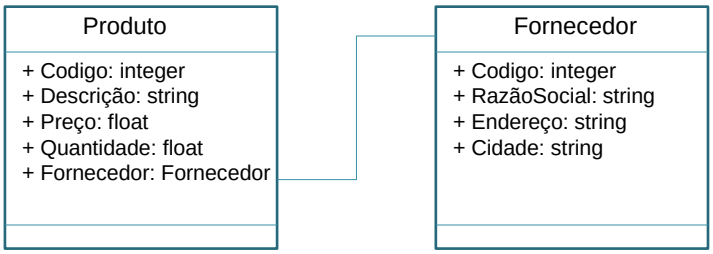
Orientação a Objetos

- **Associação, Agregação e Composição**

Associação: é uma relação comum entre dois objetos, de modo que um possui uma referência à posição de memória onde o outro se encontra, podendo visualizar seus atributos ou mesmo acionar uma de suas funcionalidades (métodos).

Fornecedor1.class.php

Associacao.php



Orientação a Objetos

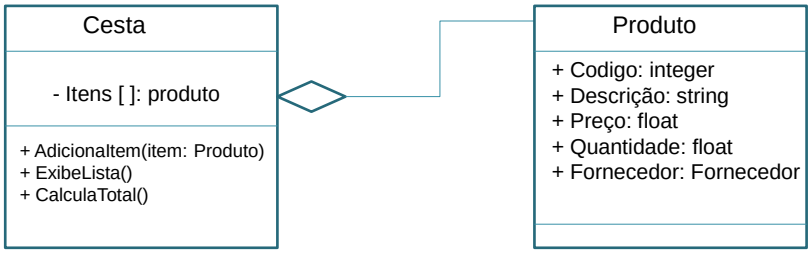
- **Associação, Agregação e Composição**

Agregação: é o tipo de relação entre objetos conhecida como todo/parte. Na agregação, um objeto agrega outro objeto, ou seja, se torna um objeto externo parte de si mesmo pela utilização de um dos seus métodos. Assim, o objeto-pai poderá utilizar funcionalidades do objeto agregado.

Cesta.class.php

Agregacao.php

agregacao2.php



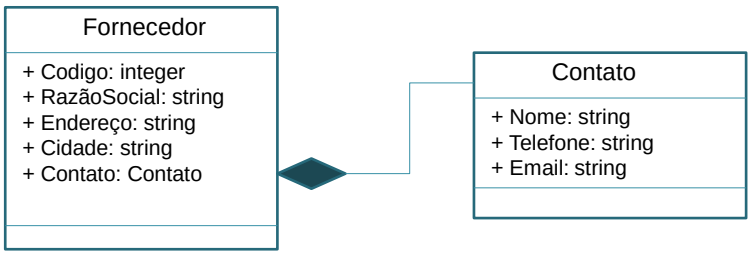
Orientação a Objetos

- **Associação, Agregação e Composição**

Composição: também é uma relação que demonstra uma relação todo/parte. A diferença em relação á agregação é que, na composição, o objeto-pai ou "todo" é responsável pela criação e destruição de suas partes.

Contato.class.php

Composição.php



Orientação a Objetos

- Intercepções

O PHP5 introduziu o conceito de intercepção em operações realizadas por objetos por meio dos métodos __set(), __get(), __call().

Orientação a Objetos

- Intercepções

Método __set():

Intercepta a atribuição de valores a propriedades do 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, podendo atribuí-lo ou não.

Cachorro.class.php

Set.php

Orientação a Objetos

- Intercepções

Método __get():

Intercepta requisições de propriedades do objeto. Sempre que for requisitada uma propriedade, automaticamente essa requisição passará pelo método __get(), o qual devolve o nome da propriedade requisitada, podendo retorná-la ou não.

Produto3.class.php

Get.php

Orientação a Objetos

- Intercepções

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.

Produto4.class.php

Call.php

Orientação a Objetos

- Intercepções

Método __toString():

Até a versão 5.1, quando eram impressos objetos na tela, por meio de comandos como o echo ou print, o PHP exibia no console o identificador interno do objeto.

Atualmente, caso venhamos a utilizar o comando echo ou print sobre um objeto, o PHP exibe um erro. Para alterar esse comportamento, podemos definir o método __toString() para cada classe.

toString.php

Orientação a Objetos

- Intercepções

Método __toXml():

O PHP não possui método para retornar o objeto em formato XML, mas isso não é problema, uma vez que é muito fácil desenvolver tal funcionalidade.

toXml.php

XMLBase.php

Orientação a Objetos

- Método __clone()

O comportamente-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 de memória. Este é o comportamento desejado na maioria das vezes, mas como proceder quando quisermos de fato duplicar um objeto na memória? Simples! Utilizaremos o operador clone.

Clone.php

Orientação a Objetos

- Método __autoload():

Sempre que se instancia um objeto em PHP, é necessário ter a declaração da classe na memória; caso contrário, a aplicação é finalizada com um erro. Para introduzir uma classe na memória podemos utilizar comandos como include_once. Podemos realiar tal operação no início da aplicação introduzindo todas as classes que poderão ser necessárias durante a execução da aplicação.

Autoload.php

Orientação a Objetos

- Objetos dinâmicos

O PHP 5 nos oferece diversas facilidades para manipulação de objetos. Uma delas é a possibilidade de criar objetos dinamicamente, sem ter a classe previamente definida. Naturalmente, esses objetos irão armazenar somente dados, e não funções.

Dinamico.php
Objarray.php

Orientação a Objetos

- Tratamento de Erros
  - A função die():

A forma de manipulação de erro mais simples é abortar a execução da aplicação.

Erro_die.php

  - Retorno de flags:

A segunda forma de manipulação de erros é o retorno de flags TRUE ou FALSE. Retornamos TRUE em caso de sucesso na operação e FALSE em caso de erros.

Erro_flag.php

Orientação a Objetos

- Tratamento de Erros
  - Lançando erros:

Uma forma mais elegante de realizar a manipulação de erros é por meio das funções trigger_error(), que lança um erro, e a função set_error_handler(), a qual define uma função que realizará a manipulação dos erros lançados.

Orientação a Objetos

- Tratamento de Erros
  - Lançando erros:

bool trigger_error (string erro, int tipo)

Parâmetro	Descrição
erro	Mensagem de erro gerada
tipo	Tipo de erro.

Tipo	Significado
E_USER_ERROR	Gera erro fatal
E_USER_WARNING	Gera uma warning
E_USER_NOTICE	Gera um notice

Erro_trigger.php

Orientação a Objetos

- Tratamento de exceções

O PHP 5 implementa o conceito de tratamento de exceções, da mesma forma que ele é implementado em linguagens como C++ ou Java. Uma exceção é um objeto especial derivado da classe Exception, que contém alguns métodos para informar ao programador um relato do que aconteceu.

Método	Descrição
getMessage()	Retorna a mensagem de erro.
getCod()	Retorna o código de erro.
getFile()	Retorna o arquivo no qual ocorreu o erro.
getLine()	Retorna a linha na qual ocorreu o erro.
getTrace()	Retorna um array com as ações até o erro.
getTraceAsString()	Retorna as ações em forma de string.

Erro_exception.php

Erro_subexception.php