



DISCIPLINA

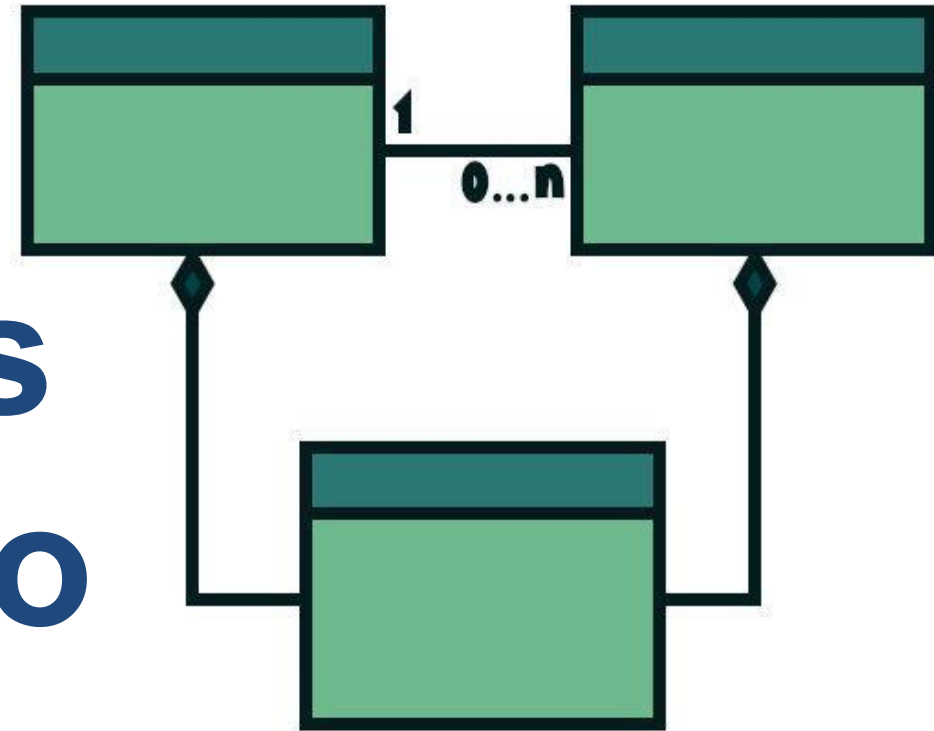
ANÁLISE DE SISTEMAS

ORIENTADO A OBJETOS

AULA 9

Profº. Me. Flávio Henrique Fernandes Volpon
flavio.volpon@docente.unip.br

Modelo de Classes de Domínio



Modelo de Classes de Domínio

- **Desenvolvido** na **fase de análise**, o modelo de classes de domínio **representa** os **objetos**, ou **classes**, inerentes ao **domínio** do **problema** que queremos **resolver**.
- **Deixamos de lado**, nessa visão, **detalhes tecnológicos** da solução do problema.
- Como vimos, um **objeto** é **composto** por **atributos** e **métodos**, e uma **classe** pode ser **considerada** como a **especificação** de um **objeto**.

Um objeto pode ser considerado como a
instância de uma classe

Modelo de Classes de Domínio

- **Normalmente identificamos** uma **classe** pelo que ela **representa** dentro do **domínio** do **negócio** em que estamos trabalhando, por exemplo:
- A **classe Cliente Pessoa Física** representa todos os **clientes** do **tipo pessoa física** que podem **efetuar saques** em nosso terminal de **autoatendimento**.
- No **entanto**, quando estamos **representando** em um **modelo**, seguimos determinadas **regras** de **nomenclatura** e, no caso da **representação** de **classes**, devemos nos atentar a **algumas boas práticas**, como:

Modelo de Classes de Domínio

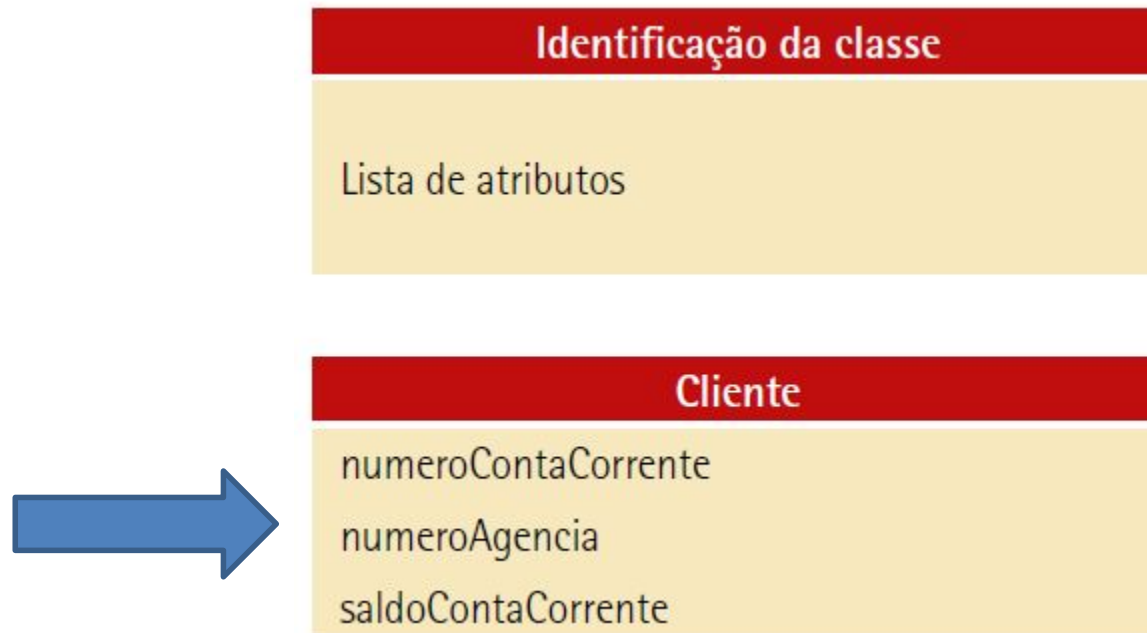
Na **UML**, representamos uma **classe**, uma **caixa** sempre com a **identificação** do **nome da classe** na parte **superior**.



Representação de uma Classe na UML

Modelo de Classes de Domínio

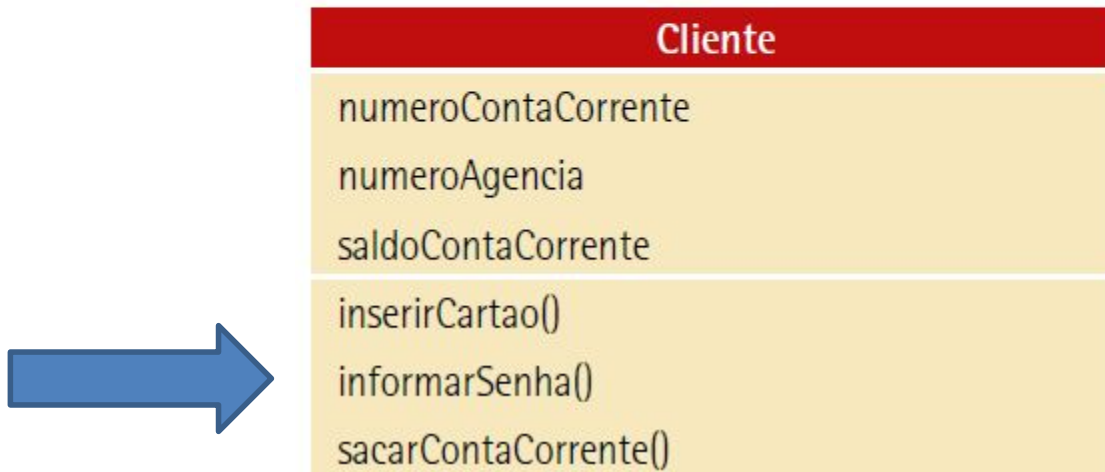
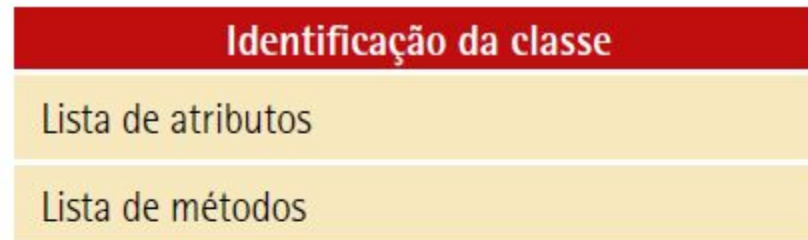
Na **UML** representamos os **atributos** de uma **classe** **abaixo** da **identificação** da classe, de forma **sequencial** e um **abaixo** do **outro**.



Representação de Atributos em uma Classe UML

Modelo de Classes de Domínio

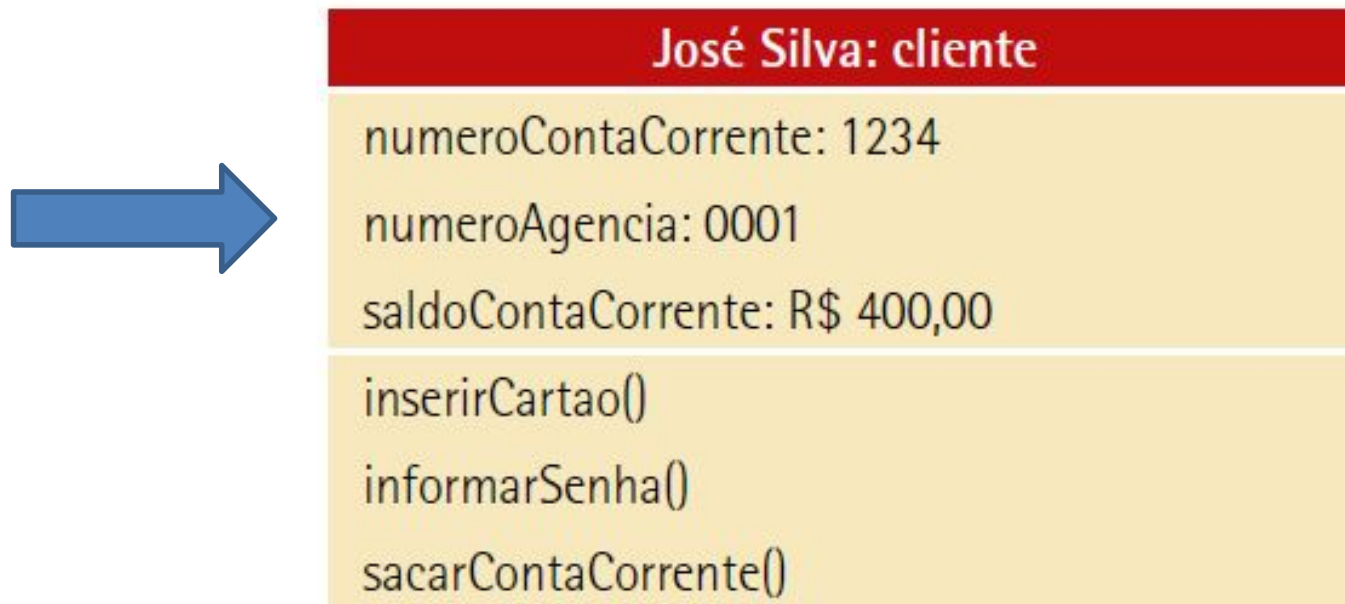
Os **métodos** de uma **classe** são **representados**, na **UML**, **abaixo** da identificação de **atributos**.



Representação de Métodos em uma Classe UML

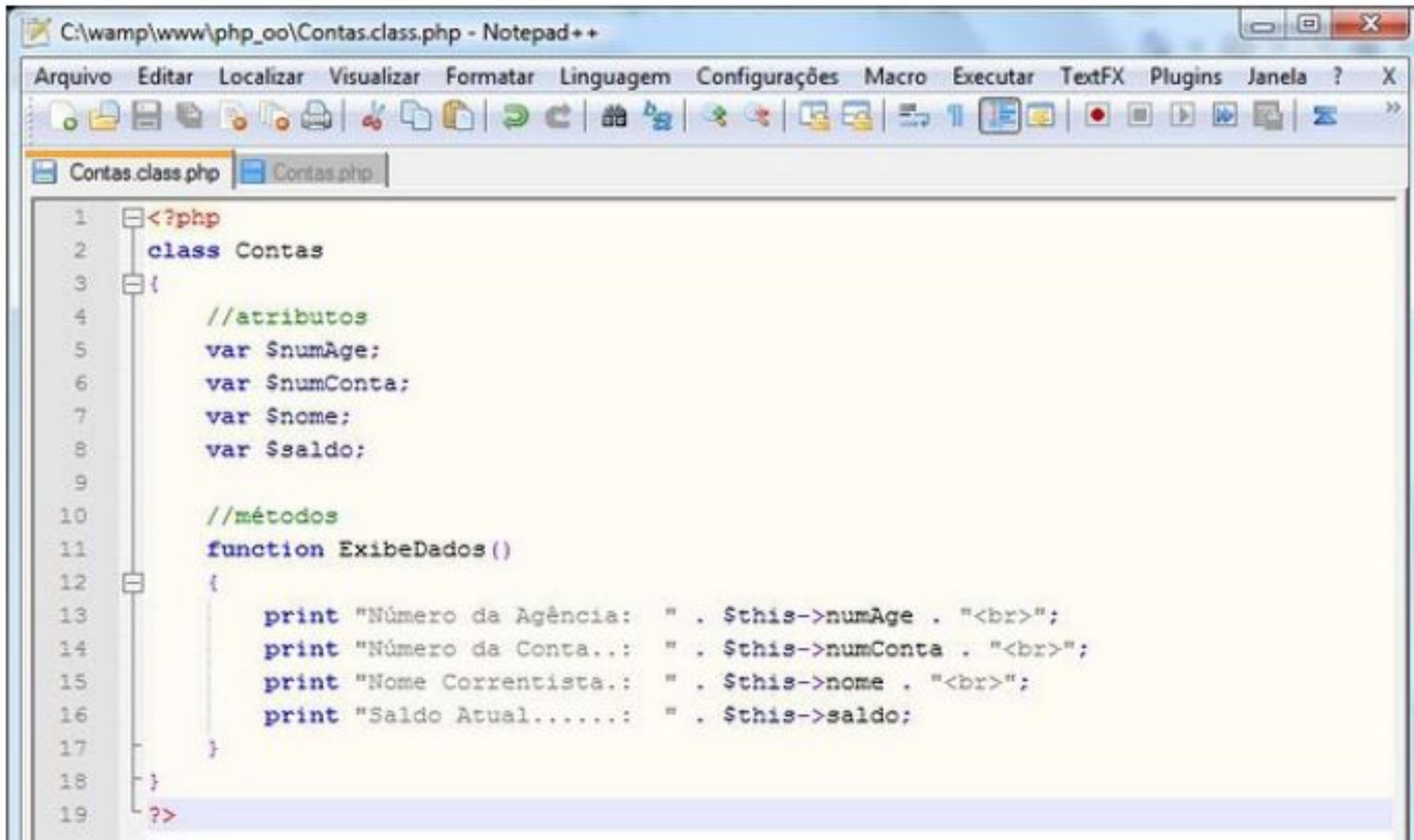
Modelo de Classes de Domínio

- Um **objeto** é uma **instância** de uma **classe**, note que na figura representamos o **nome** do **objeto**, seus **atributos** e os respectivos **valores**, assim como seus **métodos**.



Representação de um objeto UML

Modelo de Classes de Domínio



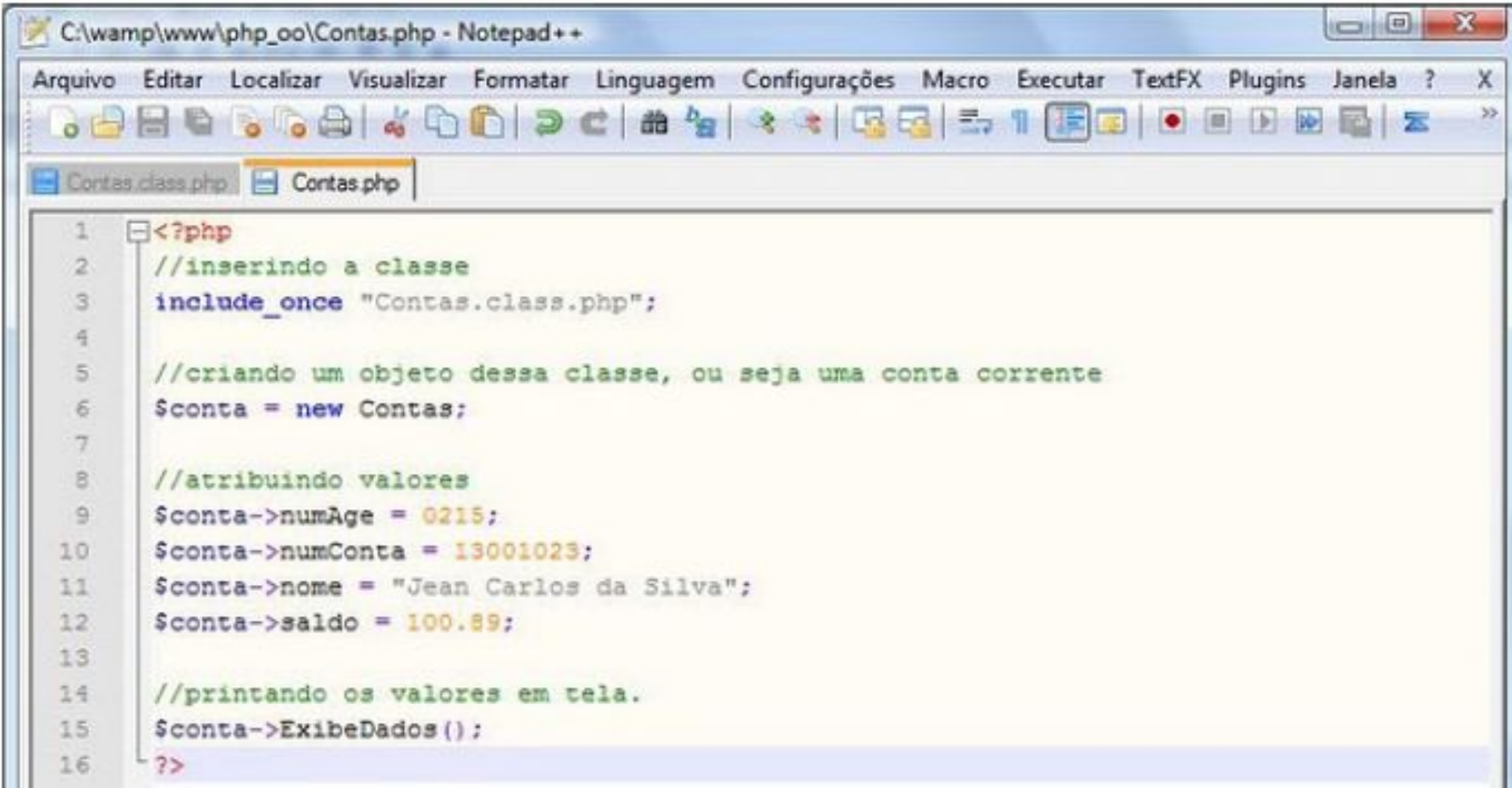
```
C:\wamp\www\php_oo\Contas.class.php - Notepad++
Arquivo  Editar  Localizar  Visualizar  Formatar  Linguagem  Configurações  Macro  Executar  TextFX  Plugins  Janela  ?  X

Contas.class.php  Contas.php

1  <?php
2  class Contas
3  {
4      //atributos
5      var $numAge;
6      var $numConta;
7      var $nome;
8      var $saldo;
9
10     //métodos
11     function ExibeDados()
12     {
13         print "Número da Agência: " . $this->numAge . "<br>";
14         print "Número da Conta..: " . $this->numConta . "<br>";
15         print "Nome Correntista.: " . $this->nome . "<br>";
16         print "Saldo Atual.....: " . $this->saldo;
17     }
18 }
19 ?>
```

Exemplo do Código fonte de uma Classe em PHP

Modelo de Classes de Domínio

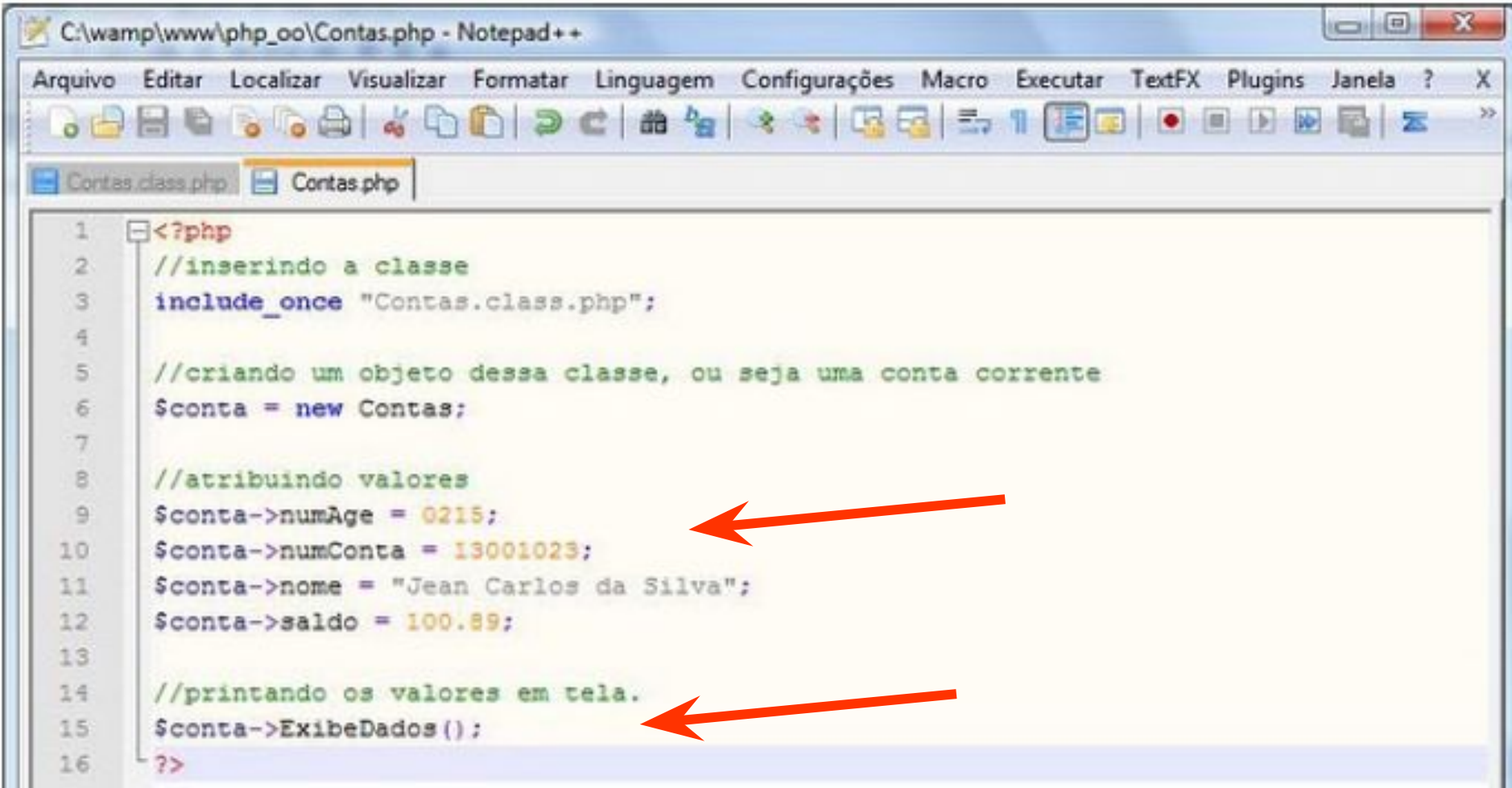


The image shows a Notepad++ window titled "C:\wamp\www\php_oo\Contas.php - Notepad++". The menu bar includes "Arquivo", "Editar", "Localizar", "Visualizar", "Formatar", "Linguagem", "Configurações", "Macro", "Executar", "TextFX", "Plugins", "Janela", "?", and "X". The toolbar contains various icons for file operations and editing. The active tab is "Contas.php". The code is as follows:

```
1 <?php
2 //inserindo a classe
3 include_once "Contas.class.php";
4
5 //criando um objeto dessa classe, ou seja uma conta corrente
6 $conta = new Contas;
7
8 //atribuindo valores
9 $conta->numAge = 0215;
10 $conta->numConta = 13001023;
11 $conta->nome = "Jean Carlos da Silva";
12 $conta->saldo = 100.89;
13
14 //printando os valores em tela.
15 $conta->ExibeDados();
16 ?>
```

Exemplo do Código fonte de uma Instância de Classe em PHP

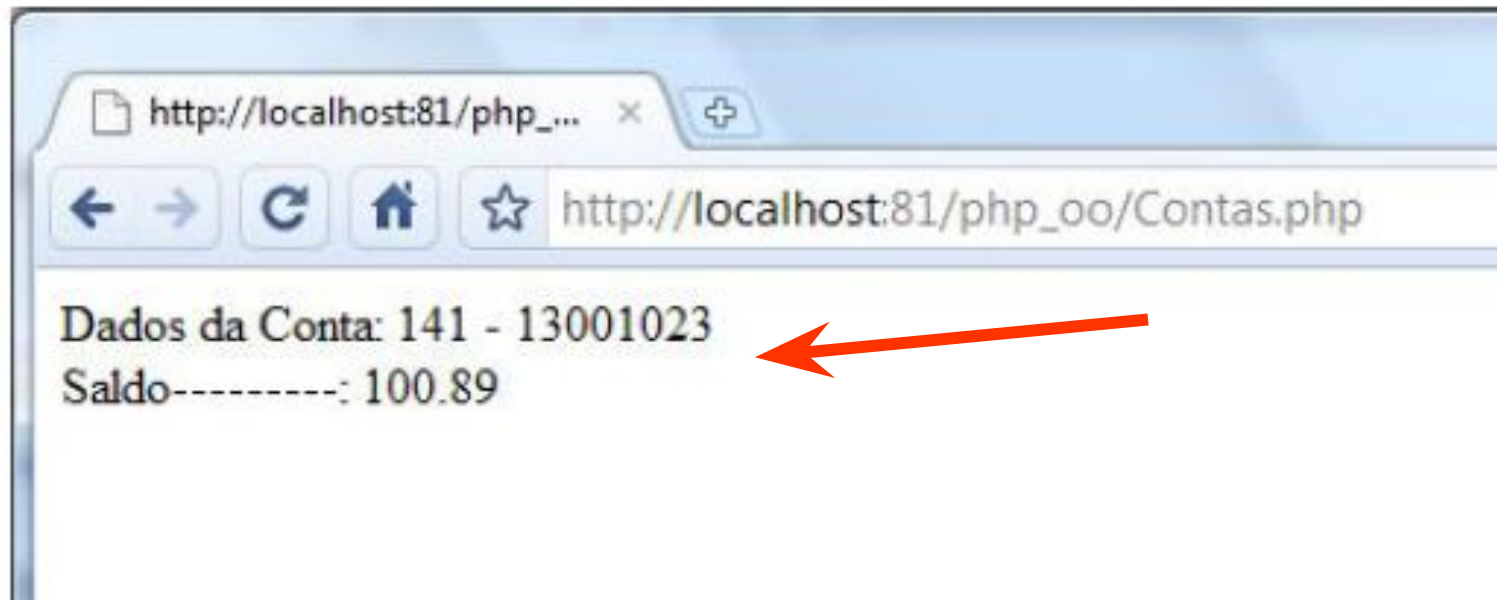
Modelo de Classes de Domínio



```
1 <?php
2 //inserindo a classe
3 include_once "Contas.class.php";
4
5 //criando um objeto dessa classe, ou seja uma conta corrente
6 $conta = new Contas;
7
8 //atribuindo valores
9 $conta->numAge = 0215;
10 $conta->numConta = 13001023;
11 $conta->nome = "Jean Carlos da Silva";
12 $conta->saldo = 100.89;
13
14 //printando os valores em tela.
15 $conta->ExibeDados();
16 ?>
```

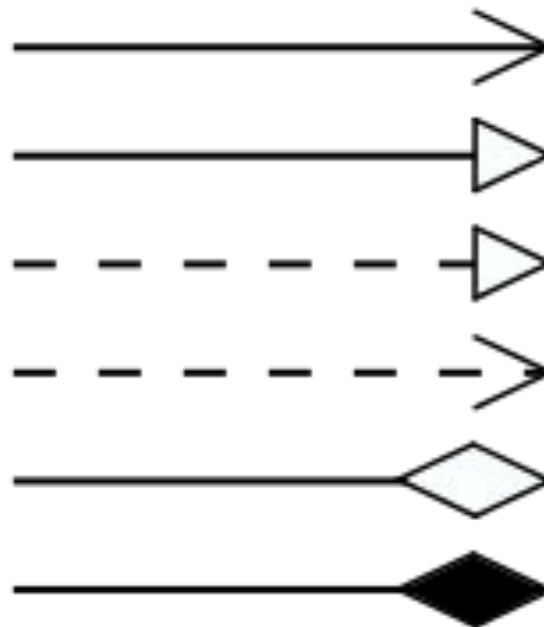
Exemplo do Código fonte de uma Instância de Classe em PHP

Modelo de Classes de Domínio



Exemplo de uma Instância da Classe Contas

Relacionamento entre Classes



Relacionamento entre Classes

- Assim **como** no **mundo real**, **objetos** de **sistema** se **relacionam** entre si dentro de um determinado **contexto** para **resolução** de um determinado **problema**.
- Existem cinco tipos de relacionamento entre objetos. Quatro deles, estudaremos na sequencia, são eles:
 1. dependência
 2. associação
 3. agregação
 4. composição
- O quinto e ultimo será debatido separadamente.
 5. herança

Relacionamento entre Classes

DEPENDÊNCIA

- **Dependência** é um **relacionamento** em que um objeto **depende** de **informações** de **outro objeto** para a **execução** de um determinado comportamento.
- Na **UML**, **representamos** a dependência utilizando uma **seta tracejada**, como mostra a figura a seguir.
- Ainda na imagem, podemos fazer a **leitura** de que a **Classe** **A depende da Classe B.**

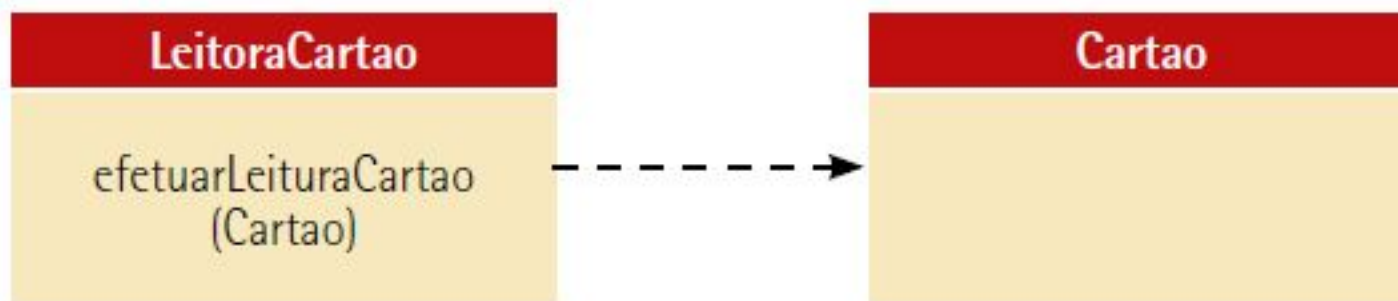
Relacionamento entre Classes

DEPENDÊNCIA



Representação de Dependência de Classes

Comumente, a relação de dependência se dá quando uma **classe utiliza informações de outra classe** em um **determinado método**, como mostra a figura a seguir, na qual podemos notar que a classe Leitora de Cartao utiliza informações do objeto Cartao para realizar o comportamento “efetuarLeituraCartao”.



Exemplo de Dependência de Classes

Relacionamento entre Classes

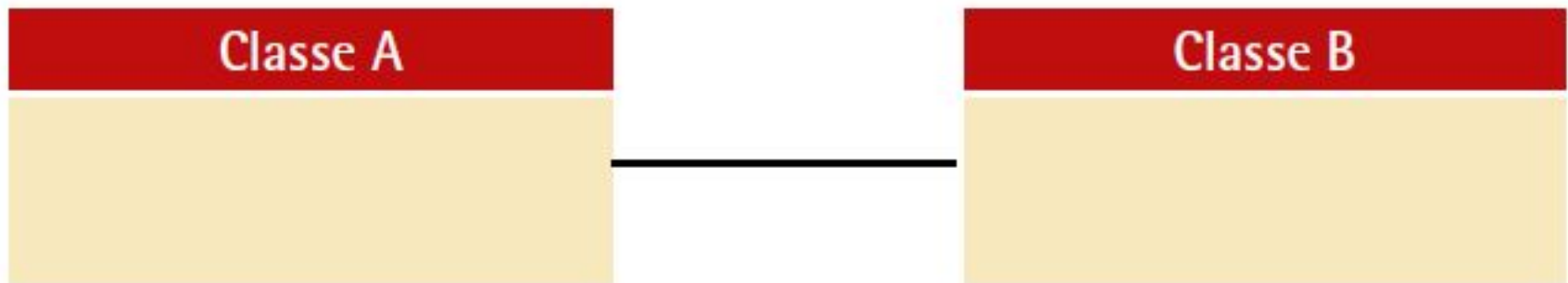
ASSOCIAÇÃO

- A **associação** mostra que um **objeto** pode se **relacionar** com **outro**, que existe uma **conexão entre** esses **objetos**.
- Existem **dois tipos** de **associação**:
 1. **Associação Binária**: amplamente utilizada, é a associação entre duas classes apenas.
 2. **Associação Enésima**: raramente utilizada, é a associação entre mais de duas classes

Relacionamento entre Classes

ASSOCIAÇÃO

- Na **UML**, representamos a associação como **uma reta**, ou uma **linha**, que **conecta** as **classes** que se associam de alguma forma.



Representação de Associação de Classes

Note que, embora representado corretamente, não conseguimos compreender, ou fazer a leitura, sobre o que se trata essa associação. Por isso, é comum que utilizemos algum tipo de notação que identifique a relação entre as classes em um determinado domínio, como mostra o quadro a seguir.

Relacionamento entre Classes

ASSOCIAÇÃO

- Essa **notação**, além de **identificar a relação** entre as **classes**, **estabelece o papel** que cada **objeto** ou **classe** estabelece nessa **relação**.

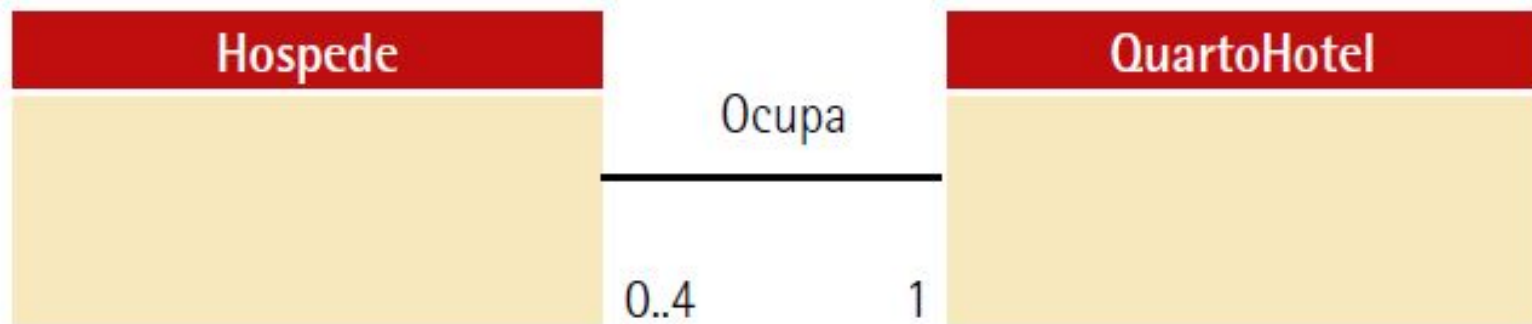
Domínio	Representação UML			Interpretação
Vendas	Cliente	Compra	Produto	Um cliente compra produtos.
Bancário	ContaCorrente	Possui	Transacao	Uma conta-corrente possui transações efetuadas.
Hotelaria	Hospede	Ocupa	QuartoHotel	Um hóspede ocupa um quarto de hotel.

Exemplos de Associação UML

Relacionamento entre Classes

ASSOCIAÇÃO

- Voltando ao **exemplo** da **hotelaria**, suponhamos que, em um problema imaginário de reserva de quartos pela web, **cada quarto** de hotel possa **ter**, no **máximo**, **quatro hóspedes**, sendo que eles podem, **eventualmente**, **estar vazios**.



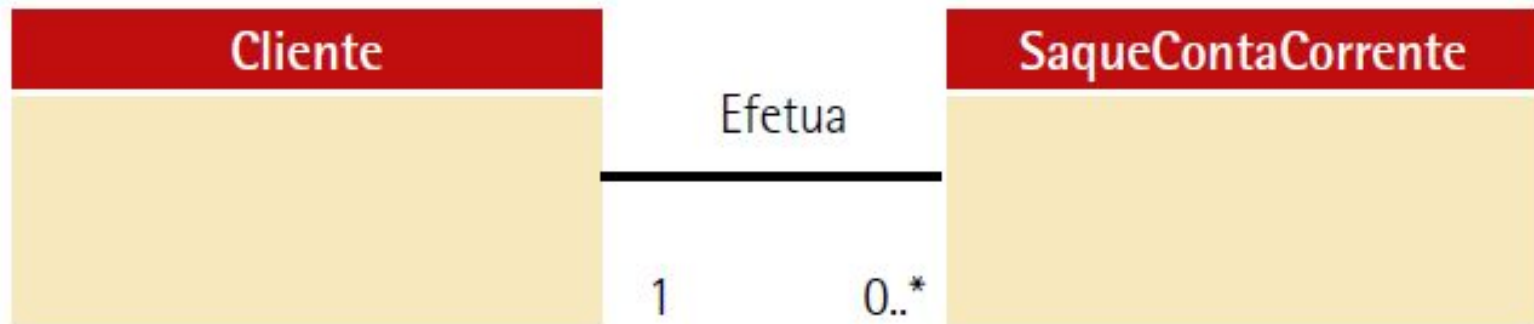
Exemplo de Multiplicidade na UML

Um hospede ocupa no máximo um quarto de hotel.
Um quarto de hotel pode ser ocupado por no mínimo zero ou no máximo quatro hóspedes.

Relacionamento entre Classes

ASSOCIAÇÃO

- Utilizando o caso do terminal de **autoatendimento**, no qual o **cliente** pode **efetuar infinitos saques**, poríamos representar o modelo conforme da seguinte forma:



Exemplo de Multiplicidade em Saque de Conta-corrente

Um cliente pode efetuar no mínimo zero ou no máximo infinitos saques. Uma operação de saque de conta-corrente é feita por apenas um cliente.

Relacionamento entre Classes

ASSOCIAÇÃO

Quando estamos tratando de **multiplicidade**, temos as **possíveis opções de simbologia**:

Representação	Interpretação
0..*	No mínimo zero e no máximo infinito OU Zero ou Muitos
1..*	No mínimo um e no máximo infinito OU Um ou Muitos
0..1	No mínimo zero e no máximo 1 OU Zero ou Um
1	Apenas Um
X..Y	Intervalo específico de no mínimo X e no máximo Y

Opções para Representações de Multiplicidade

O conceito de multiplicidade é muito semelhante ao conceito de cardinalidade, que pode ser observado no MER (Modelo Entidade Relacionamento) quando do desenvolvimento do modelo conceitual de um banco de dados.

Relacionamento entre Classes

ASSOCIAÇÃO

- Existe ainda outra **variação** de associação, chamada **associação reflexiva**,
- Quando **objetos** da **mesma classe** são **associados**, sendo que nessa associação esses objetos possuem **papéis distintos**, como mostra a figura a seguir.

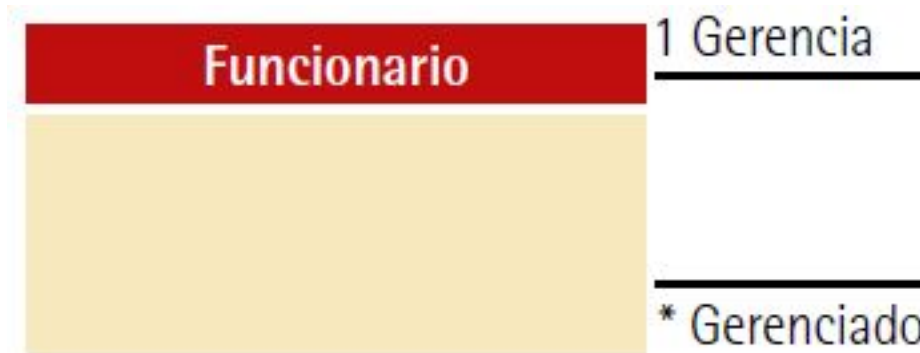


Representação de Associação Reflexiva

Relacionamento entre Classes

ASSOCIAÇÃO

- A figura a seguir mostra um **exemplo** de **aplicação** da **associação reflexiva**, incluindo o conceito de **multiplicidade**.
- Note que a **identificação** da **associação** é **fundamental** para que **não** haja **ambiguidade** na **interpretação** dos papéis dos objetos.



Exemplo de Associação Reflexiva

Relacionamento entre Classes

ASSOCIAÇÃO



Exemplo de Associação Reflexiva

Uma instancia da classe funcionário, como o objeto gerente, gerencia muitos funcionários ou instancias da mesma classe.

Uma instancia da classe funcionário, como o objeto analista, e gerenciada por no máximo um gerente ou instancia da mesma classe funcionário.

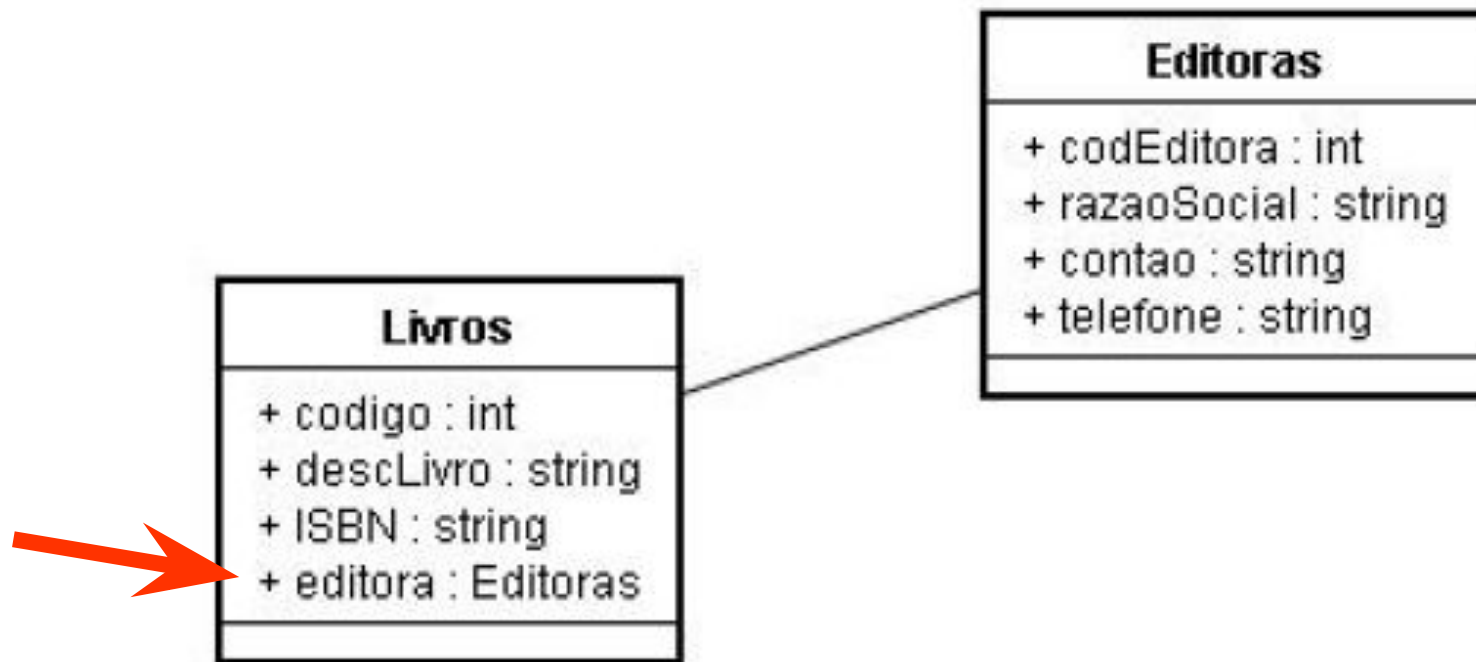
Nesse caso, ambos, gerente e analista, são instancias da classe funcionário, podemos dizer que ambos sao tipos de funcionario..

Relacionamento entre Classes

ASSOCIAÇÃO

Exemplo: A forma mais **comum** de implementar **associação** é ter um **objeto** como **atributo** de **outro**, neste exemplo, abaixo temos uma associação entre a Classe Livros e a classe Editoras. No código cria-se um **objeto** do tipo **Livro** e outro do **tipo Editora**. Um dos **atributos** do **Livro** é a **Editora**.

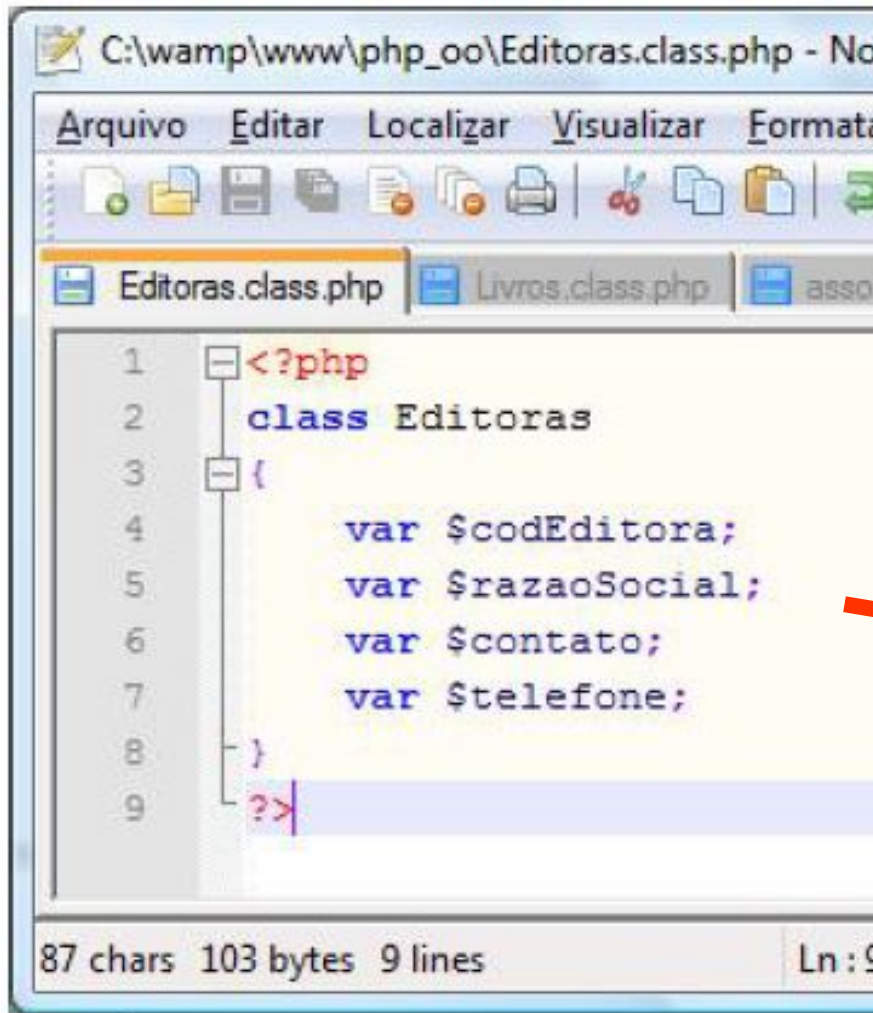
Veja a figura:



Associação entre as Classes Editoras e Livros

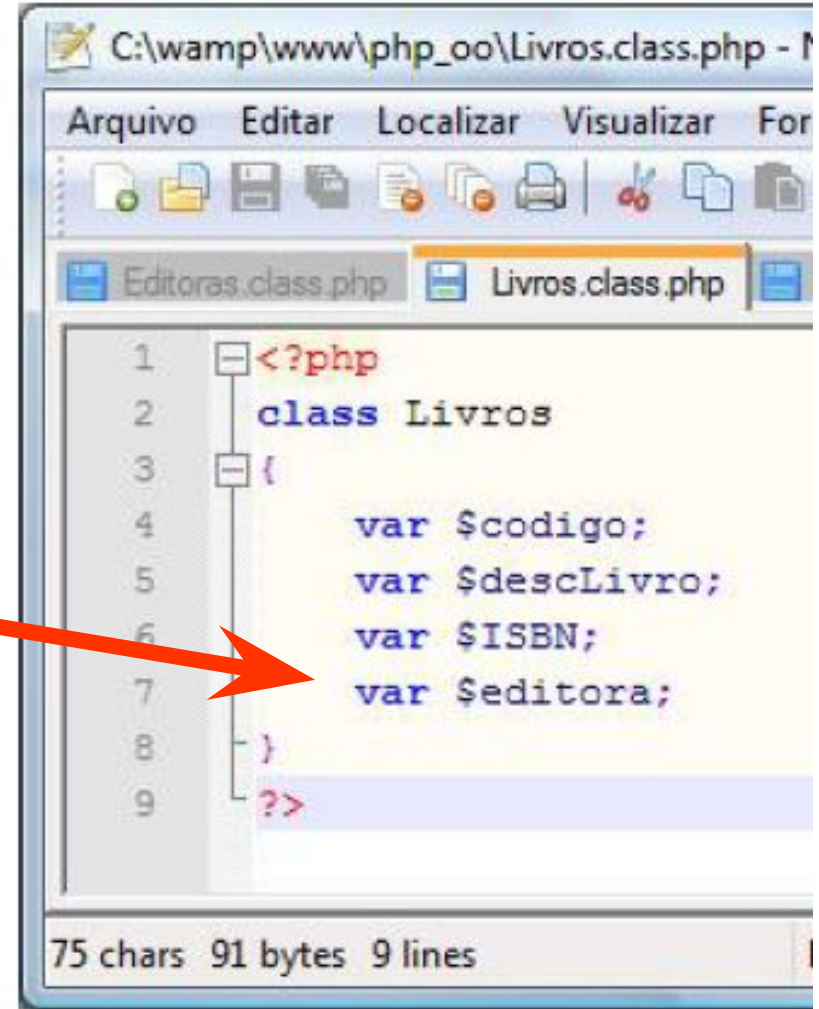
Relacionamento entre Classes

ASSOCIAÇÃO



```
1 <?php
2 class Editoras
3 {
4     var $codEditora;
5     var $razaoSocial;
6     var $contato;
7     var $telefone;
8 }
9 ?>
```

87 chars 103 bytes 9 lines



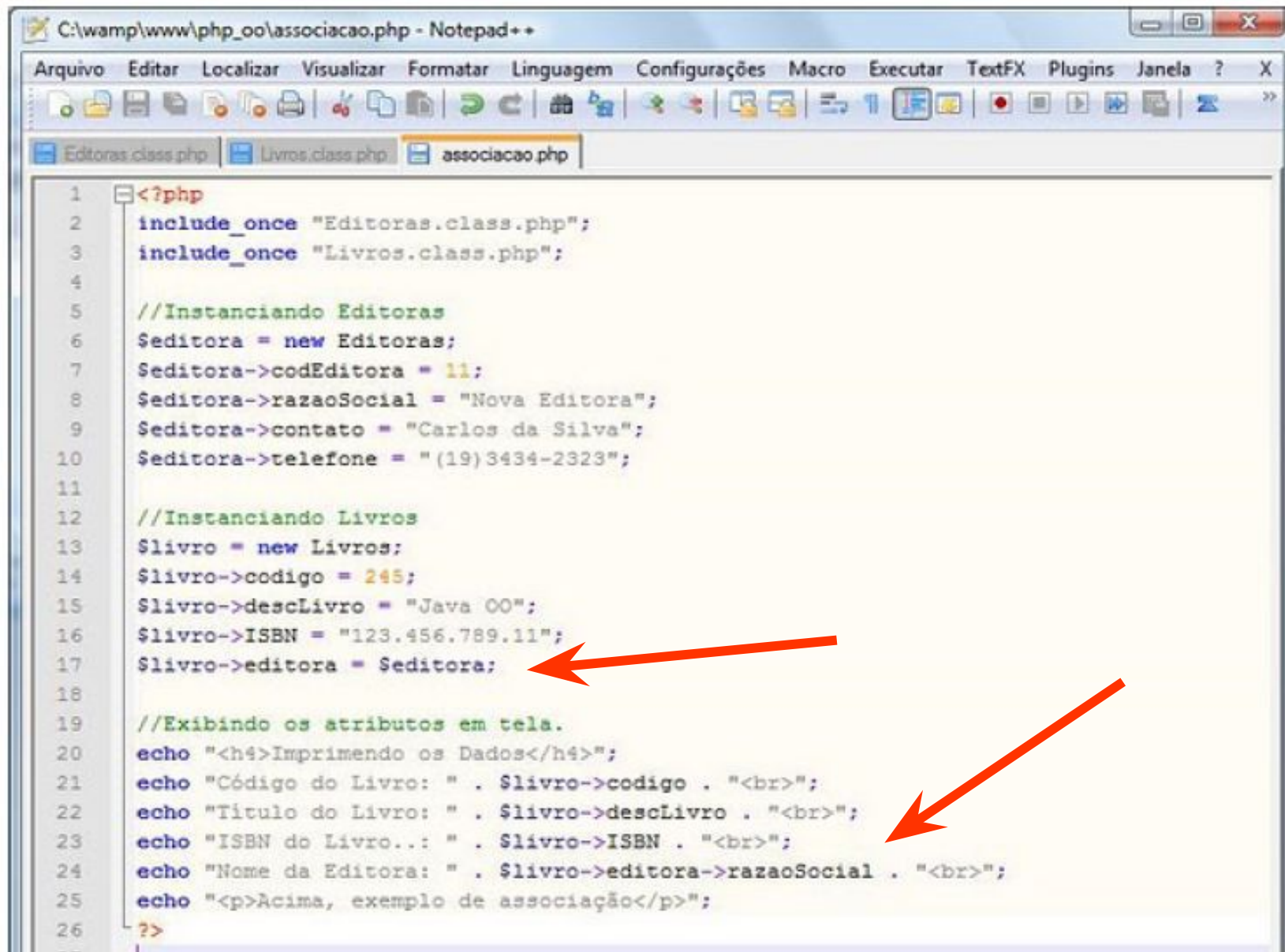
```
1 <?php
2 class Livros
3 {
4     var $codigo;
5     var $descLivro;
6     var $ISBN;
7     var $editora;
8 }
9 ?>
```

75 chars 91 bytes 9 lines

Declaração das Classes Editoras e Livros

Relacionamento entre Classes

ASSOCIAÇÃO



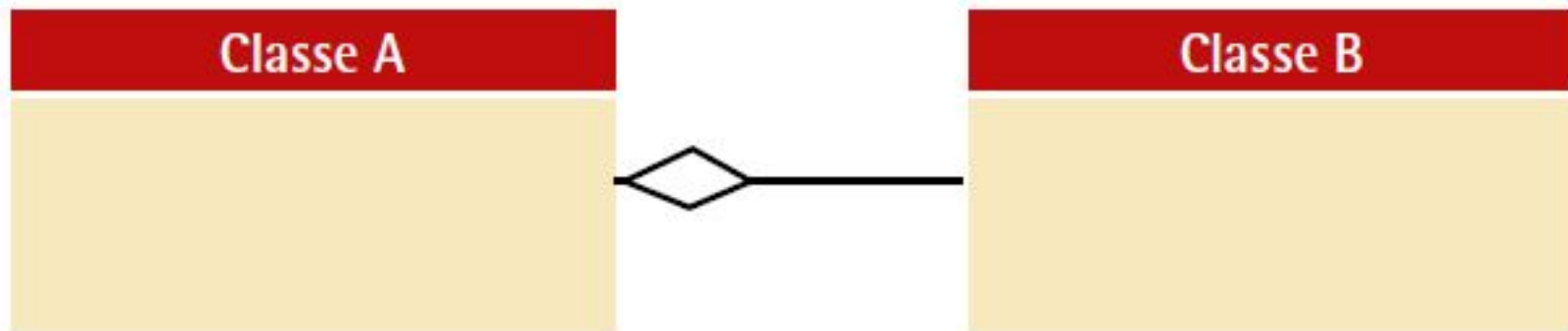
```
1 <?php
2 include_once "Editoras.class.php";
3 include_once "Livros.class.php";
4
5 //Instanciando Editoras
6 $editora = new Editoras;
7 $editora->codEditora = 11;
8 $editora->razaoSocial = "Nova Editora";
9 $editora->contato = "Carlos da Silva";
10 $editora->telefone = "(19) 3434-2323";
11
12 //Instanciando Livros
13 $livro = new Livros;
14 $livro->codigo = 245;
15 $livro->descLivro = "Java OO";
16 $livro->ISBN = "123.456.789.11";
17 $livro->editora = $editora;
18
19 //Exibindo os atributos em tela.
20 echo "<h4>Imprimindo os Dados</h4>";
21 echo "Código do Livro: " . $livro->codigo . "<br>";
22 echo "Título do Livro: " . $livro->descLivro . "<br>";
23 echo "ISBN do Livro...: " . $livro->ISBN . "<br>";
24 echo "Nome da Editora: " . $livro->editora->razaoSocial . "<br>";
25 echo "<p>Acima, exemplo de associação</p>";
26 ?>
```

Instância das Classes Editoras e Livros

Relacionamento entre Classes

AGREGAÇÃO

- A **agregação** é utilizada para **representar** uma **conexão todo-parte** entre **objetos**, ou seja, um **objeto está contido** no **outro**.
- Na UML, representamos agregação utilizando uma **reta saindo da classe** que representa a **parte** e se **conectando** a um **losango** na **classe** que representa o **todo**.

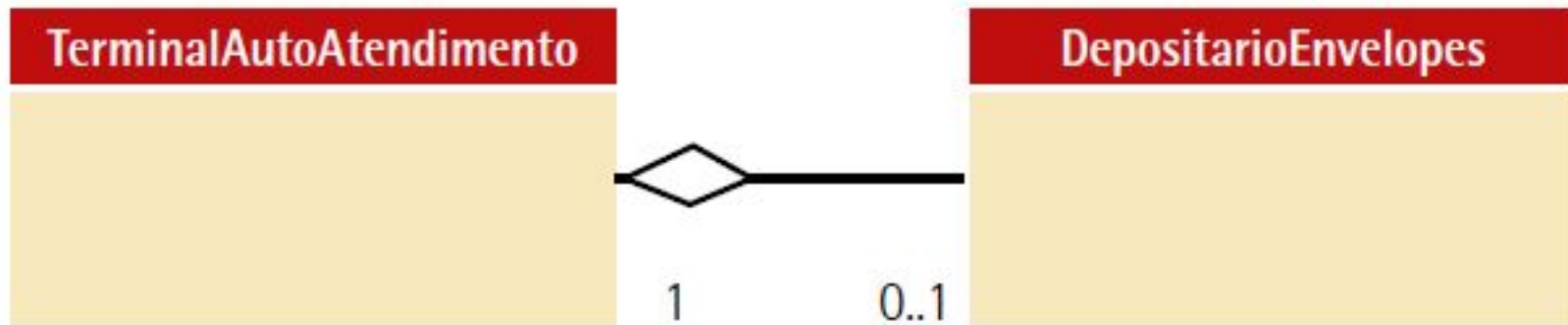


Exemplo de Associação Reflexiva

Relacionamento entre Classes

AGREGAÇÃO

- Assim **como** na **associação**, na **agregação** temos o conceito de **multiplicidade**, que define a **quantidade** de **objetos agregados** na **relação todo-parte**.



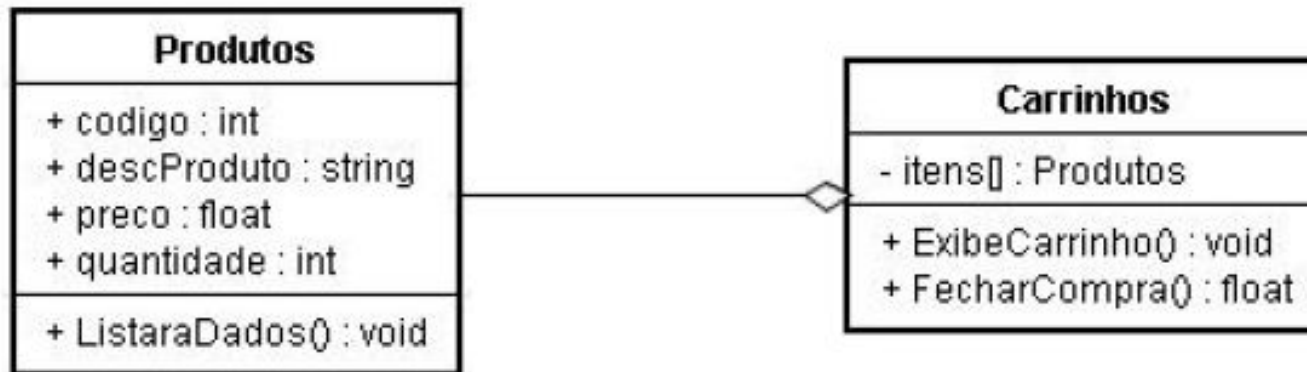
Exemplo de Agregação com Multiplicidade na UML

Relacionamento entre Classes

AGREGAÇÃO

Exemplo: Em um ambiente Web, onde teríamos o carrinho de compras (classe Carrinhos) com vários itens do tipo produtos (classe Produtos).

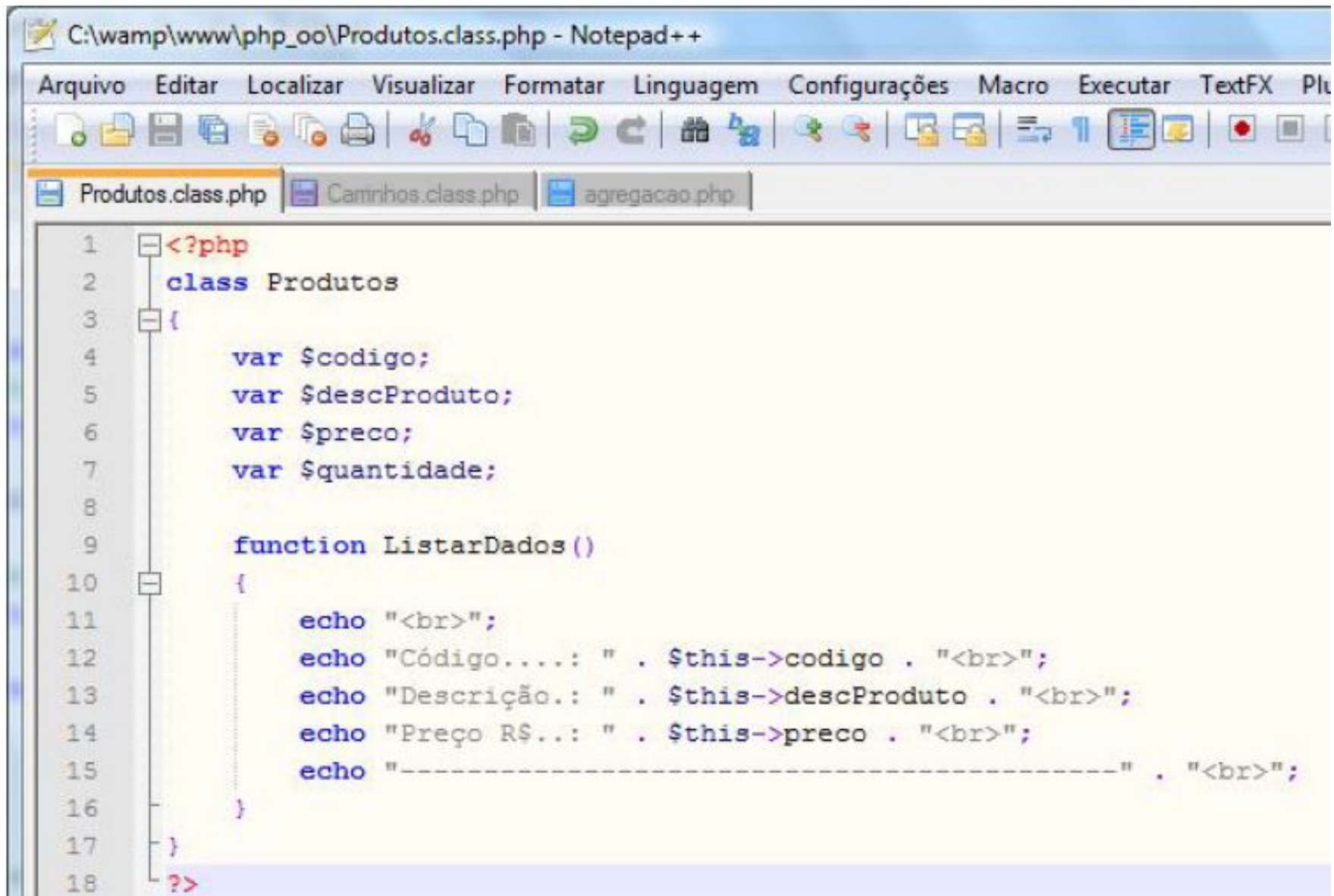
Produtos (parte) agrega Carrinhos (todo)



Agregação entre as Classes Produtos e Carrinhos

Relacionamento entre Classes

AGREGAÇÃO



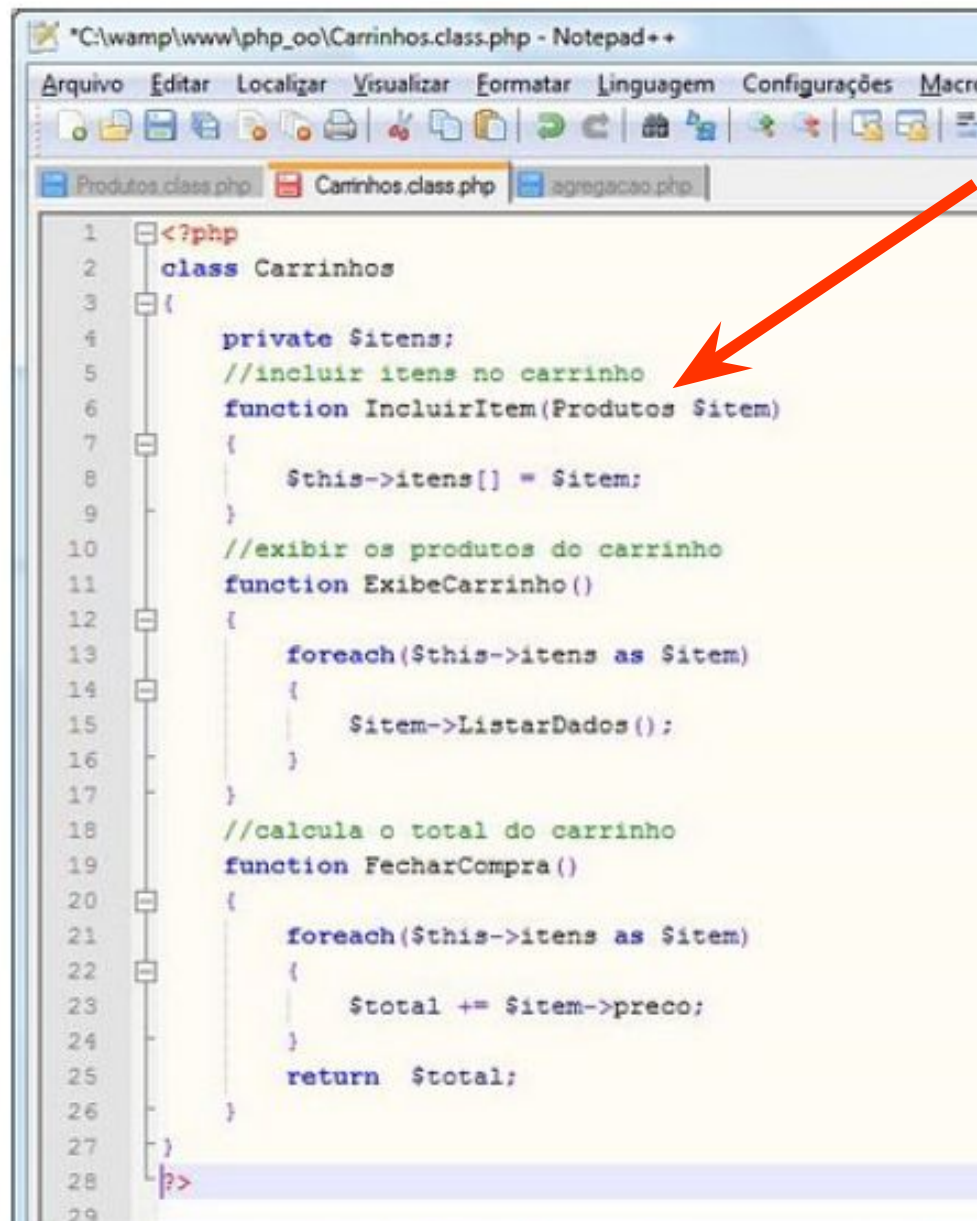
The screenshot shows a Notepad++ window with the title bar "C:\wamp\www\php_oo\Produtos.class.php - Notepad++". The menu bar includes "Arquivo", "Editar", "Localizar", "Visualizar", "Formatar", "Linguagem", "Configurações", "Macro", "Executar", "TextFX", and "Plu". The toolbar contains various icons for file operations, editing, and development. The tab bar shows three open files: "Produtos.class.php", "Caminhos.class.php", and "agregacao.php". The main text area displays the following PHP code:

```
1  <?php
2  class Produtos
3  {
4      var $codigo;
5      var $descProduto;
6      var $preco;
7      var $quantidade;
8
9      function ListarDados()
10     {
11         echo "<br>";
12         echo "Código.....: " . $this->codigo . "<br>";
13         echo "Descrição.: " . $this->descProduto . "<br>";
14         echo "Preço R$...: " . $this->preco . "<br>";
15         echo "-----" . "<br>";
16     }
17 }
18 ?>
```

Classe Produtos

Relacionamento entre Classes

AGREGAÇÃO

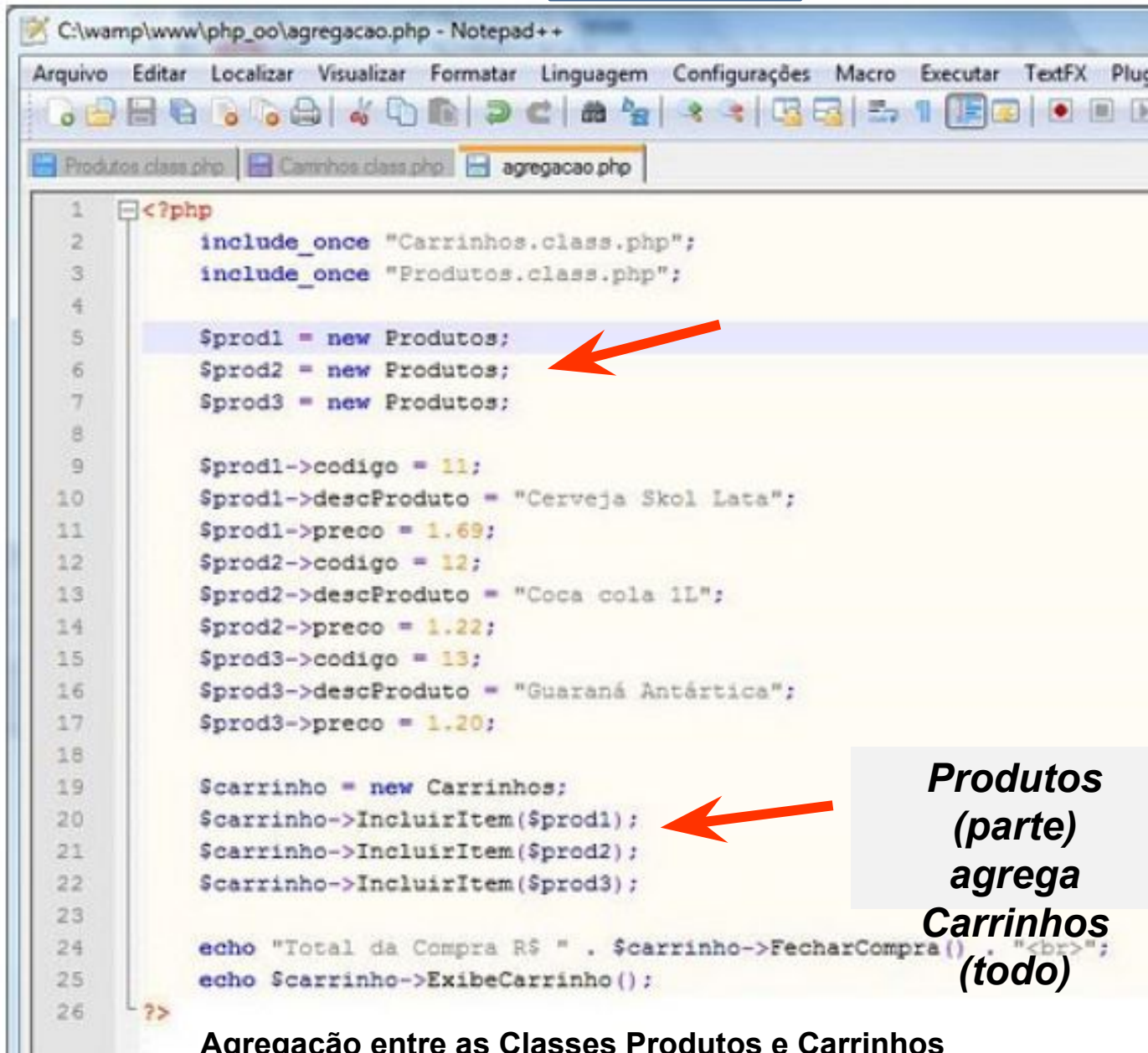


```
*C:\wamp\www\php_oo\Carrinhos.class.php - Notepad++  
Arquivo  Editar  Localizar  Visualizar  Formatar  Linguagem  Configurações  Macro  
Produtos.class.php  Carrinhos.class.php  agregacao.php  
1  <?php  
2  class Carrinhos  
3  {  
4      private $itens;  
5      //incluir itens no carrinho  
6      function IncluirItem(Produtos $item)  
7      {  
8          $this->itens[] = $item;  
9      }  
10     //exibir os produtos do carrinho  
11     function ExibeCarrinho()  
12     {  
13         foreach($this->itens as $item)  
14         {  
15             $item->ListarDados();  
16         }  
17     }  
18     //calcula o total do carrinho  
19     function FecharCompra()  
20     {  
21         foreach($this->itens as $item)  
22         {  
23             $total += $item->preco;  
24         }  
25         return $total;  
26     }  
27 }  
28 ?>  
29
```

Classe Carrinhos

Relacionamento entre Classes

AGREGAÇÃO



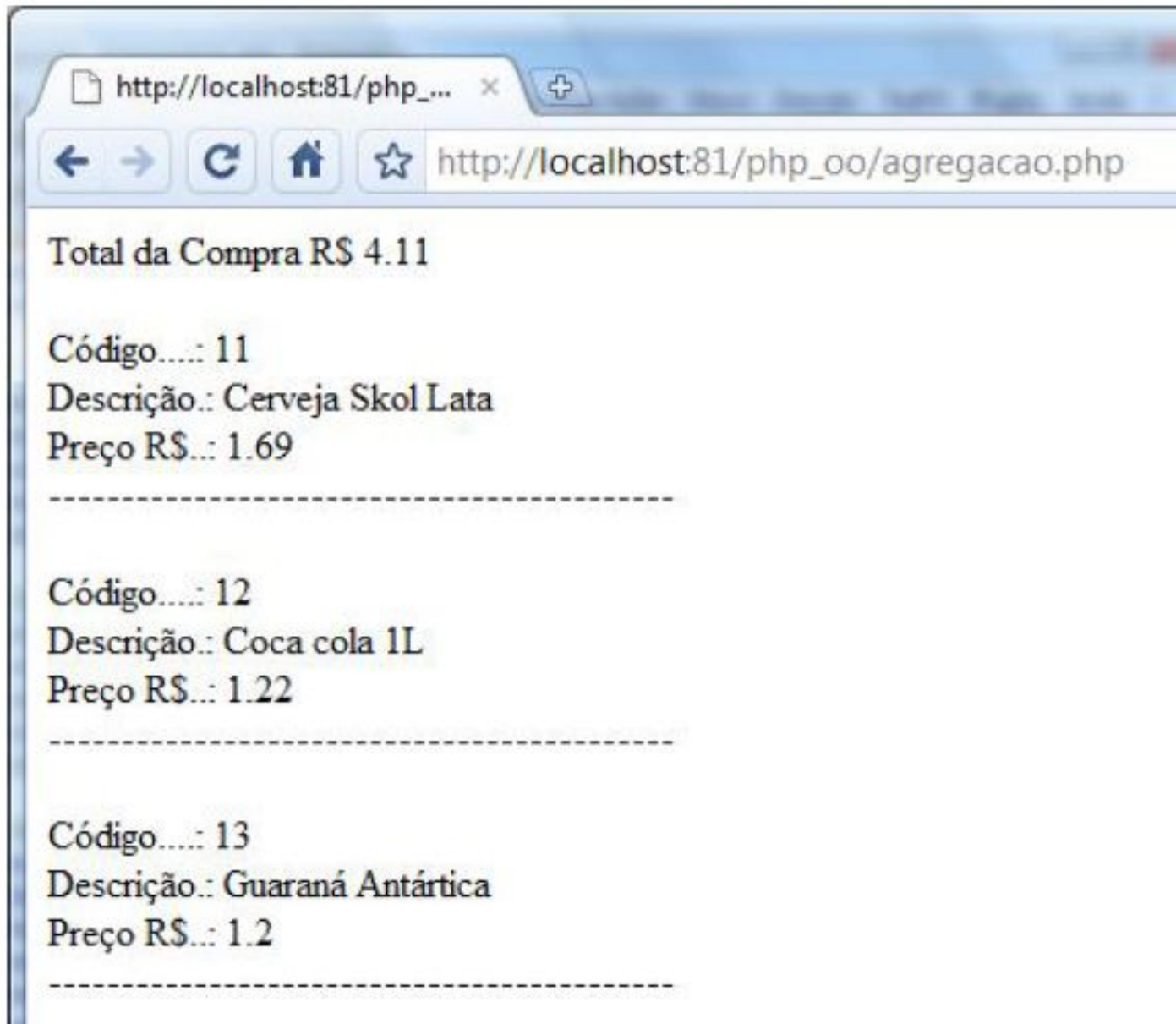
```
1 <?php
2     include_once "Carrinhos.class.php";
3     include_once "Produtos.class.php";
4
5     $prod1 = new Produtos;
6     $prod2 = new Produtos;
7     $prod3 = new Produtos;
8
9     $prod1->codigo = 11;
10    $prod1->descProduto = "Cerveja Skol Lata";
11    $prod1->preco = 1.69;
12    $prod2->codigo = 12;
13    $prod2->descProduto = "Coca cola 1L";
14    $prod2->preco = 1.22;
15    $prod3->codigo = 13;
16    $prod3->descProduto = "Guaraná Antártica";
17    $prod3->preco = 1.20;
18
19    $carrinho = new Carrinhos;
20    $carrinho->IncluirItem($prod1);
21    $carrinho->IncluirItem($prod2);
22    $carrinho->IncluirItem($prod3);
23
24    echo "Total da Compra R$ " . $carrinho->FecharCompra() . "<br>";
25    echo $carrinho->ExibeCarrinho();
26    ?>
```

**Produtos
(parte)
agrega
Carrinhos
(todo)**

Agregação entre as Classes Produtos e Carrinhos

Relacionamento entre Classes

AGREGAÇÃO



Resultado da Agregação entre as Classes Produtos e Carrinhos

Relacionamento entre Classes

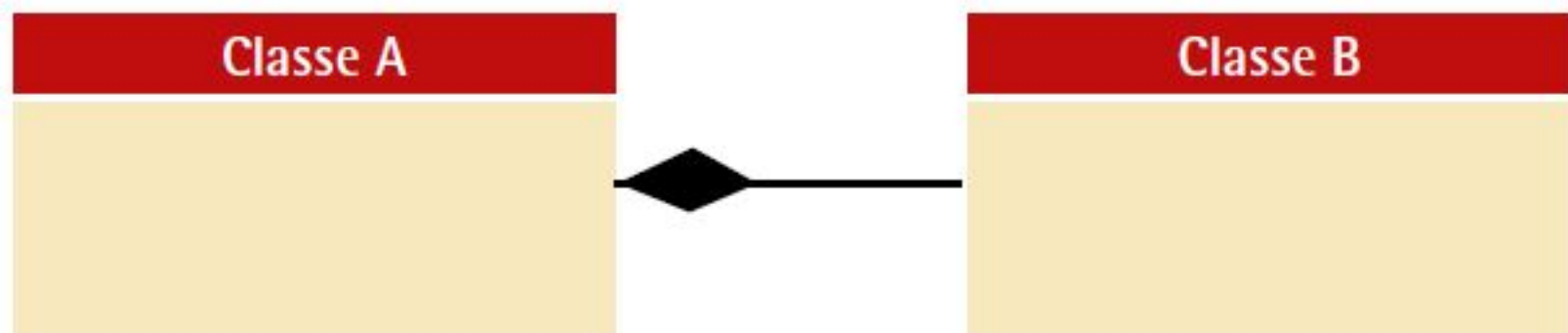
COMPOSIÇÃO

- A **composição** tem **exatamente** o **mesmo** conceito da **agregação**, ou seja, estabelece uma **relação todo-parte** entre dois **objetos**.
- A **única diferença** entre composição e agregação é que a **classe** que representa a **parte** da relação, para **existir**, **depende** da **classe** que representa o **todo**, ou seja, o ciclo de vida do objeto da classe parte depende do ciclo de vida do objeto da classe todo.

Relacionamento entre Classes

COMPOSIÇÃO

- Na **UML**, representamos a **composição** de forma **semelhante a agregação**, utilizando uma **reta saindo da classe** que representa a **parte** e se conectando a um **losango na classe** que representa o **todo**.
- Todavia, na **composição**, o **losango** é **preenchido**.

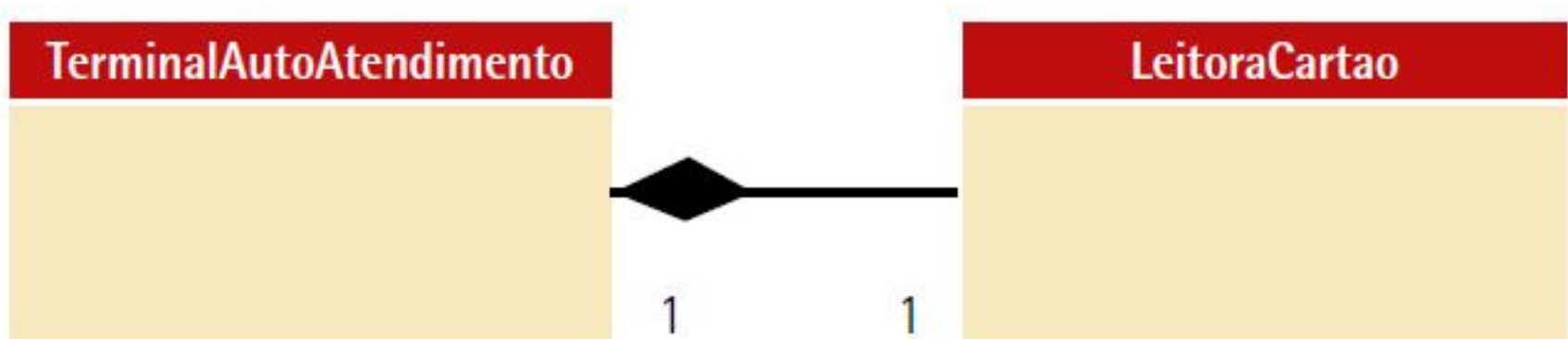


Representação de Composição na UML

Relacionamento entre Classes

COMPOSIÇÃO

- Assim **como** na **agregação**, na **composição** temos o conceito de **multiplicidade**, que define a **quantidade** de **objetos agregados** na **relação todo-parte**.

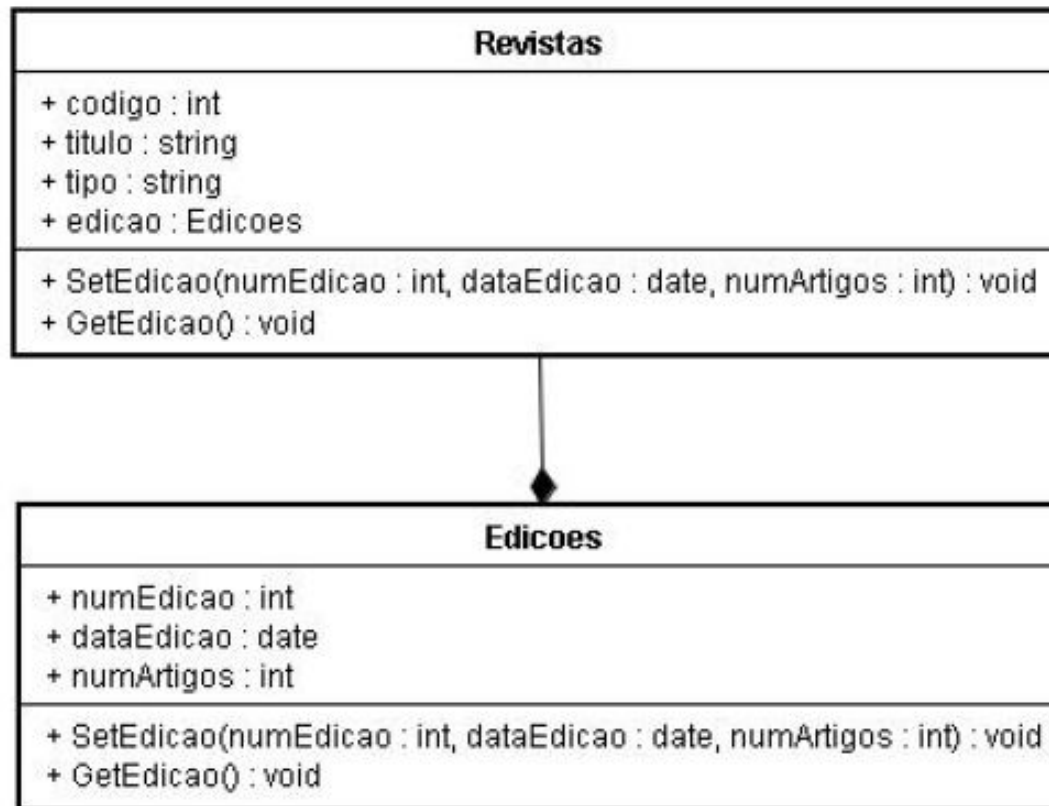


Exemplo de Agregação com Multiplicidade na UML

Relacionamento entre Classes

COMPOSIÇÃO

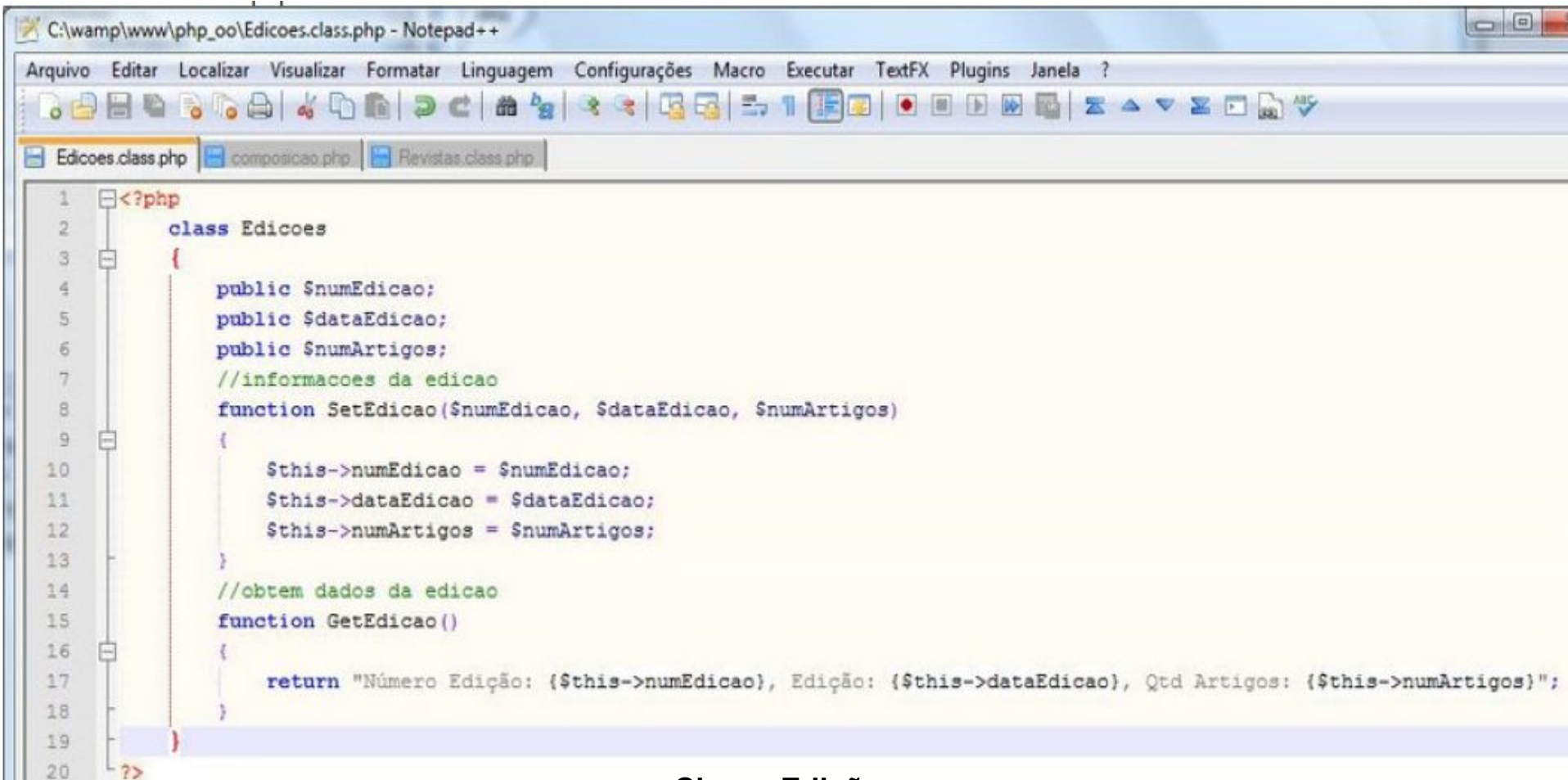
Exemplo: Neste exemplo vemos que não existem Edições sem Revistas



Composição entre as Classes Revistas e Edições

Relacionamento entre Classes

COMPOSIÇÃO



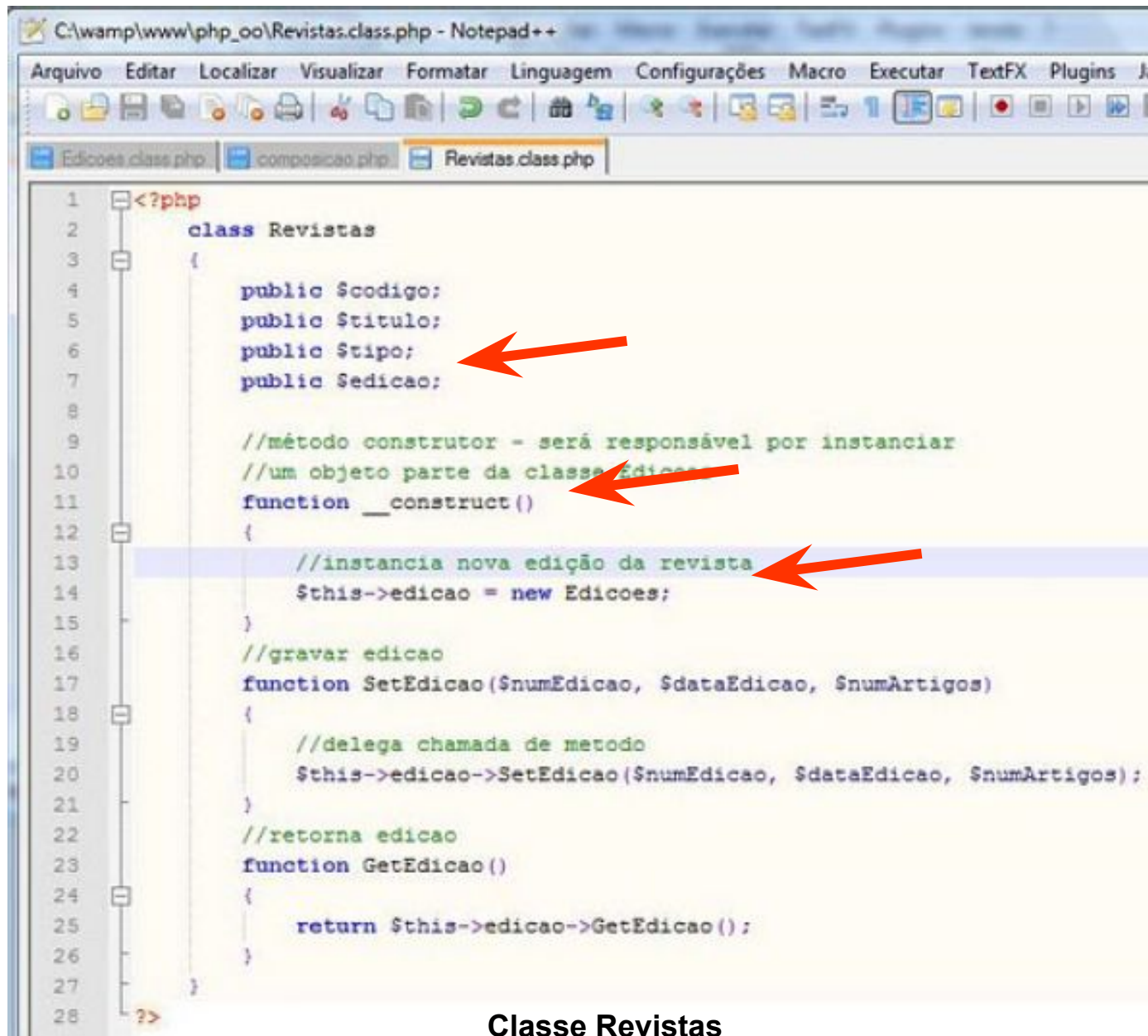
The screenshot shows a Notepad++ window with the file path `C:\wamp\www\php_oo\Edicoes.class.php`. The editor contains the following PHP code:

```
1 <?php
2 class Edicoes
3 {
4     public $numEdicao;
5     public $dataEdicao;
6     public $numArtigos;
7     //informacoes da edicao
8     function SetEdicao($numEdicao, $dataEdicao, $numArtigos)
9     {
10         $this->numEdicao = $numEdicao;
11         $this->dataEdicao = $dataEdicao;
12         $this->numArtigos = $numArtigos;
13     }
14     //obtem dados da edicao
15     function GetEdicao()
16     {
17         return "Número Edição: {$this->numEdicao}, Edição: {$this->dataEdicao}, Qtd Artigos: {$this->numArtigos}";
18     }
19 }
20 ?>
```

Classe Edições

Relacionamento entre Classes

COMPOSIÇÃO

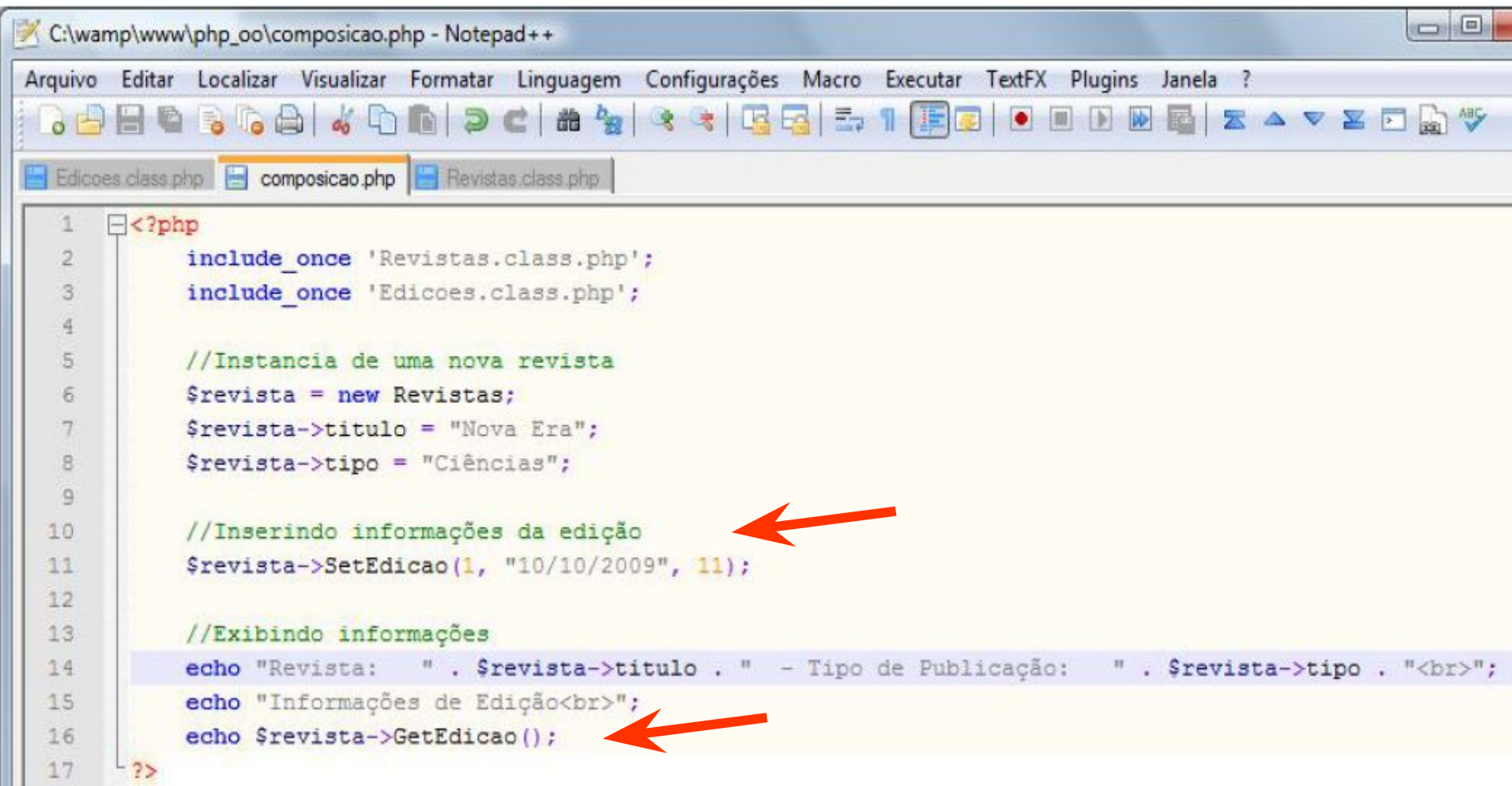


```
1 <?php
2 class Revistas
3 {
4     public $codigo;
5     public $titulo;
6     public $tipo;
7     public $edicao;
8
9     //método construtor - será responsável por instanciar
10    //um objeto parte da classe Edicoes
11    function __construct()
12    {
13        //instancia nova edição da revista
14        $this->edicao = new Edicoes;
15    }
16
17    //gravar edicao
18    function SetEdicao($numEdicao, $dataEdicao, $numArtigos)
19    {
20        //delega chamada de metodo
21        $this->edicao->SetEdicao($numEdicao, $dataEdicao, $numArtigos);
22    }
23
24    //retorna edicao
25    function GetEdicao()
26    {
27        return $this->edicao->GetEdicao();
28    }
29 }
```

Classe Revistas

Relacionamento entre Classes

COMPOSIÇÃO

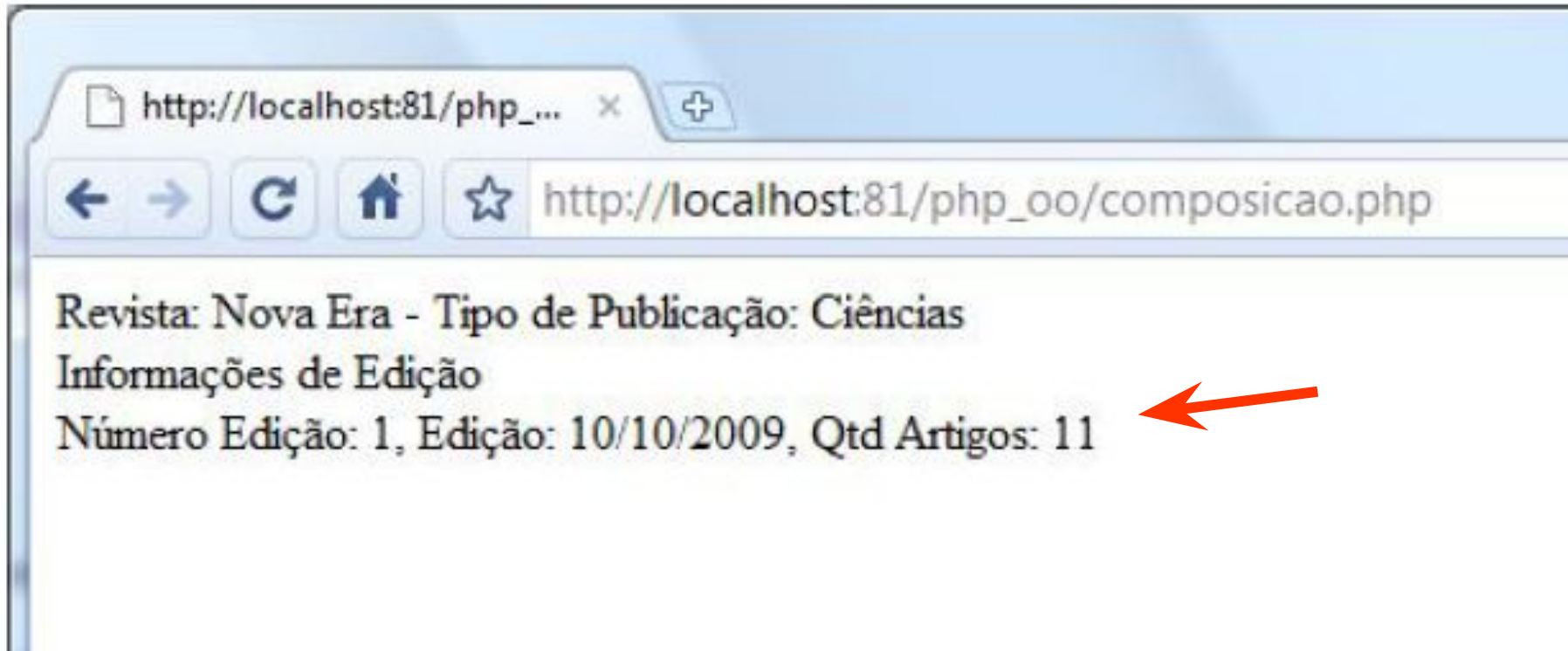


```
1 <?php
2     include_once 'Revistas.class.php';
3     include_once 'Edicoes.class.php';
4
5     //Instancia de uma nova revista
6     $revista = new Revistas;
7     $revista->titulo = "Nova Era";
8     $revista->tipo = "Ciências";
9
10    //Inserindo informações da edição
11    $revista->SetEdicao(1, "10/10/2009", 11);
12
13    //Exibindo informações
14    echo "Revista:  " . $revista->titulo . " - Tipo de Publicação:  " . $revista->tipo . "<br>";
15    echo "Informações de Edição<br>";
16    echo $revista->GetEdicao();
17    ?>
```

Composição entre as Classes Revistas e Edições

Relacionamento entre Classes

COMPOSIÇÃO

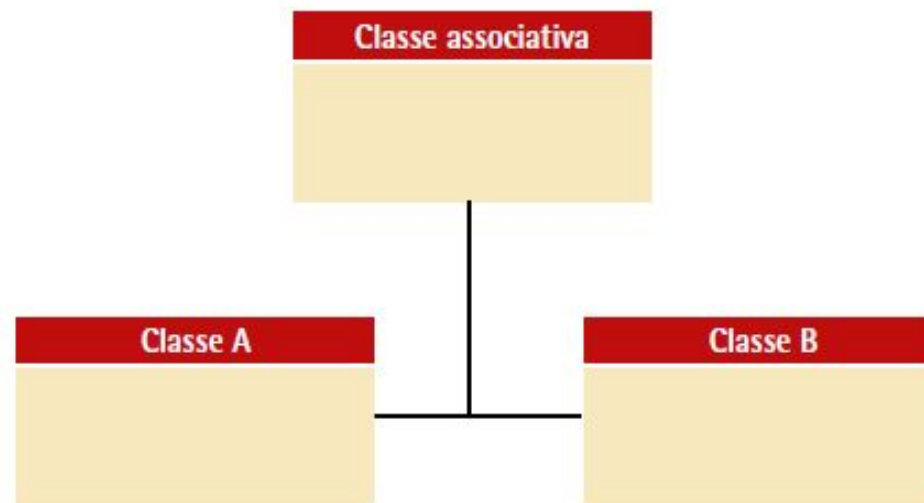


Resultado da Composição entre as Classes Revistas e Edições

Relacionamento entre Classes

CLASSES ASSOCIATIVAS

- **Utilizamos** classes **associativas** quando desejamos **armazenar informações importantes** de uma **associação**.
- Na **UML**, **representamos** uma classe **associativa** da **mesma forma** que representamos uma **classe “normal”**, como mostra a figura a seguir.

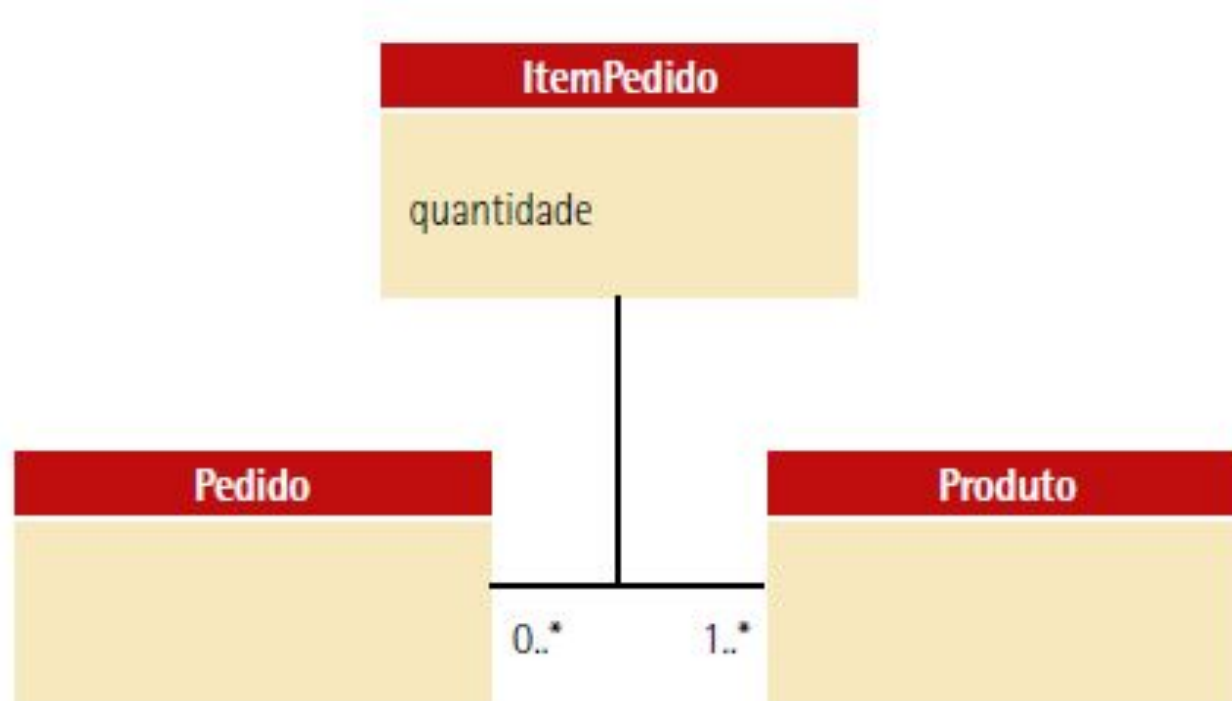


Representação de Classes Associativas UML

Relacionamento entre Classes

CLASSES ASSOCIATIVAS

- Um **exemplo clássico** na utilização de **classes associativas** pode ser notado na figura a seguir:



Exemplo de Classes Associativas UML

Relacionamento entre Classes

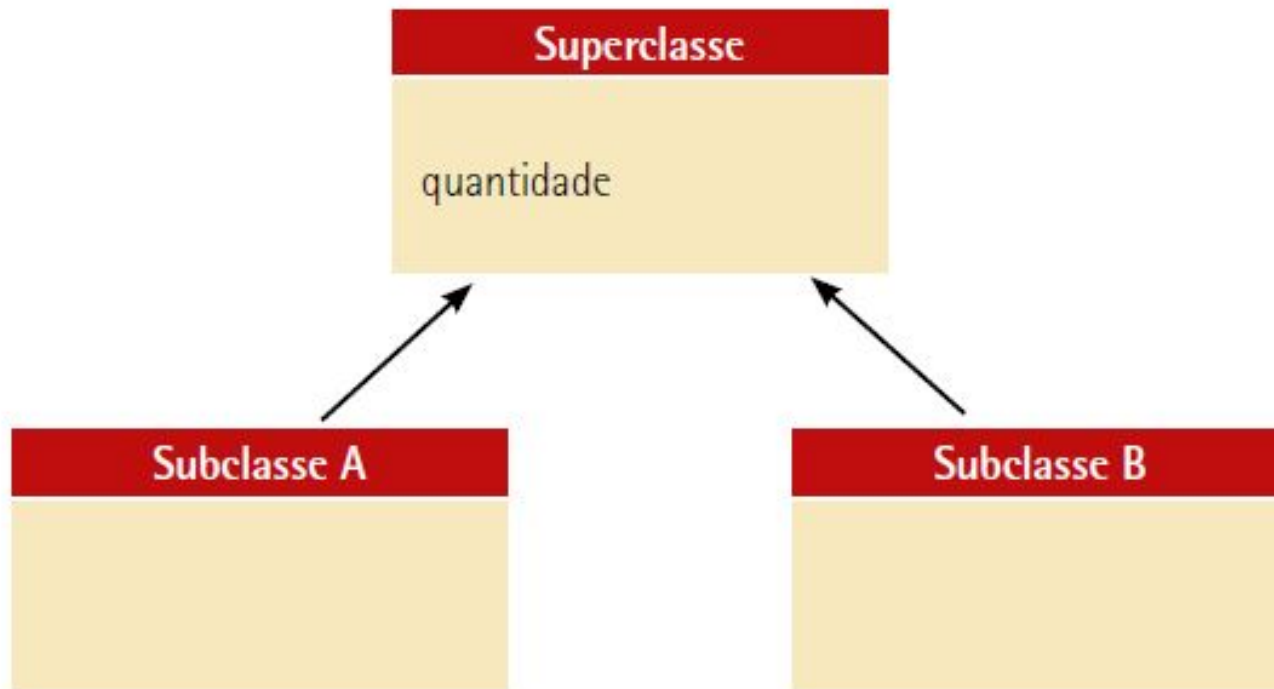
HERANÇA

- Na O.O., um **objeto** pode **herdar características e comportamentos** de **outro objeto**.
- Traduzindo para a terminologia correta, um **objeto** pode **herdar atributos e métodos** de **outro objeto**.
- Nessa situação temos **dois papéis** de **classes** na relação, chamados de **classe mãe** e **classe filha**.
- Classe **mãe** (ou superclasse), possui **atributos e métodos** que podem ser **herdados** por **uma ou mais classes filhas**, (ou subclasses).

Relacionamento entre Classes

HERANÇA

- Na **UML**, representamos **herança** com uma **seta direcional**, sendo que a **ponta da seta é cheia**, como mostra a figura:

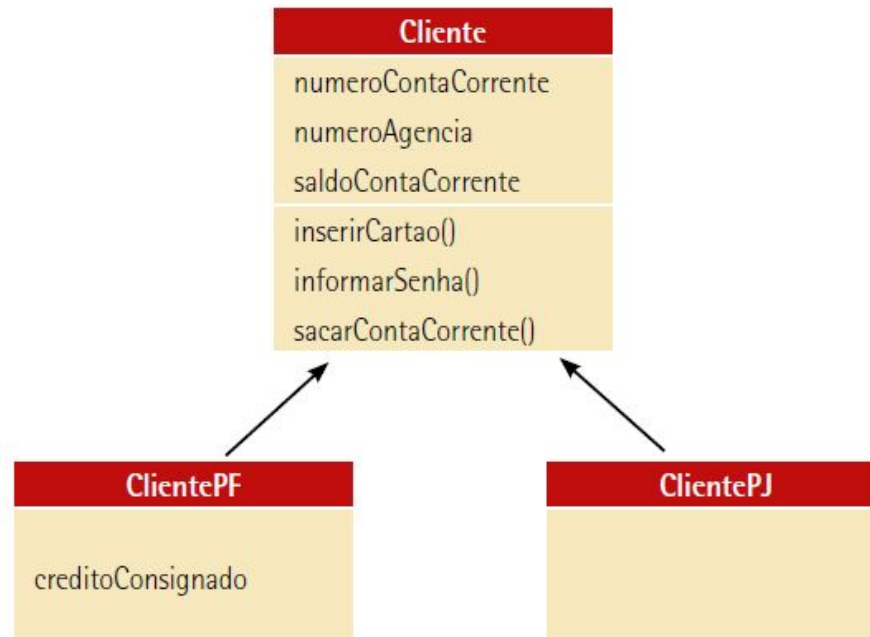


Representação de Herança na UML

Relacionamento entre Classes

HERANÇA

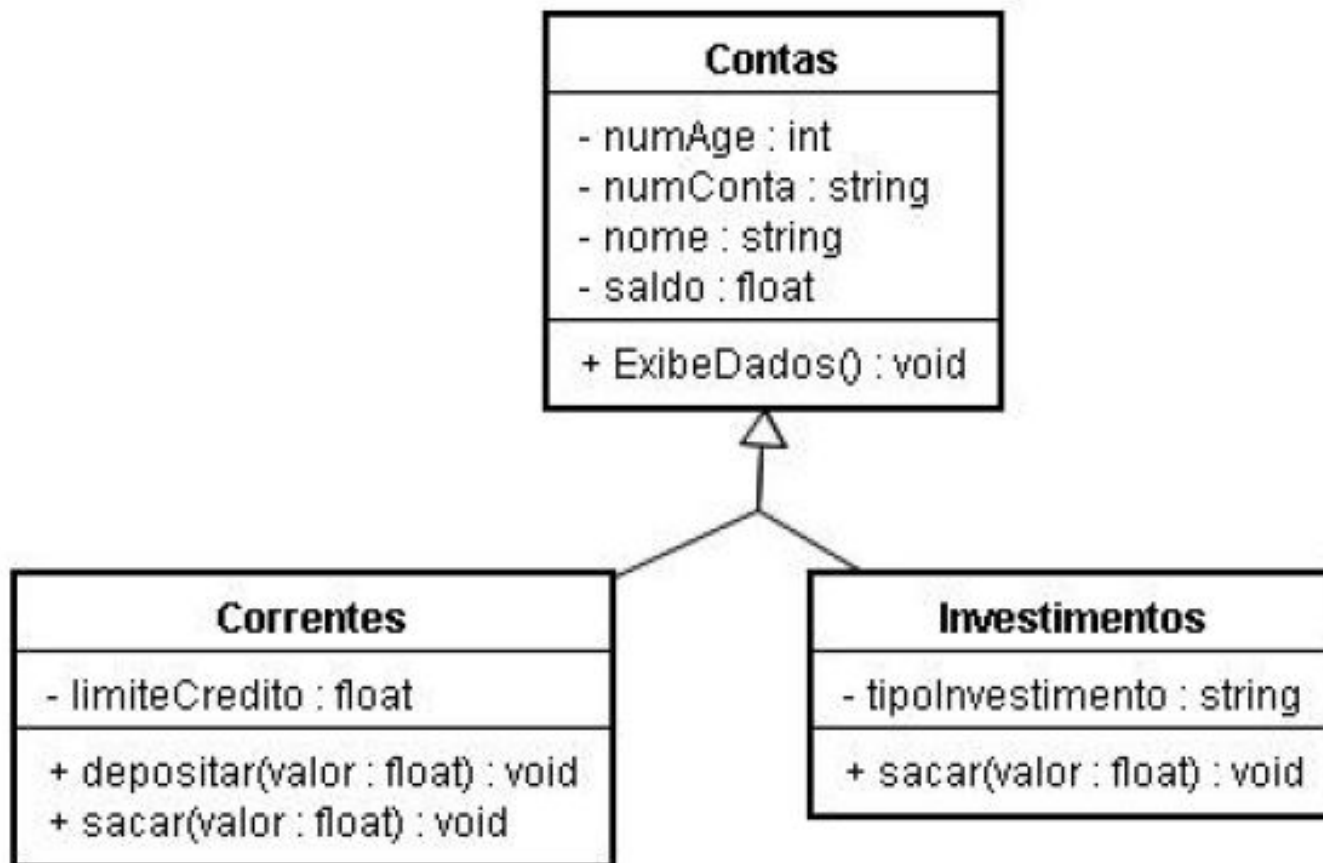
- Aqui um **exemplo** classe **Cliente**, que representa **todos** os **clientes** de um banco, que podem **efetuar saque** em terminais de autoatendimento, e **duas especializações** dessa classe: **Cliente Pessoa Física** e **Cliente Pessoa Jurídica**.



Exemplo de Herança na UML

Relacionamento entre Classes

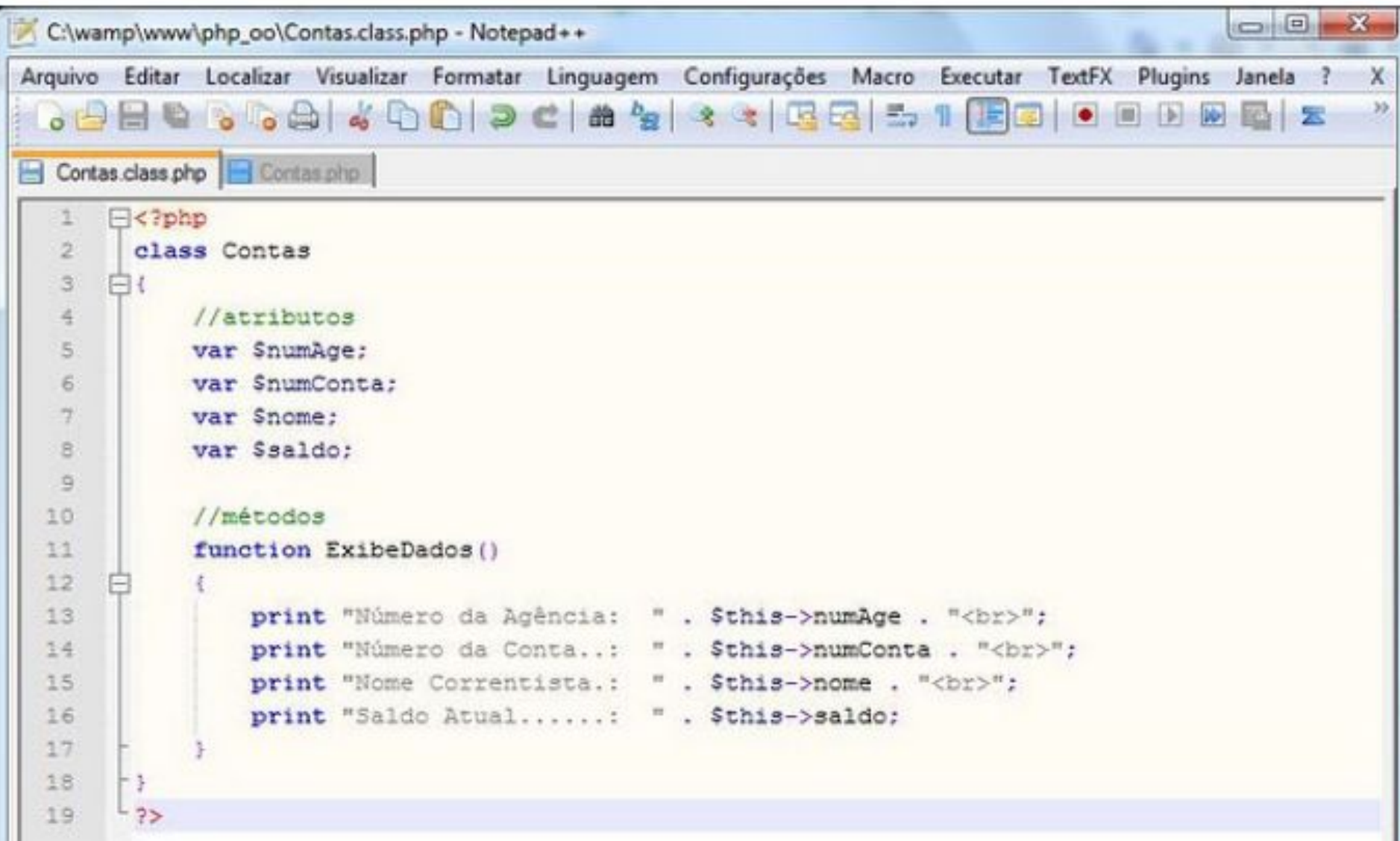
HERANÇA



Exemplo de Herança

Relacionamento entre Classes

HERANÇA



The screenshot shows a Notepad++ window with the file 'C:\wamp\www\php_oo\Contas.class.php'. The code defines a PHP class named 'Contas'. It includes attributes for agency number, account number, name, and balance, and a method to display these details. The code is as follows:

```
1 <?php
2 class Contas
3 {
4     //atributos
5     var $numAge;
6     var $numConta;
7     var $nome;
8     var $saldo;
9
10    //métodos
11    function ExibeDados()
12    {
13        print "Número da Agência: " . $this->numAge . "<br>";
14        print "Número da Conta..: " . $this->numConta . "<br>";
15        print "Nome Correntista.: " . $this->nome . "<br>";
16        print "Saldo Atual.....: " . $this->saldo;
17    }
18 }
19 ?>
```

Classe Mãe: Contas

Relacionamento entre Classes

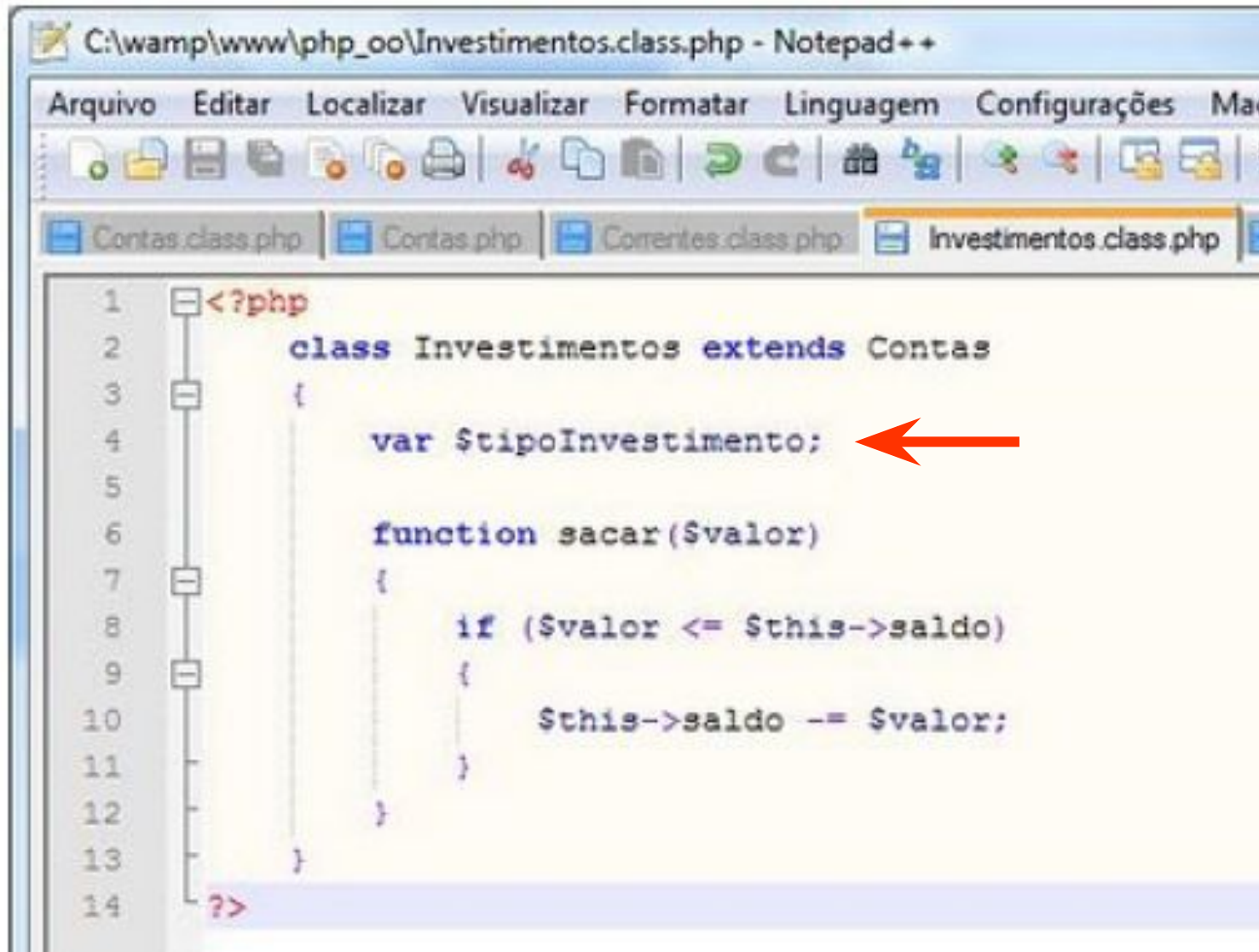
HERANÇA

```
1 <?php
2 class Correntes extends Contas
3 {
4     var $limiteCredito;
5
6     function depositar($valor)
7     {
8         $this->saldo += $valor;
9     }
10    function sacar($valor)
11    {
12        if($valor <= $this->saldo)
13        {
14            $this->saldo -= $valor;
15        }
16    }
17 }
18 ?>
```

Classe Filha: Correntes

Relacionamento entre Classes

HERANÇA

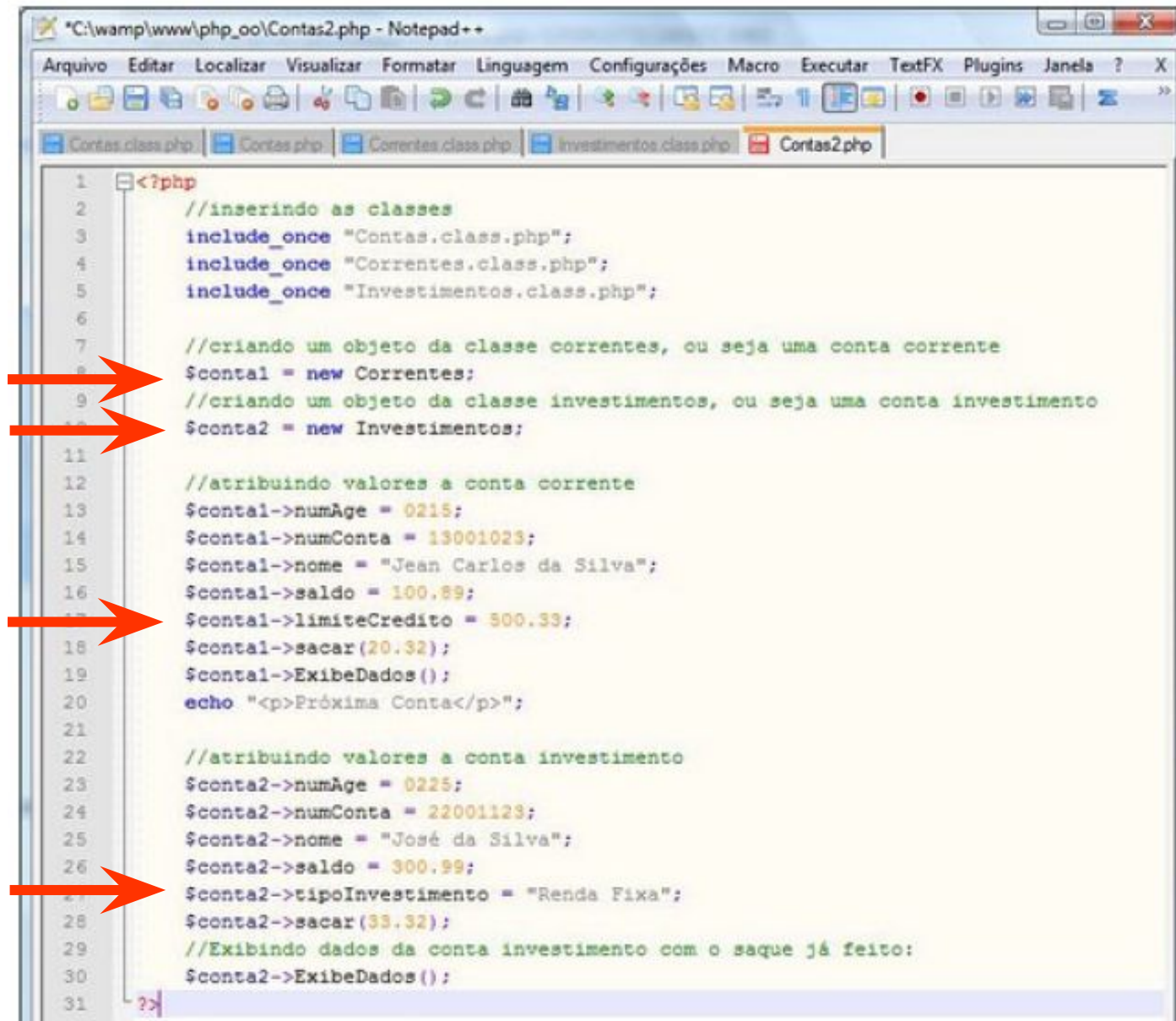


```
1  <?php
2      class Investimentos extends Contas
3      {
4          var $tipoInvestimento;
5
6          function sacar($valor)
7          {
8              if ($valor <= $this->saldo)
9              {
10                 $this->saldo -= $valor;
11             }
12         }
13     }
14  ?>
```

Classe Filha: Investimentos

Relacionamento entre Classes

HERANÇA



```
*C:\wamp\www\php_oo\Contas2.php - Notepad++
Arquivo  Editar  Localizar  Visualizar  Formatar  Linguagem  Configurações  Macro  Executar  TextFX  Plugins  Janela  ?  X

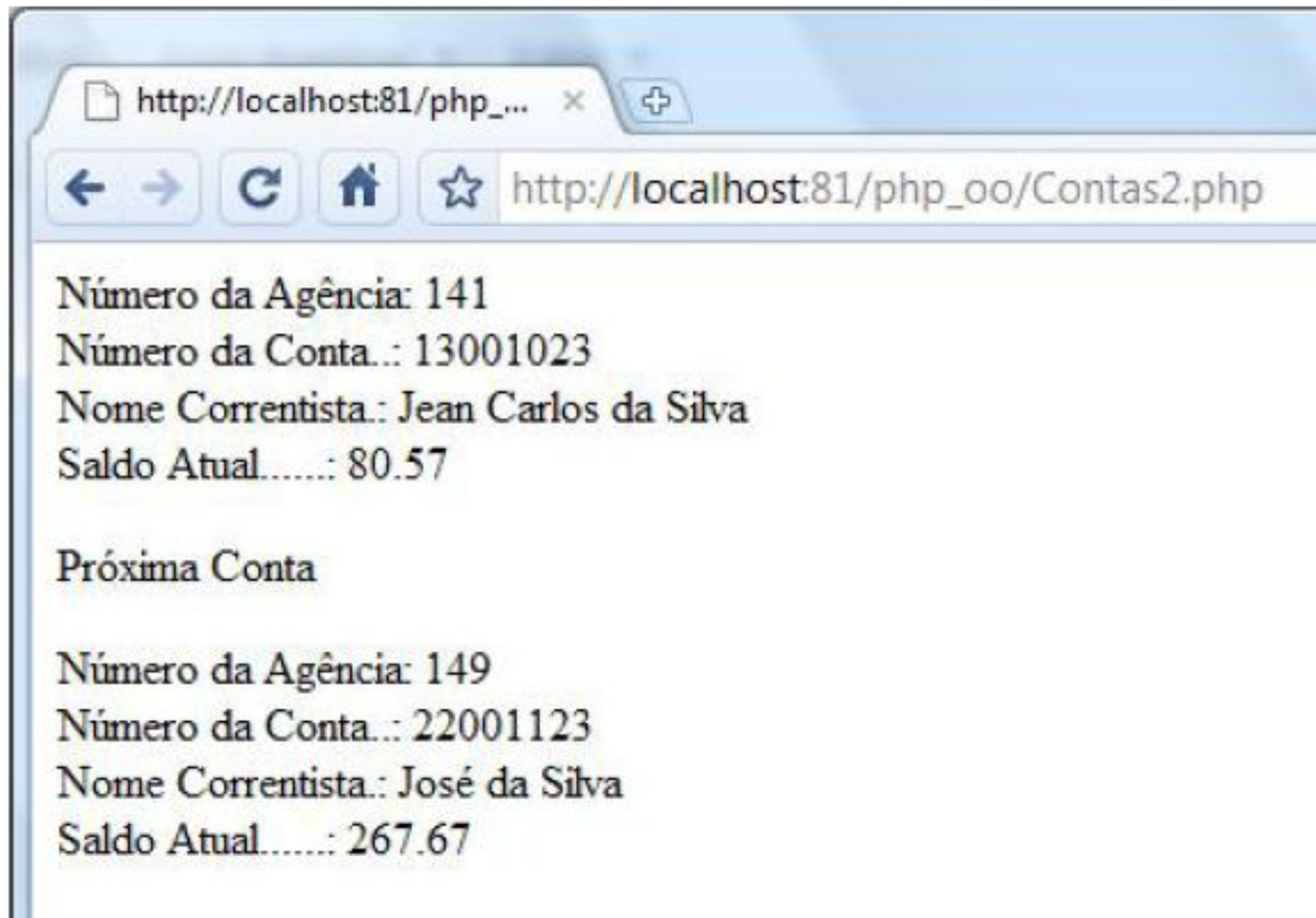
Contas.class.php  Contas.php  Correntes.class.php  Investimentos.class.php  Contas2.php

1  <?php
2      //inserindo as classes
3      include_once "Contas.class.php";
4      include_once "Correntes.class.php";
5      include_once "Investimentos.class.php";
6
7      //criando um objeto da classe correntes, ou seja uma conta corrente
8      $contal = new Correntes;
9      //criando um objeto da classe investimentos, ou seja uma conta investimento
10     $conta2 = new Investimentos;
11
12     //atribuindo valores a conta corrente
13     $contal->numAge = 0215;
14     $contal->numConta = 13001023;
15     $contal->nome = "Jean Carlos da Silva";
16     $contal->saldo = 100.89;
17     $contal->limiteCredito = 500.33;
18     $contal->sacar(20.32);
19     $contal->ExibeDados();
20     echo "<p>Próxima Conta</p>";
21
22     //atribuindo valores a conta investimento
23     $conta2->numAge = 0225;
24     $conta2->numConta = 22001123;
25     $conta2->nome = "José da Silva";
26     $conta2->saldo = 300.99;
27     $conta2->tipoInvestimento = "Renda Fixa";
28     $conta2->sacar(35.32);
29     //Exibindo dados da conta investimento com o saque já feito:
30     $conta2->ExibeDados();
31     ?>
```

Classe Filha: Correntes

Relacionamento entre Classes

HERANÇA



Classe Filha: Correntes

Diagrama de Classes

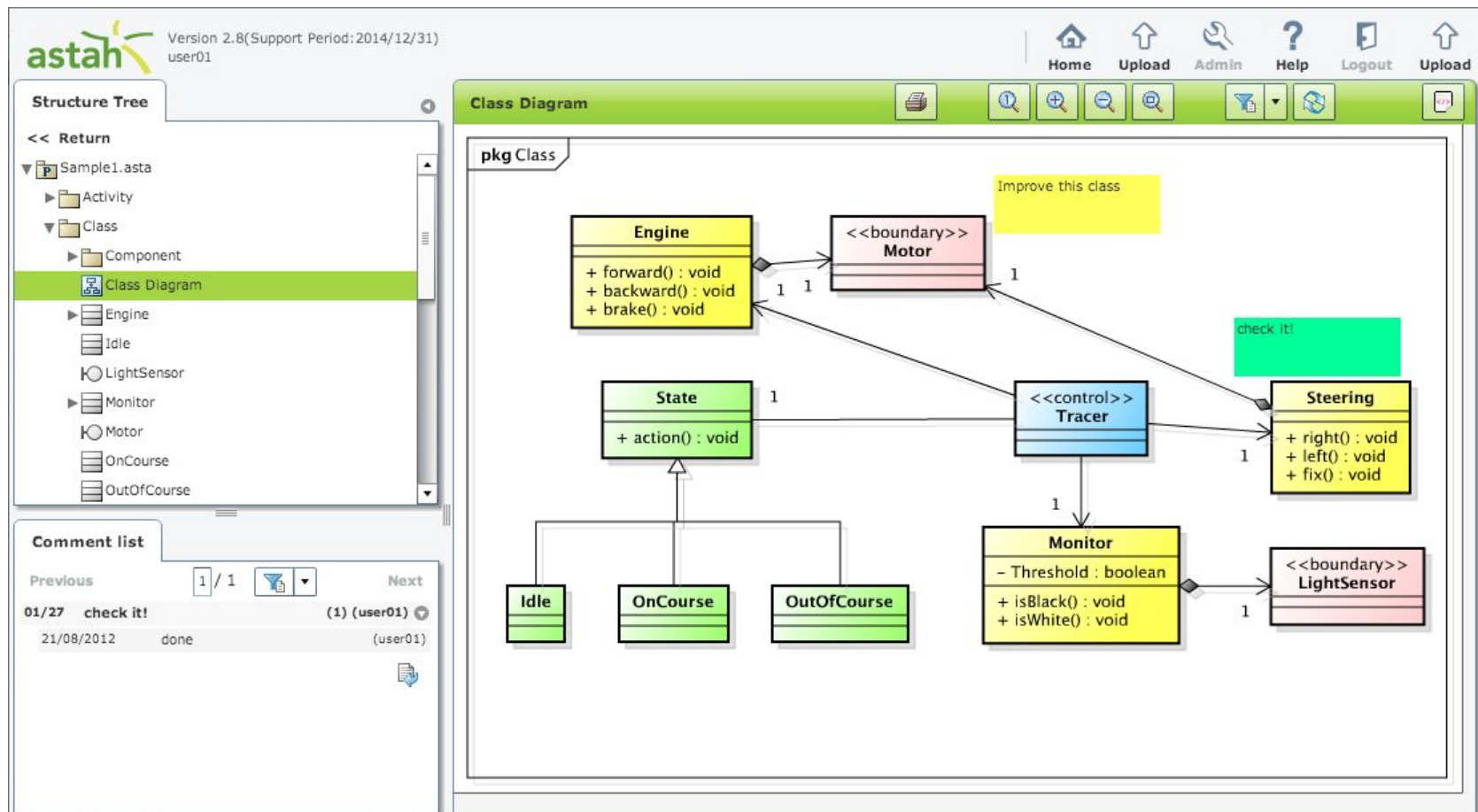
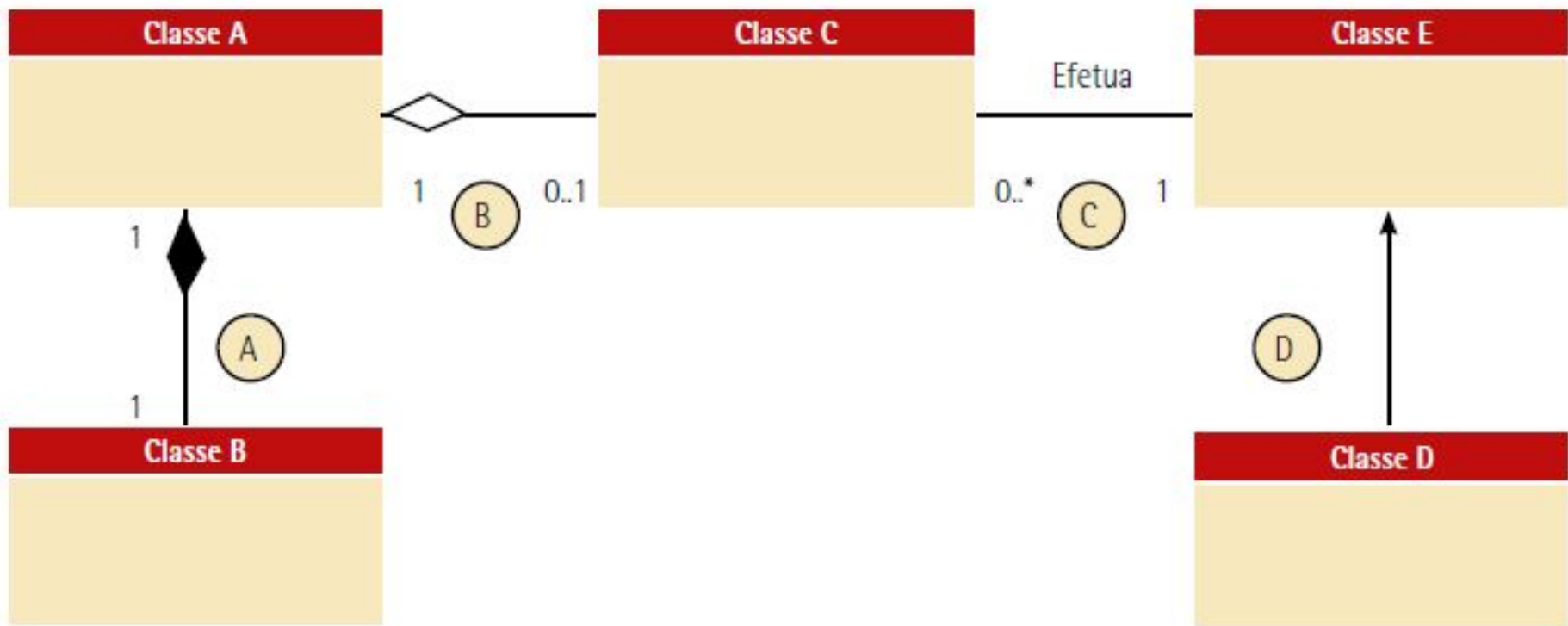


Diagrama de Classes

- O **diagrama de classes** é um diagrama **UML** que tem como **objetivo representar a estrutura estática** das classes de um **sistema de software**.



Exemplo de diagrama de classes na UML

Diagrama de Classes

Letra	Conceitos	Interpretações
A	<ul style="list-style-type: none"> - Composição - Multiplicidade 	<ul style="list-style-type: none"> - Um objeto da Classe A é composto por no máximo um objeto da Classe B. - Um objeto da Classe B compõe no máximo um objeto da Classe A. - O objeto da Classe B só existe se um objeto da Classe A também existir.
B	<ul style="list-style-type: none"> - Agregação - Multiplicidade 	<ul style="list-style-type: none"> - Um objeto da Classe A agrega em sua estrutura no mínimo zero e no máximo um objeto da Classe C. - Um objeto da Classe C agrega no máximo um objeto da Classe A. - O objeto da Classe C existe independentemente do objeto da Classe A.
C	<ul style="list-style-type: none"> - Associação - Multiplicidade - Navegabilidade - Papéis 	<ul style="list-style-type: none"> - Um objeto da Classe C possui uma associação com um objeto da Classe E. Essa associação é identificada utilizando o verbo "Efetua". - A identificação da associação define os papéis dos objetos nessa associação. No exemplo, quem é o sujeito e o predicado do "efetuar". - A direção da associação define o sentido da associação, a navegabilidade da associação. - A identificação da multiplicidade indica que um objeto da Classe C efetua no máximo um objeto da Classe E da mesma forma que um objeto da Classe E é efetuado por no mínimo zero objetos da Classe C e no máximo infinitos objetos dessa classe.
D	<ul style="list-style-type: none"> - Herança - Hierarquia de Classes - Especialização - Generalização 	<ul style="list-style-type: none"> - Um objeto da Classe D herda os atributos e métodos de um objeto da Classe E. - A Classe D é um tipo da Classe E. - A Classe E é uma generalização da Classe D. - A Classe D é uma especialização da Classe E.

Conceitos e interpretações para o diagrama de classes

Diagrama de Classes

- A figura a seguir mostra um **possível diagrama de classes** do modelo de domínio para o problema de **saque** em um **terminal de autoatendimento**.
- Note que estamos **representando apenas** a visão da **operação de saque**, **embora**, em um terminal de autoatendimento, tenhamos **outras operações**, como **deposito**, e outros **periféricos**, como **impressora** ou **depositário** de envelopes.

É comum que tenhamos **diversos diagramas** que representem **visões parciais** do domínio, mas que em **conjunto** representem a **visão do todo**.

Diagrama de Classes

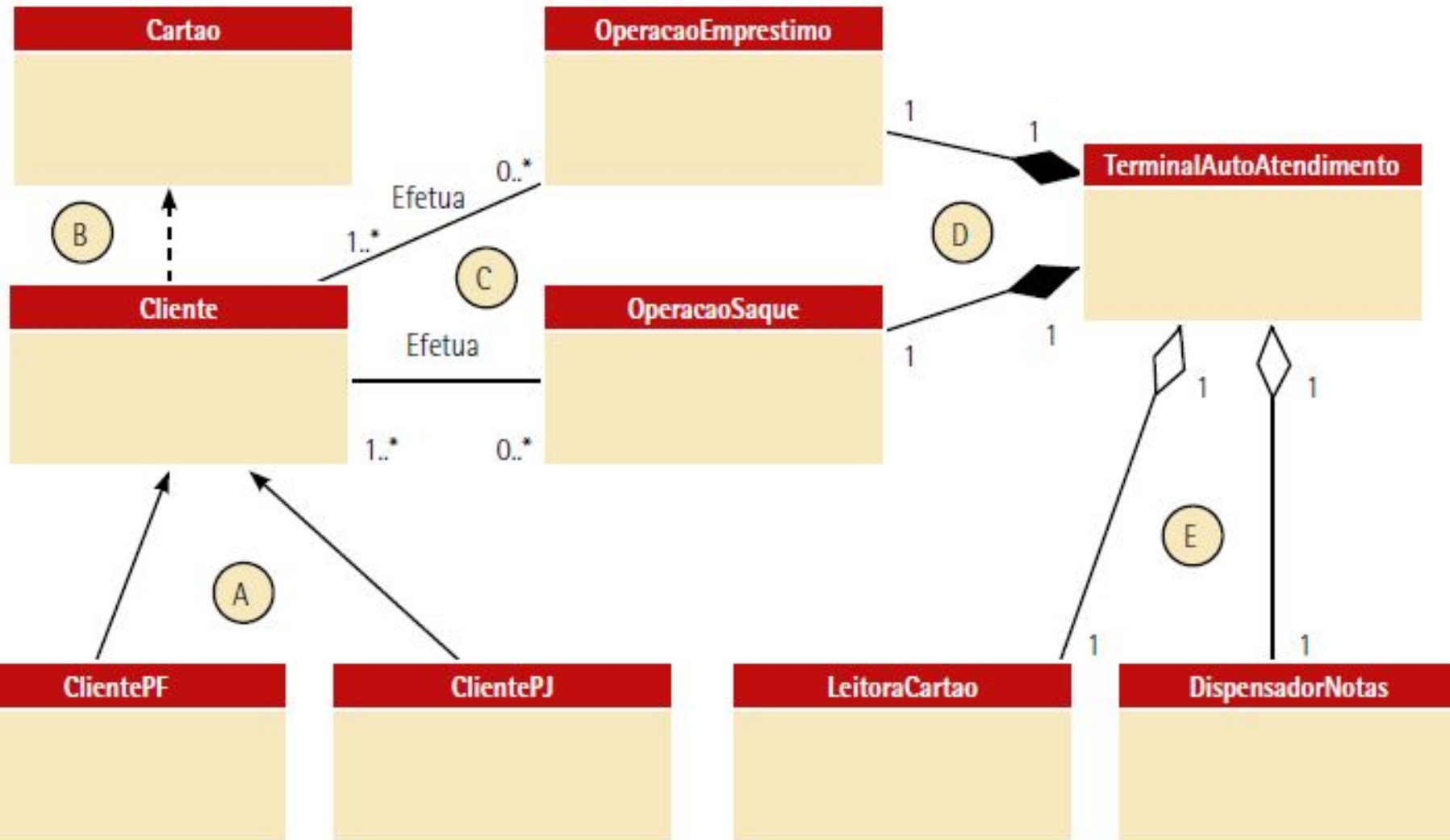


Diagrama de classes do saque em autoatendimento

Diagrama de Classes

Letra	Conceitos	Interpretações
A	<ul style="list-style-type: none"> - Herança - Hierarquia de Classes - Especialização - Generalização 	<ul style="list-style-type: none"> - Cliente Pessoa Jurídica e Cliente Pessoa Física herdam atributos e métodos de Cliente. - Cliente Pessoa Jurídica e Cliente Pessoa Física herdam as mesmas associações e dependência de Cliente. - Cliente Pessoa Jurídica e Cliente Pessoa Física são tipos de Cliente. - Cliente Pessoa Jurídica e Cliente Pessoa Física são especializações de Cliente. - Cliente é uma generalização de Cliente Pessoa Jurídica e Cliente Pessoa Física.
B	<ul style="list-style-type: none"> - Dependência - Navegabilidade 	<ul style="list-style-type: none"> - Cliente depende de um objeto do tipo Cartão para realização de algum comportamento. - A direção da seta da dependência dá sentido e navegabilidade na dependência.
C	<ul style="list-style-type: none"> - Associação - Multiplicidade - Navegabilidade - Papéis 	<ul style="list-style-type: none"> - Um Cliente efetua no mínimo zero e no máximo inúmeras operações de empréstimo. - Um Cliente efetua no mínimo zero e no máximo inúmeras operações de saque. - Operações de Saque e de Empréstimo são efetuadas por no mínimo um Cliente e no máximo por inúmeros.

Conceitos e interpretações diagrama de classes: efetuar saque

Diagrama de Classes

D	<ul style="list-style-type: none">- Composição- Multiplicidade	<ul style="list-style-type: none">- Um terminal de autoatendimento é composto por algumas operações, entre elas: Empréstimo e Saque (*).- Falando em multiplicidade, um terminal é composto por apenas uma operação de Empréstimo e uma operação de Saque. Assim como um objeto que represente essas operações compõe apenas um objeto do tipo Terminal de Autoatendimento.- Os objetos das operações de Empréstimo e Saque não existem se um objeto do tipo Terminal de Autoatendimento não existir.
E	<ul style="list-style-type: none">- Agregação- Multiplicidade	<ul style="list-style-type: none">- Um objeto Terminal de Autoatendimento agrega em sua estrutura objetos do tipo Leitor de Cartão e Dispensador de Notas.- Falando em multiplicidade, um Terminal de Autoatendimento agrega em sua estrutura apenas uma Leitora de Cartão e apenas um Dispensador de Notas. Assim como uma Leitora de Cartão ou um Dispensador de Notas agrega apenas um objeto do tipo Terminal de Autoatendimento.- Objetos do tipo Leitora de Cartão e do tipo Dispensador de Notas existem independentemente da existência de objetos do tipo Terminal de Autoatendimento.

Conceitos e interpretações diagrama de classes: efetuar saque

Atividade 6

O próximo passo agora será realizar a Atividade 6.

Nos vemos nela!



Dúvidas!

Profº. Me. Flávio Henrique Fernandes Volpon
flavio.volpon@docente.unip.br





Obrigado!

Bibliografia

I - BIBLIOGRAFIA BÁSICA

- PAULA FILHO, W. de P. **Engenharia de software: fundamentos, métodos e padrões**. 3.ed. Rio de Janeiro: LTC, 2012.
- PRESSMAN, R. S. **Engenharia de Software: Uma Abordagem Profissional**. 7. ed. AMGH, 2011.
- SOMMERVILLE, I. **Engenharia de Software**. 9.ed. São Paulo: Adison-Wesley, 2011.

II - BIBLIOGRAFIA COMPLEMENTAR

- PRIKLADNICKI., Rafael, WILLI, Renato, and MILANI, Fabiano. **Métodos Ágeis para Desenvolvimento de Software**. Bookman, 2014.
- COHN, M. **Desenvolvimento de Software com Scrum**. Bookman, 2011.
- SCHACH, S. R. **Engenharia de software: os paradigmas clássico e orientado a objetos**. 7.ed. São Paulo: McGraw-Hill, 2009.
- HIRAMA, K. **Engenharia de software: qualidade e produtividade com tecnologia**. Rio de janeiro, campus, 2011.
- WAZLAWICK, R. **Engenharia de software: conceitos e práticas**. Rio de janeiro, campus, 2009.