



# Implementando Enterprise Patterns em PHP

**Pablo Dall'Oglio**  
Adianti Solutions  
[www.adianti.com.br](http://www.adianti.com.br)



- 2004 :: FISL5.0
  - Aplicações Gráficas com PHP-GTK;
- 2005 :: FISL6.0
  - As Novidades do PHP5;
- 2008 :: FISL9.0
  - Enterprise Patterns em PHP;



**O que são padrões ?**  
google sabe a resposta !!





## Padrões de Bar

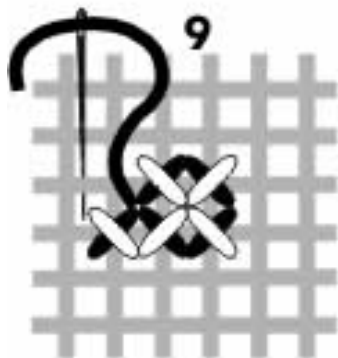
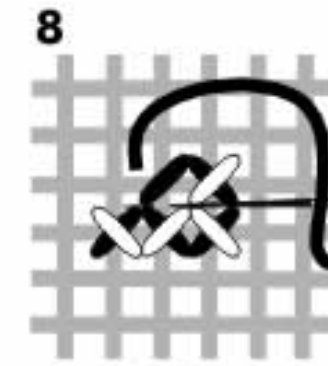
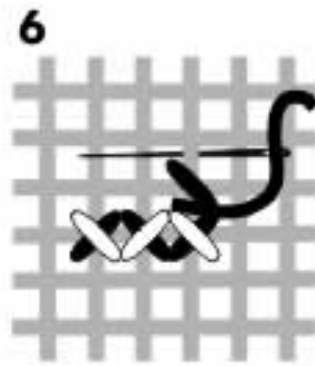
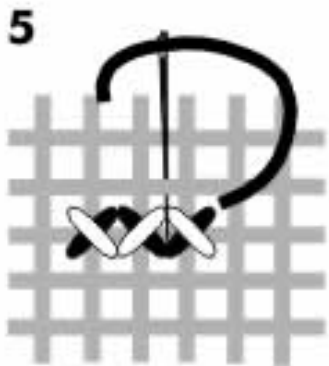
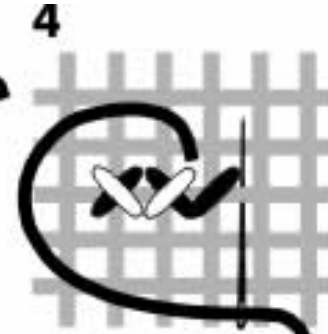
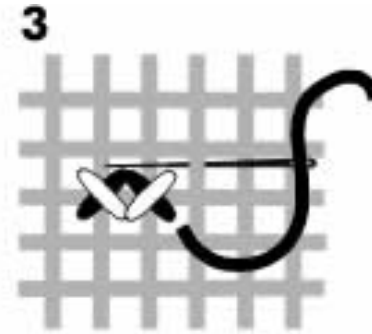
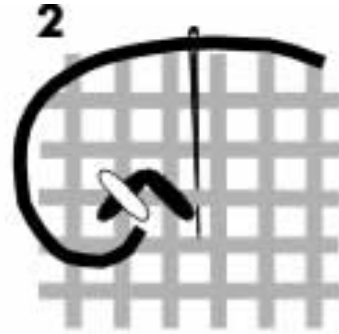
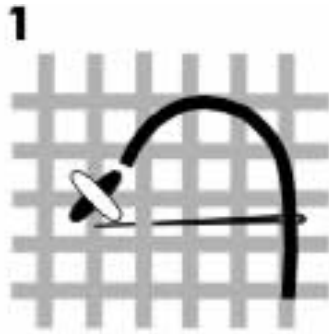


© copyright 1998 Thomas Erickson

organização



## Padrões de Bordado

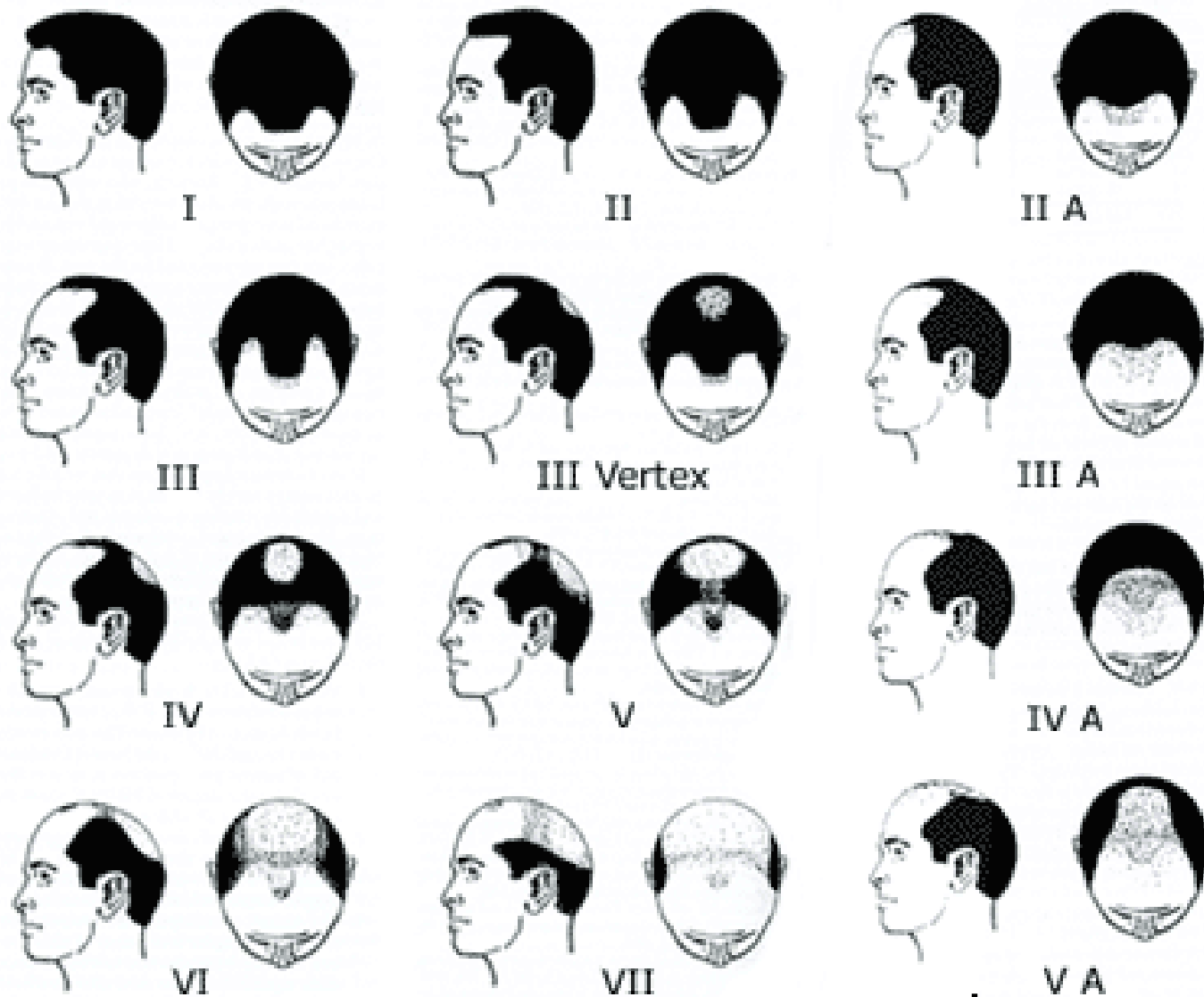


repetição





## Padrões de calvície



enquadramento

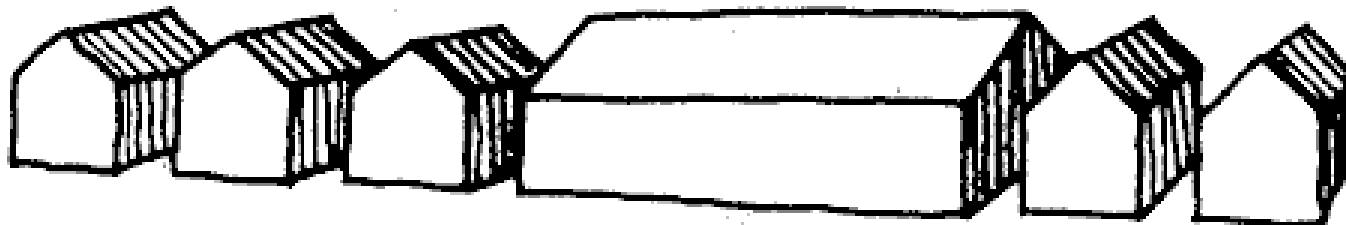




## Padrões de construção



*THIS*



*NOT THIS*



- O que são Design Patterns ?

“Um pattern descreve um problema que ocorre com frequência em nosso ambiente, e então explica a essência da solução para este problema, de forma que tal solução possa ser utilizada milhões de outras vezes...”

*Christopher Alexander (1936 arquiteto)*

- Onde encontro Design Patterns ?

- Erick Gamma. **Design Patterns: Elements of Reusable OO Software**; (Addison-Wesley - 1994);
- Martin Fowler. **Patterns of Enterprise Application Architecture**; (Addison-Wesley - 2002);



- Não existe padrão melhor ou pior, existem padrões que são mais indicados para determinadas situações;
- Na maioria das vezes, já utilizamos algum padrão, sem saber. Padrões são descobertos, não inventados;
- O conhecimento dos padrões nos leva a distinguir em quais situações utilizá-los;
- Os códigos a seguir são apenas **UMA** das formas de implementar os patterns, provavelmente não a melhor, visto que aqui o enfoque é **DIDÁTICO** :-)





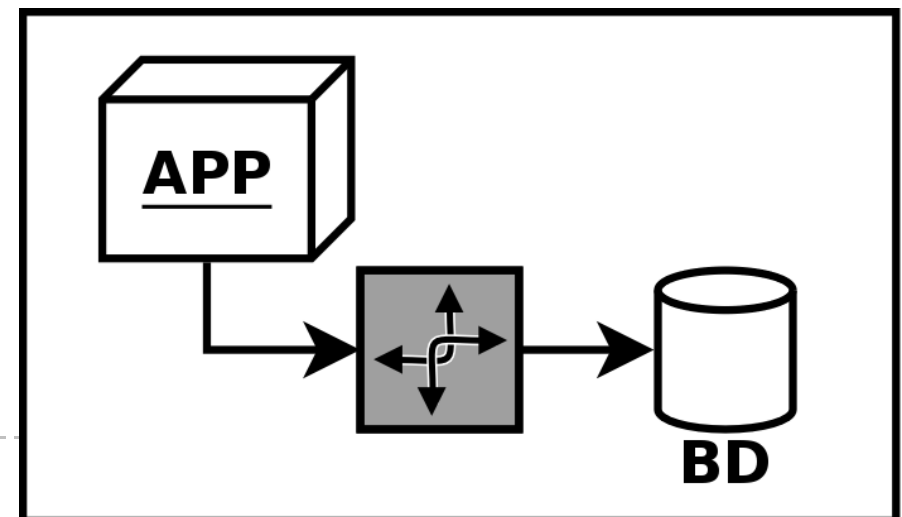
```
<?php
// conecta ao banco
$conn = pg_connect("dbname=livro user=postgres");
// string sql
$sql = "select id, nome from cidade order by id";
echo '<table border=1>'; // abre tabela
$result=pg_query($conn,$sql); // roda o sql
while ($row=pg_fetch_array($result))
{
    // exibe resultados
    echo '<tr>';
    echo "<td>{$row['id']}</td>";
    echo "<td>{$row['nome']}</td>";
    echo "</tr>";
}
// fecha tabela
echo "</table>";
pg_close($conn);
?>
```

- *detalhes de conexão explicitos;*
- *camada visual misturada com acesso ao banco de dados;*
- *SQL espalhado pelo código -fonte;*

- Persistência significa continuar a existir, perseverar, durar longo tempo ou permanecer;
- A possibilidade dos objetos existirem em um meio externo à aplicação que os criou (ex. bancos de dados);
- BDR surgiram na década 70 (padrão estruturado);
- BDR não suportam diversos conceitos OO (herança, composição, agregação, polimorfismo, métodos, etc);
- BD OO ainda tem pouca adoção (desempenho inferior, pouca documentação, ferramentas não tão maduras);
- BDR são amplamente utilizados para armazenar objetos, mas há uma dissonância corrigida pelo uso de técnicas, ou enterprise patterns;

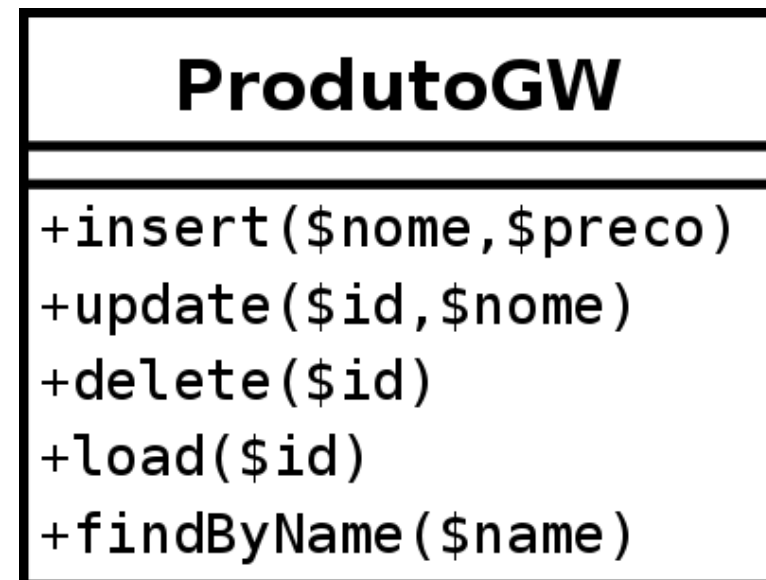


- Em algum momento, precisaremos armazenar as informações da aplicação em um banco de dados;
- Devemos evitar ter o código de manipulação de dados (SQL) espalhado pelo código-fonte da aplicação;
- Um Gateway é uma interface que se comunica com um recurso externo, escondendo seus detalhes;
- Para implementar um gateway, criamos uma classe para manipular o acesso à uma tabela do banco de dados;



- Pode ser implementada por uma classe responsável por persistir (inserir, alterar, excluir) e retornar dados do BD;
- Uma (1) classe por tabela do banco de dados. Apenas uma instância desta classe irá manipular todos os registros;
- Por isto é necessário sempre identificar o registro sobre o qual o método estará operando;

*Obs: Table Data Gateway é por natureza State Less, ou seja, não mantém o estado de suas propriedades, atua simplesmente como ponte entre o objeto de negócio e o banco de dados.*

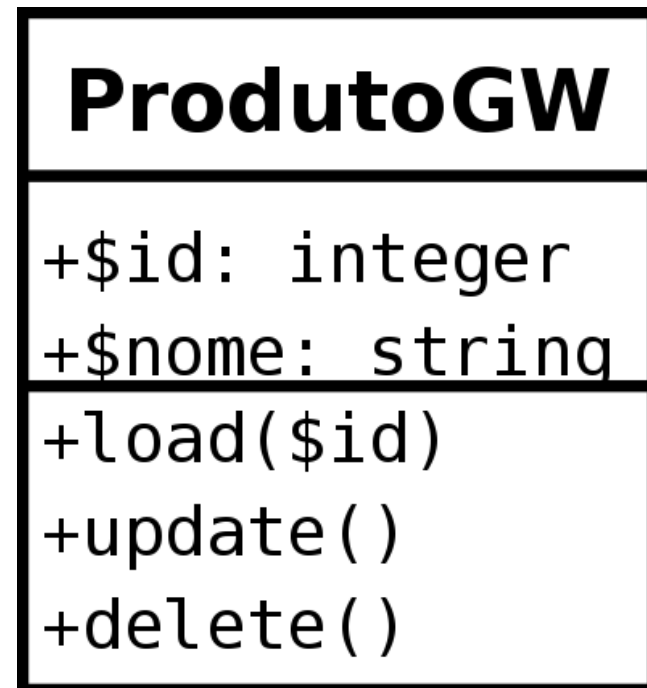


code, please...



- Uma classe para persistir um objeto no banco de dados;
- Cada instância (objeto) representa um registro diferente;
- Cada objeto se comporta exatamente como um registro, suas propriedades representem as colunas do banco de dados, além de ter métodos para armazená-lo no banco de dados;

**Obs:** Row Data Gateway é StateFull, ou seja, mantém os valores de suas propriedades ao longo do seu ciclo de vida, não sendo necessário passar todos valores novamente via parâmetros, como no caso do Table Data Gateway.



code, please...



- Row Data Gateway não possui métodos do modelo de negócios, somente métodos de acesso à base de dados;
- Quando adicionamos lógica de negócio à um Row Data Gateway, temos um Active Record;
- Com o Active Record, temos uma única camada, onde temos lógica de negócios (modelo conceitual) e métodos de persistência do objeto na base de dados (gateway);

## Produto

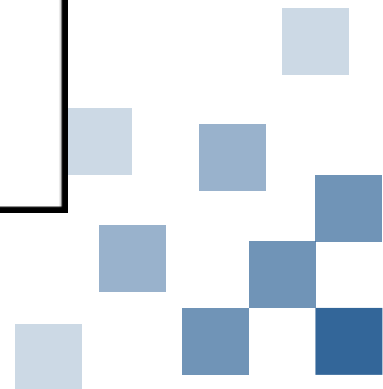
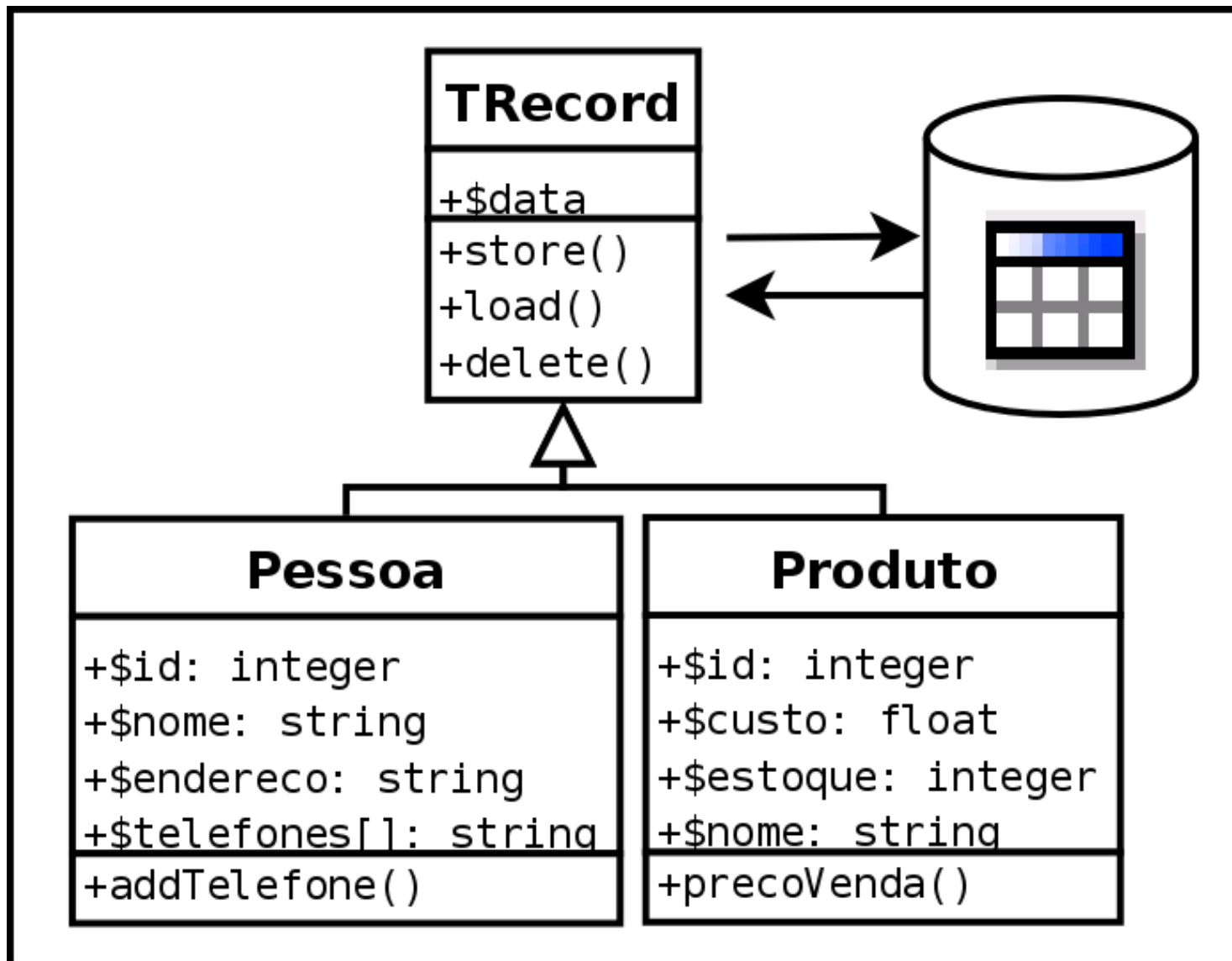
```
+$id: integer
+$nome: string
+$estoque: float
+load($id)
+update()
+delete()
+compra($qtde)
+venda($qtde)
```

code, please...



- Existem métodos que se repetem em todas as classes do sistema (load, update, delete, ...);
- Estes métodos podem ser generalizados !!
- Para tal, podemos criar uma super-classe (herdada por todos objetos de negócio) para implementar os métodos de persistência de forma genérica;
- Quando criamos uma super-classe que reúne funcionalidades em comum para toda uma camada de objetos, estamos utilizando um pattern conhecido como *Layer Supertype*;
- Funcionalidades comuns para todos objetos na super-classe;
- Métodos de NEGÓCIO ficam na classe filha;





```
<?php
// cria a classe
class Pessoa extends TRecord
{
    // métodos de negócio
}

// instancia novo objeto
$maria = new Pessoa;
$maria->nome = 'Maria da Silva';
$maria->fone = '93849384';

// armazena objeto
$maria->store()
?>
```



```
<?php
// cria a classe
include_once 'Pessoa.class.php';

// instancia novo objeto
$ pessoa = new Pessoa;
// carrega objeto do banco
$ pessoa->load(1);
// altera uma propriedade
$ pessoa->fone = '93759375';
// armazena objeto
$ pessoa->store()
// deleta o objeto
$ pessoa->delete();
?>
```

• Como objeto é um active record, possui memória;



- Sistemas complexos precisam de consultas complexas;
- Necessário escrever **diferentes** métodos para manipular objetos baseados em diferentes critérios de seleção;

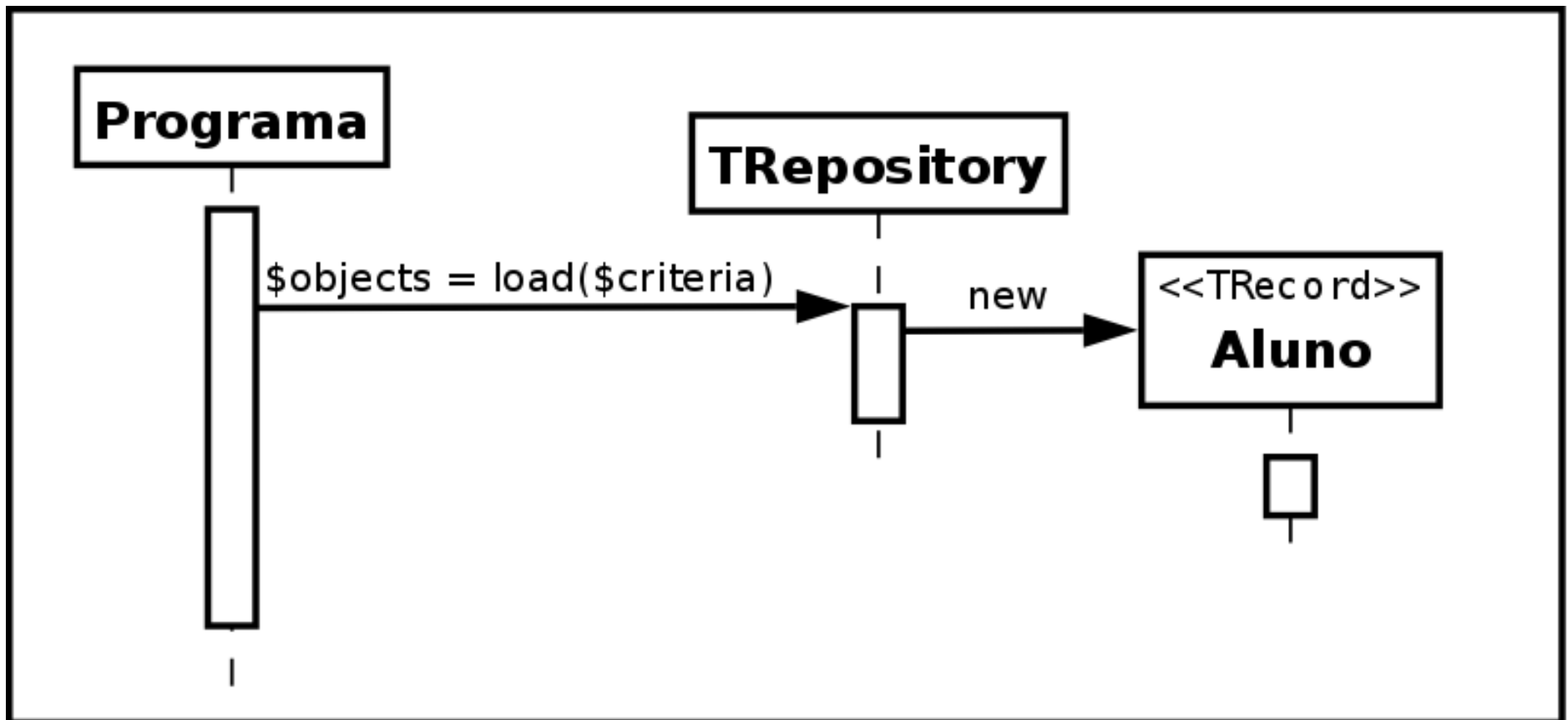
```
<?php
function listarPessoasPorNome($nome)
{ ... }

function listarPessoasPorCidade($cidade)
{ ... }
?>
```

- E se tivéssemos uma classe para manipular coleções de objetos, assim como temos o Layer Supertype (TRecord) para manipular objetos únicos ?



- Um Repository, ou um “repositório”, é uma camada na aplicação que atua como um gerenciador de coleções de objetos (carregar, contar, excluir, ...);



*Normalmente apoia-se em um objeto de definição de critérios*




```
<?php
class Turma extends TRecord
{
    // métodos de negócio
}

// instancia um repositório para Turma
$repository = new TRepository('Turma');

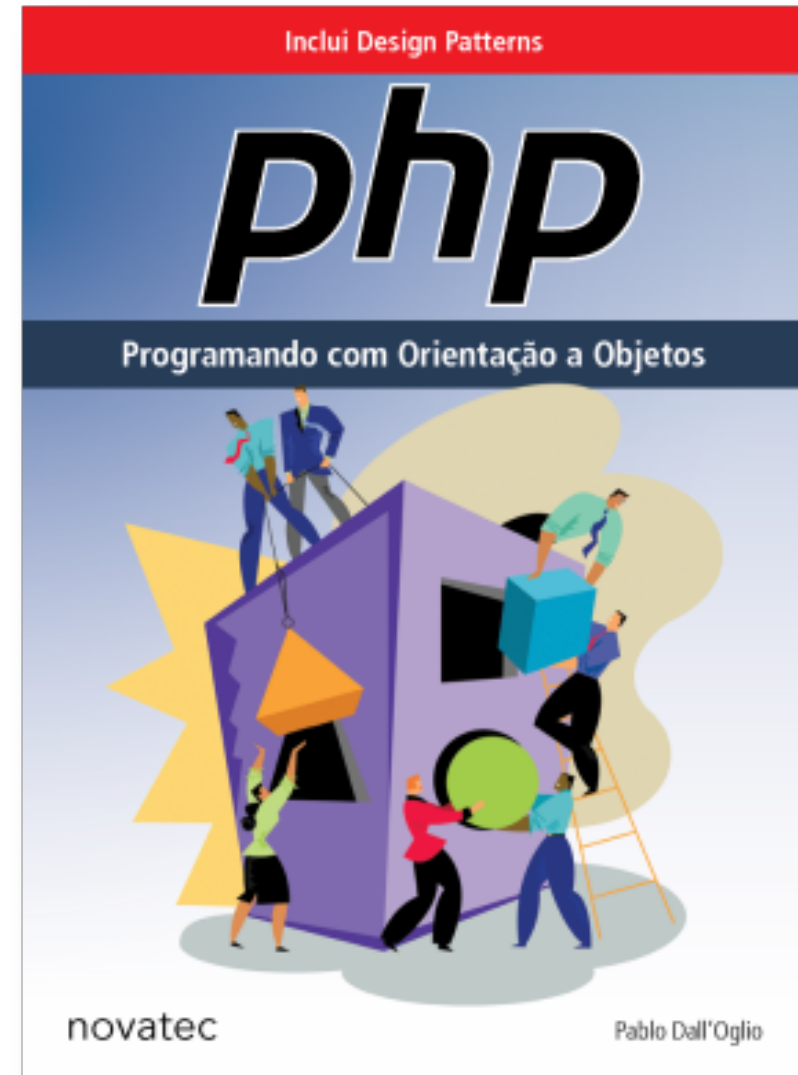
$turmas = $repository->load("turno = 'T'");

foreach ($turmas as $turma)
{
    echo ' ID      : ' . $turma->id;
    echo ' Dia    : ' . $turma->dia_semana;
}
?>
```

A grey speech bubble with a black outline and a drop shadow, containing the text "OK, but show me your code !!".

OK, but show  
me your  
code !!





# Obrigado!



## E-Mail

[pablo@php.net](mailto:pablo@php.net)

[pablo@dalloaglio.net](mailto:pablo@dalloaglio.net)

## URL

<http://www.adianti.com.br>

<http://www.pablo.blog.br>

