

Estruturação em Camadas

**“MVC” Simplificado
com Persistência em
Listas**

Desenvolvimento Sistema de Livraria

- Objetivo

➡ Desenvolver uma sistema gráfico de gestão de livraria

- Características

➡ Uso de Java e Swing

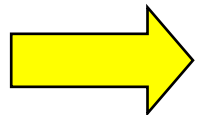
- Linguagem e pacote de desenvolvimento gráfico adotado

➡ Uso do NetBeans

- Para estruturação do projeto
- Para modelagem da interface

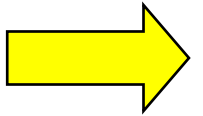
➡ Uso do modelo “MVC” simplificado

- Mas o que é isso mesmo.... ???
- Vamos primeiro conversar sobre camadas...



Arquitetura de Camadas

- ➡ A Arquitetura de Camadas é muito utilizada para:
 - Separar responsabilidades em uma aplicação moderna.
- ➡ A organização em camadas é a chave para a independência entre os componentes.
 - E esta independência é que vai atingir os objetivos de eficiência, escalabilidade , reutilização e facilidade de manutenção.
- Vamos analisar como essa estrutura evoluiu..



Arquitetura de Camadas

- No início nós temos as Aplicações monolíticas

➡ No reinado do grande porte e do computador pessoal independente

- Um aplicativo era desenvolvido para ser usado em uma única máquina .

➡ Geralmente este aplicativo continha todas a funcionalidades em um único módulo

- Gerado por uma grande quantidade de linhas de código e de manutenção nada fácil.

➡ A entrada do usuário, verificação, lógica de negócio e acesso a banco de dados:

- Estavam presente em um mesmo lugar



Arquitetura de Camadas

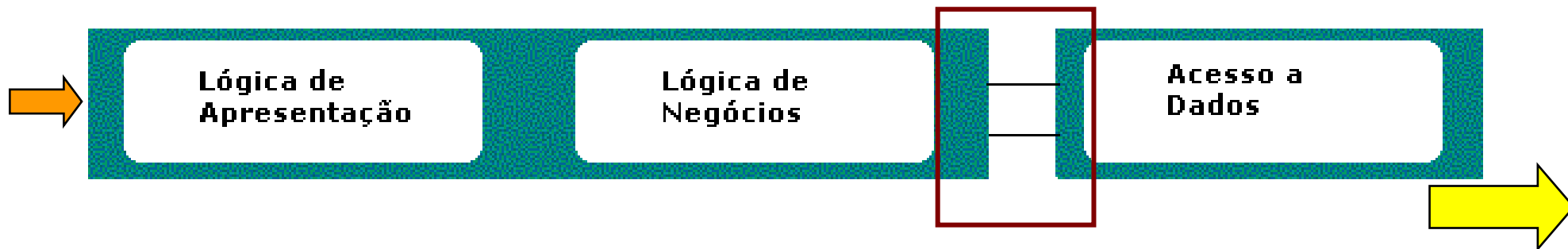
- Depois veio as aplicações em duas camadas

➡ A necessidade de compartilhar a lógica de acesso a dados entre vários usuários simultâneos

- Fez surgir as aplicações em duas camadas.

➡ Nesta estrutura a base de dados foi colocada em uma máquina específica,

- Separada das máquinas que executavam as aplicações



Arquitetura de Camadas

- E finalmente as Aplicações em três camadas

➡ Houve um movimento para separar:

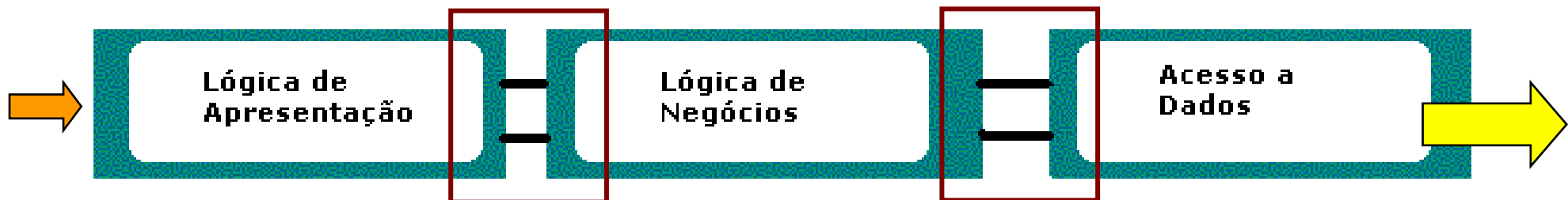
- A lógica de negócio da interface com o usuário e acesso a dados

➡ O modelo de **três camadas físicas (3-tier)** divide um aplicativo de modo que:

- A lógica de negócio reside no meio das três camadas físicas.

– Isto é chamado de camada física intermediária ou camada física de negócios.

- A maior parte do código escrito reside na camada de apresentação e de negócio.

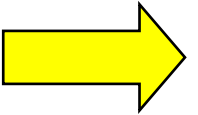


Arquitetura de Camadas

- Observação importante...

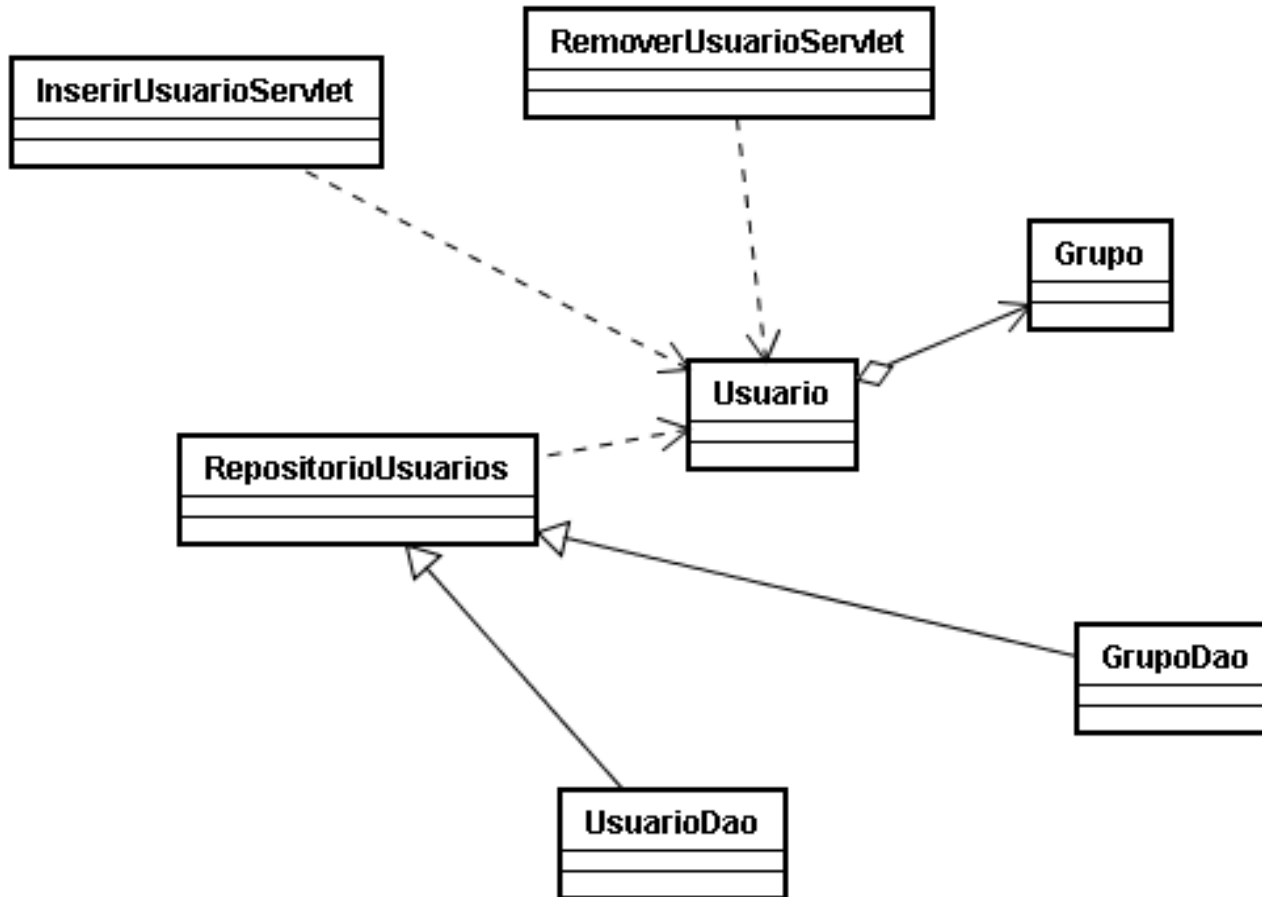
➡ Podemos ter aplicações em N-camadas, quantas forem necessárias...

➡ O importante é sempre observar o Custo x Benefício entre desempenho e flexibilidade...



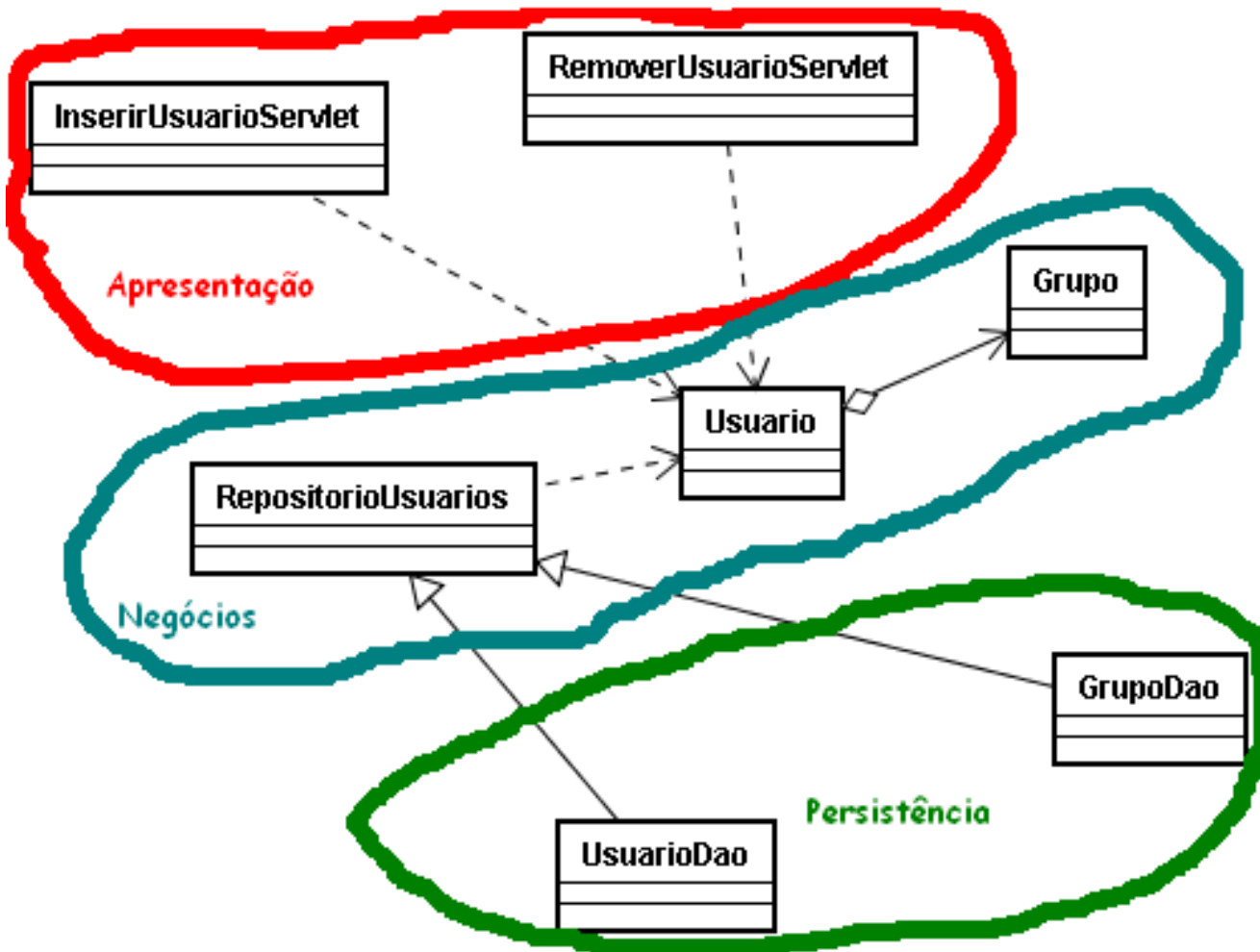
Arquitetura de Camadas

- Vamos analisar um exemplo...
 - O diagrama abaixo mostra os objetos em uma aplicação simples.
- ➡ Neste caso não há separação lógica qualquer.



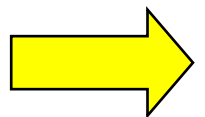
Arquitetura de Camadas

- Aplicando a técnica de Camadas,
➔ Vamos agrupar os objetos com responsabilidade semelhante.



As Camadas promovem:

- **Encapsulamento:**
Uma Camada não vai saber o que tem dentro da outra
- **E a Coesão:**
Componentes parecidos ficam no mesmo grupo



O modelo MVC simplificado

➡ Partindo do ponto que temos os nossos componentes separados utilizando Camadas

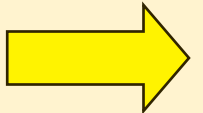
➡ Precisamos de um modelo de interação entre elas.

➡ O Modelo MVC, define que nossos componentes são divididos em 3, os View, Model e Controller.

➡ A View é a parte exposta,

➡ O Controller é o controle sobre a comunicação que vem do usuário para o sistema e

➡ O Model representa o estado do sistema.



O modelo MVC simplificado

➔ Uma vez que temos uma divisão em camada definidas, podemos aplicar o modelo MVC da seguinte forma....:

➔ A Camada de Negócios como o Model,

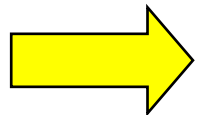
- Modela os dados e o comportamento por trás do processo de negócios
- Se preocupa apenas com o armazenamento, manipulação e geração de dados

➔ A Apresentação como a View.

- Inclui os elementos de exibição.
- É a camada de interface com o usuário.
- É usada para receber a entrada de dados e apresentar o resultado

➔ O componente Controller para determina o fluxo da apresentação

- Serve como uma camada intermediária entre a camada de apresentação e a lógica.
- Controla e mapeia as ações



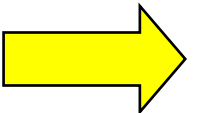
O modelo MVC simplificado

- Solução Com “MVC” Comunicando Diretamente entre as Camadas (variante mais simples):

⇒ Uma variação mais simples do MVC é realizando as chamadas diretamente entre as camadas, tanto na “ida para a camada de modelo como na volta para a camada de visão”

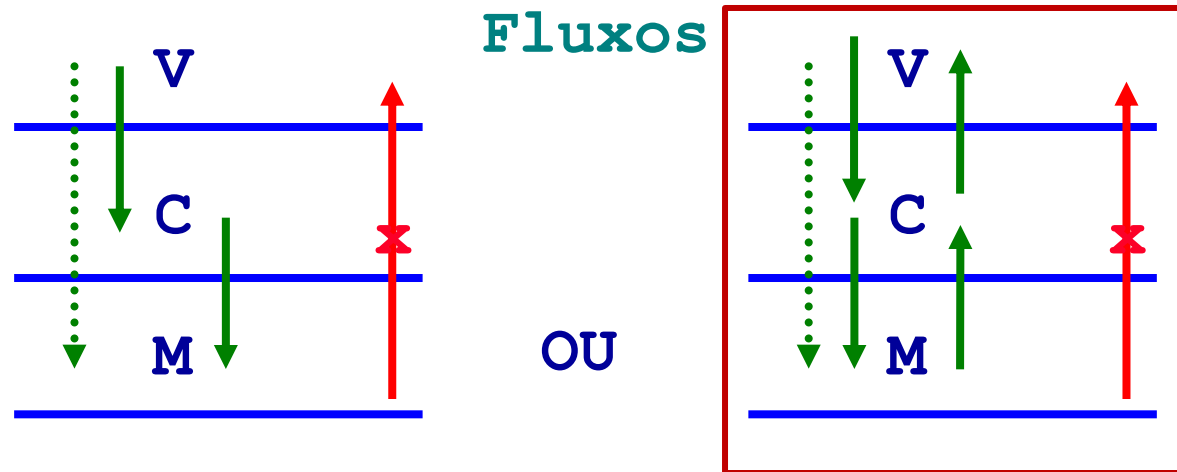
⇒ Exemplo:

- ⇒ O controlador, chama a lógica de negócios (modelo), recebe resultados e os repassa para a view apropriada
- ⇒ É papel do controlador escolher a view mais apropriada

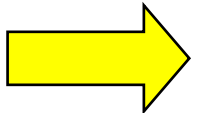


O modelo MVC simplificado

⇒ Resumindo as representações temos....



⇒ O mais importante é sempre manter o modelo desacoplado da visão...



O modelo MVC simplificado

- **Resumo:**

- **Camadas**

- Dizem como **agrupar** os componentes.

- **MVC**

- Diz como **interagem** os componentes.

- **Componentes do modelo MVC**

- **Modelo**

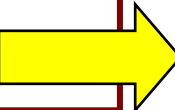
- Classes de elementos que **são os “atores básicos”** do sistema. A lógica de negócio do sistema. Também faz a persistência

- **Visão**

- Classes que **fazem o front-end** com o usuário

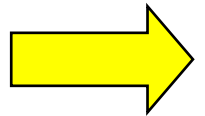
- **Controle**

- Classes que fazem **a ponte entre a camada de visão e a camada dos modelos** utilizados no sistema



Desenvolvimento Sistemas - Exercício

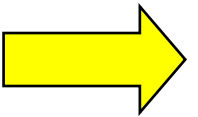
- Blz,
 - ⇒ Chega de conversa, vamos trabalhar....
 - ⇒ O modelo pode ser alterado para trabalhar com Coleções ou outras facilidades da API...



Desenvolvimento Sistemas - Exercício

- Soluções com MVC

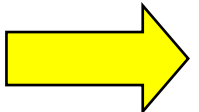
- ➡ Vamos fazer um exercício para construir um sistema que utiliza o “MVC” Comunicando Diretamente entre as Camadas
 - Solução mais simples



Desenvolvimento Sistemas - Exercício


⇒ Enunciado do Problema....:

- ⇒ **Codificar o Sistema Livraria**, utilizando o “MVC”
Comunicando Diretamente entre as Camadas...
- ⇒ Seguindo as indicações apresentadas nos próximos slides e alterando o que achar necessário...
- ⇒ Uma sugestão de Interface do sistema é apresentada a seguir....



Desenvolvimento Sistema de Livraria

→ Interface com o
usuário do sistema
a ser desenvolvido

 Sistema de Controle de Livraria - SCL

Dados Livraria


Razão Social

CNPJ

Rua

CEP Número Cidade

Bairro Estado

 **Cadastrar Livraria**

Dados Livro





Título

ISBN Num. Pag.


Editora

Categoria

Nome Autor Idade Sexo

 **Cadastrar**  **Remover**  **Consultar**  **Alterar**

Listagem de Livros

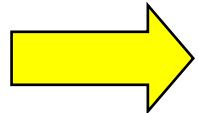
 **Listar**

→

Desenvolvimento Sistema de Livraria

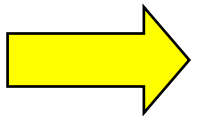
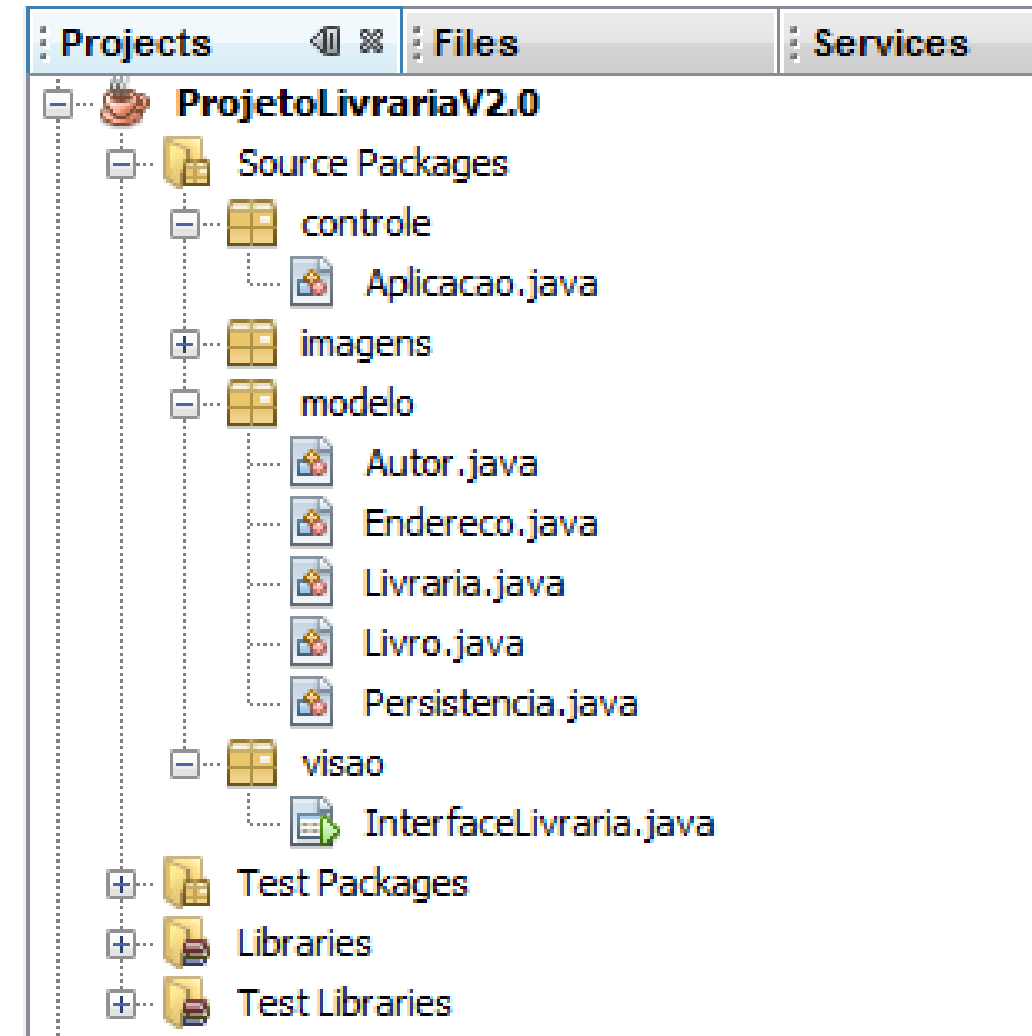
Desenhar as camadas
e a ideia do código
pra comunicação...

- Vamos fazer os seguintes passos:
 - 1. Construir a estrutura de pacotes e as classes relevantes para o nosso sistema
 - 2. Construir a interface e dados no modelo inicialmente somente com os seguintes elementos:
 - Razão Social, CNPJ e botão Cadastrar da Livraria
 - Título , ISBN e Autor do Livro (Nome e Idade do Autor) e botão Cadastrar Livro
 - 3. Cadastrar um livro utilizando os campos definidos anteriormente para a Livro



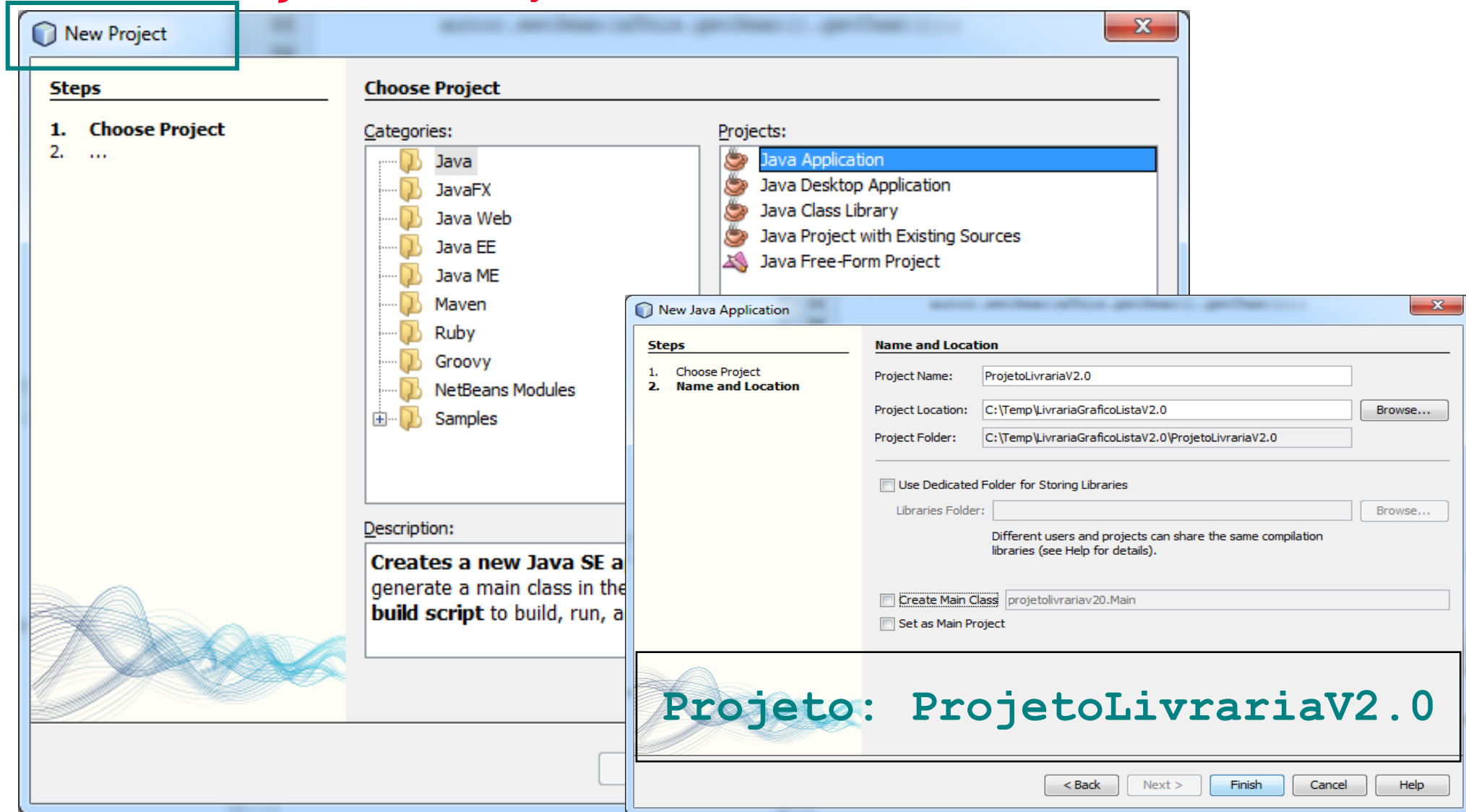
Desenvolvimento Sistema de Livraria

→ Estruturação no padrão MVC



Desenvolvimento Sistema de Livraria

➔ Construção do Projeto



Desenvolvimento Sistema de Livraria

➡ Construção de Pacotes

– Modelo / Visao / Controle

New Java Package

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Package Name:

Project:

Location:

Created Folder:

Pacotes: modelo / visao / controle

Provide valid Java Package name

< Back Next > Finish Cancel Help

Desenvolvimento Sistema de Livraria

➔ Construção do Arquivo da Interface

New JFrame Form

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name: NewJFrame

Project: ProjetoLivrariaV2.0

Location: Source Packages

Package: visao

Created File: C:\Temp\LivrariaGraficoListaV2.0\ProjetoLivrariaV2.0\src\visao\NewJFrame.java

InterfaceLivraria

Pacote: visao

< Back Next > Finish Cancel Help

Desenvolvimento Sistema de Livraria

➔ Construção do Arquivo de Controle

New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name: **Aplicacao**

Project:

Location:

Package: **Pacote: controle**

Created File:

< Back Next > **Finish** Cancel Help

Desenvolvimento Sistema de Livraria

➔ Construção do Arquivo de Modelo

New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:

Project:

Location:

Package:

Created File:

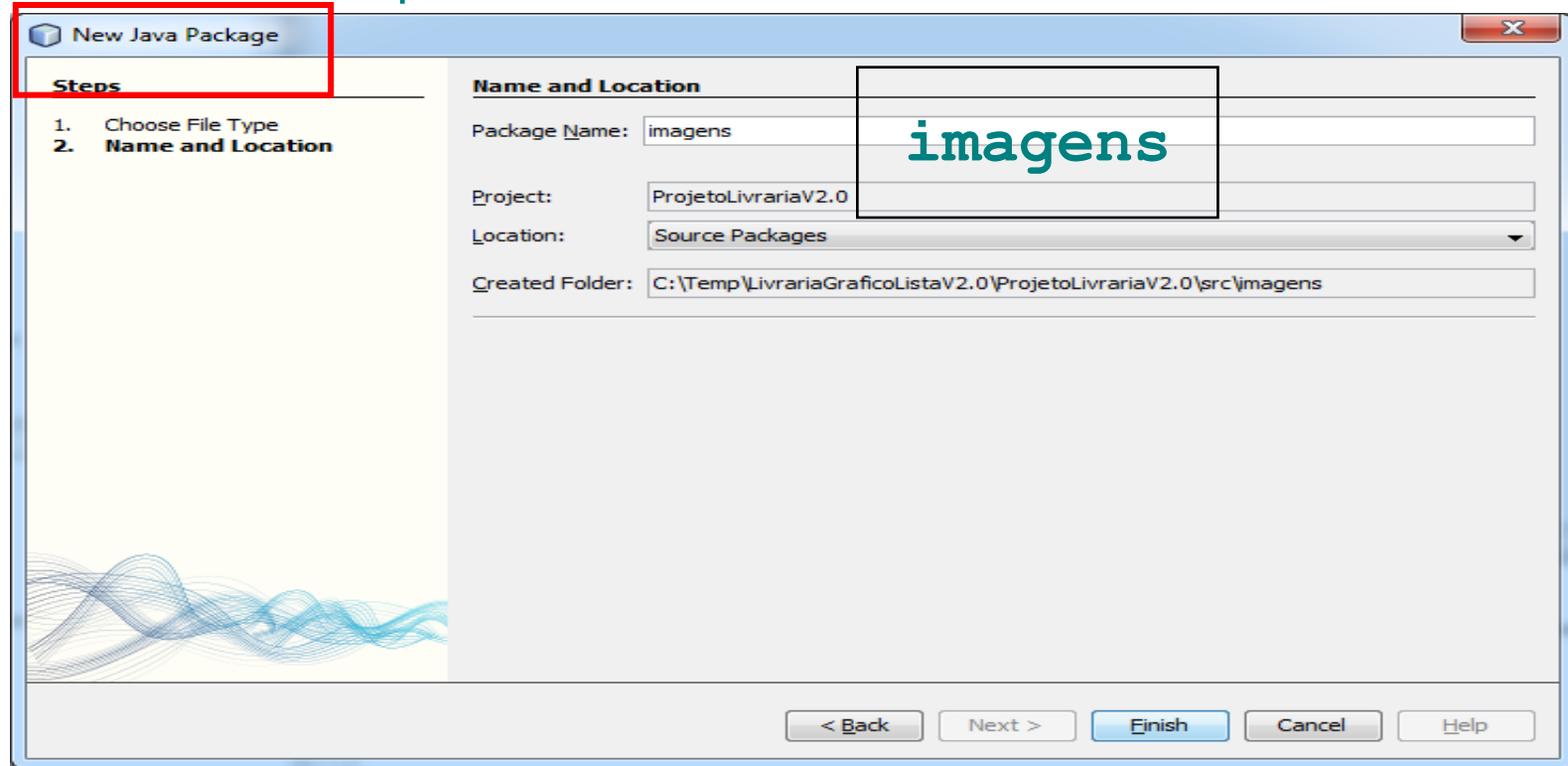
Pacote: modelo

Livraria / Autor / Endereco / Livro

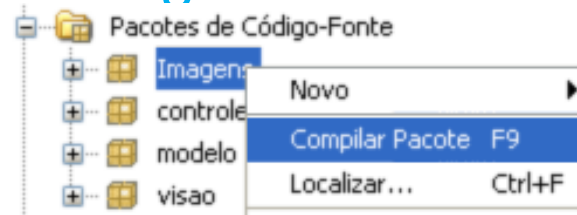
< Back Next > Finish Cancel Help

Desenvolvimento Sistema de Livraria

- ➔ Construção da Pasta de Imagens
 - Construa um novo pacote



- ➔ Copie os arquivos para a pasta **imagem** dentro de src, do projeto em questão
- ➔ Depois compile o pacote



Desenvolvimento Sistema de Livraria

- Interface com o usuário inicial do sistema com as funcionalidades que definimos inicialmente...
 - Razão Social, CNPJ e botão Cadastrar da Livraria
 - Título, ISBN e Autor do Livro (Nome e Idade do Autor) e botão Cadastrar Livro

Source Design

Use the Connection Mode button (in the toolbar) to establish a co

Pacote: visao

JPanel/Border/Titled Border/ Dados Livraria

JLabel/Text/ Razao Social: JTextField/Text/ Limpe o campo e escreva o dado

JLabel/Text/ CNPJ: JTextField/Text/ Limpe o campo e escreva o dado

JButton/Text/ Cadastrar Livraria

JPanel/Border/Titled Border/ Dados Livro

JLabel/Text/ Título: JTextField/Text/ Limpe o campo e escreva o dado

JLabel/Text/ ISBN: JTextField/Text/ Limpe o campo e escreva o dado

JLabel/Text/ Nome do Autor: JTextField/Text/ Limpe o campo JLabel/Text/ Idade: JTextFi

JButton/Text/ Cadastrar Livro

JButton1 [JButton] - icon

Set JButton1's icon property using: Image chooser

Image Within Project

Package: imagens

File: Save16.gif

External Image

File or URL:

Import to Project...

No Image

OK Reset to Default Cancel

Lembre-se de colocar nomes significativos para os campos

Desenvolvimento Sistema de Livraria

- Interface com o usuário inicial do sistema com as funcionalidades que definimos inicialmente...
 - Razão Social, CNPJ e botão Cadastrar da Livraria
 - Título , ISBN e Autor do Livro (Nome e Idade do Autor) e botão Cadastrar Livro

```
private javax.swing.JTextField cnpjUI;  
private javax.swing.JTextField idadeAutorUI;  
private javax.swing.JTextField isbnUI;  
private javax.swing.JButton jButton1;  
private javax.swing.JButton jButton2;  
private javax.swing.JLabel jLabel1;  
private javax.swing.JLabel jLabel2;  
private javax.swing.JLabel jLabel3;  
private javax.swing.JLabel jLabel4;  
private javax.swing.JLabel jLabel5;  
private javax.swing.JLabel jLabel6;  
private javax.swing.JPanel jPanel1;  
private javax.swing.JPanel jPanel2;  
private javax.swing.JTextField nomeAutorUI;  
private javax.swing.JTextField razaoSocialUI;  
private javax.swing.JTextField tituloUI;
```

Members View

InterfaceLivraria :: JFrame

- InterfaceLivraria()
- getCnpjUI() : JTextField
- getIdadeAutorUI() : JTextField
- getIsbnUI() : JTextField
- getNomeAutorUI() : JTextField
- getRazaoSocialUI() : JTextField
- getTituloUI() : JTextField

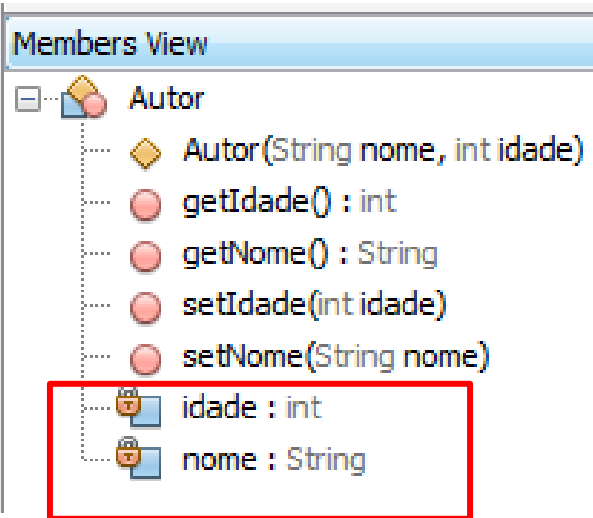
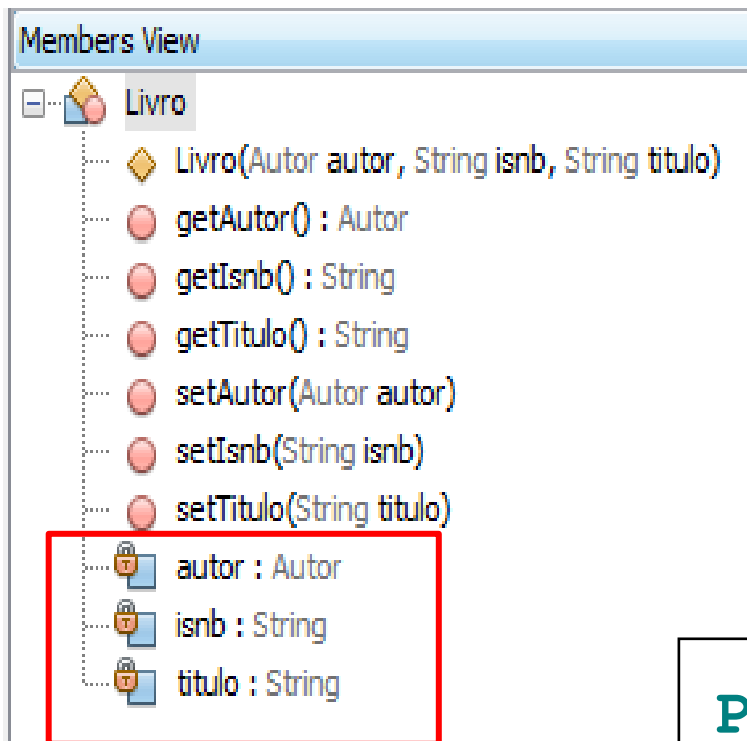
- setCnpjUI(JTextField cnpjUI)
- setIdadeAutorUI(JTextField idadeAutorUI)
- setIsbnUI(JTextField isbnUI)
- setNomeAutorUI(JTextField nomeAutorUI)
- setRazaoSocialUI(JTextField razaoSocialUI)
- setTituloUI(JTextField tituloUI)

Pacote: visao

Lembre-se de colocar
nomes significativos
para os campos e os
métodos get/set

Desenvolvimento Sistema de Livraria

- Propriedades dos objetos de negócio no modelo, de acordo com o que definimos inicialmente..
 - Título , ISBN a Autor do Livro (na classe Livro), Nome e Idade do Autor (na classe Autor)

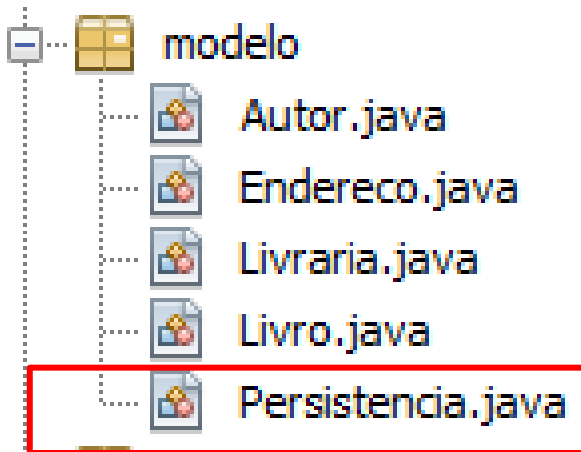


Pacote: modelo

Lembre-se de colocar
o **constutor padrão**
também...

Desenvolvimento Sistema de Livraria

- Feito isso, vamos criar uma classe que irá simular a persistência dos dados
 - Dentro desta classe terá apenas Coleções (LinkedList, por exemplo) para persistir Livros, por exemplo.



Members View

Persistencia

Pacote: modelo

addLivroPersistencia(Livro l)

getLivroPesistencia() : LinkedList<Livro>

vetLivro : LinkedList<Livro>

```
private static LinkedList<Livro> vetLivro = new LinkedList<Livro>();
```

```
public static void addLivroPersistencia(Livro l){
```

```
    vetLivro.add(l);
```

```
}
```

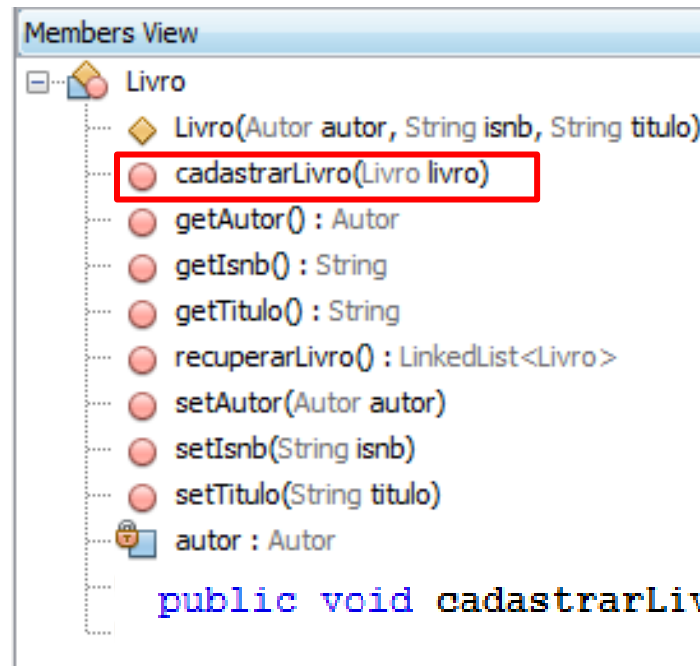
```
public static LinkedList<Livro> getLivroPesistencia(){
```

```
    return vetLivro;
```

```
}
```

Desenvolvimento Sistema de Livraria

- Agora podemos criar os nossos métodos de persistência, consulta e alteração (CRUD)
 - Neste momento estes métodos deverão ser implementados em cada objeto de negócio respectivo, no pacote de modelo.
 - De acordo com o que definimos antes, vamos criar somente os métodos de cadastro para Livro

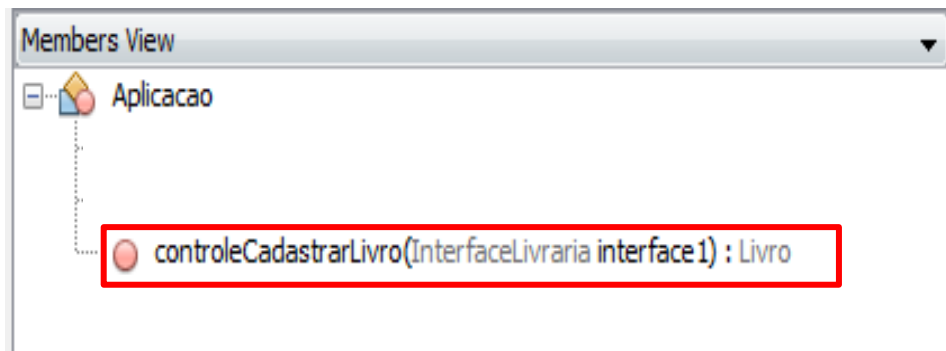


Pacote: modelo

```
public void cadastrarLivro(Livro livro) {  
  
    Persistencia.addLivroPersistencia(livro);  
}
```

Desenvolvimento Sistema de Livraria

- Tendo criado os objetos de negócios e os métodos de persistência, podemos agora trabalhar na camada de controle.
 - Esta camada repassa as requisições da visão para o modelo e acessa a persistência
 - Em uma modelagem mais simplificado, pode centralizar as regras de negócios de vários objetos de negócios do modelo
 - Ela também pode fazer verificações de integridade, pode exemplo, verificar se um livro já existe antes de incluí-lo



Pacote: controle

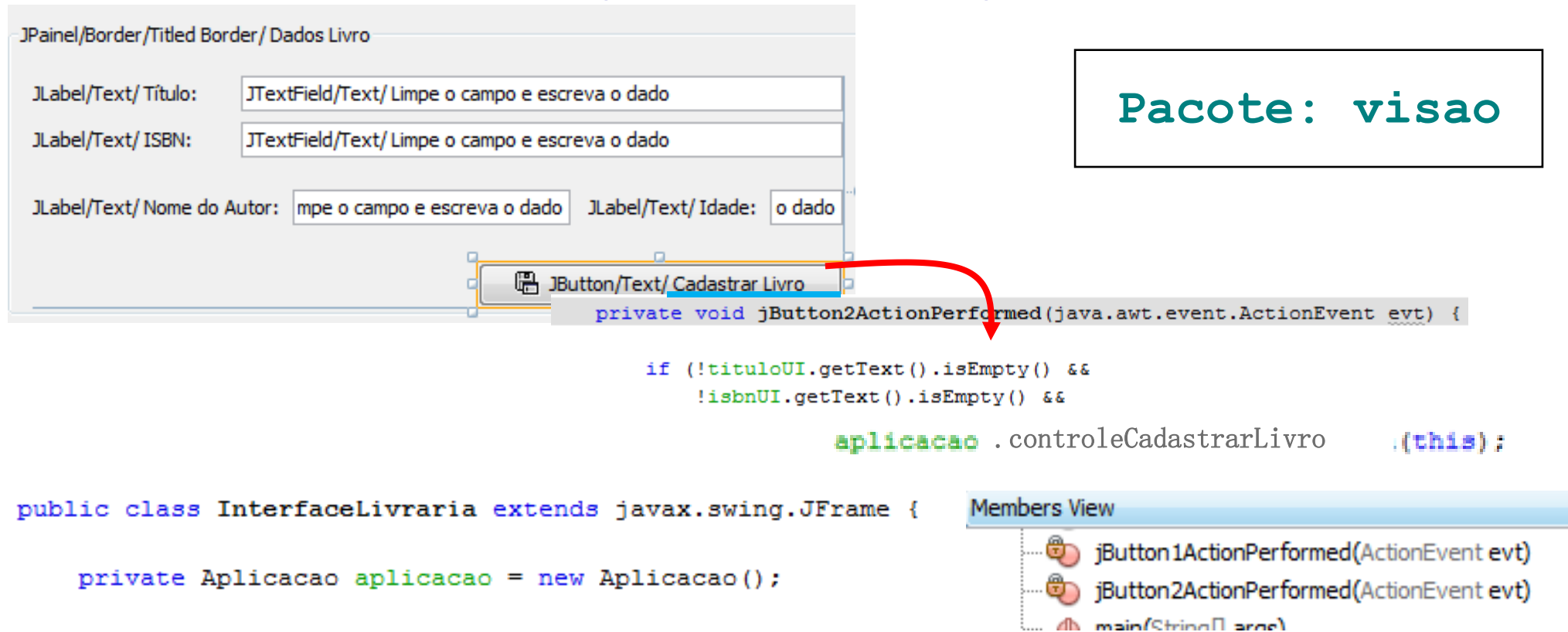
```
public Livro controleCadastrarLivro(InterfaceLivraria interface1) {
```

... new Autor() e seta os seus dados....

```
Livro livro = new Livro();  
livro.setAutor(autor);  
livro.setIsbn(interface1.getIsbnUI().getText());  
livro.setTitulo(interface1.getTituloUI().getText());  
  
livro.cadastrarLivro(livro);  
  
return livro;  
}
```


Desenvolvimento Sistema de Livraria

- Feito tudo isso, agora finalmente podemos ligar a nossa interface gráfica com o resto das camadas.
 - Para tanto, precisamos associar o evento do botão em questão a chamada ao método do controle correspondente
 - Neste camada também podemos fazer controle de erro, por exemplo, verificando se o usuário digitou os campos obrigatórios para cadastro



Pacote: visao

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
  
    if (!tituloUI.getText().isEmpty() &&  
        !isbnUI.getText().isEmpty() &&  
        !nomeUI.getText().isEmpty()) {  
        aplicacao.controleCadastrarLivro(this);  
    }  
}
```

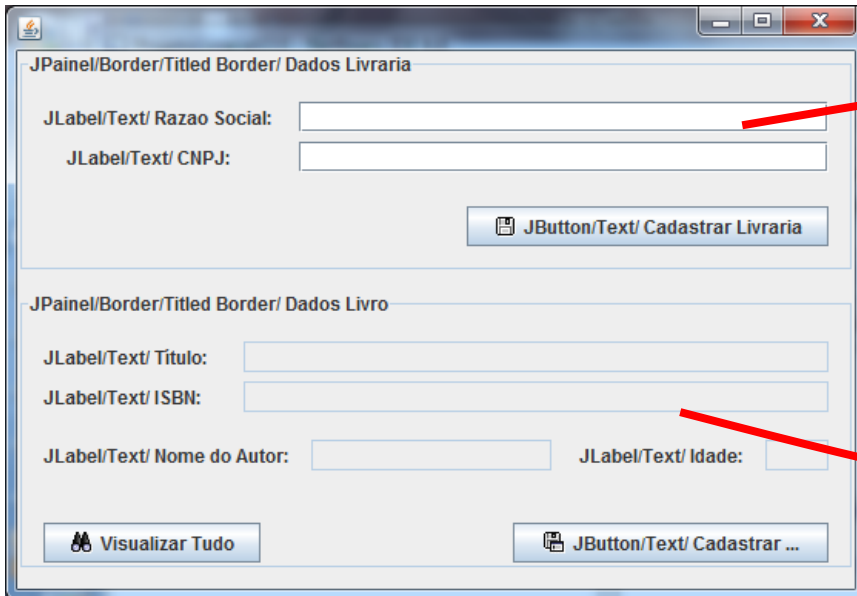
```
public class InterfaceLivraria extends javax.swing.JFrame {  
  
    private Aplicacao aplicacao = new Aplicacao();  
  
}
```

Members View

- jButton1ActionPerformed(ActionEvent evt)
- jButton2ActionPerformed(ActionEvent evt)
- main(String[] args)

Desenvolvimento Sistema de Livraria

- Requisitos importantes para a Interface
 - Neste exemplo teremos somente o cadastro de uma Livraria
 - E Livros só podem ser cadastrados após o cadastro de uma Livraria
- Logo a nossa interface vai ter o seguinte comportamento
 - Os campos de edição de Livro deverão ficar desabilitados até que tenha ocorrido o cadastro de uma Livraria
 - Após esse cadastro, o botão de cadastro de Livraria fica desabilitado e o resto habilitado.



JPainel/Border/Titled Border/ Dados Livraria

JLabel/Text/ Razao Social:

JLabel/Text/ CNPJ:

JButton/Text/ Cadastrar Livraria

JPainel/Border/Titled Border/ Dados Livro

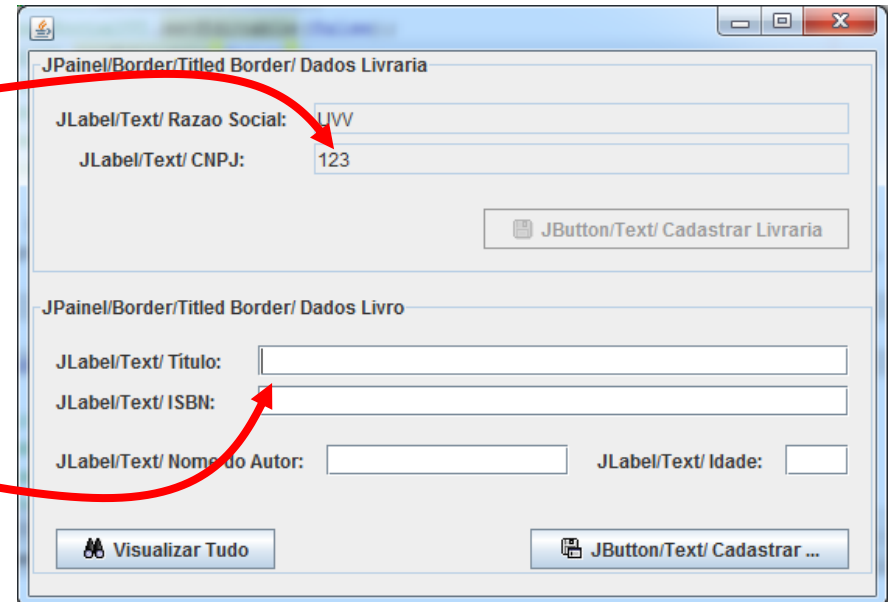
JLabel/Text/ Titulo:

JLabel/Text/ ISBN:

JLabel/Text/ Nome do Autor: JLabel/Text/ Idade:

Visualizar Tudo

JButton/Text/ Cadastrar ...



JPainel/Border/Titled Border/ Dados Livraria

JLabel/Text/ Razao Social: LUV

JLabel/Text/ CNPJ: 123

JButton/Text/ Cadastrar Livraria

JPainel/Border/Titled Border/ Dados Livro

JLabel/Text/ Titulo:

JLabel/Text/ ISBN:

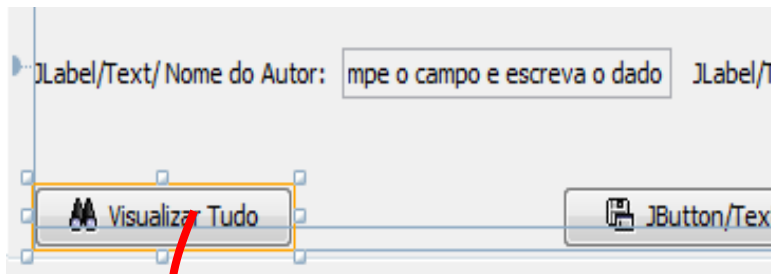
JLabel/Text/ Nome do Autor: JLabel/Text/ Idade:

Visualizar Tudo

JButton/Text/ Cadastrar ...

Desenvolvimento Sistema de Livraria

- Só para efeito de rapidamente visualizarmos se os dados estão inserido corretamente no sistema, vamos criar um novo botão na Interface e mandar imprimir diretamente as coleções que se encontram na classe Persistência



```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {  
    // Código de teste.. apenas para verificar se o livro foi incluido  
  
    LinkedList<Livro> listaLivros = Persistencia.getLivroPesistencia();
```

```
public String toString() {  
    return " " + titulo + " " + autor + " " + isbn;  
}
```

```
System.out.println("Lista de Livros Cadastrados.....");  
for (int i = 0; i<listaLivros.size(); i++) {  
    System.out.println(listaLivros.get(i));  
}
```

Desenvolvimento Sistema de Livraria

- A forma correta de consultar utilizando o padrão “MVC” deve passar pelas camadas.. Por exemplo..

```
private void jButtonConsultarLivroPeloTituloActionPerformed(java.awt.event.ActionEvent evt)
```

```
Livro livro;  
Autor autor;
```

```
if (!tituloUI.getText().isEmpty())  
{  
    livro = aplicacao.controleRecuperaLivroByTitulo(tituloUI.getText());
```

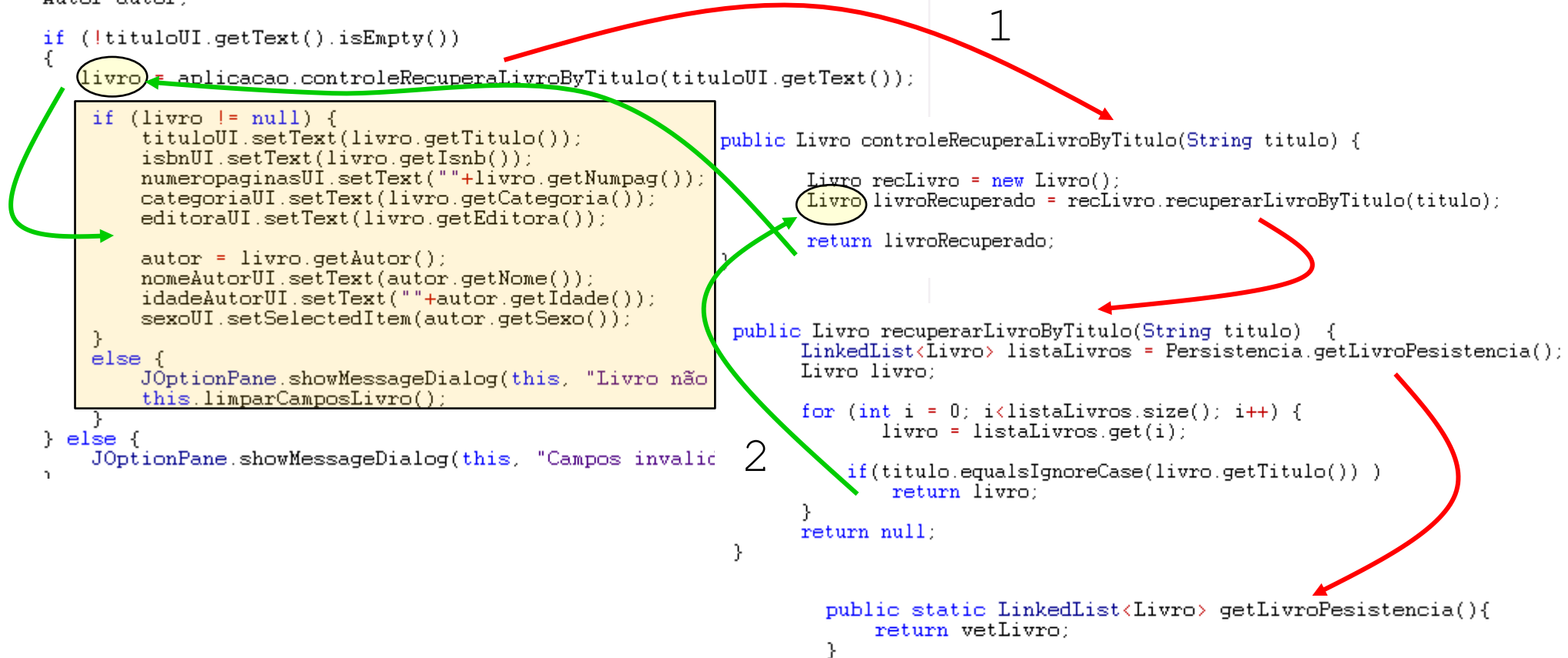
```
    if (livro != null) {  
        tituloUI.setText(livro.getTitulo());  
        isbnUI.setText(livro.getIsbn());  
        numeropaginasUI.setText(""+livro.getNumpag());  
        categoriaUI.setText(livro.getCategoria());  
        editoraUI.setText(livro.getEditora());  
  
        autor = livro.getAutor();  
        nomeAutorUI.setText(autor.getNome());  
        idadeAutorUI.setText(""+autor.getIdade());  
        sexoUI.setSelectedItem(autor.getSexo());  
    }  
    else {  
        JOptionPane.showMessageDialog(this, "Livro não  
        this.limparCamposLivro();  
    }  
}
```

```
} else {  
    JOptionPane.showMessageDialog(this, "Campos inválidos");  
}
```

```
public Livro controleRecuperaLivroByTitulo(String titulo) {  
    Livro recLivro = new Livro();  
    Livro livroRecuperado = recLivro.recuperarLivroByTitulo(titulo);  
    return livroRecuperado;  
}
```

```
public Livro recuperarLivroByTitulo(String titulo) {  
    LinkedList<Livro> listaLivros = Persistencia.getLivroPesistencia();  
    Livro livro;  
  
    for (int i = 0; i < listaLivros.size(); i++) {  
        livro = listaLivros.get(i);  
  
        if (titulo.equalsIgnoreCase(livro.getTitulo()) )  
            return livro;  
    }  
    return null;  
}
```

```
public static LinkedList<Livro> getLivroPesistencia(){  
    return vetLivro;  
}
```



Desenvolvimento Sistema de Livraria

- Agora é com vocês... 😊 Com base no que aprendemos até agora (seguindo o padrão “MVC” comunicando entre as camadas):
 - Incremente o sistema com os campos que faltam...
 - Implemente os métodos de Consulta, Alteração e Remoção para os objetos de negócios Livro e Autor (modelo).
 - Use boas práticas de programação, por exemplo, verificando possíveis problemas, etc...

LivrariaGraficoListaV3.0

Dados Livraria

Razao Social:

CNPJ:

Rua:

CEP: Número: Cidade:

Bairro: Estado:

Cadastrar Livraria

Dados Livro

Título:

ISBN: Número de Páginas:

Editora:

Categoria:

Nome do Autor: Idade: Sexo:

Cadastrar Remover Consultar Alterar Limpar Campos

Listagem dos Livros e Autores

Visualizar Tudo