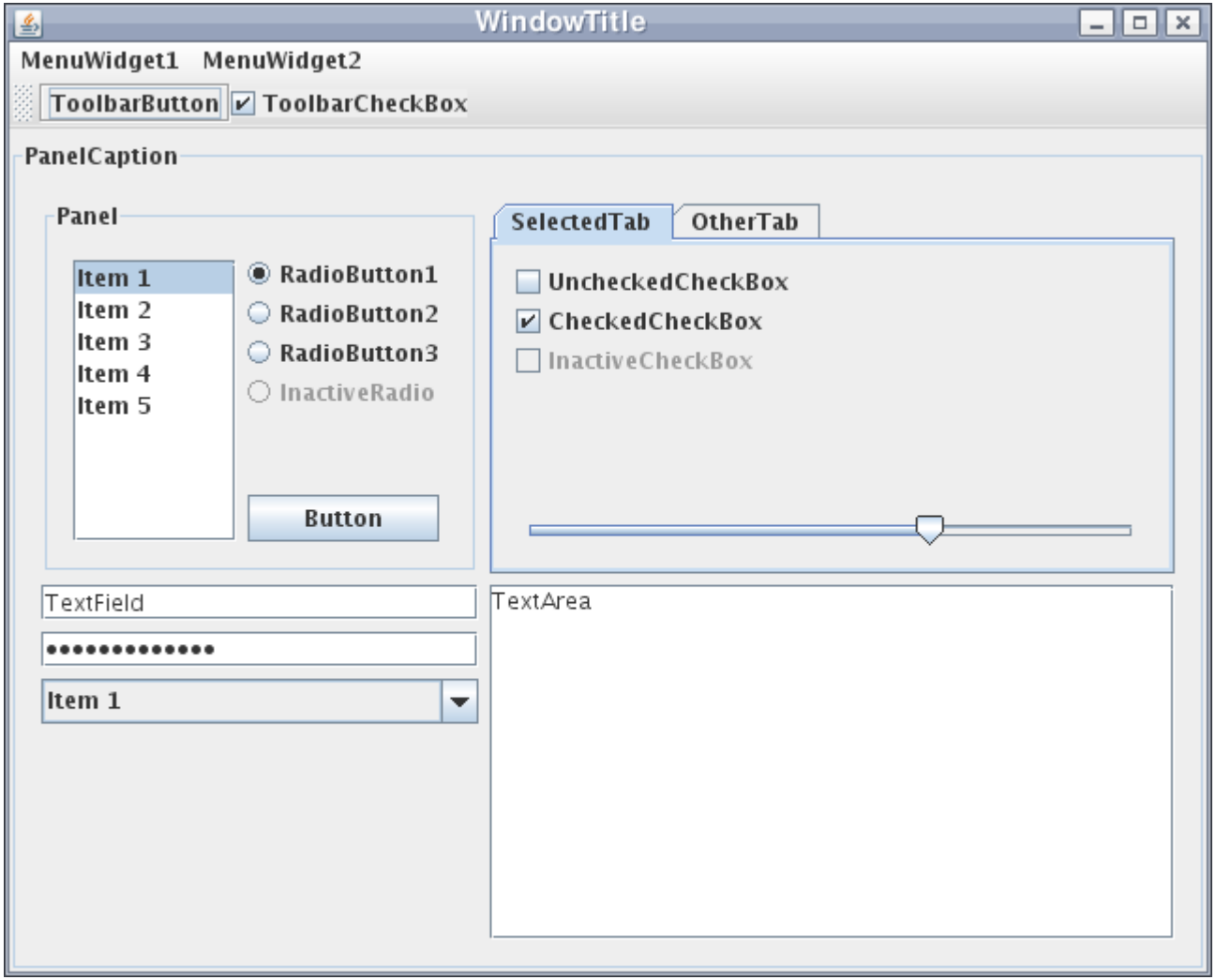
Interfaces gráficas em Java

Profa Andréa Schwertner Charão

DELC/CT/UFSM

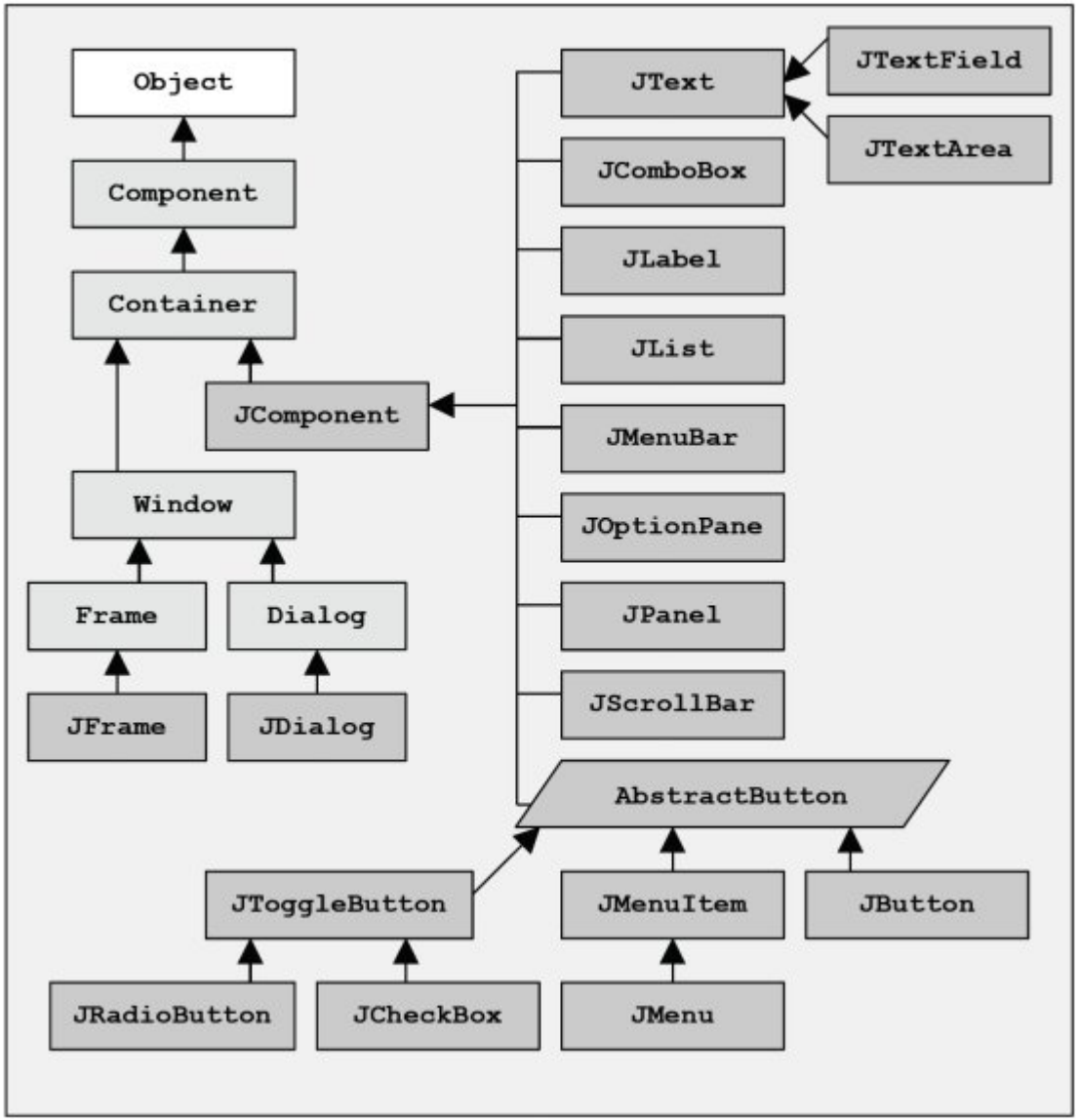
Graphical User Interfaces (GUIs)



Graphical User Interfaces (GUIs)

- Java: interfaces gráficas **portáveis**
- Desktop (ou Web via Java Web Start)
- Pacotes de classes da plataforma Java:
  - **java.awt.***: elementos básicos
  - **javax.swing.***: componentes de alto nível

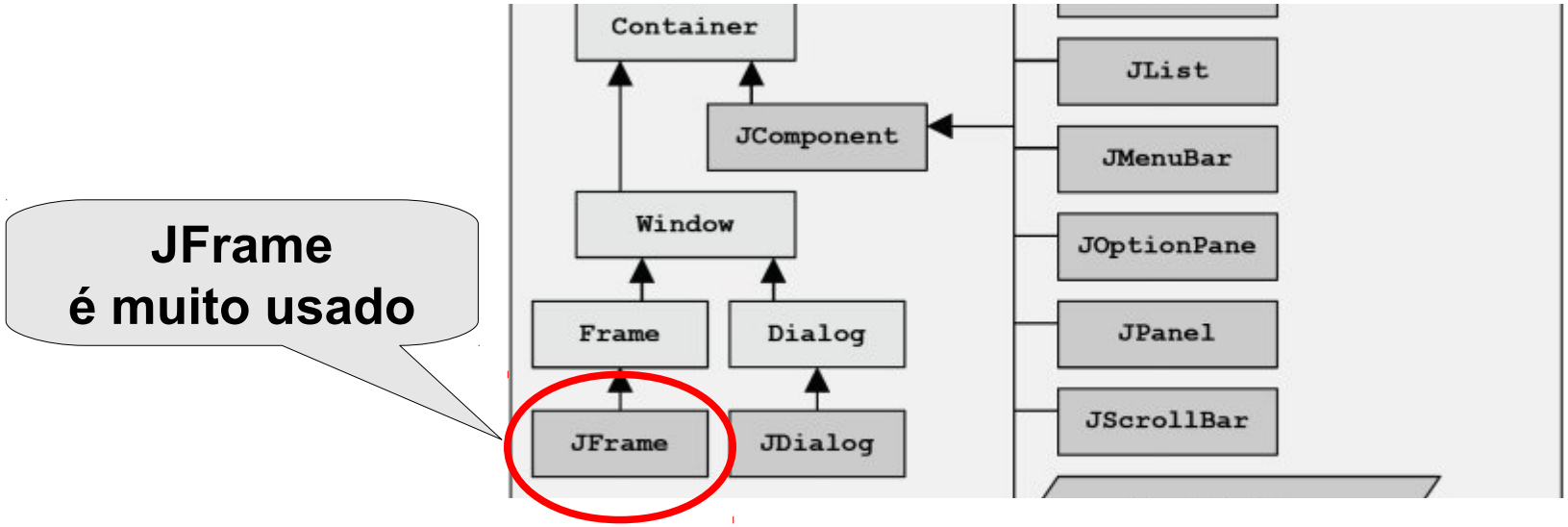
Hierarquia de classes



Fonte:
http://www.particle.kth.se/~lindsey/JavaCourse/Book/Part1/Java/Chapter06/swing.html

Categorias de classes

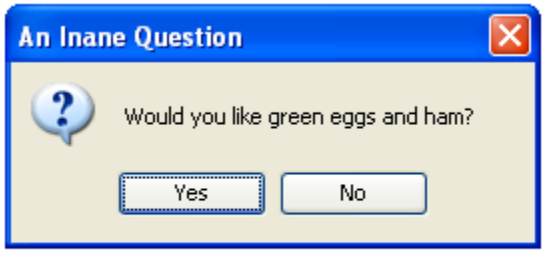
- **Containers** (ex: JFrame, JDialog, JPanel): são componentes que contêm outros objetos
- Em geral, qualquer GUI usa no mínimo uma classe desta categoria

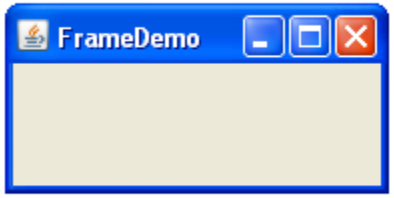


Categorias de classes

- Alguns **containers**:
  - JFrame
  - JDialog

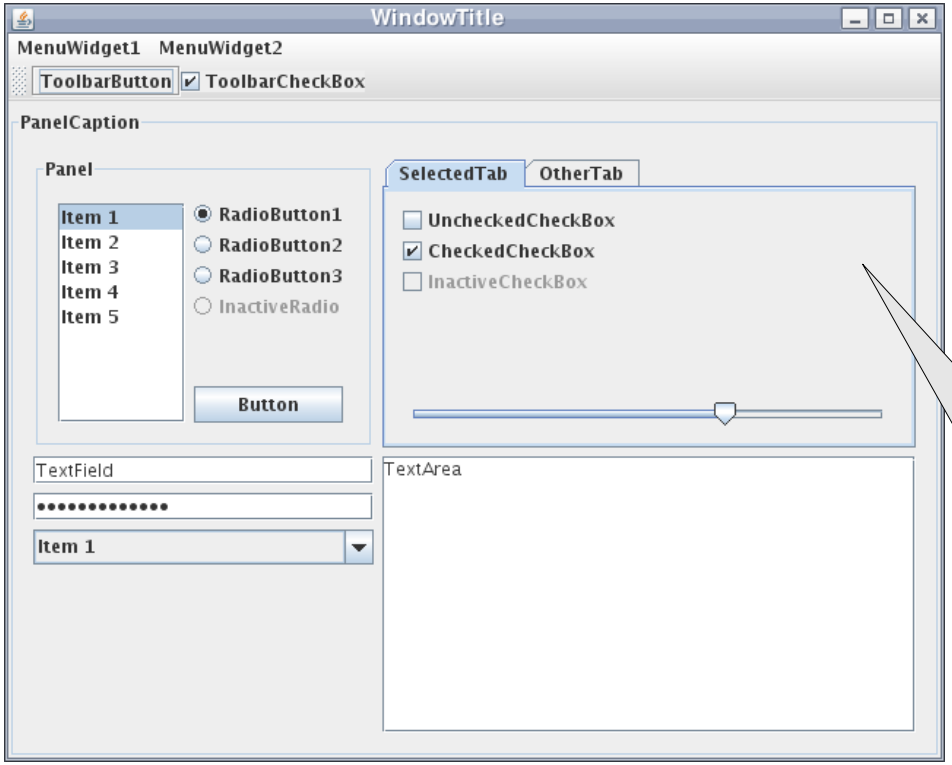
JFrame vazio





JDialog

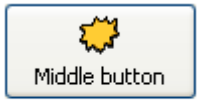
JFrame



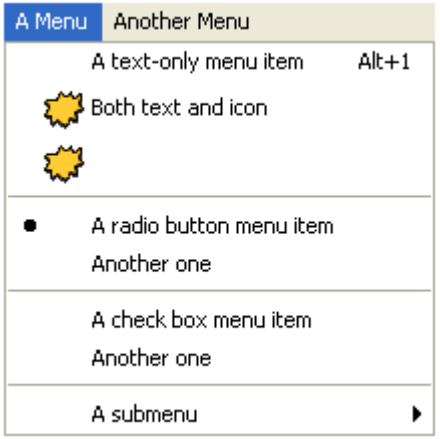
JFrame contendo outros objetos

Categorias de classes

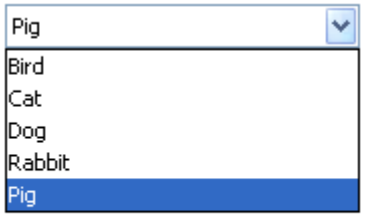
- **Componentes interativos básicos** (ex: JButton, JMenu, etc.): são componentes para entrada/saída



JButton



JCheckBox



JComboBox

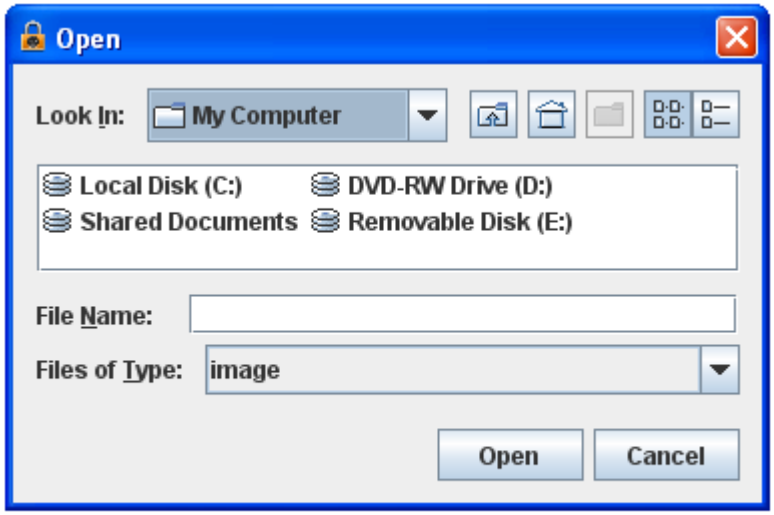
JTextField

JList

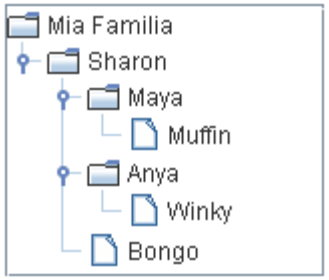
Fonte: http://download.oracle.com/javase/tutorial/ui/features/compWin.html

Categorias de classes

- **Componentes de alto nível** (ex: JFileChooser, JTable, etc.): são componentes "sofisticados" para interação com o usuário



JFileChooser



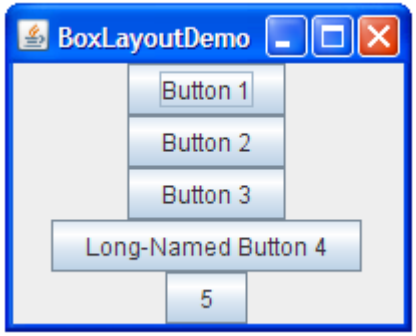
JTree

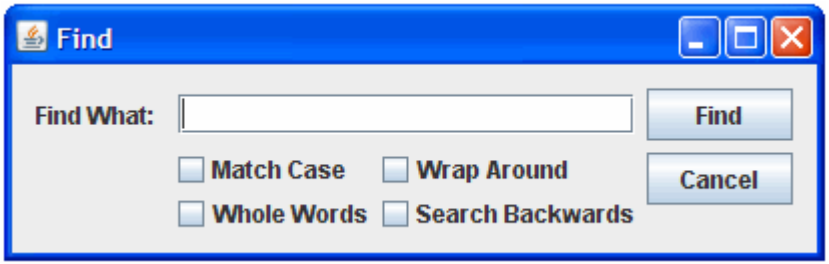
Host	User	Password	Last Modified
Biocca Games	Freddy	!#asf6Awwzb	Mar 16, 2006
zabble	ichabod	Tazb!34$fZ	Mar 6, 2006
Sun Developer	fraz@hotmail.co...	AasW541!fbZ	Feb 22, 2006
Heirloom Seeds	shams@gmail....	bkz[ADF78!	Jul 29, 2005
Pacific Zoo Shop	seal@hotmail.c...	vbAf124%z	Feb 22, 2006

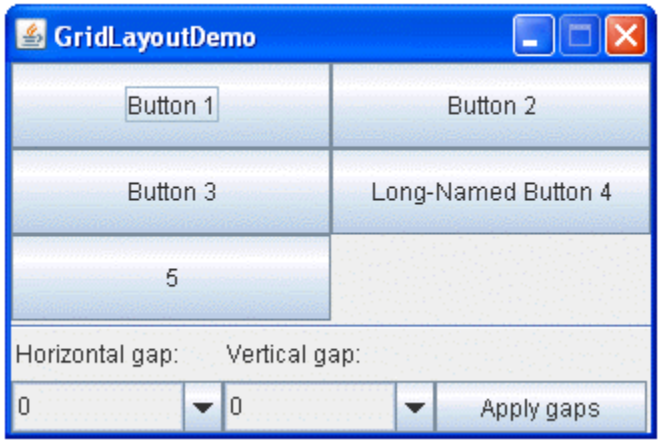
JTable

Categorias de classes

- **Layout Managers:** são classes que ajudam a organizar componentes em um container







Fonte:
http://download.oracle.com/javase/tutorial/uiswing/layout/visual.html

Categorias de classes

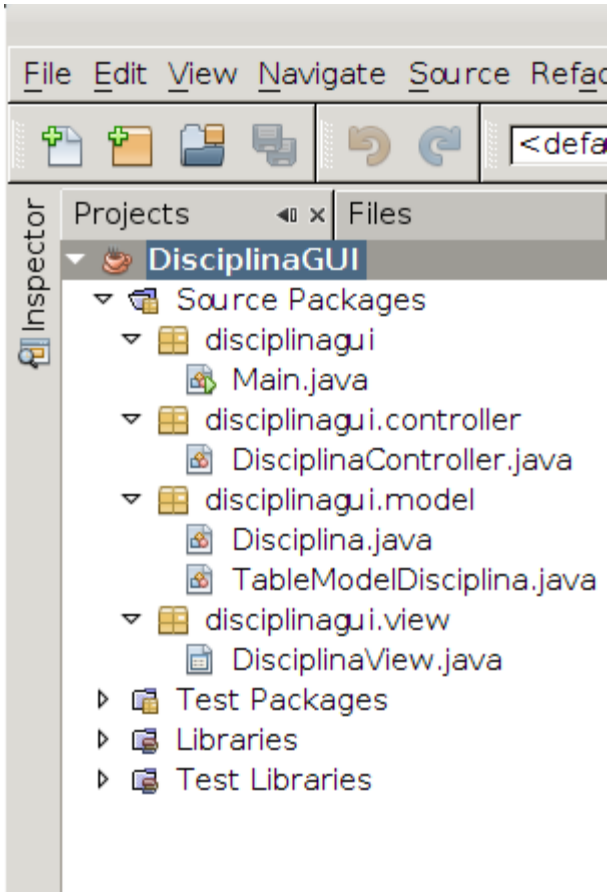
- **Eventos/ações:** são classes que representam a interação propriamente dita (ver interface ActionListener)

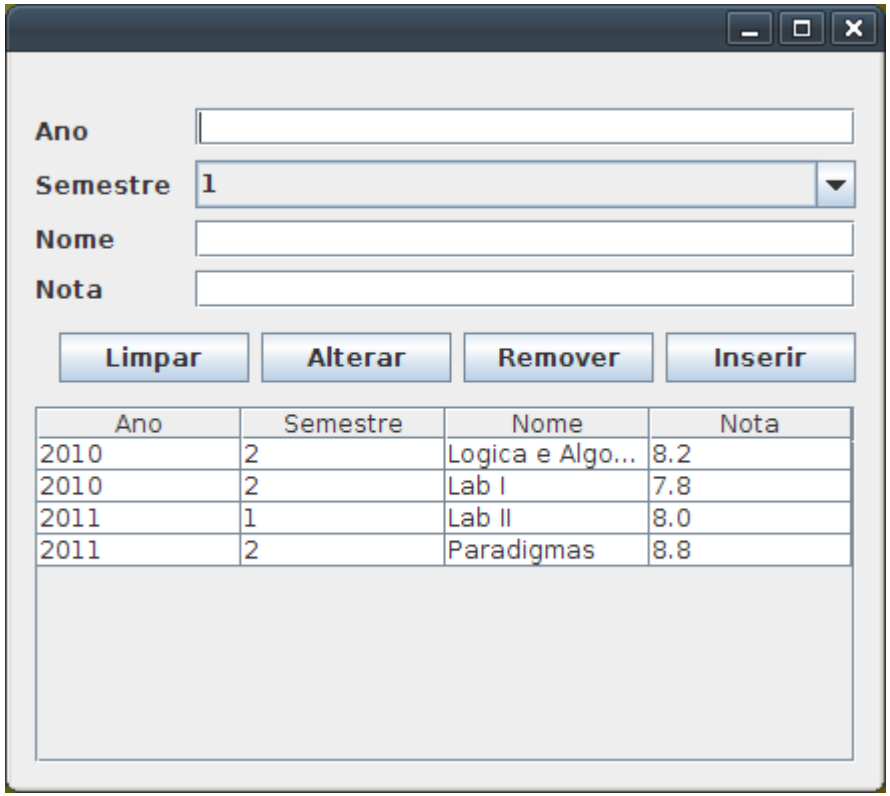
```
class ArrayListGUI extends JFrame {
  ...
    buttonLimpar = new javax.swing.JButton();
    buttonLimpar.setText("Limpar");
    buttonLimpar.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        buttonLimparActionPerformed(evt);
      }
    });
  ...
  private void buttonLimparActionPerformed(java.awt.event.ActionEvent evt) {
    textAno.setText("");
    comboSemestre.setSelectedIndex(0);
    textNome.setText("");
    textNota.setText("");
  }
}
```

Construindo um aplicativo

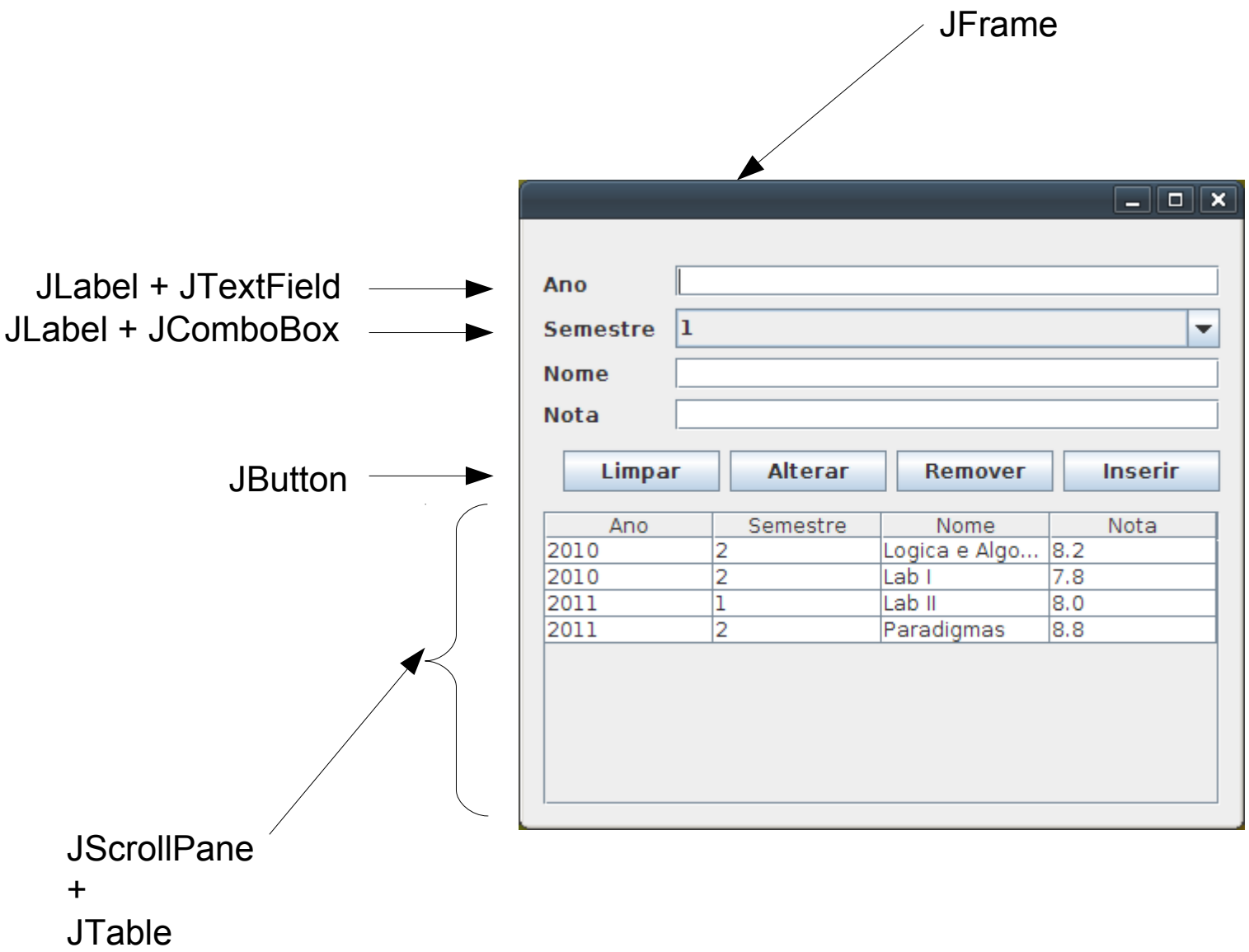
- Modelo MVC (Model-View-Controller): lógica e dados do aplicativo ficam isolados da interface gráfica
- **Model**: uma ou mais classes que representam os dados e a lógica do programa.
- **View**: mostra os dados e apresenta a interface com o usuário
- **Controller**: reage às ações do usuário, fazendo Model e View se atualizarem

Construindo um aplicativo: DisciplinaGUI





DisciplinaGUI: View



NetBeans gera DisciplinaView.java

DisciplinaGUI - NetBeans IDE 6.9

File Edit View Navigate Source Refactor Run Debug Team Tools Window Help

Projects | Files

- DisciplinaGUI
  - Source Packages
    - disciplinagui
      - Main.java
    - disciplinagui.controller
      - DisciplinaController.java
    - disciplinagui.model
      - Disciplina.java
      - TableModelDisciplina.java
    - disciplinagui.view
      - DisciplinaView.java
  - Test Packages
  - Libraries
  - Test Libraries

DisciplinaView.java

Source | Design

Ano

Semestre 1

Nome

Nota

Limpar | Alterar | Remover | Inserir

Palette

Swing Containers

Panel | Tabbed Pane | Split Pane | Scroll Pane | Tool Bar | Desktop Pane | Internal Frame | Layered Pane

Swing Controls

Label | Button | Toggle Button | Check Box | Radio Button | Button Group | Combo Box | List | Text Field | Text Area

Output | Tasks

60 | 30

NetBeans gera DisciplinaView.java

DisciplinaView.java ×

Source Design

```
26  public class DisciplinaView extends javax.swing.JFrame {
27
```

DisciplinaView.java ×

Source Design

```
69      // <editor-fold defaultstate="collapsed" desc="Generated Code">
70      private void initComponents() {
71
72          scrollpaneTable = new javax.swing.JScrollPane();
73          tableDisciplina = new javax.swing.JTable();
74          labelNome = new javax.swing.JLabel();
75          textNome = new javax.swing.JTextField();
76          labelNota = new javax.swing.JLabel();
77          textNota = new javax.swing.JTextField();
78          buttonLimpar = new javax.swing.JButton();
79          buttonRemover = new javax.swing.JButton();
80          buttonInserir = new javax.swing.JButton();
81          labelAno = new javax.swing.JLabel();
82          labelSemestre = new javax.swing.JLabel();
83          comboSemestre = new javax.swing.JComboBox();
84          buttonAlterar = new javax.swing.JButton();
85          textAno = new javax.swing.JTextField();
86
```

DisciplinaGUI: Controller reage às ações

DisciplinaView.java

Source Design

```
209     private void buttonRemoverActionPerformed(java.awt.event.ActionEvent evt) {
210         controller.remove();
211     }
212
213     private void buttonInserirActionPerformed(
214         controller.insert();
215     }
216
217     private void buttonAlterarActionPerformed(j
218         controller.update();
219     }
220
221     private void buttonLimparActionPerformed(java.a
222         controller.clear();
223     }
224
225     private void tableDisciplinaMouseClicked(java.awt.event.MouseEvent evt) {
226         controller.select();
227     }
228
```

buttonRemover
|
Events
|
Action
|
buttonRemoverActionPerformed

DisciplinaGUI: DisciplinaController.java

DisciplinaView.java × | DisciplinaController.java ×

```
13   * @author andrea
14   */
15
16  public class DisciplinaController {
17
18      private DisciplinaView view;
19      private TableModelDisciplina model;
20
21      public DisciplinaController(DisciplinaView view, TableModelDisciplina model) {...}
25
26      public void insert() {...}
32
33      public void update() {...}
43
44      public void remove() {...}
51
52      public void select() {...}
63
64      public void clear() {...}
70
71      private Disciplina newFromView() {...}
84  }
85
```

DisciplinaGUI: DisciplinaController.java

getSelectedRow()
é um método de
JTable

DisciplinaController.java

```
43
44      public void remove() {
45          int index = view.getTableDisciplina().getSelectedRow();
46          if (index == -1) {
47              return;
48          }
49          model.remove(index);
50      }
```

Atualiza o modelo

DisciplinaGUI: Model

- class **Disciplina**
  - dados de uma disciplina (ano, semestre, nome, nota)
- class **TableModelDisciplina**
  - várias disciplinas (ArrayList<Disciplina>)
  - extends AbstractTableModel
  - implementa métodos desta classe:
    - ✓ getColumnCount
    - ✓ getColumnName(int columnIndex)
    - ✓ getRowCount
    - ✓ getValueAt(int rowIndex, int ColumnIndex)

DisciplinaGUI: TableModelDisciplina.java

Todo JTable tem um modelo derivado de AbstractTableModel

TableModelDisciplina.java

```
16
17  public class TableModelDisciplina extends AbstractTableModel {
18
19      private static final String[] columnNames = {"Ano", "Semestre", "Nome", "Nota"};
20
21      private ArrayList<Disciplina> discips;
22
23      public TableModelDisciplina() {
24          discips = new ArrayList<Disciplina>();
25          discips.add(new Disciplina(2010,2,"Logica e Algoritmo", 8.2f));
26          discips.add(new Disciplina(2010,2,"Lab I", 7.8f));
27          discips.add(new Disciplina(2011,1,"Lab II", 8.0f));
28          discips.add(new Disciplina(2011,2,"Paradigmas", 8.8f));
29      }
30
31      public void remove(int index) {
32          discips.remove(index);
33          fireTableRowsDeleted(index, index);
34      }
35
36      public Disciplina select(int index) {
37          return discips.get(index);
38      }
```

DisciplinaGUI: TableModelDisciplina.java

Estes métodos são invocados para exibir a JTable

TableModelDisciplina.java

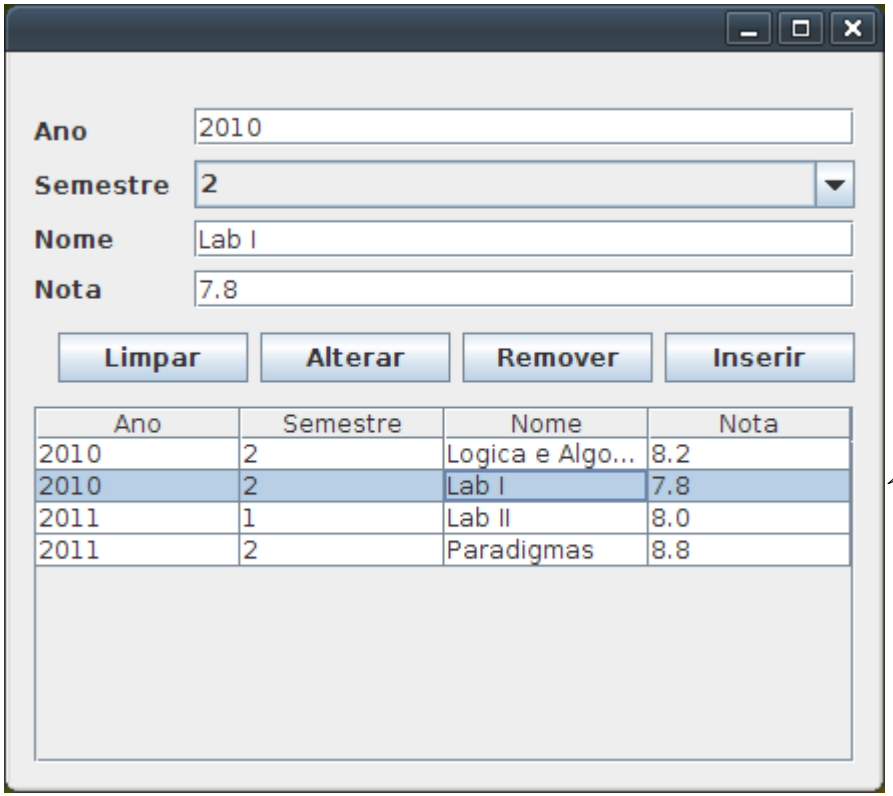
```java
    @Override
    public int getColumnCount() {
        return columnNames.length;
    }

    @Override
    public String getColumnName(int columnIndex) {
        return columnNames[columnIndex];
    }

    @Override
    public int getRowCount() {
        return discips.size();
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        switch(columnIndex) {
            case 0: return discips.get(rowIndex).getAno();
            case 1: return discips.get(rowIndex).getSemestre();
            case 2: return discips.get(rowIndex).getNome();
            case 3: return discips.get(rowIndex).getNota();
        }
        return null;
    }
```

Remover disciplina



Usuário:

Seleciona linha da tabela
Clica no botão Remover

Remover disciplina

DisciplinaView.java

```
208
209     private void buttonRemoverActionPerformed(java.awt.event.ActionEvent evt) {
210         controller.remove();
211     }
```

DisciplinaController.java

```
43
44     public void remove() {
45         int index = view.getTableDisciplina().getSelectedRow();
46         if (index == -1) {
47             return;
48         }
49         model.remove(index);
50     }
```

TableModelDisciplina.java

```
31     public void remove(int index) {
32         discips.remove(index);
33         fireTableRowsDeleted(index, index);
34     }
35
```