

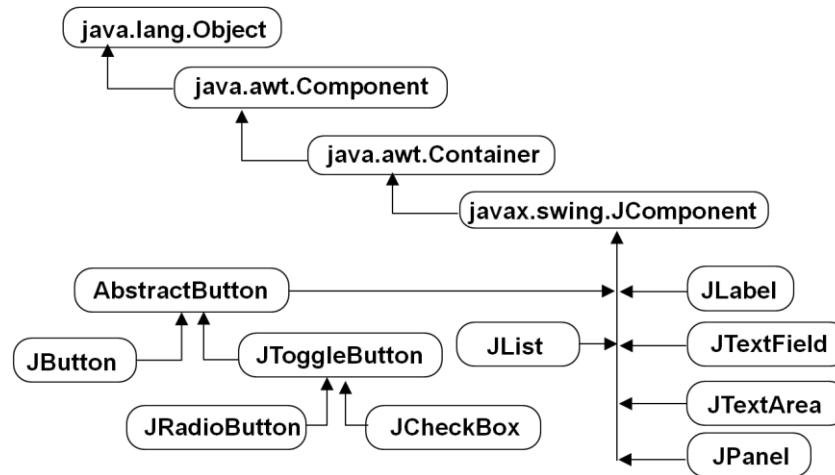


Linguagem de Programação Java
Interface Gráfica

Profa. Joyce Miranda

API Gráfica

► **Swing:** API gráfica chamada



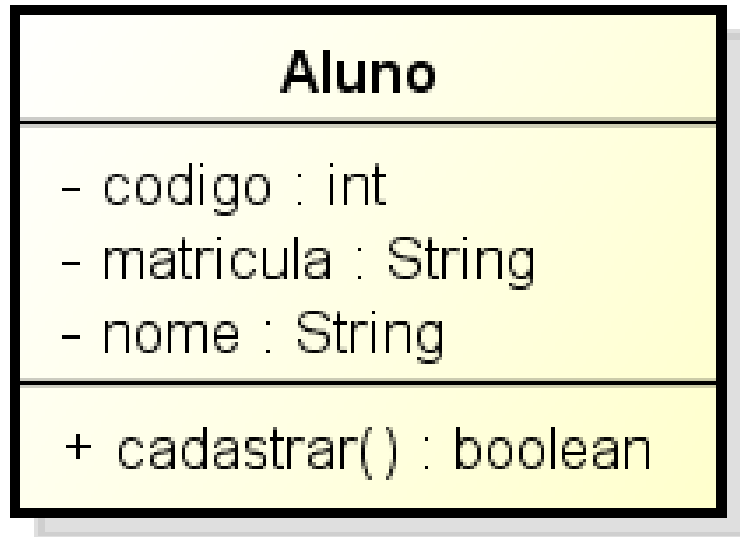
► **Component**

- É uma “peça” que exerce alguma função na interface.

► **Container**

- É um componente capaz de conter outros componentes.
- Define métodos como `add()` para adicionar componentes em seu interior.

API Gráfica



powered by astah*

- ▶ *Criar o JAVABEAN da classe*
- ▶ *Implementar os métodos*

The screenshot shows a Java Swing window titled "Tela de Cadastro de Aluno". It features a standard Windows-style title bar with minimize, maximize, and close buttons. The main content area has a light gray background. On the left, there are three labels: "Código:", "Matricula:", and "Nome:". To the right of each label is a white text input field. At the bottom of the window, there are two blue buttons with white text: "Cadastrar" on the left and "Limpar" on the right.

API Gráfica

1. Criar a classe de execução: “TelaAluno”
 - ▶ Ela herdará a classe **Jframe**, responsável pela janela principal.
2. Declarar Componentes da tela
3. Modificar Construtor



API Gráfica

```
7 import javax.swing.*;
8
9 public class TelaAluno extends JFrame {
10
11     //declarando componentes da tela
12     JPanel painel = new JPanel();
13     JLabel lblCodigo = new JLabel("Código: ");
14     JLabel lblMatricula = new JLabel("Matrícula: ");
15     JLabel lblNome = new JLabel("Nome: ");
16     JTextField txtCodigo = new JTextField();
17     JTextField txtMatricula = new JTextField();
18     JTextField txtNome = new JTextField();
19     JButton btnCadastrar = new JButton("Cadastrar");
20     JButton btnLimpar = new JButton("Limpar");
21
22     //continua...
```

API Gráfica

```
22  //continua...
23
24  //modificando construtor
25  public TelaAluno() {
26
27      //setando configurações de layout
28      //linhas,colunas, espaçamento direita, espaçamento inferior
29      GridLayout grid = new GridLayout(4,2, 5, 5);
30      painel.setLayout(grid);
31
32      //continua...
```

API Gráfica

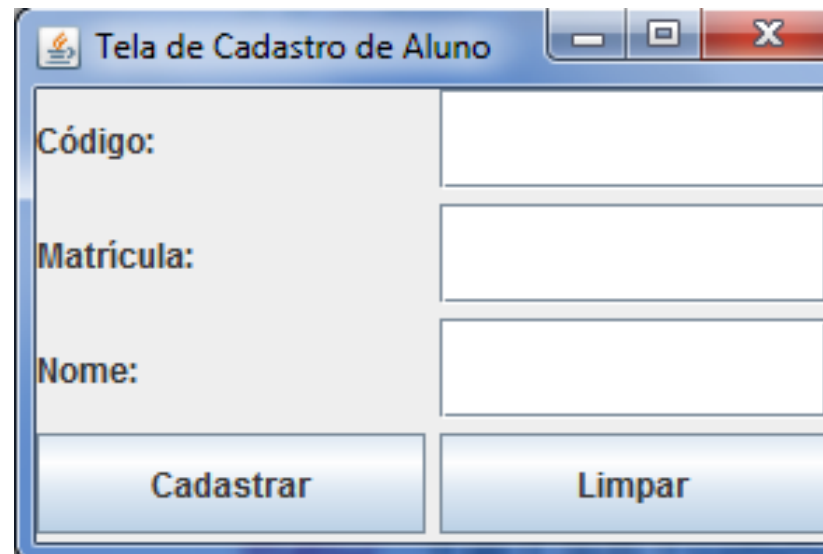
```
32      //continua...
33
34      //adicionando componentes ao painel
35      painel.add(lblCodigo);
36      painel.add(txtCodigo);
37      painel.add(lblMatricula);
38      painel.add(txtMatricula);
39      painel.add(lblNome);
40      painel.add(txtNome);
41      painel.add(btnCadastrar);
42      painel.add(btnLimpar);
43
44      //adicionando painel ao frame
45      add(painel);
46
47      //continua...
```

API Gráfica

```
47      //continua...
48
49      //setando titulo
50      setTitle("Tela de Cadastro de Aluno");
51      //setando tamanho da tela
52      //x, y, largura, altura
53      setBounds(200, 200, 300, 200);
54      //tornando o frame visivel
55      setVisible(true);
56
57      //fim construtor
58
59  }
```


API Gráfica

```
61 public static void main(String args[]){  
62     //chamando construtor  
63     new TelaAluno();  
64 }
```



API Gráfica

► Exercício:

Adição
<ul style="list-style-type: none">- numero1 : double- numero2 : double- resultado : double
+ somar() : void

Número 1:

Número 2:

Resultado:

API Gráfica

- ▶ Programação orientada a eventos
 - ▶ Criar programas cuja ordem de execução seja determinada pela ocorrência de eventos.
 - ▶ Todo objeto pode ser notificado por um evento.
 - ▶ Alguns exemplos:
 - ❑ Usuário clica em um botão ActionListener
 - ❑ Usuário fecha um frame WindowListener
 - ❑ Usuário pressiona um botão do mouse MouseListener
 - ❑ Usuário move o mouse MouseMotionListener
 - ❑ Componentes se tornam visíveis ComponentListener

API Gráfica

► Tratador de eventos

1. Criar a classe que implemente uma interface de um listener:

```
public class MyClassListener implements ActionListener {
```

2. Implementar os métodos obrigatórios da interface

```
public void actionPerformed(ActionEvent e) {  
    ...//code that responds to the event...  
}
```

3. Adicionar esse evento a um componente

```
someComponent.addActionListener(new MyClassListener() );
```

API Gráfica

- Incluir na classe TelaAluno uma classe aninhada (uma classe dentro da outra) que implementa a interface Listener

```
69  class CadastroAlunoListener implements ActionListener{
70
71      public void actionPerformed(ActionEvent e) {
72
73          int codigo = Integer.parseInt(txtCodigo.getText());
74          String matricula = txtMatricula.getText();
75          String nome = txtNome.getText();
76
77          Aluno al = new Aluno(codigo, matricula, nome);
78          al.cadastrar();
79
80      }
81  }
```

API Gráfica

- ▶ Incluir no construtor da classe “TelaAluno” o código abaixo:

```
57      //adicionando evento ao componente
58      btnCadastrar.addActionListener(new CadastroAlunoListener());
```

API Gráfica

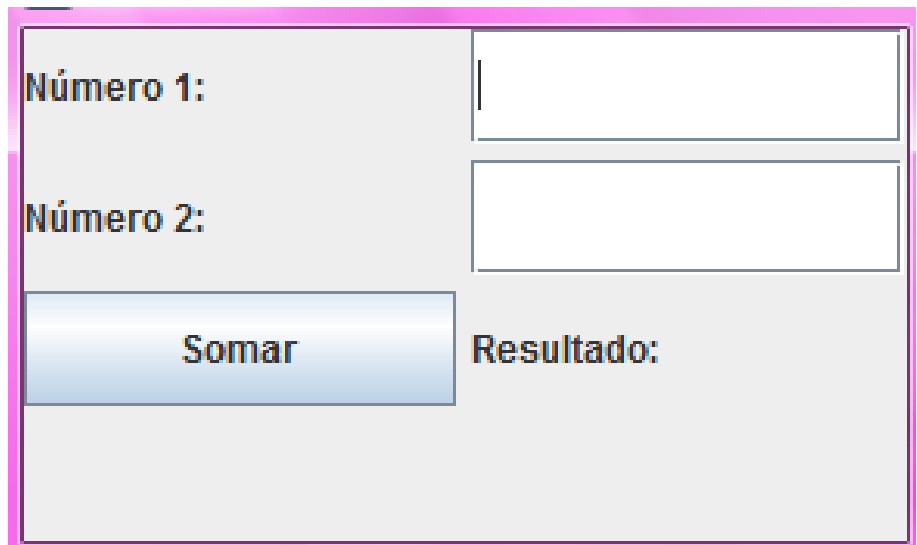
► Exercício:

Adição

- numero1 : double
- numero2 : double
- resultado : double

+ somar() : void

powered by astah*

A graphical user interface for an addition application. It features two input fields labeled 'Número 1:' and 'Número 2:'. Below these is a blue button labeled 'Somar'. To the right of the button is a label 'Resultado:' followed by a large empty rectangular area for the output.

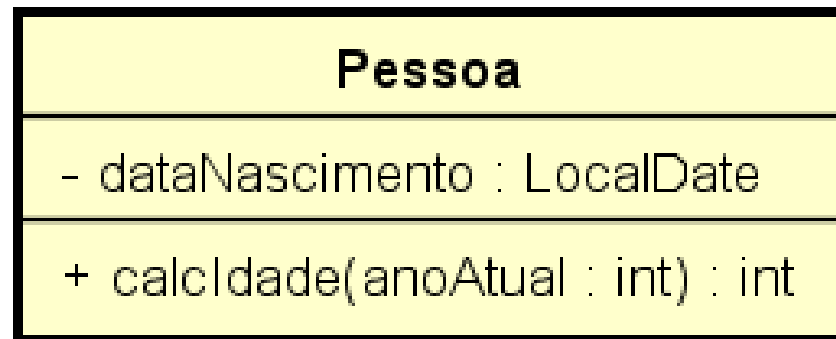
Número 1:

Número 2:

Resultado:

API Gráfica

► Exercício:



Data de Nascimento	<input type="text"/>	Calcular Idade	Idade	<input type="text"/>
--------------------	----------------------	----------------	-------	----------------------