

Programação em Java usando orientação a eventos

Programação Orientada a Objetos

Passo para criar interface gráfica

- Criar uma relação de especialização com a classe que representa sua tela
- Declarar como atributos os elementos que serão adicionados à tela
- Definir a forma de alocação dos elementos gráficos na tela
- No construtor, instanciar, configurar e posicionar os itens na tela
- Tratar os eventos dos componentes para tratar as ações do usuário com a interface gráfica

Quadro 1.5 | Exemplo de interface da orientação a objetos do Java

```
1. public interface AcessoElementos {  
2.     public int getElemento (int index);  
3.     public void setElemento (int index);  
4. }
```

define quais métodos as classes que utilizarem essa interface deverão ter

- interface
 - elementos da OO que descrevem um conjunto de métodos
- interface gráfica
 - elementos como botões, janelas, campos, etc.
 - interação do usuário com o sistema

import javax.swing.*;  biblioteca

public class Aluno **extends** JFrame **implements** AcessoElementos {

@Override

public int getElemento(int index) {

return 0;

}

 interface

@Override


public void setElemento(int index) {

// ... códigos ...

}

}

 métodos da interface

 diz ao compilador
que é uma
sobreposição

Tratamento de eventos

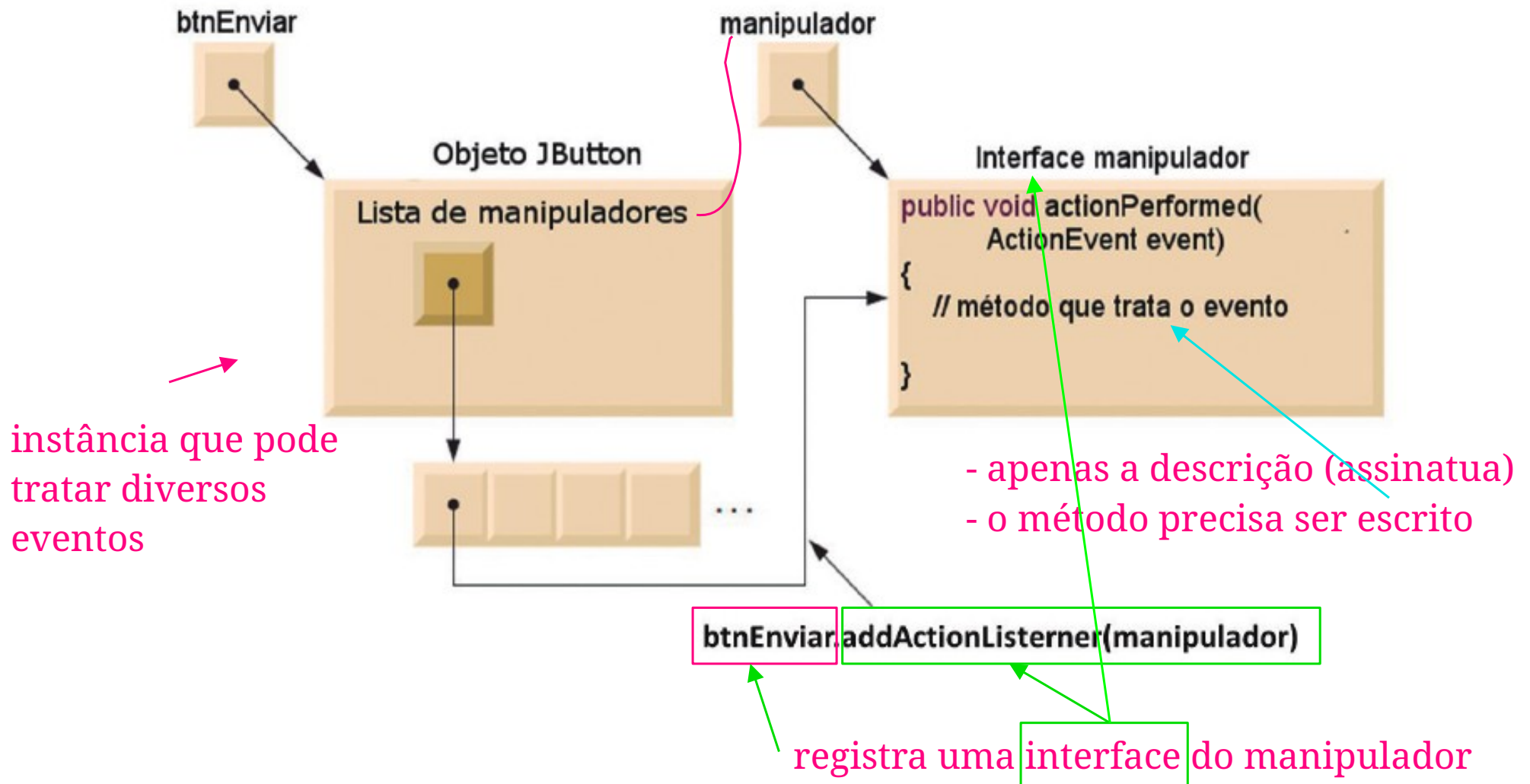
Quadro 1.7 | Eventos na linguagem Java

Evento	Descrição
ActionListener	Evento gerado pelo clique com o mouse em um botão.
ItemListener	Evento gerado quando um item em uma lista é selecionado.
FocusListener	Evento gerado quando um elemento ganha ou perde foco.
WindowListener	Evento gerado quando ocorre uma mudança na janela.

- Uma ação do usuário gera uma reação do sistema
 - Clicar em botão, pressionar tecla, maximizar janela, etc.
 - Cada evento precisa de um método

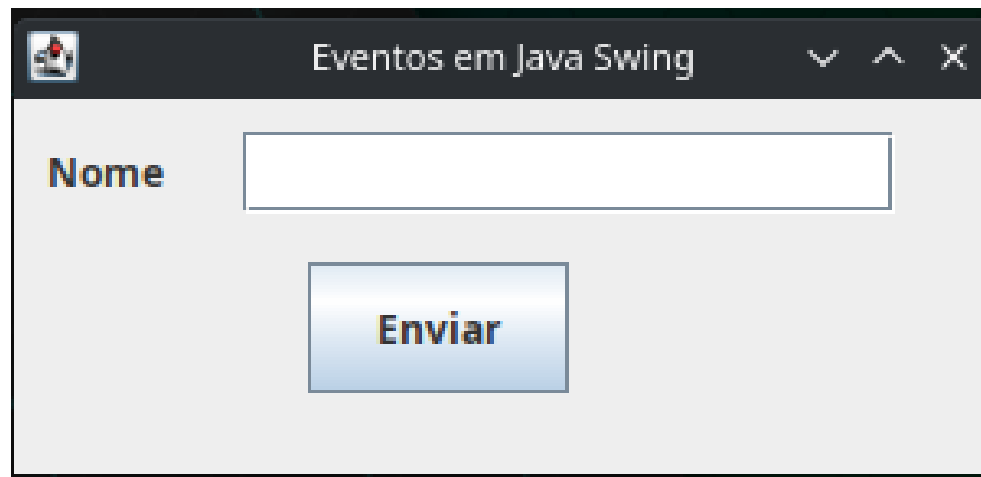
Exemplo de como tratar evento

Figura 1.5 | Exemplo de como tratar o evento de clique em JButton do Java Swing



Exemplo de interface gráfica

- Implementação do tratamento de eventos
 - a classe que está especializando o JFrame
 - deve implementar métodos da interface que tratam os eventos e associam essa interface aos componentes que enviarão os eventos



(implementação no próximo slide)

```

1  import java.awt.Container;
2  import javax.swing.*;
3  public class PrimeiraTela extends JFrame {
4      private JButton btnok;
5      private JTextField txtNome;
6      private JLabel lblNome;
7      private Container ctn;
8      public PrimeiraTela() {
9          setSize(300, 140);
10         setTitle("Eventos em Java Swing");
11         ctn = getContentPane();
12         ctn.setLayout(null);
13         btnok = new JButton("Enviar");
14         lblNome = new JLabel("Nome");
15         txtNome = new JTextField();
16         lblNome.setBounds(10, 10, 100, 25);
17         txtNome.setBounds(70, 10, 200, 25);
18         btnok.setBounds(90, 50, 80, 40);
19         ctn.add(lblNome);
20         ctn.add(txtNome);
21         ctn.add(btnok);
22         // nesse ponto vamos inserir tratamento dos eventos
23         setVisible(true);
24     }
25     public static void main(String[] args) {
26         PrimeiraTela tela = new PrimeiraTela();
27     }
28     // nesse ponto vamos inserir tratamento dos eventos
29 }

```

vide próximo slide

classes que serão importadas

definição da classe que especializa a classe JFrame

atributos que representam os componentes da tela

tamanho da janela e título

permite usar posicionamento absoluto na tela

faz a tela aparecer

vide slide 10

cria o objeto

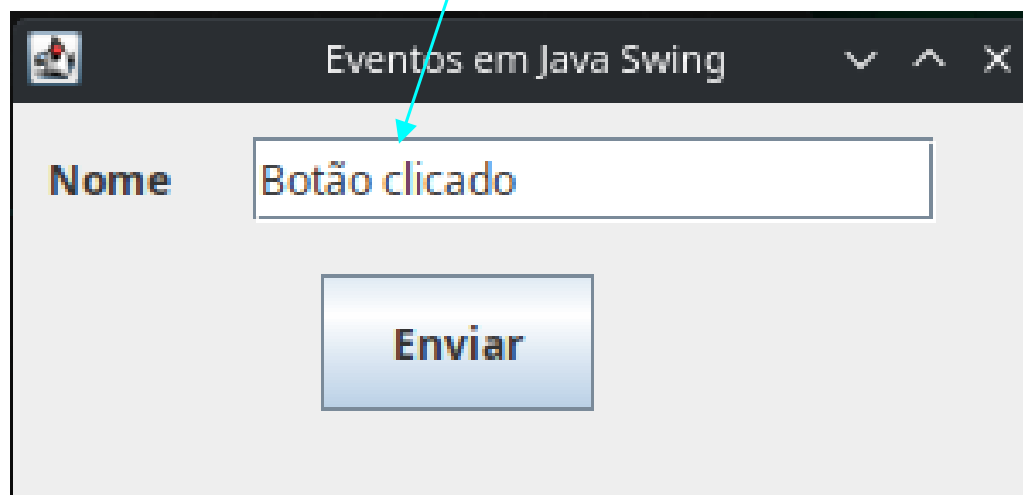
quando cria, executa o construtor

Campos

- Abstract Window Toolkit (AWT)
 - GUI original do Java, anterior ao Swing
- Um contêiner fornece um espaço onde um componente pode ser localizado (Panel, Frame, Window)
 - Um layout padrão está presente em cada contêiner que pode ser substituído usando o método `setLayout`
- `setLayout(null)`
 - permite usar posicionamento absoluto
- `awt.event.ActionEvent`
 - elemento semântico que indica se uma ação ocorreu

- `public static void main(String[] args)`
 - `public`
 - acesso por qualquer classe dentro ou fora do projeto
 - `static`
 - o método não precisa ser instanciado para ser usado
 - `void`
 - tipo de retorno do método (não retorna nada)
 - `String[] args`
 - argumentos da linha de comando

Tratamento de evento



(implementação no próximo slide)

```
import java.awt.event.ActionEvent; import java.awt.event.ActionListener;
import java.awt.Container; import javax.swing.*;
```

```
public class PrimeiraTela2 extends JFrame implements ActionListener {
    private JButton btnok; private JTextField txtNome;
    private JLabel lblNome; private Container ctn;
```

```
public PrimeiraTela2() {
    setSize(300, 140); setTitle("Eventos em Java Swing");
    ctn = getContentPane(); ctn.setLayout(null);
    btnok = new JButton("Enviar"); lblNome = new JLabel("Nome");
    txtNome = new JTextField();
    lblNome.setBounds(10, 10, 100, 25); txtNome.setBounds(70, 10, 200, 25);
    btnok.setBounds(90, 50, 80, 40);
    ctn.add(lblNome); ctn.add(txtNome); ctn.add(btnok);
    btnok.addActionListener(this);
    setVisible(true); }
```

```
public static void main(String[] args) {
    PrimeiraTela2 tela = new PrimeiraTela2(); }
```

@Override

```
public void actionPerformed(ActionEvent e) {
    // caso seja necessário tratar eventos de mais de um botão
    if (e.getActionCommand().equals("Enviar")) {
        txtNome.setText("Botão clicado"); } }
```

implementa os
métodos na interface
em espera (listener)

uma ação chama

objeto corrente (atual)

Usando classe anônima

```

import java.awt.event.ActionEvent; import java.awt.event.ActionListener;
import java.awt.Container; import javax.swing.*;
public class PrimeiraTela3 extends JFrame {
    private JButton btnok; private JTextField txtNome;
    private JLabel lblNome; private Container ctn;
    public PrimeiraTela3() {
        setSize(300, 140); setTitle("Eventos em Java Swing");
        ctn = getContentPane(); ctn.setLayout(null);
        btnok = new JButton("Enviar"); lblNome = new JLabel("Nome");
        txtNome = new JTextField();
        lblNome.setBounds(10, 10, 100, 25); txtNome.setBounds(70, 10, 200, 25);
        btnok.setBounds(90, 50, 80, 40);
        ctn.add(lblNome); ctn.add(txtNome); ctn.add(btnok);
        setVisible(true);
        btnok.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                txtNome.setText("Botão clicado"); } }); }
    public static void main(String[] args) {
        PrimeiraTela3 tela = new PrimeiraTela3(); } }

```

basicamente o
mesmo código dos
slides anteriores

a classe anônima pode
acessar os elementos da
classe que a engloba

para cada elemento gráfico
da tela é possível criar um
método específico

Tratar evento de mudança de texto em um
JTextField

```

import java.awt.event.ActionEvent; import java.awt.event.ActionListener;
import java.awt.Container; import javax.swing.event.DocumentListener;
import javax.swing.event.DocumentEvent; import javax.swing.*;

public class PrimeiraTela4 extends JFrame {
    private JButton btnok; private JTextField txtNome;
    private JLabel lblNome; private Container ctn;
    public PrimeiraTela4() {
        setSize(300, 140); setTitle("Eventos em Java Swing");
        ctn = getContentPane(); ctn.setLayout(null);
        btnok = new JButton("Enviar"); lblNome = new JLabel("Nome");
        txtNome = new JTextField(); lblNome.setBounds(10, 10, 100, 25);
        txtNome.setBounds(70, 10, 200, 25); btnok.setBounds(90, 50, 80, 40);
        ctn.add(lblNome); ctn.add(txtNome); ctn.add(btnok); setVisible(true);

        txtNome.getDocument().addDocumentListener(new DocumentListener() {
            public void removeUpdate(DocumentEvent e) {
                /* ações quando texto for apagado */ }
            public void insertUpdate(DocumentEvent e) {
                /* ações quando texto for inserido */ }
            public void changedUpdate(DocumentEvent e) {
                /* ações quando texto for alterado */ } }); }

    public static void main(String[] args) {
        PrimeiraTela4 tela = new PrimeiraTela4(); } }

```

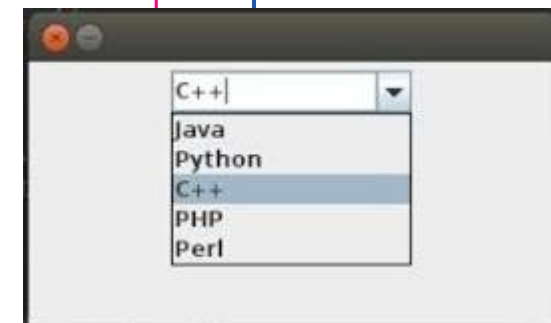
eventos relacionados a
uma caixa de texto

Tratamento de seleção de um JComboBox

```
1. import java.awt.event.ItemEvent;
2. import java.awt.event.ItemListener;

3. jmbTipos.addItemListener(new ItemListener() {
4.     public void itemStateChanged(ItemEvent e) {
5.         if (e.getStateChange() == ItemEvent.
6.             SELECTED)
7.             {
8.                 trataJmbTipos();
9.             }
10.    });

11. public void trataJmbTipos()
12. {
13.     JOptionPane.showMessageDialog(this,
14.         "Item selecionado: " + jmbTipos.
15.         getSelectedItem());
16. }
```



Exemplo

- Rodar os códigos da unidade 1 seção 2