

# Programação Orientada a Objetos

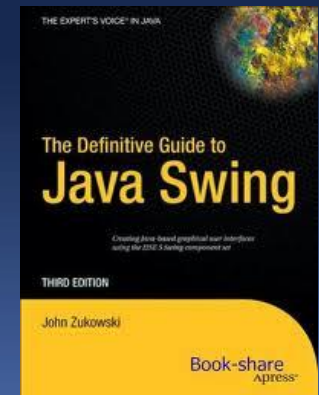
## Unidade 7 – Interface Gráfica



Prof. Aparecido V. de Freitas  
Doutor em Engenharia  
da Computação pela EPUVSP  
[aparecido.freitas@prof.uscs.edu.br](mailto:aparecido.freitas@prof.uscs.edu.br)  
[aparecidovfreitas@gmail.com](mailto:aparecidovfreitas@gmail.com)

# Bibliografia

- The Definitive Guide to Java Swing – Third edition – John Zukowski – 2005 - Apress
- Beginning Java 2 – Ivor Horton – 2011 WROX
- Java – The Complete Reference – 8th Edition – Herbert Schildt – Oracle Press - 2011
- Core Java Fundamentals – Horstmann / Cornell – PTR- Volumes 1 e 2 – 8th Edition
- Inside the Java 2 – Virtual Machine Venners – McGrawHill
- Understanding Object-Oriented Programming with JAVA – Timothy Budd – Addison Wesley
- Head First Java, 2nd Edition by Kathy Sierra and Bert Bates
- Effective Java, 2nd Edition by Joshua Bloch - 2008
- Thinking in Java (4th Edition) by Bruce Eckel
- Java How to Program - 9th Edition by Paul Deitel and Harvey Deitel



# Introdução

- A **API** Java Swing inclui suporte para diversas coisas que você geralmente vê em interfaces não-web: janelas, botões, menus, etc...
- Swing utiliza a **JFC** – Java Foundation Class, um **toolkit** que suporta a criação de aplicações que podem ser executadas em diferentes sistemas operacionais.



# Uma simples aplicação Swing

- Aplicações **Swing** seguem um fluxo básico para o desenvolvimento.
- Primeiro, cria-se um objeto **JFrame** que corresponde à **janela principal** e dá suporte aos outros componentes da interface (menu, botões, etc.)



# Uma simples aplicação Swing

```
package br.uscs;

import javax.swing.JFrame;

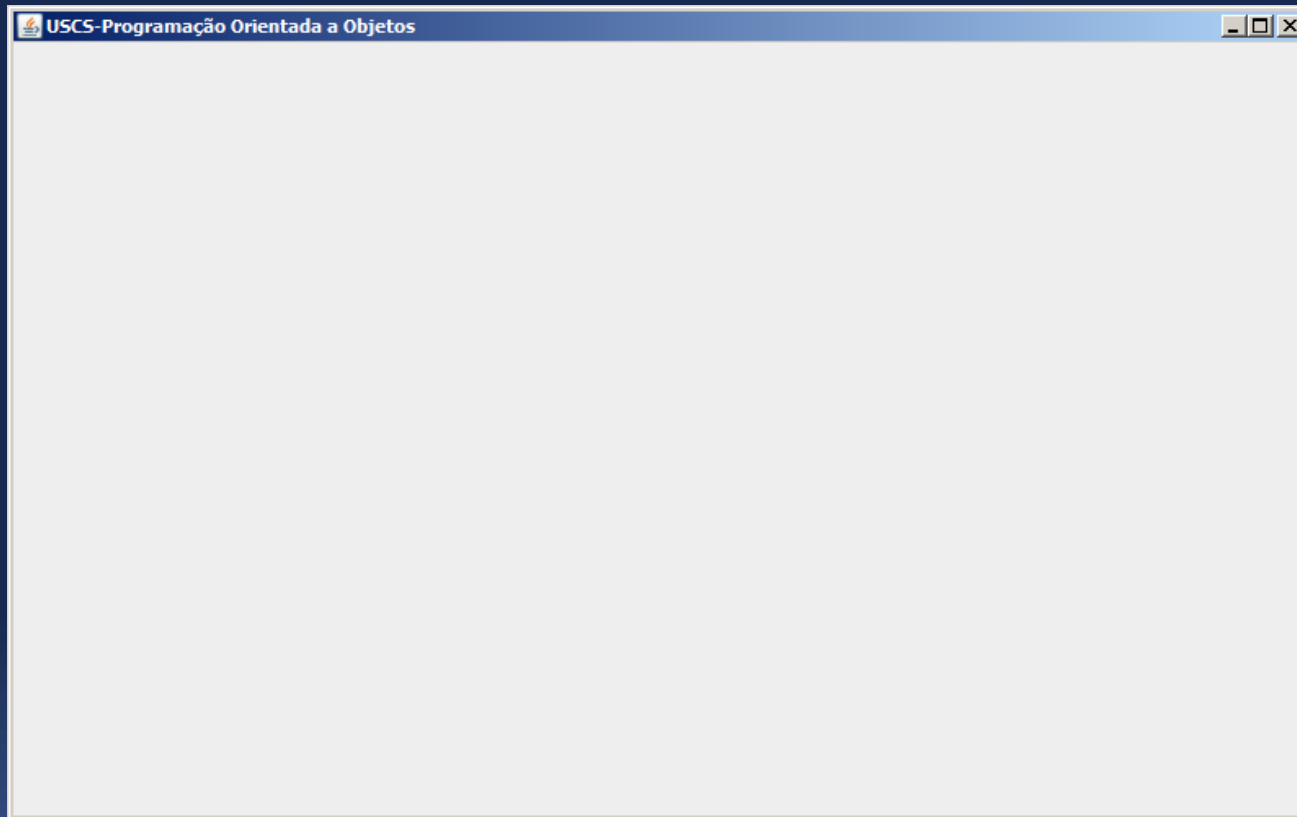
public class JFrame_01 extends JFrame{

    public JFrame_01() {
        super("USCS-Programação Orientada a Objetos");
        this.setSize(800,500);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setVisible(true);
    }

    public static void main(String[] args) {
        JFrame_01 app = new JFrame_01();
    }
}
```

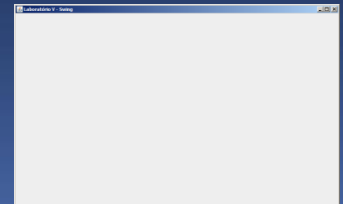


# Uma simples aplicação Swing



# Uma simples aplicação Swing

- O que o programa anterior faz é simplesmente criar e exibir uma janela vazia.
- O objeto do tipo **Frame** define a janela.
- Em seguida, definimos alguns atributos da janela (tamanho e forma de fechar a janela).
- O método **setVisible** define se o componente será visível ou não.



# Modificando a cor da janela



```
package br.uscs;

import java.awt.Color;
import javax.swing.JFrame;

public class JFrame_01 extends JFrame{

    public JFrame_01() {
        super("USCS-Programação Orientada a Objetos");
        this.setSize(800,500);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

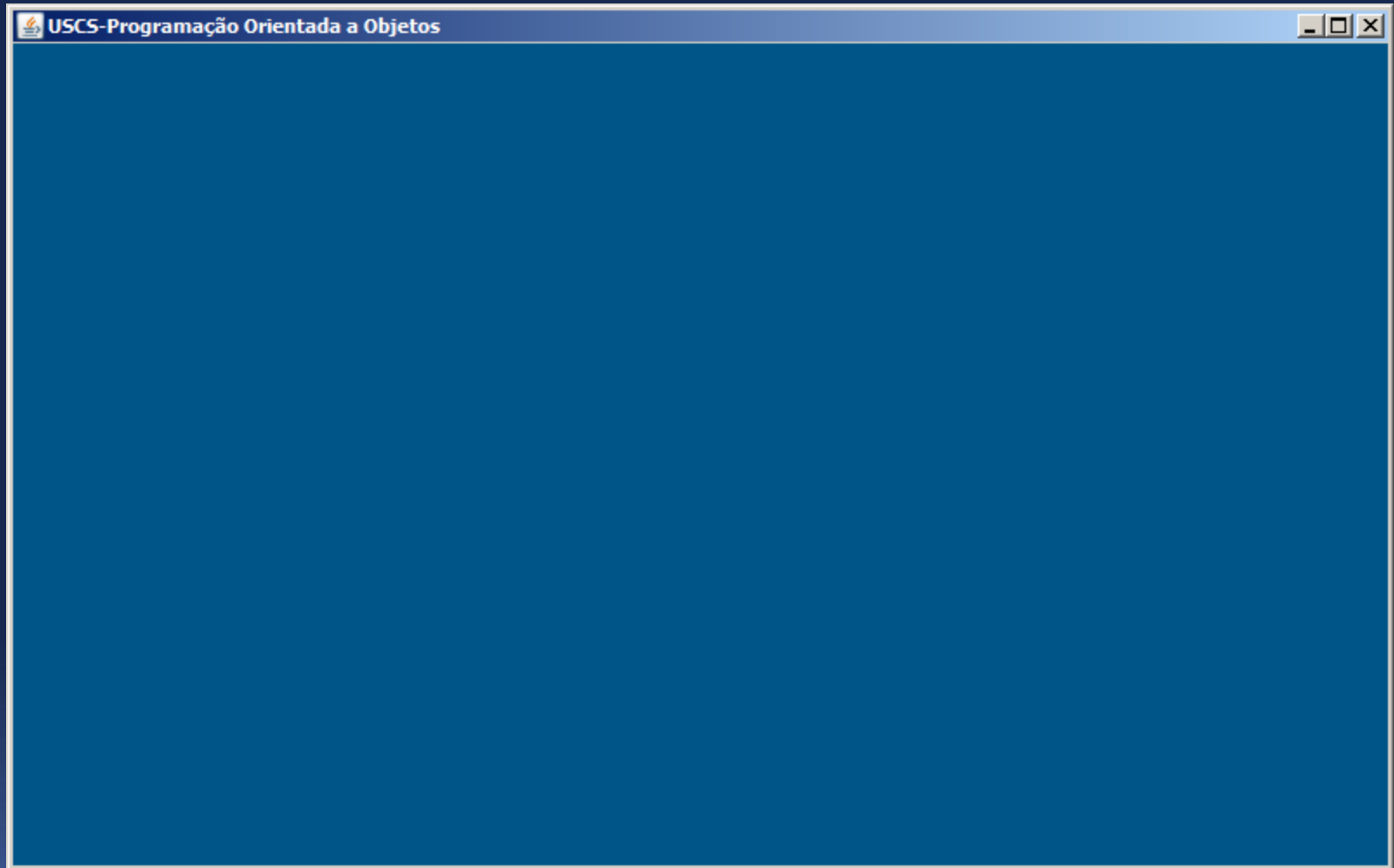
        Color cor = new Color (0x005588);
        this.getContentPane().setBackground(cor);
        this.setVisible(true);
    }

    public static void main(String[] args) {
        JFrame_01 app = new JFrame_01();
    }
}
```





# Modificando a cor da janela



# Como centralizo a janela no computador ?



# Centralizando a janela

```
package br.uscs;

import java.awt.Color;
import javax.swing.JFrame;

public class JFrame_01 extends JFrame{

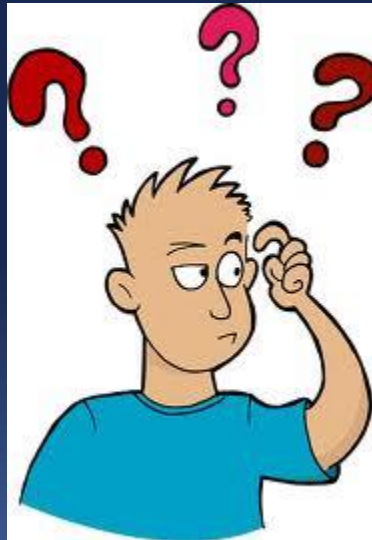
    public JFrame_01() {
        super("USCS-Programação Orientada a Objetos");
        this.setSize(800,500);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        Color cor = new Color (0x005588);
        this.getContentPane().setBackground(cor);

        this.setLocationRelativeTo(null);
        this.setVisible(true);
    }

    public static void main(String[] args) {
        JFrame_01 app = new JFrame_01();
    }
}
```

# Como acrescento um label à janela ?





```
package br.uscs;
```

```
import java.awt.Color;
```

```
import javax.swing.JFrame;
```

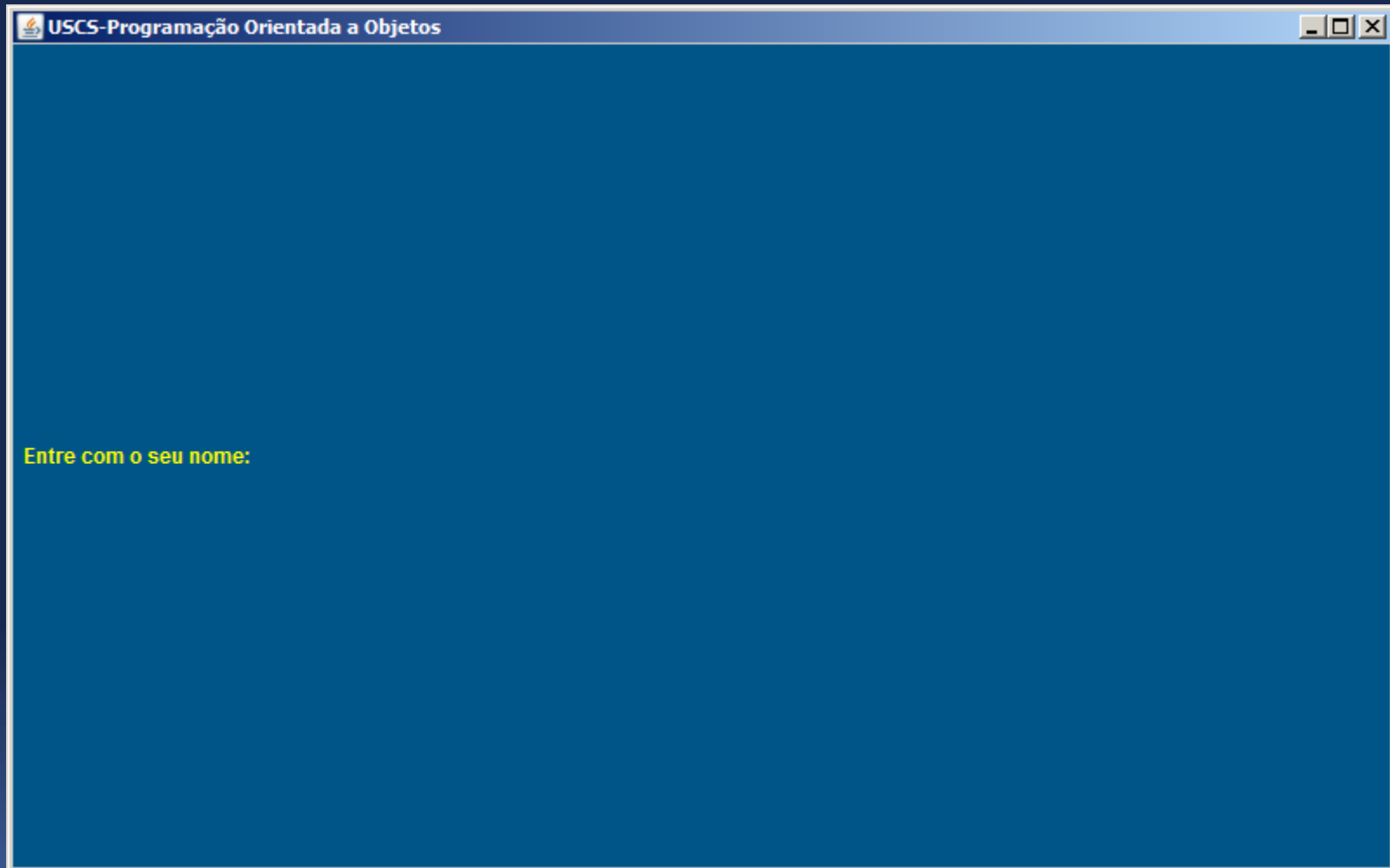
```
import javax.swing.JLabel;
```

# Acrescentando label

```
public class JFrame_01 extends JFrame {  
    public JFrame_01() {  
        super("USCS-Programação Orientada a Objetos");  
        JLabel labelNome;  
        labelNome = new JLabel("  Entre com o seu nome: ");  
        labelNome.setForeground(Color.yellow);  
  
        this.add(labelNome);  
  
        this.setSize(800,500);  
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        Color color = new Color(0x005588);  
        this.getContentPane().setBackground(color);  
  
        this.setLocationRelativeTo(null);  
  
        this.setVisible(true);  
    }  
    public static void main(String[] args) {  
        JFrame_01 app = new JFrame_01();  
    }  
}
```



# Acrescentando label



# Gerenciamento de Layout

- À medida em que vamos inserindo componentes no **Frame**, será necessário um modo de gerenciar estes componentes na tela e assim darmos funcionalidade à interface.
- Com isso, iremos organizar os componentes no **layout**, posicionando-os de forma mais adequada na interface.



# Como implementar um gerenciamento de Layout ?





```
package br.uscs;
```

```
import java.awt.*;
```

```
import javax.swing.*;
```

```
public class JFrame_01 extends JFrame {
```

```
    public JFrame_01() {
```

```
        super("USCS-Programação Orientada a Objetos");
```

```
        FlowLayout layout = new FlowLayout();
```

```
        this.setLayout(layout);
```

```
        JLabel labelNome;
```

```
        labelNome = new JLabel("  Entre com o seu nome: ");
```

```
        labelNome.setForeground(Color.yellow);
```

```
        this.add(labelNome);
```

```
        JTextField textFieldNome = new JTextField(30);
```

```
        this.add(textFieldNome);
```

```
        this.setSize(800,500);
```

```
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
        Color color = new Color(0x005588);
```

```
        this.getContentPane().setBackground(color);
```

```
        this.setLocationRelativeTo(null);
```

```
        this.setVisible(true);
```

```
    }
```

```
    public static void main(String[] args) {
```

```
        JFrame_01 app = new JFrame_01();
```

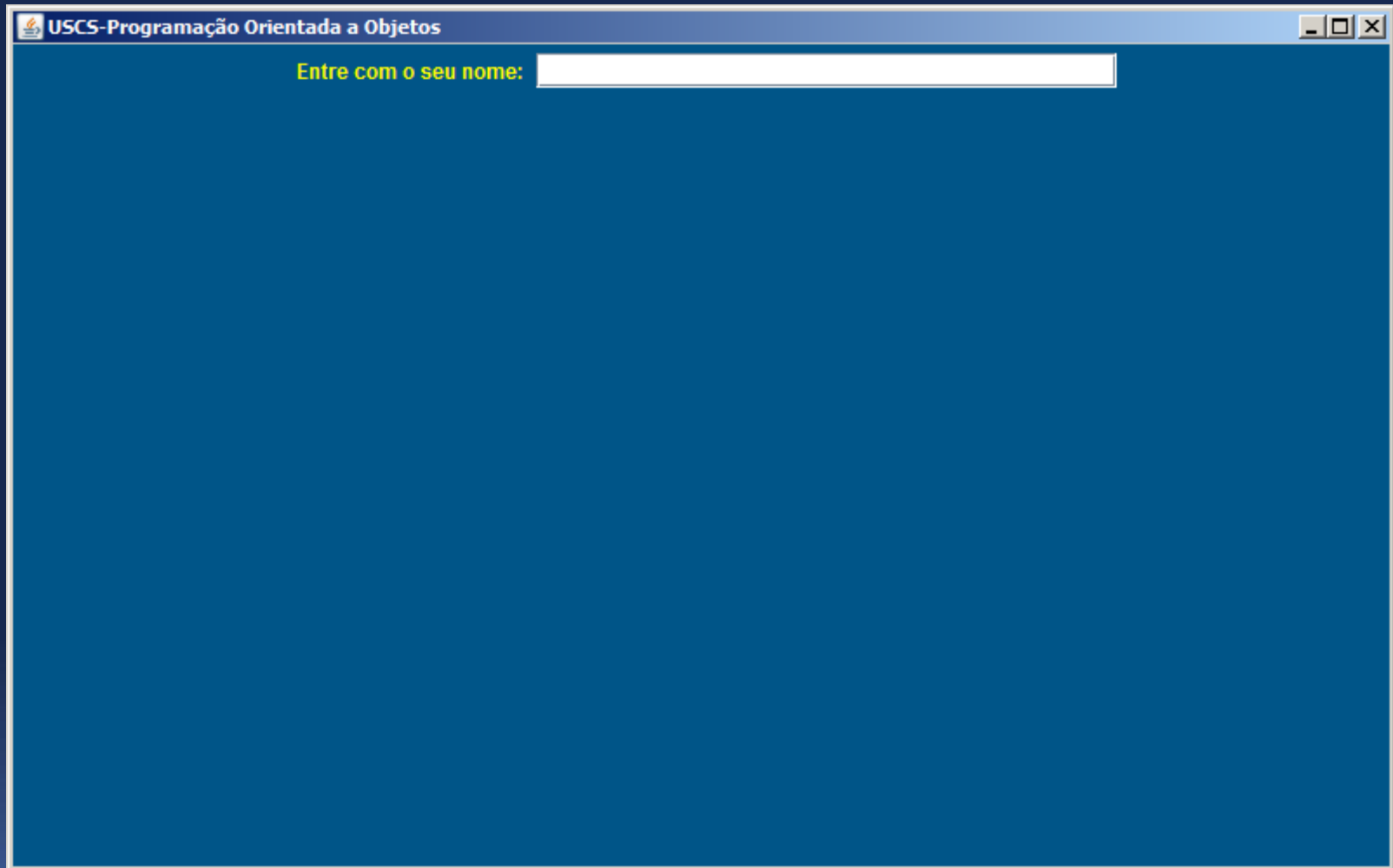
```
    }
```

```
}
```

# Gerenciamento de Layout



# Gerenciamento de Layout



USCS-Programação Orientada a Objetos

Entre com o seu nome:

# Como acrescento um botão à janela ?



# Incluindo um botão



```
package br.uscs;

import java.awt.*;
import javax.swing.*;

public class JFrame_01 extends JFrame {
    public JFrame_01() {

        super("USCS-Programação Orientada a Objetos");
        FlowLayout layout = new FlowLayout();
        this.setLayout(layout);

        JLabel labelNome;
        labelNome = new JLabel("  Entre com o seu nome: ");
        labelNome.setForeground(Color.yellow);
        this.add(labelNome);

        JTextField textFieldNome = new JTextField(30);
        this.add(textFieldNome);
    }
}
```



# Incluindo um botão



```
    JButton botao_OK = new JButton(" OK ");
    this.add(botao_OK);

    this.setSize(800,500);
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    Color color = new Color(0x005588);
    this.getContentPane().setBackground(color);

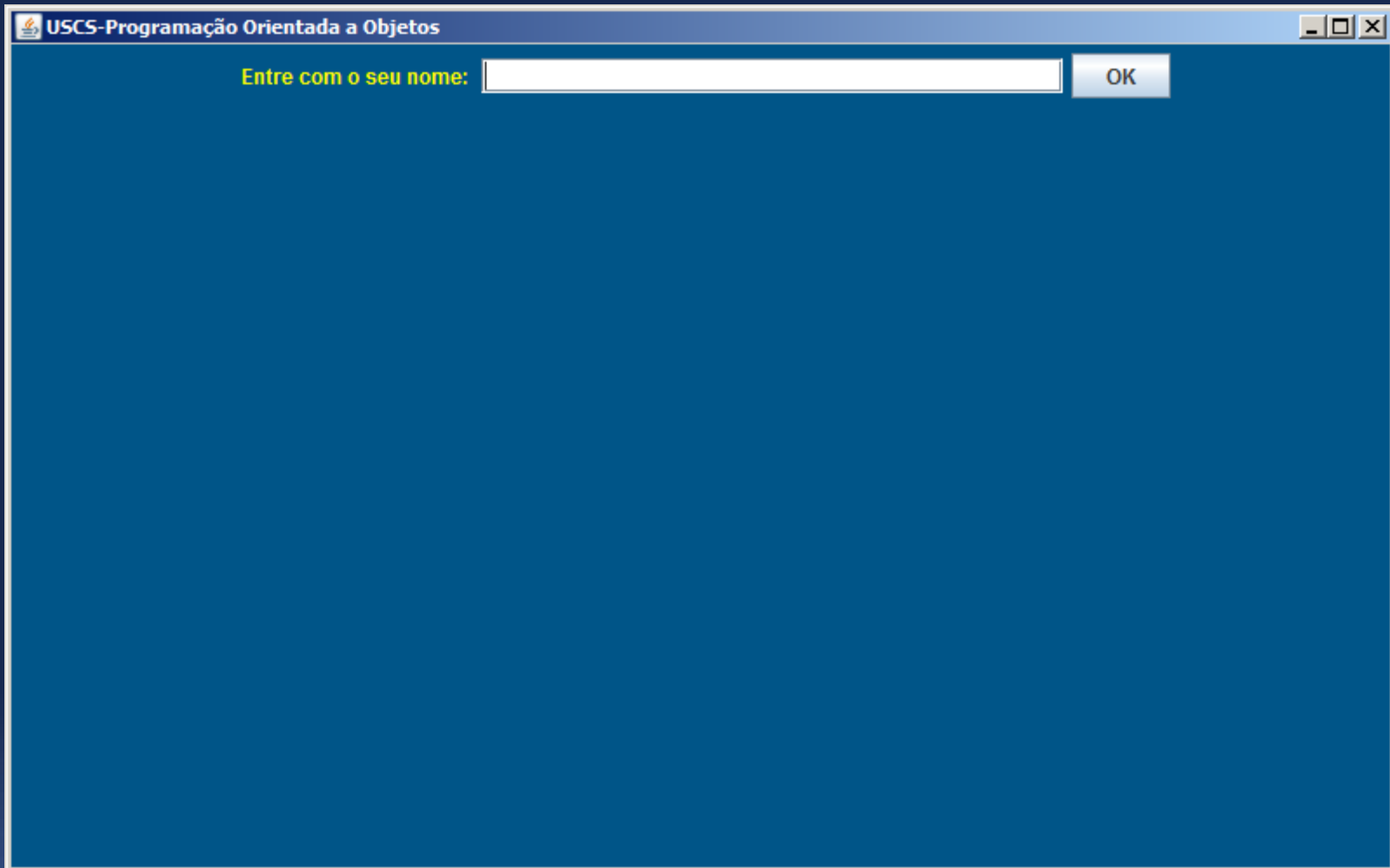
    this.setLocationRelativeTo(null);

    this.setVisible(true);
}

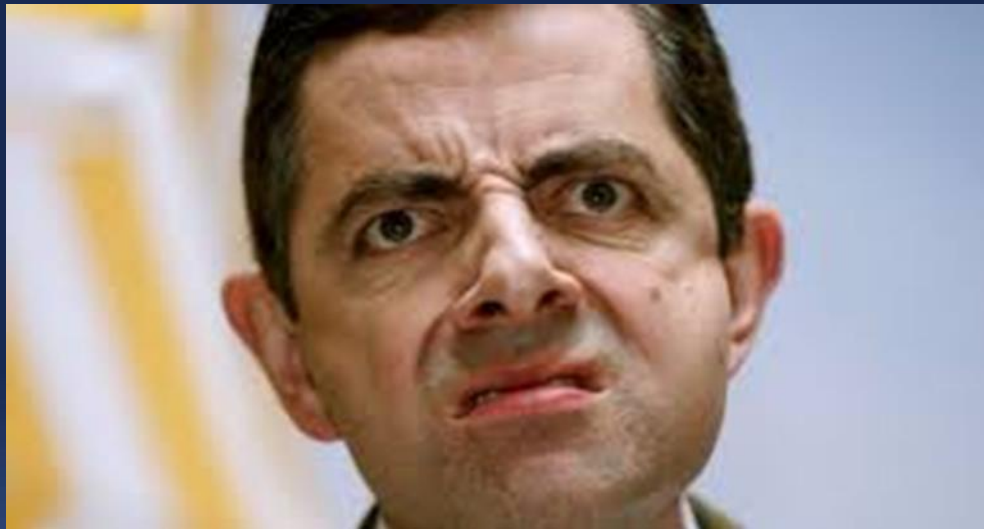
public static void main(String[] args) {
    JFrame_01 app = new JFrame_01();
}
}
```



# Incluindo um botão

A screenshot of a Java Swing window titled "USCS-Programação Orientada a Objetos". The window has a blue background. At the top, there is a text label "Entre com o seu nome:" in yellow. To the right of the label is a white text input field. Further to the right is a light gray button with the text "OK" in black. The window has standard OS window controls (minimize, maximize, close) in the top right corner.

# Porque ao clicar no botão nada acontece ?



# Tratamento de Eventos

- Para implementarmos uma funcionalidade no botão precisamos gerar o código que trate o **evento** de **clique** do usuário no componente.
- Iremos implementar uma classe responsável pelo tratamento dos eventos no botão.
- Esta classe implementa funções da interface **ActionListener**.





# Tratamento de Eventos



```
package br.uscs;

import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

public class JFrame_01 extends JFrame {

    private JLabel labelNome;
    private JTextField textFieldNome;
    private FlowLayout layout;
    private JButton botao_OK;

    public JFrame_01() {
        super("USCS - Lab V - Frame centralizado");
        this.layout = new FlowLayout();
        this.setLayout(layout);
    }
}
```



# Tratamento de Eventos



```
labelNome = new JLabel(" Entre com o seu nome: ");
labelNome.setForeground(Color.yellow);
this.add(labelNome);

this.textFieldName = new JTextField(30);
this.add(textFieldName);
this.botao_OK = new JButton(" OK ");
this.add(botao_OK);

ButtonHandler handler = new ButtonHandler();
botao_OK.addActionListener(handler);
this.setSize(800,500);
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
Color color = new Color(0x005588);
this.getContentPane().setBackground(color);

this.setLocationRelativeTo(null);

this.setVisible(true);

}
```

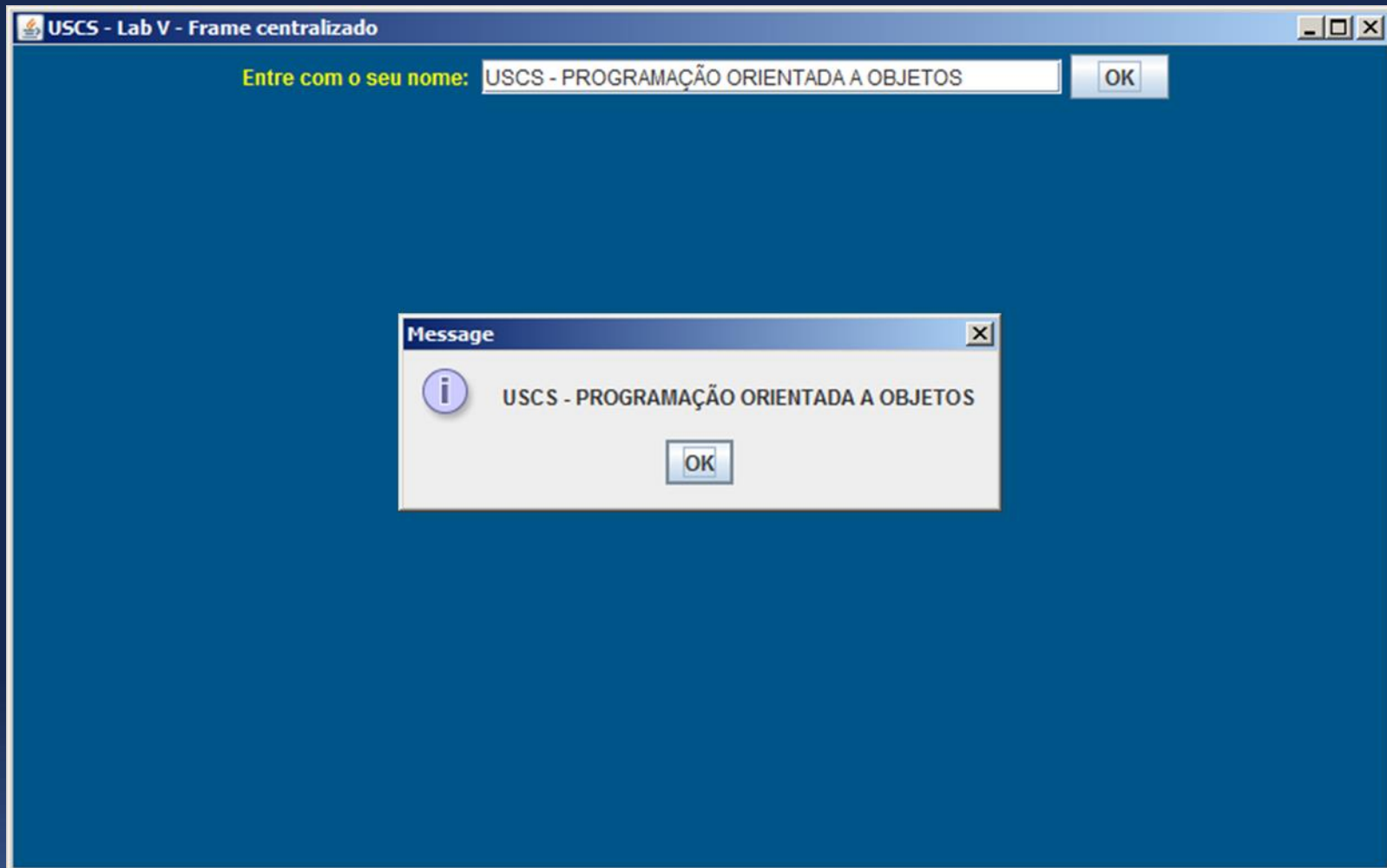


# Tratamento de Eventos

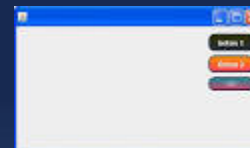
```
class ButtonHandler implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent event) {
        if (event.getSource() == botao_OK) {
            String nome = textFieldNome.getText();
            JOptionPane.showMessageDialog(null, nome);
        }
    }
}

public static void main(String[] args) {
    JFrame_01 app = new JFrame_01();
}
```

# Tratando o evento do botão



# Exemplo com 2 botões



# Exemplo com 2 botões



```
package br.uscs;
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

public class JFrame_01 extends JFrame {

    private JLabel labelNome;
    private JTextField textFieldNome;
    private FlowLayout layout;
    private JButton botao_OK;
    private JButton botao_FIM;

    public JFrame_01() {
        super("USCS - Programação Orientada a Objetos");
        this.layout = new FlowLayout();
        this.setLayout(layout);

        labelNome = new JLabel("  Entre com o seu nome: ");
        labelNome.setForeground(Color.yellow);
        this.add(labelNome);
```



# Exemplo com 2 botões



```
this.textFieldName = new JTextField(30);
this.add(textFieldName);

this.botao_OK = new JButton(" OK ");
this.add(botao_OK);

this.botao_FIM = new JButton(" FIM ");
this.add(botao_FIM);

ButtonHandler handler = new ButtonHandler();

botao_OK.addActionListener(handler);
botao_FIM.addActionListener(handler);

this.setSize(800,500);
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
Color color = new Color(0x005588);
this.getContentPane().setBackground(color);

this.setLocationRelativeTo(null);

this.setVisible(true);

}
```



# Exemplo com 2 botões



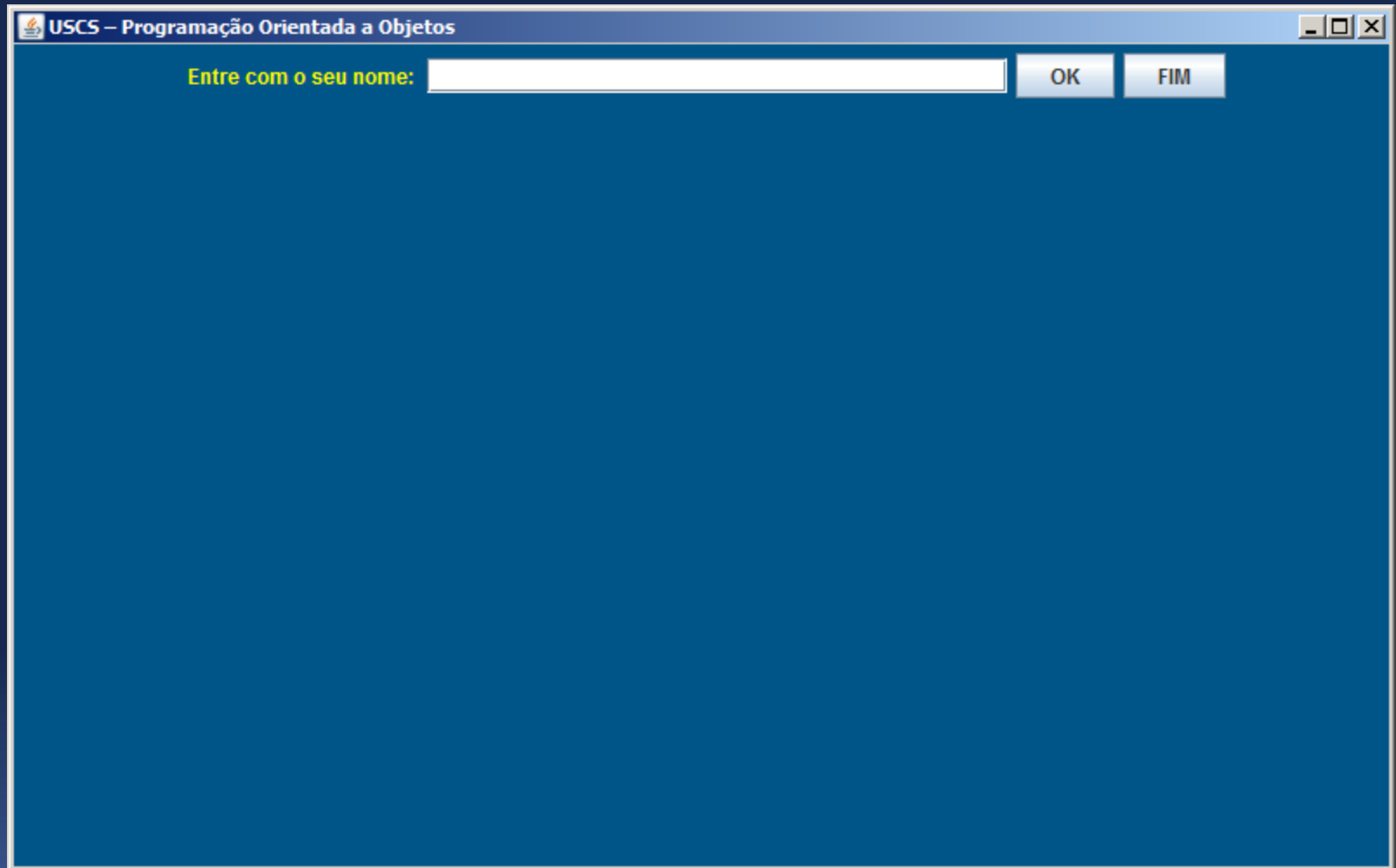
```
class ButtonHandler implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent event) {
        if (event.getSource() == botao_OK) {
            String nome = textFieldNome.getText();
            JOptionPane.showMessageDialog(null, nome);
        }
        else if (event.getSource() == botao_FIM) {
            System.out.println("Botao FIM pressionado....");
            System.out.println("Fim de programa....");
            System.exit(NORMAL);
        }
    }
}

public static void main(String[] args) {
    JFrame_01 app = new JFrame_01();
}
}
```





# Exemplo com 2 botões

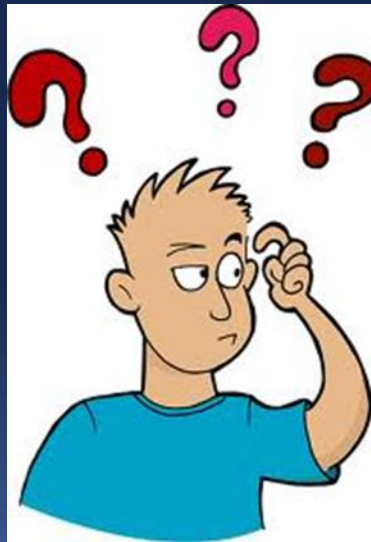


USCS – Programação Orientada a Objetos

Entre com o seu nome:

OK FIM

# Porque o código implementa ActionListener ?



# ActionListener

- A classe escrita implementa a interface **ActionListener**.
- Um **listener**, como o nome indica, monitora os eventos no programa e permite que seja especificada alguma ação a ser tomada quando o evento ocorrer.
- No nosso caso, queremos que o programa encerre quando o usuário escolher **Exit** no menu.
- Java oferece diversos listeners tais como: mouse, teclado, etc., todos eles estendendo a interface **EventListener**.

