- **UESC - Universidade Estadual de Santa Cruz**

Cap 5 – Interface Gráfica e Collections

Parte 1 – Interface Gráfica I

Disciplina: Linguagem de Programação III
Professor: Otacílio José Pereira

Plano de Aula

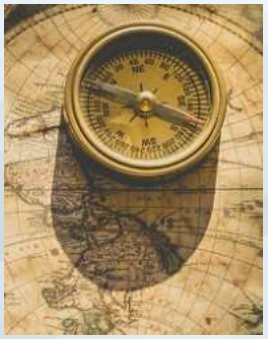
- ***Objetivos***
  - Iniciar com os primeiros conceitos de Interface Gráfica
  - Passo a passo entender os elementos com compõem a solução da biblioteca Swing para Interface Gráfica
- ***Tópicos***
  - Passo 1: Aquecendo com JOptionPane
  - Passo 2: Primeira Janela
  - Passo 3: Controles
  - Passo 4: Gerenciamento de Layout
  - Passo 5: Outros controles de tela
  - Passo 6: Eventos

Contexto

- Onde estamos?
- Foco agora!
- Cenário de exemplo

Onde estamos?

- *Considerando nosso planejamento inicial*



- Capítulo 1 – Introdução
- Capítulo 2 - Conceitos básicos de Orientação a Objetos
- Capítulo 3 – Herança
  - Parte 1 – Herança
  - Parte 2 - Polimorfismo
- Capítulo 4 – Classes abstratas e interfaces
- Capítulo 5 – Interface Gráfica e Collections
- Capítulo 6 – Desenvolvimento de um projeto em OO

Foco

Passo 1: Aquecendo!

- Observe e execute o código
- Pacote Swing
- JOptionPane
- Exercício

Observe e execute!

- Observe o código a seguir!
  - O que ele está fazendo?
  - Quais elementos desconhecidos?
  - O que você suspeita que eles estão fazendo?

```
11  public class ExemploUI {
12
13  public static void main(String[] args)
14  {
15
16  String strn1 = JOptionPane.showInputDialog("Primeiro inteiro");
17  String strn2 = JOptionPane.showInputDialog("Segundo inteiro");
18
19  int n1 = Integer.parseInt(strn1);
20  int n2 = Integer.parseInt(strn2);
21
22  int s = n1 + n2;
23
24  // exibe o resultado em um diálogo de mensagem JOptionPaneÿ
25  JOptionPane.showMessageDialog(null, "A soma é " + s,
26  "Sum of Two Integers", JOptionPane.PLAIN_MESSAGE);
27  }
28  }
```

Tópicos

- Pacote Swing
  - Uma das formas de se produzir interfaces, no caso Desktop, é por meio dos elementos que existem no pacote Swing
- JOptionPane
  - Observer que um primeiro meio de interagir com o usuário já com algum estímulo gráfico é por meio do JOptionPane
  - Método ShowInputDialog
    - Permitiu capturar a string digitada no campo disponibilizado
  - Método ShowMessageDialogo
    - Apenas apresente uma mensagem

```
11  public class ExemploUI {
12
13  public static void main(String[] args)
14  {
15
16  String strn1 = JOptionPane.showInputDialog("Primeiro inteiro");
17  String strn2 = JOptionPane.showInputDialog("Segundo inteiro");
18
19  int n1 = Integer.parseInt(strn1);
20  int n2 = Integer.parseInt(strn2);
21
22  int s = n1 + n2;
23
24  // exibe o resultado em um diálogo de mensagem JOptionPaneÿ
25  JOptionPane.showMessageDialog(null, "A soma é " + s,
26  "Sum of Two Integers", JOptionPane.PLAIN_MESSAGE);
27  }
28  }
```

Exercício

- Exercício 1:
  - Faça um ajuste na Interface para que ele capture 3 números
- Exercício 2:
  - Permita agora que ele capture números float
- Exercício 3
  - Investigue quais outras formas de executar os métodos ShowMessage e ShowInputDialogo

```
11  public class ExemploUI {
12
13  public static void main(String[] args)
14  {
15
16  String strn1 = JOptionPane.showInputDialog("Primeiro inteiro");
17  String strn2 = JOptionPane.showInputDialog("Segundo inteiro");
18
19  int n1 = Integer.parseInt(strn1);
20  int n2 = Integer.parseInt(strn2);
21
22  int s = n1 + n2;
23
24  // exibe o resultado em um diálogo de mensagem JOptionPaneÿ
25  JOptionPane.showMessageDialog(null, "A soma é " + s,
26  "Sum of Two Integers", JOptionPane.PLAIN_MESSAGE);
27  }
28  }
```

Tópicos

- Pacote Swing
  - Uma das formas de se produzir interfaces, no caso Desktop, é por meio dos elementos que existem no pacote Swing
- JOptionPane
  - Observer que um primeiro meio de interagir com o usuário já com algum estímulo gráfico é por meio do JOptionPane
  - Método ShowInputDialog
    - Permitiu capturar a string digitada no campo disponibilizado
  - Método ShowMessageDialogo
    - Apenas apresente uma mensagem

```
11  public class ExemploUI {
12
13  public static void main(String[] args)
14  {
15
16  String strn1 = JOptionPane.showInputDialog("Primeiro inteiro");
17  String strn2 = JOptionPane.showInputDialog("Segundo inteiro");
18
19  int n1 = Integer.parseInt(strn1);
20  int n2 = Integer.parseInt(strn2);
21
22  int s = n1 + n2;
23
24  // exibe o resultado em um diálogo de mensagem JOptionPaneÿ
25  JOptionPane.showMessageDialog(null, "A soma é " + s,
26  "Sum of Two Integers", JOptionPane.PLAIN_MESSAGE);
27  }
28  }
```

Passo 2: Janelas

- Observe e execute o código
- JFrame
- Elo com conceitos de OO
- Exercícios

Observe e execute!

- Observe o código a seguir!
  - Há alguma herança?
  - O que você acha que pode ser a Jframe?
  - O que cada linha está fazendo?

```
7  import javax.swing.JFrame;
8
9  public class ExemploUI extends JFrame{
10
11     public static void main(String[] args)
12     {
13         JFrame f = new ExemploUI();
14         f.setTitle("Primeira Janela");
15         f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16         f.setBounds(300, 300, 400, 200);
17         f.setVisible(true);
18     }
19 }
```

  - O que aconteceria se a linha 17 fosse suprimida? Haveria o objeto Janela?

Tópicos

- Classe JFrame
  - A classe JFrame (do Swing) é uma implementação de janela em programas desktop do Windows
- Conceitos de Orientação a Objetos
  - O que são classes, objetos, atributos e métodos no programa
  - Percebe que com poucas linhas pode-se criar uma janela
  - Como a herança ajudou neste caso?

```
7  import javax.swing.JFrame;
8
9  public class ExemploUI extends JFrame{
10
11     public static void main(String[] args)
12     {
13         JFrame f = new ExemploUI();
14         f.setTitle("Primeira Janela");
15         f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16         f.setBounds(300, 300, 400, 200);
17         f.setVisible(true);
18     }
19 }
```

Exercício

- Exercício 1:
  - Altere o código para uma nova posição e uma nova dimensão da Janela
- Exercício 2:
  - Apresente como você criaria um construtor para a Janela

```
7  import javax.swing.JFrame;
8
9  public class ExemploUI extends JFrame{
10
11     public static void main(String[] args)
12     {
13         JFrame f = new ExemploUI();
14         f.setTitle("Primeira Janela");
15         f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16         f.setBounds(300, 300, 400, 200);
17         f.setVisible(true);
18     }
19 }
```

Passo 3: Controles

- Observe e execute o código
- Composição de uma tela
- Exemplos de controles
- Exercícios

Observe e execute!

- Qual novidade existe no código?
- Como traduzir label do inglês para o português?
- O que você acha que o “add” está fazendo?

```
11  public class ExemploUI extends JFrame{
12
13      JLabel lblHello;
14
15      ExemploUI()
16      {
17          lblHello = new JLabel();
18          lblHello.setText("Olá, beleza?");
            add(lblHello);
20      }
21
22      public static void main(String[] args)
23      {
24          JFrame f = new ExemploUI();
25          f.setTitle("Primeira Janela");
26          f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
27          f.setBounds(300, 300, 400, 200);
28          f.setVisible(true);
29      }
30  }
```

Tópicos

- Controles
  - Os controles ou controls são os elementos que compõem a interface
  - Eles que são responsáveis por conter as informações que são manipuladas na interface
  - No caso do Jlabel ele apenas apresenta dados (saída), não serve para capturar alguma informação do usuário
- JFrame (e o ContentPane)
  - Toda janela possui um componente intermediário, um container que abriga os controles.
  - Add adiciona os controles na Janela

```
11  public class ExemploUI extends JFrame{
12
13      JLabel lblHello;
14
15      ExemploUI()
16      {
17          lblHello = new JLabel();
18          lblHello.setText("Olá, beleza?");
            add(lblHello);
20      }
21
22      public static void main(String[] args)
23      {
24          JFrame f = new ExemploUI();
25          f.setTitle("Primeira Janela");
26          f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
27          f.setBounds(300, 300, 400, 200);
28          f.setVisible(true);
29      }
30  }
```

Exercício

- Exercício
  - Experimente criar mais dois labels e adicionar na interface
  - Se quiser, pode adicionar alguns outros controles.

```
11  public class ExemploUI extends JFrame{
12
13      JLabel lblHello;
14
15      ExemploUI()
16      {
17          lblHello = new JLabel();
18          lblHello.setText("Olá, beleza?");
            add(lblHello);
20      }
21
22      public static void main(String[] args)
23      {
24          JFrame f = new ExemploUI();
25          f.setTitle("Primeira Janela");
26          f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
27          f.setBounds(300, 300, 400, 200);
28          f.setVisible(true);
29      }
30  }
```

Passo 4: Layout

- Observe e execute o código
- Layout
- Gerenciadores de Layout
- Exercícios

Observe e execute!

- E agora, qual a novidade a ser discutida
- Percebam que existem dois códigos relacionados com Layout
- Alterne a execução com um grupo de comandos ou de outro e avalie como a interface fica

```
12  public class ExemploUI extends JFrame{
13      JLabel lblHello;
14
15      ExemploUI()
16      {
17          lblHello = new JLabel();
18          lblHello.setText("Olá, beleza?");
19
20          //setLayout(new FlowLayout());
21          //add(lblHello);
22
            setLayout(new GridLayout(3, 2));
            add(lblHello);
            add(new JLabel("Label 2"));
            add(new JLabel("Label 3"));
            add(new JLabel("Label 4"));
            add(new JLabel("Label 5"));
29      }
```

Tópicos

- Gerenciamento de Layout
  - Um gerenciador de Layout é responsável por organizar a disposição dos controles na interface
  - Existe diversos tipos, estamos usando apenas dois no caso o FlowLayout: que coloca um elemento após o outro como um fluxo
  - GridLayout: organiza a interface como um grid (uma matriz).

```
12  public class ExemploUI extends JFrame{
13      JLabel lblHello;
14
15      ExemploUI()
16      {
17          lblHello = new JLabel();
18          lblHello.setText("Olá, beleza?");
19
20          //setLayout(new FlowLayout());
21          //add(lblHello);
22
            setLayout(new GridLayout(3, 2));
            add(lblHello);
            add(new JLabel("Label 2"));
            add(new JLabel("Label 3"));
            add(new JLabel("Label 4"));
            add(new JLabel("Label 5"));
29      }
```

Exercício

- Exercício 1
  - Experimente adicionar outros elementos e avaliar como o layout organiza os componentes
- Exercício 2
  - Pesquise na Internet quais outros gerenciadores de layout que existem e experimente alguns para avaliar a disposição dos controles

```
12  public class ExemploUI extends JFrame{
13      JLabel lblHello;
14
15      ExemploUI()
16      {
17          lblHello = new JLabel();
18          lblHello.setText("Olá, beleza?");
19
20          //setLayout(new FlowLayout());
21          //add(lblHello);
22
            setLayout(new GridLayout(3, 2));
            add(lblHello);
            add(new JLabel("Label 2"));
            add(new JLabel("Label 3"));
            add(new JLabel("Label 4"));
            add(new JLabel("Label 5"));
29      }
```

Passo 5: Outros Controles

- Observe e execute o código
- Jbutton
- Edição de tela no Netbeans
- Experimentando outros controles
- Exercícios

Observe e execute!

- Pensando o que você viu de gerenciamento de layout, como a tela a seguir ficará disposta.

- Quais outros tipos de controles que estão envolvidos?

```
14  public class ExemploUI extends JFrame{
15      JLabel      lblCodigo, lblNome;
16      JTextField  txtCodigo, txtNome;
17      JButton     btnSalvar, btnCancelar;
18      ExemploUI()
19      {
20          lblCodigo = new JLabel("Código");
21          txtCodigo = new JTextField();
22          lblNome = new JLabel("Nome");
23          txtNome = new JTextField();
24          btnSalvar   = new JButton("Salvar");
25          btnCancelar = new JButton("Cancelar");
26
            setLayout(new GridLayout(3, 2));
            add(lblCodigo);     add(txtCodigo);
            add(lblNome);       add(txtNome);
            add(btnSalvar);     add(btnCancelar);
31      }
```

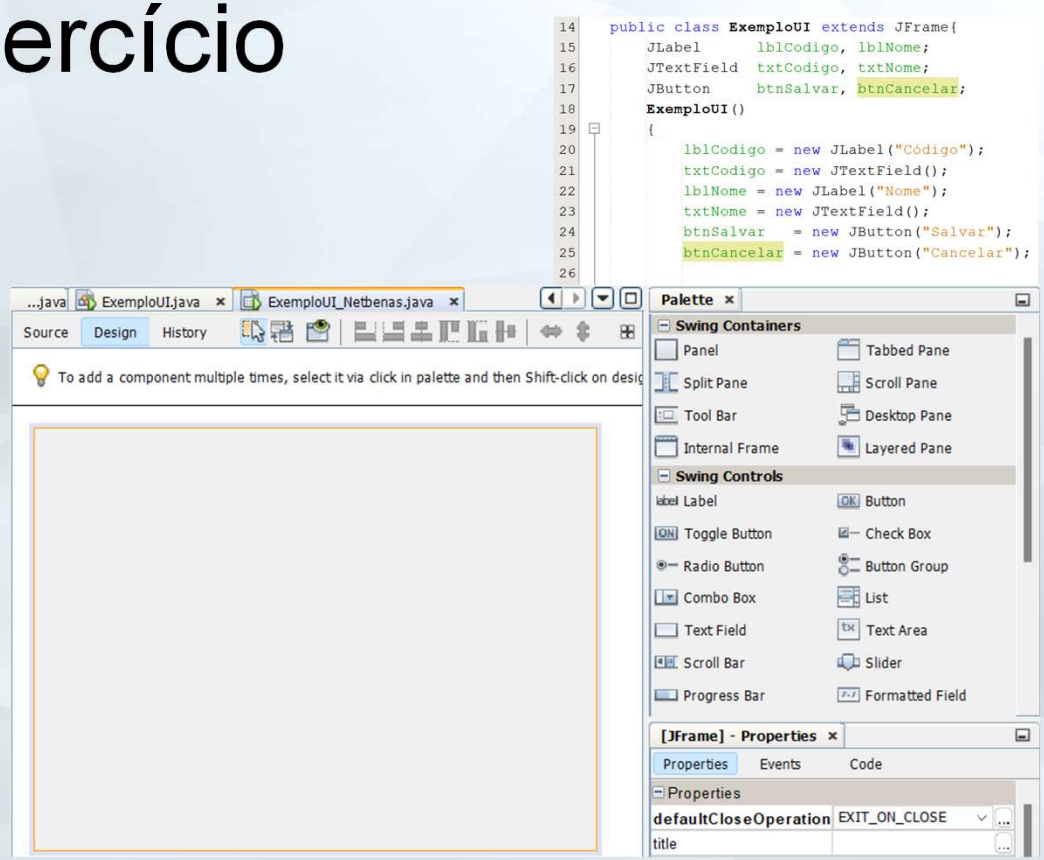
Tópicos

- Outros controles
  - Neste passo a ideia é compreender alguns outros elementos que existem para a produção de interface
  - Ainda estamos muito simples, trabalhando apenas com Label, TextoField e Button
  - Se você observar o NetBeans, você verá um conjunto bem mais amplos de elementos que pode compor sua interface.

```
14  public class ExemploUI extends JFrame{
15      JLabel      lblCodigo, lblNome;
16      JTextField  txtCodigo, txtNome;
17      JButton     btnSalvar, btnCancelar;
18      ExemploUI()
19      {
20          lblCodigo = new JLabel("Código");
21          txtCodigo = new JTextField();
22          lblNome = new JLabel("Nome");
23          txtNome = new JTextField();
24          btnSalvar   = new JButton("Salvar");
25          btnCancelar = new JButton("Cancelar");
26
            setLayout(new GridLayout(3, 2));
            add(lblCodigo);     add(txtCodigo);
            add(lblNome);       add(txtNome);
            add(btnSalvar);     add(btnCancelar);
31      }
```

Exercício

- Exercício 1
  - No Netbeans experimento criar um projeto com uma JFrame e ao tentar editá-la observe no NetBeans quais tipos de controles que existem
  - Experimento arrastar alguns para a interface e a trabalhar com eles
- Exercício 2
  - Experimente construir uma interface com estes diversos controles
  - Qual jeito mais fácil de produzir IU?
  - Qual vocÊ entende melhor o que está ocorrendo?



Passo 6: Eventos

- Observe e execute o código
- Eventos na IU
- Classe Handler
- Conceito de Listeners
- Exercícios

Observe e execute!

- Você conseguia realizar alguma coisa ao clicar nos botões da interface anterior?
- Quais pontos que você percebe de diferente?

```
ExemploUI()
{
    lblCodigo = new JLabel("Código");
    txtCodigo = new JTextField();
    lblNome = new JLabel("Nome");
    txtNome = new JTextField();
    btnSalvar   = new JButton("Salvar");
    btnCancelar = new JButton("Cancelar");

    setLayout(new GridLayout(3, 2));
    add(lblCodigo);     add(txtCodigo);
    add(lblNome);       add(txtNome);
    add(btnSalvar);     add(btnCancelar);

    EventoHandler handler = new EventoHandler();
    btnSalvar.addActionListener(handler);
    btnCancelar.addActionListener(handler);
}
```

```
51      // classe interna private para tratamento de evento
52      private class EventoHandler implements ActionListener
53      {
54          @Override
            public void actionPerformed(ActionEvent event)
56          {
57              JOptionPane.showMessageDialog(null, "Ops, houve um clique aqui!");
58          }
59      }
```

Tópicos

- Eventos
  - Os controles da interface vão além de apenas manipular as informações que são editadas pelo usuário
  - Controles disparam eventos, isto é, qualquer coisa que é realizada com ele é sinalizado para o sistema que ocorreu
  - Na maioria das vezes ele não tem ninguém que "o escute", ele dispara o evento mas sem resposta
  - Para realizar algo, é necessário criar uma classe Handler (manuseador) que escuta e responde os eventos

```
ExemploUI()
{
    lblCodigo = new JLabel("Código");
    txtCodigo = new JTextField();
    lblNome = new JLabel("Nome");
    txtNome = new JTextField();
    btnSalvar   = new JButton("Salvar");
    btnCancelar = new JButton("Cancelar");

    setLayout(new GridLayout(3, 2));
    add(lblCodigo);     add(txtCodigo);
    add(lblNome);       add(txtNome);
    add(btnSalvar);     add(btnCancelar);

    EventoHandler handler = new EventoHandler();
    btnSalvar.addActionListener(handler);
    btnCancelar.addActionListener(handler);
}
```

```
51      // classe interna private para tratamento de evento
52      private class EventoHandler implements ActionListener
53      {
54          @Override
            public void actionPerformed(ActionEvent event)
56          {
57              JOptionPane.showMessageDialog(null, "Ops, houve um clique aqui!");
58          }
59      }
```

Tópicos

- Handler
  - Perceba que na própria classe foi criada uma classe interna com o nome de EventoHandler
  - Ele tem um método “actionPerformed” que é o método executado quando percebermos um click do mouse
  - Ele será quem escuta o botão e dispara a ação
- Listener
  - Não basta criar a classe EventoHandler
  - Ele precisa está registrado nos controles para os quais escutará os eventos.
  - O addListener faz isso. Acompanhe no quadro de aula as explicações!

```
ExemploUI()
{
    lblCodigo = new JLabel("Código");
    txtCodigo = new JTextField();
    lblNome = new JLabel("Nome");
    txtNome = new JTextField();
    btnSalvar   = new JButton("Salvar");
    btnCancelar = new JButton("Cancelar");

    setLayout(new GridLayout(3, 2));
    add(lblCodigo);     add(txtCodigo);
    add(lblNome);       add(txtNome);
    add(btnSalvar);     add(btnCancelar);

    EventoHandler handler = new EventoHandler();
    btnSalvar.addActionListener(handler);
    btnCancelar.addActionListener(handler);
}
```

```
51      // classe interna private para tratamento de evento
52      private class EventoHandler implements ActionListener
53      {
54          @Override
            public void actionPerformed(ActionEvent event)
56          {
57              JOptionPane.showMessageDialog(null, "Ops, houve um clique aqui!");
58          }
59      }
```

Exercício

- Exercício
  - Experimente mostrar o JOptionPane a informação que foi digitada no campo de texto Nome.

```
ExemploUI()
{
    lblCodigo = new JLabel("Código");
    txtCodigo = new JTextField();
    lblNome = new JLabel("Nome");
    txtNome = new JTextField();
    btnSalvar   = new JButton("Salvar");
    btnCancelar = new JButton("Cancelar");

    setLayout(new GridLayout(3, 2));
    add(lblCodigo);     add(txtCodigo);
    add(lblNome);       add(txtNome);
    add(btnSalvar);     add(btnCancelar);

    EventoHandler handler = new EventoHandler();
    btnSalvar.addActionListener(handler);
    btnCancelar.addActionListener(handler);
}
```

```
51      // classe interna private para tratamento de evento
52      private class EventoHandler implements ActionListener
53      {
54          @Override
            public void actionPerformed(ActionEvent event)
56          {
57              JOptionPane.showMessageDialog(null, "Ops, houve um clique aqui!");
58          }
59      }
```

Conclusões

- Retomando Plano de Aula
- Revisão
- Para saber mais

Plano de Aula

- ***Objetivos***
  - Iniciar com os primeiros conceitos de Interface Gráfica
  - Passo a passo entender os elementos com compõem a solução da biblioteca Swing para Interface Gráfica
- ***Tópicos***
  - Passo 1: Aquecendo com JOptionPane
  - Passo 2: Primeira Janela
  - Passo 3: Controles
  - Passo 4: Gerenciamento de Layout
  - Passo 5: Outros controles de tela
  - Passo 6: Eventos