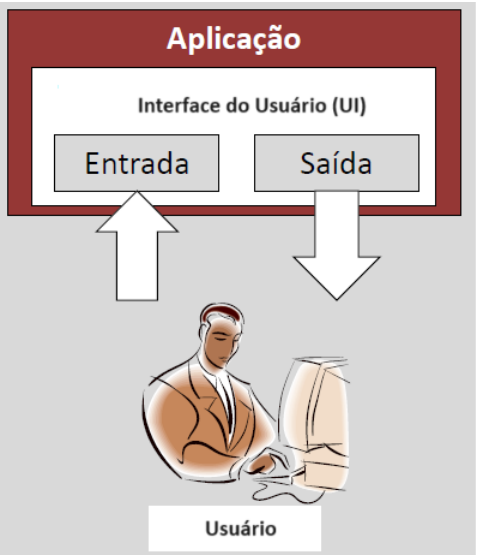
Roteiro 10

Neste roteiro trabalharemos com as **interfaces gráficas em Java - Swing**.

O objetivo deste roteiro é passar uma breve noção de como funcionam as interfaces gráficas com o usuário. Existem plataformas que facilitam a construção de interfaces automatizadas, mas não utilizaremos estes recursos. Faremos a construção das interfaces de forma manual. Utilizaremos recursos básicos, mas compreendendo a construção do código. Trabalharemos especificamente com a biblioteca gráfica Swing.

Como funciona a Interface com o Usuário – *User Interface* (UI)

Entendemos como UI o segmento de código que permitem os meios de interação com o usuário com *Inputs* e *Outputs.*



A Interface com o usuário pode ser de 2 formas :

- **Console** – Todos os roteiros anteriores utilizamos o console para fazer input e output
- **Gráfica (GUI – *Graphical User Interface*)** – neste caso toda a interação é feita com componentes gráficos como objetos de Janelas, Botões e etc.

Tipos de Objetos Gráficos

Para compreender melhor as interfaces gráficas em Java vamos separar os Objetos Gráficos em 3 grandes Grupos :

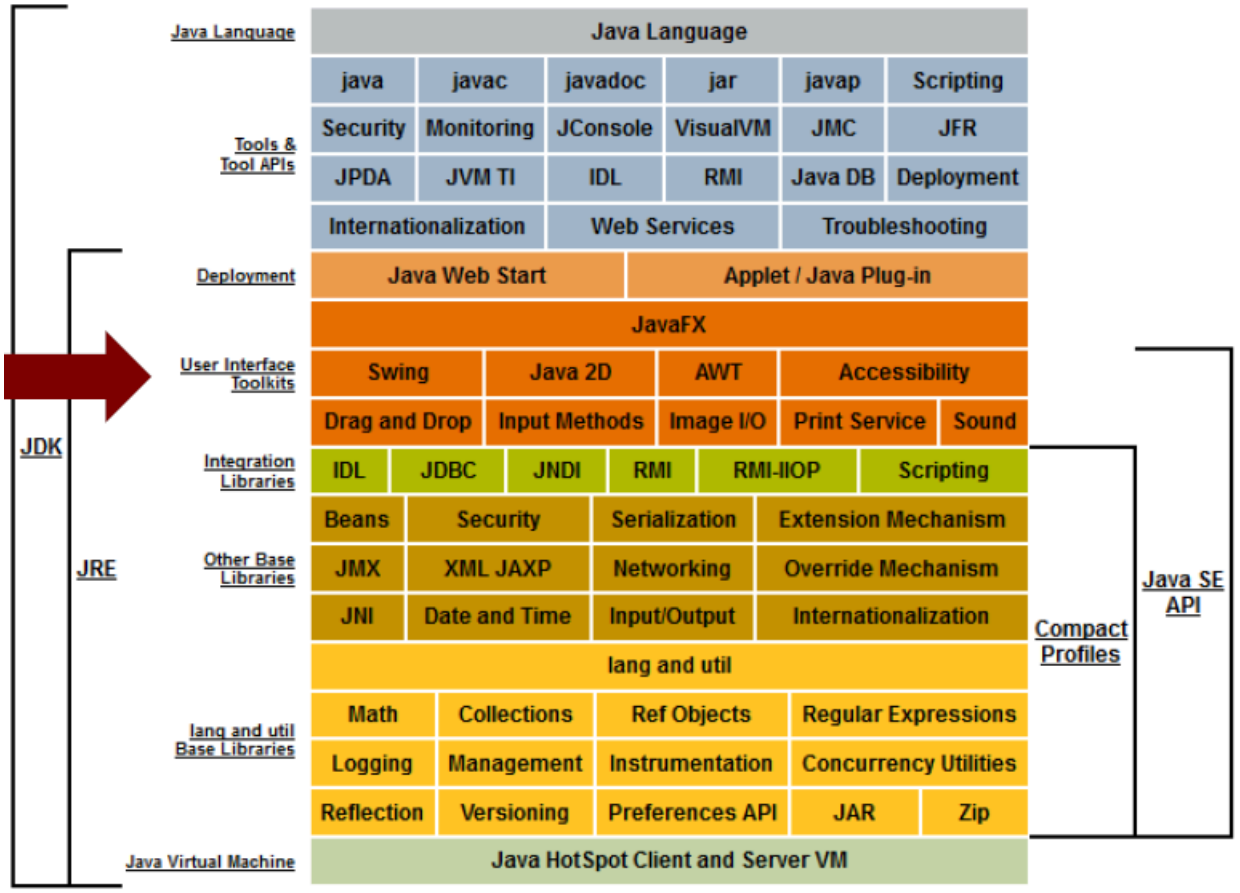
- Componentes Gráficos –
  - São os componentes visuais. Ex.: Janela, Botões, Caixa de Texto e etc.
- Gestores de Posicionamento (*Layout Managers*)
  - São responsáveis pelo dimensionamento das janelas e posicionamento de outros componentes visuais em tela.
- Manipuladores de Eventos (*Event Listeners*)
  - São responsáveis por responder aos eventos de solicitações do usuário em tela. Ex.: Cliques nos botões, movimentação do mouse, e outros eventos.

Toolkits Java para GUI

A principais bibliotecas gráficas são :

- AWT
  - Em geral trata da Gestão de Posicionamento e Manipulação de Eventos
- Swing
  - Em geral trata dos componentes visuais em tela
- Java 2D
  - Em geral trata de componentes gráficos 2D

Segue abaixo um gráfico nos situar no contexto de Interfaces Gráficas

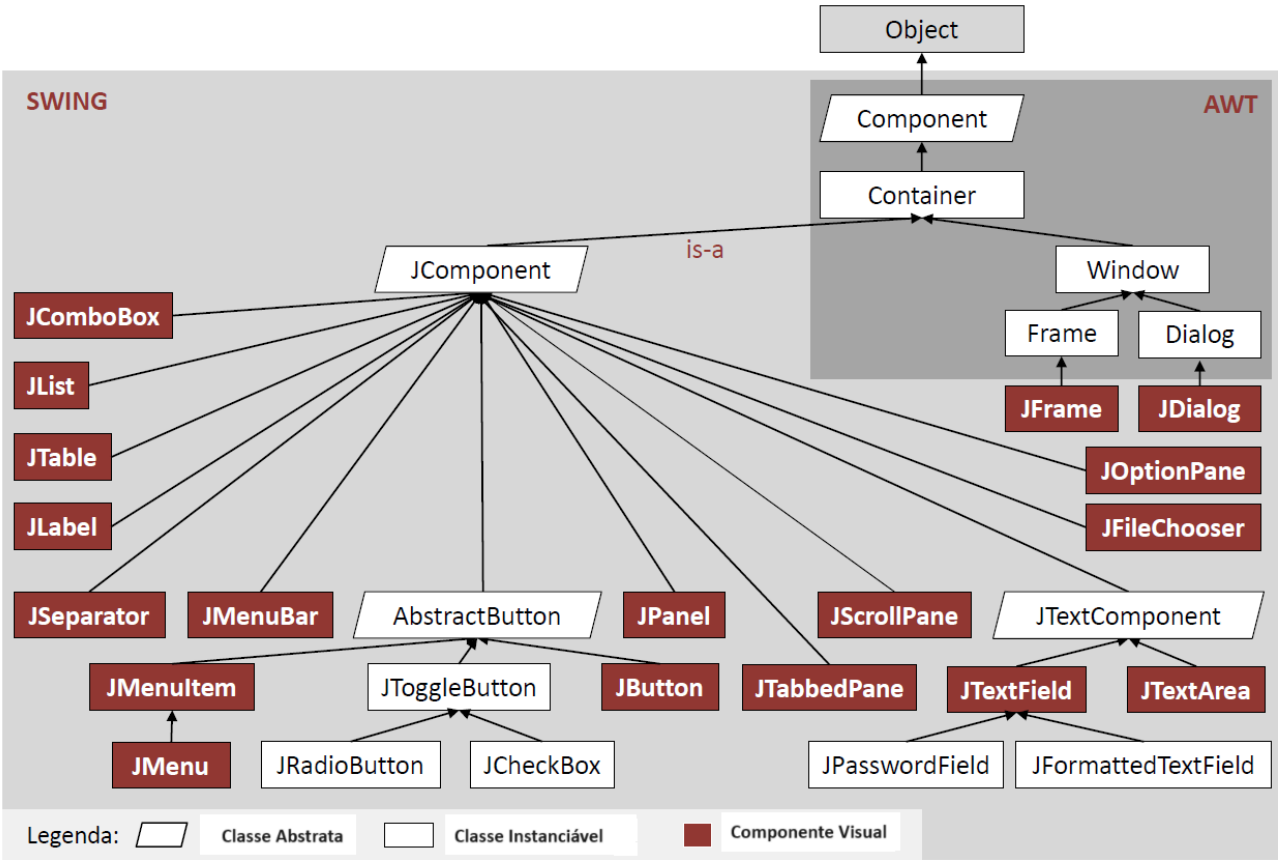


Hierarquia de Classes de Componentes Gráficos

Todas as bibliotecas e pacotes em Java seguem a POO e os conhecimentos adquiridos nos roteiros anteriores nos ajudam a compreender melhor o diagrama abaixo.

Veja que o diagrama abrange conceitos da OO como : Herança, Classes Abstratas, Interfaces.

Observe que todos os componentes visuais herdam de JComponent que por sua vez vem de Container – Component – Object.



Vamos para o roteiro

Parte 1 (roteiro10.parte1) – JFrame

1 – Crie o pacote **roteiro10.parte1** com a classes **Principal** contendo o método main.

2 – Crie a classe **Janela01** que herda de **JFrame**. Esta classe representa um componente visual do tipo "Janela". Para criar esta Janela utilizamos o conceito de herança, e dessa forma conseguimos utilizar todos os métodos e atributos da classe JFrame.

Veja que nossa classe possui apenas 2 atributos (largura e altura) para determinar as dimensões da nossa janela. Mas, estamos utilizando 3 métodos da classe JFrame :

`this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)` - seta a operação padrão que será executada ao fechar a janela. Neste caso, a aplicação será encerrada ao fechar a Janela.

`this.setSize(largura,altura)` - seta as dimensões de largura e altura da janela.

`this.setVisible(true)` - torna a janela visível.

```
import javax.swing.JFrame;

public class Janela01 extends JFrame{

    private int largura = 600;
    private int altura = 500;

    public Janela01()
    {
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(largura,altura);
        this.setVisible(true);
    }
}
```

3 – Para visualizar a tela em execução implemente o código abaixo na classe **Principal**. Novamente estamos usando um princípio de OO, pois estamos instanciando um objeto chamado janela01. Só que agora este objeto é visual.

Faça os testes rodando a classe Principal.

```
public class Principal {

    public static void main(String[] args) {

        Janela01 janela01 = new Janela01();

    }
}
```

4 – Crie uma nova janela com o nome da classe Janela02, semelhante ao que foi feito para Janela01. Mude apenas as dimensões para diferenciar uma Janela da outra.

```
import javax.swing.JFrame;

public class Janela02 extends JFrame{

    private int largura = 300;
    private int altura = 300;

    public Janela02()
    {
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(largura,altura);
        this.setVisible(true);
    }
}
```

5 – Teste na classe Principal a criação das duas janelas conforme o código abaixo.

Nas classes Janela01 e Janela02 teste os métodos `setTitle("Janela01");` e `setTitle("Janela02");`. Para diferenciar uma janela da outra pelo título da janela.

```
public class Principal {

    public static void main(String[] args) {

        Janela01 janela01 = new Janela01();

        Janela02 janela02 = new Janela02();

    }
}
```

6 – Vamos agora adicionar outros componentes visuais na Janela02. Crie 2 novos atributos do tipo JButton na classe Janela02 conforme o código abaixo.

Atenção para os seguintes itens no código :

- Importar o pacote JButton
- Definir os novos atributos do tipo JButton : btn01 e btn02
- Instanciar os objetos btn01 e btn02 dentro do construtor da Janela02
- Definir o posicionamento e as dimensões dos botões
- Adicionar os objetos ao container da Janela
- Setar o Layout da Janela – `this.setLayout(null);`

    - Este item merece um pouco mais de atenção neste momento. Observe que ele foi setado para null. **Ou seja, estamos assumindo a gestão do Layout dos componentes visuais em tela, e por isso, precisamos inserir manualmente as posições e dimensões dos botões** (`btn01.setBounds`).

    - Este item está relacionado ao grupo de objetos que chamamos de Gestores de Posicionamento (*Layout Managers*) que citamos no início do roteiro.

    - **Faça o seguinte teste** : Comente a linha `this.setLayout(null);` . A aplicação tentará controlar o layout dos componentes visuais em tela e ocorrerá uma sobreposição dos botões, além de ocupar todo o espaço da janela.

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

public class Janela02 extends JFrame{

    private int largura = 300;
    private int altura = 300;
    private JButton btn01;
    private JButton btn02;

    public Janela02()
    {
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(largura,altura);
        this.setTitle("Janela02");
        this.setLayout(null);

        btn01 = new JButton("Botão 01");
        btn02 = new JButton("Botão 02");

        // setando as dimensões dos botões
        // setBounds(posição x , posição y, largura e altura)
        btn01.setBounds(20, 50, 100, 40);
        btn02.setBounds(130, 50, 100, 40);

        this.add(btn01);
        this.add(btn02);

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

Parte 2 (roteiro10.parte2) – Gestores de Posicionamento (Layout Managers)

Os Gestores de Posicionamento (Layout Managers) são responsáveis pelo dimensionamento das janelas e posicionamento de outros componentes visuais em tela.

Na parte 1 do roteiro criamos uma janela com alguns componentes visuais em que o posicionamento e as dimensões dos componentes foram feitos de forma manual no código. Nesta parte faremos uso de alguns gerenciadores de layout que fazem este trabalho de forma automática. Segue alguns que veremos neste roteiro :

- **FlowLayout** – É um gerenciador de layout padrão que simplesmente dispõe os componentes visuais em uma única linha, iniciando uma nova linha se o container não for suficientemente largo.

- **GridLayout** – Também é um gerenciador de layout que dispõe os componentes visuais em um certo número de linhas e colunas formando um grid com componentes de igual proporção.

- **BorderLayout** – Este gerenciador permite organizar os componentes visuais em cinco áreas diferentes : superior, inferir, esquerda, direita e centro

Existem vários outros gerenciadores de Layout, mas utilizaremos apenas estes como forma de aprendizado. Segue o link de referência com outros Layouts : https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html

1 – Crie o pacote **roteiro10.parte2** com a classe Principal conforme o código abaixo. Observe que estamos instanciando 4 diferentes objetos que representam 4 janelas com diferentes layouts. Precisamos criar as classes correspondentes para estes 3 tipos de janelas.

```
public class Principal {

    public static void main(String[] args) {

        JanelaSemLayout janela01 = new JanelaSemLayout();

        JanelaFlowLayout janela02 = new JanelaFlowLayout();

        JanelaGridLayout janela03 = new JanelaGridLayout();

        JanelaBorderLayout janela04 = new JanelaBorderLayout();
    }
}
```

2 – Crie as 4 classes conforme o código abaixo que representam diferentes janelas e seus respectivos layouts. (**JanelaSemLayout, JanelaFlowLayout, JanelaGridLayout, JanelaBorderLayout**). Observe que criamos um método chamado iniciarComponentes para facilitar o entendimento do código.

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

public class JanelaSemLayout extends JFrame{

    private int largura = 500;
    private int altura = 500;
    private JButton btn01;
    private JButton btn02;
    private JButton btn03;
    private JButton btn04;
    private JButton btn05;
    private JButton btn06;

    public JanelaSemLayout()
    {
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(largura,altura);
        this.setTitle("Janela Sem Layout");

        this.setLayout(null);

        this.iniciarComponentes();

        this.setVisible(true);
    }

    public void iniciarComponentes(){

        btn01 = new JButton("Botão 01");
        btn02 = new JButton("Botão 02");
        btn03 = new JButton("Botão 03");
        btn04 = new JButton("Botão 04");
        btn05 = new JButton("Botão 05");
        btn06 = new JButton("Botão 06");

        btn01.setBounds(20, 50, 90, 30);  // (x,y, largura, altura)
        btn02.setBounds(130, 50, 90, 30);  // (x,y, largura, altura)

        btn03.setBounds(30, 90, 90, 30);  // (x,y, largura, altura)
        btn04.setBounds(140, 90, 90, 30);  // (x,y, largura, altura)

        btn05.setBounds(40, 130, 90, 30);  // (x,y, largura, altura)
        btn06.setBounds(150, 130, 90, 30);  // (x,y, largura, altura)

        this.add(btn01);
        this.add(btn02);
        this.add(btn03);
        this.add(btn04);
        this.add(btn05);
        this.add(btn06);
    }

}
```

Observe que a dimensão da janela não importa quando estamos utilizando o gerenciador de layout. Por isso o comando `this.setSize(largura,altura);` foi retirado nas classes seguintes. A janela vai se ajustar conforme os componentes existentes e o comando responsável por esta ação é : `this.pack();`

```
import java.awt.FlowLayout;
import javax.swing.JButton;
import javax.swing.JFrame;

public class JanelaFlowLayout extends JFrame {

    private JButton btn01;
    private JButton btn02;
    private JButton btn03;
    private JButton btn04;
    private JButton btn05;
    private JButton btn06;

    public JanelaFlowLayout(){
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setTitle("Janela FlowLayout");

        this.setLayout(new FlowLayout());

        this.iniciarComponentes();

        this.pack();
        this.setVisible(true);
    }

    public void iniciarComponentes(){

        btn01 = new JButton("Botão 01");
        btn02 = new JButton("Botão 02");
        btn03 = new JButton("Botão 03");
        btn04 = new JButton("Botão 04");
        btn05 = new JButton("Botão 05");
        btn06 = new JButton("Botão 06");

        this.add(btn01);
        this.add(btn02);
        this.add(btn03);
        this.add(btn04);
        this.add(btn05);
        this.add(btn06);
    }
}
```

Observe que no caso do GridLayout conseguimos definir a quantidade de linhas e colunas da janela em que os componentes irão se organizar `new GridLayout(2,3);` . Neste caso temos 2 linhas e 3 colunas.

```
import java.awt.GridLayout;
import javax.swing.JButton;
import javax.swing.JFrame;

public class JanelaGridLayout extends JFrame {

    private JButton btn01;
    private JButton btn02;
    private JButton btn03;
    private JButton btn04;
    private JButton btn05;
    private JButton btn06;

    public JanelaGridLayout(){
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setTitle("Janela GridLayout");

        this.setLayout(new GridLayout(2,3));

        this.iniciarComponentes();

        this.pack();
        this.setVisible(true);
    }

    public void iniciarComponentes(){

        btn01 = new JButton("Botão 01");
        btn02 = new JButton("Botão 02");
        btn03 = new JButton("Botão 03");
        btn04 = new JButton("Botão 04");
        btn05 = new JButton("Botão 05");
        btn06 = new JButton("Botão 06");

        this.add(btn01);
        this.add(btn02);
        this.add(btn03);
        this.add(btn04);
        this.add(btn05);
        this.add(btn06);
    }
}
```

No caso do BorderLayout observe que temos a flexibilidade de colocar os componentes em diferentes áreas da janela (superior, inferir, esquerda, direita e centro).

```
import java.awt.BorderLayout;
import javax.swing.JButton;
import javax.swing.JFrame;

public class JanelaBorderLayout extends JFrame {

    private JButton btn01;
    private JButton btn02;
    private JButton btn03;
    private JButton btn04;
    private JButton btn05;

    public JanelaBorderLayout(){
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setTitle("Janela BorderLayout");

        this.setLayout(new BorderLayout());

        this.iniciarComponentes();

        this.pack();
        this.setVisible(true);


    }

    public void iniciarComponentes(){

        btn01 = new JButton("Botão 01");
        btn02 = new JButton("Botão 02");
        btn03 = new JButton("Botão 03");
        btn04 = new JButton("Botão 04");
        btn05 = new JButton("Botão 05");

        this.add(btn01, BorderLayout.PAGE_START);
        this.add(btn02, BorderLayout.PAGE_END);
        this.add(btn03, BorderLayout.LINE_START);
        this.add(btn04, BorderLayout.LINE_END);
        this.add(btn05, BorderLayout.CENTER);
    }

}
```

3 – Faça os testes executando a classe Principal.

Parte 3 (roteiro10.parte3) – Layout Managers

1 – Crie o pacote **roteiro10.parte3** com a classe Principal conforme o código abaixo. Observe que estamos instanciando apenas o objeto janela04 que trabalha com o BorderLayout.

```
public class Principal {

    public static void main(String[] args) {

        JanelaBorderLayout_v1 janela04 = new JanelaBorderLayout_v1();
    }
}
```

2 – Crie a classe **JanelaBorderLayout_v1** conforme o código abaixo. Nesta classe estamos utilizando o BorderLayout, mas antes de inserir os componentes estamos dividindo as áreas em painéis com o componente JPanel. O JPanel é um componente do tipo container que permite a inserção de outros componentes visuais nos ajudando na organização dos componentes em tela.

```
import java.awt.BorderLayout;
import java.awt.Color;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class JanelaBorderLayout_v1 extends JFrame{

    private JPanel pnCabecalho;
    private JPanel pnRodape;
    private JPanel pnEsquerda;
    private JPanel pnDireita;
    private JPanel pnCentro;

    public JanelaBorderLayout_v1(){
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setTitle("Janela BorderLayout");

        this.setLayout(new BorderLayout());
        this.iniciarComponentes();
        this.pack();
        this.setVisible(true);
    }

    public void iniciarComponentes(){
        this.pnCabecalho = new JPanel();
        this.pnRodape = new JPanel();
        this.pnEsquerda = new JPanel();
        this.pnDireita = new JPanel();
        this.pnCentro = new JPanel();

        this.pnCabecalho.setBackground(Color.red);
        this.pnRodape.setBackground(Color.blue);
        this.pnEsquerda.setBackground(Color.yellow);
        this.pnDireita.setBackground(Color.gray);
        this.pnCentro.setBackground(Color.green);

        this.add(pnCabecalho, BorderLayout.PAGE_START);
        this.add(pnRodape, BorderLayout.PAGE_END);
        this.add(pnEsquerda, BorderLayout.LINE_START);
        this.add(pnDireita, BorderLayout.LINE_END);
        this.add(pnCentro, BorderLayout.CENTER);
    }
}
```

Normalmente não é possível visualizar o componente JPanel e por isso colocamos deferentes cores para facilitar a identificação dos painéis

Faça os testes executando a classe Principal.

3 - Crie a classe **JanelaBorderLayout_v2** como cópia da classe JanelaBorderLayout_v1**.**

Teremos agora 2 versões da classe JanelaBorderLayout. Uma apenas com os painéis e a outra com vários outros componentes visuais.

Nesta nova classe adicione os métodos conforme o código abaixo. Eles são responsáveis por iniciar os respectivos painéis com seus componentes

Para testar lembre-se de adicionar a chamada dos métodos abaixo dentro do método já existente `iniciarComponentes()`

```
public void iniciarCabecalho(){

    this.pnCabecalho.setLayout(new FlowLayout(FlowLayout.LEFT));

    this.pnCabecalho.add(new JLabel("Curso"));
    this.pnCabecalho.add(new JTextField(10));

    this.pnCabecalho.add(new JLabel("Observações"));
    this.pnCabecalho.add(new JTextField(25));
}

public void iniciarRodape(){

    this.pnRodape.setLayout(new FlowLayout(FlowLayout.RIGHT));
    this.pnRodape.add(new JButton("Botão 04"));
    this.pnRodape.add(new JButton("Botão 05"));
}

public void iniciarCentro(){

    this.pnCentro.setLayout(new GridLayout(3,2));

    this.pnCentro.add(new JLabel("Matricula"));
    this.pnCentro.add(new JTextField());

    this.pnCentro.add(new JLabel("Nome"));
    this.pnCentro.add(new JTextField());

    this.pnCentro.add(new JLabel("Idade"));
    this.pnCentro.add(new JTextField());

}
```

4 – Faça os testes na classe Principal mudando a chamada da versão 1 para a versão 2 da janela BorderLayout. Na versão 2 foi usado apenas um tipo de layout ? Sim ou não ? Quais ?

Parte 4 (roteiro10.parte4) – Manipuladores de Eventos (Event Listeners)

As aplicações em geral precisam de alguns eventos para o funcionamento completo. Normalmente as GUIs são baseadas em eventos. Eventos são tarefas realizadas quando o usuário faz alguma interação com algum componente visual. Exemplo :

- O clique do mouse
- O mover do mouse
- Apertar alguma tecla
- Fechar a janela

Em Java temos uma interface **EventListener**, e a partir desta interface temos diversos outros tipos de eventos com suas especificidades. A primeira e mais comum delas é a **ActionListener** que iremos utilizar no roteiro.

Vamos ao roteiro !

1 – Crie o pacote **roteiro10.parte4** com a classe **Principal** e a classe **JanelaSemLayout** utilizada na parte 2 deste roteiro.

2 – Na classe **JanelaSemLayout ,** implemente a interface ActionListener conforme apresentado no código abaixo. Novamente estamos utilizando os conceitos de OO com o uso de uma interface já disponível no pacote Java.

OBS.: Ao adicionar `implements ActionListener` sua IDE apontará um erro indicando que os métodos desta interface deverão ser obrigatoriamente implementados na classe JanelaSemLayout

```
public class JanelaSemLayout extends JFrame implements ActionListener{

    private int largura = 500;
    private int altura = 500;
    private JButton btn01;
    private JButton btn02;
    private JButton btn03;
    private JButton btn04;
    private JButton btn05;
    private JButton btn06;

    public JanelaSemLayout()
    {
        {Código do Construtor}
    }

    public void iniciarComponentes(){
        {Código para iniciar os componentes}
    }

}
```

3 – Para resolver o problema indicado no item 2, adicione o método actionPerformed na classe JanelaSemLayout conforme o código abaixo

OBS.: o método abaixo foi gerado automaticamente pela IDE. Ou seja, estamos fazendo o override do método, mas ainda não implementamos o método propriamente dito.

Observe também que ao concluir estas etapas, possivelmente os imports abaixo foram adicionados na sua classe.

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

```
    @Override
    public void actionPerformed(ActionEvent e) {
        throw new UnsupportedOperationException("Not supported yet.");
    }
```

Implemente o método conforme o código abaixo

```
    @Override
    public void actionPerformed(ActionEvent e) {
        System.out.println("o clique do mouse foi acionado")
    }
```

4 – Para que os eventos ocorridos no formulário sejam “percebidos” ou “ouvidos” pelo botão, adicione o comando abaixo.

```
public class JanelaSemLayout extends JFrame implements ActionListener{

    private int largura = 500;
    private int altura = 500;
    private JButton btn01;
    private JButton btn02;
    private JButton btn03;
    private JButton btn04;
    private JButton btn05;
    private JButton btn06;

    public JanelaSemLayout()
    {
        {Código do Construtor}
    }

    public void iniciarComponentes(){
        {Código para iniciar os componentes}

        btn01.addActionListener(this);
    }

}
```

5 – Faça os devidos testes na classe Principal. Todos os botões acionam o evento ?

6 – O que acontece se adicionarmos o trecho de código abaixo ao repetir os testes do item 5 ?

```
public class JanelaSemLayout extends JFrame implements ActionListener{

    private int largura = 500;
    private int altura = 500;
    private JButton btn01;
    private JButton btn02;
    private JButton btn03;
    private JButton btn04;
    private JButton btn05;
    private JButton btn06;

    public JanelaSemLayout()
    {
        {Código do Construtor}
    }

    public void iniciarComponentes(){
        {Código para iniciar os componentes}

        btn01.addActionListener(this);
        btn02.addActionListener(this);
    }

}
```

7 – Altere o método actionPerformed conforme o código abaixo e faça os testes novamente na classe Principal. O que percebeu de diferença em relação ao teste do item 6 ?

```
@Override
public void actionPerformed(ActionEvent e) {
    Object o = e.getSource();
    JButton b = (JButton)o;
    String nome = b.getText();

    System.out.println("o Clique do mouse foi acionado por : " + nome + " da
classe " + o.getClass());

}
```