**Eduardo Sinico, Ezequiel & Matheus - *SENAI* – SA3 LayoutManager**

**Relatório das Calculadoras em JAVA**

Este é um relatório demostrando como foi feito o layout da interface da calculadora de Programador em Java Swing:

Em Java, "layout" se refere à maneira como os componentes gráficos de uma interface de usuário são organizados em um contêiner, como uma janela ou um painel. O layout determina a disposição e o posicionamento dos elementos visuais em relação uns aos outros. Layouts são essenciais para criar interfaces de usuário atraentes, responsivos e funcionais em aplicativos Java.

**Tópicos:**

* Calculadora de Programador;

- BorderLayout;

- GridLayout;

- FlowLayout.

* Calculadora de IMC;
* Calculadora Padrão.

**Calculadora de Programador – feito pelo Eduardo:**

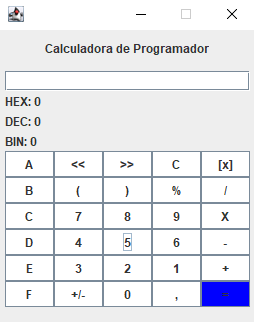
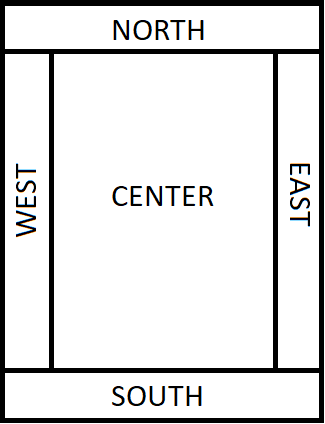


Imagem da Calculadora de Programador

Esses são os três tipos de layout usado no programa:

* **BorderLayout**: BorderLayout é um layout pré-determinado em Java onde ele define as posições de norte, sul, leste, oeste e o centro da página.



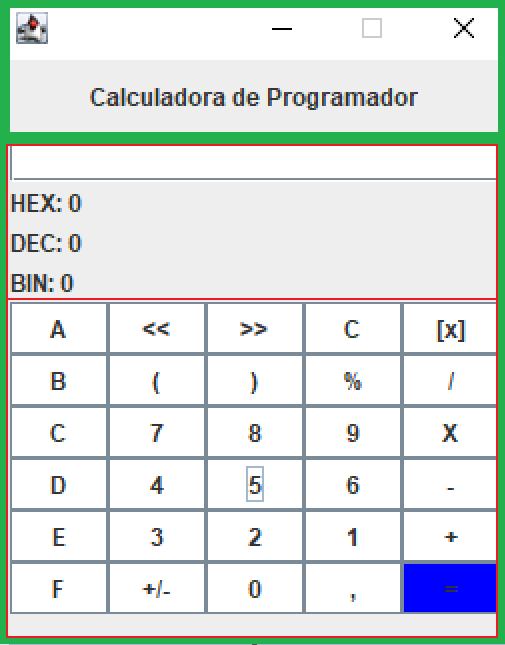
Exemplo de BorderLayout

**No projeto:**

Primeiramente eu criei um JFrame e adicionei o layout dele para BorderLayout, após isso, adicione dois painéis com suas determinadas posições.

*JFrame* janela **= new** JFrame();

janela.setLayout(**new** BorderLayout());



Exemplo da utilização de BorderLayout na calculadora

No projeto de Java, as cores verde e vermelha representam onde está localizado o BorderLayout. Começando pela parte verde, eu defini dois painéis, uma para o norte:

*JPanel* painelTop **= new** JPanel();

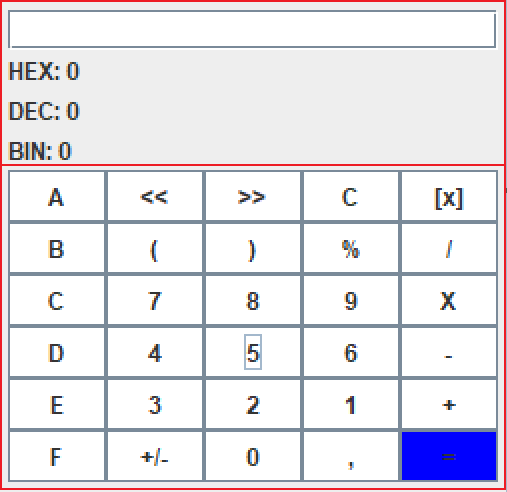
janela.add(painelTop,BorderLayout.NORTH);

E outra para o centro:

*JPanel* painelCenter **= new** JPanel();

janela.add(painelCenter,BorderLayout.CENTER);

Indo para o a parte vermelha, eu criei um segundo painel principal, com a mesma configuração de Layout para o JFrame, o BorderLayout:



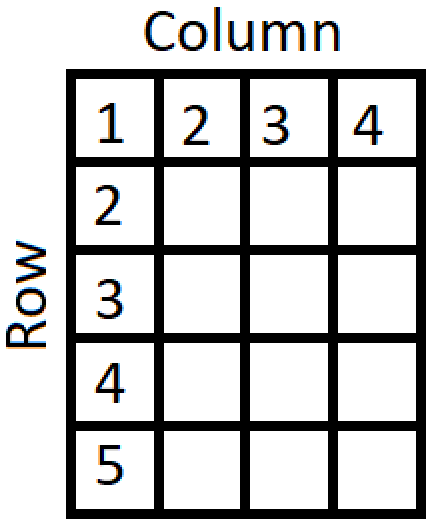
Neste segundo painel principal, eu criei dois painéis e alinhei eles um de baixo do outro. O TextField do resultado, e os valores HEX, DEC e BIN estão posicionados na parte superior, enquanto, as teclas da calculadora estão posicionadas na parte inferior, utilizando a mesma lógica nos processos anteriores.

* **GridLayout**: GridLayout é outro estilo de Layout bem utilizado em Java. Nela, podemos definir Painéis ou Frames com determinadas quantidades de linhas e colunas. Geralmente as JFrames criadas em Java já possuem este Layout.

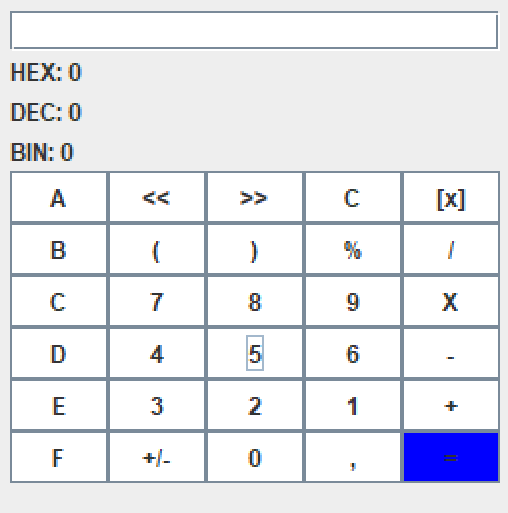
**Funcionalidade do GridLayout:**

*JPanel* painel **= new** JPanel();

painel.setLayout(**new** GridLayout(rows: 5, cols: 4));

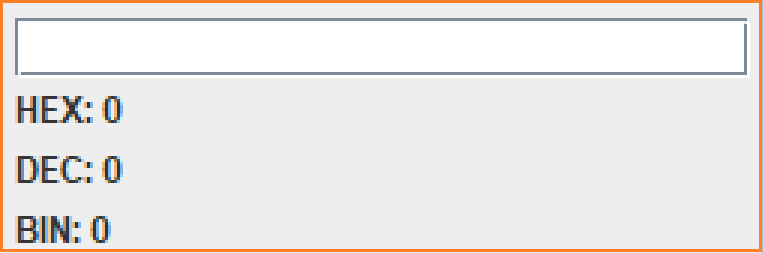
****

**No projeto:**

****

No Layout da calculadora, foi utilizado o ***GridLayout*** em dois painéis.

O primeiro painel apresenta um GridLayout com 4 linhas e uma coluna:



**Código:**

painelResults.setLayout(**new** GridLayout(rows: 4, cols: 1));

painelResults.add(**new** JTextField());

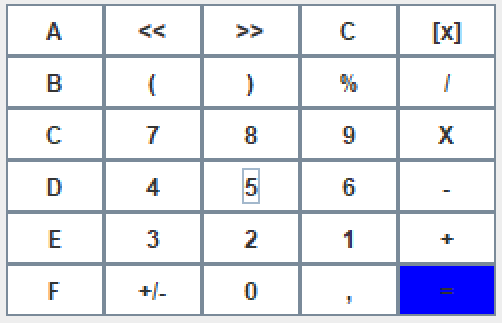
painelResults.add (**new** JLabel("HEX: " + 0));

painelResults.add(**new** JLabel("DEC: " + 0));

painelResults.add(**new** JLabel("BIN: " + 0));

**OBS.:** É necessária uma coluna para que os elementos possam estar um debaixo do outro.

Os mesmos procedimentos de ***GridLayout*** foram aplicados para o segundo painel, onde apresenta os botões da calculadora:

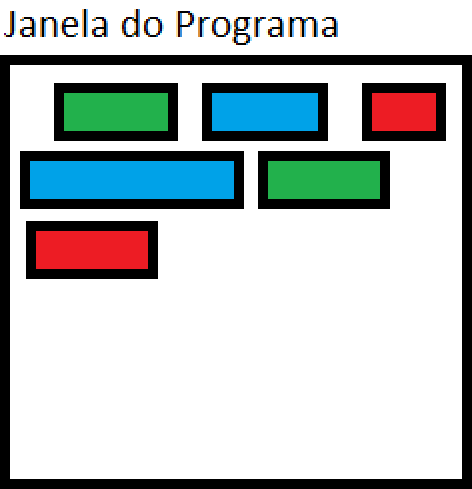


**Código:**

painelCalc.setLayout(**new** GridLayout(rows: 6, cols: 5));

Percebe-se que a “tabela” de botões apresenta 6 linhas e 5 colunas, onde destes 30 elementos, cada um ocupa um espaço desta lista, seguindo a ordem da esquerda para direita e de cima para baixo.

* **FlowLayout**: Por fim, mas não menos importante, o FlowLayout é mais um Layout pronto na Linguagem Java. Ele alinha os elementos um ao lado do outro até o fim da janela, caso não haja espaço, ele pula para parte de baixo do programa.



**No projeto:**



No projeto, a única parte que possuí o Layout Flow está no próprio título. Entretanto, não foi necessário atribuir este Layout para este painel, pois ela já vem atribuída automaticamente para os JPanels. Mas você pode seguir o seguinte código:

painel.setLayout(**new** FlowLayout());

* **Calculadora de IMC – Feito pelo Ezequiel:**

# SA3-Projeto\_Interface\_Java

1. Layout e Painéis:

\* Foi criado um JFrame chamado "painelPrincipal". Basicamente é o meu container Pai, ele que contém todos os componentes e outros containers. Seu layout foi definido como "BorderLayout", que foi dividio em quatro regiões: Norte, Leste e Sul.

\* Diante das três regiões foram criados três JPanels (todos são BorderLayout) para ocupá-las: "painelSuperior" (Norte), "painelLeste" (Leste) e "painelInferior" (Sul), que representam diferentes seções do "painelPrincipal".

\* Para ocupar as três novas seções foram criados mais três sub JPanels (BorderLayout) para que eu pudesse inserir e organizar os próximos componentes componentes: "painelResultado" (painelSuperior), "painelInfos" (painelLeste) e "painelCalcular" (painelInferior).

2. Componentes da Interface:

\* Diversos componentes foram criados, como rótulos (JLabel), campos de texto (JTextField) e um botão (JButton).

\* Os componentes foram organizados com a criação de mais três JPanels inseridos nos seguintes paineis: "painelImc" é GridLayout e foi inserido no centro do "painelResultado", "painelRecebimento" é GridLayout e foi inserido no centro do "painelInfos" e "painelEnviar" é FlowLayout e foi inserido no centro do "painelCalcular"

\* Utilizei GridLayout nos painéis "painelImc" e "painelRecebimento", pois é mais fácil e dinâmico inserir e organizar os meus componentes da forma que eu gostaria, que é uma "JLabel" em cima de um "JTextField". Para isso foi necessário que eu definisse o GridLayout com UMA coluna, a quantidade de linhas sería a minha quantidade de componentes que eu gostaria de inserir naquele local.

3. Configuração do Botão "Calcular":

\* O botão "Calcular" é configurado com cores, dimensões e um ouvinte de eventos (event listener) que define a ação a ser realizada quando o botão é clicado.

Quando o botão é clicado, ele lê os valores inseridos nos campos de altura e peso, realiza o cálculo do IMC e exibe o resultado em um "JTextField" localizado no "painelImc".

4. Comportamento do Botão:

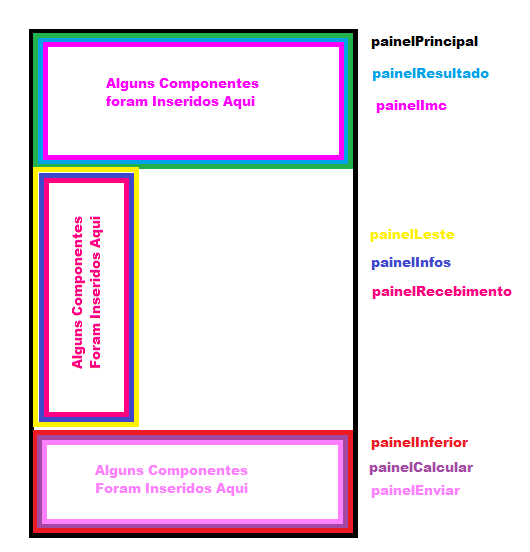
\* Quando o botão "Calcular" é clicado, os valores de altura e peso são lidos dos campos de texto. Em seguida, a altura é elevada ao quadrado utilizando "Math.pow" e o cálculo do IMC é realizado, dividindo o peso pelo quadrado da altura (convertido de cm^2 para m^2).

5. Configuração dos Painéis:

\* Os componentes são inseridos nos painéis correspondentes utilizando os layouts "BorderLayout", "GridLayout", para controlar a disposição e o alinhamento dos elementos.

\* A janela "painelPrincipal" é configurada com várias propriedades, incluindo o fechamento padrão quando a janela é fechada (setDefaultCloseOperation), posição inicial na tela (setBounds), visibilidade (setVisible), tamanho automático de acordo com o conteúdo (pack), e a janela não pode ser redimensionada (setResizable).

No geral, esta calculadora de IMC é uma aplicação simples com uma interface gráfica que permite ao usuário inserir sua altura e peso, clicar no botão "Calcular" e obter o resultado do seu Índice de Massa Corporal (IMC) exibido na janela.



* **Calculadora Simples – Feito pelo Matheus:**

**Relatório Detalhado - Calculadora Simples em Java com Ênfase no Layout**

**Introdução:**

Este relatório explora com detalhes o projeto da Calculadora Simples em Java, focando especificamente nas decisões de layout e design. A aplicação utiliza a biblioteca Swing para criar uma interface gráfica eficaz e esteticamente agradável.

**Layout da Interface:**

O layout da interface é um aspecto crítico do projeto, pois influencia diretamente na usabilidade e na experiência do usuário. Abaixo, detalhamos o design da interface da calculadora:

**Configurações do JFrame:**

O JFrame é a janela principal da aplicação.

Configurações:

Título: "Calculadora Simples".

Ação de encerramento: JFrame.EXIT\_ON\_CLOSE.

Redimensionamento desabilitado: setResizable(false).

Tamanho da janela: 300x400 pixels.

**Painel Principal (JPanel):**

Um JPanel principal é usado como um contêiner para organizar os elementos da interface.

Layout configurado como BorderLayout, permitindo a organização de componentes nas regiões norte, centro e sul.

**Título Descritivo (JLabel):**

Um JLabel denominado titleLabel é posicionado na parte superior da interface, acima do campo de resultado.

Configurações:

Texto: "Calculadora".

Fonte em negrito, tamanho 20.

Alinhamento horizontal central (JLabel.CENTER).

**Campo de Resultado (JTextField):**

Um JTextField chamado textField ocupa a área central do JPanel.

Configurações:

Fonte com tamanho 24.

Alinhamento à direita (JTextField.RIGHT).

Modo somente leitura (setEditable(false)).

**Botões Numéricos e Operadores (JButton):**

Os botões numéricos (0 a 9) e operadores (+, -, \*, /) são representados como JButton.

Organizados em um painel de botões com layout de grade (GridLayout) de 4x4, com espaçamento de 5 pixels entre eles.

Configurações:

Fonte com tamanho 18.

Cor de fundo personalizada (cinza claro) para melhorar a estética.

Associados a um ActionListener para responder a interações do usuário.

**Lógica da Calculadora:**

A lógica da calculadora é gerenciada por uma classe interna ButtonClickListener, que lida com as ações dos botões.

Os botões numéricos permitem a entrada de números no campo de texto.

O botão "C" limpa o campo de texto e reinicia as variáveis relacionadas à calculadora.

Os botões de operadores configuram a operação pendente com base no operador pressionado e no número atual exibido no campo de texto.

O botão "=" realiza a operação pendente e exibe o resultado no campo de texto, com tratamento especial para a divisão por zero.

O resultado é formatado para conter até 6 casas decimais utilizando a classe DecimalFormat.

**Conclusão:**

Os projetos das calculadoras em Java demonstra um layout bem planejado e organizado, com foco na usabilidade e na estética. A atenção aos detalhes, como o título descritivo, a configuração da janela principal e a escolha de cores agradáveis, contribui para uma experiência de usuário mais eficaz e agradável. Este projeto serve como um exemplo sólido de desenvolvimento de aplicativos de interface gráfica em Java usando a biblioteca Swing, onde o layout é cuidadosamente projetado para melhorar a usabilidade e a estética.