**LAYOUT DE CALCULADORAS**

Relatório final

# 1. Objetivo

Este trabalho tem como objetivo a compreensão e utilização de componentes necessários à aplicações gráficas a partir do desenvolvimento de um sistema java simples que contenha três calculadoras simples. Cada calculadora deve implementar layouts e ter uma interface visual, atraente e intuitiva, além de que sua interface deve ser responsiva para diferentes tamanhos de tela.

# 2. Descriç**ão**

## 2.1 Conceitos

Layouts são utilizados para definir a estrutura de organização dos componentes de uma interface gráfica, como botões, caixas de texto, rótulos etc. Eles ajudam a garantir que os elementos da GUI (Graphical User Interface) sejam exibidos de maneira organizada e responsiva em variados tamanhos de tela e resolução.

## 2.2 Escolha de Layouts

Será aprofundado mais à frente a motivação por trás de cada escolha.

* **Calculadora Padrão**
  + GridBagLayout
  + BorderLayout
* **Calculadora de Temperatura**
  + GridBagLayout
  + CardLayout
* **Calculadora IMC**

### 2.2.1 **Resumo Layouts**

Um breve resumo explicativo sobre layouts escolhidos.

* **GridBagLayout -** layout flexível que permite posicionar os componentes em uma grade de células. Cada célula pode ter tamanhos e espaçamentos diferentes, tornando-o adequado para layouts complexos e personalizados. É útil quando você precisa de controle preciso sobre o posicionamento e o dimensionamento dos componentes em uma interface de usuário.
* **BorderLayout –** layout que organiza seu contâiner em cinco regiões: norte, sul, leste, oeste e centro. Cada região pode conter um único componente e ocupa todo o espaço disponível em sua área. É fácil de usar para criar layouts simples e é adequado para interfaces de estilo "painel de controle".
* **CardLayout –** layout que permite empilhar vários painéis (cartões) um sobre o outro e alternar entre eles para mostrar um painel de cada vez. É útil para criar interfaces com várias "telas" onde apenas uma é visível de cada vez, como guias ou assistentes de configuração. Você pode controlá-lo dinamicamente para alternar entre os cartões, facilitando a implementação de interfaces de usuário interativas.

### 2.2.2 Motivaç**ão**

Breve explicação sobre a comparação/motivação de escolha de cada layout.

* **GridBagLayout –** layout escolhido devido ao seu controle mais preciso sobre o posicionamento e dimensionamento dos componentes, além de também oferecer uma flexibilidade ao definir tamanhos e espaçamentos diferentes para cada célula da grade, permitindo assim criar layouts complexos que não podem ser facilmente alcançados com o **GridLayout** por exemplo, considerando tais pontos, o **GridBagLayout** se torna uma escolha ideal para uma interface que requer um grau mais elevado quanto a personalização e adaptação específica para um design.
* **BorderLayout –** layout escolhido pois ele é a solução mais eficiente para a criação de interfaces simples. Em comparação com o **FlowLayout** por exemplo, ele alinha os componentes em uma única linha, enquanto o **BorderLayout** oferece uma organização mais clara e ampla com suas regiões de “agregação” (norte, sul, leste, oeste e centro), tornando-o uma melhor opção para a criação de designs claros e intuitivos.
* **CardLayout –** layout escolhido devido a sua capacidade de exibir diversos painéis, onde apenas um deles deve ser visível por vez. Em comparação com o **JtabbedPane** por exemplo, que usa guias para alternar entre os painéis, o **CardLayout** oferece uma abordagem mais dinâmica e flexível, permitindo alternar entre os cartões (painéis) de forma programática, criando interfaces de usuário interativas. A simplicidade e capacidade de controlar dinamicamente o conteúdo exibito tornam este layout uma escolha superior para o cenário aplicado.

## 2.3 Funcionalidades

* **Calculadora Padrão –** Realizar as quatro operações básicas da matemática. Deletar último caractere inserido. Limpar campo e histórico (caso viesse ocorrendo uma sequência de contas).­
* **Calculadora de Temperatura –** Escolher qual tipo de medição de temperatura inserida. Escolher qual tipo de medição de temperatura a ser convertida. Realizar a conversão de temperatura.
* **Calculadora IMC -**

## 2.4 Estrutura do Programa

A aplicação foi desenvolvida e estruturada de forma que resultou em:

* **Interface Gráfica Base –** Três classes bases (App; FrameApp; TabbedPane) que organizam e gerenciam a exibição da janela da aplicação.
* **Calculadora Padrão –** Contém as classes (Calculadora; Visor; Botoes) que controlam a exibição e lógica da calculadora padrão
* **Calculadora de Temperatura -** Contém as classes (CalcTemperatura; TemperaturaTop; TemperaturaBottom; CalcTempController) que controlam a exibição e lógica da calculadora de temperatura
* **Calculadora IMC -**

# 3. Detalhaç**ão**

## 3.1 Calculadora Padr**ão**

Calculadora feita para realizar as quatro operações matemáticas básicas. Para facilitar o controle esta calculadora foi dividida em três classes.

### 3.1.1 Calculadora.java

O **Calculadora.java** é o painel principal que contém todos os elementos visuais da calculadora. Abaixo estão os detalhes da estrutura gráfica que tornam esse componente eficaz:

* **GridBagLayout:** O **Calculadora.java** utiliza o layout **GridBagLayout**. Este layout é flexível e poderoso, permitindo que os componentes sejam organizados em linhas e colunas de acordo com as restrições definidas por meio de **GridBagContraints**. Essa flexibilidade é crucial para posicionar precisamente os componentes da calculadora.
* **Visor:** O componente **Visor.java** é uma parte integrante da **Calculadora.java**. Ele é posicionado na parte superior do painel (**GridBagConstraints.gridy = 1**) e ocupa toda a largura (**GridBagContraints.gridWidth = 1**) e uma única linha (**GridBagConstraints.gridheight = 1**). Isso garante que o visor esteja no topo do painel e se expanda horizontalmente para mostrar os números e operações de forma clara.
* **Botões.java:** Os botões numéricos e de operação são organizados em uma grade na parte inferior do painel (**GridBagConstraints.gridy = 1**). Eles ocupam várias linhas e colunas na grade, com tamanhos e proporções específicos definidos por meio de **GridBagContraints**. Cada botão é configurado com um estilo de fonte atraente e cores para tornar a interface da calculadora agradável visualmente.
* **Background Personalizado:** O pano de fundo do **Calculadora.java** é definido como uma cor escura (**Color.DARK\_GRAY)**. Isso fornece um contraste agradável com os elementos de interface, tornando-os mais visíveis e destacados.

### 3.1.2 Visor.java

O **Visor.java** desempenha um papel importante na apresentação visual da calculadora. Abaixo estão os detalhes da estrutura gráfica que tornam esse componente eficaz:

* **LayoutBorder:** O  **Visor.java** utiliza um layout **BorderLayout.** Esse layout divide o componente em cinco regiões: norte (topo), sul (parte inferior), leste (direita), oeste (esquerda) e centro. Essa estrutura é particularmente útil para posicionar o **JTextArea** que exibe os números e as operações no centro da área do visor. A divisão permite que o texto seja exibido de forma centralizada e clara.
* **JTextAreaPersonalizado:** O componente **JtextArea** usado no **Visor.java** é personalizado para atender às necessidades específicas da calculadora. Ele é configurado com **setLineWrap(true)** para permitir a quebra automática de linha quando o texto atinge o limite horizontal da área. Essa quebra de linha é crucial para garantir que números longos ou resultados extensos sejam exibidos corretamente.
* **Estilo de Fonte:** A fonte do **JtextArea** é definida como "Arial" em negrito com tamanho 40. Essa escolha de estilo e tamanho de fonte é esteticamente agradável e garante que os números sejam claramente visíveis, mesmo em tamanhos de tela menores. O texto em negrito adiciona ênfase aos números e às operações.
* **Borda Estilizada:** O **Visor.java** é decorado com uma borda estilizada usando **EtchedBorder**. Essa borda fornece um visual agradável ao componente e também ajuda a destacar o **Visor.java** na interface da calculadora. A borda contribui para a identificação rápida da área onde os resultados são exibidos.

### 3.1.3 Botoes.java

A interface gráfica desempenha um papel fundamental na experiência do usuário ao interagir com a calculadora. Alguns detalhes da parte gráfica do componente **Botoes.java**:

* **Layout com GridBagLayout**: O componente **Botoes.java** utiliza o layout **GridBagLayout** para organizar os botões. Esse layout oferece flexibilidade na disposição dos componentes, permitindo que os botões sejam posicionados de maneira personalizada.
* **Botões Personalizados:** Cada botão é um objeto **Jbutton** personalizado. Eles são configurados com uma fonte grande e negrito para garantir que os números e operadores sejam facilmente legíveis. Além disso, o texto dos botões é colorido em branco para criar contraste com o fundo escuro, tornando a interface mais atraente e legível.
* **Cores Contrastantes:** Os botões numéricos e de operação são configurados com cores contrastantes em relação ao fundo escuro da calculadora. Os botões numéricos têm uma cor de fundo escura, enquanto os botões de operação têm cores distintas, como vermelho para "CLEAR" e verde para "=", para indicar suas funções.
* **Feedback Visual:** Ao clicar em um botão, ele fornece feedback visual ao alterar sua cor de fundo temporariamente. Isso ajuda a indicar visualmente que o botão foi pressionado, melhorando a experiência do usuário e fornecendo um retorno imediato sobre a interação.
* **Posicionamento Personalizado:** Cada botão é posicionado de maneira personalizada usando as coordenadas do **GridBagConstraints**. Isso permite que os botões sejam organizados em uma grade específica, seguindo um layout intuitivo que os usuários esperam de uma calculadora.
* **Gestão de Eventos:** Os botões são configurados com ouvintes de eventos que respondem aos cliques do mouse. Quando um botão é clicado, a função **Decisao()** é chamada para determinar a ação apropriada com base no botão pressionado.

O **Botoes.java** desempenha um papel crucial na interatividade da calculadora. Aqui estão detalhes adicionais sobre suas funcionalidades:

* **Teclado Virtual:** O componente **Botoes.java** lida com entradas de teclado, tornando a calculadora mais versátil. Ele responde aos eventos do teclado, permitindo que o usuário insira números e operações pressionando as teclas correspondentes no teclado físico. Isso é especialmente útil para usuários que preferem usar o teclado em vez do mouse.
* **Histórico de Cálculos:** O **Botoes.java** mantém um histórico de números e operadores inseridos pelo usuário. Isso é fundamental para permitir cálculos complexos que envolvem várias etapas. Os números e operadores são armazenados em ArrayLists separados (**‘numeros’** e **‘operadores’**), o que facilita o acompanhamento das entradas do usuário.
* **Validação de Entrada:** O **Botoes.java** valida as teclas pressionadas para garantir que apenas entrada válida seja processada. Ele reconhece operadores (+, -, \*, /), números e o ponto decimal como entradas válidas. Qualquer tecla que não seja uma entrada válida é ignorada, prevenindo erros de entrada.
* **Botões de Controle:** Além dos botões numéricos e de operação, o **Botoes.java** inclui botões de controle essenciais, como "**CLEAR**" e "**DEL**" (Delete). O botão "**CLEAR**" permite ao usuário limpar todo o histórico e o visor, enquanto o botão "**DEL**" permite ao usuário apagar o último caractere inserido.

## 3.2 Calculadora **de Temperatura**

### 3.2.1 CalcTemperatura.java

O **CalcTemperatura.java** é um painel que contém os elementos visuais para a conversão de temperaturas. Abaixo estão os detalhes da estrutura gráfica que tornam esse componente eficaz:

* **Layout GridBag:** O **CalcTemperatura.java** utiliza o layout **GridBagLayout**, o mesmo usado no exemplo anterior. Isso permite uma organização precisa dos elementos na interface gráfica.
* **TemperaturaTop.java:** O componente **TemperaturaTop.java** representa a parte superior da interface onde o usuário pode inserir o valor da temperatura e escolher a unidade de origem (Celsius, Fahrenheit ou Kelvin). Ele é posicionado no topo do painel (**GridBagConstraints.gridy = 0**) e ocupa toda a largura (**GridBagConstraints.gridwidth = 1**) e uma única linha (**GridBagConsstraints.gridheight = 1**).
* **Botão de Conversão:** Um botão "CONVERTER" é adicionado ao painel. Ele é posicionado logo abaixo do **TemperaturaTop.java (GridBagConstraints.gridy = 1**). O botão é usado para iniciar a conversão da temperatura.
* **TemperaturaBottom.java:** O componente **TemperaturaBottom.java** representa a parte inferior da interface, onde o resultado da conversão é exibido nas três unidades de temperatura (Celsius, Fahrenheit e Kelvin). Ele é posicionado abaixo do botão de conversão (**GridBagConstraintes.gridy = 2**) e ocupa toda a largura do painel.
* **Event Listener para Conversão:** Um **ActionListener** é adicionado ao botão "CONVERTER". Quando o botão é clicado, o código dentro do **ActionListener** é executado. Ele pega os valores inseridos no **TemperaturaTop.java**, realiza a conversão com base nas unidades selecionadas e exibe o resultado no **TemperaturaBottom.java**.

O **CalcTemperatura.java** é responsável por fornecer uma interface para a conversão de temperaturas entre três unidades: Celsius, Fahrenheit e Kelvin. Aqui estão as funcionalidades detalhadas:

* **Controle de Conversão:** O componente **CalcTemperatura.java** utiliza um objeto **CalcTempController.java** para lidar com a lógica de conversão de temperatura. O controlador recebe entradas do usuário, executa as conversões e fornece os resultados.
* **TemperaturaTop.java:** A parte superior da interface gráfica é onde o usuário insere o valor da temperatura e seleciona a unidade de origem (Celsius, Fahrenheit ou Kelvin). Os campos de entrada e o menu suspenso são usados para isso.
* **Botão de Conversão:** Um botão "CONVERTER" é adicionado abaixo do **TemperaturaTop.java**. Quando o usuário clica neste botão, a ação de conversão é acionada.
* **Conversão de Temperatura** Quando o botão "CONVERTER" é clicado, o código dentro do **ActionListener** é executado. Ele lê o valor da temperatura e a unidade de origem selecionada e, em seguida, usa o controlador **CalcTempController.java** para realizar a conversão.
* **Exibição de Resultados:** Os resultados da conversão são exibidos na parte inferior da interface, no componente **TemperaturaBottom.java**. As três unidades de temperatura (Celsius, Fahrenheit e Kelvin) são atualizadas com os valores convertidos.
* **Manipulação de Erros:** O código no **ActionListener** também lida com exceções, como entradas inválidas. Se um valor inválido for inserido, a mensagem "Insira um valor válido" será exibida nos campos de resultado.
* **Integração de Componentes:** O **CalcTemperatura.java** integra os componentes **TemperaturaTop.java**, **TemperaturaBottom.java** e o botão de conversão para criar uma experiência de usuário coesa e eficiente.

### 3.2.2 LimitedDocument.java

O **LimitedDocument.java** é uma classe que estende **PlainDocument** e é usada para limitar o número de caracteres que podem ser inseridos em um campo de texto, de acordo com o tipo de calculadora **(**“Temp” ou outra – calculadora padrão-). Abaixo estão os detalhes da estrutura e da função deste componente:

* **Herança do PlainDocument: LimitedDocument** herda da classe **PlainDocument**, que é uma classe da biblioteca Java Swing usada para manipular documentos de texto. Essa herança fornece funcionalidades básicas de documento de texto, que são estendidas para limitar o número de caracteres.
* **Construtor Personalizaod:** O construtor **LimitedDocument** aceita dois parâmetros: **maximoCaracteres** e **tipoCalculadora**. O **maximoCaracteres** define o número máximo de caracteres permitidos no campo de texto, enquanto **tipoCalculadora** indica se o campo de texto está associado a uma calculadora de temperatura (**“Temp”**) ou outra calculadora. Isso permite que o documento limite caracteres de acordo com o contexto da calculadora.
* **Método insertString Personalizado:** O método **insertString** é sobrescrito para controlar a inserção de texto no campo. Ele verifica se a string a ser inserida é válida de acordo com o tipo de calculadora:
  + **Calculadora de Temperatura(“Temp”):** Para esse tipo de calculadora, o documento permite números e um único hífen na primeira posição, que é usado para números negativos. Ele também garante que não haja mais de um hífen no texto. Essas restrições são aplicadas para garantir que apenas entradas válidas sejam aceitas.
  + **Outras Calculadoras:** Para outros tipos de calculadoras, o documento permite apenas os caracteres válidos para operações matemáticas, como números e operadores aritméticos (+, -, \*, /). Isso garante que apenas fórmulas matemáticas válidas possam ser inseridas no campo.

### 3.2.3 TemperaturaTop.java e TemperaturaBottom.java

A calculadora de conversão de temperatura é dividida em duas partes principais: **TemperaturaTop.java** e **TemperaturaBottom.java**, que se complementam para fornecer uma interface de usuário eficaz.

* **Layout Card:** Ambos os componentes (**TemperaturaTop** e **TemperaturaBottom**) utilizam um layout C**ardLayout**. Esse layout permite alternar entre diferentes unidades de temperatura (Celsius, Fahrenheit e Kelvin) e é controlado por uma caixa de seleção.
* **Caixa de Seleção:** Uma caixa de seleção (C**omboBox**) é adicionada à parte superior dos dois painéis para permitir que o usuário escolha entre as unidades de temperatura disponíveis (Celsius, Fahrenheit e Kelvin). Quando o usuário seleciona uma unidade, o **CardLayout** é usado para exibir o cartão correspondente (Celsius, Fahrenheit ou Kelvin).
* **Campos de Texto:** Para cada unidade de temperatura (Celsius, Fahrenheit e Kelvin), há um campo de texto correspondente (c**ampoC**, **campoF** e c**ampoK**). Esses campos de texto são configurados para permitir que o usuário insira valores para conversão e são configurados com um limite de caracteres (**LimitedDocument**). Os campos de texto são projetados para quebrar automaticamente as linhas quando o texto é muito longo.
* **Estilo de Testo:** A fonte dos campos de texto é definida como "Arial" com negrito e tamanho 40 para garantir que os números sejam exibidos de forma clara e legível. Além disso, o componente de texto é configurado para que o texto seja exibido da direita para a esquerda para corresponder à entrada de dados de temperatura.
* **Bordas e Rolagem:** Cada campo de texto é colocado em um painel com borda para enfatizar visualmente a área de entrada. Além disso, os campos de texto são colocados em um componente de rolagem (**JScrollPane**) para permitir que o usuário role o texto caso a entrada seja muito longa.

**Diferenças entre TemperaturaTop.java e TemperaturaBottom.java**

* **TemperaturaTop** é responsável pela entrada de valores de temperatura pelo usuário e permite que o usuário escolha as unidades de conversão.
* **TemperaturaBottom** é responsável por exibir os resultados da conversão de temperatura. Os campos de texto nesse componente são configurados como somente leitura, impedindo que o usuário altere os valores exibidos. Os campos são atualizados automaticamente conforme as conversões são realizadas.

### 3.2.4 CalcTempController.java

O **CalcTempController.java** é uma classe controladora que gerencia a conversão de temperaturas em um aplicativo de calculadora de temperatura. Embora não tenha uma interface gráfica diretamente, ele controla a lógica por trás das operações de conversão. Aqui estão detalhes importantes relacionados à sua estrutura:

* **Conversão de temperatura:** O **CalcTempController.java** realiza a conversão de temperatura com base nas unidades de temperatura selecionadas (Celsius, Fahrenheit, Kelvin) e nos valores de temperatura fornecidos. Ele implementa métodos para todas as possíveis conversões, como de Celsius para Fahrenheit, de Celsius para Kelvin, de Fahrenheit para Celsius, etc. Esses métodos são chamados com base nas escolhas do usuário.
* **Gerenciamento de Campos de Entrada:** A classe interage com campos de entrada de temperatura, que podem estar em diferentes unidades (Celsius, Fahrenheit, Kelvin). Ele lê os valores desses campos, verifica se são válidos e os converte conforme necessário.
* **Lógica de conversão** A lógica de conversão é implementada de acordo com as fórmulas matemáticas padrão para conversões de temperatura entre as unidades mencionadas. Isso garante que as conversões sejam precisas e confiáveis.
* **Manipulação de exceções:** A classe inclui manipulação de exceções para lidar com possíveis erros de conversão ou entradas inválidas. Se um usuário inserir um valor inválido ou ocorrer um erro durante a conversão, o código está preparado para tratar essas situações de forma apropriada.
* **Integração com ComponentesVisuais:** Embora esta classe seja principalmente responsável pela lógica de conversão, ela está intimamente relacionada aos componentes visuais da interface do usuário. Ela lê os valores inseridos pelo usuário por meio desses componentes e fornece os resultados da conversão para exibição na interface do usuário.
* **Estrutura MVC:** A estrutura geral do código segue o padrão de arquitetura Modelo-Visão-Controlador (MVC). O **CalcTempController.java** desempenha o papel do controlador, separando a lógica de negócios das operações de interface do usuário, permitindo uma melhor organização e manutenção do código.

## 3.3 **Calculadora IMC**

### 3.3.1 Layouts

### 3.3.2 Funcionalidades