**Disciplina: Programação Orientada a Objetos em Java**

**Tema: Tipos de Layout com a Biblioteca `LayoutManager`**

**Situação de Aprendizagem 3: Criando uma Calculadora de Layouts**

**Introduçao/Preparação**

Em interfaces gráficas, o termo "layout" refere-se à maneira como os componentes (como botões, caixas de texto, etc.) são organizados e posicionados dentro de um contêiner, como um JFrame ou um JPanel. Escolher o layout certo é crucial para garantir que a interface seja apresentada de forma coerente e esteticamente agradável em diferentes dispositivos e resoluções de tela.

Na biblioteca **LayoutManager** em Java, existem diversos tipos de layouts disponíveis para ajudar na organização dos elementos. Vou introduzir alguns dos layouts mais comuns:

**FlowLayout**:

* Organiza os componentes em uma única linha, adicionando um ao lado do outro até que o espaço horizontal acabe. Quando a linha está cheia, os componentes continuam na próxima linha.
* É útil para botões ou controles onde a ordem de exibição não é crítica.

**Exemplo com FlowLayout**:

* Uma calculadora simples onde os botões são organizados em uma única linha até que não haja mais espaço horizontal.

**BorderLayout**:

* Divide o contêiner em cinco regiões: norte, sul, leste, oeste e centro.
* Cada região pode conter um único componente, e o tamanho dos componentes é ajustado automaticamente conforme o tamanho do contêiner.

**Exemplo com BorderLayout**:

* Uma aplicação com um menu na parte superior, uma barra de ferramentas na parte inferior e um painel central para o conteúdo principal.

**GridLayout**:

* Organiza os componentes em uma grade retangular, onde todos os componentes têm o mesmo tamanho.
* É útil para criar layouts de estilo tabuleiro, onde os componentes são organizados em linhas e colunas.

**Exemplo com GridLayout**:

* Um jogo da velha, onde os botões são organizados em uma grade 3x3.

**BoxLayout**:

* Organiza os componentes em uma única linha, na vertical ou na horizontal.
* É útil quando se deseja ter um controle mais preciso sobre o tamanho dos componentes.

**Exemplo com BoxLayout**:

* Uma aplicação com uma barra de navegação vertical e uma área de conteúdo principal ao lado.

# **Relatório Final**

## **Introdução**

O presente relatório aborda a criação de três calculadoras utilizando Java Swing: uma Calculadora Básica, uma Calculadora de Custo de Viagem e uma Calculadora de Peso Ideal. Cada uma dessas calculadoras possui características específicas e layouts distintos. Este relatório tem como objetivo fornecer uma introdução aos conceitos de layouts e destacar a importância de escolher o layout adequado para uma interface gráfica.

# **Relatório da Calculadora Básica**

## **Introdução**

Este relatório aborda a criação da Calculadora Básica, uma aplicação Java Swing que oferece funcionalidades de cálculos simples, como adição, subtração, multiplicação e divisão. O principal objetivo desta calculadora é fornecer uma interface intuitiva e fácil de usar para realizar operações matemáticas básicas.

## **Descrição Detalhada**

A Calculadora Básica possui os seguintes elementos e funcionalidades:

* **Visor de Resultados**:
* Um JTextField de tamanho ampliado é utilizado como visor para exibir os números e o resultado dos cálculos. A fonte foi configurada para ser maior, proporcionando melhor legibilidade.
* **Botoes Numéricos e de Operação**:
* Os botões numéricos de 0 a 9 foram adicionados ao layout. Ao pressionar um desses botões, o respectivo número é exibido no visor.
* Foram incluídos botões para as operações de adição (+), subtração (-), multiplicação (x) e divisão (/).
* Um botão "AC" foi adicionado para limpar o visor.
* **Funcionalidades de Cálculo**:
* Ao pressionar os botões numéricos, os números são exibidos no visor.
* A funcionalidade de limpar o visor ("AC") foi implementada.
* A lógica para calcular o resultado ainda precisa ser implementada.

## **Avaliação do Layout**

O layout escolhido para a Calculadora Básica é o **GridLayout**, que organiza os elementos em uma grade. Essa escolha é adequada para uma calculadora de operações simples, pois proporciona uma organização limpa e simétrica dos botões.

### **Vantagens do GridLayout:**

* Facilita a organização de componentes em linhas e colunas.
* É adequado para interfaces com um conjunto fixo de elementos.

## **Possíveis Limitações:**

* Pode não ser tão flexível para interfaces mais complexas ou com layouts irregulares.

## **Conclusão**

A Calculadora Básica fornece uma interface inicial funcional, mas a lógica de cálculo precisa ser implementada para torná-la completamente operacional. O layout escolhido é apropriado para este tipo de aplicação, proporcionando uma organização clara e intuitiva dos elementos.

# **Relatório da Calculadora de Custo de Viagem**

## **Introdução**

Este relatório descreve a criação da Calculadora de Custo de Viagem, uma aplicação Java Swing desenvolvida para calcular o custo de uma viagem com base na distância percorrida, consumo do veículo e preço do combustível. O objetivo desta calculadora é fornecer uma ferramenta prática para estimar os custos de deslocamento.

## **Descrição Detalhada**

A Calculadora de Custo de Viagem possui os seguintes elementos e funcionalidades:

* **Campos de Entrada**:
* Três campos de texto são fornecidos para inserir a distância percorrida (em km), o consumo do veículo (em km/l) e o preço do combustível (em R$/l).
* **Botões de Ação**:
* Dois botões foram adicionados: "Calcular" e "Apagar".
* O botão "Calcular" executa o cálculo do custo da viagem com base nos valores inseridos nos campos.
* O botão "Apagar" limpa os campos de entrada.
* **Exibição do Resultado**:
* Após o cálculo, uma mensagem é exibida em uma caixa de diálogo com detalhes sobre a viagem, incluindo a quantidade de litros de combustível e o custo total.

## **Avaliação do Layout**

O layout escolhido para a Calculadora de Custo de Viagem é o **GridBagLayout**, que oferece maior flexibilidade na organização de componentes. Isso é especialmente importante para posicionar os campos de entrada e botões de ação de forma precisa.

## **Vantagens do GridBagLayout:**

* Permite o posicionamento preciso de elementos.
* É mais flexível e adequado para interfaces mais complexas.

## **Possíveis Limitações:**

* Pode ser mais complexo de configurar do que outros layouts.

## **Conclusão**

A Calculadora de Custo de Viagem fornece uma ferramenta útil para estimar os custos de uma viagem. O layout escolhido (GridBagLayout) permitiu uma organização precisa dos elementos, resultando em uma interface limpa e intuitiva.

## **Relatório da Calculadora de Peso Ideal**

## **Introdução**

Este relatório aborda a criação da Calculadora de Peso Ideal, uma aplicação Java Swing projetada para estimar o peso ideal de uma pessoa com base na altura e no sexo. O objetivo principal desta calculadora é fornecer uma ferramenta para auxiliar as pessoas a monitorarem sua saúde e bem-estar.

## **Descrição Detalhada**

A Calculadora de Peso Ideal possui os seguintes elementos e funcionalidades:

* **Campos de Entrada**:
* Três campos de entrada são fornecidos para inserir o nome, a altura (em metros) e a seleção do sexo (masculino ou feminino).
* **Botões de Ação**:
* Dois botões foram incluídos: "Peso Ideal" e "Apagar".
* O botão "Peso Ideal" realiza o cálculo do peso ideal com base nos valores inseridos nos campos.
* O botão "Apagar" limpa os campos de entrada.
* **Exibição do Resultado**:
* Após o cálculo, uma mensagem é exibida em uma caixa de diálogo com o peso ideal estimado.

## **Avaliação do Layout**

O layout escolhido para a Calculadora de Peso Ideal também é o **GridBagLayout**. Este layout oferece uma organização precisa dos campos de entrada e botões de ação, permitindo uma interface clara e bem estruturada.

## **Vantagens do GridBagLayout:**

* Permite o posicionamento preciso de elementos.
* É mais flexível e adequado para interfaces mais complexas.

## **Possíveis Limitações:**

* Pode ser mais complexo de configurar do que outros layouts.

## **Conclusão**

A Calculadora de Peso Ideal é uma ferramenta útil para estimar o peso ideal de uma pessoa com base na altura e no sexo. O layout escolhido (GridBagLayout) permitiu uma organização precisa dos elementos, resultando em uma interface intuitiva e fácil de usar.

## **Comparação dos Layouts Usados nas 3 calculadoras**

Os layouts utilizados nas três calculadoras têm suas vantagens e desvantagens:

* **GridLayout** (Calculadora Básica):

Vantagens:

* Facilita a organização de componentes em linhas e colunas.
* É adequado para interfaces com um conjunto fixo de elementos.

Desvantagens:

* Pode não ser tão flexível para interfaces mais complexas.
* **GridBagLayout** (Calculadoras de Custo de Viagem e Peso Ideal):
* Vantagens:
* Oferece maior flexibilidade na organização de componentes.
* Permite o posicionamento preciso de elementos.
* Desvantagens:
* Pode ser mais complexo de configurar do que outros layouts.

A escolha do layout depende da complexidade da interface e das necessidades de posicionamento dos componentes. O GridLayout é adequado para layouts simples e regulares, enquanto o GridBagLayout é mais versátil e indicado para interfaces mais complexas.

## **Reflexão sobre a Experiência de Aprendizado**

A criação dessas calculadoras proporcionou uma valiosa experiência de aprendizado. Os principais aprendizados incluem:

* Compreensão dos conceitos de layouts em Java Swing.
* Familiarização com a criação de interfaces gráficas usando diferentes layouts.
* Prática na implementação de funcionalidades interativas.

Esses conhecimentos adquiridos serão aplicados em projetos futuros, possibilitando a criação de interfaces mais intuitivas e eficientes.

## **Conclusão**

A escolha do layout é crucial para o design e a usabilidade de uma interface gráfica. Cada layout possui suas vantagens e deve ser selecionado de acordo com as necessidades específicas do projeto. A criação das calculadoras proporcionou uma valiosa experiência de aprendizado, que será aplicada em futuros projetos de desenvolvimento de interfaces gráficas em Java Swing.