Desenvolvimento Desktop

Aula 11 - Gerenciadores de Layouts I

Apresentação

Esta aula dará a você uma visão geral sobre gerenciadores de layout. Primeiramente, descobriremos o que são e para que eles servem. Em seguida, apresentaremos os gerenciadores de layout fornecidos pela plataforma Java, mostrando os tipos de layout existentes, bem como suas propriedades e aplicações.

Vídeo 01 - Apresentação

Objetivos

Ao final desta aula, você será capaz de:

- Definir o que são e para que servem os gerenciadores de layout.
- Identificar quais os gerenciadores de layout fornecidos pela plataforma Java.
- Descrever como cada gerenciador organiza os componentes visuais do Swing.
- Criar um pequeno projeto chamado Agenda Pessoal.
- Adicionar componentes ao projeto e redimensionar seus componentes.

Conhecendo os Gerenciadores de Layouts

Gerenciadores de layout são objetos que auxiliam os **contêineres** no posicionamento relativo de seus componentes. Isso inclui a adição de componentes no **contêiner**, o redimensionamento, o alinhamento e a ancoragem dos componentes, assim como o comportamento de autorredimensionamento dos mesmos.

Isso quer dizer que ao redimensionar uma janela de um aplicativo, a reorganização dos componentes depende do gerenciador de layout utilizado. O Swing também permite que não utilize qualquer gerenciador de **layout**, ficando a cargo do programador posicionar cada componente em coordenadas que indicam a posição absoluta dentro do **contêiner**. Nesse caso, ao redimensionar a janela, você não perceberá mudança no tamanho dos componentes.

O Swing provê vários gerenciadores de layout para uso geral. Entre eles estão:

- BorderLayout
- BoxLayout
- CardLayout
- FlowLayout
- GridBagLayout
- GridLayout
- GroupLayout
- SpringLayout

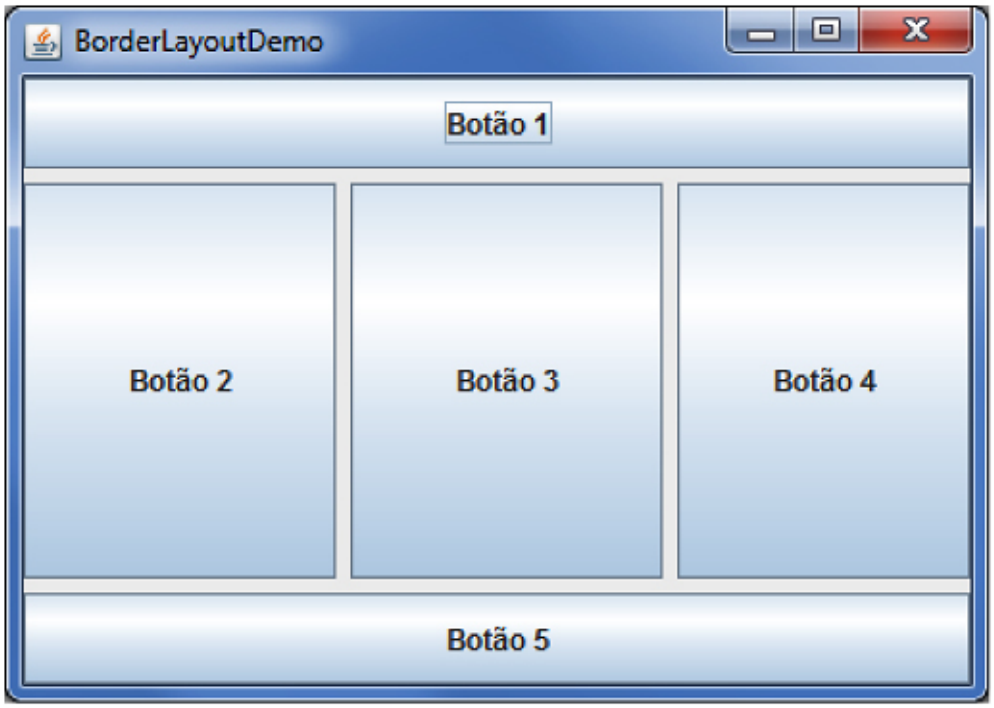
Vamos ver o que cada um deles faz?

Vídeo 02 - Gerenciadores de Layout

Border Layout

O **BorderLayout** é o gerenciador de layout padrão em todo o **content pane**, que é o contêiner principal de todos os **frames, applets** e **janelas de diálogos**. Ele posiciona os componentes em até cinco áreas distintas: no topo (*BorderLayout.NORTH*), na base (*BorderLayout.SOUTH*), à esquerda (*BorderLayout.WEST*), à direita (*BorderLayout.EAST*) e no centro (*BorderLayout.CENTER*). Todo o espaço extra é posicionado na área central. A **Figura 1** mostra uma interface gráfica utilizando esse tipo de layout.

Figura 01 - Utilizando o BorderLayout

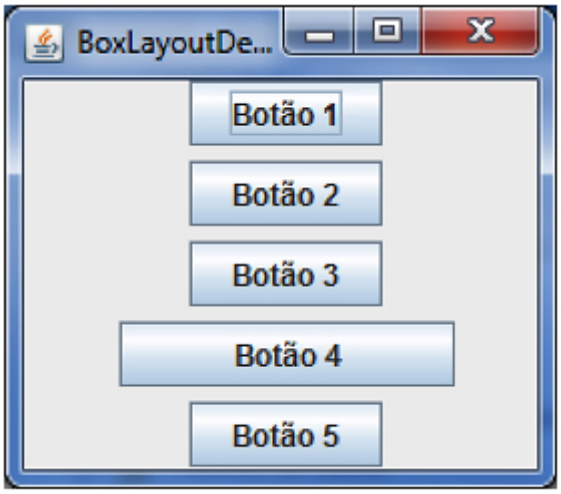


Vídeo 03 - BorderLayout

BoxLayout

O **BoxLayout** posiciona os componentes em uma simples linha ou coluna. Ele respeita o tamanho máximo requerido por cada componente e permite que você alinhe os componentes. A **Figura 2** mostra um exemplo utilizando esse gerenciador de layout.

Figura 02 - Exemplo de BoxLayout



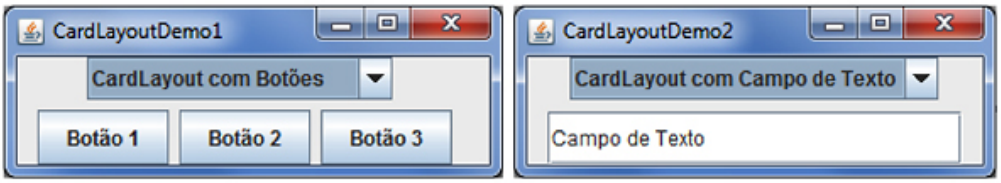
Nesse exemplo, os botões estão alinhados pelo ponto central. Repare que os componentes têm tamanhos diferentes e o gerenciador terá como referência o de maior tamanho.

Vídeo 04 - BoxLayout

CardLayout

O **CardLayout** permite que você implemente uma área que contenha diferentes componentes em diferentes momentos. Ele é, geralmente, controlado por um **comboBox**, com o estado do **comboBox** determinando que grupo de componentes o **CardLayout** irá mostrar. A **Figura 3** mostra um exemplo em dois momentos diferentes da mesma aplicação utilizando o **CardLayout**.

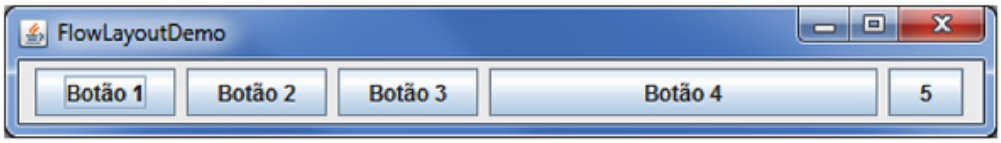
Figura 03 - Exemplos de CardLayout



FlowLayout

O **FlowLayout** é o gerenciador padrão para todo **JPanel**. Ele, simplesmente, organiza os componentes em uma linha, começando uma nova linha, caso o contêiner não seja largo o suficiente. O exemplo da **Figura 4** mostra o uso desse gerenciador.

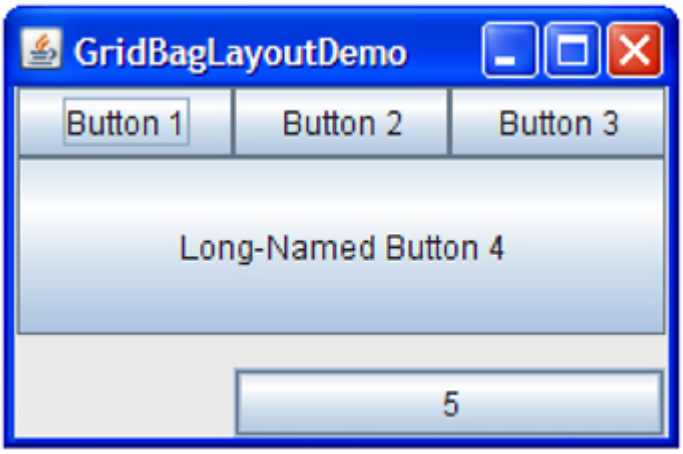
Figura 04 - Utilizando o FlowLayout



GridBagLayout

O GridBagLayout é um gerenciador sofisticado e flexível. Ele alinha componentes, posicionando-os dentro de uma grade de células, permitindo aos componentes ocupar mais de uma célula. As linhas na grade podem ter alturas diferentes, assim como as colunas podem ter larguras diferentes, conforme mostra a Figura 5.

Figura 05 - Exemplo de GridBagLayout

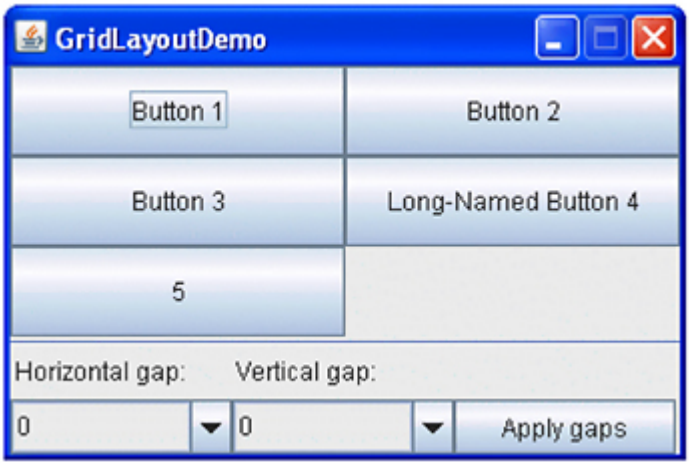


Fonte: http://download.oracle.com/javase/tutorial/uiswing/layout/gridbag.html. Acesso em: 8 maio 2012.

GridLayout

O **GridLayout**, simplesmente, cria um grupo de componentes iguais em tamanho e os mostra em uma quantidade especificada de linhas e colunas semelhante a uma tabela. Veja um exemplo na **Figura 6**:

Figura 06 - Exemplo usando o GridLayout

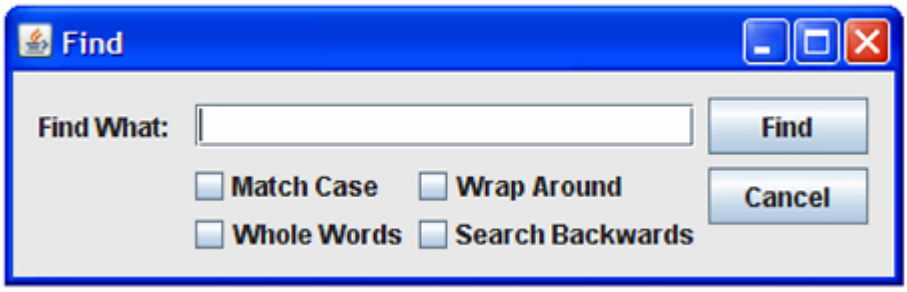


Fonte: http://download.oracle.com/javase/tutorial/uiswing/layout/grid.html. Acesso em: 8 maio 2012.

GroupLayout

O **GroupLayout** é um gerenciador que foi desenvolvido para ser usado por ferramentas de construção de interfaces gráficas (como o Netbeans), mas ele também pode ser usado manualmente. O **GroupLayout** trabalha com os **layouts** horizontal e vertical, separadamente. O **layout** é definido para cada dimensão, independentemente. A **Figura 7** mostra um exemplo desse gerenciador:

Figura 07 - Exemplo de GoupLayout

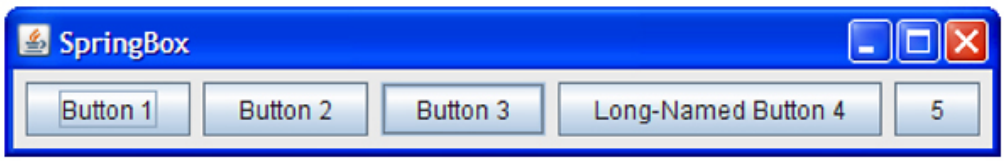


Fonte: http://download.oracle.com/javase/tutorial/uiswing/layout/groupExample.html. Acesso em: 8 maio 2012.

SpringLayout

O **SpringLayout** é um gerenciador flexível que também foi projetado para ser usado com ferramentas de construção de interfaces gráficas. Ele permite que você especifique precisamente a relação entre as bordas dos componentes sob seu controle. Por exemplo, você pode definir que a borda esquerda de um determinado componente ficará a certa distância da borda direita de um segundo componente. A **Figura 8** mostra um exemplo do uso do **SpringLayout**.

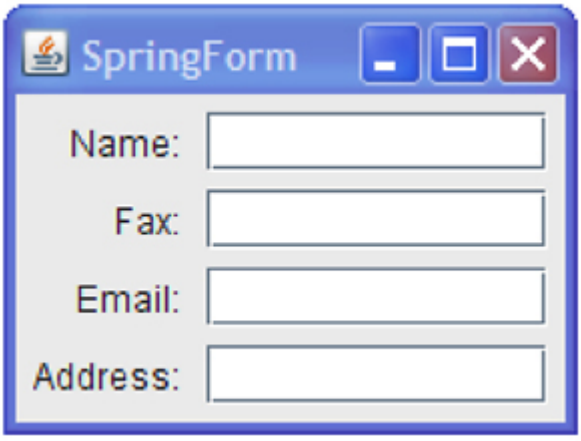
Figura 08 - Exemplo de SpringLayout



Fonte: http://download.oracle.com/javase/tutorial/uiswing/layout/spring.html. Acesso em: 8 maio 2012.

Outro exemplo do **SpringLayout** pode ser visualizado na **Figura 9**.

Figura 09 - Segundo exemplo de SpringLayout



Fonte: http://download.oracle.com/javase/tutorial/uiswing/layout/spring.html. Acesso em: 8 maio 2012.

Atividade 01

1. O que são gerenciadores de **Layout** e para que eles servem?
2. Quais as consequências de não se utilizar um gerenciador de **layout**?
3. Que tipo de gerenciador de layout podemos utilizar de forma tabular?
4. Quais são os gerenciadores de **layout** que foram criados para serem utilizados por uma ferramenta de construção de interfaces gráficas?

Projeto Agenda Pessoal

Como você já sabe o que são gerenciadores de layouts e para que eles servem, vejamos como criar um projeto simples utilizando-os.

O nosso primeiro passo é a criação de um novo projeto, onde ficarão armazenados todos os arquivos envolvidos na nossa aplicação. Uma aplicação Java pode ser formada por vários projetos, mas, no nosso caso, faremos uma aplicação simples contida inteiramente em um projeto apenas. Nesse exemplo, criaremos uma agenda pessoal bem simples para você ter uma ideia de como funciona um gerenciador de layout na prática, sem você ter que se preocupar como é feito. Para isso:

1. Selecione **Arquivo > Novo Projeto**. Você poderá utilizar também o ícone **Novo Projeto**, na barra de ferramentas do Netbeans.

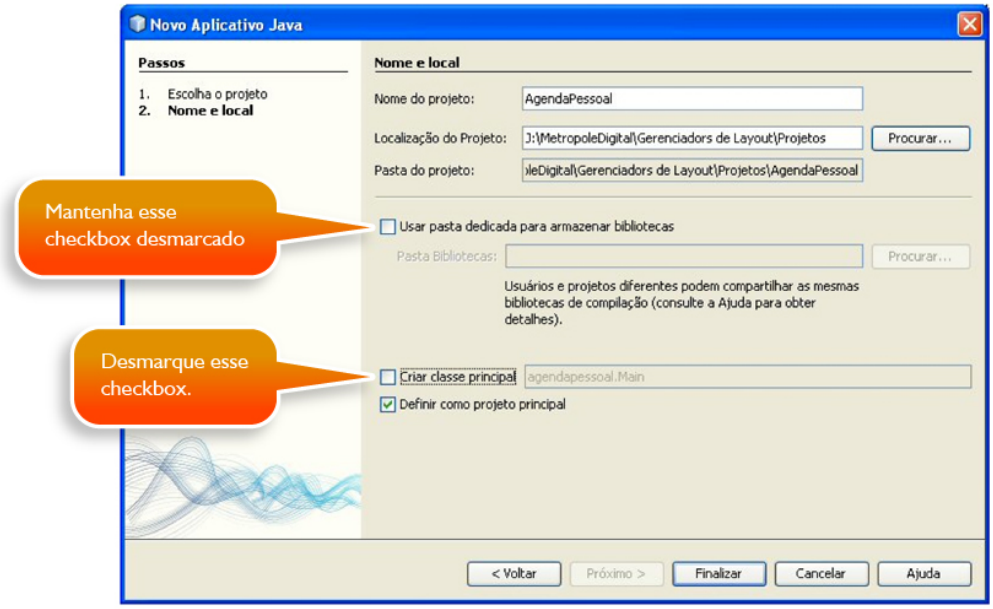
2. No painel de categorias, selecione **Java**, e no painel **Projetos**, escolha **Aplicativo Java**. Clique no botão **Próximo**.

3. Digite **AgendaPessoal** no campo nome do projeto e especifique a localização dele.

4. Deixe a opção **Usar pasta dedicada para armazenar bibliotecas desmarcada**.

5. Certifique-se que a opção **Definir como Projeto Principal** esteja marcada e desmarque a opção **Criar Classe Principal**.

Figura 10 - Janela para a criação da aplicação Agenda Pessoal



6. Clique em **Finalizar**.

O Netbeans cria a pasta com o nome do projeto no lugar especificado. Essa pasta contém todos os arquivos associados ao projeto, incluindo pastas para armazenar os códigos-fonte e para testes, metadados do projeto e outros arquivos. Para ver a estrutura do projeto, use a janela **Arquivos** do Netbeans.

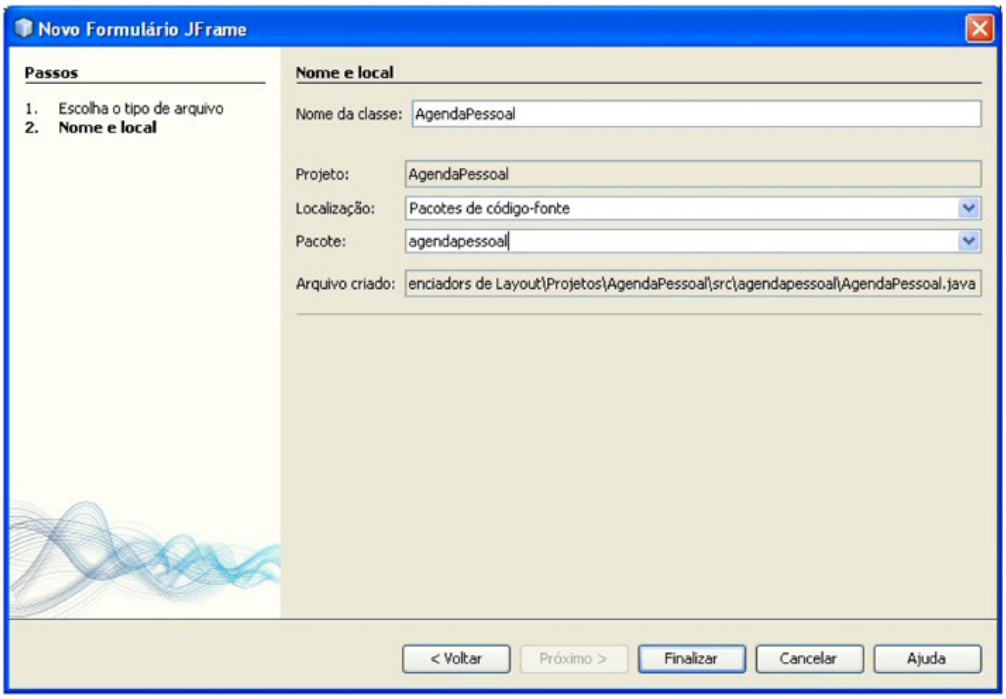
Criando um Contêiner JFrame

Depois de criar uma nova aplicação, você deve ter observado que a pasta **Pacotes de código-fonte** na janela **Projetos** contém um nó vazio chamado **< pacote padrão >**. Para proceder com a criação da nossa interface, é preciso criar um contêiner Java onde serão inseridos os outros componentes visuais necessários para a aplicação. Nesse passo, nós iremos criar um contêiner usando o componente **JFrame**, como foi feito em aulas anteriores, e colocando-o em um novo pacote.

Para adicionar um contêiner JFrame:

1. No painel **Projetos** do NetBeans, clique com o botão direito do mouse no nome do projeto recém-criado, ou seja, **AgendaPessoal**.
2. Digite **AgendaPessoal** para o nome da classe.
3. Digite **agendapessoal** para o nome do pacote. Veja **Figura 11**:

Figura 11 - Janela para a criação do formulário JFrame



O Netbeans cria o formulário e a classe **AgendaPessoal** dentro do arquivo **AgendaPessoal.java** e abre o formulário no construtor de Interfaces Gráficas. Perceba que o pacote **agendapessoal** substitui o pacote padrão **<pacote padrão>**.

Conceitos-Chave

O construtor de interfaces gráficas resolve um problema central na criação de interfaces com o usuário em Java, libertando os desenvolvedores das complexidades dos gerenciadores de layout apresentados no início deste capítulo. Ele consegue isso estendendo o Netbeans para esse suportar o simples paradigma **Free Design** com regras simples de layout que são fáceis de entender e usar. À medida que você projeta a interface gráfica, o Netbeans provê linhas guias, sugerindo espaçamento e alinhamento ótimos para os componentes. Ao mesmo tempo, o Netbeans traduz suas decisões de projeto para uma UI (**User Interface**) funcional, que é implementada usando o novo gerenciador de layouts, o **GroupLayout**, juntamente com outros recursos do Swing. Pelo fato do layout seguir um modelo dinâmico, as interfaces gráficas construídas com o Netbeans se comportam da mesma forma que foram criadas durante o projeto, quando a aplicação é executada, sem alterar a posição original dos componentes, se ajustando para acomodar as mudanças realizadas. Sempre que você redimensiona o formulário, muda localidades (alterando rótulos para outras línguas), ou especificando um **look and feel** diferente, sua interface gráfica, automaticamente, se ajusta para respeitar a aparência definida considerando as particularidades de cada **look and feel**.

Free Design

No construtor de interfaces gráficas, você pode construir seus formulários, simplesmente, colocando seus componentes onde você deseja, como se estivesse usando posicionamento absoluto. O construtor de interfaces gráficas entende que atributos de layout são necessários e gera o código correspondente de forma automática. Você não precisa se preocupar com inserções, âncoras, preenchimentos e assim por diante.

Posicionamento Automático de Componentes (Snapping)

À medida que você insere componentes em um formulário, o construtor de interfaces provê uma interação visual com o programador, que auxilia no posicionamento de componentes baseado nas características visuais (**look and feel**) do seu sistema operacional. O construtor fornece dicas úteis à medida que você está posicionando os componentes e dá um retorno visual sobre onde os componentes devem ser colocados no formulário, forçando os componentes a se posicionar ao longo das linhas de orientação.

Visual Feedback

O Construtor de GUI também fornece **feedback** visual sobre a fixação de componentes e as relações de encadeamento. Esses indicadores permitem que você rapidamente identifique as várias relações de posicionamento e o comportamento de fixação de componentes que afeta a maneira como sua interface irá aparecer e se comportar em tempo de execução. Isso acelera o processo de criação de GUI's (interfaces gráficas), possibilitando que você crie rapidamente interfaces visuais com aparência profissional que realmente funcionam.

Construindo a Interface

Agora que você já está familiarizado com o construtor de interfaces do Netbeans, é hora de começar a desenvolver a UI (**User Interface**, ou interface com o usuário) da nossa **Agenda Pessoal**. Nesta seção, usaremos a janela **Paleta** para adicionar os vários componentes que precisaremos ao formulário.

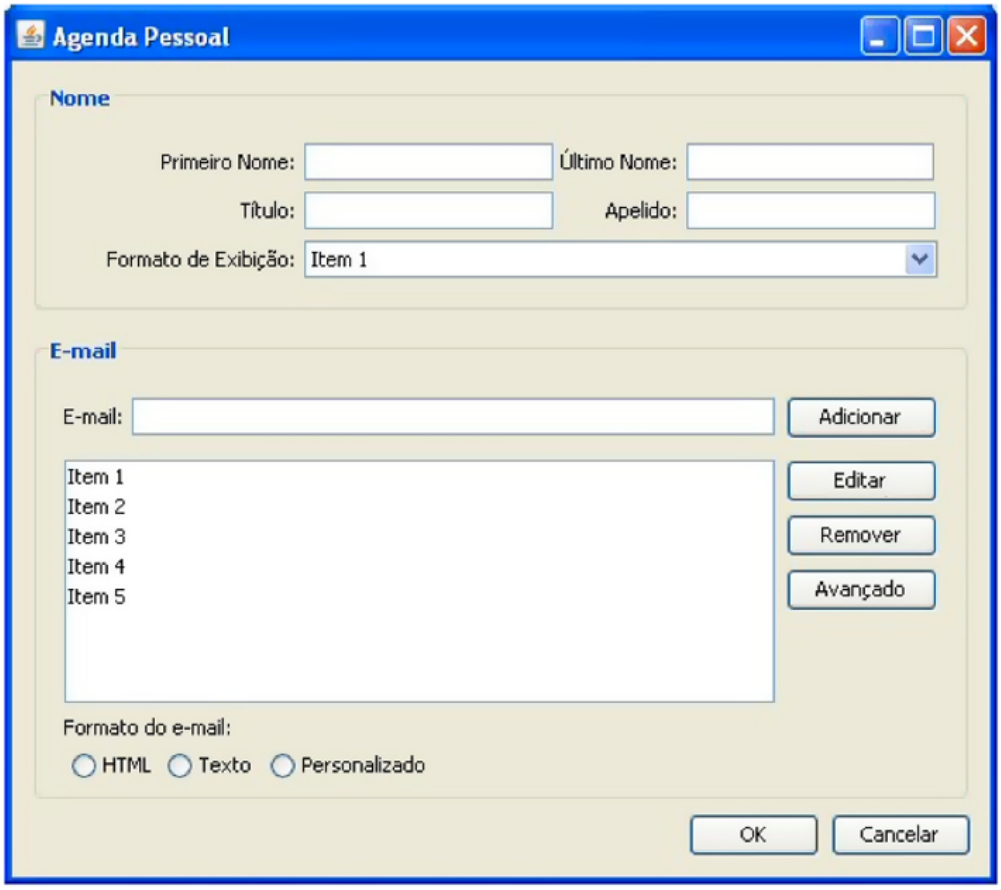
Graças ao paradigma **Free Design** do Netbeans, você não precisa quebrar a cabeça com os gerenciadores de layout para controlar o tamanho e a posição dos componentes dentro dos contêineres. Tudo que você precisa fazer é arrastar e soltar os componentes que você precisa no formulário como mostrado nas ilustrações a seguir.

Adicionando Componentes

Embora o construtor de GUIs simplifique o processo de criação de interfaces em Java, é frequentemente útil fazer um rascunho de como você deseja que a sua interface se pareça antes de começar a realmente construí-la. Muitos projetistas de interfaces consideram essa uma boa prática de programação.

No nosso caso, mostramos, na **Figura 12**, como deverá ficar a interface gráfica da agenda pessoal, quando o projeto estiver concluído.

Figura 12 - Interface gráfica final da Agenda Pessoal



Considerando que nós já criamos um **JFrame**, que é o nosso **contêiner** principal, vamos agora adicionar alguns painéis (componentes **JPanel**) que tornará possível o agrupamento de componentes em contêineres menores representados

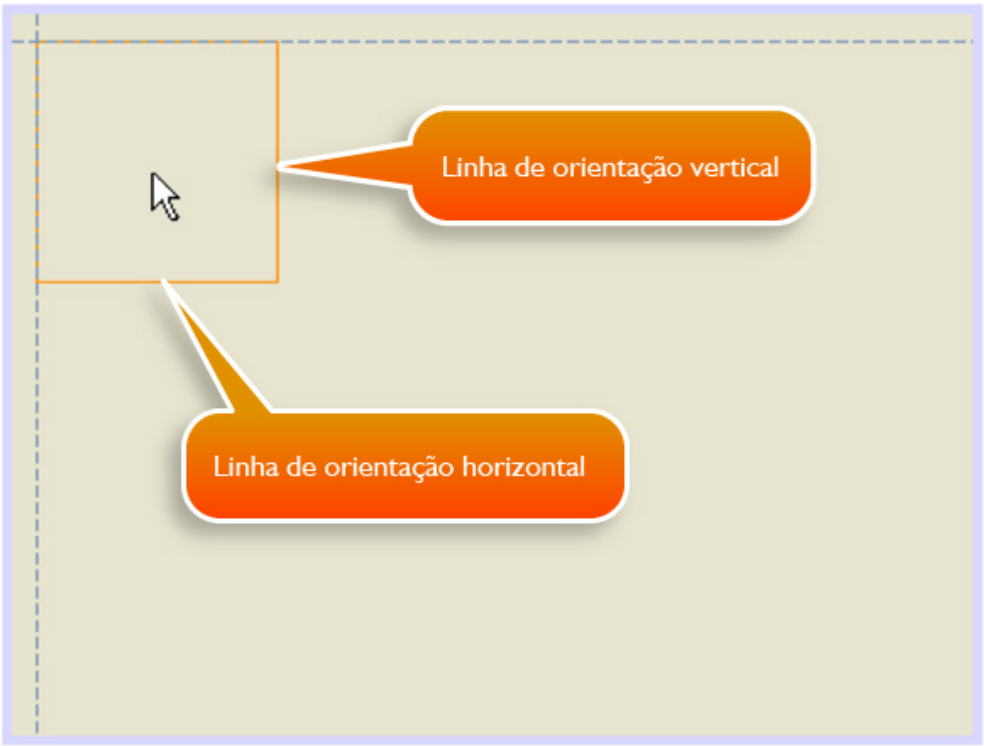
graficamente por uma borda com um título, como visto na **Figura 12**. Mas antes de iniciarmos a inserir os componentes no **JPanel**, devemos criar o título da nossa aplicação. Para isso:

1. Clique no **JFrame** e, no painel **Propriedades**, clique na propriedade **title** e digite: **Agenda Pessoal**.

Para adicionar o **JPanel**:

1. Na janela **Paleta**, selecione o componente **Painel**, na categoria **Contêineres Swing**, clicando e soltando o botão do mouse.
2. Mova o cursor do mouse para o canto superior esquerdo, como mostrado na **Figura 13**. Quando o componente estiver localizado próximo às bordas esquerda e superior e as linhas de orientação horizontal e vertical aparecerem, clique no formulário para confirmar a posição do **JPanel** no **JFrame**.

Figura 13 - Posicionando um componente no JFrame



O **JPanel** aparece no formulário destacado por uma linha laranja, indicando que ele está selecionado. Depois de soltar o botão, pequenas alças aparecerão indicando a relação de fixação do componente. Além disso, um novo nó é mostrado na janela **Inspetor**, conforme mostra a **Figura 14**.

Figura 14 - Componente JPanel adicionado ao JFrame

Redimensionando JPanel

Agora, precisamos redimensionar o **JPanel** para que seja possível inserir outros componentes dentro dele nos próximos passos. Mas antes disso vamos nos concentrar em outros recursos de visualização do construtor de interfaces. Primeiro, tire a seleção do **JPanel** que você acabou de inserir. Note que o **Painel** fica invisível no **JFrame**. Isso acontece porque ainda não adicionamos uma borda ao componente. Mas perceba que, ao movermos o cursor do mouse sobre o **JPanel**, suas bordas mudam de cor para cinza claro e sua posição pode ser claramente vista. Basta que você clique em qualquer lugar dentro do componente para selecioná-lo fazendo com que reapareçam os manipuladores de redimensionamento e fixação.

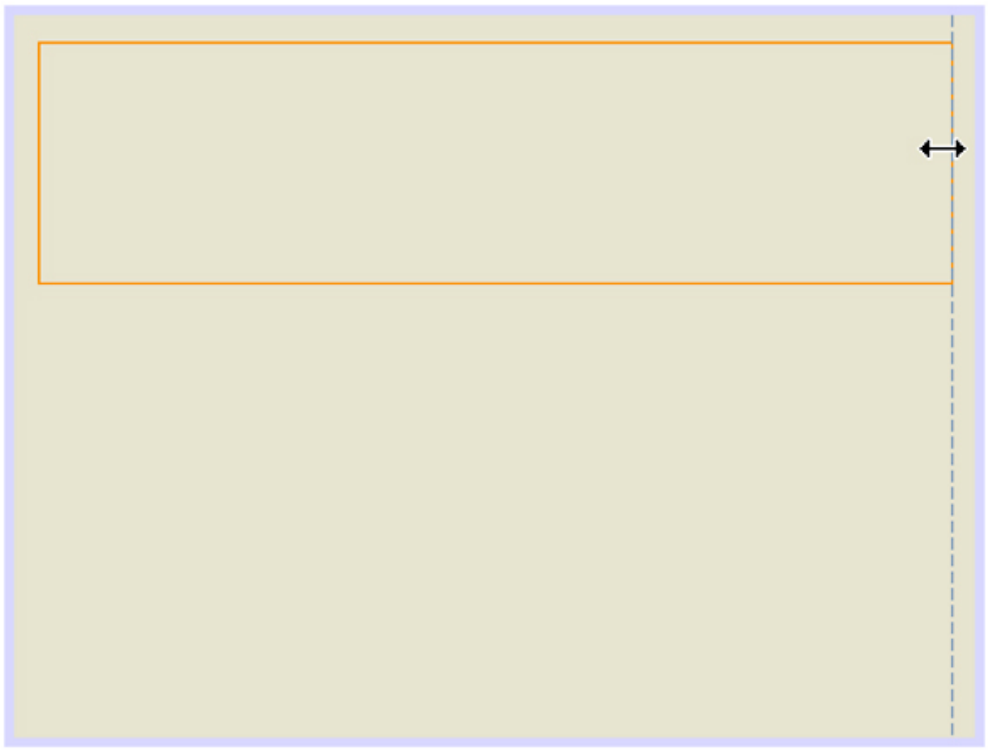
Para redimensionar o JPanel:

1. Selecione o **JPanel** que você acabou de adicionar. Os pequenos quadrados indicando os manipuladores de redimensionamento reaparecem ao redor do componente.
2. Clique e segure um dos manipuladores de redimensionamento no canto direito do **JPanel** e arraste até que a linha guia tracejada apareça próxima a borda do formulário à direita.

3. Solte o botão do mouse liberando o manipulador de redimensionamento para redimensionar o **JPanel**.

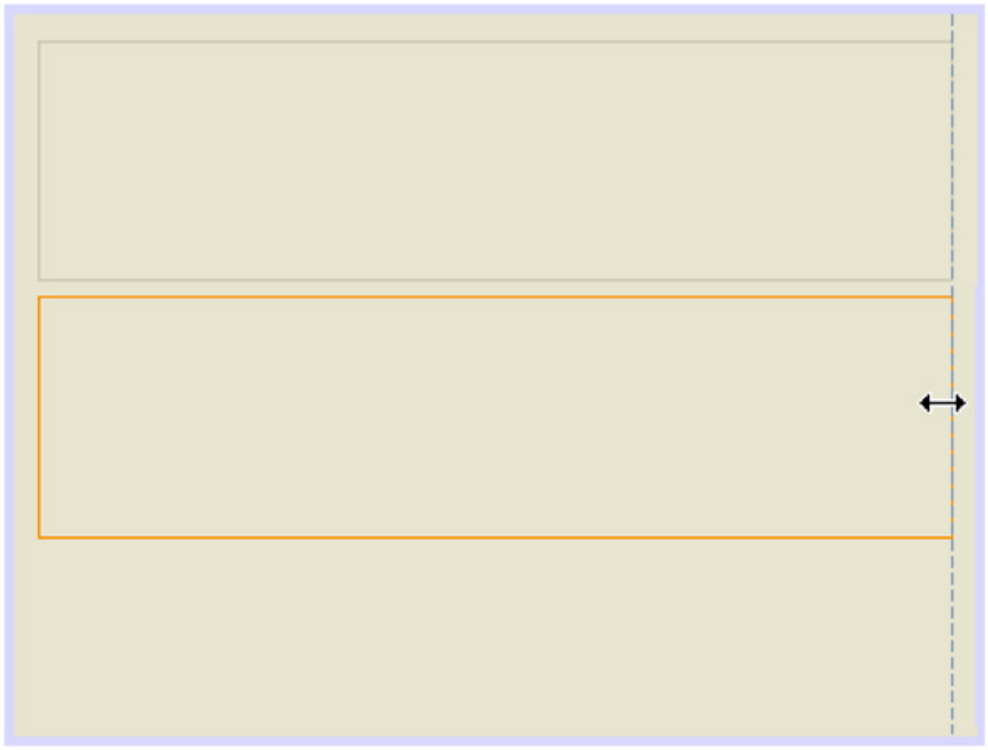
O **JPanel** é estendido para ocupar a área entre as bordas laterais do formulário obedecendo as margens recomendadas, como mostrado na **Figura 15**:

Figura 15 - Redimensionando o JPanel



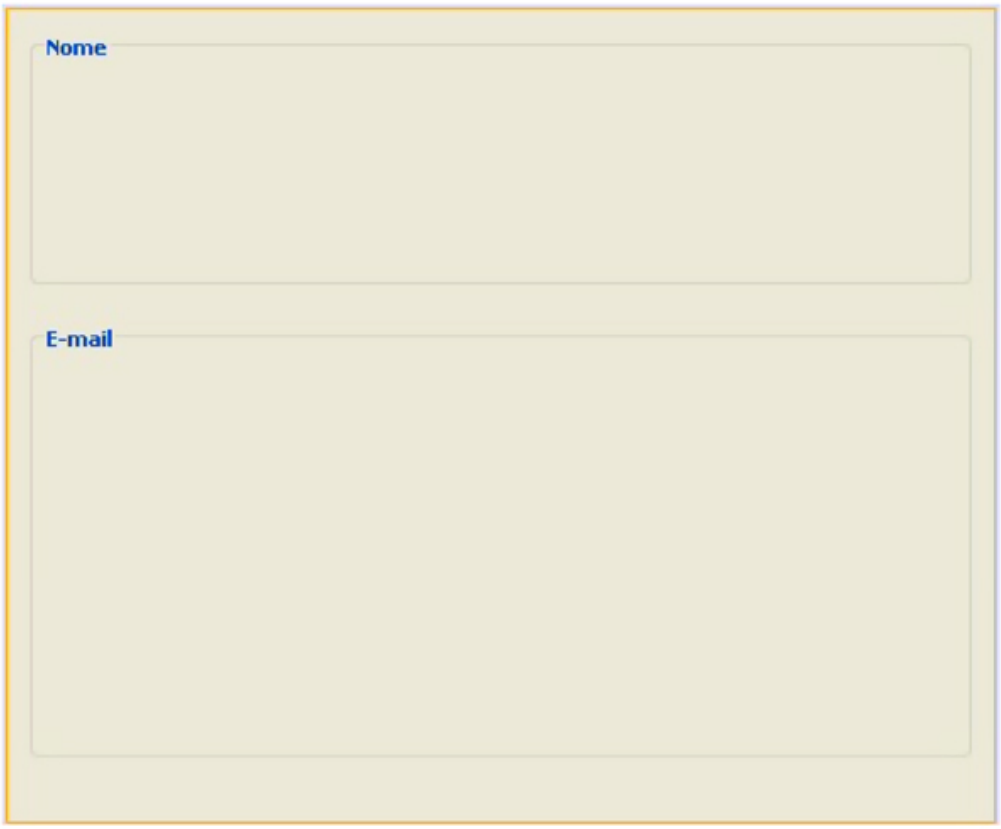
Agora que já adicionamos um painel para receber informações relativas aos nomes da nossa agenda, precisamos repetir o processo para adicionarmos outro painel abaixo do primeiro, que conterá as informações de e-mail da agenda. Guiando-se pela ilustração seguinte, siga os passos anteriores, prestando atenção nas sugestões de posicionamento do Netbeans. Note que o espaço sugerido entre os **JPanel's** é bem menor que o das bordas. Uma vez adicionado o segundo **JPanel**, redimensione-o de forma que ele ocupe toda a área vertical disponível, conforme mostra a **Figura 16**:

Figura 16 - Redimensionando o segundo JPanel



Como queremos distinguir, visualmente, as funções das seções de baixo e de cima da nossa interface gráfica, precisamos adicionar uma borda e um título a cada **JPanel**. Para fazer isso, basta que você altere a propriedade **border** para **Borda de título**. Altere a propriedade **Título** para **Nome** e selecione negrito em estilo de fonte e **12** para o tamanho. Para o segundo **JPanel**, siga as mesmas instruções alterando apenas o **Título** que deve ser **E-mail**, e redimensione-o verticalmente de forma que fique com mais ou menos o dobro do primeiro painel. Confira na **Figura 17** como deverá ficar seu projeto:

Figura 17 - Projeto com os dois painéis (JPanel) configurados



Por enquanto é só. Na próxima aula daremos continuidade à criação de nossa agenda, preenchendo os painéis com vários tipos de componentes e organizando-os de forma que fique uma aplicação com layout profissional. Enquanto isso, tente preencher um dos painéis sem a ajuda da aula seguinte, guiando-se apenas pela **Figura 12**, e depois confira com a próxima aula para ver se chegou perto ou se fez tudo direitinho. Até lá.

Atividade 02

---

1. Qual a finalidade de utilizarmos painéis (JPanel) em uma aplicação?
2. Que gerenciador de layout posiciona os componentes em até cinco áreas distintas: no topo, na base, à esquerda, à direita e no centro?

Resumo

Nesta aula, tivemos uma visão geral sobre gerenciadores de layout, para que são utilizados, os tipos de gerenciadores e como funcionam. Você vai utilizar essa ferramenta de acordo com o ajuste necessário à sua interface, quando houver necessidade de um redimensionamento. Através do tipo de gerenciador, você define a organização dos componentes dentro da área do JFrame, de forma a manter uma interface agradável para o usuário, contando com uma possível variação desse espaço. Em seguida, iniciamos o projeto de Agenda Pessoal para mostrar como utilizar os gerenciadores de layout, sem se preocupar com os detalhes, deixando essa tarefa para o NetBeans. Na próxima aula, daremos continuidade a esse assunto, abordando mais detalhes.

Autoavaliação

1. Qual a principal função dos gerenciadores de layout?
2. Cite três tipos de gerenciadores, definindo, de forma sucinta, como eles funcionam.
3. Que tipo de gerenciador de layout organiza os componentes em uma linha, começando uma nova linha, caso o contêiner não seja largo o suficiente?

Referências

THE JAVA tutorials. **Lesson**: Laying Out Components Within a Container. Disponível em: <http://download.oracle.com/javase/tutorial/uiswing/layout/index.html>. Acesso em: 8 maio 2012a.

_____. **How to Use BorderLayout**. Disponível em: <http://download.oracle.com/javase/tutorial/uiswing/layout/border.html>. Acesso em: 22 abr. 2012b.

_____. **How to Use BoxLayout**. Disponível em: <http://download.oracle.com/javase/tutorial/uiswing/layout/box.html>. Acesso em: 22 abr. 2012c.

_____. **How to Use CardLayout**. Disponível em: <http://download.oracle.com/javase/tutorial/uiswing/layout/card.html>. Acesso em: 22 abr. 2012d.

_____. **How to Use FlowLayout**. Disponível em: <http://download.oracle.com/javase/tutorial/uiswing/layout/flow.html>. Acesso em: 22 abr. 2012e.

_____. **How to Use GridBagLayout**. Disponível em: <http://download.oracle.com/javase/tutorial/uiswing/layout/gridbag.html>. Acesso em: 22 abr. 2012f.

_____. **How to Use GridLayout**. Disponível em: <http://download.oracle.com/javase/tutorial/uiswing/layout/grid.html>. Acesso em: 22 abr. 2012g.

_____. **A GroupLayout Example**. Disponível em: <http://download.oracle.com/javase/tutorial/uiswing/layout/groupExample.html>. Acesso em: 22 abr. 2012h.

_____. **How to Use SpringLayout**. Disponível em: <http://download.oracle.com/javase/tutorial/uiswing/layout/spring.html>. Acesso em: 22 abr. 2012i.