CST8221 – Java Application Programming

Unit 3 Part 1 – Layout Managers

Source Link: http://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html

A Visual Guide to Layout Managers

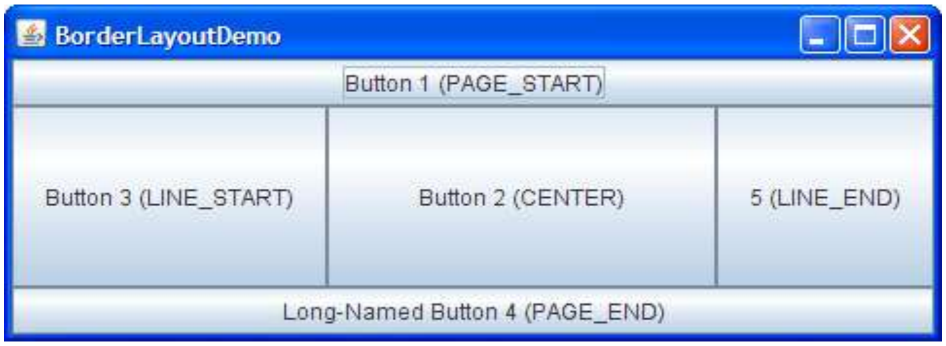
Several AWT and Swing classes provide layout managers for general use:

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

This section shows example GUIs that use these layout managers, and tells you where to find the how-to page for each layout manager. You can find links for running the examples in the how-to pages and in the example index.

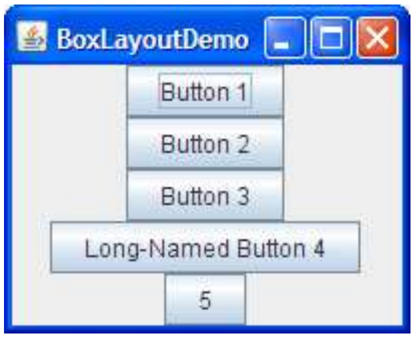
Note: This lesson covers writing layout code by hand, which can be challenging. If you are not interested in learning all the details of layout management, you might prefer to use the `GroupLayout` layout manager combined with a builder tool to lay out your GUI. One such builder tool is the NetBeans IDE. Otherwise, if you want to code by hand and do not want to use `GroupLayout`, then `GridBagLayout` is recommended as the next most flexible and powerful layout manager.

BorderLayout



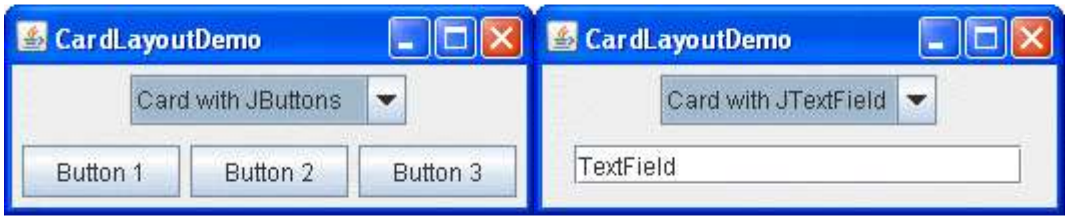
Every content pane is initialized to use a `BorderLayout`. (As Using Top-Level Containers explains, the content pane is the main container in all frames, applets, and dialogs.) A `BorderLayout` places components in up to five areas: top, bottom, left, right, and center. All extra space is placed in the center area. Tool bars that are created using JToolBar must be created within a `BorderLayout` container, if you want to be able to drag and drop the bars away from their starting positions. For further details, see How to Use BorderLayout.

BoxLayout



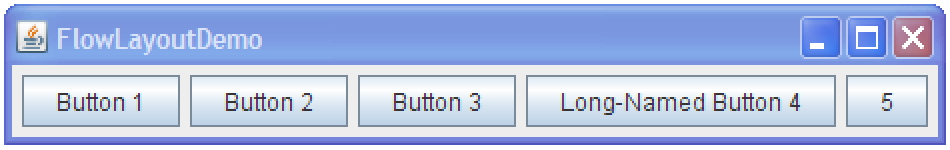
The `BoxLayout` class puts components in a single row or column. It respects the components' requested maximum sizes and also lets you align components. For further details, see How to Use BoxLayout.

CardLayout



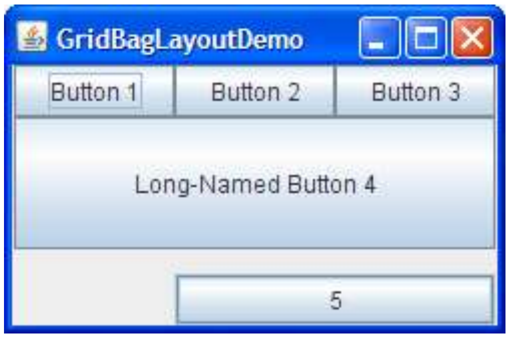
The `CardLayout` class lets you implement an area that contains different components at different times. A `CardLayout` is often controlled by a combo box, with the state of the combo box determining which panel (group of components) the `CardLayout` displays. An alternative to using `CardLayout` is using a tabbed pane, which provides similar functionality but with a pre-defined GUI. For further details, see How to Use CardLayout.

FlowLayout



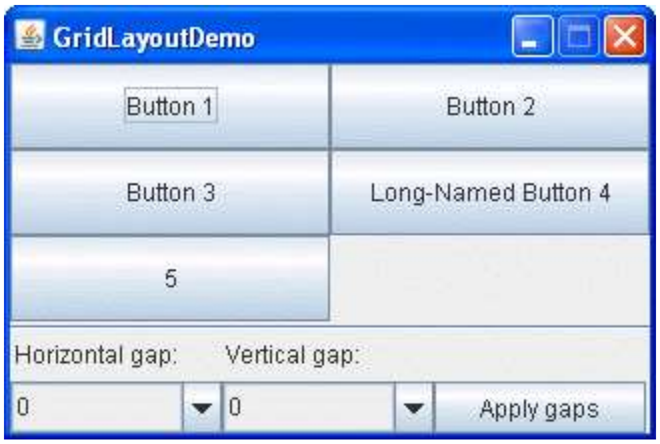
`FlowLayout` is the default layout manager for every `JPanel`. It simply lays out components in a single row, starting a new row if its container is not sufficiently wide. Both panels in CardLayoutDemo, shown previously, use `FlowLayout`. For further details, see How to Use FlowLayout.

GridBagLayout



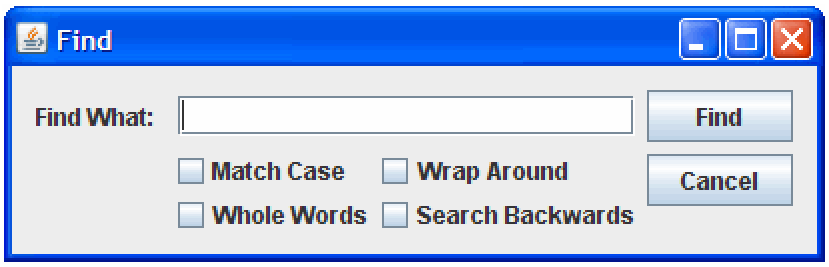
`GridBagLayout` is a sophisticated, flexible layout manager. It aligns components by placing them within a grid of cells, allowing components to span more than one cell. The rows in the grid can have different heights, and grid columns can have different widths. For further details, see How to Use GridBagLayout.

GridLayout



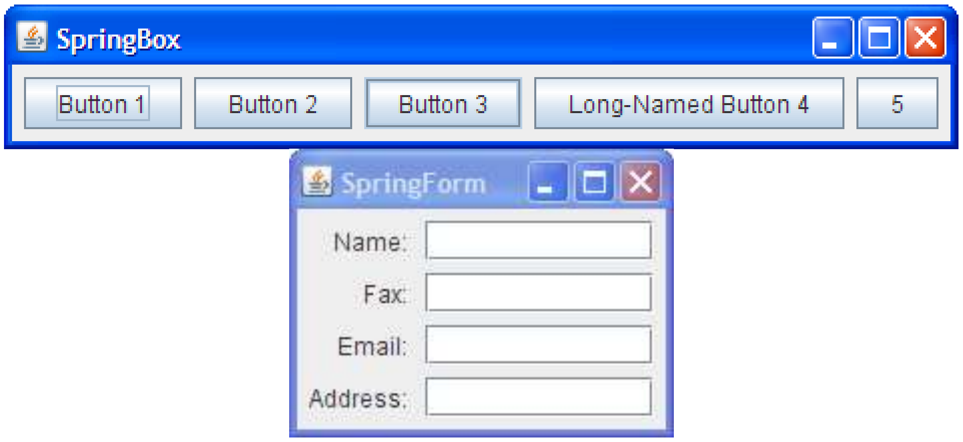
`GridLayout` simply makes a bunch of components equal in size and displays them in the requested number of rows and columns. For further details, see How to Use GridLayout.

GroupLayout



`GroupLayout` is a layout manager that was developed for use by GUI builder tools, but it can also be used manually. `GroupLayout` works with the horizontal and vertical layouts separately. The layout is defined for each dimension independently. Consequently, however, each component needs to be defined twice in the layout. The Find window shown above is an example of a `GroupLayout`. For further details, see How to Use GroupLayout.

SpringLayout



`SpringLayout` is a flexible layout manager designed for use by GUI builders. It lets you specify precise relationships between the edges of components under its control. For example, you might define that the left edge of one component is a certain distance (which can be dynamically calculated) from the right edge of a second component. `SpringLayout` lays out the children of its associated container according to a set of constraints, as shall be seen in [How to Use SpringLayout].

JavaFX provides a similar set of predefined layout manager. In JavaFX they are called panes. For example, BorderPane in JavaFX corresponds to BorderLayout in Swing. The most commonly used JavaFX layout panes are listed below

Class	*Description*
`Pane`	Base class for layout panes. It contains the `getChildren()` method for returning a list of nodes in the pane.
`StackPane`	Places the nodes on top of each other in the center of the pane.
`FlowPane`	Places the nodes row-by-row horizontally or column-by-column vertically.
`GridPane`	Places the nodes in the cells in a two-dimensional grid.
`BorderPane`	Places the nodes in the top, right, bottom, left, and center regions.
`HBox`	Places the nodes in a single row.
`VBox`	Places the nodes in a single column.

For more details, visit the link:

http://docs.oracle.com/javafx/2/layout/builtin_layouts.htm

https://docs.oracle.com/javase/8/javafx/layout-tutorial/index.html