Table of Contents

One can hardly imagine GUI development with layout managers. However, often multiple layout managers are necessary to develop appealing GUIs. These layout mangers are usually nested to achieve the desired results. To facilitate nesting, a panel is required for each layout manager. Before you know it, a large amount of code is created just to define the GUI.

The use of visual designers only eases the positioning of the components; it does not reduce the number of components. Also, the more nesting is used for a GUI, the more sluggish and complex it becomes.

To keep the GUI code at a minimum and to keep it manageable, JVx offers three layout managers that can help accomplish this goal:

- BorderLayout
- FlowLayout
- FormLayout

BorderLayout, unlike *java.awt.BorderLayout*, allows the setting of margins. Otherwise, it offers the usual regions: NORTH, WEST, EAST, SOUTH, and CENTER.

FlowLayout offers the vertical and horizontal alignment of components, the use of margins, and it allows stretching as well as control of the alignment of the components within the layout. It basically supplements *java.awt.FlowLayout* with additional functionality.

FormLayout

As we can guess from the name, this layout manager is used for the creation of forms. However, it can also be used for a variety of other tasks. The basic idea behind this layout manager was to enable the creation of complex forms without nesting or complexity.

The form layout is anchor-based. An anchor defines a position in the GUI and can be positioned using absolute references or relative references to another anchor.

This allows, for example, subdivision of a panel in table form, such as the *java.awt.GridLayout*. In addition, it can also be viewed as the *java.awtGridBagLayout* without the corresponding level of complexity.

The following examples illustrate the use of layout managers.

Simple Example

A form containing two columns (default setting):

The anchors are defined as follows:

```
UIFormLayout layout = new UIFormLayout();

UIPanel panDetail = new UIPanel(layout);
```

```
panDetail.add(new UILabel("Filename:"));
panDetail.add(new UIEditor(rdbFiles, "FILENAME"));
panDetail.add(new UILabel("Type:"));
panDetail.add(new UIEditor(rdbFiles, "TYPE"));
panDetail.add(new UILabel("Size:"));
panDetail.add(new UIEditor(rdbFiles, "FILESIZE"));
panDetail.add(new UILabel("Created:"));
panDetail.add(new UIEditor(rdbFiles, "CREATED"));
panDetail.add(new UILabel("Created by:"));
panDetail.add(new UIEditor(rdbFiles, "CREATED_BY"));
panDetail.add(new UILabel("Changed:"));
panDetail.add(new UIEditor(rdbFiles, "CHANGED"));
panDetail.add(new UILabel("Changed by:"));
panDetail.add(new UIEditor(rdbFiles, "CHANGED_BY"));
```

The following call:

```
layout.setNewlineCount(4);
```

is sufficient to create four columns:

Complex Example

A somewhat more complex GUI is shown below:

and the respective source code:

```
UIFormLayout flDetails = new UIFormLayout();
UIGroupPanel gpanDedails = new UIGroupPanel();

gpanDedails.setLayout(flDetails);
gpanDedails.add(icoImage, flDetails.getConstraints(0, 0, 1, 7));
gpanDedails.add(butLoadImage, flDetails.getConstraints(0, 8));
gpanDedails.add(edtFilename, flDetails.getConstraints(1, 8));

flDetails.setHorizontalGap(15);
gpanDedails.add(lblSalutation, flDetails.getConstraints(2, 0));
flDetails.setHorizontalGap(5);
gpanDedails.add(edtSalutation, flDetails.getConstraints(3, 0));
gpanDedails.add(lblAcademicTitle, flDetails.getConstraints(2, 1));
gpanDedails.add(edtAcademicTitle, flDetails.getConstraints(3, 1));
gpanDedails.add(lblFirstName, flDetails.getConstraints(2, 2));
gpanDedails.add(edtFirstName, flDetails.getConstraints(3, 2, -1, 2));
gpanDedails.add(lblLastName, flDetails.getConstraints(2, 3));
gpanDedails.add(edtLastName, flDetails.getConstraints(3, 3, -1, 3));
```

```
gpanDedails.add(lblSocialSecurityNr, flDetails.getConstraints(2, 4));
gpanDedails.add(edtSocialSecurityNr, flDetails.getConstraints(3, 4));
gpanDedails.add(lblBirthday, flDetails.getConstraints(4, 4));
gpanDedails.add(edtBirthday, flDetails.getConstraints(5, 4, -1, 4));

gpanDedails.add(lblHealthInsurance, flDetails.getConstraints(2, 5));
gpanDedails.add(edtHealthInsurance, flDetails.getConstraints(3, 5, -1, 5));

gpanDedails.add(lblStreet, flDetails.getConstraints(2, 6));
gpanDedails.add(edtStreet, flDetails.getConstraints(3, 6, -3, 6));
gpanDedails.add(lblNr, flDetails.getConstraints(-2, 6));
gpanDedails.add(editNr, flDetails.getConstraints(-1, 6));
gpanDedails.add(lblZip, flDetails.getConstraints(2, 7));
gpanDedails.add(edtZip, flDetails.getConstraints(3, 7));
gpanDedails.add(lblTown, flDetails.getConstraints(4, 7));
gpanDedails.add(edtTown, flDetails.getConstraints(5, 7, -1, 7));
```

Error Dialog

As a final example, we want to show the framework´s error message, this time using a direct anchor:

and the source code:

```
foMain = new UIFormLayout();
foMain.setVerticalAlignment(IAlignmentConstants.ALIGN_TOP);
foMain.setMargins(new UIInsets(7, 7, 7, 7));

setLayout(foMain);

add(icon);
add(taMessage,
    foMain.getConstraints(foMain.getTopMarginAnchor(),
                          foMain.createAnchor(foMain.getConstraints(icon).
                                              getRightAnchor(), 5),
                          null,
                          foMain.getRightMarginAnchor()));
add(butOK,
    foMain.getConstraints(null,
                          null,
                          foMain.getBottomMarginAnchor(),
                          foMain.getRightMarginAnchor()));
add(butDetails,
    foMain.getConstraints(null,
                          null,
                          foMain.getBottomMarginAnchor(),
                          foMain.createAnchor(foMain.getConstraints(butOK).
                                              getLeftAnchor(), -5)));
```

Hint

There's a demo application which allows us to inspect the anchors of the FormLayout. The source code is hosted on github.