Layout and Panel Building in Swing

Karsten Lentzsch
www.JGoodies.com

Presentation Goal

Learn how to layout and implement elegant and consistent panels quickly.

Speaker Qualifications

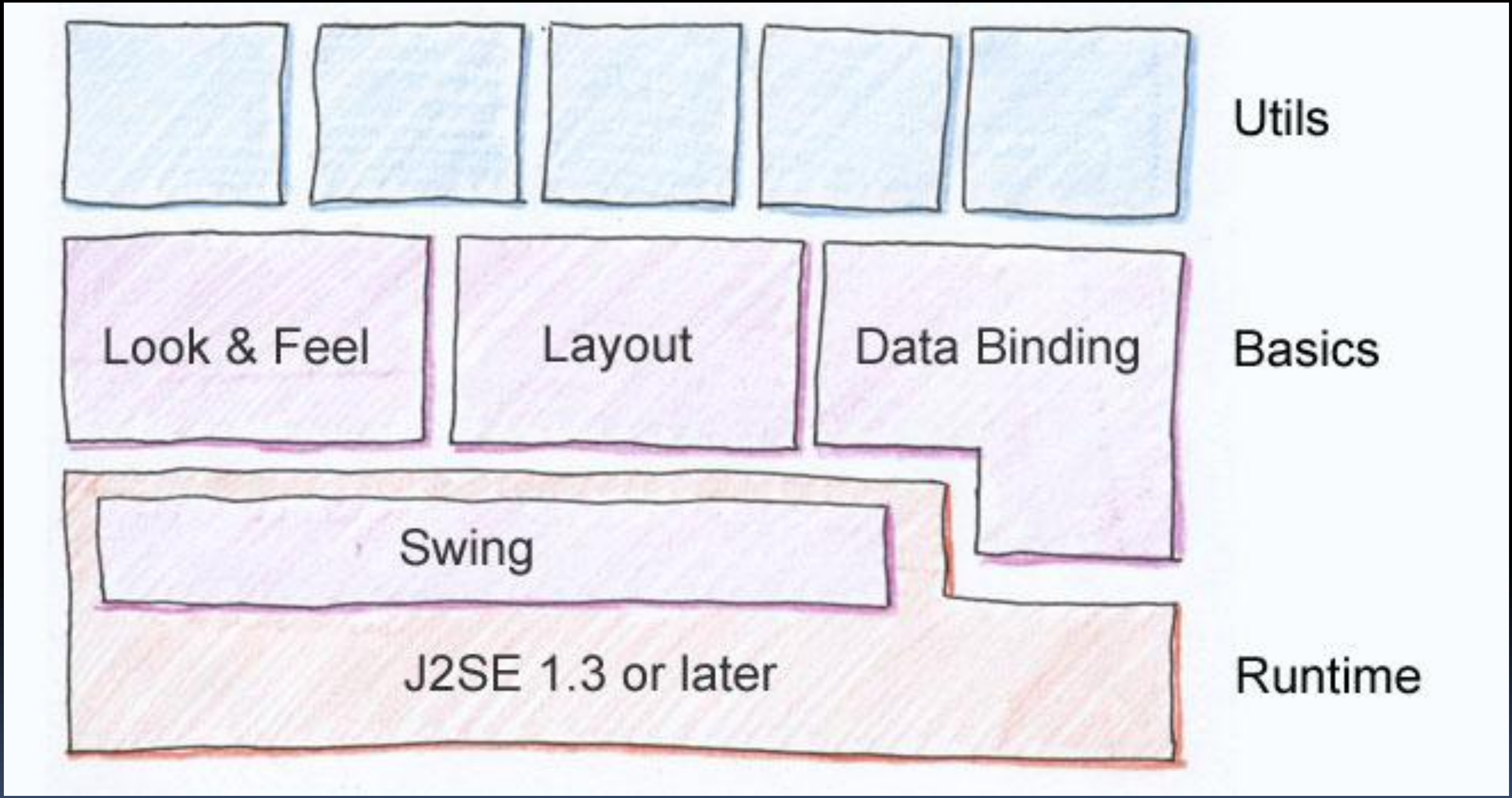
- Karsten builds elegant Swing Apps
- he provides libraries that complement Swing
- he works with emulated looks since 1995
- he assists others in visual design
- he writes about user interface issues

Agenda

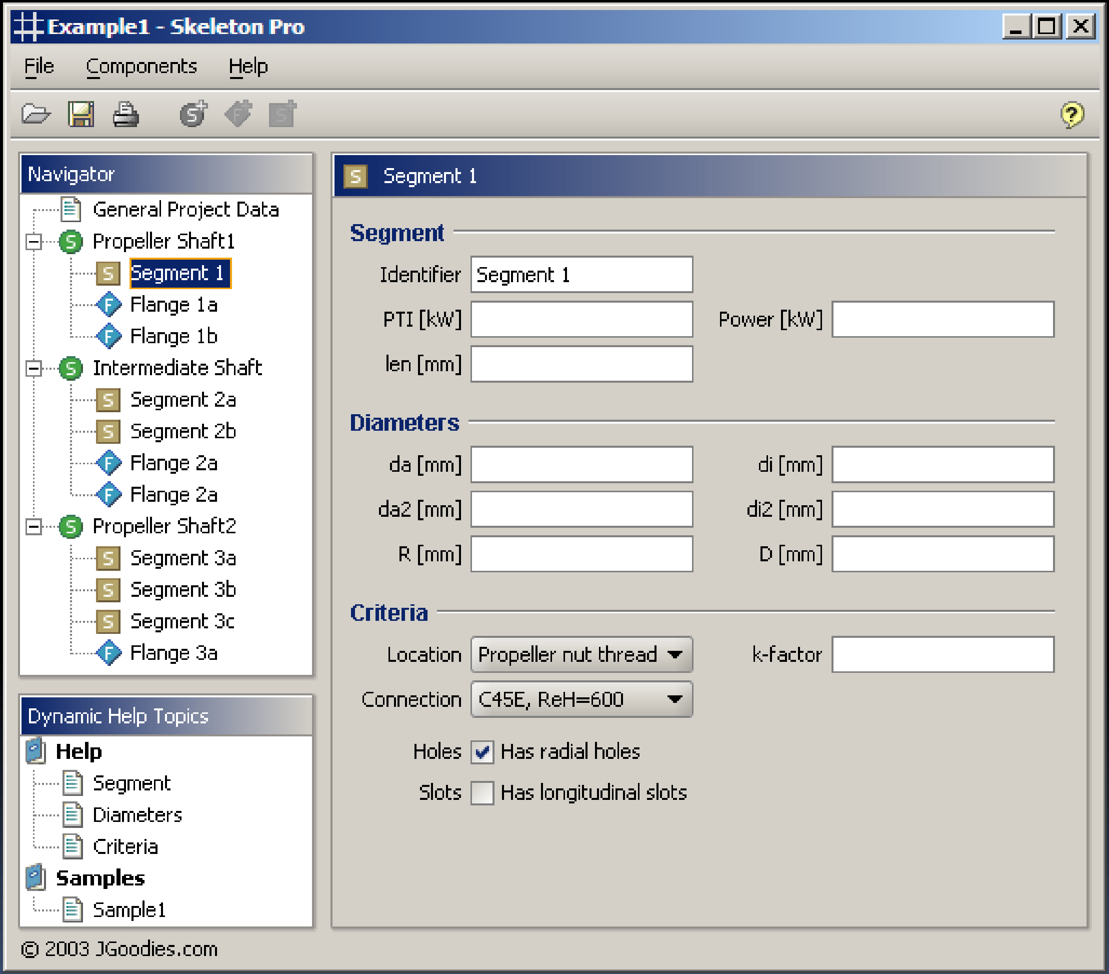
▪ Introduction	
▪ Analysis:	What are the problems?
▪ Goals:	What do we want to have?
▪ Concepts:	How to achieve the goals?
▪ Solution:	How to do layout right?
▪ Future:	What comes next?

I. Introduction

How to succeed with Swing?



We focus on form-oriented Panels



Segment 1

Segment

Identifier	Segment 1		
PTI [kW]		Power [kW]	
len [mm]			

Diameters

da [mm]		di [mm]	
da2 [mm]		di2 [mm]	
R [mm]		D [mm]	

Criteria

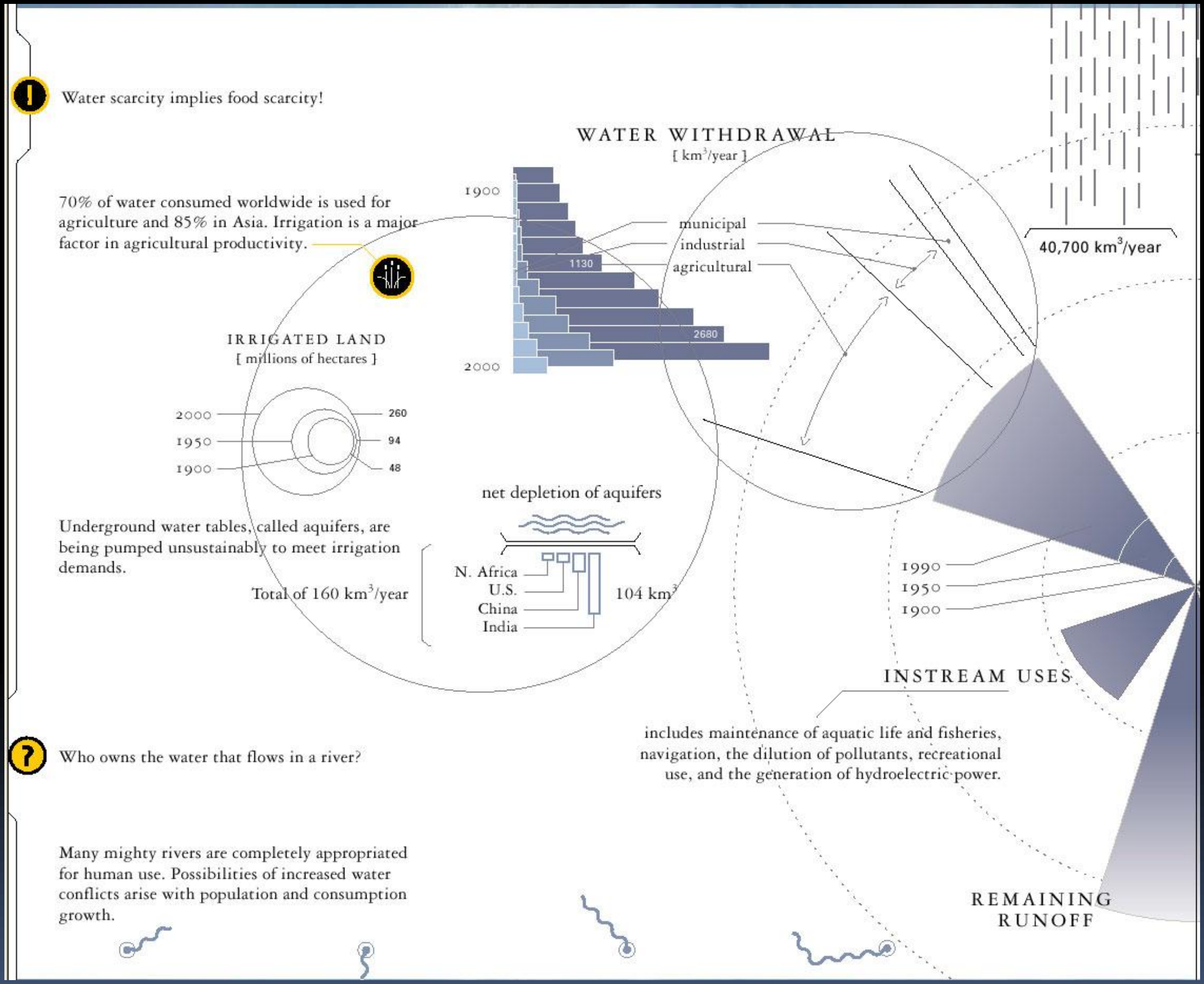
Location	Propeller nut thread	k-factor	
Connection	C45E, ReH=600		
Holes	✓ Has radial holes		
Slots	☐ Has longitudinal slots		

We focus on Usability

We aim to improve:

- Readability
- Legibility
- Elegance (appropriate choice)
- Usability

We focus on Forms – not Art



What is Design?

- To design is:
  - to plan, to control,
  - to assemble, to order, to align,
  - to relate, to scale, to balance,
  - to add value, to simplify
  - to clarify
- Layout is an essential part in GUI design

Layout Roles and Actitivies

- Meta designer defines a style
- (Human) visual designer finds a layout
- Developer constructs the layout
- Builder code adds components to a container
- Layout manager computes and sets bounds

II. Analysis

What are the problems?

We're going to analyze:

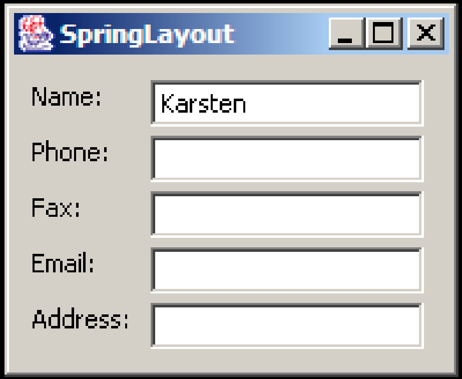
- Problems that humans face
- Essentials for good design
- How to make layout easier

Problems we Face

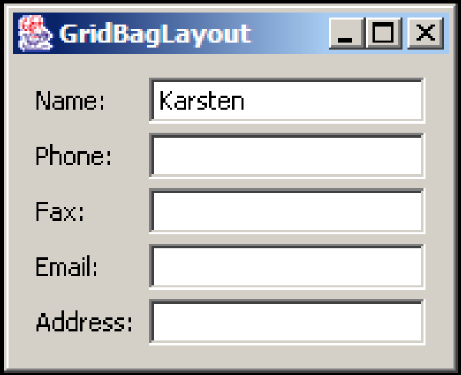
- Tutorials and books demonstrate poor design
- Layout manager is difficult to learn
- Layout manager is difficult to work with
- Layout code is hard to read
- It's difficult to determine a layout from code

Layout Code Length

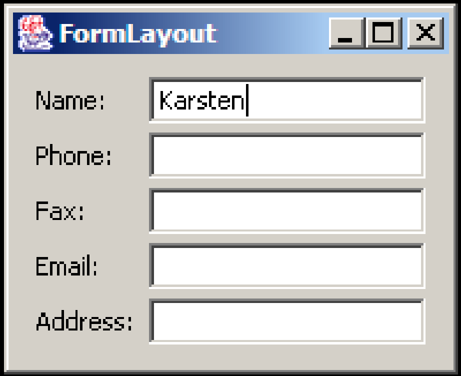
An example from Sun‵s Java tutorial


SpringLayout

Name:	Karsten
Phone:	
Fax:	
Email:	
Address:	

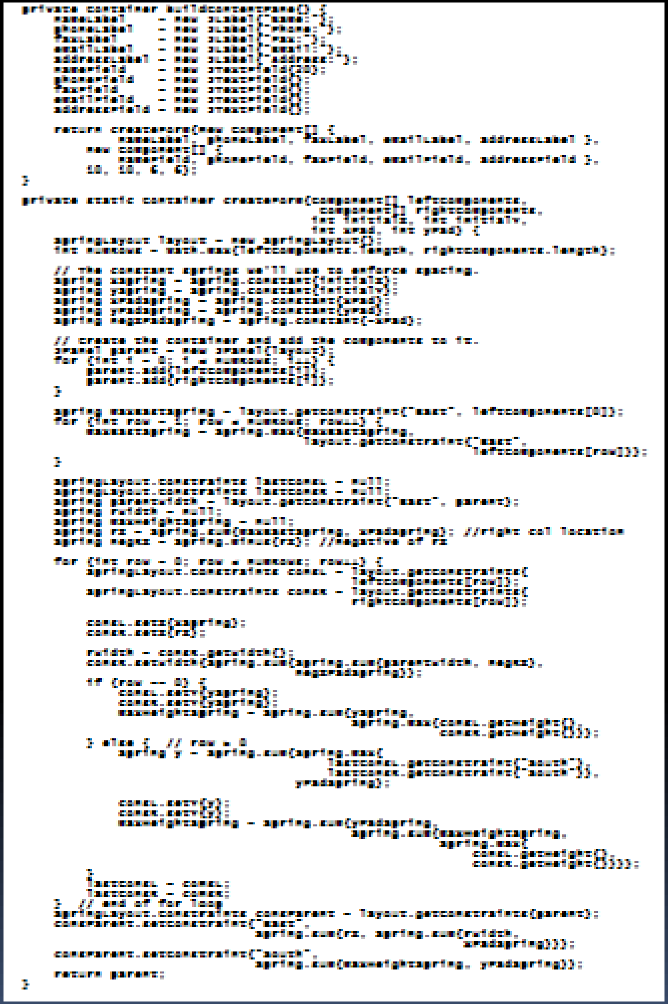

GridBagLayout

Name:	Karsten
Phone:	
Fax:	
Email:	
Address:	

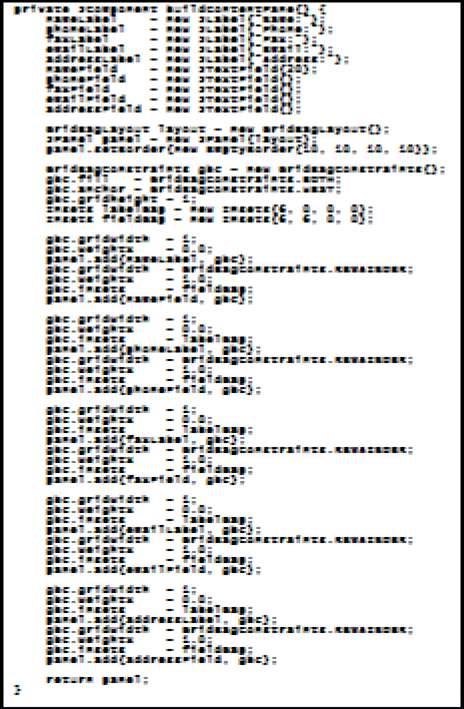

FormLayout

Name:	Karsten
Phone:	
Fax:	
Email:	
Address:	

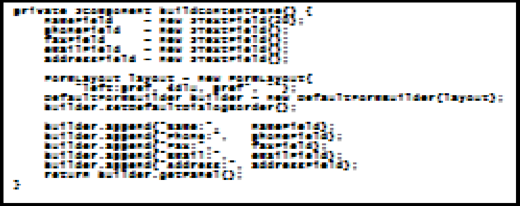
3 Layout Code Styles



```java
private Container buildContentPane() {
    nameLabel    = new JLabel("Name:");
    phoneLabel   = new JLabel("Phone:");
    faxLabel     = new JLabel("Fax:");
    emailLabel   = new JLabel("Email:");
    addressLabel = new JLabel("Address:");
    nameField    = new JTextField(20);
    phoneField   = new JTextField();
    faxField     = new JTextField();
    emailField   = new JTextField();
    addressField = new JTextField();

    return createForm(new Component[] {
            nameLabel, phoneLabel, faxLabel, emailLabel, addressLabel },
        new Component[] {
            nameField, phoneField, faxField, emailField, addressField },
        10, 10, 6, 6);
}

private static Container createForm(Component[] leftComponents,
                                     Component[] rightComponents,
                                    int initialX, int initialY,
                                    int xPad, int yPad) {
    SpringLayout layout = new SpringLayout();
    int numRows = Math.max(leftComponents.length, rightComponents.length);

    // The constant springs we'll use to enforce spacing.
    Spring xSpring = Spring.constant(initialX);
    Spring ySpring = Spring.constant(initialY);
    Spring xPadSpring = Spring.constant(xPad);
    Spring yPadSpring = Spring.constant(yPad);
    Spring negXPadSpring = Spring.constant(-xPad);

    // Create the container and add the components to it.
    JPanel parent = new JPanel(layout);
    for (int i = 0; i < numRows; i++) {
        parent.add(leftComponents[i]);
        parent.add(rightComponents[i]);
    }

    Spring maxEastSpring = layout.getConstraint("East", leftComponents[0]);
    for (int row = 1; row < numRows; row++) {
        maxEastSpring = Spring.max(maxEastSpring,
                                   layout.getConstraint("East",
                                                        leftComponents[row]));
    }

    SpringLayout.Constraints lastConsL = null;
    SpringLayout.Constraints lastConsR = null;
    Spring parentWidth = layout.getConstraint("East", parent);
    Spring rWidth = null;
    Spring maxHeightSpring = null;
    Spring rX = Spring.sum(maxEastSpring, xPadSpring); //right col location
    Spring negRX = Spring.minus(rX); //negative of rX

    for (int row = 0; row < numRows; row++) {
        SpringLayout.Constraints consL = layout.getConstraints(
                                             leftComponents[row]);
        SpringLayout.Constraints consR = layout.getConstraints(
                                             rightComponents[row]);

        consL.setX(xSpring);
        consR.setX(rX);

        rWidth = consR.getWidth();
        consR.setWidth(Spring.sum(Spring.sum(parentWidth, negRX),
                                  negXPadSpring));
        if (row == 0) {
            consL.setY(ySpring);
            consR.setY(ySpring);
            maxHeightSpring = Spring.sum(ySpring,
                                         Spring.max(consL.getHeight(),
                                                    consR.getHeight()));
        } else { // row > 0
            Spring y = Spring.sum(Spring.max(
                                      lastConsL.getConstraint("South"),
                                      lastConsR.getConstraint("South")),
                                  yPadSpring);

            consL.setY(y);
            consR.setY(y);
            maxHeightSpring = Spring.sum(yPadSpring,
                                         Spring.sum(maxHeightSpring,
                                                    Spring.max(
                                                       consL.getHeight(),
                                                       consR.getHeight())));
        }
        lastConsL = consL;
        lastConsR = consR;
    } // end of for loop
    SpringLayout.Constraints consParent = layout.getConstraints(parent);
    consParent.setConstraint("East",
                             Spring.sum(rX, Spring.sum(rWidth,
                                                       xPadSpring)));
    consParent.setConstraint("South",
                             Spring.sum(maxHeightSpring, yPadSpring));
    return parent;
}
```

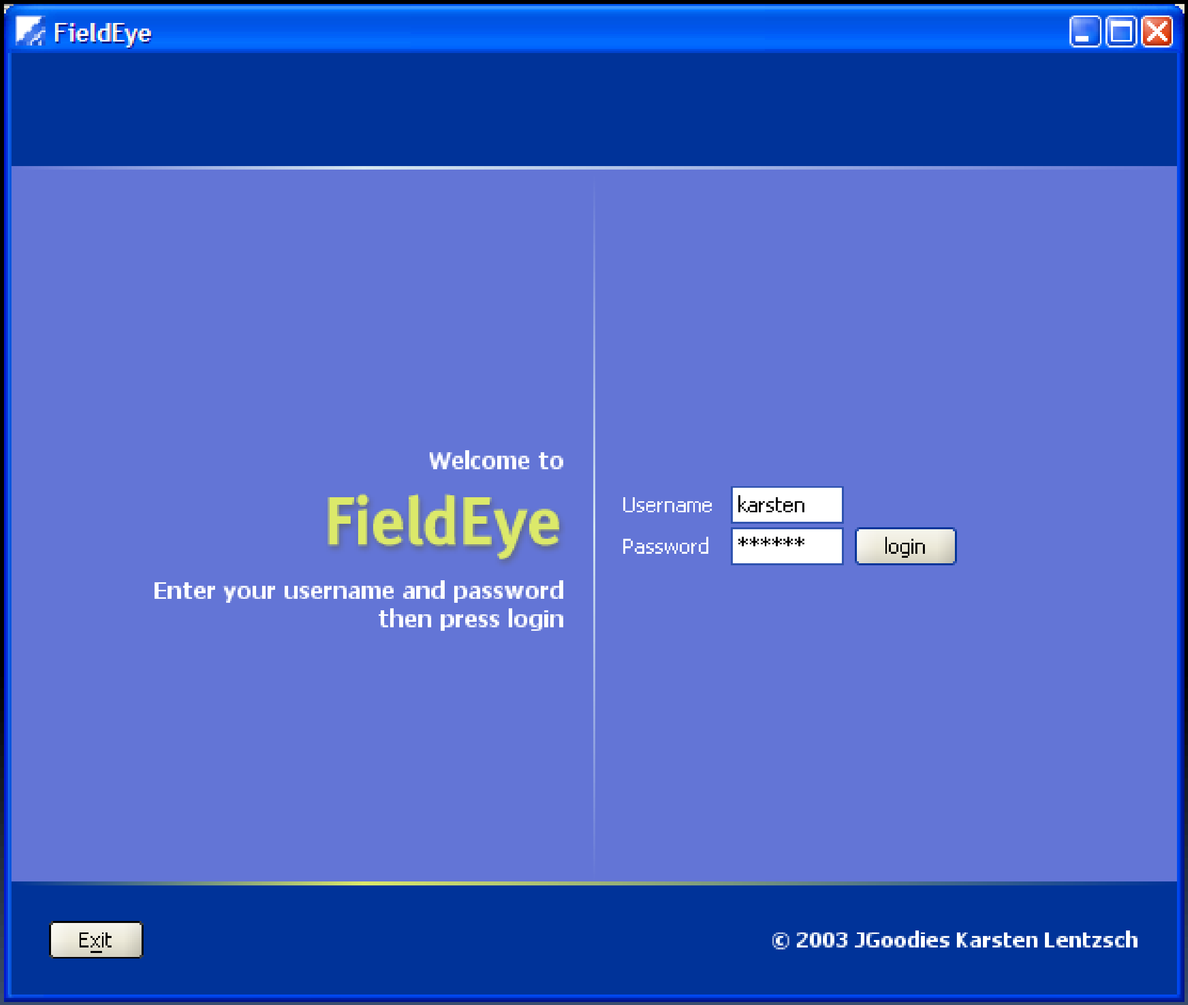


```java
private JComponent buildContentPane() {
    nameLabel    = new JLabel("Name:");
    phoneLabel   = new JLabel("Phone:");
    faxLabel     = new JLabel("Fax:");
    emailLabel   = new JLabel("Email:");
    addressLabel = new JLabel("Address:");
    nameField    = new JTextField(20);
    phoneField   = new JTextField();
    faxField     = new JTextField();
    emailField   = new JTextField();
    addressField = new JTextField();

    GridBagLayout layout = new GridBagLayout();
    JPanel panel = new JPanel(layout);
    panel.setBorder(new EmptyBorder(10, 10, 10, 10));

    GridBagConstraints gbc = new GridBagConstraints();
    gbc.fill   = GridBagConstraints.NONE;
    gbc.anchor = GridBagConstraints.WEST;
    gbc.gridheight = 1;
    Insets labelGap = new Insets(6, 0, 0, 0);
    Insets fieldGap = new Insets(6, 6, 0, 0);

    gbc.gridwidth  = 1;
    gbc.weightx    = 0.0;
    panel.add(nameLabel, gbc);
    gbc.gridwidth  = GridBagConstraints.REMAINDER;
    gbc.weightx    = 1.0;
    gbc.insets     = fieldGap;
    panel.add(nameField, gbc);

    gbc.gridwidth  = 1;
    gbc.weightx    = 0.0;
    gbc.insets     = labelGap;
    panel.add(phoneLabel, gbc);
    gbc.gridwidth  = GridBagConstraints.REMAINDER;
    gbc.weightx    = 1.0;
    gbc.insets     = fieldGap;
    panel.add(phoneField, gbc);

    gbc.gridwidth  = 1;
    gbc.weightx    = 0.0;
    gbc.insets     = labelGap;
    panel.add(faxLabel, gbc);
    gbc.gridwidth  = GridBagConstraints.REMAINDER;
    gbc.weightx    = 1.0;
    gbc.insets     = fieldGap;
    panel.add(faxField, gbc);

    gbc.gridwidth  = 1;
    gbc.weightx    = 0.0;
    gbc.insets     = labelGap;
    panel.add(emailLabel, gbc);
    gbc.gridwidth  = GridBagConstraints.REMAINDER;
    gbc.weightx    = 1.0;
    gbc.insets     = fieldGap;
    panel.add(emailField, gbc);

    gbc.gridwidth  = 1;
    gbc.weightx    = 0.0;
    gbc.insets     = labelGap;
    panel.add(addressLabel, gbc);
    gbc.gridwidth  = GridBagConstraints.REMAINDER;
    gbc.weightx    = 1.0;
    gbc.insets     = fieldGap;
    panel.add(addressField, gbc);

    return panel;
}
```



```java
private JComponent buildContentPane() {
    nameField    = new JTextField(20);
    phoneField   = new JTextField();
    faxField     = new JTextField();
    emailField   = new JTextField();
    addressField = new JTextField();

    FormLayout layout = new FormLayout(
        "left:pref, 4dlu, pref", "");
    DefaultFormBuilder builder = new DefaultFormBuilder(layout);
    builder.setDefaultDialogBorder();

    builder.append("Name:",     nameField);
    builder.append("Phone:",    phoneField);
    builder.append("Fax:",      faxField);
    builder.append("Email:",    emailField);
    builder.append("Address:",  addressField);
    return builder.getPanel();
}
```

Essentials: Symmetry


FieldEye

Welcome to

FieldEye

Enter your username and password
then press login

Username karsten

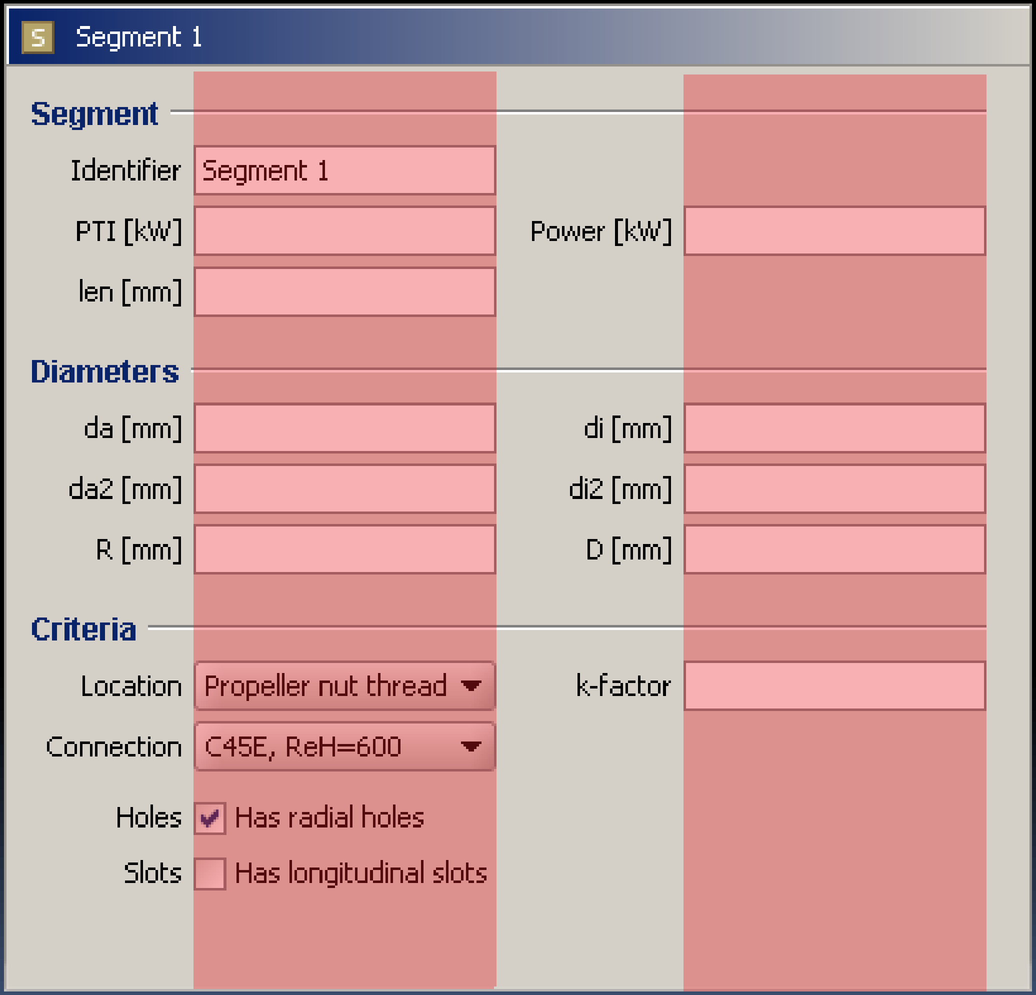
Password ******

login

Exit

© 2003 JGoodies Karsten Lentzsch

Essentials: Equal Widths



Segment 1

Segment

Identifier	Segment 1		
PTI [kW]		Power [kW]	
len [mm]			

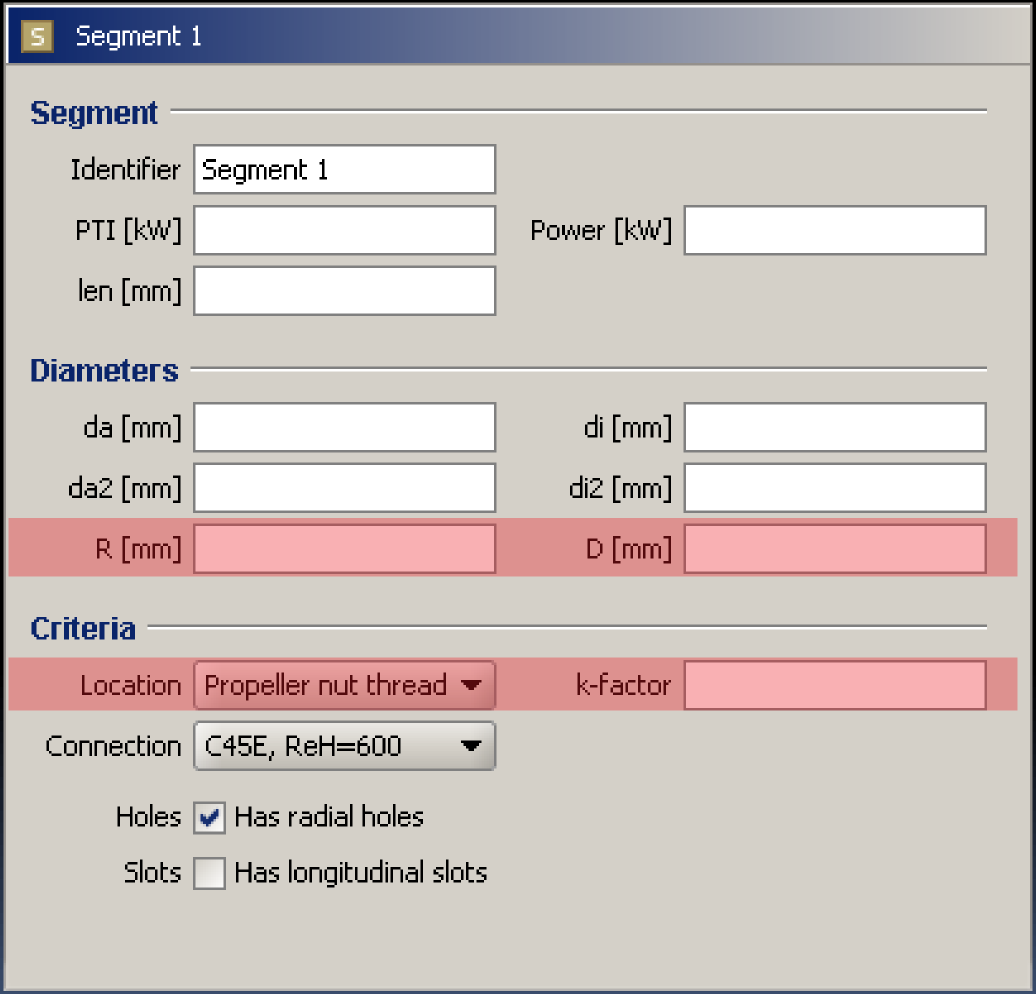
Diameters

da [mm]		di [mm]	
da2 [mm]		di2 [mm]	
R [mm]		D [mm]	

Criteria

Location	Propeller nut thread	k-factor	
Connection	C45E, ReH=600		
Holes	✓ Has radial holes		
Slots	☐ Has longitudinal slots		

Essentials: Equal Line Heights



Segment 1

Segment

Identifier	Segment 1		
PTI [kW]		Power [kW]	
len [mm]			

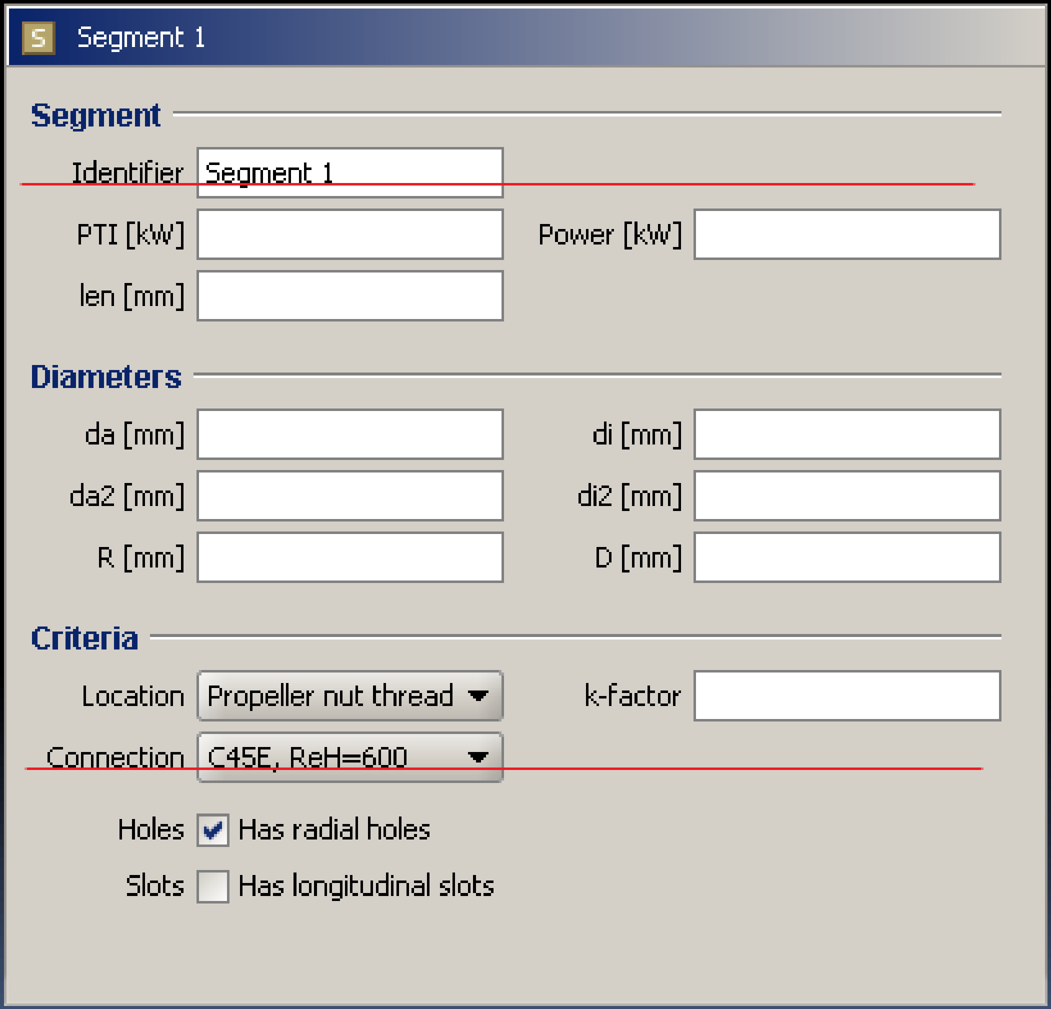
Diameters

da [mm]		di [mm]	
da2 [mm]		di2 [mm]	
R [mm]		D [mm]	

Criteria

Location	Propeller nut thread	k-factor	
Connection	C45E, ReH=600		
Holes	[✓] Has radial holes		
Slots	[ ] Has longitudinal slots		

Essentials: Align Baselines



Segment 1

Segment

Identifier: Segment 1

PTI [kW]:

Power [kW]:

len [mm]:

Diameters

da [mm]:

di [mm]:

da2 [mm]:

di2 [mm]:

R [mm]:

D [mm]:

Criteria

Location: Propeller nut thread

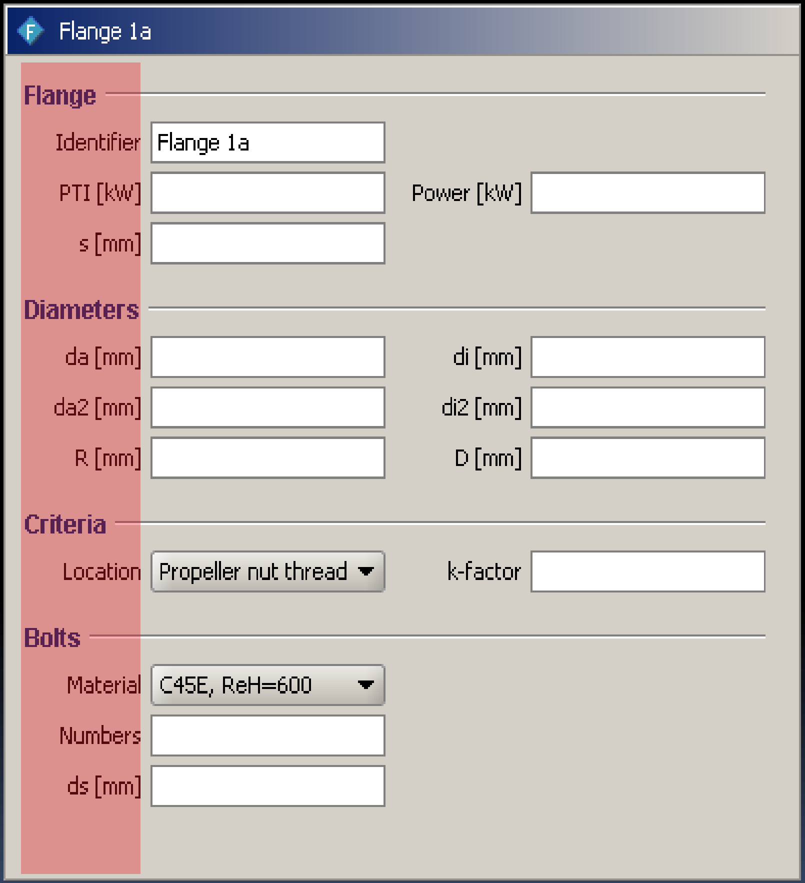
k-factor:

Connection: C45E, ReH=600

Holes: [✓] Has radial holes

Slots: [ ] Has longitudinal slots

Essentials: Stable Layout



Flange 1a

Flange

Identifier	Flange 1a		
PTI [kW]		Power [kW]	
s [mm]			

Diameters

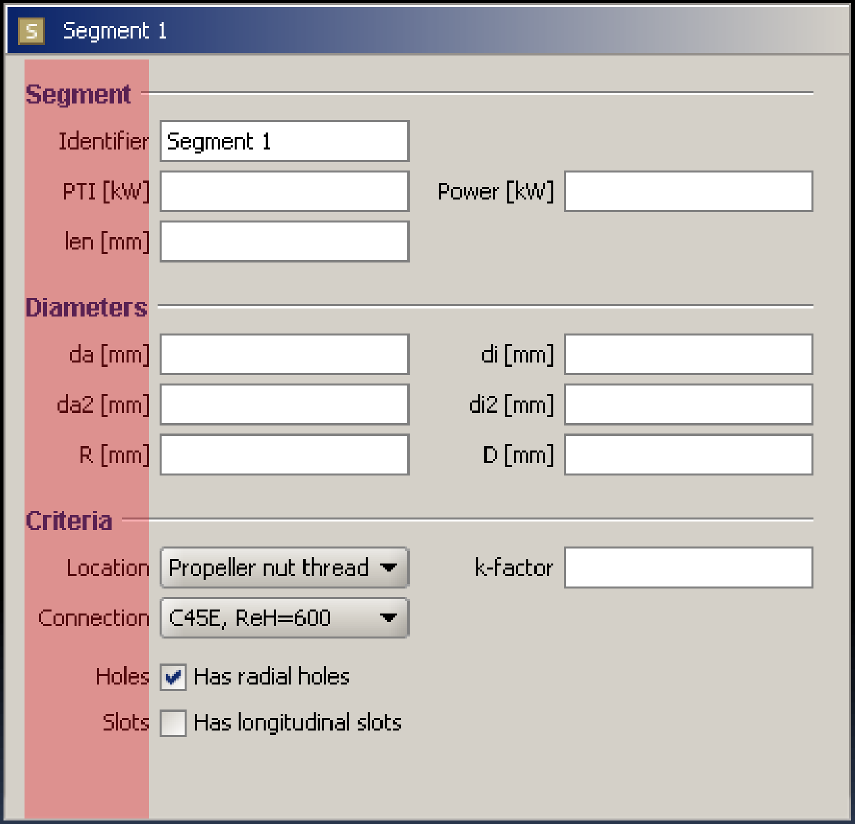
da [mm]		di [mm]	
da2 [mm]		di2 [mm]	
R [mm]		D [mm]	

Criteria

Location	Propeller nut thread	k-factor	

Bolts

Material	C45E, ReH=600
Numbers	
ds [mm]	



Segment 1

Segment

Identifier	Segment 1		
PTI [kW]		Power [kW]	
len [mm]			

Diameters

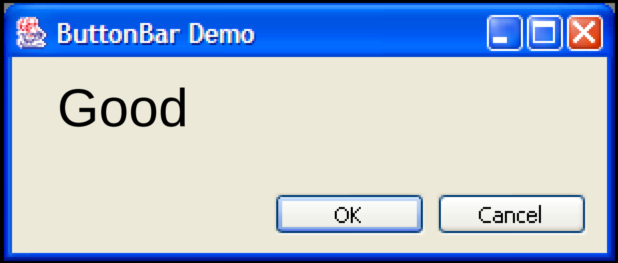
da [mm]		di [mm]	
da2 [mm]		di2 [mm]	
R [mm]		D [mm]	

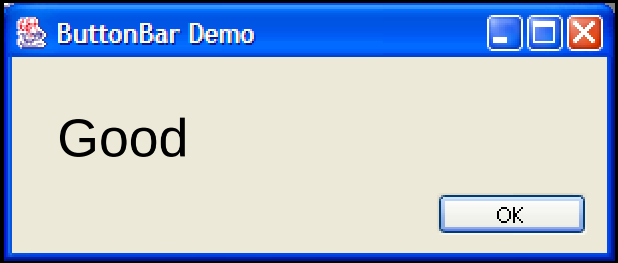
Criteria

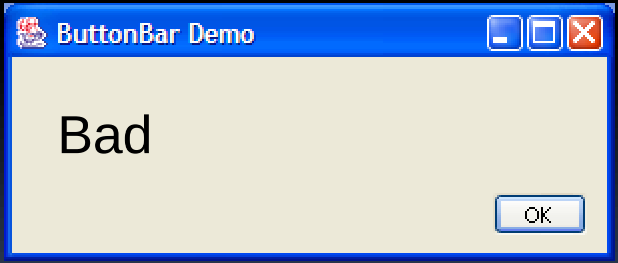
Location	Propeller nut thread	k-factor	
Connection	C45E, ReH=600		
Holes	✓ Has radial holes		
Slots	☐ Has longitudinal slots		

Essentials: Minimal Widths

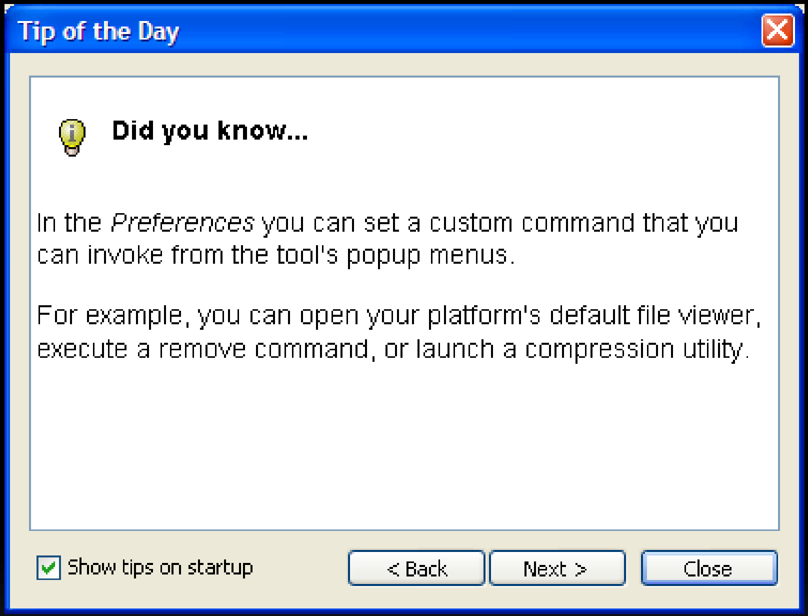
- For example buttons need a minimal width
- The button cannot provide the minimum width; buttons are narrow/wide in different contexts







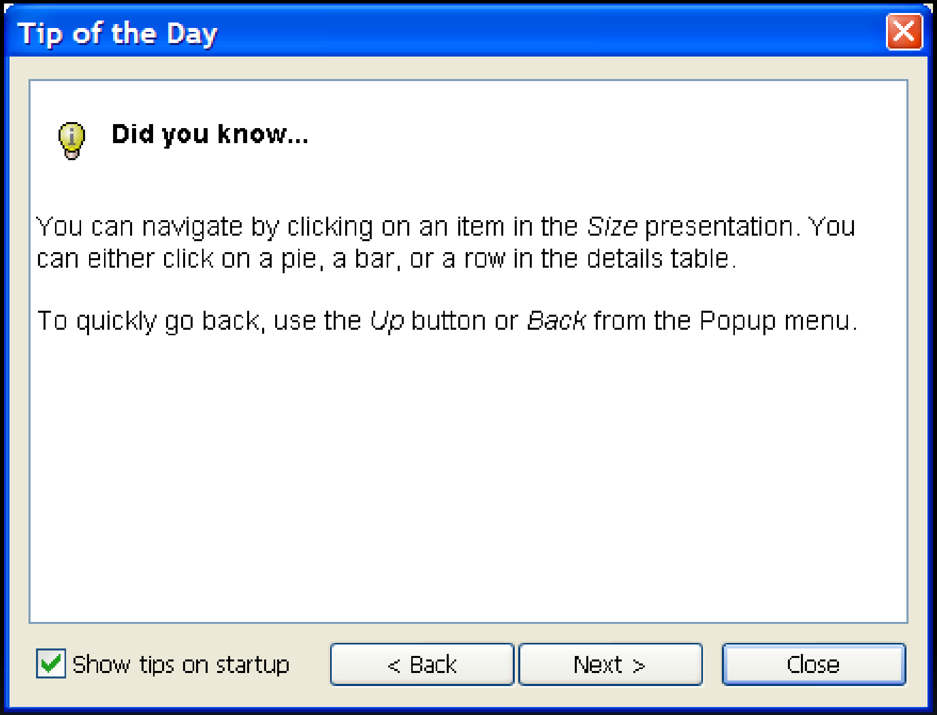
Scale with Font and Resolution



In the *Preferences* you can set a custom command that you can invoke from the tool's popup menus.

For example, you can open your platform's default file viewer, execute a remove command, or launch a compression utility.

96dpi



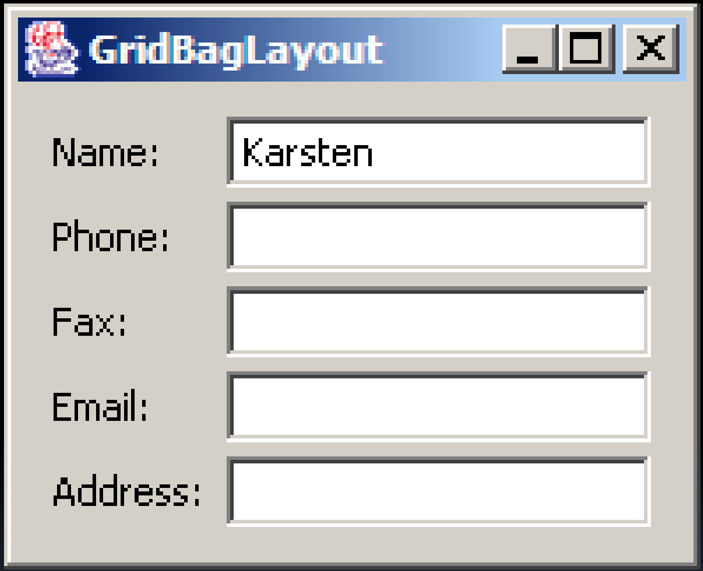
You can navigate by clicking on an item in the *Size* presentation. You can either click on a pie, a bar, or a row in the details table.

To quickly go back, use the *Up* button or *Back* from the Popup menu.

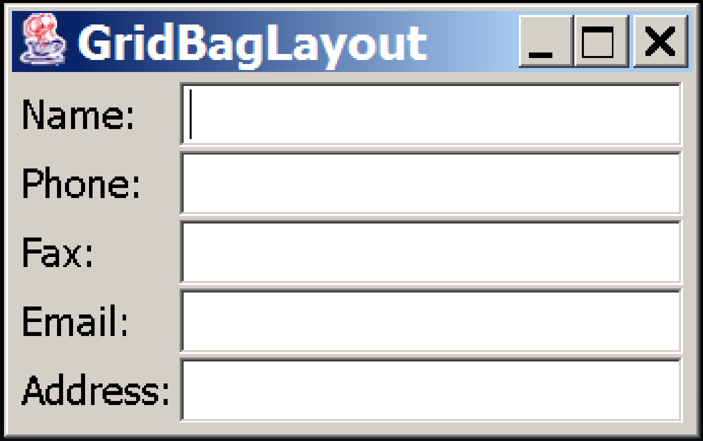
120dpi

No Pixel Sizes in Screen Design!

Otherwise layout does not retain proportions



96dpi



180dpi

Other Weaknesses

- Simple things are difficult to do
- It's difficult to reuse design or layout
- No support for logical layout (Mac vs. PC)
- Out-of-the-box the layout lacks function
- LM implementation is hard to understand
- Layout Manager API is cluttered
- Layout Manager is slow

Layout Summary

The hard stuff is impossible…

… and simple things are difficult to do.

III. Goals

How do we want to layout?

Overall Goal

Make good design easy …

… and the bad difficult.

Goals I

- Build form-oriented panels quickly
- Solution covers 90% of all panels
- Novice users achieve good results
- Expert users save time
- Code is easy to read and to understand
- Design is consistent over panels, applications, team members, and teams

Goals II

- Solution works well with visual editors that
  - increase the productivity
  - Improve the design quality
- The UI construction process is easy to learn
- Solution ships with well designed examples
- Solution ships with all parts out-of-the-box

IV. Concepts

How to achieve the goals?

How to achieve the goals?

- Use a grid for a single layout
- Use a grid system for many layouts
- Use a powerful layout specification language
- Allow string representations to shorten code
- Separate concerns
- Provide layers on top of the layout manager

Grids

- Grids are powerful, flexible and simple
- Grids are easy to understand
- Visual designers use grids
  - to find a layout
  - to align components
- Many people use grids implictly
  when working with paper and pencil

Grid Systems

- Grids scale well
- Grid systems solve many of our problems
- Grid systems assist in finding good design
- They guide us, so we can focus on creativity

Layout Spec: Order and Languae

- Specify the layout first – then build the panel
- Use a powerful specification language
- Apply column and row alignments to cells

- So we can:
  - determine the layout from the spec
  - describe frequently used layout shortly
  - describe complex design with a few lines

Layout Spec with Strings

- Specify the layout with object or
  with a human-readable String representation

- As a result:
  - Simple design requires two lines of code
  - Complex design can be defined in a few lines

Separation of Concerns

- The layout manager shall compute and set component bounds – nothing else
- Other classes jump in
  - to traverse the grid
  - to create frequently used components
  - to extend the grid dynamically
  - to ensure style guide compliance
  - to build a panel from XML

Separation of Concerns: Benefits

- The layout manager API is small
- We can combine helper parts freely
- Changes in a part don't affect the whole
- The layout system is powerful
  – each part is simple

V. Solution

An implementation of our Concepts

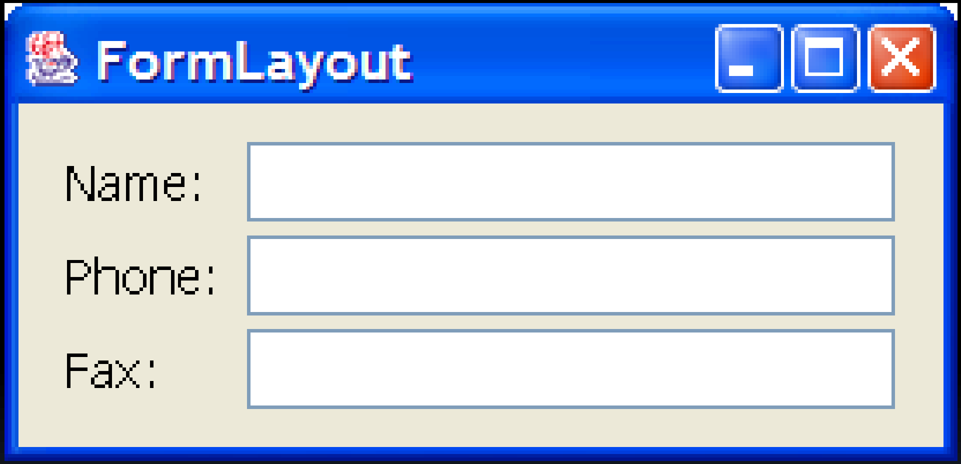
Solution: JGoodies Forms

- Forms is a solution that implements our concepts and meets our goals

- Learn how to work with the Forms
- Learn how Forms makes design easier

Example: A simple Form

How to build this simple form with Forms?



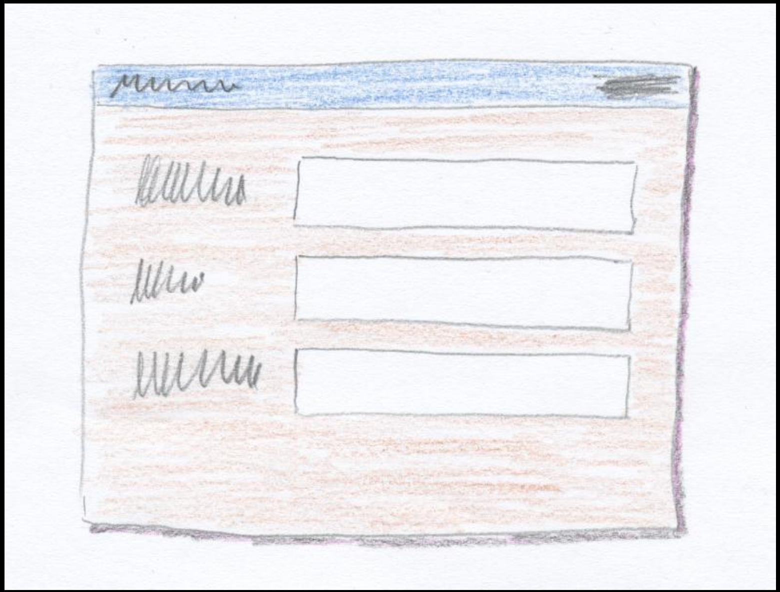
1: Requirements

Boss says:

”We need a panel to edit an address
with fields for: name, phone and email.“

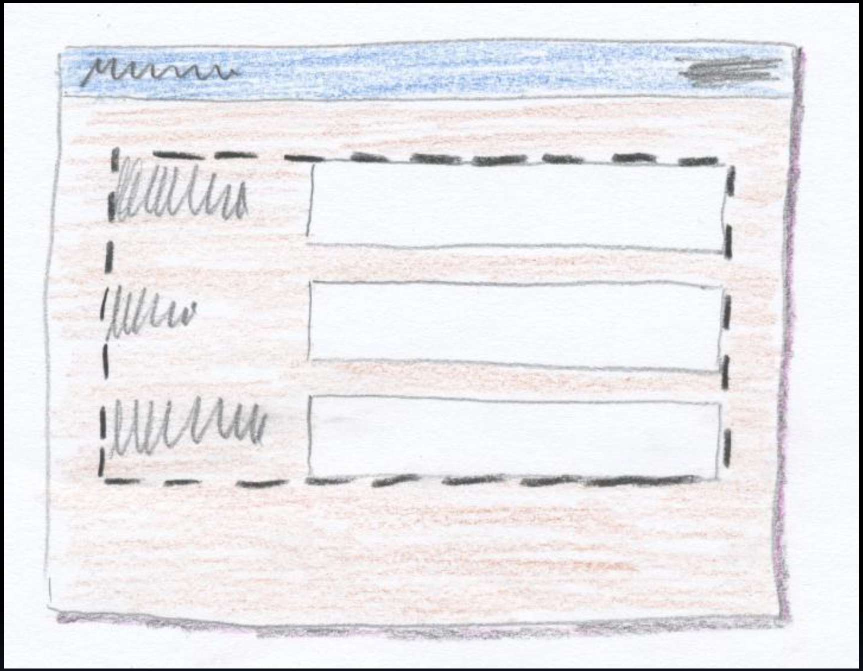
2: Finding the Layout

- A visual designer produces a design draft – with paper & pencil or a visual design tool
- She hands it over to a developer and says: "Follow the Microsoft Layout Style Guide!"



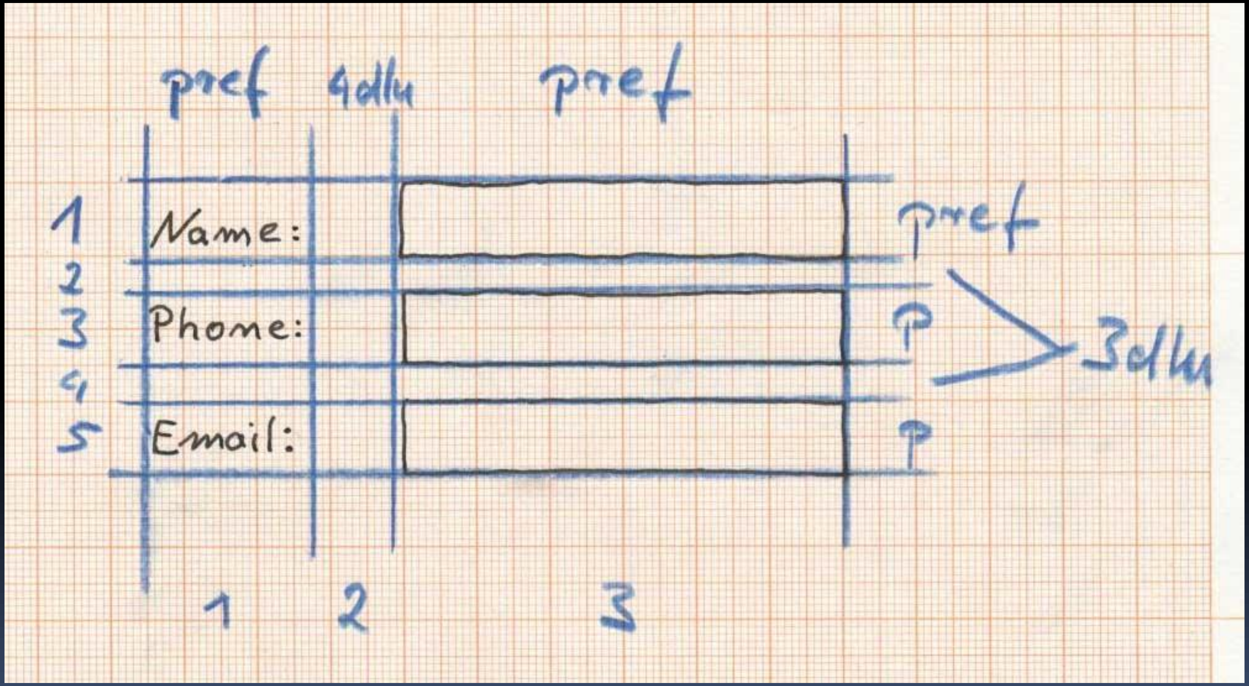
3a: Focus on Content

- Developer identifies a default border



3b: Find the Grid

- Developer finds the grid
- Developer identifies column and row sizes



4: Specify the Layout

The developer specifies the layout:

```
FormLayout layout = new FormLayout(
    “pref, 4dlu, pref”,
    “pref, 3dlu, pref, 3dlu, pref”);
```

4b: Refine the Layout

Left-aligned labels, fields grow

```
FormLayout layout = new FormLayout(
   "left:pref, 4dlu, pref:grow",
   "pref, 3dlu, pref, 3dlu, pref");
```

4c: Refine Layout

Minimum widths; Abbreviations

```
FormLayout layout = new FormLayout(
   “left:[75dlu,pref], 4dlu, pref:grow”,
   “p, 3dlu, p, 3dlu, p”);
```

4d: Refine Layout

Default layout variables

```
FormLayout layout = new FormLayout(
   “left:[75dlu,pref], $lcgap, pref:grow”,
   “p, $lg, p, $lg, p”);
```

4e: Refine Layout

Custom layout variables

```
FormLayout layout = new FormLayout(
    “$label, $lcgap, pref:grow”,
    “p, $lg, p, $lg, p”);
```

5: Add Components

```
JPanel panel = new JPanel(layout);
CellConstraints cc = new CellConstraints();

panel.add(new JLabel(“Name:”),  cc.xy(1, 1));
panel.add(nameField,            cc.xy(3, 1));

panel.add(new JLabel(“Phone:”), cc.xy(1, 3));
panel.add(phoneField,           cc.xy(3, 3));

...
```

5b: Use a Builder

Uses PanelBuilder (recommended)

```
PanelBuilder builder = new PanelBuilder(layout);
CellConstraints cc = new CellConstraints();

builder.addLabel("Name:",     cc.xy(1, 1));
builder.add(nameField,        cc.xy(3, 1));

builder.addLabel("Phone:",    cc.xy(1, 3));
builder.add(phoneField,       cc.xy(3, 3));

...
```

5c: Row Variable

Uses a row variable (not recommended)

```
PanelBuilder builder = new PanelBuilder(layout);
CellConstraints cc = new CellConstraints();
int row = 1;

builder.addLabel("Name:",     cc.xy(1, row));
builder.add(nameField,        cc.xy(3, row));

row += 2;
builder.addLabel("Phone:",    cc.xy(1, row));
builder.add(phoneField,       cc.xy(3, row));

...
```

5d: Use a high-level Builder

Uses DefaultFormBuilder

```
FormLayout layout = new FormLayout(
    “l:p, $lcgap, p:g”);  // Columns
                          // Add rows dynamically

DefaultFormBuilder builder =
    new DefaultFormBuilder(layout);

builder.append(“Name:”,  nameField);
builder.append(“Phone:”, phoneField);
builder.append(“Email:”, emailField);

return builder.getPanel();
```

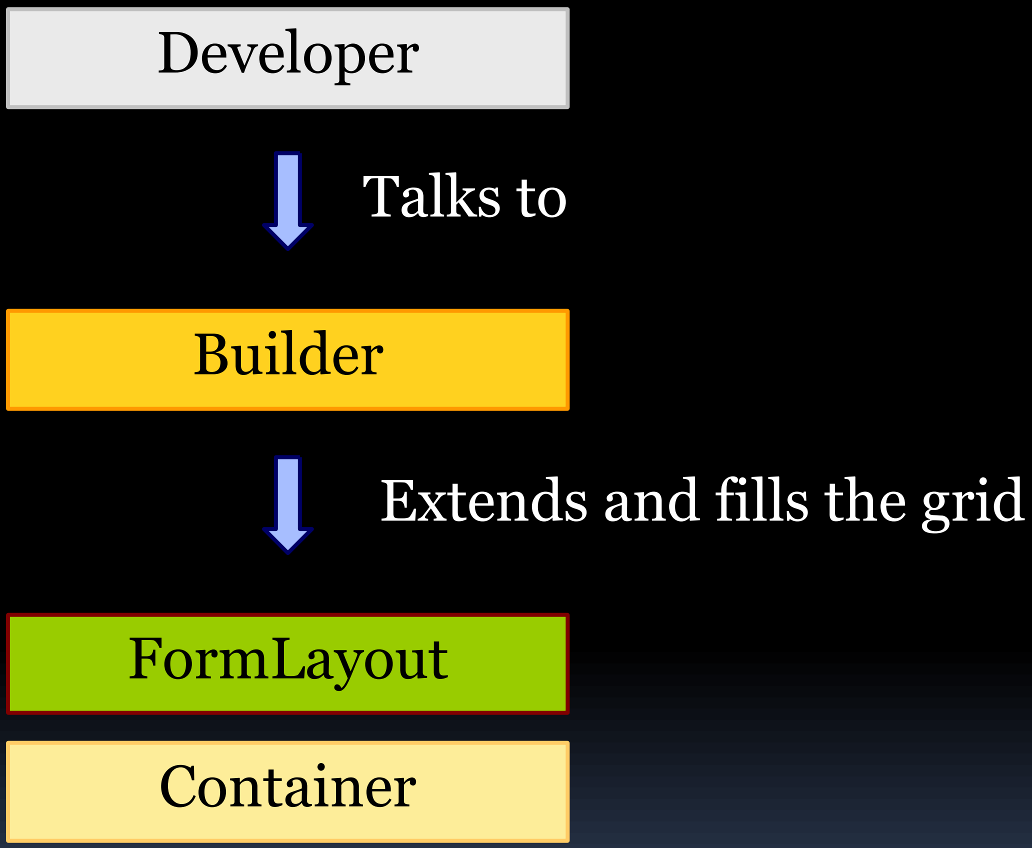
6: Add a Default Border

```
DefaultFormBuilder builder =
    new DefaultFormBuilder(layout);

builder.setDefaultDialogBorder();

...
```

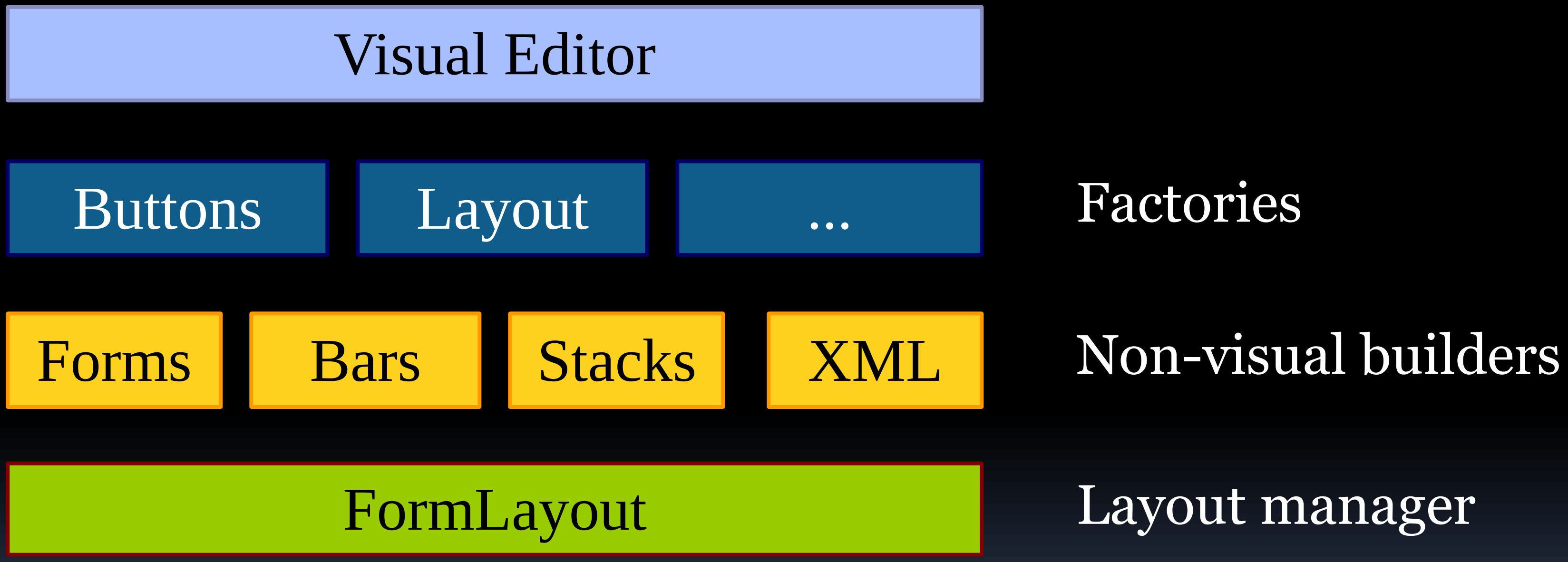
Factories and Logical Sizes

- The **`ButtonBarBuilder2`**:
  - builds consistent button bars
  - honors the platform's style
  - uses logical sizes
    e.g.: gap between 2 related buttons
- The **`ButtonBarFactory`**:
  - vends prepared button bars
  - uses logical layout (Mac vs. PC)

Non-visual Builders



Layers in Forms



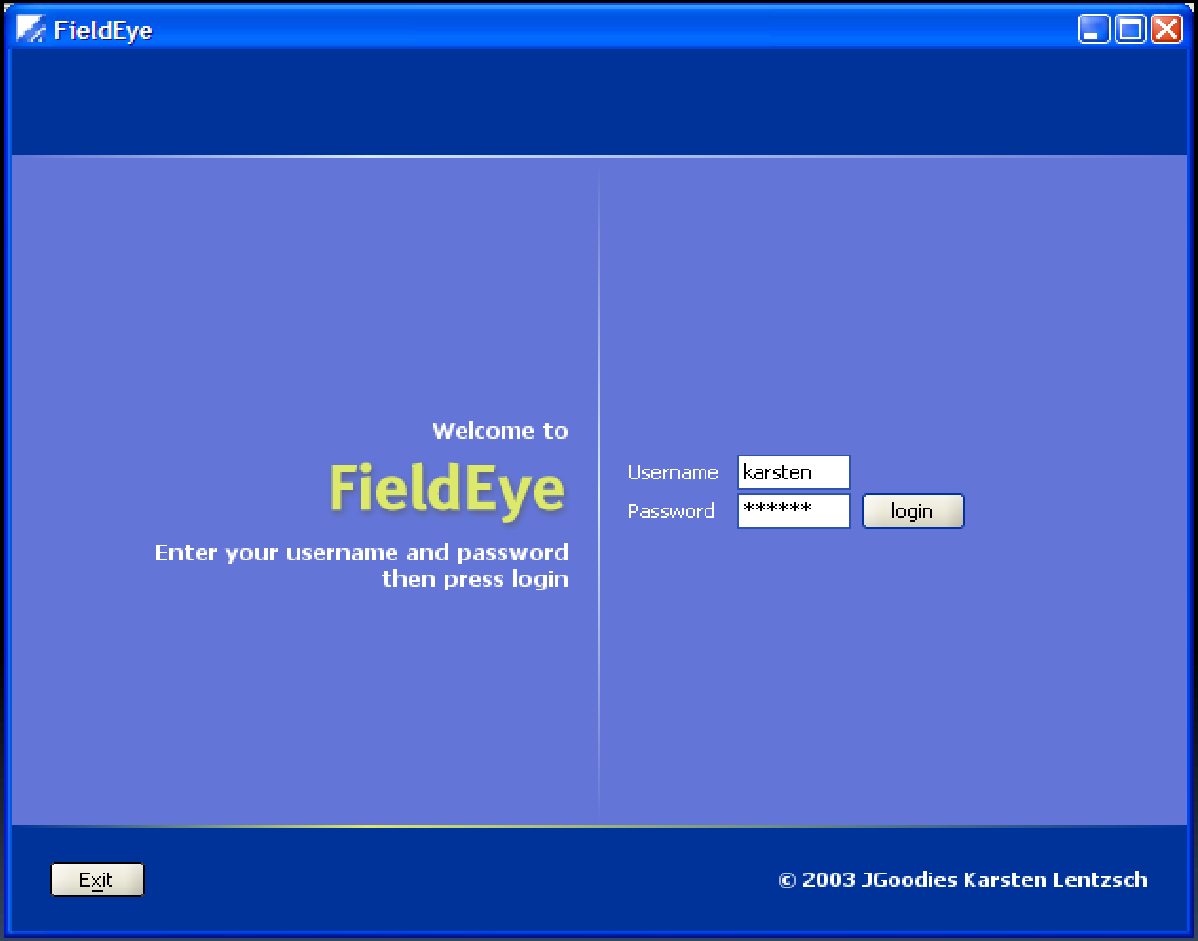
VI. Future Directions

What comes next?

How to achieve good design?

The 'average' developer won‘t design this dialog.

How can we assist him in getting such a design?



Welcome to

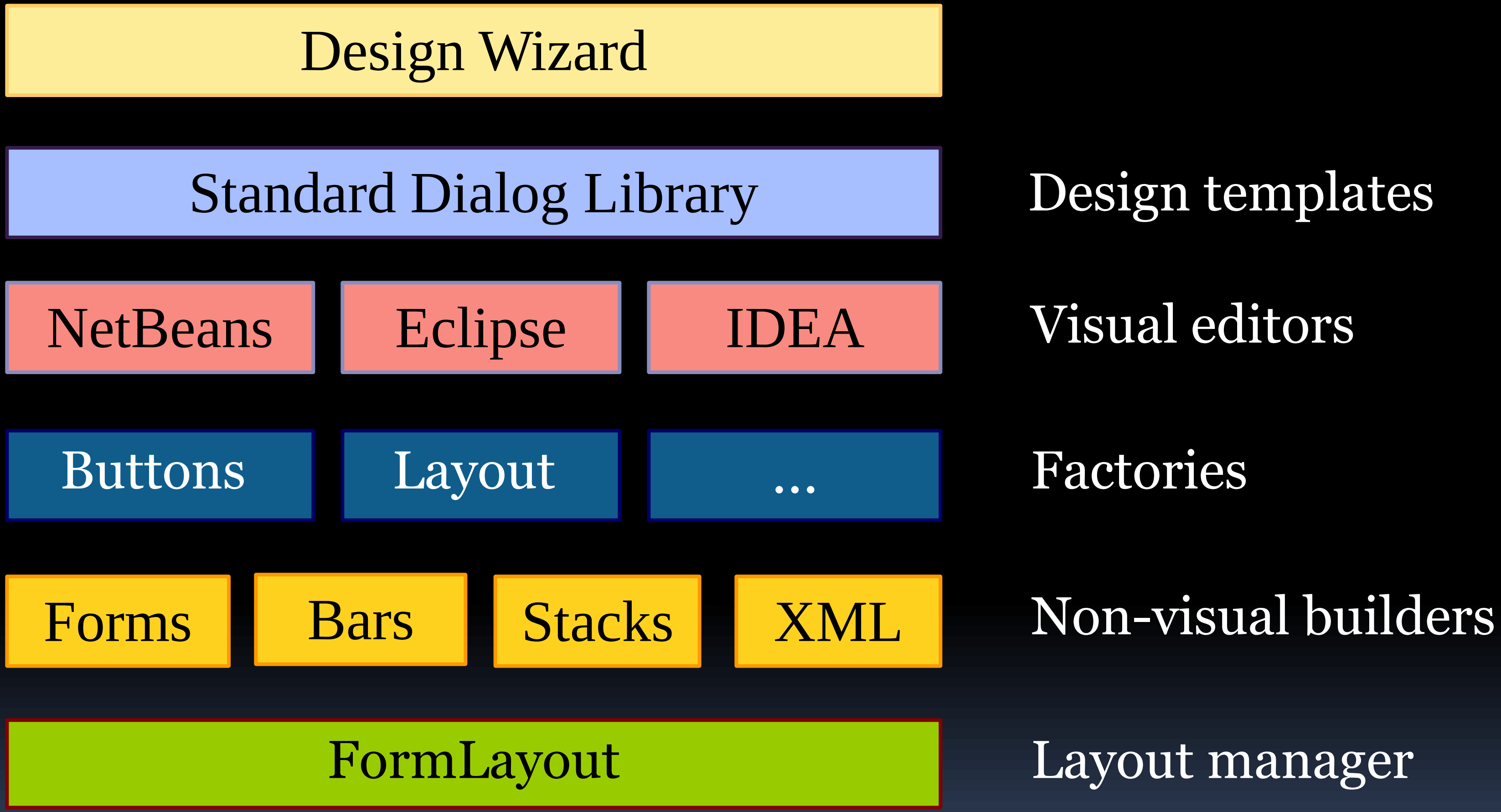
FieldEye

Enter your username and password then press login

Username	karsten
Password	******

© 2003 JGoodies Karsten Lentzsch

Templates and Wizards



What comes next?

- More layout templates
- Visual form editors
- More non-visual builders
- Improved support for logical sizes
- Inter-panel constraints
- Support for perceived bounds

Summary

- We have analyzed layout problems
- We have learned how to address them
- We have seen a layout solution
- We have outlined further improvements

References

- Design Specifications and Guidelines
  msdn.microsoft.com
- Aqua Human Interface Guidelines
  www.apple.com/developer
- JGoodies Forms Framework
  forms.dev.java.net

A Powerful Layout Manager

- ExplicitLayout, see www.zookitec.com
  - is powerful
  - provides *styles* – much like our builders
  - supports non-rectangular layouts
  - pixel sizes only
  - no logical sizes
  - available under the LGPL

Recommended Reading

Kevin Mullet & Darrel Sano

Designing Visual Interfaces

- All about: visual variables, scale, contrast, proportion, organization and visual structure
- Many useful examples
- Interesting and easy to read
- 250 pages with many screenshots

If you only remember one thing:

Use professional L&Fs and use the Forms!

Questions and Answers

Question

Does Forms work with AWT and SWT?

Answer

FormLayout works with AWT not SWT

Additional layers work with Swing only

Question

How mature is the Forms release?

Does it meet production quality?

Answer

FormLayout is stable since Dec-2002.

The framework is ready for production.

End

Hope that helps!

Good luck!