

Karsten Lentzsch – JGoodies

SWING WITH STYLE

JGoodies: Karsten Lentzsch

- Open source Swing libraries
- Example applications
- Consultant for Java desktop
- Design assistance
- Training for visual design and implementation

- Study GUI production process

Before

Vk0010 Kamp All S: Window 0

Vertrag bearbeiten

Vertrag Nr.	Vertrag Beg.	Vertrag End.	min. Dauer	Künd.Frist	Künd. Datum	Abt. Per. Dauer
475450	KOM	KAM Verkauf Zürich	Auftragsabwicklung West	25/07	25/07	in Bearbeitung

Kampagne

Kampagne Nr.	VB	Verkaufs-OE	Abwicklung	von KW	bis KW	Status	Bruttobetrag
475450	KOM	KAM Verkauf Zürich	Auftragsabwicklung West	25/07	25/07	in Bearbeitung	89,491

Auftraggeber: Omega SA
 Mediaagentur:
 Werbesagentur: Provisorisches Sujet
 Suche von W: fest dgu

Kundenbeziehung

Aufträge	Referenz/Notiz	Kampagnen-Dokumente
Auftraggeber: 109013 Werbesagentur: Mediaagentur: VB H.Betr: KOM	Omega SA, 2504 Biel/Bienne Verkaufs-OE H.Betr: KAM Verkauf Zürich	Institution Institution Institution

Zahlungsart Beraterkommission: keine Beraterkommission
 Anzahl RO-Kopien: 1
 Rechnungsempfänger:
 Institution:
 Person:
 Kontakt:
 Kontakt:
 Kontakt:
 Endkunde: 109013 Omega SA, 2504 Biel/Bienne
 Institution:
 Person:
 Kontakt:
 GU-Fall

Flächen an APOVis | Qualität der Kamp. | Status-History | Dokument erstellen | Kamp. bestätigen | Kamp. annullieren

Flächen an APOVis | Qualität der Kamp. | Status-History | Dokument erstellen | Kamp. bestätigen | Kamp. annullieren

Endkunde: 109013 Omega SA, 2504 Biel/Bienne
 Institution:
 Person:
 Kontakt:
 Kontakt:
 Kontakt:
 Endkunde: 109013 Omega SA, 2504 Biel/Bienne
 Institution:
 Person:
 Kontakt:
 GU-Fall

After

IT21: Auftrag 589434

IT21 | Gepard | Flächen | Produkte | Verkauf | Hilfe

Auftrag 589434, KW 20 – 25/07, [Vertrag 32168](#), [Kampagne 475450](#), [Omega SA](#), Cindy Crawford

Anwendungen:
 Verfügbare Flächen
 Verfügbare Netze
 Flächenreservierungen
 Netzreservierungen
 Verträge
 Kampagnen
 Aufträge
 Auftragspositionen
 Lokaldispo
 Statistiken
 Verwaltung

Aufgaben:
 Auftrag bestätigen
 Auftrag annullieren
 Statushistorie zeigen
 Flächen in APGVis öffnen

Auftrag | Rabatte | Zusatzkosten | Visierung

Auftrag: ausführbar, visumpflichtig Bruttobetrag: 89.491 CHF

Auftragsart: Zahlungsmodus:

Verrechnungsebene: Amtl. Abgaben

Hauptbranche: Nebenbranche2:

Nebenbranche1: Nebenbranche3:

Positionen

Nr.	Produktname	Beginn	Datum	Dauer	Ende	Prov. Subj.	Status
827275	Zürich Agglo	25/07	18.06.2007	7	25/07	Cindy Crawford	bestätigt
827218	Zug	25/07	18.06.2007	7	25/07	Cindy Crawford	annuliert
827226	Zug Innenstadt	25/07	18.06.2007	7	25/07	Cindy Crawford	annuliert
827229	Winterthur	20/07	14.05.2007	7	20/07	Cindy Crawford	annuliert
827269	Bern	25/07	18.06.2007	7	25/07	Cindy Crawford	annuliert

Neu... Bearbeiten... Bestätigen Annullieren

K 475434, 40 – 42/07 K 475199, 20 – 30/07 A 589434, 20 – 25/07 A 589478, 25 – 30/07 P 586234, 20 – 25/07
 Omega SA Omega SA Cindy Crawford Omega SA Omega SA Omega SA
 Cindy Crawford Cindy Crawford Cindy Crawford Boris Becker Zürich

IT21: Auftrag 589434

IT21 | Gepard | Flächen | Produkte | Verkauf | Hilfe

Auftrag 589434, KW 20 – 25/07, [Vertrag 32168](#), [Kampagne 475450](#), [Omega SA](#), Cindy Crawford

Anwendungen:
 Verfügbare Flächen
 Verfügbare Netze
 Flächenreservierungen
 Netzreservierungen
 Verträge
 Kampagnen
 Aufträge
 Auftragspositionen
 Lokaldispo
 Statistiken
 Verwaltung

Aufgaben:
 Auftrag bestätigen
 Auftrag annullieren
 Statushistorie zeigen
 Flächen in APGVis öffnen

Auftrag | Rabatte | Zusatzkosten | Visierung

Auftrag: ausführbar, visumpflichtig Bruttobetrag: 89.491 CHF

Auftragsart: Zahlungsmodus:

Verrechnungsebene: Amtl. Abgaben

Hauptbranche: Nebenbranche2:

Nebenbranche1: Nebenbranche3:

Positionen

Nr.	Produktname	Beginn	Datum	Dauer	Ende	Prov. Subj.	Status
827275	Zürich Agglo	25/07	18.06.2007	7	25/07	Cindy Crawford	bestätigt
827218	Zug	25/07	18.06.2007	7	25/07	Cindy Crawford	annuliert
827226	Zug Innenstadt	25/07	18.06.2007	7	25/07	Cindy Crawford	annuliert
827229	Winterthur	20/07	14.05.2007	7	20/07	Cindy Crawford	annuliert
827269	Bern	25/07	18.06.2007	7	25/07	Cindy Crawford	annuliert

Neu... Bearbeiten... Bestätigen Annullieren

K 475434, 40 – 42/07 K 475199, 20 – 30/07 A 589434, 20 – 25/07 A 589478, 25 – 30/07 P 586234, 20 – 25/07
 Omega SA Omega SA Cindy Crawford Omega SA Omega SA Omega SA
 Cindy Crawford Cindy Crawford Cindy Crawford Boris Becker Zürich

Before

Erkunde von uns, die hier wohnen.
in zinger waren zwar von Unbrauchbarkeit,
bestenfalls planken wir die Kijordbrände
planken, geben uns ein typisches
gke es gäbe, dort schwebte Kralle, was
Mutter eine Seelkarte. So malen wir
Toben, Kijordbrände, was an hellen
und eben im Bunde von uns, was
war fast mit der Mutter die Karte
d. Was fanden wir noch nicht
w. So langsam wurden es dunkel
d. hier jeder Zeite, und wir die wir
es keinen Platz mehr. Die
im blind unglücklichen Toren
allein. Mein Vater suchte das
Leitung war tot. Todesurteil, die
eine schuld. Das ist einfach unbeschreiblich.
Verlieren! Mutter kann gar nicht
rüd. was ein stürzen für mit
in Toren und

After

MADAM, After her six years' residence at the Mall, I have the honour and happiness of presenting Miss Amelia Sedley to her parents, as a young lady not unworthy to occupy a fitting position in their polished and refined circle. Those virtues which characterize the young English gentlewoman, those accomplishments which become her birth and station, will not be found wanting in the amiable Miss Sedley, whose INDUSTRY and OBEDIENCE have endeared her to her instructors, and whose delightful sweetness of temper has charmed her aged and her youthful companions.

Goal

See success factors for Swing projects

How to implement Swing clients?

Learn criteria for good writing style

Agenda

Visual Dos and Don'ts

Implementation Patterns

Writing Style

Visual Patterns

Visual Don'ts

- Color
- Fonts
- Icons



Packages

- Chartster
 - src
 - build
- JDiskReport
- JGoodies Animat...
- JGoodies Charts
 - src
 - build
- JGoodies Looks
 - src
 - com.jgood...
 - com.jgood...
 - com.jgood...
 - com.jgood...
 - build
- JGoodies Looks
- JGoodies Swing

description.txt

THE OVERALL APPEARANCE sucks! It
 common GUI design errors that one ca
 Java clients.

THE ICON SET is a mixture of several i
 different sizes, colors schemes and sy
 icons use oversaturated colors.

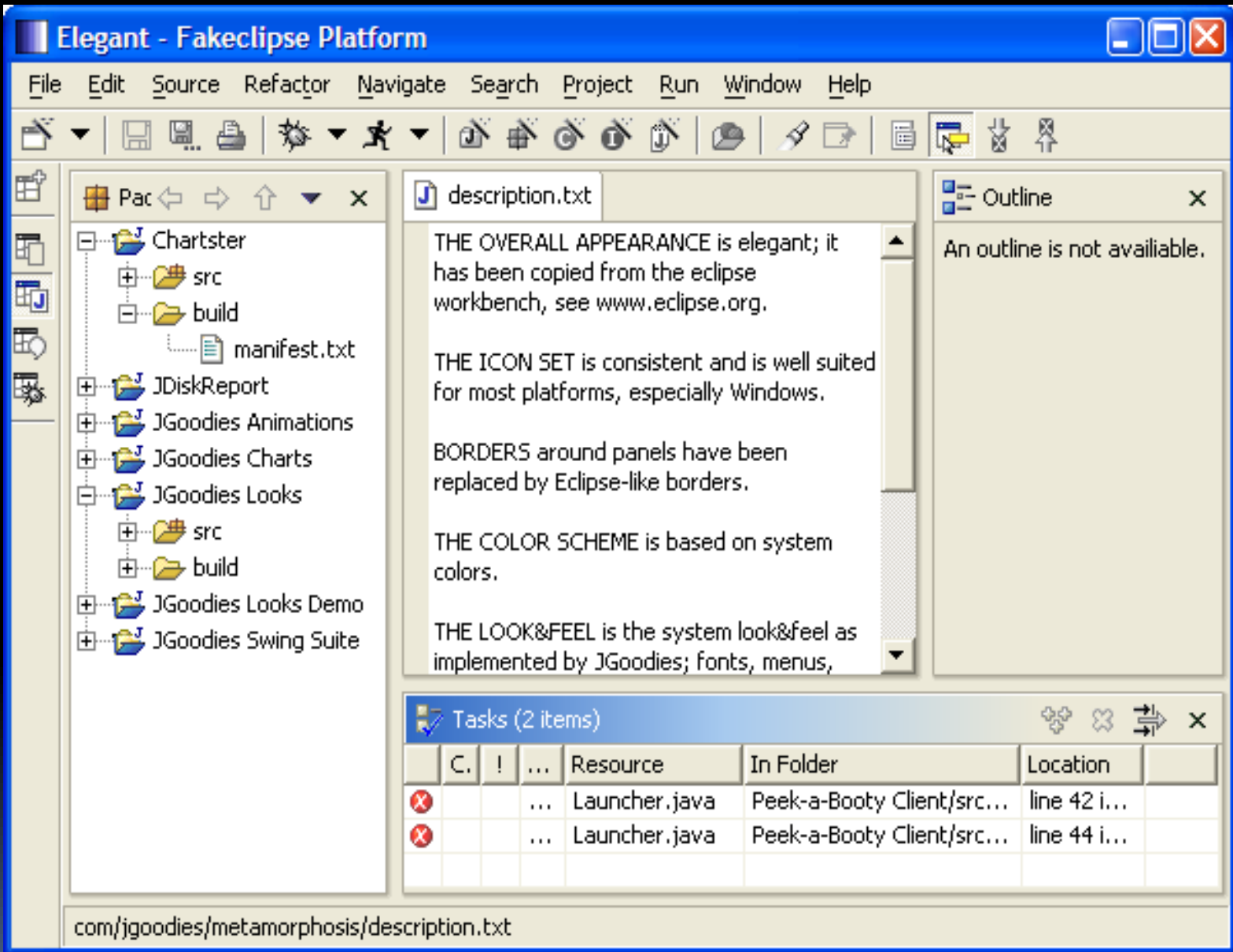
BORDERS are used heavily and clutter
 A source of trouble is the JSplitPane th
 three borders for both components and

Outline

An outline is not avail

Tasks (2 items)

	C.	!	Descri...	Resou...	In Fold...	Locati...
			org.c...	Laun...	Peek...	line ...
			org.c...	Laun...	Peek...	line ...



Elegant - Fakeclipse Platform

File Edit Source Refactor Navigate Search Project Run Window Help



Pac

- Chartster
 - src
 - build
 - manifest.txt
- JDiskReport
- JGoodies Animations
- JGoodies Charts
- JGoodies Looks
 - src
 - build
- JGoodies Looks Demo
- JGoodies Swing Suite

description.txt

THE OVERALL APPEARANCE is elegant; it has been copied from the eclipse workbench, see www.eclipse.org.

THE ICON SET is consistent and is well suited for most platforms, especially Windows.

BORDERS around panels have been replaced by Eclipse-like borders.

THE COLOR SCHEME is based on system colors.

THE LOOK&FEEL is the system look&feel as implemented by JGoodies; fonts, menus,

Outline

An outline is not available.

Tasks (2 items)

	C.	!	...	Resource	In Folder	Location	
✖			...	Launcher.java	Peek-a-Buty Client/src...	line 42 i...	
✖			...	Launcher.java	Peek-a-Buty Client/src...	line 44 i...	

com/jgoodies/metamorphosis/description.txt

Visual Dos

- Align
- Remove unnecessary borders
- Remove all unnecessary elements

- Use white space
- Learn about contrast, balance



Institution

Gepard-Nr. **Bezeichnung**

Favorit für Sie **Interne Bezeichnung**

Favorit für andere MA **Bezeichnungen-Zusatz**

Hierarchie **Sprache** **Verkaufspartner**

Inst. beenden **MWST-pflichtig** **Hauptbetreuer APG**

9-stellige ESR-Nr **ESR-Nr.** **Hauptbetreuer Montagne**

Kopien Faktura **Gültig von** **Gültig bis**

Unternehmung/Enterprise

Rechtsform

MWST-Nr. **MWST 90.2.a Internat. Org.**

Auftraggeber/Donneur d'ordre

Bonität

Begründg.

ABC-Kunden

Kontakt/Personne de contact

HauptGepard-Nr.	Nachname	Vorname	Titel	Email	Direktwahl	Funktion	Gültig von	Gültig bis		
111	Mihu	Lucian							Zuordn. erstellen	Zuordn. löschen
									Zuordn. erstellen	Zuordn. löschen

Adresse Fon/Fax Konto Druckerei SAP Ext. Logistik Logging-Einträge Kategorien Ka

Adress Typ

Strasse

Strasse 2

Postfach **Postfach-Nr.**

PLZ **Ort**

Region **Land**

Gültig von **Gültig bis**

- Anwendungen:
- Verfügbare Flächen
- Verfügbare Netze
- Flächenreservierungen
- Netzreservierungen
- Verträge
- Kampagnen
- Aufträge
- Auftragspositionen
- Lokaldispo
- Statistiken
- Verwaltung
- Aufgaben:**
- Favorit für mich
- Favorit für Andere

12399, Karsten Lentzsch, [JGoodies Karsten Lentzsch](#)

Person | Kontakt | Konto/SAP | Kategorien | Kampagnen

Nachname:

Vorname:

Anrede:

Briefanrede:

Titel:

Titel 2:

Funktion:

Gültigkeit:

MWSt.-pflichtig

ESR-Nr.: 9-stellig

Fakturakopien:

Ist Verkaufspartner

Hauptbetr. APG:

Hauptbetr. Montagne:

Institutionen:

Nr.	Name	Kurzname	Zusatz	Funktion	Gültigkeit
58434	JGoodies Karsten Lentzsch	JGoodies		Präsident	04.04.07...

Hinzufügen...
Entfernen

Adresstyp:

Strasse:

Strasse 2:

Postfach: Hat Postfach

PLZ, Ort:

Region, Land:

Gültigkeit:

Visual Dos and Don'ts

For details see:

JGoodies.com -> Downloads -> Presentations

„First Aid for Swing UIs“

Agenda

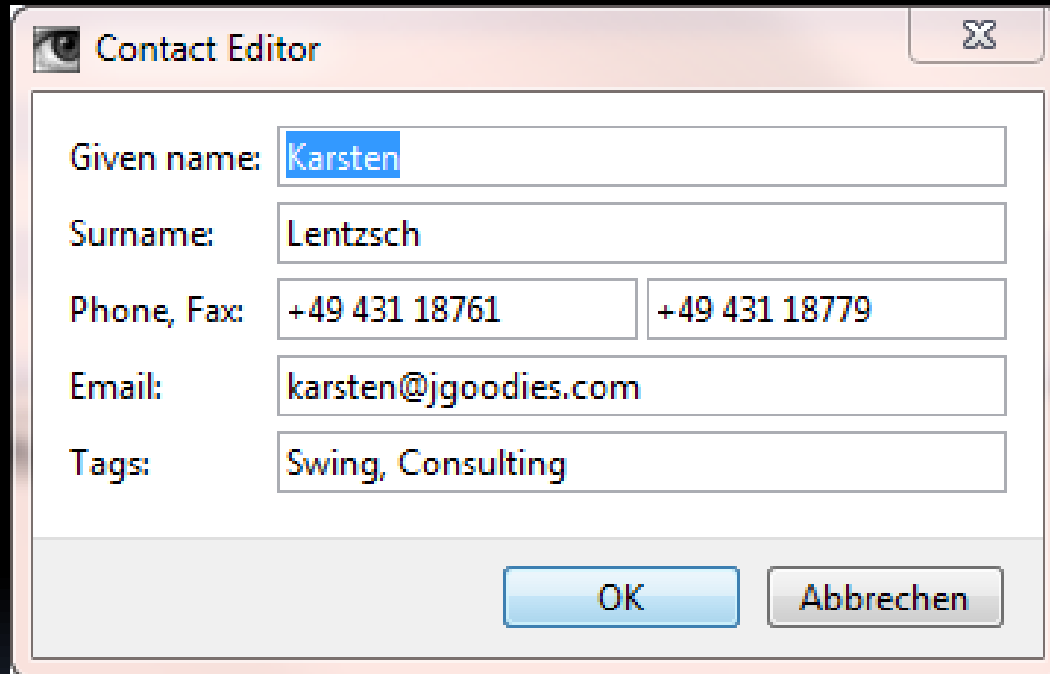
Visual Dos and Don'ts

Implementation Patterns

Writing Style

Visual Patterns

Contact Editor



Contact Editor

Given name:

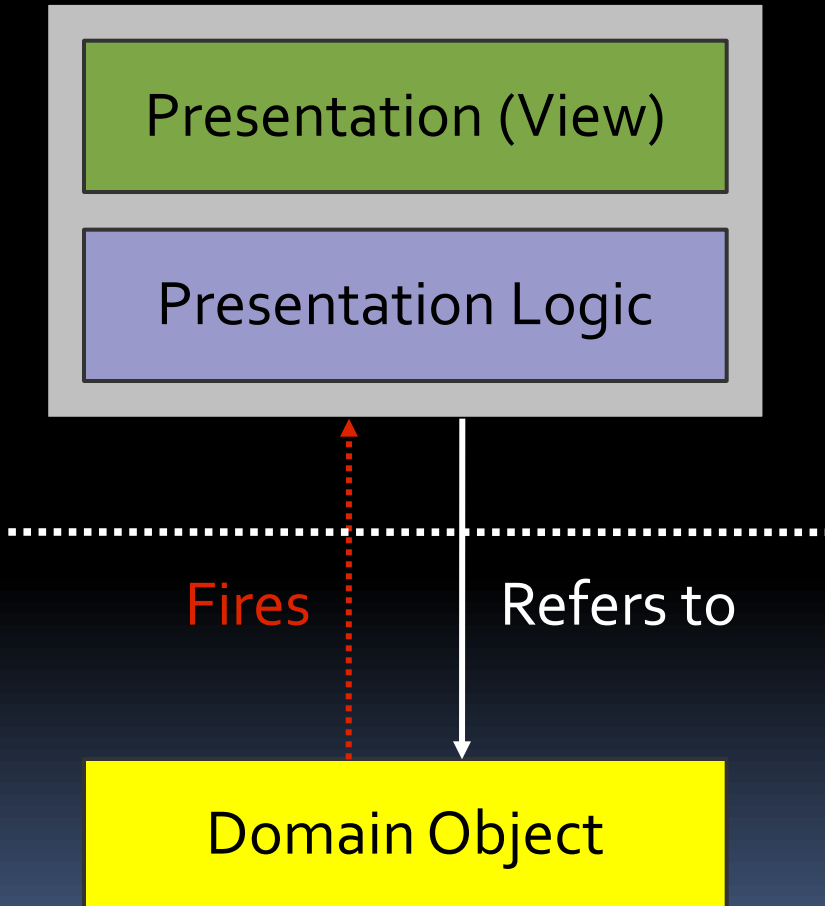
Surname:

Phone, Fax:

Email:

Tags:

Legend



Tip 1

- Separate domain data

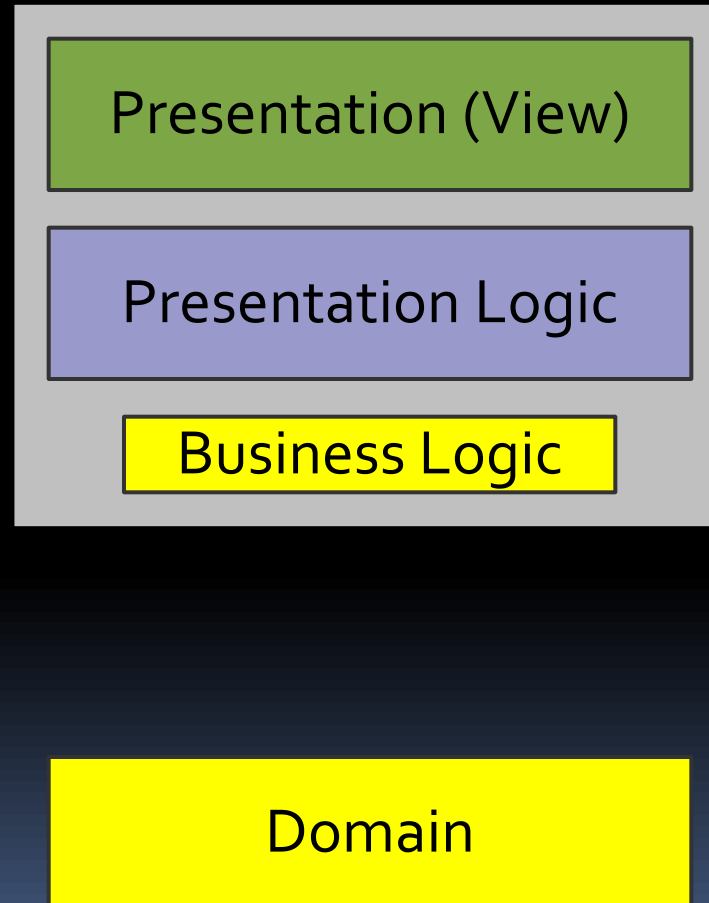
Pattern: Separated Presentation

Presentation (View)

Presentation Logic

Domain

Domain Logic in Presentation



Tip 1

- Separate domain data
- Separate domain logic
- Consider object orientation

Domain Class

```
public class Contact {  
  
    private String givenName;  
    private String surname;  
    private String phone;  
    ...  
  
    public String getPhone() {  
        return phone;  
    }  
  
    public void setPhone(String newPhone) {  
        phone = newPhone;  
    }  
    ...  
}
```


With Bound Properties

```
public class Contact extends Bean {  
  
    public static final String PROPERTY_PHONE  
        = "phone";  
    ...  
  
    public void setPhone(String newPhone) {  
        String oldPhone = getPhone();  
        phone = newPhone;  
        firePropertyChange(  
            PROPERTY_PHONE, oldPhone, newPhone);  
    }  
    ...  
}
```

Pattern: *Autonomous View*

Presentation (View)

Presentation Logic

Autonomous View I

```
public class ContactEditorDialog extends JDialog {  
    private final Contact contact;  
  
    private JTextField givenNameField;  
    ...  
  
    public ContactEditorDialog(Contact contact) { ... }  
  
    private void initComponents() { ... }  
  
    private void initEventHandling() { ... }  
  
    private JComponent buildContent() { ... }
```

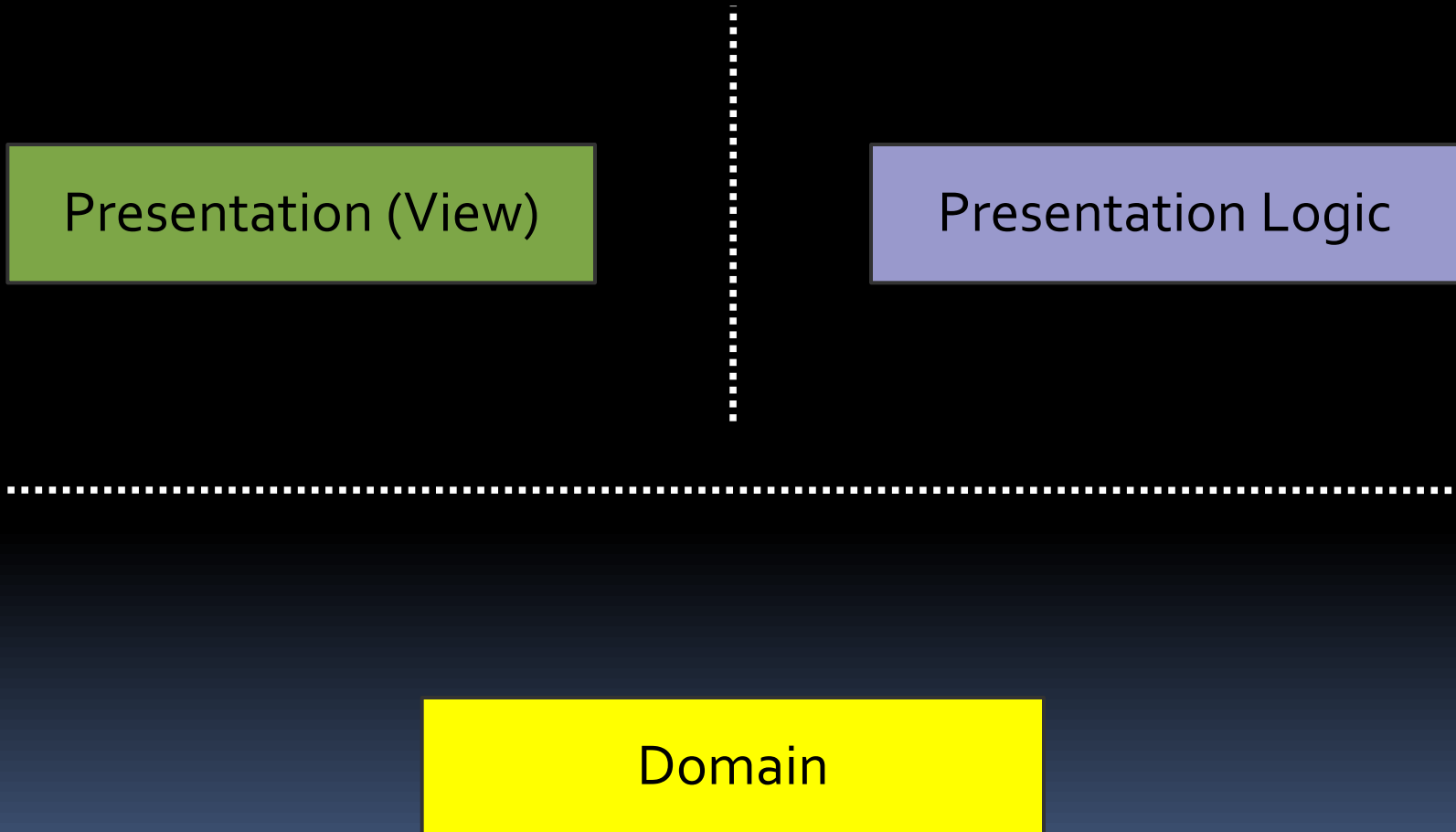
Autonomous View II

```
class OKAction extends AbstractAction {  
    private OKAction() {  
        super("OK");  
    }  
  
    public void actionPerformed(ActionEvent e) {  
        ...  
    }  
}
```

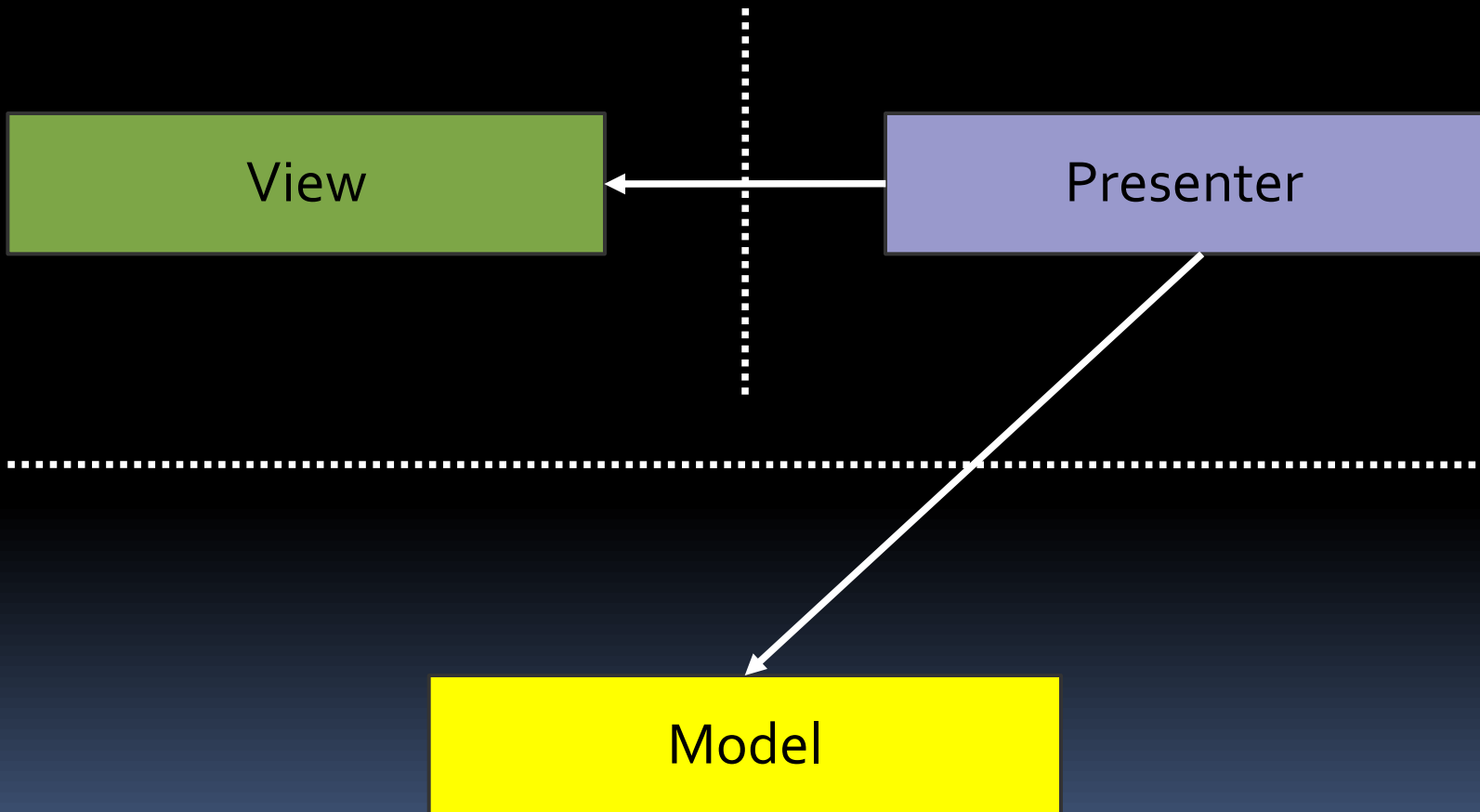
Tip 2

- Separate presentation from presentation logic

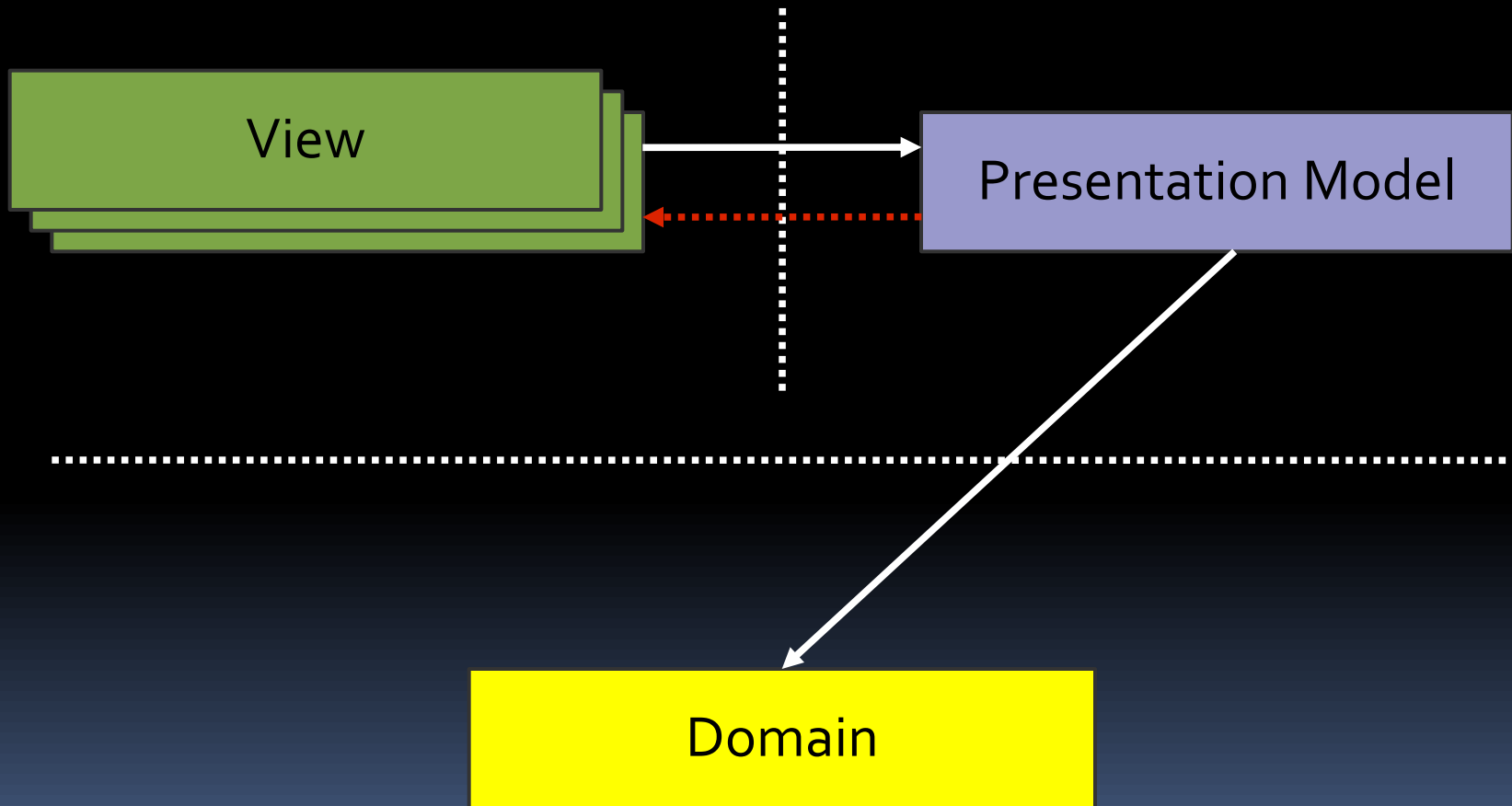
Separated Presentation Logic



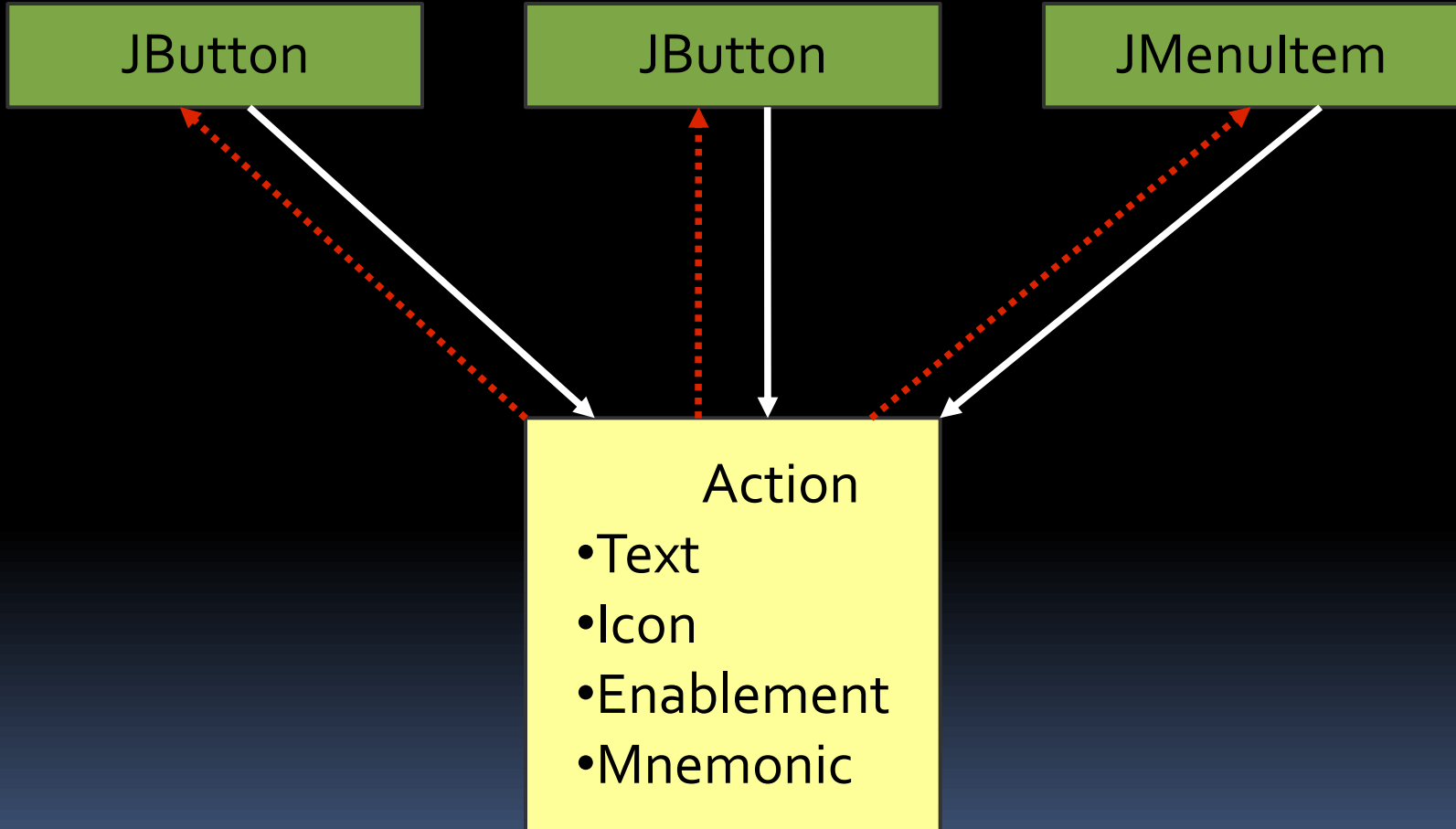
Pattern: Model-View-Presenter



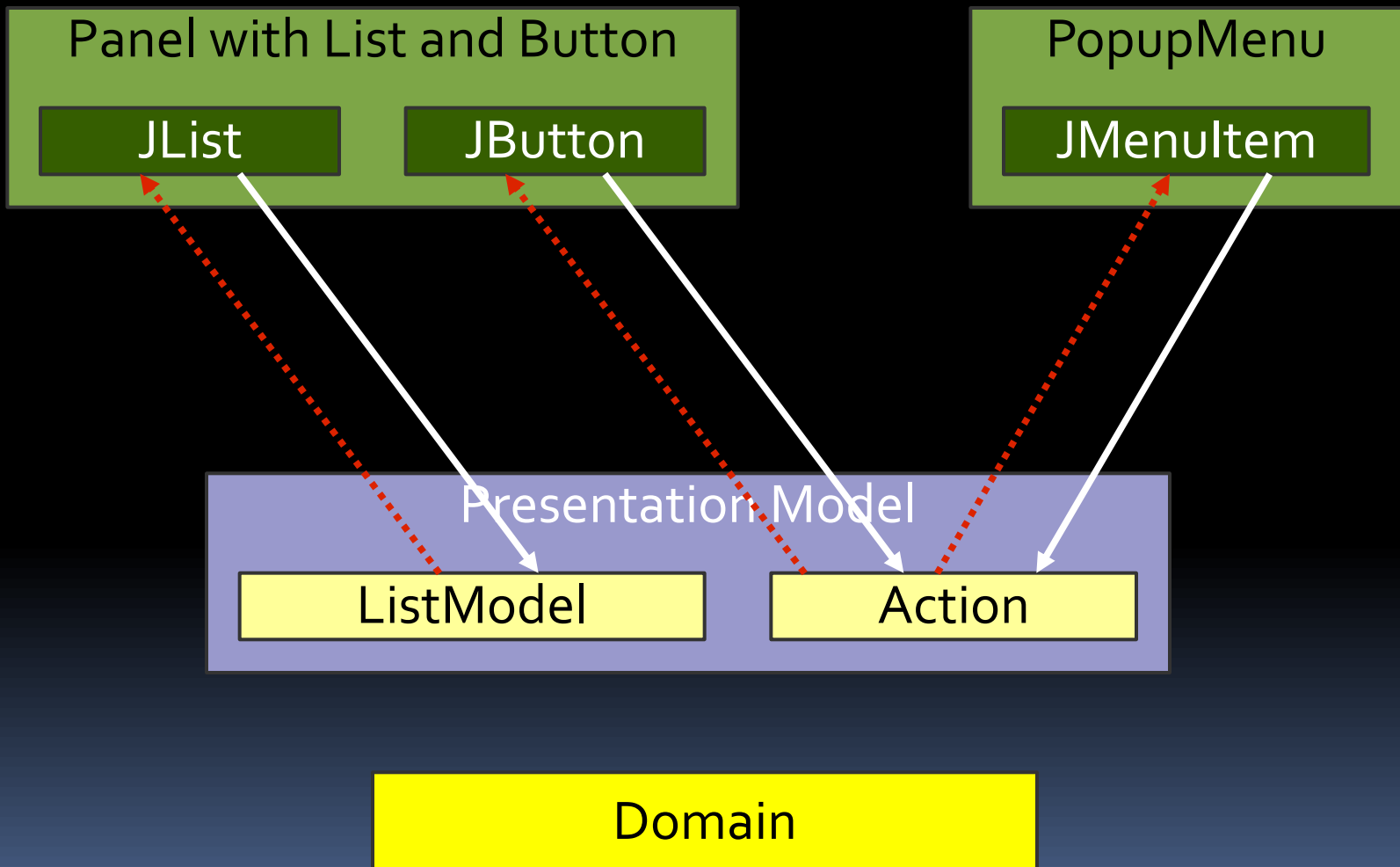
Pattern: Presentation Model



Action with Multiple Views



PM: Multiple Views



Implementation Patterns

For details see:

JGoodies.com -> Downloads -> Presentations

„Desktop Patterns & Data Binding“

Agenda

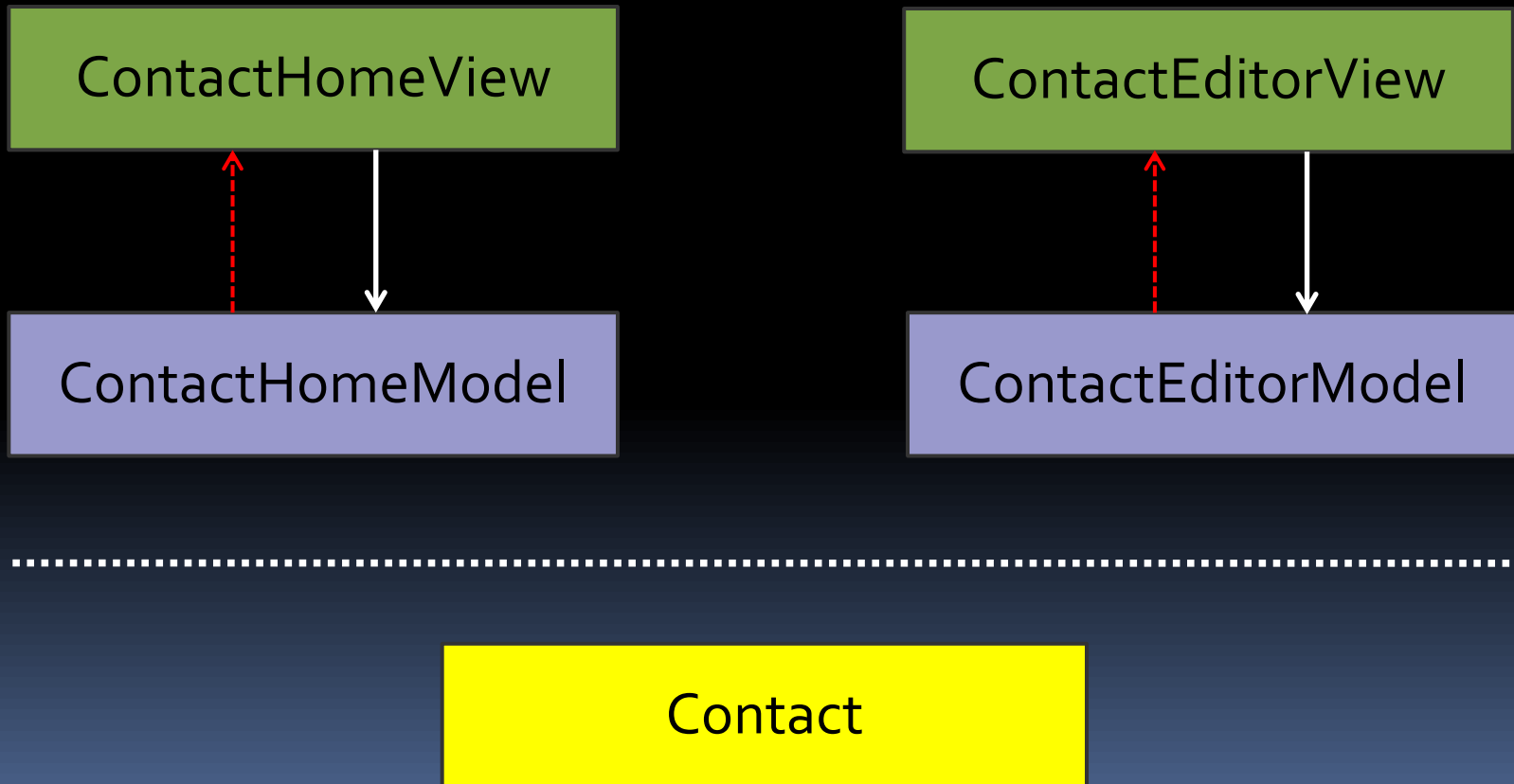
Visual Dos and Don'ts

Implementation Patterns

Writing Style

Visual Patterns

Contact Example



ContactEditorView

```
F > ContactEditorView 1.19
  ▲ ContactEditorView(ContactEditorModel)
  ■ initComponents() : void
  ■ initBindings() : void
  ▲ showDialog(EventObject) : void
  ■ buildContent() : JComponent
```

ContactEditorView (1/4)

```
final class ContactEditorView {  
  
    private final ContactEditorModel model;  
  
    private JTextField givenNameField;  
    private JTextField surnameField;  
    private JTextField phoneField;  
    ...  
    private JButton okButton;  
  
    ContactEditorView(ContactEditorModel model) {  
        this.model = model;  
        initComponents();  
        initBindings();  
    }  
}
```

Tip 3

- **Build** dialogs, frames, panels
- **Extend** JDialog, JFrame, JPanel if necessary.
Do you extend or use HashMap?

Tip 4

- Use view superclasses judiciously

Too Many Superclasses

JGoodiesAbstractDialog

JGoodiesDefaultDialog

CompanyAbstractDialog

CompanyDefaultDialog

CompanyValidationDialog

ProjectDefaultDialog

Tip 4

- Use view superclasses judiciously
- Favor composition over inheritance
- Consider interfaces or a single superclass for:
 - ensuring consistency
 - life-cycle

Tip 5

- Write for the reader
- Reduce visibilities
 - Classes
 - Fields
 - Methods
- Mark as final
- Follow Joshua Blochs „Effective Java“ tips

ContactEditorView (2/4)

```
private void initComponents() {
    JGComponentFactory factory =
        JGComponentFactory.getCurrent();

    givenNameField = factory.createTextField();
    surnameField   = factory.createTextField();
    phoneField     = factory.createPhoneField();
    faxField       = factory.createFaxField();
    emailField     = factory.createEmailField();
    tagsField      = factory.createTextField();
    Options.setSelectOnFocusGainEnabled(
        tagsField, false);

    okButton = factory.createButton(); // No text
}
```

...

Tip 6

- Use a component factory
 - extensible
 - exchangeable
 - shared by multiple projects
 - project specific
- Consider special component types, e.g. for: Email, Money, Power, Weight, Lengths, etc.

ContactEditorView (3/4)

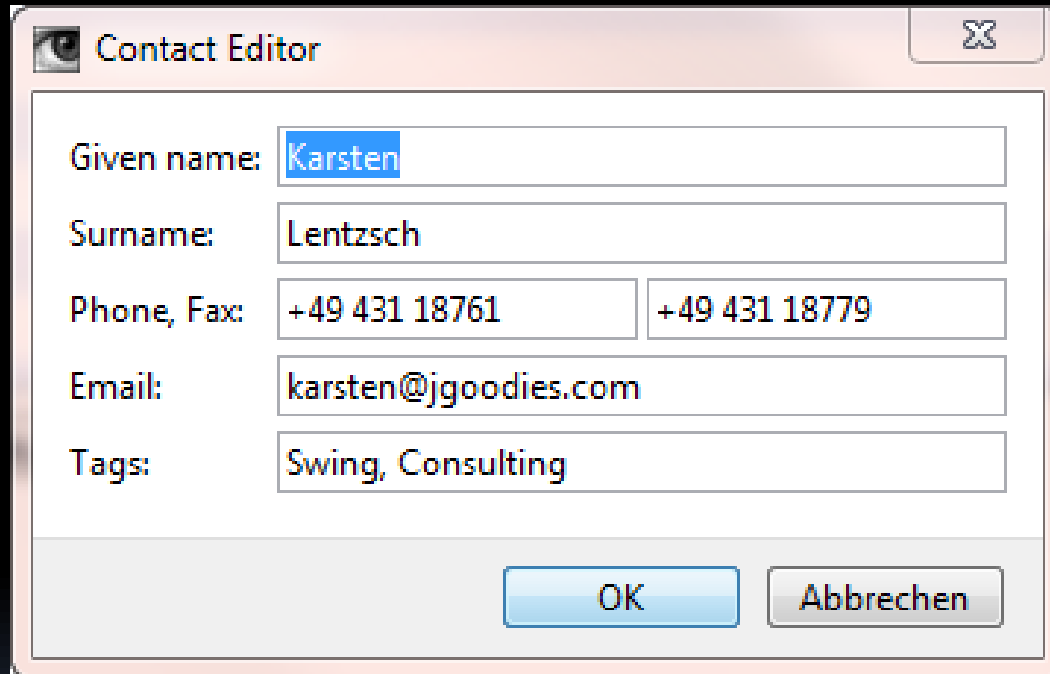
```
private void initBindings() {  
    Binder binder = Binders.binderFor(model);  
  
    binder.bindBeanProperty("givenName")  
        .to(givenNameField);  
    binder.bindBeanProperty("surname")  
        .to(surnameField);  
    binder.bindBeanProperty("phone")  
        .to(phoneField);  
  
    ...  
    binder.bindAction("OK").to(okButton);  
}  
  
...
```

ContactEditorView (3a/4)

```
private void initBindings() {
    Binder binder = Binders.binderFor(model);

    binder.bindBeanProperty(PROPERTY_GIVEN_NAME)
        .to(givenNameField);
    binder.bindBeanProperty(PROPERTY_SURNAME)
        .to(surnameField);
    binder.bindBeanProperty(PROPERTY_PHONE)
        .to(phoneField);
    ...
    binder.bindAction(ACTION_OK).to(okButton);
}
...
```


Contact Editor



Contact Editor

Given name: Karsten

Surname: Lentzsch

Phone, Fax: +49 431 18761 +49 431 18779

Email: karsten@jgoodies.com

Tags: Swing, Consulting

OK Abbrechen

ContactEditorView (3/4)

```
private JComponent buildContent() {
    FormLayout layout = new FormLayout(
        "pref, $lgap, 74dlu, 2dlu, 74dlu",
        "p, $lg, p, $lg, p, $lg, p, $lg, p");

    PanelBuilder builder = new PanelBuilder(layout);
    builder.addLabel("&Given name:", CC.xy (1, 1));
    builder.add(givenNameField, CC.xyw(3, 1, 3));
    builder.addLabel("&Surname:", CC.xy (1, 3));
    builder.add(surnameField, CC.xyw(3, 3, 3));
    builder.addLabel("&Phone, Fax:", CC.xy (1, 5));
    builder.add(phoneField, CC.xy (3, 5));
    builder.add(faxField, CC.xy (5, 5));
    ...
    return builder.getPanel();
}
```

Tip 7

- Use a grid-based layout
 - MigLayout
 - JGoodies FormLayout
- Avoid 2-pass-code
 - Describe the grid first
 - Then fill it with components – without state

Tip 8

- Favor flat over nested layouts
- Nest sublayouts that are not aligned
- Write #build-methods for sublayouts

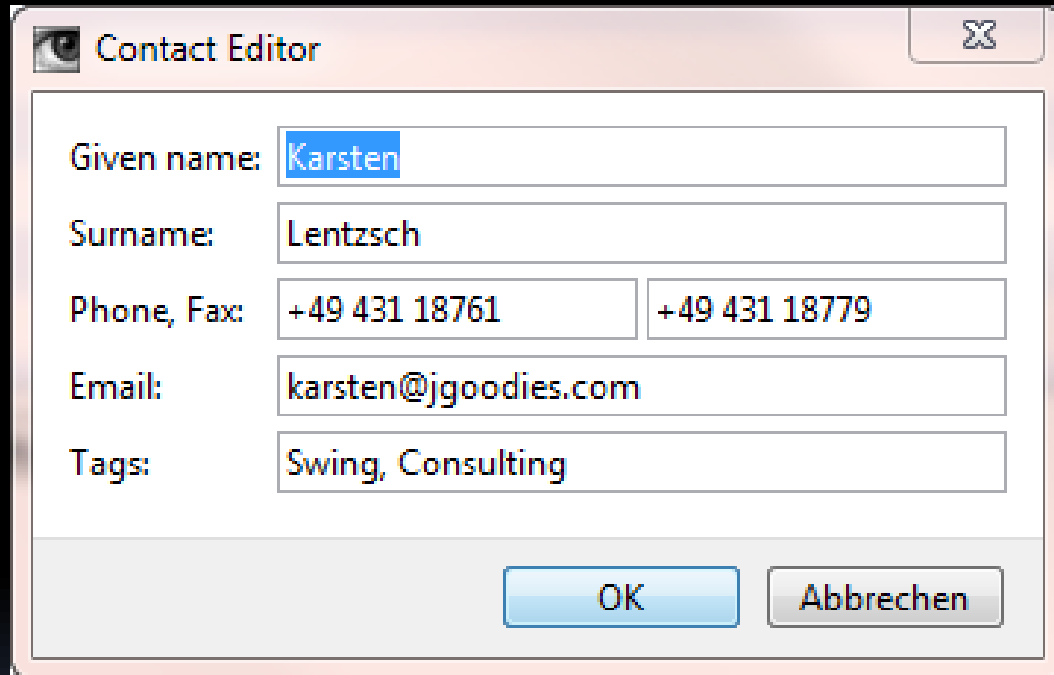
ContactEditorView (4/4)

```
void showDialog(EventObject e) {  
    PropertyPane pane = new PropertyPane(  
        buildContent(),  
        okButton, CommandValue.CANCEL);  
  
    pane.showDialog(e, "Contact Editor");  
}
```

Tip 9

- Separate content from decoration
- Here:
 - Editor content
 - Property dialog command area, borders, etc.

Contact Editor



Contact Editor

Given name:

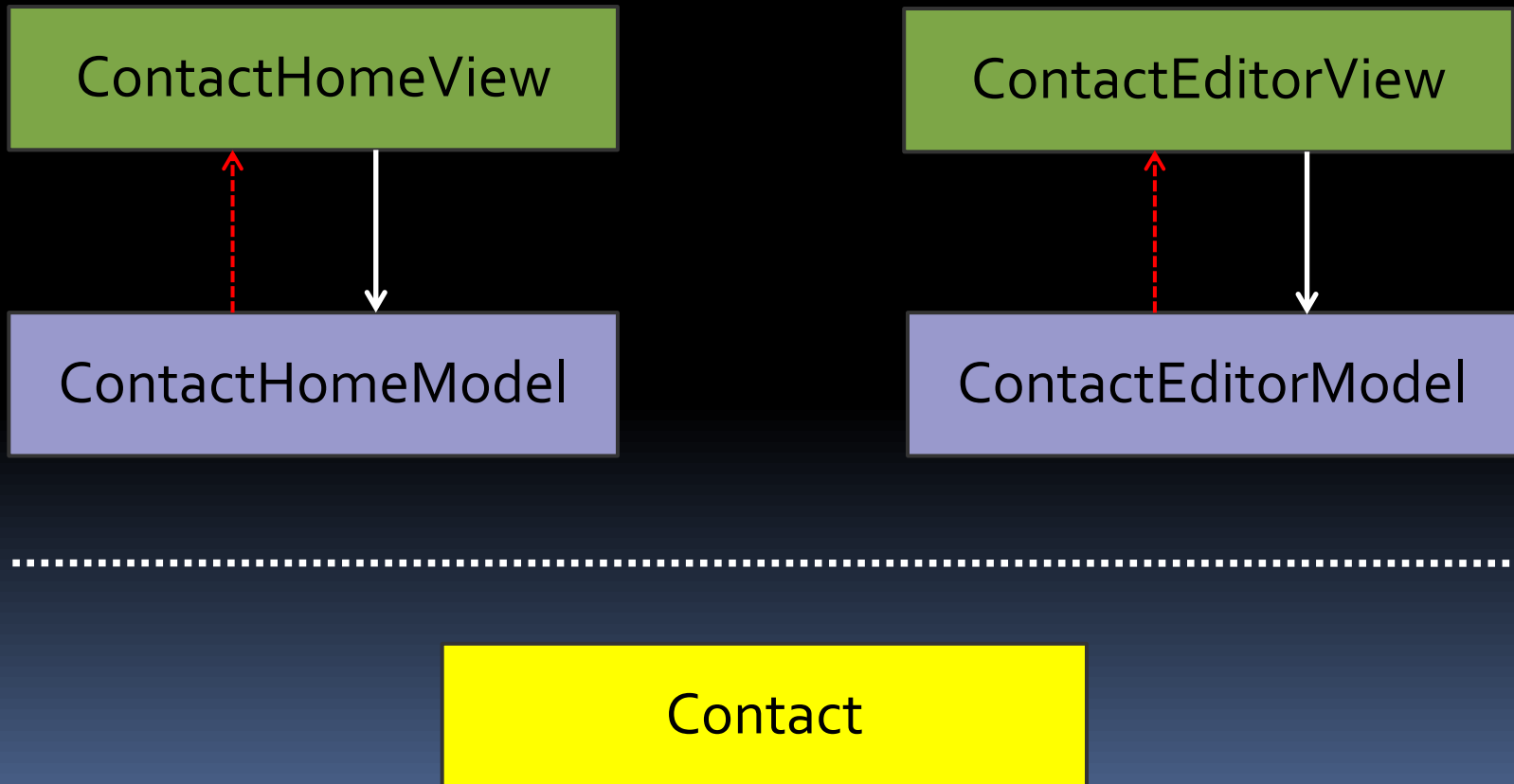
Surname:

Phone, Fax:

Email:

Tags:

Contact Example



ContactEditorModel (1/2)

```
public final class ContactEditorModel
    extends ActionPresentationModel<Contact> {

    private final ContactService service;
    private final EventBus eventBus;

    public ContactEditorModel(
        Contact contact,
        ContactService service,
        EventBus eventBus) {
        super(contact);
    }
}
```

Tip 10

- Illustrate the object context
 - Which objects are used?
 - Who reads/writes data?
 - How will changes be reported to others?
- E.g. use **Constructor Injection**

ContactEditorModel

```

G F ContactEditorModel 1.9
  ▲ C ContactEditorModel(Contact, ContactService, EventBus)
  ● onOKPerformed(ActionEvent) : void
  ▲ G SF SaveTask
    ▲ C SaveTask(Contact, ContactService, EventBus, boolean)
    ◆ ▲ doInBackground() : Contact
    ◆ ▲ succeeded(Contact) : void
    ◆ ▲ failed(Throwable) : void
```

Tip 11

- Consider a shorthand for Actions
- Know the @Action mechanism of the JSR 296 – Swing Application Framework

ContactEditorModel (2/2)

```
@Action
public void onOKPerformed(ActionEvent e) {
    // Save the data
    ...
}
```

ContactEditorModel (2a/2)

```
@Action(text="OK")
public void onOKPerformed(ActionEvent e) {
    // Commit pending edits
    TextComponentUtils.commitImmediately();
    // Save the data
    ...
}
```

Tip 12

- Know the JSR 296 – Swing App Framework
 - organizes, simplifies, standardizes
 - Resource management
 - Action management
 - Background tasks

Swing Application Framework

For details see:

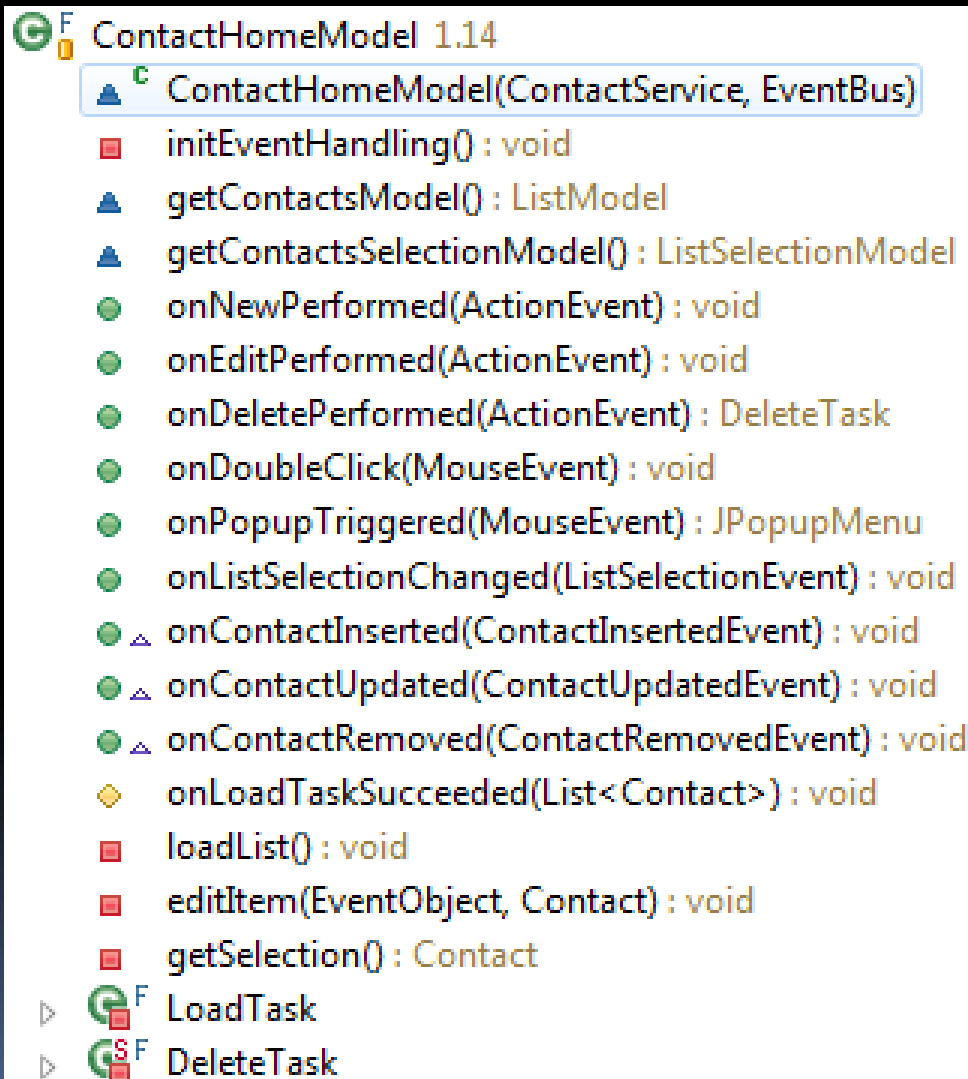
JGoodies.com -> Downloads -> Presentations

„JSR 296 –Swing App Framework“

Tip 13

- Show clearly, which events are handled by the presentation logic
- Use a shorthand similar to @Action
 - PropertyChangedListener
 - ListSelectionListener
 - Popup trigger
 - Double-click

ContactHomeModel



The screenshot shows the class hierarchy for ContactHomeModel 1.14. The class ContactHomeModel(ContactService, EventBus) is highlighted. Below it, a list of methods is shown, each with a small icon indicating its type (e.g., constructor, public, private, abstract, etc.). At the bottom, two subclasses are visible: LoadTask and DeleteTask.

```
ContactHomeModel 1.14
├── ContactHomeModel(ContactService, EventBus)
│   ├── initEventHandling() : void
│   ├── getContactsModel() : ListModel
│   ├── getContactsSelectionModel() : ListSelectionModel
│   ├── onNewPerformed(ActionEvent) : void
│   ├── onEditPerformed(ActionEvent) : void
│   ├── onDeletePerformed(ActionEvent) : DeleteTask
│   ├── onDoubleClick(MouseEvent) : void
│   ├── onPopupTriggered(MouseEvent) : JPopupMenu
│   ├── onListSelectionChanged(ListSelectionEvent) : void
│   ├── onContactInserted(ContactInsertedEvent) : void
│   ├── onContactUpdated(ContactUpdatedEvent) : void
│   ├── onContactRemoved(ContactRemovedEvent) : void
│   ├── onLoadTaskSucceeded(List< Contact >) : void
│   ├── loadList() : void
│   ├── editItem(EventObject, Contact) : void
│   └── getSelection() : Contact
├── LoadTask
└── DeleteTask
```

ContactHomeModel

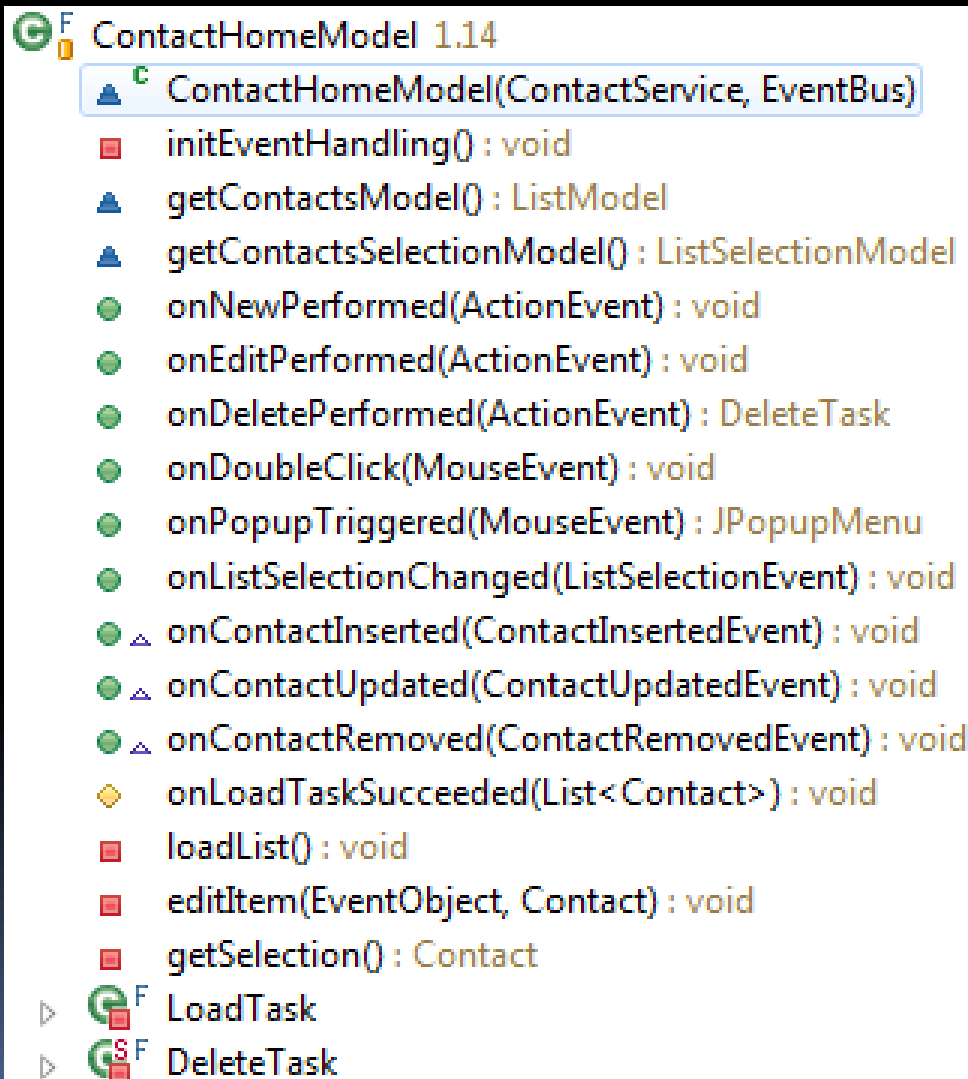
```
@ListSelectionListener
public void onListSelectionChanged(
    ListSelectionEvent e) {
    // Enable/disable the Edit and Delete Action
}
```

```
@DoubleClickListener
public void onDoubleClickPerformed(MouseEvent e) {
    editItem(e, getSelection());
}
```

Tip 14

- Standardize the client source structure
 - Sections
 - (Method) names
 - Method order
 - API

ContactHomeModel



The screenshot shows the class hierarchy for ContactHomeModel 1.14. The class is ContactHomeModel(ContactService, EventBus). The methods listed are:

- initEventHandling() : void
- getContactsModel() : ListModel
- getContactsSelectionModel() : ListSelectionModel
- onNewPerformed(ActionEvent) : void
- onEditPerformed(ActionEvent) : void
- onDeletePerformed(ActionEvent) : DeleteTask
- onDoubleClick(MouseEvent) : void
- onPopupTriggered(MouseEvent) : JPopupMenu
- onListSelectionChanged(ListSelectionEvent) : void
- onContactInserted(ContactInsertedEvent) : void
- onContactUpdated(ContactUpdatedEvent) : void
- onContactRemoved(ContactRemovedEvent) : void
- onLoadTaskSucceeded(List< Contact >) : void
- loadList() : void
- editItem(EventObject, Contact) : void
- getSelection() : Contact

Below the methods, there are two sub-classes:

- LoadTask
- DeleteTask

Duty

- Perform time-consuming tasks **outside** the Event-Dispatch-Thread
 - I-/O-operations
 - Server access
 - Service calls

Duty

- Perform time-consuming tasks **outside** the Event-Dispatch-Thread
 - I-/O-operations
 - Server access
 - Service calls

Tip 15

- Consider performing long tasks **in** the Event-Dispatch-Thread, if
 - most tasks perform in the twinkling of an eye
 - you cannot block the GUI [correctly]
 - your team doesn't master Swing's 1-thread-rule

Tip 16

- Write short
- Favor clear over short code

More vs. Less

```
private void initComponents() {  
    ...  
    givenNameField = factory.createTextField();  
    ...  
}
```

```
private void initBindings() {  
    binder.bindBeanProperty(PROPERTY_GIVEN_NAME)  
        .to(givenNameField);  
    ...  
}
```

```
private void initComponents() {  
    givenNameField = bindingFactory  
        .createBoundTextField(PROPERTY_GIVEN_NAME);  
    ...  
}
```

Very Short

```
private JComponent buildContent() {  
    return BeanBuilder.createEditor(Contact.class);  
}
```

Tip 17

- Avoid implicit operations, that
 - Nobody knows
 - Nobody understands
 - Nobody documents
 - Nobody maintains
 - You can't debug
- Automatic data binding
- Automatic form construction using reflection

Tip 18

- Use views, that can be shown without context
- Already done in MVP
- Possible in Presentation Model

Model Access in PM-View

```
private void initComponents() {  
    ...  
    okButton = new JButton(  
        // NPE if model == null  
        model.getAction(ACTION_OK));  
    ...  
}
```

```
ContactEditorView(ContactEditorModel model) {  
    this.model = model;  
    initComponents();  
    if (model != null) {  
        initBindings();  
    }  
}
```

Tip 19

- Write JavaDocs that increase the readability
 - Avoid repeating method names
 - Avoid empty tags
 - Follow standards (write in 3rd person, etc.)
 - Does the comment add value or blur sources?
 - Use static checks (by the IDE, CheckStyle)

Criteria for Good Style

- Short
- Simple, easy to understand
- Stable against changes
 - Pattern – MVP vs. PM
 - With or without visual editor
 - Component factory
 - Binding system
 - Layout
 - Toolkit
 - Language

Tip 20: Clean Up

- Factor out:
 - Domain data, business logic, services
 - Component construction
 - Action construction, Listener construction
 - Resource management
 - Event system (EventBus)
 - Mechanism for background tasks
 - Standard layouts
 - Standard dialogs

Agenda

Visual Dos and Don'ts

Implementation Patterns

Writing Style

Visual Patterns

ContactHomeView

```

ContactHomeView 1.19
├── ContactHomeView(ContactHomeModel)
│   ├── initComponents() : void
│   ├── initBindings() : void
│   ├── initEventHandling() : void
│   ├── buildPanel() : JComponent
│   └── buildButtonBar() : JComponent
└── ContactTableModel

```

ContactHomeView Panel Layout

```
private JComponent buildPanel() {
    FormLayout layout = new FormLayout(
        "fill:250dlu:grow",
        "p, $lcg, fill:200dlu, $rgap, p");

    PanelBuilder builder = new PanelBuilder(layout);
    builder.addLabel("&Contacts:", CC.xy(1, 1));
    builder.add(
        new JScrollPane(contactsTable), CC.xy(1, 3));
    builder.add(buildButtonBar(), CC.xy(1, 5));
    return builder.getPanel();
}
```

ContactHomeView Button Bar I

```
private JComponent buildButtonBar() {
    FormLayout layout = new FormLayout(
        "pref, 4px, pref, 4px, pref",
        "p");

    PanelBuilder builder = new PanelBuilder(layout);
    builder.add(newButton,    CC.xy(1, 1));
    builder.add(editButton,   CC.xy(3, 1));
    builder.add(deleteButton, CC.xy(5, 1));

    return builder.getPanel();
}
```

ContactHomeView Button Bar II

```
private JComponent buildButtonBar() {  
    return new ButtonBarBuilder()  
        .addButton(newButton, editButton, deleteButton)  
        .getPanel();  
}
```


Tip 21

- Search for similar panels, subpanels, forms, ...
- Build it consistently with:
 - Panel building code (nested)
 - Grid filling code (embedded)
 - Visual components (Beans)

Anwendungen:

Belege

Kunden

Verwaltung:

Länder

Rechnungscodes

Belege

Status *	Nummer	Erstellt	Empfänger	Betreff	Endsumme
B	2006-06-01	05.07.06	Schenker Düsseldorf	Beratung Mai 2006	1.200,34 EUR
B	2006-06-02	05.07.06	Schenker Düsseldorf	Consulting May 2006	2.300,45 USD
B	2006-06-03	08.07.06	Schenker Düsseldorf	Abschlag Projekt1	10.000,00 EUR
B	2006-06-04	08.07.06	Air France	Endrechnung Projekt1	15.000,00 EUR
S	2006-07-01	01.08.06	Air France	Beratung Juni 2006	1.200,34 EUR
S	2006-07-02	01.08.06	Schenker Düsseldorf	Reisekosten Juni 2006	2.300,45 EUR
S	2006-07-03	03.08.06	JGoodies Karsten Lentzsch	Abschlag Projekt1	10.000,00 EUR
S	2006-07-04	03.08.06	British Airways	Endrechnung Projekt1	15.000,00 EUR
E	2006-08-01	02.05.11	British Airways	Beratung Juni 2006	1.000,00 EUR
E	2006-08-02	02.05.11	Schenker Düsseldorf	Beratung Juli 2006	2.000,00 EUR
E	2006-08-03	02.05.11	Schenker Düsseldorf	Abschlag Projekt1	10.000,00 EUR
E	2006-08-04	02.05.11	Schenker Düsseldorf	Endrechnung Projekt1	15.000,00 EUR

Neue Rechnung

Neue Gutschrift

Öffnen

Drucken...

*) E=Entwurf, S=eingereicht, B=bezahlt

Schenker Deutschland AG
Geschäftsstelle Düsseldorf
Wanheimer Straße 61
40472 Düsseldorf

Rechnung 2006-06-01 vom 05.07.06
Beratung Mai 2006

Erstellt: 05.07.2006
Status: Bestätigt
Endsumme: 10.348,32 EUR
13.123,45 USD

Offen:

R2006-06-01*

R2006-06-03*

- [-] Strecken
 - [-] Strecke 1
 - [-] EW 7 / 1445
 - [-] EW 7
 - ▲ **Antrieb HW61, AVV_ZVW**
 - ◇ Heizung HWH
 - Steuerung HN-P 7 / 1445
 - [-] Außenanlagen
 - [-] EW 12 / 1449
 - [-] Strecke 2
 - [-] Strecke 3
 - [-] Strecke 4
 - [-] Strecke 5
- [-] Datensammler
 - [-] GUV Albertplatz
 - [-] Btf. Gorbitz
 - [-] Btf. Reich (wird angepasst)
 - [-] Btf. Trachenberge
 - [-] GUV Coswig
 - [-] GUV Jahnstraße
 - [-] GUV Meschwitzstrasse
 - [-] GUV Postplatz
- [-] Anlagen

Antrieb EW 7 - Strecke 1
Friedrichstr. - Maxstr./Köneritzstr.

Status: In Betrieb
Montage: Gleismitte
Ausführung: Flachbettweiche

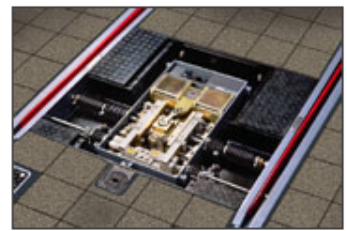


Gerätenr.: 320 513
EDV-Nr.: 109 244 318
Einbau: 12.03.2005
Inspektion: 21.02.2007

[Eigenschaften bearbeiten](#)

Produkt

Baureihe:	HW 61	Stellkraft:	5000 N
Antriebsform:	Elektromagnetisch	Verschluß:	Ja
Variante:	AVV-ZVW	Feder:	Ja
Art:	Umstellweiche	Dämpfung:	Ja
Zungenaufschlag:	30 - 70 mm	Richtungsschalter:	nein
Höhe mit Kasten:	300 mm	Zungenprüfer:	Ja
Höhe ohne Kasten:	205 mm	Verschlossen:	Ja
Betriebsspannung:	600/750 V DC	Auffahrbar:	Ja



[Produktdetails zeigen](#)

Baugruppen:

Bezeichnung	EDV-Nummer	Gerätenr.	Einbau	Bemerkung
Erdkasten		320 514	12.03.2005	
Doppelzugmagnet	30032002		12.03.2005	
Antriebsswelle	31051008		30.11.2006	

ListViewBuilder

```
private JComponent buildPanel() {  
    return new ListViewBuilder()  
        .title("&Contacts:")  
        .listView(contactsTable)  
        .listBar(newButton, editButton, deleteButton)  
        // .detailsView(...)  
        .build();  
}
```

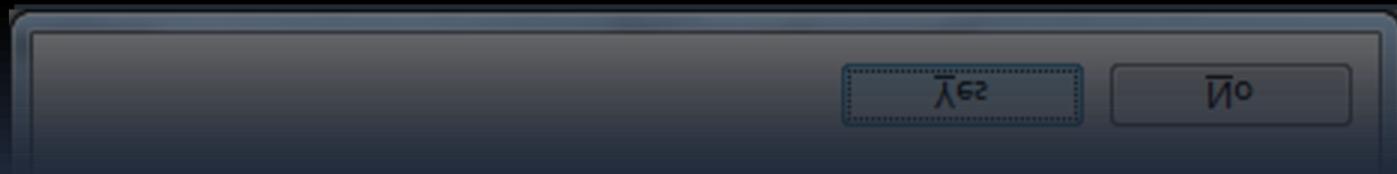
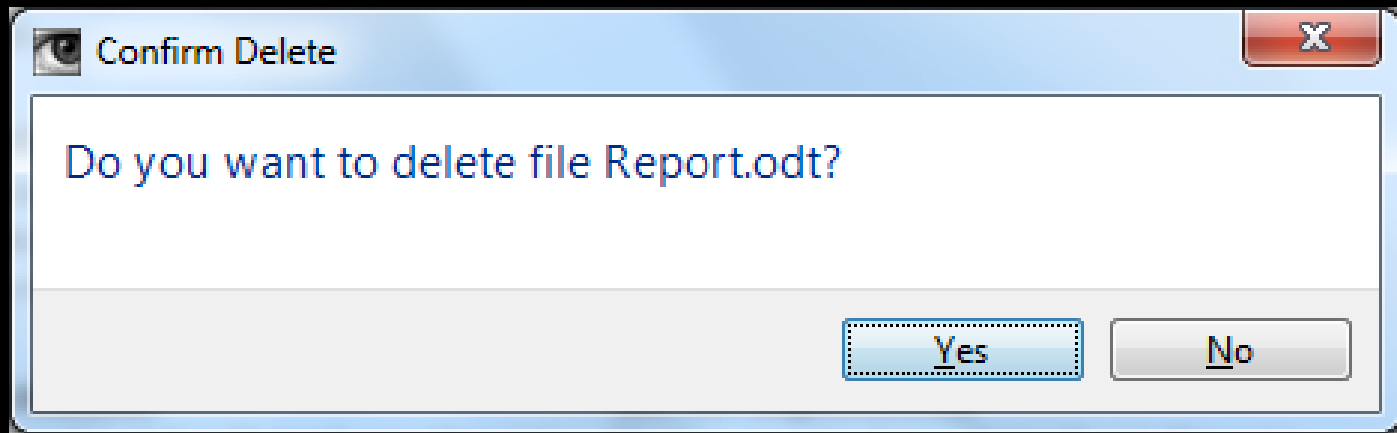
ContactHomeModel

```
@Action
public void onDeletePerformed(ActionEvent e) {
    Contact selection = view.getSelection();
    String objectName = selection.getDisplayString();

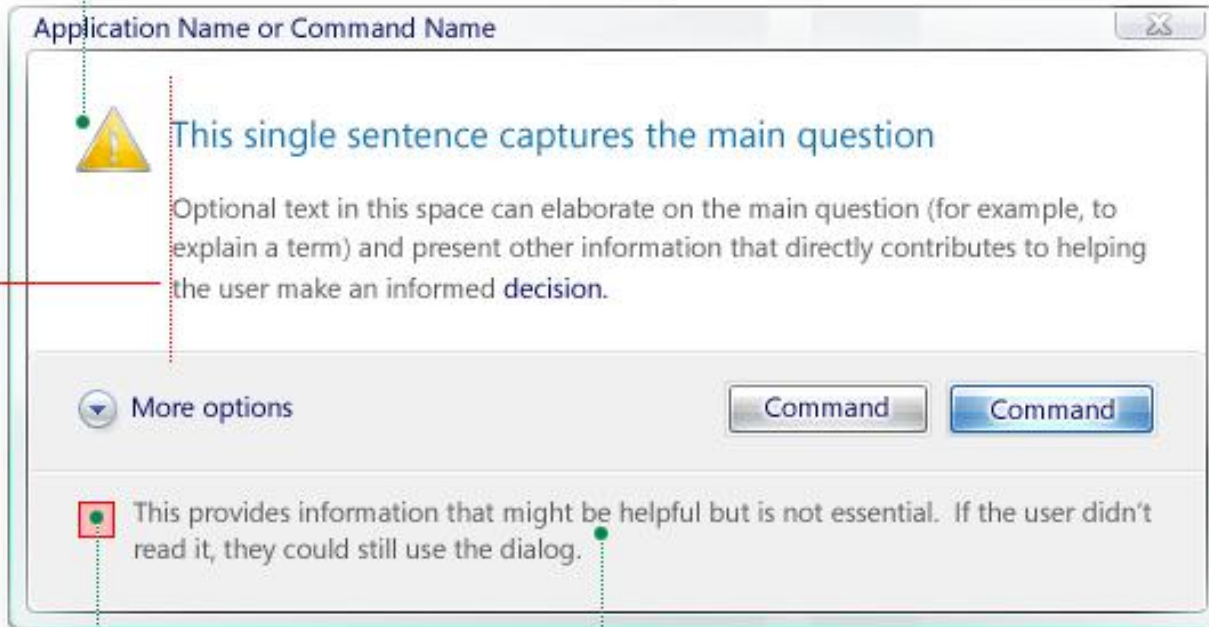
    TaskPane pane = new TaskPane(
        MessageType.QUESTION,
        "Do you want to delete " + objectName + "?",
        CommandValue.YES, CommandValue.NO);

    pane.showDialog(e, "Delete Contact");

    if (pane.getCommitValue() == CommandValue.YES) {
        // Delete the selection
    }
}
```



<part name="MainIcon">



With icon in the header area, indent descriptive information and left-align with main instruction

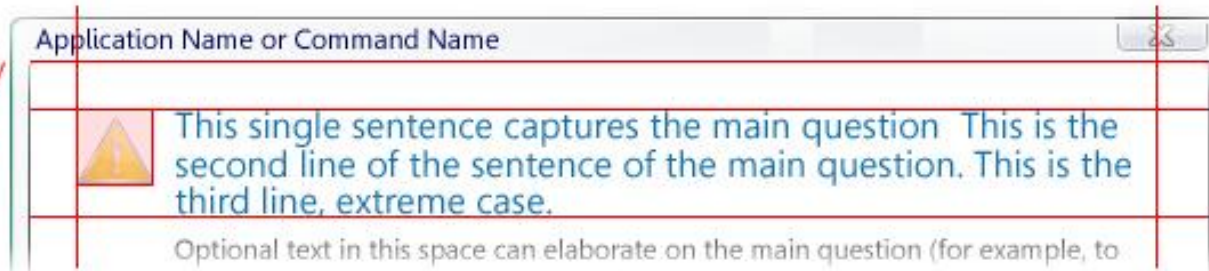
<part name="FootnoteIcon">

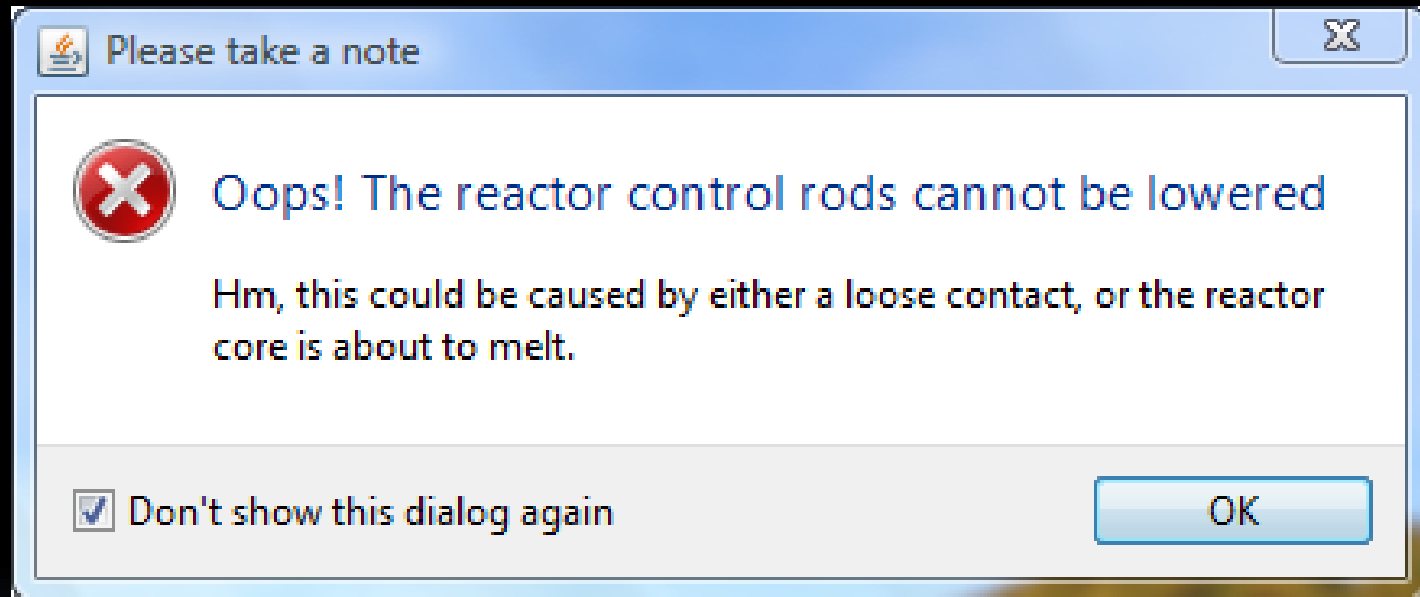
<part name="FootnotePane">

##TextStyle.BodyText.Font

<TextColor>##TextStyle.BodyText.TextColor</TextColor>

Task dialog automatically expands and top aligns longer main instruction text strings





Don't show this dialog again

OK

Don't show this dialog again

OK



Save Resource



'IconFeedbackPanel.java' has been modified. Save changes?

Yes

No

Cancel



JGoodies Showcase



Save changes to Invoice 2010-09-03?

Save

Don't save

Cancel

Save

Don't save

Cancel

Style Guide Compliance API

```
@ActionListener
public void onDeletePerformed(ActionEvent e) {
    Contact selection = view.getSelection();
    String objectName = selection.getDisplayString();

    boolean proceed = MessagePanes.getCurrent()
        .showConfirmation(e,
            "Delete Contact",
            "Do you want to delete " + objectName + "?");

    if (proceed) {
        // Delete the selection
    }
}
```

Class MessagePanels

`#showError(...)`

`#showAwarenessWarning(...)`

`#showImminentProblem(...)`

`#showInformation(...)`

`#showConfirmation(...)`

`#showRiskyActionConfirmation(...)`

`#showHelp(...)`

...

Standard Dialogs

```
@ActionListener
public void onDeletePerformed(ActionEvent e) {
    Contact selection = view.getSelection();
    String objectName = selection.getDisplayString();

    boolean proceed = StandardPanels.getCurrent()
        .showDeleteConfirmation(e, objectName);

    if (proceed) {
        // Delete the selection
    }
}
```

Class StandardPanels

`#showDeleteConfirmation (...)`

`#showRiskyDeleteConfirmation (...)`

`#showExitConfirmation (...)`

`#showSaveChangesConfirmation (...)`

`#showSaveAllChangesConfirmation (...)`

`#showNotYetImplemented (...)`

`#showNotYetTestedConfirmation (...)`

`...`

Visual Patterns (Meta Design)

For more information see:

JGoodies.com -> Downloads -> Presentations

„Efficient Swing Design“

More Information

- Martin Fowler: *Further P of EAA*
 - google "Organizing Presentation Logic"
- Chris Ramsdale:
Large scale application development and MVP

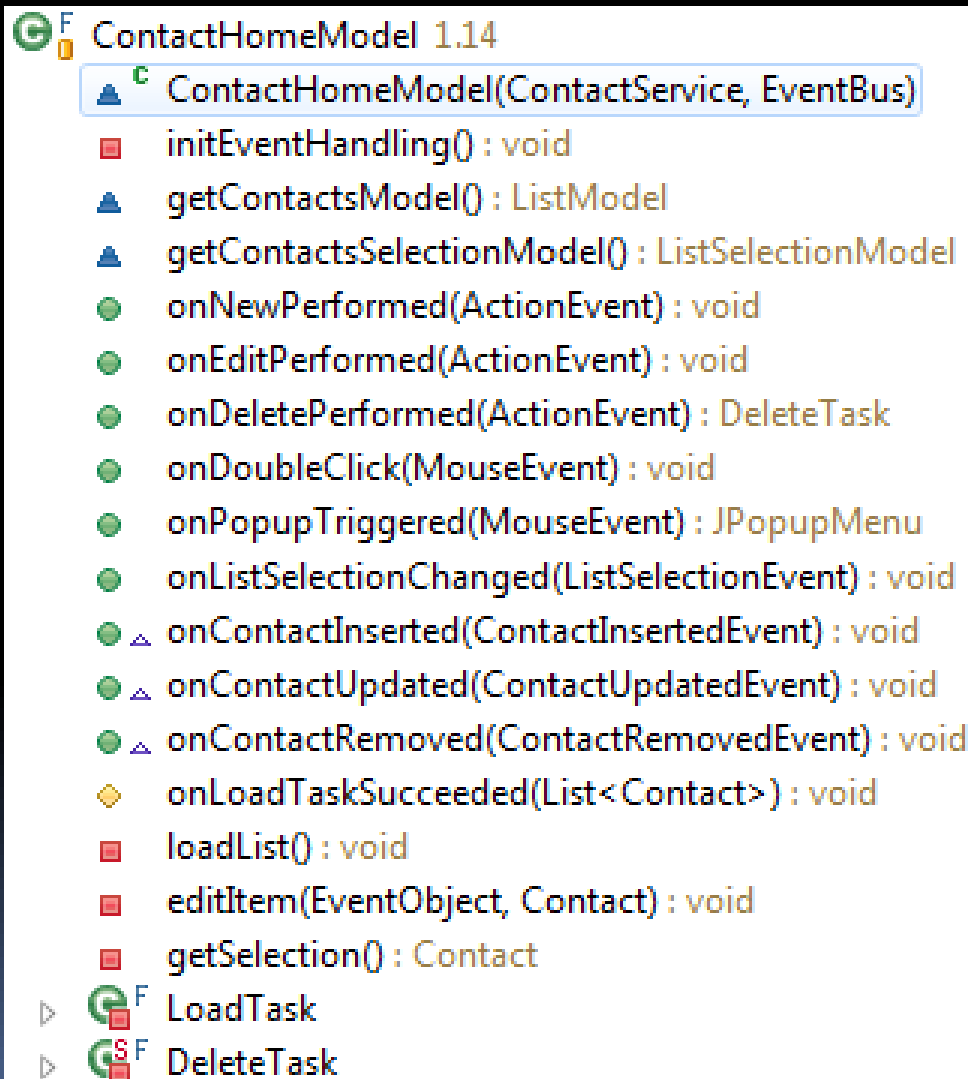
More Information: Human Factors

See the video:

„Warum so viele kluge Leute so schlechte Oberflächen entwickeln“ – German only

„Why so many smart people develop poor GUIs“

ContactHomeModel



The screenshot shows the class hierarchy for ContactHomeModel 1.14. The class is ContactHomeModel(ContactService, EventBus). The methods listed are:

- initEventHandling() : void
- getContactsModel() : ListModel
- getContactsSelectionModel() : ListSelectionModel
- onNewPerformed(ActionEvent) : void
- onEditPerformed(ActionEvent) : void
- onDeletePerformed(ActionEvent) : DeleteTask
- onDoubleClick(MouseEvent) : void
- onPopupTriggered(MouseEvent) : JPopupMenu
- onListSelectionChanged(ListSelectionEvent) : void
- onContactInserted(ContactInsertedEvent) : void
- onContactUpdated(ContactUpdatedEvent) : void
- onContactRemoved(ContactRemovedEvent) : void
- onLoadTaskSucceeded(List< Contact >) : void
- loadList() : void
- editItem(EventObject, Contact) : void
- getSelection() : Contact

Below the methods, there are two sub-classes:

- LoadTask
- DeleteTask

QUESTIONS AND ANSWERS

Karsten Lentzsch – JGoodies

SWING WITH STYLE