Informatik B - Objektorientierte Programmierung in Java

Vorlesung 13:
GUI 3: Swing und SWT

Inhalt

- GUI-Programmierung mit Swing
- Swing-Komponenten
- Eclipse SWT

GUI-Programmierung mit Swing

- Das AWT gibt dem Entwickler die Möglichkeit, Benutzungsoberflächen zu entwickeln, die auf unterschiedlichen Plattformen lauffähig sind.
- Für die Ausführung eines AWT-basierten Programms, z.B. unter MS-Windows oder OSF Motif verwendet der Java-Interpreter aber eine plattformspezifische Implementierung der AWT-Klassen, die nicht in Java geschrieben ist.

GUI-Programmierung mit Swing

- Das bedeutet u.a., dass man keine völlige Kontrolle über das Aussehen eines Programms in einer anderen als der Entwicklungsumgebung hat.
- Um die Funktionalität und die Portabilität von Benutzungsoberflächen zu erhöhen, haben Sun und Netscape die Java Foundation Classes (JFC) entwickelt, die als Hauptbestandteil die Swing-Pakete enthalten.

GUI-Programmierung mit Swing

- Swing ist die Implementierung einer vollständig in Java geschriebenen Menge von Komponenten zum Aufbau von Benutzungsschnittstellen.
- Dabei stehen folgende Merkmale im Vordergrund:
  - Vollständige Implementierung in Java und Unabhängigkeit von plattformspezifischen Einschränkungen.
  - Konzeptionelle Trennung von Funktionalität und Aussehen einer Komponente
  - Trennung von Darstellung und Daten der Swing-Komponenten.

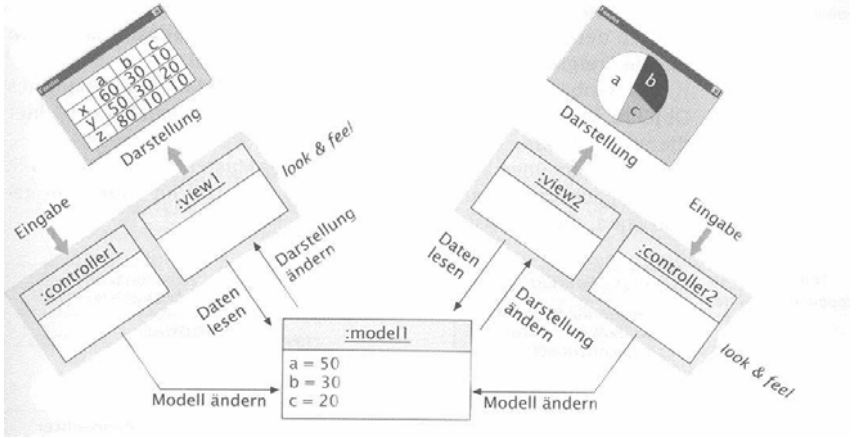
GUI-Programmierung mit Swing

- *Pluggable look-and-feel*: Die Swing-Komponenten können ein unterschiedliches Aussehen annehmen, unabhängig von der Plattform, auf der sie gerade ausgeführt werden.
- Einführung einer Reihe zusätzlicher Komponenten.
- Erweiterte Möglichkeiten der Fensterprogrammierung.

GUI-Programmierung mit Swing

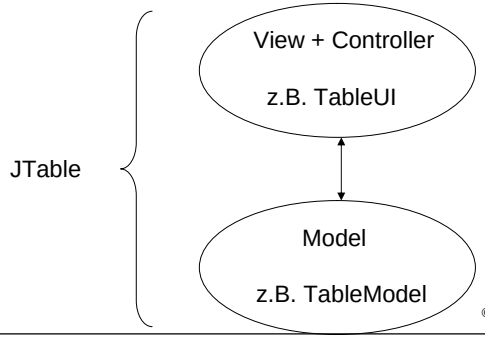
- Trennung von Darstellung und Daten
  - Die Swing-Komponenten sind nach dem sog. Model-View-Controller-Muster aufgebaut.
  - Aufgaben der drei Teile:
    - Model: Datenspeicherung
    - View: Darstellung der Daten für den Benutzer
    - Controller: Entgegennahme von Benutzereingaben
  - Da das Erscheinungsbild und spezifische Bedieneigenschaften fast immer zusammengehören („look-and-feel“) werden View und Controller bei Swing-Komponenten zusammengefasst (Delegate).

GUI-Programmierung mit Swing



GUI-Programmierung mit Swing

- Bei Swing-Komponenten spielen daher immer drei Klassen zusammen:
  - Die eigentliche Komponente (z.B. JTable),
  - ihr Datenmodell (z.B. TableModel)
  - und ihre UI-Klasse (z.B. TableUI).



Swing-Komponenten

- Überblick
  - Swing-Komponenten sind mit Ausnahme der Fensterklassen von der Oberklasse `JComponent` abgeleitet.
  - `JComponent` selbst ist eine Unterklasse von `java.awt.Container` (und daher von `Component`).
  - Für jede Komponente des AWT (außer `Choice` und `Canvas`) existiert eine Entsprechung in Swing (jeweils mit einem „J“ vorangestellt).

Swing-Komponenten

AWT (java.awt.*)	Swing (javax.swing.*)
Button	JButton
CheckBox	JCheckBox
Component	JComponent
Label	JLabel
List	JList
Menu	JMenu
MenuBar	JMenuBar
MenuItem	JMenuItem
Panel	JPanel
PopMenu	JPopMenu
ScrollBar	JScrollBar
ScrollPane	JScrollPane
TextArea	JTextArea
TextField	JTextField

Beispiele – jetzt mit Swing

- JAlleLayouts.java
- JControlTest.java
- JButtonMitListenerAnonym.java
- JMenuClipBoard.java

Swing-Komponenten

- Neben diesen Klassen, die jeweils eine direkte Entsprechung im Paket `java.awt` besitzen und auf ähnliche Weise programmiert werden, enthält Swing eine Reihe zusätzlicher Steuerelemente und Hilfsklassen.
- Dazu gehören u.a.:

Swing-Komponenten

Komponente	Beschreibung
JComboBox	Eine ComboBox, d.h. eine Kombination aus Klappliste und Texteingabefeld
JEditorPane	Eine Textfläche, die formatierten Text aufnehmen und darstellen kann (in verschiedensten Formaten, z.B. HTML oder RTF)
JOptionPane	Einfache Hinweis- und Abfragefenster (message boxes), z.B. für Ja/Nein-Entscheidungen
JPasswordField	Ein Textfeld für die verdeckte Eingabe z.B. von Passwörtern
JSlider	Ein Stellregler, über den man Werte eingeben kann
JSplitPane	Eine zweigeteilte Fläche, deren anteilige Sichtbarkeit über einen Schieber eingestellt werden kann
JTabbedPane	Eine Komponente, die mehrere Flächen von Komponenten hintereinander stapelt und mit Reitern versieht („Registerkarten“)
JTable	Eine Tabellenkomponente, die die flexible Programmierung von Tabellen beliebiger Struktur erlaubt
JToggleButton	Basisklasse für Wechselschalter (Schaltflächen mit zwei Zuständen, z.B. JRadioButton)
JToolBar	Eine Containerklasse für die Aufnahme von Komponenten, die entweder fest verankert an einer bestimmten Position im Fenster erscheint oder frei schwebend als losgelöstes Subfenster („Werkzeugleiste“)
JTree	Eine Komponente für die graphische Darstellung von Hierarchien (Baumdarstellung)

Swing-Komponenten

- Auch die Fensterklassen sind in Swing gegenüber dem AWT reimplementiert oder werden zusätzlich neu eingefügt:

AWT (java.awt.*)	Swing (java.swing.*)	Beschreibung
Applet	JApplet	Unterklasse von Applet mit zusätzlicher Funktionalität für die Fensterprogrammierung mit Swing
--	JColorChooser	Standard-Dialogfenster für die Farbwahl nach unterschiedlichen Farbmodellen
Dialog	JDialog	Basisklasse für Dialogfenster
FileDialog	JFileChooser	Standarddialogfenster für die Dateiauswahl
Frame	JFrame	Basisklasse für Fenster mit Rahmen
--	JInternalFrame	Subfenster mit Rahmen (enthalten in einem Hauptfenster z.B. vom Typ JFrame oder JApplet)
Window	JWindow	Basisklasse für ein einfaches Swing-Fenster ohne Rahmen

Swing-Komponenten

- Zusätzliche Funktionalität der Swing-Komponenten
  - Für alle Komponenten des Swing-Pakets steht eine Reihe zusätzlicher funktionaler Möglichkeiten zur Verfügung:
    - Rahmen für Komponenten (border)
      - Im Paket `javax.swing.border` wird die nötige Funktionalität bereitgestellt, um Rahmen unterschiedlichster Art zu definieren.
    - Verwendung von Tooltips
      - Ein ToolTip ist eine kleines Beschriftungsfenster, das über einer Komponente angezeigt wird, wenn man mit der Maus über der Komponente stehen bleibt.

Swing-Komponenten

- Tastaturunterstützung
  - Den Swing-Komponenten kann man in Abhängigkeit von Ihrem Zustand (sichtbar, mit/ohne Focus) Tastaturkombinationen zuweisen. Ähnlich wie bei der Zuweisung von Listener-Objekten registriert man eine Tastaturaktion für die Komponente.
- Ereignisverarbeitung
  - In Swing wird das Modell der Nachrichtenverarbeitung wie im AWT verwendet.
  - Die neuen Komponenten benötigen aber eine Reihe zusätzlicher Nachrichtentypen, die im Paket `javax.swing` zusammengefasst sind (z.B. `DocumentEvent` / `DocumentListener` für Ereignisse, die Änderungen an einem Textdokument signalisieren oder `TableModelEvent` / `TableModelListener` für Nachrichten, die signalisieren, dass sich der Inhalt des Modells einer Tabelle geändert hat.

Swing-Komponenten

- Layout und Viewport
  - Swing fügt den bekannten Layoutmanager-Klassen des AWT ein weiteres, sehr flexibles Layoutverfahren hinzu, das `BoxLayout`.
  - Darin werden alle Komponenten als Rechtecke behandelt, die mit Hilfe von Stützen (struts) und flexiblem „Klebstoff“ (glue) gesetzt werden können.
  - Da in vielen Fällen die Darstellungsfläche einer Komponente kleiner ist als die Komponente selbst, also nur ein Teil der Komponente angezeigt werden kann (z.B. sichtbarer Textausschnitt eines Dokuments in einem Texteditor), führt Swing das Konzept eines Viewports (`JViewPort`) und eines dazugehörigen Layoutmanagers (`ViewportLayout`) ein, der die Darstellung der Komponente in ihrem Sichtfenster regelt. Ein Viewport wird beispielsweise von Komponenten innerhalb einer Darstellungsfläche mit Rollbalken (`JScrollPane`) verwendet.

Swing-Komponenten

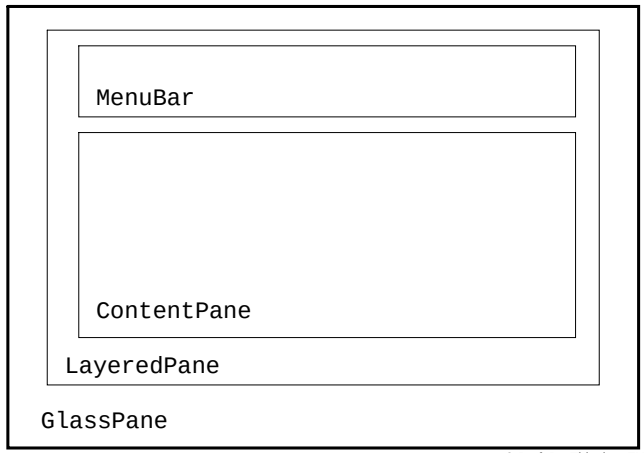
- Verwendung von Swing-Komponenten
  - Verwendung von Fenstern
    - Ein Unterschied zwischen dem AWT-`Frame` und `JFrame` in Swing ist die Aufteilung des `JFrame` in verschiedene Flächen.
    - Die beiden wichtigsten Basisklassen der Fensterprogrammierung in Swing – `JFrame` und `JApplet` – sind so strukturiert, dass kaskadierende Subfenster möglich sind.
    - Als Basisklasse für den Fensterinhalt dient ein Objekt vom Typ `JRootPane`, das in die folgenden Bestandteile gegliedert ist, auf die man über die Operationen von `JRootPane` zugreifen kann:

Swing-Komponenten

- `GlassPane`
  Eine nicht sichtbare Fläche, die u. a. dem Abfangen von Mausereignissen an das Fenster dient.
- `LayeredPane`
  enthält `ContentPane` und `MenuBar`.
- `ContentPane`
  Dies ist die Darstellungsfläche des Fensters oder Applets selbst. Soll in der Darstellungsfläche eines Fensters z.B. eine Komponente eingefügt werden, so geschieht dies nicht wie beim AWT-Frame durch `myFrame.add(myComponent)`, sondern die Komponente wird in die `ContentPane` des `JFrame` eingefügt: `myJframe.getContentPane().add(myComponent)`.
- `MenuBar`
  Die Menüleiste des Fensters. Sie gilt für das Hauptfenster und alle seine Subfenster.

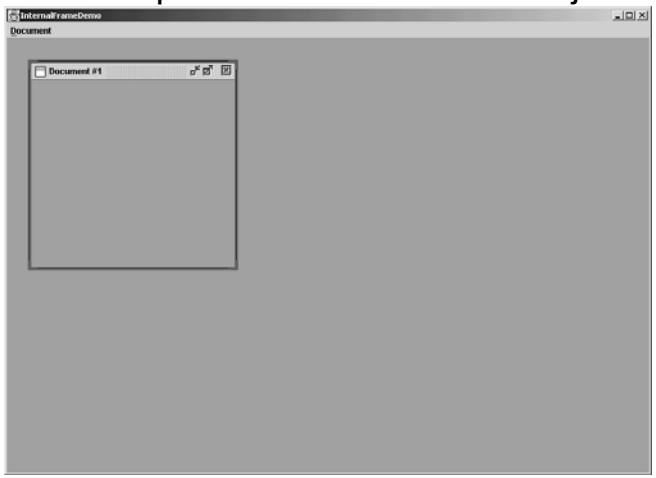
Swing-Komponenten

- Fensterobjekt (`JFrame`)



Swing-Komponenten

- Beispiel: InternalFrameDemo.java

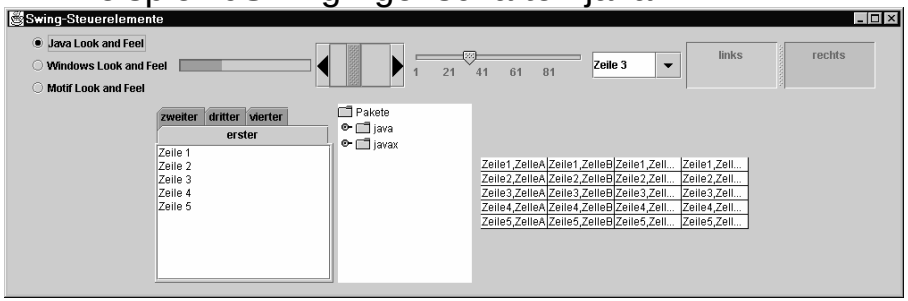


Swing-Komponenten

- Verwendung weiterer Swing-Elemente
  - Schaltflächen – `JRadioButton/JCheckBox`
  - Fortschrittsbalken – `JProgressBar`
  - Rollbalken – `JScrollBar`
  - Skala – `JSlider`
  - Editierbare Klapplisten – `JComboBox`
  - Splitviewer – `JSplitPanel`
  - Registrierkarten mit Reitern – `JTabbedPanel`
  - Visualisierung von Hierarchien – `JTree`
  - Tabellen – `JTable`
  - Siehe Beispiel Swing-Steuerelemente

Swing-Komponenten

- Beispiel: JSwingEigenschaften.java

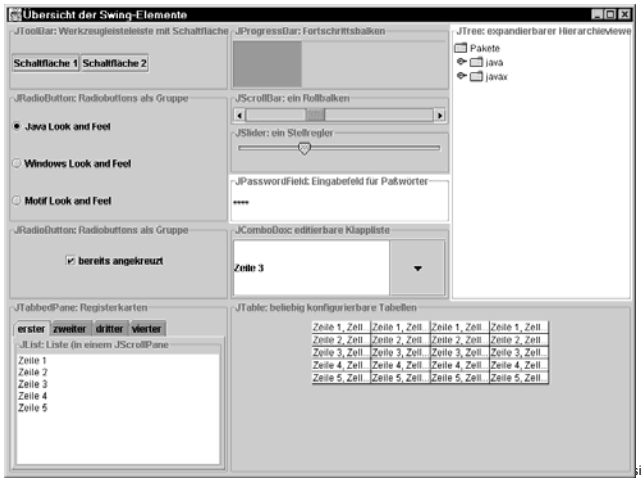


Swing-Komponenten

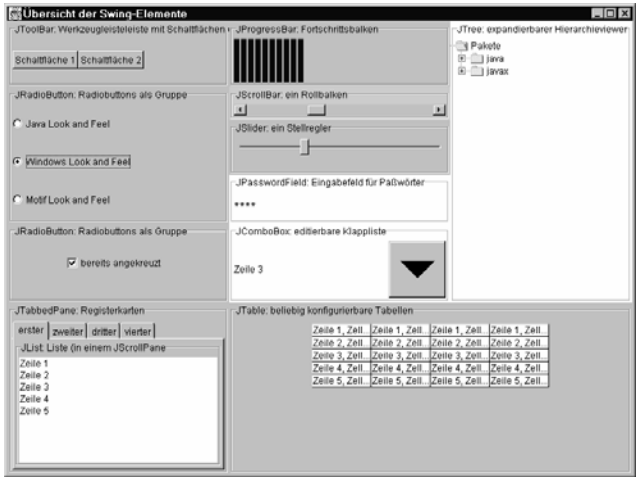
- Pluggable look-and-feel (plaf)
  - Swing erlaubt die Verwendung unterschiedlicher Aussehensweisen (look-and-feel, Paket `javax.swing.plaf`) von Benutzungsschnittstellen unabhängig von der Ausführungsplattform eines Programms.
  - Mit den Swing-Paketen werden verschiedene Standard-look-and-feel-Varianten ausgeliefert (z.B. Java, Windows, Motif).
  - Zur Steuerung des Aussehens dient die Klasse `UIManager`, über die man ermitteln kann, welche Aussehensweisen zur Verfügung stehen und welches look-and-feel von der aktuellen Systemplattform verwendet wird.

Swing-Komponenten

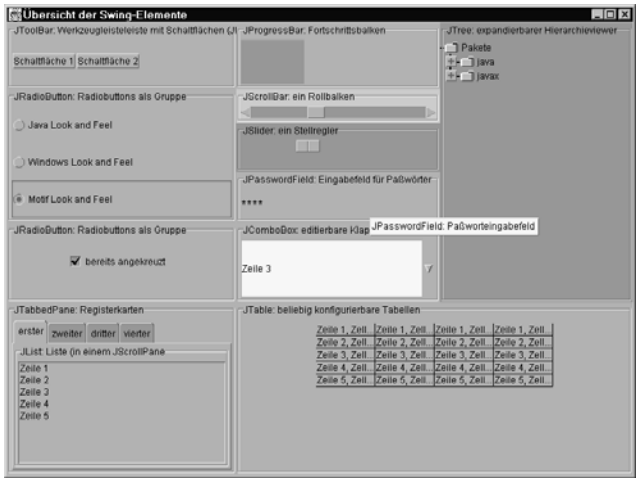
- Beispiel: JSwingBeispiel.java



Swing-Komponenten



Swing-Komponenten



Zusammenfassung: AWT ↔ Swing

- AWT
  - Konzipiert als sehr einfaches Benutzer-Interface für Applets. Wenig geeignet für allgemeine Benutzungsschnittstellen, wie z.B. in betrieblichen Informationssystemen.
- Swing
  - Seit JDK 1.1: Swing-Komponenten sind AWT-Container
  - Entspricht mehr dem modernen Software-Engineering
  - MVC (Model-View-Controller Architektur)
  - Plugable Look-and-Feel erlaubt angepasstes Aussehen der GUI
  - Aber wenig performant, da komplett in Java geschrieben (heute z.T. kein Problem mehr).

Eclipse

- Mehrere Jahre galt die IBM Entwicklungsplatt-form „Visual Age for Java“ als das Nonplusultra bei sehr vielen Entwicklern und Firmen.
- Diese professionelle Umgebung wartete mit Merkmalen auf, die bei den Mitbewerbern vergeblich gesucht wurden.
- Allerdings wurden diese besonderen Merkmale, wie bei den meisten kommerziellen Entwicklungs-umgebungen, durch Änderungen am Original-JDK von Sun erreicht.

Eclipse

- Die dynamische Entwicklung von Suns Original-JDK und etablierte Standards der OMG oder der Apache Foundation konnten dann aber von IBM‘s „Visual Age for Java“ nicht mehr zeitlich schnell genug umgesetzt werden.
- Dies führte bei IBM zur Einsicht, dass solch eine mächtige Technologie nicht mehr nur von einer Firma alleine getragen werden konnte, sondern einen Open-Source-Ansatz erforderten.
- Die bisherige Entwicklung (40 Mio $) wurde zu Open-Source erklärt und das Eclipse-Konsortium gegründet.

Eclipse

- Es gibt zahlreiche Plugins für eclipse:
  - DB-Anbindungen
  - GUI Programmierung (z.B. Visual Editor)
  - Browserwidgets
  - XML-Plugins
  - UML-Plugins
  - und viele mehr – teils kostenlos, aber auch kommerzielle

Eclipse

- Die Eclipse-Entwicklungsumgebung ist seit dem Jahr 2001 eine sehr gute, sehr mächtige und weit verbreitete Entwicklungsumgebung für die professionelle Erstellung von Java-Programmen.
- Aus diesem Grund wird sie auch neben dem JCreator in dieser Vorlesung behandelt.
- Eine in Entwicklerkreisen hochgelobte Eigenschaft von Eclipse ist das neue GUI-API namens SWT (Standard Widget Toolkit).

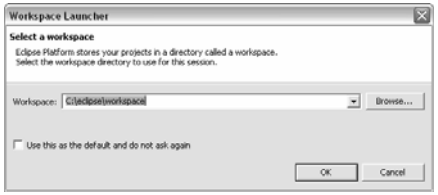
AWT ↔ Swing ↔ SWT

- SWT
  - Von Grund auf neu entwickeltes GUI-System
  - IDE zu allen üblichen Betriebssystemen
  - Sollte so eine Art „Visual Studio“ für Java werden
  - Mit SWT entwickelte Anwendungen unterscheiden sich im Look&Feel für den Benutzer nicht mehr von plattformnah entwickelten Programmen, die beispielsweise in C oder C++ geschrieben wurden. Und dies trotz eines weitgehend abstrakten Programmiermodells für alle Plattformen.
  - Vorteil: Effiziente Anbindung an das Betriebssystem
  - Nachteil: MVC-Konzept gibt es nicht mehr.

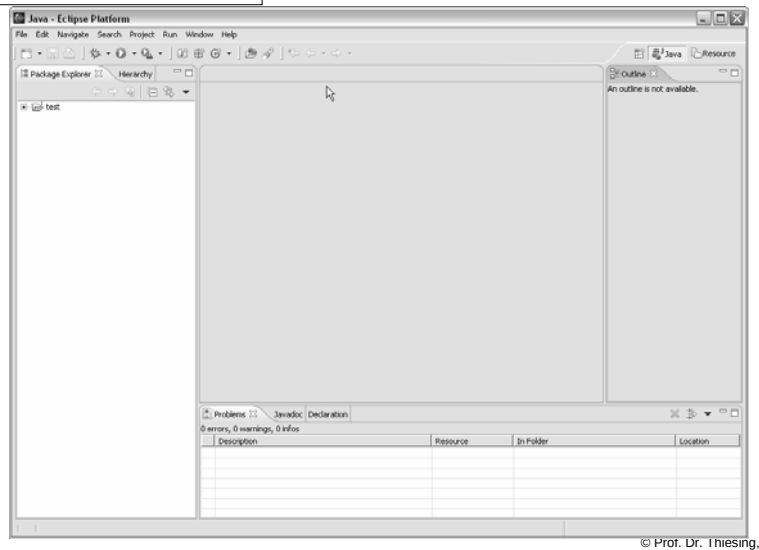
Eclipse / SWT

- Installation
  - Ein JDK 1.4/1.5 muss sich auf dem System befinden.
  - Unter www.eclipse.org/downloads die aktuelle Version herunterladen
    - Z.B. für Windows:
      - `eclipse-SDK-3.0.2-win32.zip`
      - `eclipse-examples-3.0.2-win32.zip`
  - und beide im gewünschten Zielverzeichnis entpacken.
  - Durch Doppelklick auf die Datei `eclipse.exe` und nach einem Moment Wartezeit öffnet sich die Anwendung, und es erscheint der folgende Startbildschirm:

Eclipse / SWT

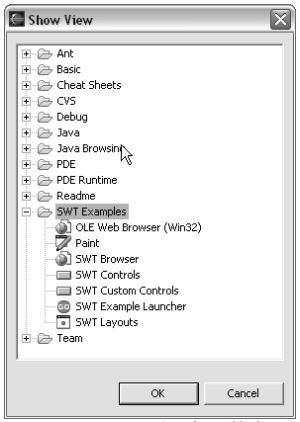


Eclipse / SWT



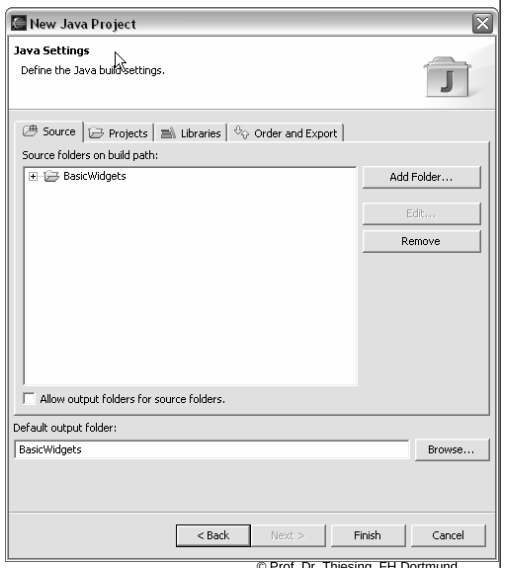
Eclipse / SWT

- Eine Reihe von Beispielen befinden sich unter
  - *Window > Show View > Other … > SWT Examples*



eclipse: Neues Project anlegen

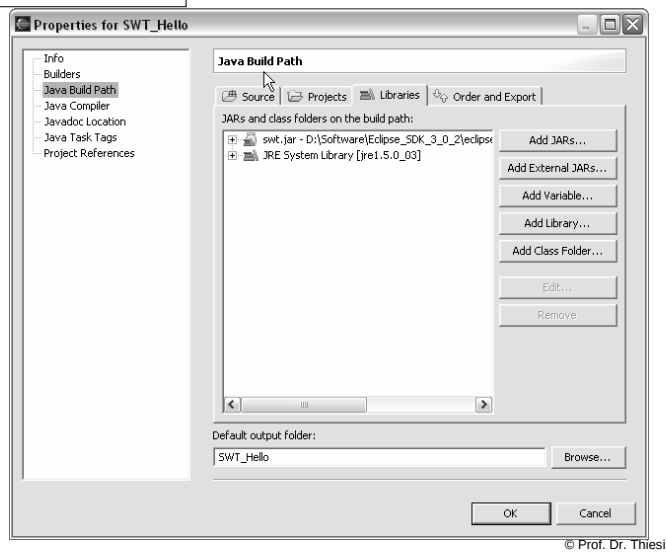
- File / New / Project…
- Java Project



SWT in eclipse benutzen

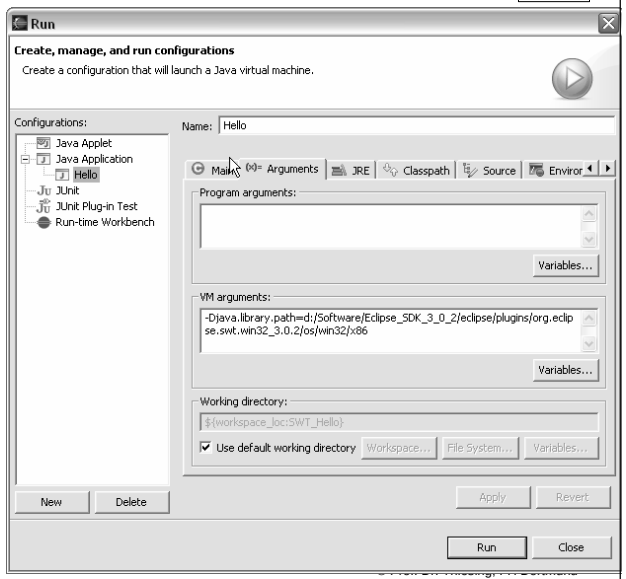
- (Projects / Properties / Java Build Path)
- Reiter „Libraries“
- Add External JARs
- Pfad angeben zu:
  org.eclipse.swt.win32_3.0.2\ws\win32\swt.jar
- Siehe folgenden Screenshot:

SWT in eclipse benutzen



SWT-Library einbinden

- Run
- Run..



Eclipse / SWT

- Da die Eclipse-Entwicklungsumgebung sehr mächtig ist und an dieser Stelle die SWT-GUI vorgestellt werden soll, kann nach der Installation von Eclipse auch die JCreator-Entwicklungs-umgebung einfach angepasst werden, um SWT-Programme übersetzen und starten zu können.
- Die SWT-Pakete für Java befinden sich in einem eigenen Paket: **`org.eclipse.swt`** sowie den Unterpaketen **`org.eclipse.swt.widgets`** und **`org.eclipse.swt.events`.** Diese müssen importiert werden.

SWT / JCreator

- SWT Klassen zum JCreator JDK Profil hinzufügen:
  - Menueintrag: `Configure->Options`
  - Eintrag: `JDK Profile->Edit`
  - Tab: `Classes->Add->Add Archive...`
  - Angabe der Archive-Datei swt.jar:
    - Wenn Eclipse installiert ist, so befindet sich die Datei unter:
      - \plugins\org.eclipse.swt.win32_3.0.2\ws\win32\swt.jar
    - Die entsprechende DLL unter:
      - \plugins\org.eclipse.swt.win32_3.0.2\os\win32\x86\swt-win32-3064.dll
        Diese muss ins system32 Verzeichnis von Windows kopiert werden.

GUI-Programmierung mit SWT

- Die Architektur von SWT
  - Ein typisches SWT-Programm ist nach dem folgenden Schema aufgebaut:
    - Display erzeugen
    - Shell mit Widgets erzeugen
    - Shell anzeigen
    - In einer Schleife Events bearbeiten und auf das Programmende warten.
    - Aufräumen
- Beispiel: Hello.java

GUI-Programmierung mit SWT

```
import org.eclipse.swt.*;
import org.eclipse.swt.events.*;
import org.eclipse.swt.widgets.*;

public class Hello {
   public static void main(String[] args) {

   Display display = new Display();


/* Das Display steht für den Bildschirm und die
Schnittstelle zum Betriebssystem. Hier werden
später beispielsweise die Events abgeholt.*/
```


GUI-Programmierung mit SWT

```
Shell shell = new Shell(display);
shell.setText("Das erste Fenster");


/* Eine Shell ist bei SWT das, was bei AWT der
Frame ist - ein normales Anwendungsfenster.
Dieses wird nun als Kind des Displays angelegt
und mit einem Fenstertitel versehen.
Die Shell-Klasse sollte allerdings NICHT durch
Vererbung erweitert werden, was möglich wäre,
da Shell nicht final ist.*/
```


GUI-Programmierung mit SWT

```
Button button = new Button(shell, SWT.PUSH);
button.setText("Klick mich!");

button.addSelectionListener(new
   SelectionAdapter() {
   public void widgetSelected(SelectionEvent e)
   { System.out.println("Danke"); }
});

/* Definition eines Push-Button mit einem
Labeltext und einer Aktion. Im Gegensatz zu
AWT/Swing heißt der Listener nun
SelectionListener, der Button funktioniert aber
ansonsten genauso.*/
```

GUI-Programmierung mit SWT

```
button.setBounds(0, 0, 120, 30);
/* absolute Position und Größe setzen */

shell.open();
/* anzeigen */
```


GUI-Programmierung mit SWT

```
while (!shell.isDisposed()) {
   if (!display.readAndDispatch()) {
        display.sleep();
   }
}

/* Solange die Shell noch nicht geschlossen wurde -
solange nur ein einziges Fenster vorhanden ist, ist das
ein einfacher Weg, zu entscheiden, wann das Programm
beendet werden soll - solange der Anwender also noch
nicht den Close-Button gedrückt hat, werden jetzt
Events, die beim Display ankommen, in das SWT-Rahmenwerk
geleitet. Das ist die Aufgabe von readAndDispatch(). Die
Methode liefert true, wenn weitere Events darauf warten,
verarbeitet zu werden und false, wenn momentan nichts
weiter zu tun ist. In diesem Fall wird das Programm
schlafen gelegt. Die Methode sleep() kehrt automatisch
zurück, wenn neue Events vom Betriebssystem angekommen
sind. */
```


GUI-Programmierung mit SWT

```
        display.dispose();
   }
}

/* Ganz wichtig bei jedem SWT-Programm ist es,
angeforderte Ressourcen wieder explizit
freizugeben. In diesem Beispiel reicht es, das
Display-Objekt wieder zu beseitigen. */
```


GUI-Programmierung mit SWT

GUI-Programmierung mit SWT

- Widgets
  - Zu den Widgets gehören die unmittelbaren Bedienelemente wie Tasten (`Button`), Textfelder (`Text`) oder Schieberegler (`Slider`), aber auch Elemente, die der Gruppierung von Bedienelementen dienen wie die Klassen `Group` und `Composite`.

GUI-Programmierung mit SWT

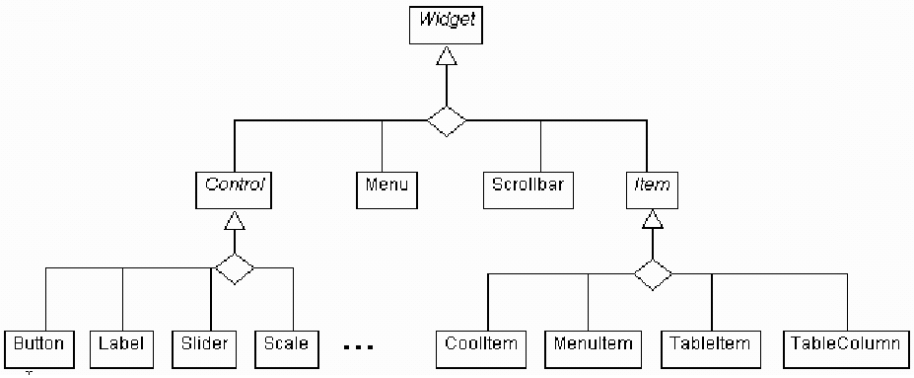
- Widgets
  - Vergleichbar mit der Klasse `Component` aus AWT
    - Gemeinsamkeiten: Ereignisse und Layouts
    - Zwar gibt es auch bei SWT Layout-Manager, doch standardmäßig (und im Unterschied zu AWT) hat eine Shell keinen.
    - Technischer Unterschied: In SWT wird die Betriebssystem-funktionalität genutzt und nicht, wie beim AWT, nur emuliert.
    - Benötigt wird zunächst ein Display-Objekt (Schnittstelle zum OS), um darauf alle Widgets aufzusetzen.

GUI-Programmierung mit SWT

- Widgets
  - Unterschied beim Aufruf: Bereits im Konstruktor eines Widget müssen einige Eigenschaften im Konstruktor übergeben werden, da diese an die entsprechende Betriebssystem-funktion weitergegeben werden. In AWT und Swing werden sämtliche Eigenschaften nach der Erzeugung einer Komponente gesetzt.
  - Allgemein gibt es weniger, aber durch Konstruktor-Parameter mächtigere Widget-Klassen als Swing (Bsp: `Button`).
  - Philosophie anders als bei Swing: Parent muss schon im Konstruktor bekannt sein.
  - Bei SWT gilt das Grundprinzip, dass ein Vater automatisch alle seine Kinder beseitigt ("Eltern haften für ihre Kinder" sozusagen). Das explizite Vorgeben der Vater-Kind-Beziehungen in den Konstruktoren hilft hier, den Aufwand klein zu halten.

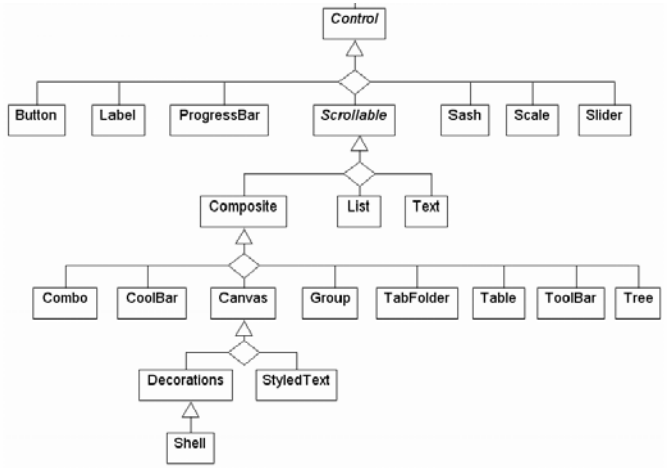
GUI-Programmierung mit SWT

- Klassenhierarchie Widgets



GUI-Programmierung mit SWT

- Klassenhierarchie Controls



GUI-Programmierung mit SWT

- Controls
  - Zu beachten ist, dass SWT eine gemeinsame Klasse für alle Arten von Buttons hat: Checkboxes oder Radiobuttons (Optionsschaltflächen) werden mit **`new Button(shell, <Style>)`** angelegt.
  - Als Style wird dann **`SWT.CHECK`** bzw. **`SWT.RADIO`** benutzt.

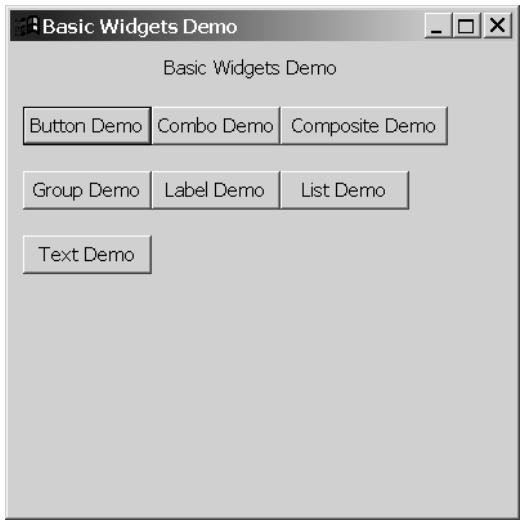
GUI-Programmierung mit SWT

- Event-Loop
  - Im Gegensatz zu AWT wird nicht implizit irgendwo in den Tiefen des Rahmenwerks ein geheimnisvoller UI-Thread gestartet, der dann das Programm am Leben hält, sondern dies muss bei SWT explizit geschehen. Z.B.:

```
while (!shell.isDisposed()) {
     if (!display.readAndDispatch()) {
          display.sleep();
     }
}
```


GUI-Programmierung mit SWT

- Beispiel
  - Widgets



  - BasicWidgetsDemo.java

GUI-Programmierung mit SWT

AWT (java.awt.*)	Swing (javax.swing.*)	SWT (org.eclipse.swt.*)
Button	JButton	Button
Label	JLabel	Label
List	JList	List
Menu	JMenu	Menu
MenuItem	JMenuItem	MenuItem
ScrollBar	JScrollBar	Slider
ScrollPane	JScrollPane	ScrolledComposite
TextField	JTextField	Text
Choice	JComboBox	Combo
	JPanel	Composite
	JSlider	Scale
	JTree	Tree
		ToolItem
		Tracker

GUI-Programmierung mit SWT

- Weitere Controls in SWT

ProgressBar	verschiedene Arten, nicht-selektierbar
Sash	verschiebbare Trennlinie zwischen zwei Widgets
Scale	selektiert Integer-Zahl aus einem bestimmten Bereich
ToolBar	Buttons mit Icons, die Programm-Funktionen ansteuern
CoolBar	Composite, das freies Anordnen mehrerer Toolbars erlaubt
TabFolder	Gruppen-Element zur Auswahl von Sub-Widgets
Table	Tabelle mit verschiedenen Ansichten
Text	ein- oder mehrzeiliges Feld zur Texteingabe
Shell	Fenster unter Kontrolle des Windowmanagers

GUI-Programmierung mit SWT

- Ereignisse

Komponente	Ereignis
Button (~Checkbox)	Selection
MenuItem	Selection
Scale	Selection
Slider	Selection
List	Selection
Tree	Selection, Tree
StyledText Text	Selection, Verify, Modify
Menu	Menu
Control	Control
Shell	Shell
Widget	Dispose

GUI-Programmierung mit SWT

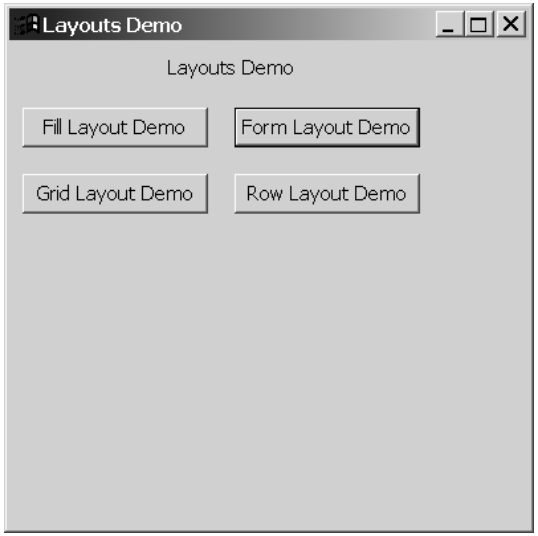
- Beispiele – jetzt in SWT
  - ButtonMitListener.java
  - ButtonMitListenerAnonym.java
  - MausLauschen.java
  - MausLauschenAnonym.java
  - MausLauschenMitAdapter.java
  - ControlTest.java
  - SteuerFenster.java
  - Multicast.java
  - MenueClipBoard.java

GUI-Programmierung mit SWT

- Layouts
  - FillLayout: einfach in einer Zeile/Spalte (maximal-gross und breit)
  - RowLayout: flexibler, kann Umbrechen, falls zuwenig Platz (Abstände auch einstellbar)
  - FormLayout: attached Widgets an Position, Brüder- oder eine Seite des Eltern-Widgets
  - GridLayout: sehr komplexe und mächtige Anordnung in einem Gitter (viele Einstellungen)
  - StackLayout: Stapel von Widgets (mit gleicher Größe und Position), nur oberstes sichtbar
  - CustomLayout

GUI-Programmierung mit SWT

- Beispiel
  - Layouts



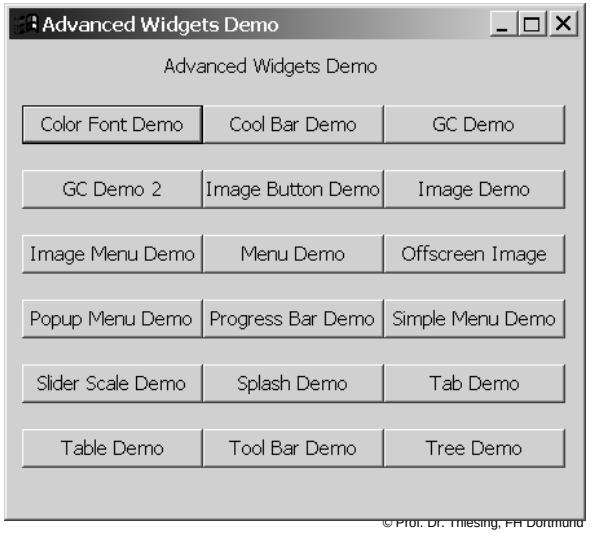
  - LayoutsDemo.java

GUI-Programmierung mit SWT

- Der Grafikkontext
  - Die Klasse `GC` enthält alle für Zeichenoperationen benötigten Methoden – Methoden wie `drawLine()`, `drawOval()`, `drawPolygon()`, `setFont()`, `getFontMetrics()`, u.v.a.m.
  - Gezeichnet werden kann auf Instanzen aller Klassen, die das Interface `Drawable` implementieren. Insbesondere sind das die Klassen `Image`, `Control` und ihre Unterklassen wie `Canvas` und `Display`.
  - Um jedes Mal darüber informiert zu werden, dass ein Ausschnitt neu gezeichnet werden muss, ist es erforderlich, sich bei dem Fenster oder dem Steuerelement als `PaintListener` zu registrieren.

GUI-Programmierung mit SWT

- Beispiel
  - Advanced Widgets



GUI-Programmierung mit SWT

- Fazit
  - SWT ist in vielen Bereichen schneller als Swing
  - Swing ist intuitiver/komfortabler
  - Eine Mischung von AWT/Swing-Komponenten mit SWT-Komponenten nicht empfohlen.
  - Die Konversion von Swing-Applikationen in SWT ist nicht immer einfach.

GUI-Programmierung mit SWT

- Tipps: Swing oder SWT
  - Möchte man einen Client schreiben,
    - dessen Look and Feel man selbst bestimmen kann, sollte man weiterhin bei **Swing** bleiben.
    - dessen Look and Feel zu 100% den Anwendungen entspricht, die speziell für die Plattform geschrieben wurden, sollte man **SWT** oder die Programmiersprache der Plattform wählen.
    - der sich bei Benutzerinteraktionen nicht so träge wie eine Swing-Anwendung verhält, sollte man **SWT** verwenden.
    - der Komponenten von Windows wie zum Beispiel den Internet Explorer beinhaltet, sollte man **SWT** wählen.
    - der innerhalb der Eclipse-Plattform als Plug-In realisiert wird, muss man **SWT** oder dessen Erweiterung JFace wählen.