

Praktikum aus Softwareentwicklung, Stunde 1

Lehrziele/Inhalt

1. Swing: JTree
2. File-API
3. Eclipse-Tastenkürzel

Graphische Oberflächen

Graphische Oberflächen (GUI) sind teil jeder Desktopanwendung. In Java kann man eine GUI mit AWT oder Swing erstellen. AWT ist die Grundlage der graphischen Oberflächen in Java, es bildet die Verbindung zwischen dem Betriebssystem und der Java VM. Wir werden uns in diesem Kapitel die Komponente zur Baumdarstellung in der Java Klassenbibliothek ansehen.

Baumdarstellung in Java

In vielen Programmen werden Modelle als Baum dargestellt. Die Java Klassenbibliothek bietet dafür die Klasse *JTree*. *JTree* ist eine graphische Komponente nach dem Model-View-Controller-Muster und benötigt *JTree* ein Modell. Solche Modelle müssen die Schnittstelle *TreeModel* implementieren. Da Business-Modelle unabhängig von der GUI-Technologie sein sollen, wird hier oft das Adapter-Muster eingesetzt.

JTree stellt für jeden Knoten ein Icon und die Ausgabe von *toString* dar. Das Icon zeigt an ob ein Knoten Kinder hat oder ein Blatt-Knoten ist. Durch setzen eines *TreeCellRenderers* über *setCellRenderer* kann man die Darstellung ändern. Will nur wenig an der Darstellung ändern, dann kann man die Klasse *DefaultTreeCellRenderer* beerben.

Beispiel: TreeTest

Theorie

Grundlegender Aufbau von Oberflächen in Java

In Java werden die GUI-Komponenten hierarchisch angeordnet, außen ein Fenster, im Fenster ein Menü und eine Komponente. Diese Komponente kann ein Container sein und damit weitere Komponenten enthalten.

Ein frei wählbarer Layout-Manager (*p.setLayout(LayoutManager)*) legt fest wie die Komponenten in einem Container angeordnet werden. Komponenten werden immer relativ zu ihrem Container angeordnet. Die X-Koordinate ist die Entfernung vom linken Rand des Containers zum linken Rand der Komponente. Die Y-Koordinate ist die Entfernung vom oberen Rand des Containers zum oberen Rand der Komponente.

Abstract Window Toolkit

Die Klassen des Abstract Window Toolkit (AWT) liegen im Paket *java.awt*. Das AWT ist die Grundlage der Graphischen Oberflächen in Java, es abstrahiert die GUI-Komponenten des Betriebssystems. Für jede unterstützte Betriebssystemkomponente in AWT gibt es eine Peer-Klasse, die die Verbindung zu Java herstellt. Java-Programme mit AWT Oberflächen sehen wie nativ entwickelte Programme aus.

Die Nachteile von AWT sind: es werden Betriebssystem-Ressourcen verwendet, und es werden nur Komponenten unterstützt die auf allen Plattformen verfügbar sind. Denn eine Java Anwendung muss unverändert auf allen Plattformen laufen für die eine Java-Laufzeit-Umgebung existiert.

Die Basisklasse aller AWT-Komponenten ist *java.awt.Component*.

Swing

Die Klassen von Swing liegen im Paket *javax.swing*, Klassennamen von Graphischen Komponenten in Swing beginnen mit einem „J“, wie zum Beispiel: *JTree*, *JList*, *JButton*. Swing Komponenten zeichnen in Java, nur Basiskomponenten wie Fenster nutzen AWT und damit Betriebssystem-Ressourcen. Komponenten wie Buttons oder Listen werden ausschließlich in Java gezeichnet, daher bezeichnen wir Swing als leichtgewichtig. Die Vorteile dieses Vorgehens sind: es können beliebige Komponenten gestaltet werden, das Aussehen der Komponenten kann beliebig gestaltet werden (Pluggable Look-and-Feel). Der Nachteil, Java Programme sind optisch von nativen Programmen unterscheidbar, auch wenn versucht wird das native Aussehen so gut wie möglich nachzubilden.

Swing-AWT-Integration

Die Basisklasse aller Swing-Komponenten ist *javax.swing.JComponent*. Diese leitet von *java.awt.Container* ab, damit können Swing-Komponenten in AWT-Oberflächen eingebettet werden. Das soll aber nur bei vorhandenen AWT-Anwendungen genutzt werden, bei Neuentwicklungen sollte man die Oberfläche ganz in Swing gestalten.

Wichtige Swing-Komponenten

Die folgende Aufzählung gibt eine thematische Übersicht über die graphischen Komponenten in Swing:

- Buttons:
 - JButton
 - JRadioButton
 - JCheckBox
- Drop-Down
 - JComboBox
- Fenster
 - JFrame: mit Rahmen
 - JWindow: ohne Rahmen
 - JDialog: Dialogfenster, modal und nicht modal
 - JFileChooser: Standarddialog, Datei öffnen, Datei speichern
- Layout
 - JPanel: Container für weitere UI-Komponenten
 - JScrollPane, JScrollBar
 - Siehe Interface: layoutManager
- Listen
 - JComboBox: Dropdown-Liste
 - JList: Liste
 - JSpinner: selektieren eines Elements, „flache Dropdown-Liste“
 - JTable: Tabelle
 - JTree: Baum

- Menü:
 - JMenuBar, JMenu, JMenuItem
 - JToolBar
 - JPopupMenu: Kontextmenü
 - JSeparator: Trennstrich zwischen Menüeinträgen
- Statusanzeige
 - JProgressBar
- Text
 - JLabel: kurze Beschreibungstexte
 - JTextField: einzeilig, Text-Eingabe/-Ausgabe
 - JTextArea: mehrzeilig, Text-Eingabe/-Ausgabe

Ereignisse

Das Java Ereignismodell baut auf das Beobachter(Observer)-Muster auf, die Interessierten Objekte werden Listener genannt.

Dateien

Über die Klasse *File* kann in Java auf das Dateisystem zugegriffen werden. *File* ist eine reine Verwaltungsklasse, ein Objekt enthält nur Informationen über eine Datei, aber keinen Inhalt. Mit *File* kann man beispielsweise feststellen ob eine Datei existiert, lesbar und schreibbar ist. Die Klasse *File* enthält die statischen Hilfsmethoden: *listRoots*, sie liefert die Laufwerke des Computers und *createTempFile*, mit ihr kann man temporäre Dateien anlegen.

Beispiel: FileTest

Eclipse-Tastenkürzel

1. Ctrl+1: Kontextabhängige Vorschläge, zB: Code-Erzeugung und Korrekturen
2. Ctrl+Space: Code-Vervollständigung
3. Alt+Shift+M: Extrahieren einer Methode