ulm university universität ulm

GUI Programmierung

Tipps & Tricks

Überblick

1. Motivation
2. Grundlagen
   - Java Coding Conventions
   - Javadoc Comments
   - GroupLayout unter NetBeans
3. OpenOffice & HSQL
4. Tipps & Tricks zu GUIs
   - Java Events
   - Java Observer
   - Eingabeüberprüfung auf GUI-Ebene
   - JTree Model
5. Arbeiten mit Versionskontrolle
6. Zusammenfassung

GUI Programmierung

1 Motivation

2 Grundlagen
- Java Coding Conventions
- Javadoc Comments
- GroupLayout unter NetBeans

3 OpenOffice & HSQL

4 Tipps & Tricks zu GUIs
- Java Events
- Java Observer
- Eingabeüberprüfung auf GUI-Ebene
- JTree Model

5 Arbeiten mit Versionskontrolle

6 Zusammenfassung

Motivation

- Aufzeigen von Möglichkeiten
- Tipps & Tricks für effizienteres Arbeiten
- Mögliche Fehlerquellen vermeiden
- Hilfreiche Praxisbeispiele

GUI Programmierung

1 Motivation

2 Grundlagen
- Java Coding Conventions
- Javadoc Comments
- GroupLayout unter NetBeans

3 OpenOffice & HSQL

4 Tipps & Tricks zu GUIs
- Java Events
- Java Observer
- Eingabeüberprüfung auf GUI-Ebene
- JTree Model

5 Arbeiten mit Versionskontrolle

6 Zusammenfassung

Warum Coding Conventions?

- Bessere Wartbarkeit des Codes
- 8 Leute im Team
- Code verständlicher für die anderen
- Siehe auch:

  `http://java.sun.com/docs/codeconv/`

Wo steht Was?

1. Klassendeklaration
2. Variablen
   - Zuerst die statischen Variablen
   - Reihenfolge jeweils `public`, `protected`, `private`
3. Konstruktoren
4. Methoden

Was nenne ich Wie?

- Klassen → Substantive (beginnen mit Großbuchstabe)

```
public class Beispiel { ..
```

- Methoden → Verben (beginnen mit Kleinbuchstabe)

```
private void machWas() { ..
```

- Variablen
  - Möglichst keine Ein-Buchstaben-Variablen, falls doch:
  - i,j,k,m,n für `int`
  - c,d,e für `char`
  - Konstanten in Großbuchstaben, Wörter getrennt durch '_'

Leerzeichen und Leerzeilen

- Leerzeilen
  - Zwischen Methoden
  - Zwischen Variablen und Statements
  - Vor Kommentaren
- Leerzeichen
  - Beispiel:

```
a + b = c;
String s = "Irgendwas " + a + " noch etwas";
```

  - Aber:

```
a++;
```

  - Vor Casts

```
a = (int) b;
```

Kommentare allgemein

- Am Anfang jeder Datei
- Vor Klassendeklarationen
- Vor Methoden
- Evtl. vor Variablen

Verschiedene Arten von Kommentaren

```
/* single-line comment */

a + b = c;                  // end-of-line comment

/*
 * block comment
 */

/**
 * javadoc comment
 */
```

Aufbau

- Tags

  @author Name des Autors
  @param Name des Parameters Beschreibung
  @return Beschreibung des Rückgabewerts

- Formatierungen
  - Einfache HTML-Elemente können zur Formatiierung benutzt werden (`<p>,<ul>,<li>,...`)
  - `<code></code>`-Style für z.B. Package- oder Klassennamen
- Erster Satz dient als Zusammenfassung

GroupLayout

- Neues Layoutprinzip
- Erstmals verwendet in Netbeans (Matisse GUI-Builder)
- In Java SE 6 enthalten
- In der Zwischenzeit auch von anderen Gui-Buildern unterstützt
  - Jigloo `http://www.cloudgarden.com/jigloo/`
  - Windowbuilder Pro `http://windowbuilderpro.com/`
- Tutorial: `http://www.netbeans.org/kb/55/quickstart-gui.html`

GUI Programmierung

1 Motivation

2 Grundlagen
- Java Coding Conventions
- Javadoc Comments
- GroupLayout unter NetBeans

3 OpenOffice & HSQL

4 Tipps & Tricks zu GUIs
- Java Events
- Java Observer
- Eingabeüberprüfung auf GUI-Ebene
- JTree Model

5 Arbeiten mit Versionskontrolle

6 Zusammenfassung

Kommunikation mit OpenOffice Base über HSQL

LIVEDEMO

GUI Programmierung

1 Motivation

2 Grundlagen
- Java Coding Conventions
- Javadoc Comments
- GroupLayout unter NetBeans

3 OpenOffice & HSQL

4 Tipps & Tricks zu GUIs
- Java Events
- Java Observer
- Eingabeüberprüfung auf GUI-Ebene
- JTree Model

5 Arbeiten mit Versionskontrolle

6 Zusammenfassung

Java Event Model

- 2 verschiedene Eventkategorien
  - low-level Events (Tastatur-/Mauseingabe)
  - semantische Events (z.B. `actionPerformed`)
- Semantische Events müssen nicht zwangsläufig durch low-level Events ausgelöst werden (z.B. `TableModelEvent`)
- Wenn möglich semantische Events verwenden
  - sicherer
  - komfortabler

Java Event Handling

- 3 Arten von Threads
  - Initial threads (Initialisierung/Systemstart)
  - Event-Dispatch thread (Benutzerinteraktion)
  - Worker threads (hintergründige Prozessabarbeitung)
- Event-Dispatch thread arbeitet alle Events sequentiell ab
  - Verarbeitet auch `paint`-Routinen
  - EventListener-Code sollte schnell ablaufen
- Aufwendige Berechnungen in Worker threads auslagern
  - `javax.swing.SwingUtilities`
    `invokeLater(Runnable task)`
  - `javax.swing.SwingWorker` (ab JDK 1.6)
    `doInBackground()`, `run()`

Adapterklassen

- Spart überflüssige Schreibarbeit
- Bläht den Code nicht unnötig auf
- Übersicht der wichtigsten EventListener & Adapterklassen:

`http://java.sun.com/docs/books/tutorial/uiswing/events/api.html`

Adapterklassen - Beispiel

```
public class MyClass implements MouseListener {
   ...
   someObject.addMouseListener(this);
   ...
   /* Empty method definitions. */
   public void mousePressed(MouseEvent e) {
   }

   public void mouseReleased(MouseEvent e) {
   }

   public void mouseEntered(MouseEvent e) {
   }

   public void mouseExited(MouseEvent e) {
   }

   public void mouseClicked(MouseEvent e) {
      //Event listener implementation...
   }
}
```

```
public class MyClass extends MouseAdapter {
   ...
   someObject.addMouseListener(this);
   ...

   public void mouseClicked(MouseEvent e) {
   ...//Event listener implementation
   }
}
```

Java Observer

LIVEDEMO

Eingabeüberprüfung auf GUI-Ebene

- Eingabeüberprüfung sollte wenn möglich in der GUI stattfinden
  - Erspart komplizierte Abfrageroutienen
  - Unzulässige Eingabemöglichkeiten sind direkt sichtbar
- 2 Heransgehenweisen möglich
  1. Überprüfung im Nachhinein
     - `InputVerifier`
       Nur zulässige Eingaben erlauben den Focus weiterzugeben
  2. Überprüfung während der Eingabe
     - `JFormattedTextField`
       Mittels `MaskFormatter` wird eine eindeutige Eingabemaske vorgegeben
     - `DocumentFilter`
       Nur zulässige Eingaben werden übernommen

JTree Model

LIVEDEMO

GUI Programmierung

1 Motivation

2 Grundlagen
- Java Coding Conventions
- Javadoc Comments
- GroupLayout unter NetBeans

3 OpenOffice & HSQL

4 Tipps & Tricks zu GUIs
- Java Events
- Java Observer
- Eingabeüberprüfung auf GUI-Ebene
- JTree Model

5 Arbeiten mit Versionskontrolle

6 Zusammenfassung

CVS & SVN

- CVS
  - WinCVS
  - in Eclipse und Netbeans bereits integriert
- SVN
  - TortoiseSVN (Shell-Integrierung von SVN-Kommandos)
    `http://tortoisesvn.tigris.org/`
  - SubClipse (SVN Plug-in für Eclipse)
    `http://subclipse.tigris.org/`
  - Subversion for NetBeans
    `http://subversion.netbeans.org/`

Arbeiten mit SVN

- Genereller Arbeitszyklus
  1. Update
  2. evtl. Konflikte bereinigen
  3. Bearbeiten
  4. Commit (immer mit sinnvoller LogMessage)
- Repository sollte folgende Unterverzeichnisse enthalten

trunk	Für alle Dateien im Hauptentwicklungszweig
branches	Für abzweigende Entwicklungen

- Abzweigungen können mittels Branch erstellt werden
- Merge führt verschiedene Zweige wieder zusammen

GUI Programmierung

1 Motivation

2 Grundlagen
- Java Coding Conventions
- Javadoc Comments
- GroupLayout unter NetBeans

3 OpenOffice & HSQL

4 Tipps & Tricks zu GUIs
- Java Events
- Java Observer
- Eingabeüberprüfung auf GUI-Ebene
- JTree Model

5 Arbeiten mit Versionskontrolle

6 Zusammenfassung

Zusammenfassung

- Beispielcode online verfügbar
- Für Fragen stehen wir auch über Weihnachten unter `progwerkstatt.informatik@uni-ulm.de` zur Verfügung

Heute ist nicht alle Tage,
wir kommen wieder keine Frage.