Einführung in die Programmierung für NF

Übung 10

08.01.2014

Inhalt

- Programmierung Blatt 9

- Eventbehandlung in Java Swing
- InputDialoge in Java
- UML

Programmierung Blatt 9

- Musterlösung zum Download auf der Homepage
- Kann als Vorlage für Blatt 10 dienen

Eventbehandlung in Java Swing

- Wir haben bereits einige grafische Komponenten von Java Swing kennengelernt
- Aber: Buttons u.ä. nützen wenig, wenn nichts passiert, wenn man auf sie klickt

- Zur Event-Behandlung wie Mausklicks, Mausbewegungen oder Tastaturanschläge, werden Events von EventListenern bearbeitet

Events in Java Swing

- MouseEvent
  - Wird durch Mausklicks und Mausbewegungen ausgelöst, z.B.
    - Maustaste gedrückt
    - Maustaste losgelassen
    - Maustaste geklickt
    - Maus wird bewegt
    - Maus wird mit gedrückter Taste bewegt

Events in Java Swing

- MouseEvent
  - Folgende Informationen sind in einem MouseEvent abfragbar:
    - **getClickCount():** Liefert die Anzahl der Mausklicks
    - **getPoint():** Liefert die Position der Maus
    - **getX():** Liefert die x-Position der Maus zum Eventzeitpunkt
    - **getY():** Liefert die y-Position der Maus zum Eventzeitpunkt

Events in Java Swing

- KeyEvent
  - Wird durch Drücken, Loslassen oder Klicken einer Taste auf der Tastatur ausgelöst
  - Liefert folgende Informationen:
    - **getKeyChar()** liefert das entsprechende Tastenzeichen
    - **getKeyCode()** liefert den Tastencode

EventListener in Java Swing

- Zur Behandlung dieser Events werden entsprechende EventListener benötigt
- Mit diesen kann, wenn ein Event auftritt, eine entsprechende Reaktion durchgeführt werden
- Generell kann jedem Objekt der GUI ein EventListener hinzugefügt werden
  (also nicht nur Buttons o.ä.)

EventListener in Java Swing

- Die wichtigsten Arten von EventListenern:
  - **ActionListener**
    für bestehende Actions, z.B. Klick auf einen Button
  - **MouseListener**
    beachtet verschiedene Zustände der Maus
  - **MouseMotionListener**
    beachtet verschiedene Bewegungen der Maus
  - **KeyListener**
    beachtet verschiedene Zustände der Tastatur

ActionListener

- Manche Elemente (z.B. Buttons) können Aktionen (`Action`) auslösen
- `ActionListener` ist eine Schnittstelle mit der Methode `actionPerformed()`
- Ein `ActionListener` wird mit der Methode `addActionListener()` an die Objekte angeheftet, die Aktionen auslösen können

MouseListener

- Der `MouseListener` enthält die folgenden Methoden, in denen jeweils ein `MouseEvent` als Parameter übergeben wird:
  - mouseClicked(MouseEvent e) Maus geklickt
  - mouseEntered(MouseEvent e) Maus betritt etwas
  - mouseExited(MouseEvent e) Maus verlässt etwas
  - mousePressed(MouseEvent e) Maustaste gedrückt
  - mouseReleased(MouseEvent e) Maustaste losgelassen

MouseMotionListener

- Der `MouseMotionListener` enthält die folgenden Methoden, denen jeweils ein `MouseEvent` als Parameter übergeben wird:
  - mouseMoved(MouseEvent e) Maus bewegt
  - mouseDragged(MouseEvent e) Maus bei gedrückter Maustaste bewegt

- Hinzufügen mit `addMouseListener()` bzw. `addMouseMotionListener()`

KeyListener

- Der `KeyListener` enthält die folgenden Methoden, denen jeweils ein `KeyEvent` als Parameter übergeben wird:
  - keyPressed(KeyEvent e): Taste gedrückt
  - keyReleased(KeyEvent e): Taste losgelassen
  - keyTyped(KeyEvent e): Taste gedrückt und wieder losgelassen
- Hinzufügen mit `addKeyListener()`

Implementierung der EventListener

- Generell gibt es drei verschiedene Möglichkeiten, EventListener Objekten hinzuzufügen
  1. Die View selbst implementiert das entsprechende Interface und wird dadurch selbst zum Listener samt seiner Funktionen
  2. Jedes Objekt bekommt einen neuen Listener
  3. Eine andere Klasse (z.B. der Controller) implementiert die Interfaces

Implementierung der EventListener

- In kleinen Projekten macht es Sinn, alle Events vom gleichen EventListener behandeln zu lassen – wenn dies der Controller übernimmt, wird die View deutlich übersichtlicher
- Wenn jedes Objekt seinen eigenen „neuen“ EventListener bekommt, ist es leichter zu sehen, welches Objekt über welche Listener verfügt und was bei der Behandlung passiert

Aufgabe

- Hinzufügen der Listener an eine kleine GUI
  - ActionListener an einen Button
  - MouseListener und KeyListener an ein beliebiges Objekt
- Ausgabe von Text oder Mausposition auf die Konsole bei Eventbehandlung
- Verwendung der View als Listener und neue eigene Listener

Dialoge in Java

- Um einmalige kurze Eingaben oder Meldungen anzuzeigen, gibt es in Java Dialoge
  - MessageDialog
  - ConfirmationDialog
  - InputDialog
  - OptionDialog

- Es können auch eigene Dialoge erstellt werden

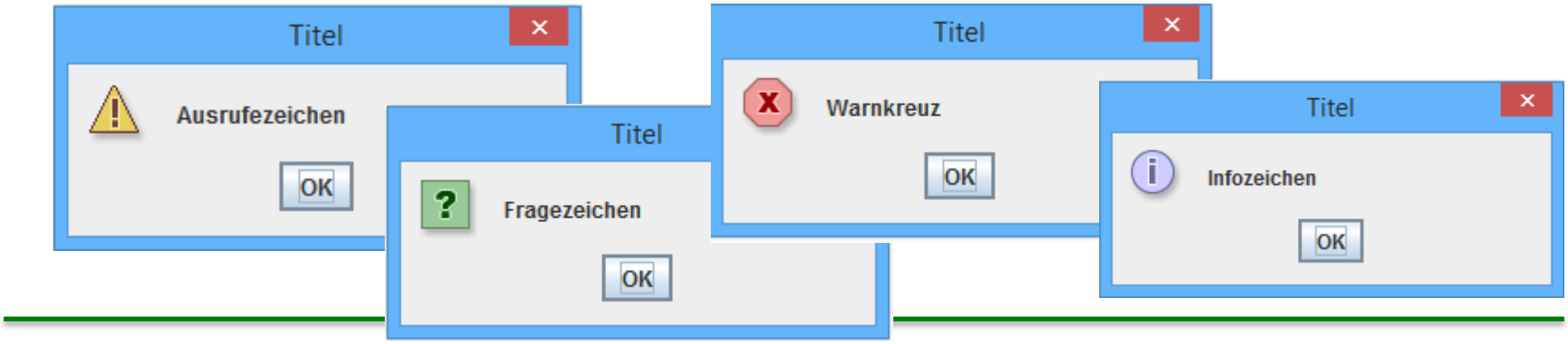
Dialoge in Java

- Dialoge bestehen aus wenigen Elementen:
  - Titel
  - Icon (optional)
  - Inhalt (z.B. Text oder Inputfeld)
  - Buttons

- Eigene Dialoge können natürlich aus mehr Elementen bestehen, z.B. zwei Inputfelder

Dialoge in Java

- Vorhandene Icons:
  - **ohne Icon** (JOptionPane.PLAIN_MESSAGE)
  - mit **Ausrufezeichen** (JOptionPane.CANCEL_OPTION)
  - mit **Fragezeichen** (JOptionPane.QUESTION_MESSAGE)
  - mit **Warnkreuz** (JOptionPane.ERROR_MESSAGE)
  - mit **Infozeichen** (JOptionPane.INFORMATION_MESSAGE)



Dialoge in Java

- Erstellung eines MessageDialogs

```
JOptionPane.showMessageDialog(null,
"Irgendein wichtiger Hinweis",
"Titel", JOptionPane.INFORMATION_MESSAGE);
```

3. Titel

2. Text im Dialog

4. Icon

1. ParentElement
kann hier null bleiben

Dialoge in Java

- Erstellung eines InputDialogs

```
String s = JOptionPane.showInputDialog(null,
"Enter some text:", "Titel",
JOptionPane.QUESTION_MESSAGE);
```

- Der eingegebene String wird so automatisch in die Variable „s“ gespeichert und kann weiter behandelt werden

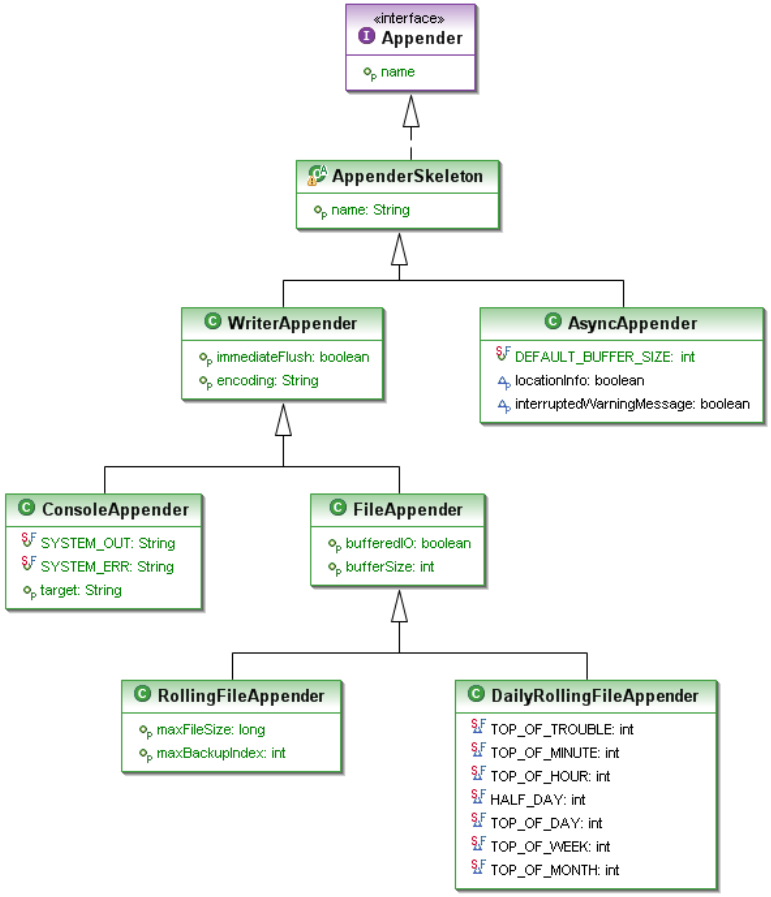
Dialoge in Java

- Andere Dialoge können auch mehrere Buttons oder Felder haben und sind in der Auswertung komplexer

- Während ein Dialog geöffnet ist, „pausiert“ das Programm solange, bis dieser geschlossen wird

UML

- Fragen zu UML?



Fragen zum Übungsblatt?