Bereich: Grafische Benutzeroberflächen (UI), Events (2)

Editor

Package: `de.dhbwka.java.exercise.ui.editor`	**Klasse:** `Editor`

Aufgabenstellung:

Schreiben Sie einen kleinen Texteditor mit „Swing“ (aufbauend auf der Klasse `EditorSimple`)! Keine Panik, das geht relativ einfach. Hier ein grobes Gerüst dafür:

```
package de.dhbwka.java.exercise.ui.editor;

import javax.swing.JFrame;

@SuppressWarnings("serial")
public class Editor extends JFrame {
   /**
    * Constructor for the editor, UI elements and listeners
    * may be created/assigned here
    */
   public Editor() {
      super( "Editor" );

      // (...)

      // Set size and show
      this.setSize( 640, 480 );
      this.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
      this.setVisible( true );
   }
}
```

Wir halten an dieser Stelle kurz fest, dass unser `Editor` bereits die Klasse `JFrame` erweitert, also bereits ein Fenster ist, zu dem weitere GUI-Komponenten hinzugefügt werden können (`this.add(Component)`).

Im Wesentlichen soll die Oberfläche aus zwei Teilen bestehen:

- einer Menüleiste mit Pull-Down-Menüs
- einer Editor-Oberfläche inkl. Scroll-Balken

Die Menüleiste müssen wir uns selbst basteln, die Editor-Oberfläche bringt Swing weitgehend fertig mit.

Eine neue Menüleiste erzeugen wir mit `JMenuBar`.
Jedes (Haupt-)Menü („Datei“, „Bearbeiten“) ist ein `JMenu`.
Jeder einzelne Menüpunkt ein `JMenuItem`.

Die Menüauswahl soll etwa so wie auf den folgenden Seiten aussehen.

Bitte beachten Sie den Menüpunkt „Senden an“, bei dem es sich um ein Untermenü handelt. „Senden an“ ist also kein `JMenuItem`, sondern ein `JMenu` und enthält seinerseits wieder `JMenuItem`s.

Die Editor-Oberfläche bringt Swing weitgehend in Form der Klasse `JTextPane` mit. In dieser kann man mit den Methoden `setText()` bzw. `getText()` den Inhalt setzen bzw. auslesen.

Bettet man diese `JTextPane` in eine `JScrollPane` ein, erhält man automatisch einen passenden Scrollbalken.

Vorgehen:

- Instanz von `JTextPane` erstellen
- Instanz von `JScrollPane` mit dieser `JTextPane` als Inhalt erstellen (über den Konstruktor)
- `JScrollPane` zum Fenster hinzufügen.

Hinweis: Das JScrollPane ist ein Container und enthält nun das JTextPane. Es reicht also, das JScrollPane hinzuzufügen; das darin enthaltener JTextPane wird damit ebenfalls hinzugefügt.

Zum Beispiel so:

```
JTextPane editPane = new JTextPane();
JScrollPane scrollPane = new JScrollPane(editPane);
scrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
frameInstance.add(scrollPane);
```

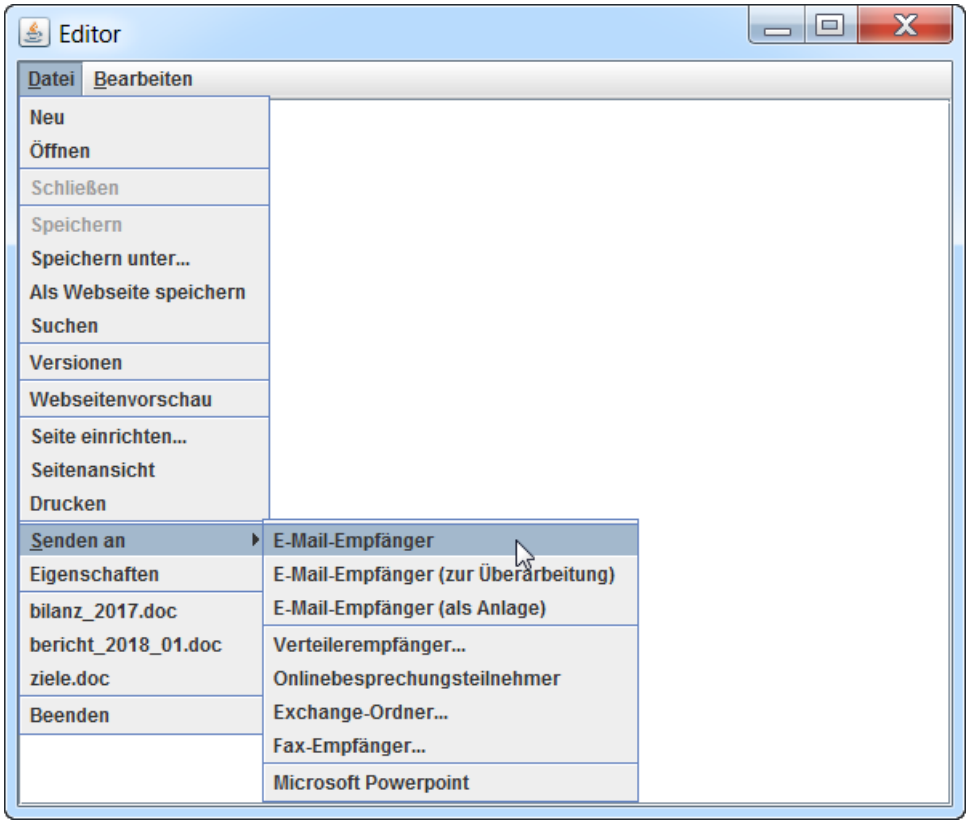
Teilaufgaben (gehen Sie am besten in dieser Reihenfolge vor!):

1) Legen Sie das Grundgerüst (`JFrame`, s.o.) an und wählen Sie einen Layoutmanager.
2) Erzeugen Sie ein Menü, das den obigen Angaben (zumindest in etwa) entspricht.
   Hinweis:
   `Component`- *bzw.* `Button`-*Objekte (hier:* `comp`*) haben u.a. folgende Methoden:*

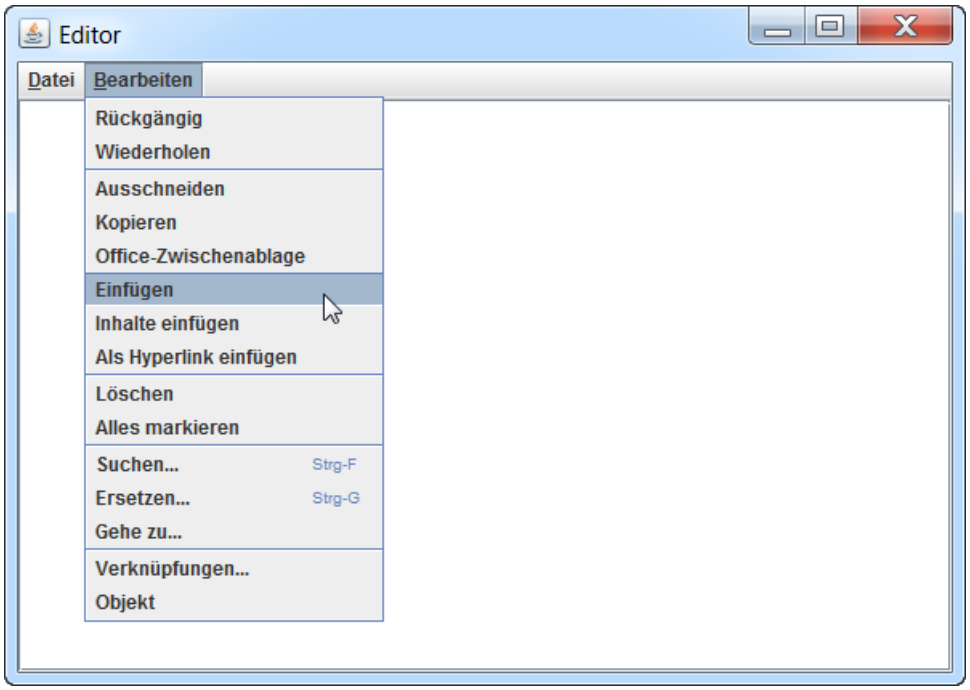
   ```
   comp.setEnabled(boolean); // Aktiviert oder nicht?
   comp.addActionListener(ActionListener);  // Instanz v. ActionListener
   comp.setMnemonic(int);    // Tastenkürzel (char), siehe JavaDoc
   ```

   Evtl. rentiert es sich, eine Methode zu schreiben, über die man in einer Zeile ein neues `JMenuItem` *erzeugen und gleichzeitig mehrere dieser Werte setzen kann.*
3) Erzeugen Sie ein `JTextPane` mit Scrollbalken wie oben beschrieben
4) Füllen Sie die Methode `actionPerformed()` mit Leben und implementieren Sie folgende Funktionen:
   a) „Neu…“ leert das `JTextPane` (leere Anzeige).
   b) „Öffnen…“ zeigt eine Dateiauswahlbox an und liest den Inhalt der gewählten Textdatei in den Editor ein.
   c) Der Menüpunkt „Speichern“ ist erst aktiviert, wenn eine Datei eingelesen wurde. Wird er ausgewählt, wird der aktuelle Inhalt des Editors in diese Datei zurück geschrieben.
   d) „Beenden“ beendet das Programm nach einer Abfrage, ob das Programm tatsächlich beendet werden soll
5) Freiwillige Erweiterungen:
   a) Wer will darf sich auch an der Funktion „Suchen und Ersetzen“ versuchen und Tastenkürzel ermöglichen.
   b) Erweitern um Copy & Paste per Bearbeiten-Menü (Hinweise am Ende beachten für das Arbeiten mit der Zwischenablage)
   c) Erweitern um ein Popup-Menü für Copy & Paste

Hinweis: Erfinden Sie das Rad nicht immer neu! Verwenden Sie zum Lesen und Schreiben der Dateien Code, den Sie bereits in anderen Übungsaufgaben entwickelt haben, wieder. Selbiges gilt für die Dateiauswahlbox und die Abfrage zum Beenden des Programms.



Screenshot Menü „Datei“



Screenshot Menü „Bearbeiten“

Arbeiten mit der Zwischenablage:
Folgende Code-Snippets können Sie verwenden, um Inhalte in der Zwischenablage programmatisch zu speichern respektive auszulesen:

```
import java.awt.Toolkit;
import java.awt.datatransfer.Clipboard;
import java.awt.datatransfer.DataFlavor;
import java.awt.datatransfer.StringSelection;

// (...)

/**
 * Copy content to clipboard
 *
 * @param content content to copy
 */
public void copySelectionToClipboard() {
   String content = this.editPane.getSelectedText();
   if (content != null) {
      StringSelection selection = new StringSelection(content);
      Clipboard clipboard = Toolkit.getDefaultToolkit().getSystemClipboard();
      clipboard.setContents(selection, selection);
   }
}

/**
 * Paste clipboard content, just inserts at current caret position and ignores
 * selection (text editors usually replace selected text with clipboard content)
 */
public void doPaste() {
   try {
      String data = (String) Toolkit.getDefaultToolkit().getSystemClipboard()
                                                   .getData(DataFlavor.stringFlavor);
      if (data != null) {
         this.editPane.getDocument()
                      .insertString(this.editPane.getCaretPosition(), data, null);
      }
   } catch (Exception e) {
   }
}
```

Bereich: Grafische Benutzeroberflächen (UI), Events (2)

Hütchenspiel

Package: `de.dhbwka.java.exercise.ui.event`	**Klasse:** `ShellGame`

Aufgabenstellung:

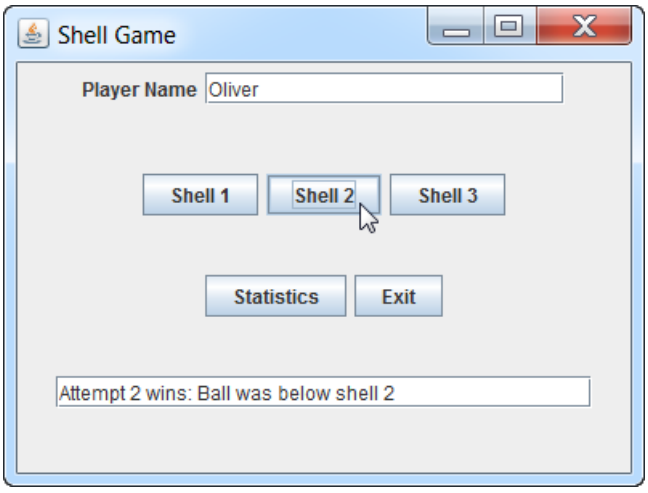
Schreiben Sie eine Java-Applikation, die das bekannte Hütchenspiel („shell game“) implementiert!

Beim Hütchenspiel gibt es drei Hütchen. Unter einem der Hütchen liegt eine Kugel.
Der Spieler muss erraten, unter welchem Hütchen die Kugel liegt.

Die drei Hütchen sollen in der Java-Applikation durch drei Buttons realisiert werden. Bei jedem Spielzug wird zufällig die Kugel unter ein Hütchen gelegt.

Wurde ein Hütchen-Button angeklickt, so bekommt der Spieler in einem Textfeld angezeigt, ob er richtig geraten hat und wie viele Versuche er bisher hatte. Hat er richtig geraten wird die Anzahl der Versuche und der Name des Spielers in ein Ergebnisfile angehängt.

Eine mögliche grafische Oberfläche zu diesem Hütchenspiel sieht wie folgt aus:



Der Benutzer muss folgendes über der Oberfläche tun können:

- Spielername eingeben
- Anklicken von drei Buttons, die die Hütchen realisieren.
  Nach jedem Spielzug (Aufdecken eines Hütchens) wird die Kugel zufällig unter einem Hütchen verteilt.
  Verwenden Sie dazu z.B. eine zufällige erzeugte Integerzahl zwischen 1 und 3!
- Ein Button zum Beenden der Applikation.

Im Ausgabefeld muss stehen:

- Wie viele Versuche bisher gemacht wurden.
- Ob das richtige Hütchen getippt wurde oder nicht.
- Wurde richtig getippt muss an eine Ergebnisdatei der Name des Spielers und die Anzahl der Versuche angehängt werden.

Zusatzaufgabe:
Erweitern Sie die Applikation um einen Button „Statistics“. Bei Anklicken des Buttons soll das Ergebnisfile gelesen werden. Aus dem Ergebnisfile soll der Durchschnitt der bisherigen Versuche ermittelt und ins Ausgabefeld ausgegeben werden.