Hochschule Harz	FB Automatisierung und Informatik
Programmierung2	Dipl.-Inf., Dipl.-Ing. (FH) M. Wilhelm
Tutorial 04:	„Programmierung 2“ für MI / WI Thema: **Swing, JSplitPane und I/O**

Versuchsziele

Kenntnisse in der Anwendung von:

- Erstellen einer Swing-Anwendung mit
  - JSplitPane
  - JTextArea
  - JList
  - JMenuBar, JMenu, JMenuItem
  - I/O-Operationen (Hausaufgabe)
  - XML (Hausaufgabe)
  - Excel (Hausaufgabe)

Tutorial04: Swing: Erstellen eines JFrames mit einem JSplitPane

In dieser Aufgabe soll das unten abgebildete Fenster erstellt werden.

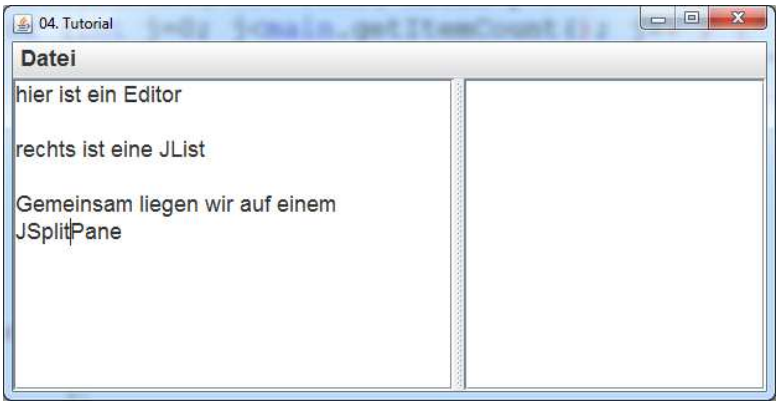

Abbildung 1 Musterlösung

Abbildung 2 Menüstruktur

1. Teilaufaufgaben: Projekt erstellen

- Erstellen Sie ein neues Eclipse–Projekt:
  - Projektname: Tutorial04 oder Aufgabe04
- Erstellen Sie eine neue Klasse
  - Menü File, Eintrag New, Eintrag class
  - Name: Tutorial04 oder Aufgabe04
- Erstellen Sie aus dem vorgegebene JFrame-Rahmen folgendes Programm:
  - Ein JSplitPane
  - Ein JTextArea als Editor. Bitte den Scroll-Mechanismus beachten.
  - Ein JList für die Ausgabe. Bitte den Scroll-Mechanismus beachten.
  - Die Menüs einbauen

Rahmen:

```
//Titel:            4. Tutorial
//Version:          1,0
//Copyright:        Copyright (c) 2013
//Autor:            M. Wilhelm
//Organisation:     HS-Harz
//Beschreibung:     BorderLayout mit SplitPane, JList, Menues und I/O mit XML


import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

import java.util.*;
import java.io.*;

import javax.swing.filechooser.*;



    // hier fehlt Code  impl...
public class Tutorial04 extends JFrame {

   public static final long serialVersionUID=1;

    // hier fehlt Code: globale Variablen der MenuItems und editor und JList

  public Tutorial04() {
    setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
    setGUI();
    setMenus();
  }

  private void setGUI() {
    setSize(700, 450);
    setTitle("04. Tutorial");

    this.getContentPane().setLayout( new BorderLayout() );

    // hier fehlt Code: JSplitPane

    // editor.setFont( new Font("Arial", Font.PLAIN,18)  );

  } // setGUI


    // setzt die Menüs
  private void setMenus() {
    JMenuBar menuBar1 = new JMenuBar();
    // hier fehlt Code: MenuItems einfuegen, Menu-Event


  } // setMenus

    // hier fehlt Code: Methoden der Menu-Events


  public static void main(String[] args) {
    Tutorial04 frame = new Tutorial04();
    frame.setVisible(true);
 }

}  // Tutorial04
```

2. Teilaufaufgaben: Menü-Events erstellen und die Aktionperformed-Methode einbauen

- Aktion-Events
    - Einfügen der Schnittstelle ActionListener
    - Registrierung der Menüs
    - Einbau der ActionListener-Methode
        - Abfragen der drei Menüs
    - Erstellen der drei Methoden
        - Erster Test mit „syso“

3. Teilaufaufgaben: Menü-Aktionen implementieren

- Menüeintrag „Open“
    - Öffnen einer Datei mittels FileChooser
    - Aufruf der Methode „loadFile“
- Methode „loadFile“
    - Eintragen des Dateinamens in den Editor (Rest in der Hausaufgabe)

- Menüeintrag „Kopieren“
    - Der Inhalt des Editors soll **fünfzigmal** in die JList eingetragen werden.
    - Dieses soll mittels einer **lokalen** Instanz der Klasse Vektor realisiert werden

- Menüeintrag „Schließen“
    - Schließen des Fensters

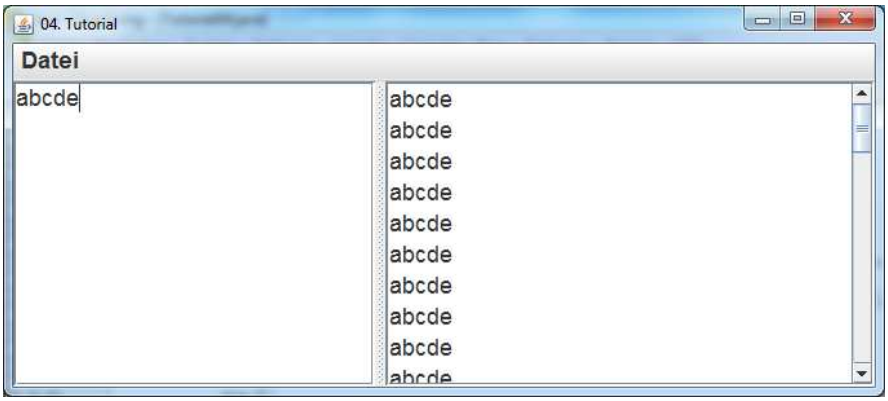

Abbildung 3 Aktion "Kopieren"

Hinweise:

Wichtige Funktionen für die Klasse JList:

- setListData(Vector liste) Update des Inhalts einer JList

Anhang:

Menü-Erstellung

Variablendeklaration:

- private JMenuBar menuBar1; // wird ins JFrame eingefuegt
- private JMenu mainFile; // kennzeichnet ein Hauptmenue
- private JMenuItem mnClose; // kennzeichnet ein Untermenue

Menü erzeugen:

```
JMenuBar menuBar1 = new JMenuBar();                 // JMenuBar erzeugen, „HauptPanel“ des Menues
JMenu mainFile = new JMenu("Datei");                // ein Hauptmenue erstellen
JMenuItem mnPaste = new JMenuItem("Einfügen");      // ein Menueeintrag erstellen
ShortKey setzen:
 mnItem.setAccelerator( KeyStroke.getKeyStroke(KeyEvent.VK_V, ActionEvent.CTRL_MASK));
Action-Event ist identisch mit denen der Schalter-Events

mainFile.add(mnPaste);                              // der Menueeintrag wird in das Hauptmenue eingefuegt
menuBar1.add(mainFile);                             // das Hauptmenue wird in den MenueBar eingefuegt

this.setJMenuBar(menuBar1);                         // am Schluss wird der MenueBar in das JFrame eingefuegt
```

Menü-Schriftarten setzen

```
for (int i=0; i<menuBar1.getComponentCount(); i++ ) {
      JMenu main  = (JMenu) menuBar1.getComponent(i);
      main.setFont( new Font("Arial", Font.BOLD,18)  );
      for (int j=0; j<main.getItemCount(); j++ ) {
         Component c = main.getItem(j);
         if (c instanceof JMenuItem) {
            JMenuItem mn = (JMenuItem) c;
            mn.setFont( new Font("Arial", Font.BOLD,18)  );
        }
      }
   }
```

FileChooser

Open-Dialog:

```
JFileChooser jFileChooser1 = new JFileChooser();
     // Ok-Schalter gedrückt ?
  jFileChooser1.setCurrentDirectory(  new File("D:\\Tutorial04\\") );
   if (jFileChooser1.showOpenDialog(this) == JFileChooser.APPROVE_OPTION) { // Ok-Schalter gedrueckt?
     String sFilename = jFileChooser1.getSelectedFile().getPath();
     // Aktion
   }
```

Save-Dialog:

```
JFileChooser jFileChooser1 = new JFileChooser();
     // Ok-Schalter gedrückt ?
  jFileChooser1.setCurrentDirectory(  new File("D:\\Tutorial04\\") );
   if (jFileChooser1.showSaveDialog(this) == JFileChooser.APPROVE_OPTION) {// Ok-Schalter gedrueckt?
     String sFilename = jFileChooser1.getSelectedFile().getPath();
     // Aktion
   }
```

Klasse JList:

public JList (); // leere Liste
public JList (Object [] liststd); // Array mit Objekten
public JList (Vector liststd); // Vector mit Objekten

setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
Nur ein Wert darf selektiert werden

setSelectionMode(ListSelectionModel.SINGLE_INTERVAL_SELECTION);
Ein Wert selektiert, oder ein Bereich

setSelectionMode(ListSelectionModel. MULTIPLE_INTERVAL_SELECTION);
Beliebiger Bereich darf selektiert werden

boolean isSelectionEmpty()
True: Es wurde kein Bereich oder Element gewählt
False: Es wurde ein Bereich oder Element gewählt

int getSelectedIndex()
Index des selektierten Eintrags

int[] getSelectedIndices()
Indizes aller selektierten Einträge

isSelectedIndex(int)

Object getSelectedValue()
Gibt das aktuelle Objekt zurück.
Falls kein Eintrag ausgewählt wurde, gibt es null zurück.

Object[] getSelectedValues()
Rückgabe aller selektierten Objekte oder null.

void setListData(Object[] listData) // Update der Einträge
void setListData(Vector listData) // Update der Einträge