Hochschule Harz	FB Automatisierung und Informatik
Programmierung 2	Dipl.-Inf. Dipl.-Ing. (FH) M. Wilhelm
Aufgabe 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)

Aufgabe 04: Erweitern des 04. Tutorials

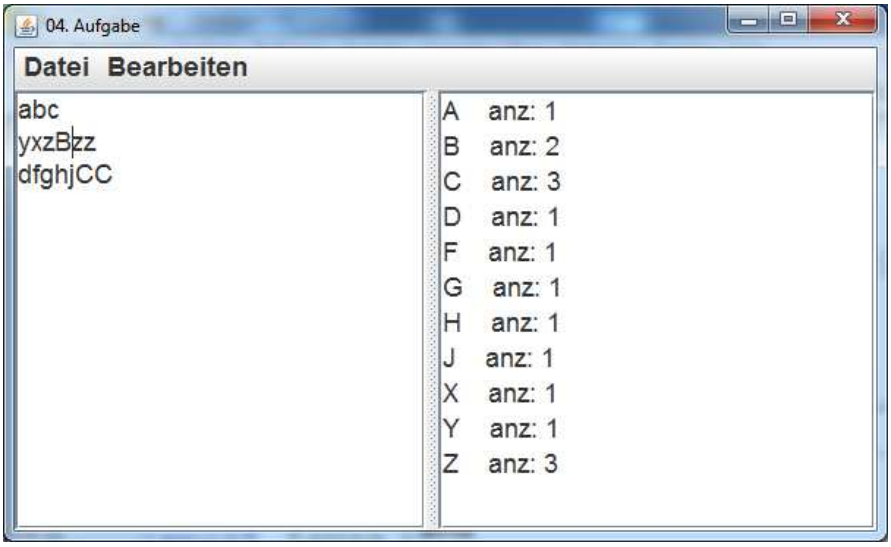

Abbildung 1 Musterlösung

Eigenschaften:

- Das Programm soll eine Häufigkeitsanalyse über den Text des Editors durchführen (A-Z).
- Aktion Berechnen:
  - Alle Buchstaben, die mindestens einmal vorkommen, werden in die JList eingetragen
- Mit Dateioperationen sollen Texte geladen und gespeichert werden.
- Als Export dienen zwei Funktionen (jeweils mit SaveDialog):
  - direkter binärer Excel-Export mittels Binary InterchangeFileFormat (BIFF)
  - einfacher XML-Export in eine XML-Datei

1. Teilaufaufgaben:

- Projekt erstellen mit dem Namen Aufgabe04
- Import des 4. Tutorials

2. Teilaufaufgaben: Menüeinträge erstellen

- Hauptmenü „Datei“ Name: mainFile
  - Eintrag „Öffnen“ Name: mnOpen
  - Eintrag „Speichern“ Name: mnSave
  - Eintrag „Schließen“ Name: mnClose
- Hauptmenü „Bearbeiten“ Name: mainEdit
  - Eintrag „Berechnen“ Name: mnCalc
  - Eintrag „Export 2 Excel“ Name: mnExcel2Excel
  - Eintrag „Export 2 XML“ Name: mnExcel2XML

3. Teilaufaufgaben: Event-Methoden erstellen und verknüpfen

- Eintrag „Öffnen“
  - Name: mnOpen_click
- Eintrag „Speichern“
  - Name: mnSave_click
- Eintrag „Schließen“
  - Name: mnClose_click
- Eintrag „Berechnen“
  - Name: mnCalc_click
- Eintrag „Export 2 Excel“
  - Name: mnExport2Excel_click
- Eintrag „Export 2 XML“
  - Name: mnExport2XML_click

Hinweise:

Wichtige Funktionen für die Klasse JList:

- setListData(Vector liste) Update des Inhalts einer JList

Test, ob eine Dateierweiterung vorhanden ist:

```
if (! sFilename.toLowerCase().endsWith(".txt") ) {
  sFilename += ".txt";
}
```

Wichtige Funktionen der Klasse Strings:

- charAt(int i) liefert das i-te Zeichen des Strings
- toCharArray() liefert den String als Array mit Zeichen. Sinnvoll für eine for-each-Schleife

Wichtige Funktionen der Klasse Char:

- Character.toString(char) Umwandlung eines Zeichen in einen String

4. Teilaufaufgaben: Erstellen einer Klasse ABC

- Attribute
  - Name: buchstabe
    - Datentyp: char
  - Name: anz
    - Datentyp: int
- Konstruktor erstellen
- Set-und Get-Methoden
- Weitere Methoden:
  - inc
    - Erhöht den Wert von „anz“
  - getBuchstabe
    - Rückgabe des Buchstabens als **String**
  - Eine Methode fehlt noch

5. Teilaufaufgaben: mnCalc_Click

- Erstellen einer globalen Variable, die man für die JList und den weiteren Funktionen benötigt.
- Prüfen der Eingabe und eine eventuelle Fehlerausgabe mittel „ErrorBox“
- 26 Instanzen der Klasse ABC erstellen
    - von A (Wert 65)
    - bis Z (Wert 90)
    - eventuell ein Cast vornehmen
- Holen des Textes
- Summieren der Häufigkeit von a bis z bzw. A bis Z.
    - Hier benötigt man einen Eintrag der oberen Hinweise
- Groß- und Kleinbuchstaben werden **NICHT** unterschieden.
- Alle Buchstaben mit der Anzahl größer Null in die JList eintragen

7. Teilaufaufgaben: mnOpen_click

- Laden einer Textdatei in den Editor
- Beispiel (siehe Homepage)

8. Teilaufaufgaben: mnSave_click

- Speichern des Editortextes in eine Textdatei.
- Beispiel (siehe Homepage)

9. Teilaufaufgaben: mnExport2Excel_click

- Speichern der Ergebnisse der JList in eine Excel-Datei.
- Es darf kein CSV-Format verwendet werden
- Beispiel (siehe Homepage)

Beispiel:

hier ein Text
auch ein x und X

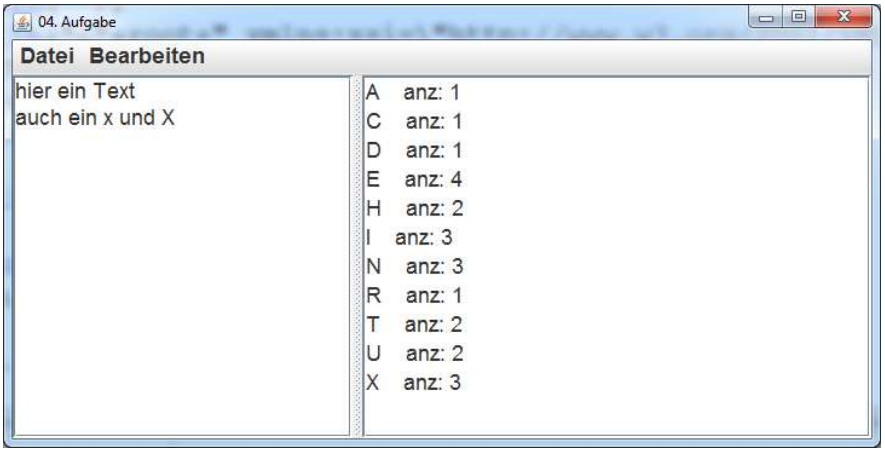

Abbildung 2 Beispielberechnung

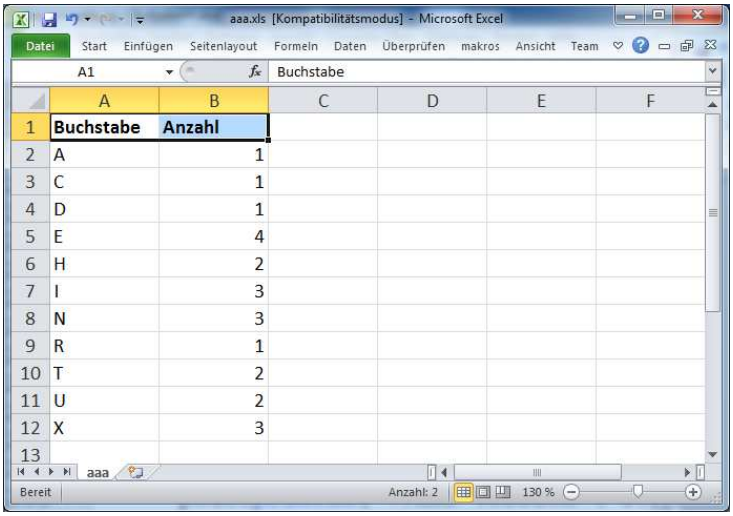

Abbildung 3 Excelexport

10. Teilaufaufgaben: mnExport2XML_click

- Speichern der Ergebnisse der JList in eine XML-Datei.

Benötigtes Interface:

```
interface IXMLExport {
  public void writeXML(PrintStream p);
}
```

Speicherfunktion:

```
private void SaveXML(String filename, Vector liste) {
  FileOutputStream fout;
  PrintStream pout;
  if (! filename.toLowerCase().endsWith(".xml")) {
    filename+=".xml";
  }
  try {
   String root="root";
   fout = new FileOutputStream(filename);
   pout = new PrintStream(fout);
   pout.println("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
   pout.println("");
   pout.println("<"+root+" xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\">");
    // hier fehlt Code

   pout.println("</"+root+">");
  }
  catch (IOException e) {
      System.err.println("IOException: " + e);
  }
}  // SaveXML
```

Ergebnis:

```
<?xml version="1.0" encoding="UTF-8"?>

<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <ABC>
    <Buchstabe>A</Buchstabe>
    <Anzahl>1</Anzahl>
  </ABC>
  <ABC>
    <Buchstabe>C</Buchstabe>
    <Anzahl>1</Anzahl>
  </ABC>
  <ABC>
    <Buchstabe>D</Buchstabe>
    <Anzahl>1</Anzahl>
  </ABC>
  <ABC>
    <Buchstabe>E</Buchstabe>
    <Anzahl>4</Anzahl>
  </ABC>
  <ABC>
    <Buchstabe>H</Buchstabe>
    <Anzahl>2</Anzahl>
  </ABC>
  <ABC>
    <Buchstabe>I</Buchstabe>
    <Anzahl>3</Anzahl>
  </ABC>
  <ABC>
    <Buchstabe>N</Buchstabe>
    <Anzahl>3</Anzahl>
  </ABC>
  <ABC>
    <Buchstabe>R</Buchstabe>
    <Anzahl>1</Anzahl>
  </ABC>
  <ABC>
    <Buchstabe>T</Buchstabe>
    <Anzahl>2</Anzahl>
  </ABC>
  <ABC>
    <Buchstabe>U</Buchstabe>
    <Anzahl>2</Anzahl>
  </ABC>
  <ABC>
    <Buchstabe>X</Buchstabe>
    <Anzahl>3</Anzahl>
  </ABC>
</root>
```

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

Save-Dialog:

```
JFileChooser jFileChooser1 = new JFileChooser();
     // Ok-Schalter gedrueckt ?
  jFileChooser1.setCurrentDirectory(  new File("D:\\Tutorial04\\") );
   if (jFileChooser1.showSaveDialog(this) == JFileChooser.APPROVE_OPTION) {
     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