Hochschule Harz	FB Automatisierung und Informatik
Programmierung2	Dipl.-Inf. Dipl.-Ing. (FH) M. Wilhelm
Aufgabe 01:	„Programmierung 2“ für MI / WI Thema: **Sortieren vonKlassen in JFrame**

Versuchsziele

Kenntnisse in der Anwendung von:

- Erstellen einer Swing-Anwendung
- Benutzung von Klassen
    - Definition
    - Erzeugung
    - Verwendung von Arrays
- Sortierung und Suchen von Objekten

Aufgabe01

In dieser Aufgabe werden verschiedene Such-Algorithmen implementiert.

- Erstellen Sie ein neues Eclipse–Projekt:
    - Projektname: Aufgabe01
    - Klassenname: Aufgabe01
- Erstellen Sie eine neue Klasse
    - Menü File, Eintrag New, Eintrag class
    - Name: **`Aufgabe01`**
- Kopieren Sie folgenden Code in den Editor:
- Tragen Sie unbedingt Ihre Matrikelnummer in den Quellcode ein.
- Bauen Sie einen dritten „Aufgaben“-Schalter ein
    - Schalter deklarieren und erzeugen
    - Eintragen in das JPanel
    - ActionListener mit dem Schalter verbinden
    - Event-Methode einbauen

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


//  Matrikelnummer:

public class Aufgabe01 extends JFrame {

  JTextField editzeile = new JTextField();
  JTextArea editor = new JTextArea();

  //Frame konstruieren
  public Aufgabe01() {
    setSize(1000, 700);
    setTitle("1. Aufgabe");
    setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
    setGUI();
  } // create


  void setGUI() {
    JButton bnAufg1 = new JButton("1. Aufgabe");
    JButton bnAufg2 = new JButton("2. Aufgabe");
    JButton bnEsc = new JButton("Abbrechen");

    editzeile.setFont( new Font("Arial", Font.BOLD,28));
```

```
    editzeile.setText("10");  // N =10

    this.getContentPane().add(editzeile, BorderLayout.NORTH);

      // Center
    editor.setFont( new Font("Arial", Font.BOLD,28));
    editor.setText("");
    getContentPane().add( new JScrollPane(editor), BorderLayout.CENTER);

      // South
    bnAufg1.setFont( new Font("Arial", Font.BOLD,28));
    bnAufg2.setFont( new Font("Arial", Font.BOLD,28));
    bnEsc.setFont( new Font("Arial", Font.BOLD,28));

    JPanel panelBn = new JPanel();
    panelBn.setLayout( new FlowLayout( FlowLayout.RIGHT) );
    panelBn.add(bnAufg1);
    panelBn.add(bnAufg2);
    panelBn.add(bnEsc) ;
    this.getContentPane().add(panelBn, BorderLayout.SOUTH);

    bnAufg1.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        bnAufg1_Click();
      }
    });
    bnAufg2.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        bnAufg2_Click();
      }
    });
    bnEsc.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        bnEsc_Click();
      }
    });

  }  // setGUI


  void bnAufg1_Click() {
    editor.setText("Aufg1" );
  }  // bn_Aufg1


  void bnAufg2_Click() {
    editor.setText("Aufg2" );
  }  // bn_Aufg2


  void bnEsc_Click() {
    this.dispose();
    System.exit(0);
  }


  public static void main(String[] args) {
    Aufgabe01 frame = new Aufgabe01();
    frame.setVisible(true);
  }
} // Aufgabe01
```

Schalter „Aufgabe1“

- Auslesen der Editorzeile in einer String-Variablen.
- Umwandeln des String in eine int-Zahl „n“.
- Danach müssen **zwei** Fehlermeldungen implementiert werden.
- Die **Fehlerausgaben** sollen im Editor erfolgen. Das heißt, dass Teile des unteren Source-Codes nicht korrekt sind.
- Erstellen einer Funktion „fakultaet“
    - Parameter: int
    - Rückgabewert: int
- Bei der Eingabe einer korrekten Zahl:
    - In einer for-Schleife sollen pro Laufindex die Fakultät mittels einer rekursiven Funktion berechnet und im Editor ausgegeben werden

Methoden eines JTextField:

- setText(String)
- **String s = getText()**
- setFont(new Font("Arial", Font.BOLD,18))

Methoden der Klasse JTextArea

- **setText(String)**
- String s = getText()
- **append(String)**
- setFont(new Font("Arial", Font.BOLD,18))

Umwandlung eines String:

```
String sValue="123.44";
int n;
try {
   n= Integer.parseInt(sValue);
}
catch ( NumberFormatException e ){
   System.out.println( "Konvertierungsfehler: '%s' "+sValue );
}
```

Lösungen:

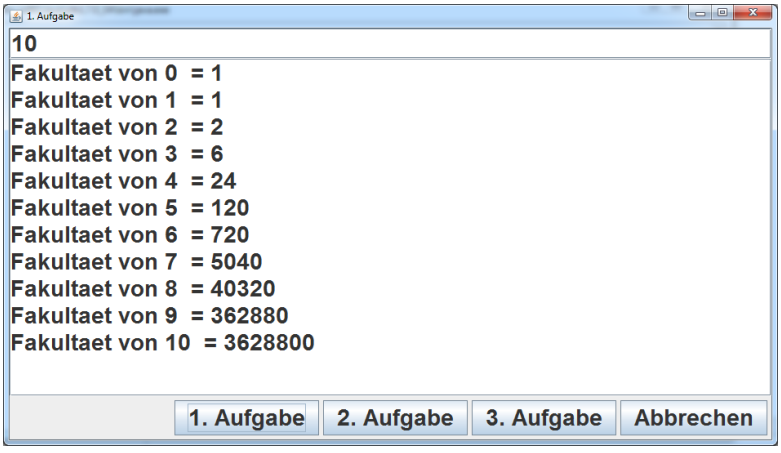


Abbildung 1 Lösung der Fakultäten von 0 bis 10

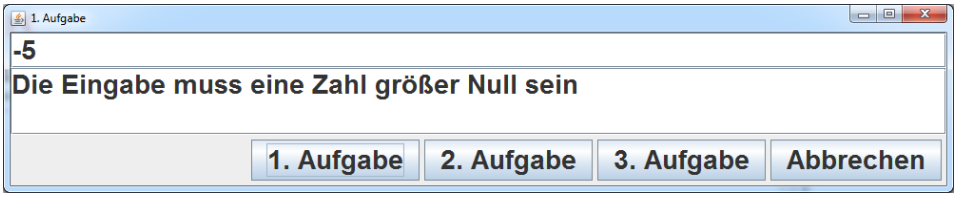


Abbildung 2 Fehler in der Eingabe (negative Zahl)

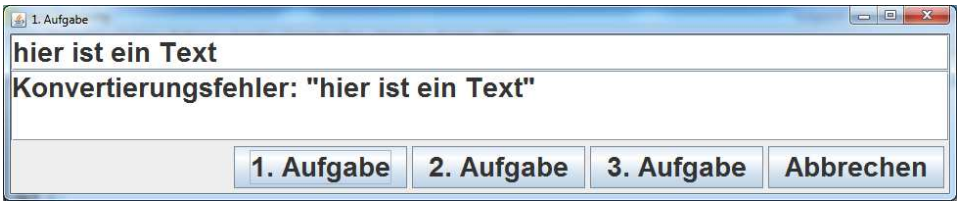


Abbildung 3 Fehler in der Eingabe (Text kann nicht in einer Zahl konvertiert werden)

Schalter „Aufgabe2“

In dieser Teilaufgabe werden Mitarbeiter in einem Array gespeichert. Danach sollen verschiedene Mitarbeiter mittels einer Methode „search2“ gesucht und im Editor ausgegeben werden.

Teilaufaufgaben:

- Erstellen einer neuen Klasse „Mitarbeiter“
  - private Attribute
    - String Variable „name“
    - Integer-Variable mnr (Mitarbeiternummer)
  - Konstruktor
    - String Variable „name“
    - Integer-Variable „mnr“
    - Setzen der Variablen
  - Einbauen der toString-Methode
  - Einbau der getter- und setter-Methoden
- Sortieren mit compareTo
  - Implementieren der compareTo-Methode
    - 1. Sortierkriterium: mnr
- Dann scheint noch etwas in der ersten Zeile zu fehlen (imp…)
- Kopieren des unteren Quellcodes in die Event-Methode „bnAufg2_Click“

```
void bnAufg2_Click() {
    final int MAX=10000;
    long t1, t2;
    Mitarbeiter[] feld = new Mitarbeiter[ MAX ];
    editor.setText("Aufg2" );
    for (int i=0; i<MAX; i++) {
      feld[i] = new Mitarbeiter("Meier"+(i+1), (int) (Math.random()*10000000) );
    }
    feld[5500] = new Mitarbeiter("Meier4000", 4000 );
    feld[500] = new Mitarbeiter("Meier100000", 100000 );
    ArrayList liste;
    t1 =  System.nanoTime();
    liste = search2(feld, 4000, 100000);
    t2 =  System.nanoTime();
    editor.append("\nAnzahl: "+liste.size() );
    editor.append("\nZeit: "+(t2-t1) );
}  // bn_Aufg2
```

- Implementieren der Suchmethode „search2“
  - Parameter:
    - Feld mit Mitarbeiten
    - Integervariable „von“ (untere Mitarbeiternummer)
    - Integervariable „bis“ (obere Mitarbeiternummer)
  - Rückgabewert:
    - Eine Liste der Mitarbeiter, deren Nummer in den Grenzen „von“ und „bis“ liegen.

Schalter „Aufgabe3“

In dieser Teilaufgabe werden Mitarbeiter in einem Array gespeichert. Danach sollen verschiedene Mitarbeiter mittels einer Methode „search3“ gesucht und im Editor ausgegeben werden. Die Methode „search3“ arbeitet aber mit Hilfe eines **sortierten Feldes** (Klasse Arrays). Das Feld muss in der Event-Methode sortiert werden. Dieses sortierte Feld wird der Methode „search3“ dann übergeben, die dann mittels der implementierten Java-Methode „binarySearch“ den Anfangs- und Endindex bestimmt. Danach ist die „Suche“ wesentlich einfacher als in Aufgabe2. In einer For-Schleife werden die Mitarbeiter in die Liste eingefügt.

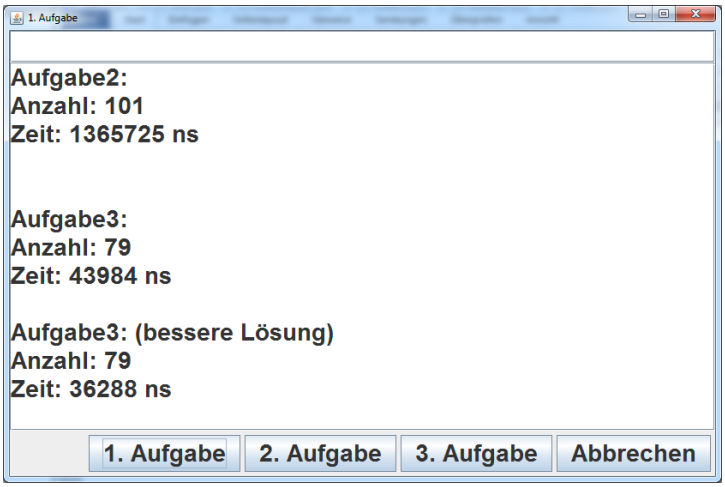
Details:

- Implementieren der Suchmethode „search3“
  - Parameter:
    - Feld mit Mitarbeiten
    - Integervariable „von“ (untere Mitarbeiternummer)
    - Integervariable „bis“ (obere Mitarbeiternummer)
  - Rückgabewert:
    - Eine Liste der Mitarbeiter, deren Nummer in den Grenzen „von“ und „bis“ liegen.
  - Quellcode in der Methode:
    - Suche den Index der Mitarbeiternummer „von“.
    - Suche den Index der Mitarbeiternummer „bis“.
    - Kopieren der Referenzen der Objekte in die Liste.
    - Rückgabe der gesuchten Objekte

Hinweise:

- parseInt / try / catch: Seite 5
- ActionListener: Seite 28
- Sortierung / Suchen: Seite 15
- Vergleichen Sie die Suchzeiten der beiden Methoden (Aufgabe2 und Aufgabe3).
- Mit weiteren „internen“ Methoden kann man den Aufwand noch einmal geringfügig reduzieren.

Beispiel-Lösung:



Zeitvergleich mit MAX gleich 100000

Aufgabe2:
Anzahl: 987
Zeit: 382300 ns

Aufgabe3:
Anzahl: 1048
Zeit: 28957 ns

Aufgabe3: (bessere Lösung):
Anzahl: 1048
Zeit: 19060 ns