Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. D. Sabel, Dr. P. Wendler

WS 2018/19
Übungsblatt 12

Übungen zu Einführung in die Informatik: Programmierung und Software-Entwicklung:

Lösungsvorschlag

Aufgabe 12-1 Ausnahmen und Ausnahmebehandlung *Präsenz*

Gegeben sei eine Methode zum Umwandeln von Strings in Zeichenketten, die nur aus Großbuchstaben bestehen. Die Methode `printUpperCase(String s)` wandelt jeweils einen einzelnen String in Großbuchstaben um und gibt den umgewandelten String zurück.

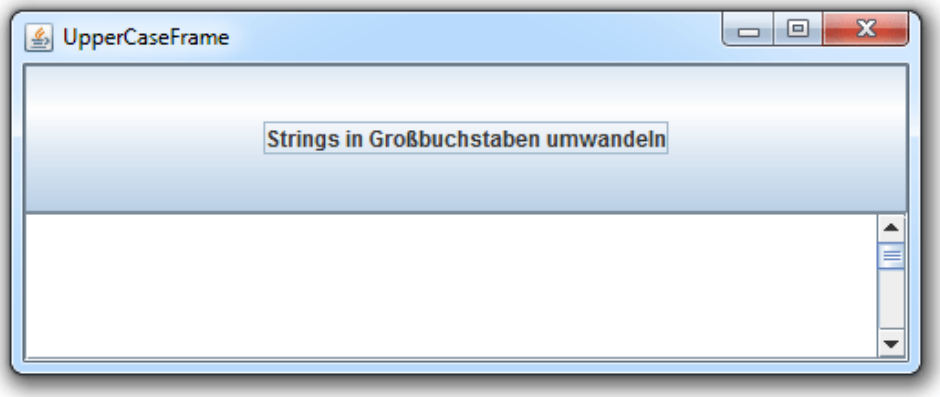
```
1  public String printUpperCase(String s)
2        throws NullStringException, EmptyStringException {
3    if (s == null) {
4      throw new NullStringException();
5    }
6    if (s.length() == 0) {
7      throw new EmptyStringException();
8    }
9
10   return s.toUpperCase();
11 }
```

Die Methode `printUpperCase` verwendet die folgenden Klassen `NullStringException` und `EmptyStringException`:

```
1 public class NullStringException extends Exception {
2 }
3
4 public class EmptyStringException extends Exception {
5 }
```

In dieser Aufgabe sollen Sie ein Programm mit einer grafischen Benutzeroberfläche implementieren, welches die Umwandlung eines Arrays von Strings testet. Die Anzahl der umzuwandelnden Strings `num` soll vom Benutzer frei wählbar sein und wie gewohnt durch einen modalen Dialog zu Beginn der Anwendung abgefragt werden.

a) Die grafische Benutzeroberfläche soll wie folgt aussehen:



Es soll einen Button mit der oben angegebenen Aufschrift geben. Darunter soll der Ausgabebereich für die umgewandelten Strings bzw. für Meldungen über den Erfolg der Umwandlung platziert werden.

Da die Ausgabe länger sein kann, können Sie mit Hilfe der Klassen `ScrollPane` und `ScrollPaneConstants` der Swing-Bibliothek dem Ausgabebereich auf folgende Weise eine vertikale Scrollbar hinzufügen:

```
1 JScrollPane scrollPane = new JScrollPane(this.ausgabeBereich);
2 scrollPane.setHorizontalScrollBarPolicy(
3         ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
```

Schreiben Sie eine Klasse `UpperCaseFrame`, die die Hauptklasse dieser grafischen Benutzeroberfläche sein soll und das Fenster erzeugt. Um Ihr Programm ausführen zu können, schreiben Sie eine weitere Klasse `UpperCaseFrameMain`, die Sie wie gewohnt im gleichen Ordner wie Ihre Klasse `UpperCaseFrame` abspeichern.

Lösung:
Bei der Implementierung der grafischen Benutzeroberfläche geht man wie folgt vor:

Deklarieren Sie eine Klasse `UpperCaseFrame`, die die Hauptklasse Ihrer grafischen Benutzeroberfläche wird und deshalb von der Klasse `JFrame` erbt. Ihre Klasse `UpperCaseFrame` soll zwei Attribute haben:

- ein Attribut vom Klassentyp `JButton` für den Button,
- ein Attribut vom Klassentyp `JTextArea`, das als Ausgabebereich für die spätere Rückmeldung über das Ergebnis dient.

Schreiben Sie einen Konstruktor, der den `UpperCaseFrame` mit einem entsprechenden Titel und Größe (wir wählen hier 500x200 Pixel) initialisiert. In dem Konstruktor sollen weiterhin alle Attribute korrekt initialisiert werden. Das Layout des `ContentPane` des `UpperCaseFrame`s wird auf ein `GridLayout` mit zwei Zeilen und einer Spalte initialisiert und der Button sowie der Ausgabebereich darauf platziert. Fügen Sie abschließend noch ein, dass das Programm ordnungsgemäß beendet wird, falls der `UpperCaseFrame` geschlossen wird. Benutzen Sie dazu die Methode `setDefaultCloseOperation`.

Eine Implementierung der Klassen `UpperCaseFrame` und `UpperCaseFrameMain` für die gesamte Aufgabe finden Sie weiter unten.

b) Ergänzen Sie Ihre Klasse `UpperCaseFrame` um eine Ereignisbehandlung für den Button. Wird dieser Button gedrückt, soll der Benutzer zunächst in einer Methode `umwandelnStrings` mit Hilfe der Klasse `JOptionPane` nach der Anzahl `num` der umzuwandelnden Strings gefragt werden. Diese Methode soll sich auch darum kümmern, dass im Ausgabebereich Inhalte vorhergehender Eingaben gelöscht werden. Ein Test-Array `testArray` mit umzuwandelnden Strings soll in der Klasse als Konstante angelegt werden und den Wert `{null, "", "test"}` haben. Die Strings des Test-Arrays sollen durch Aufruf der Methode `printUpperCaseArray` umgewandelt werden. Diese Methode verwendet die vorhergenannte Methode `printUpperCase` und greift auf das Attribut `ausgabeBereich` für den Ausgabebereich in der grafischen Benutzeroberfläche zurück.

```
1 public void printUpperCaseForArray(String[] array, int num) {
2   for (int i = 0; i < num; i++) {
3     this.ausgabeBereich.append(this.printUpperCase(array[i]) + "\n");
4   }
5   this.ausgabeBereich.append("Fertig!\n");
6 }
```

Fügen Sie die Methoden in Ihre Implementierung der Klasse `UpperCaseFrame` ein und rufen Sie sie an geeigneter Stelle auf. Warum kommt es bei der gegenwärtigen Formulierung der Methode zu einem Kompilierfehler? Fügen Sie in die Methode `printUpperCaseForArray` eine Ausnahmebehandlung ein, so dass das Programm fehlerfrei kompiliert wird und bei Ausführung mit Eingabe 3 folgender Text auf der Konsole erscheint:

```
Exception bei Versuch 1
Exception bei Versuch 2
TEST
Fertig!
```

Lösung:
Bei der vorgegebenen Formulierung der beiden Methoden kommt es zu einem Kompilierfehler, weil es sich bei `NullStringException` und `EmptyStringException` um *Checked Exceptions* handelt, die abgefangen werden müssen.

Implementierung der Methode `umwandelnStrings`:

```
1 private void umwandelnStrings() {
2   String einlesenAnzahlStrings =
3       JOptionPane.showInputDialog("Parameter \"num\" eingeben: ");
4   int num = Integer.parseInt(einlesenAnzahlStrings);
5
6   this.ausgabeBereich.setText("");
7   this.printUpperCaseForArray(testArray, num);
8 }
```

Einfügen eines `try`- und eines `catch`-Blocks in die Methode `printUpperCaseForArray`, sowie einer `append`-Anweisung für entsprechende Ausgaben:

```
1 public void printUpperCaseForArray(String[] array, int num) {
2   for (int i = 0; i < num; i++) {
3     try {
4       this.ausgabeBereich.append(this.printUpperCase(array[i]) + "\n");
5     } catch (Exception e) {
6       this.ausgabeBereich.append(
7           "Exception bei Versuch " + (i + 1) + "\n");
8     }
9   }
10  this.ausgabeBereich.append("Fertig!\n");
11 }
```

Eine Implementierung der Klasse `UpperCaseFrame` für die gesamte Aufgabe finden Sie weiter unten. Die entsprechende Methode heißt hier `printUpperCaseForArrayB`.

c) Ändern Sie jetzt den `catch`-Block in der Methode `printUpperCaseForArray` und fügen Sie gegebenenfalls weitere `catch`-Blöcke ein, so dass bei Ausführung des Programms folgender Text ausgegeben wird:

```
text[0] war null!
text[1] war leer!
TEST
Fertig!
```

Lösung:

```
1 public void printUpperCaseForArray(String[] array, int num) {
2   for (int i = 0; i < num; i++) {
```

```
3       try {
4         this.ausgabeBereich.append(this.printUpperCase(array[i]) + "\n");
5       } catch (NullStringException e) {
6         this.ausgabeBereich.append("text[" + i + "] war null!\n");
7       } catch (EmptyStringException e) {
8         this.ausgabeBereich.append("text[" + i + "] war leer!\n");
9       }
10    }
11    this.ausgabeBereich.append("Fertig!\n");
12  }
```

Eine Implementierung der Klasse `UpperCaseFrame` für die gesamte Aufgabe finden Sie weiter unten. Die entsprechende Methode heißt hier `printUpperCaseForArrayC`.

d) Rufen Sie jetzt testweise das Programm mit dem Wert 4 für den Parameter `num` auf. Wenn das Programm jetzt ausgeführt wird, dann bricht es mit einer „ungefangenen“ Exception ab. Fügen Sie weitere `try`-, `catch`- oder `finally`-Blöcke ein, so dass dies nicht mehr passiert und stattdessen folgende Ausgabe erscheint:

```
text[0] war null!
Nach Versuch 1
text[1] war leer!
Nach Versuch 2
TEST
Nach Versuch 3
Fehler! Falscher Arrayzugriff!
Nach Versuch 4
Fertig!
```

Vermeiden Sie dabei möglichst Wiederholungen. Insbesondere soll nicht die Anweisung `append` mehrmals mit den gleichen übergebenen Parametern verwendet werden. Testen Sie Ihr Programm anschließend nochmals mit dem Wert 4 für den Parameter `num`.

Lösung:

```
1  public void printUpperCaseForArray(String[] array, int num) {
2    for (int i = 0; i < num; i++) {
3      try {
4        this.ausgabeBereich.append(this.printUpperCase(array[i]) + "\n");
5      } catch (NullStringException e) {
6        this.ausgabeBereich.append("text[" + i + "] war null!\n");
7      } catch (EmptyStringException e) {
8        this.ausgabeBereich.append("text[" + i + "] war leer!\n");
9      } catch (ArrayIndexOutOfBoundsException e) {
10       this.ausgabeBereich.append("Fehler! Falscher Arrayzugriff!\n");
11     } finally {
12       this.ausgabeBereich.append("Nach Versuch " + (i + 1) + "\n");
13     }
14   }
15   this.ausgabeBereich.append("Fertig!\n");
16 }
```

Eine Implementierung der Klasse `UpperCaseFrame` für die gesamte Aufgabe finden Sie weiter unten. Die entsprechende Methode heißt hier `printUpperCaseForArrayD`.

e) Wie könnte man die `RuntimeException` aus Teil d durch bessere Programmierung vermeiden?

Lösung:

Es muss in der Methode `umwandelnStrings` durch eine `if`-Abfrage sichergestellt werden, dass beim Aufruf von `printUpperCaseForArray` vernünftige aktuelle Parameter verwendet werden.

```
1  private void umwandelnStrings() {
2    String einlesenAnzahlStrings =
3        JOptionPane.showInputDialog("Parameter \"num\" eingeben: ");
4    int num = Integer.parseInt(einlesenAnzahlStrings);
5
6    if ((num >= 0) && (num <= testArray.length)) {
7      this.ausgabeBereich.setText("");
8      this.printUpperCaseForArray(testArray, num);
9    } else {
10     this.ausgabeBereich.setText("Ungueltige Anzahl " + num);
11   }
12 }
```

Eine Implementierung der Klassen `UpperCaseFrameMain` und `UpperCaseFrame` für die gesamte Aufgabe finden Sie auch im ZIP-Archiv. Die entsprechende Methode heißt hier `umwandelnStringsE`.

```
1  /**
2   * Diese Klasse startet den UpperCaseFrame
3   *
4   * @author Annabelle Klarl
5   */
6  public class UpperCaseFrameMain {
7    /**
8     * Dieses Programmstueck startet das Programm.
9     *
10    * @param args
11    */
12   public static void main(String[] args) {
13     UpperCaseFrame uppercaseFrame = new UpperCaseFrame();
14     uppercaseFrame.setVisible(true);
15   }
16 }
```

```
1  import java.awt.Container;
2  import java.awt.GridLayout;
3  import java.awt.event.ActionEvent;
4  import java.awt.event.ActionListener;
5  import javax.swing.JButton;
6  import javax.swing.JFrame;
7  import javax.swing.JOptionPane;
8  import javax.swing.JScrollPane;
9  import javax.swing.JTextArea;
10 import javax.swing.ScrollPaneConstants;
11
12 /**
13  * Diese Klasse repraesentiert das Hauptfenster der Praesenzaufgabe
14  * zum Umwandeln von Strings in Grossbuchstaben.
15  *
16  * @author Annabelle Klarl
17  */
18 public class UpperCaseFrame extends JFrame implements ActionListener {
19   private JButton anzahlstringsButton;
20
21   private JTextArea ausgabeBereich;
22
23   public static final String[] TESTARRAY = {null, "", "test"};
24
```

```
25  /** In diesem Programmstueck wird das Fenster erzeugt */
26  public UpperCaseFrame() {
27    /* In der Kopfleiste des Fenster steht "UpperCaseFrame" */
28    this.setTitle("UpperCaseFrame");
29
30    /* Das Fenster ist 500 Pixel breit und 200 Pixel hoch. */
31    this.setSize(500, 200);
32
33    /* Hier werden alle Buttons erzeugt. */
34    this.anzahlstringsButton =
35        new JButton("Strings in Grossbuchstaben umwandeln");
36
37    /* Hier wird der Ausgabe-Bereich erzeugt. */
38    this.ausgabeBereich = new JTextArea(60, 100);
39    JScrollPane scrollPane = new JScrollPane(this.ausgabeBereich);
40    scrollPane.setHorizontalScrollBarPolicy(
41        ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
42
43    /*
44     * Der ContentPane ist der Ausschnitt des Fensters, auf dem Widgets d.h.
45     * Interaktionselemente (wie eine TextArea oder ein Button) platziert
46     * werden koennen.
47     */
48    Container contentPane = this.getContentPane();
49    contentPane.setLayout(new GridLayout(2, 1));
50    /* Hier wird der Button platziert. */
51    contentPane.add(this.anzahlstringsButton);
52    /* Hier wird der Ausgabebereich platziert. */
53    contentPane.add(scrollPane);
54
55    /*
56     * Hier wird der Frame als Listener fuer Knopfdruck-Ereignisse bei jedem
57     * der Buttons registriert.
58     */
59    this.anzahlstringsButton.addActionListener(this);
60
61    /*
62     * Wird das Fenster geschlossen (d.h. auf X gedrueckt), wird mit Hilfe
63     * dieser Programmzeile auch unser Programm ordnungsgemaess beendet.
64     */
65    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
66  }
67
68  @Override
69  public void actionPerformed(ActionEvent e) {
70    // je nach Button entsprechende Methode ausfn
71    Object source = e.getSource();
72    if (source == this.anzahlstringsButton) {
73      this.umwandelnStringsE();
74    }
75  }
76
77  /**
78   * implementiert die Funktionalitaet
79   * des "Strings in Grossbuchstaben umwandeln"-Buttons
80   */
81  private void umwandelnStrings() {
82    String einlesenAnzahlStrings =
83        JOptionPane.showInputDialog("Parameter \"num\" eingeben: ");
84    int num = Integer.parseInt(einlesenAnzahlStrings);
85
86    this.ausgabeBereich.setText("");
87    this.printUpperCaseForArray(TESTARRAY, num);
88  }
89
```

```
90   /**
91    * implementiert die Funktionalitaet
92    * des "Strings in Grossbuchstaben umwandeln"-Buttons
93    */
94   private void umwandelnStringsE() {
95     String einlesenAnzahlStrings =
96         JOptionPane.showInputDialog("Parameter \"num\" eingeben: ");
97     int num = Integer.parseInt(einlesenAnzahlStrings);
98
99     if ((num >= 0) && (num <= TESTARRAY.length)) {
100      this.ausgabeBereich.setText("");
101      this.printUpperCaseForArray(TESTARRAY, num);
102    } else {
103      this.ausgabeBereich.setText("Ungueltige Anzahl " + num);
104    }
105  }
106
107  /**
108   * wandelt einen String in Grossbuchstaben um
109   *
110   * @param s
111   * @return
112   */
113  public String printUpperCase(String s)
114        throws NullStringException, EmptyStringException {
115    if (s == null) {
116      throw new NullStringException();
117    }
118    if (s.length() == 0) {
119      throw new EmptyStringException();
120    }
121
122    return s.toUpperCase();
123  }
124
125  public void printUpperCaseForArray(String[] array, int num) {
126    printUpperCaseForArrayD(array, num);
127  }
128
129  /**
130   * wandelt ein Array von Strings in Grossbuchstaben um
131   *
132   * @param array
133   * @param num
134   * @return
135   */
136  public void printUpperCaseForArrayB(String[] array, int num) {
137    for (int i = 0; i < num; i++) {
138      try {
139        this.ausgabeBereich.append(this.printUpperCase(array[i]) + "\n");
140      } catch (Exception e) {
141        this.ausgabeBereich.append("Exception bei Versuch " + (i + 1) + "\n");
142      }
143    }
144    this.ausgabeBereich.append("Fertig!\n");
145  }
146
147  /**
148   * wandelt ein Array von Strings in Grossbuchstaben um
149   *
150   * @param array
151   * @param num
152   * @return
153   */
154  public void printUpperCaseForArrayC(String[] array, int num) {
```

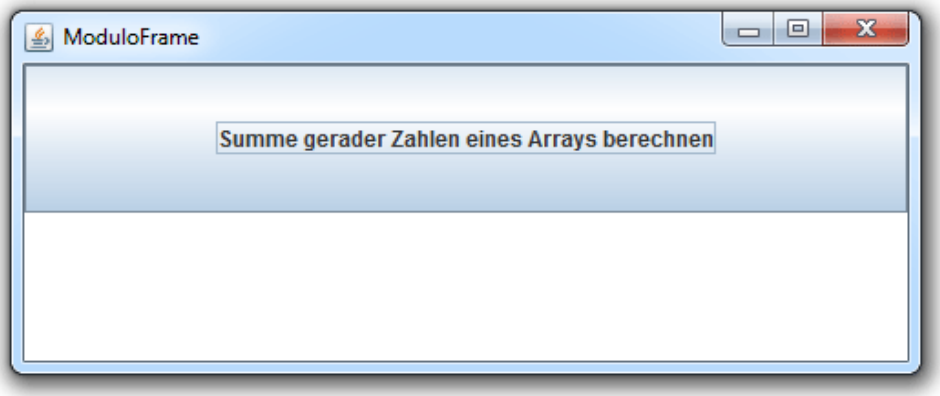
```
155     for (int i = 0; i < num; i++) {
156       try {
157         this.ausgabeBereich.append(this.printUpperCase(array[i]) + "\n");
158       } catch (NullStringException e) {
159         this.ausgabeBereich.append("text[" + i + "] war null!\n");
160       } catch (EmptyStringException e) {
161         this.ausgabeBereich.append("text[" + i + "] war leer!\n");
162       }
163     }
164     this.ausgabeBereich.append("Fertig!\n");
165   }
166
167   /**
168    * wandelt ein Array von Strings in Grossbuchstaben um
169    *
170    * @param array
171    * @param num
172    * @return
173    */
174   public void printUpperCaseForArrayD(String[] array, int num) {
175     for (int i = 0; i < num; i++) {
176       try {
177         this.ausgabeBereich.append(this.printUpperCase(array[i]) + "\n");
178       } catch (NullStringException e) {
179         this.ausgabeBereich.append("text[" + i + "] war null!\n");
180       } catch (EmptyStringException e) {
181         this.ausgabeBereich.append("text[" + i + "] war leer!\n");
182       } catch (ArrayIndexOutOfBoundsException e) {
183         this.ausgabeBereich.append("Fehler! Falscher Arrayzugriff!\n");
184       } finally {
185         this.ausgabeBereich.append("Nach Versuch " + (i + 1) + "\n");
186       }
187     }
188     this.ausgabeBereich.append("Fertig!\n");
189   }
190 }
```

Aufgabe 12-2 **Ausnahmen und Ausnahmebehandlung** *Hausaufgabe*

Der Modulo-Operator `%` kann in Java dazu verwendet werden, den Rest einer ganzzahligen Division zu berechnen. Es gilt also z.B. `4 % 2 == 0` und `5 % 2 == 1`.

In dieser Aufgabe sollen Sie ein Programm mit einer grafischen Benutzeroberfläche implementieren, welches die Summe aller geraden Zahlen (Zahlen, die ohne Rest durch 2 teilbar sind) in einem `int`-Array ermittelt. Das `int`-Array soll durch einen modalen Dialog zu Beginn der Anwendung abgefragt und vom Benutzer eingegeben werden.

a) Die grafische Benutzeroberfläche soll wie folgt aussehen:



Es soll einen Button mit der oben angegebenen Aufschrift geben. Darunter soll der Ausgabebereich für die berechnete Summe der geraden Zahlen platziert werden.

Schreiben Sie eine Klasse `ModuloFrame`, die die Hauptklasse dieser grafischen Benutzeroberfläche sein soll und das Fenster erzeugt. Um Ihr Programm ausführen zu können, schreiben Sie eine weitere Klasse `ModuloFrameMain`, die Sie wie gewohnt im gleichen Ordner wie Ihre Klasse `ModuloFrame` abspeichern.

Lösung:
Bei der Implementierung der grafischen Benutzeroberfläche geht man wie folgt vor:

Deklarieren Sie eine Klasse `ModuloFrame`, die die Hauptklasse Ihrer grafischen Benutzeroberfläche wird und deshalb von der Klasse `JFrame` erbt. Ihre Klasse `ModuloFrame` soll zwei Attribute haben:

- ein Attribut vom Klassentyp `JButton` für den Button,
- ein Attribut vom Klassentyp `JTextArea`, das als Ausgabebereich für die spätere Rückmeldung über das Ergebnis dient.

Schreiben Sie einen Konstruktor, der den `ModuloFrame` mit einem entsprechenden Titel und Größe (wir wählen hier 500x200 Pixel) initialisiert. In dem Konstruktor sollen weiterhin alle Attribute korrekt initialisiert werden. Das Layout des `ContentPane` des `ModuloFrame`s wird auf ein `GridLayout` mit zwei Zeilen und einer Spalte initialisiert und der Button sowie der Ausgabebereich darauf platziert. Fügen Sie abschließend noch ein, dass das Programm ordnungsgemäß beendet wird, falls der `ModuloFrameFrame` geschlossen wird. Benutzen Sie dazu die Methode `setDefaultCloseOperation`.

Eine Implementierung der Klassen `ModuloFrame` und `ModuloFrameMain` für die gesamte Aufgabe finden Sie weiter unten.

b) Deklarieren Sie nun eine Methode `summeGeraderZahlen` innerhalb der Klasse `ModuloFrame`, welche die Summe aller geraden Zahlen in einem `int`-Array ermittelt. Das Array soll dabei als Parameter der Methode übergeben werden und die berechnete Summe soll den Rückgabewert der Methode bilden. Falls in der Reihung eine negative Zahl vorkommt, soll eine benutzerdefinierte und gecheckte Ausnahme `NegativeElementException` ausgelöst und die Ausführung der Methode `summeGeraderZahlen` dabei abgebrochen werden. Berücksichtigen Sie dies auch bei der Deklaration des Methodenkopfs dieser Methode. Um eine Ausnahme `NegativeElementException` verwenden zu können, müssen Sie zudem erst eine Klasse `NegativeElementException` deklarieren, die eine *Checked Exception* realisiert. Eine Fehlernachricht muss nicht in das Fehlerobjekt integriert werden.

Lösung:
Deklaration der Klasse `NegativeElementException`:

```
1 public class NegativeElementException extends Exception {
2 }
```

Deklaration der Methode `summeGeraderZahlen`:

```
1 public int summeGeraderZahlen(int[] arr)
2       throws NegativeElementException {
3   int sum = 0;
4
5   for (int i = 0; i < arr.length; i++) {
6     if (arr[i] < 0) {
7       throw new NegativeElementException();
```

```
    }
    if (arr[i] % 2 == 0) {
      sum = sum + arr[i];
    }
  }
  return sum;
}
```

Eine Implementierung der Klasse `ModuloFrame` für die gesamte Aufgabe finden Sie weiter unten.

c) Ergänzen Sie die Klasse `ModuloFrame` nun um eine Ereignisbehandlung für den Button. Wird dieser Button gedrückt, soll der Benutzer zunächst in einer Methode `summeBerechnen` mit Hilfe der Klasse `JOptionPane` nach dem `int`-Array gefragt werden, das als Eingabe für die Berechnung dient. Die Methode `summeBerechnen` muss sich daher ebenfalls um die Konvertierung des eingelesenen Strings in ein `int`-Array kümmern. Im ZIP-Archiv finden Sie die Klasse `Konverter.java`, mit der Sie in der Methode `konvertiereZuIntArray` einen Kommaseparierten String in ein `int`-Array konvertieren können (d.h. die Eingabe `1,2,3` wird konvertiert in ein Array `[1,2,3]`). Aus dem durch die Konvertierung erhaltenen `int`-Array soll mithilfe der statischen Methode `summeGeraderZahlen` aus Teilaufgabe b) die Summe der geraden Zahlen berechnet und diese anschließend im Ausgabebereich des `ModuloFrame`s ausgegeben werden. Fügen Sie `try`-, und `catch`- Blöcke in die Methode `summeBerechnen` ein, so dass, falls die Methode `summeGeraderZahlen` eine `NegativeElementException` auslöst, die Meldung „Fehler!“ ausgegeben wird.

Lösung:
Deklaration der Methode `summeBerechnen`:

```
public void summeBerechnen() {
  String einlesenArray =
      JOptionPane.showInputDialog("Array eingeben: ");
  int[] arr = Konverter.konvertiereZuIntArray(einlesenArray);

  int sum = 0;
  try {
    sum = summeGeraderZahlen(arr);
    this.ausgabeBereich.setText("Summe = " + sum);
  } catch (NegativeElementException e) {
    this.ausgabeBereich.setText("Fehler!");
  }
}
```

Eine Implementierung der Klassen `ModuloFrameMain` und `ModuloFrame` finden Sie auch im ZIP-Archiv.

```
/**
 * Diese Klasse startet den ModuloFrame
 *
 * @author Annabelle Klarl
 */
public class ModuloFrameMain {
  /**
   * Dieses Programmstueck startet das Programm.
   *
   * @param args
   */
  public static void main(String[] args) {
    ModuloFrame moduloFrame = new ModuloFrame();
    moduloFrame.setVisible(true);
  }
```

```
}
```

```
import java.awt.Container;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JTextArea;

/**
 * Diese Klasse repraesentiert das Hauptfenster der Hausaufgabe
 * zur Berechnung der Summe aller geraden Zahlen mittels Modulo.
 *
 * @author Annabelle Klarl
 */
public class ModuloFrame extends JFrame implements ActionListener {
  private JButton geradezahlenButton;

  private JTextArea ausgabeBereich;

  /** In diesem Programmstueck wird das Fenster erzeugt */
  public ModuloFrame() {
    /* In der Kopfleiste des Fenster steht "ModuloFrame" */
    this.setTitle("ModuloFrame");

    /* Das Fenster ist 500 Pixel breit und 200 Pixel hoch. */
    this.setSize(500, 200);

    /* Hier werden alle Buttons erzeugt. */
    this.geradezahlenButton =
        new JButton("Summe gerader Zahlen eines Arrays berechnen");

    /* Hier wird der Ausgabe-Bereich erzeugt. */
    this.ausgabeBereich = new JTextArea(10, 100);

    /*
     * Der ContentPane ist der Ausschnitt des Fensters, auf dem Widgets d.h.
     * Interaktionselemente (wie eine TextArea oder ein Button) platziert
     * werden koennen.
     */
    Container contentPane = this.getContentPane();
    contentPane.setLayout(new GridLayout(2, 1));
    /* Hier wird der Button platziert. */
    contentPane.add(this.geradezahlenButton);
    /* Hier wird der Ausgabebereich platziert. */
    contentPane.add(this.ausgabeBereich);

    /*
     * Hier wird der ModuloFrame als Listener fuer Knopfdruck-Ereignisse bei
     * jedem der Buttons registriert.
     */
    this.geradezahlenButton.addActionListener(this);

    /*
     * Wird das Fenster geschlossen (d.h. auf X gedrueckt), wird mit Hilfe
     * dieser Programmzeile auch unser Programm ordnungsgemaess beendet.
     */
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }

  @Override
  public void actionPerformed(ActionEvent e) {
    // je nach Button entsprechende Methode ausfuehren
```

```
64      Object source = e.getSource();
65      if (source == this.geradezahlenButton) {
66        this.summeBerechnen();
67      }
68    }
69
70    /**
71     * implementiert die Funktionalitaet
72     * des "Summe gerader Zahlen berechnen"-Buttons
73     */
74    public void summeBerechnen() {
75      String einlesenArray = JOptionPane.showInputDialog("Array eingeben: ");
76      int[] arr = Konverter.konvertiereZuIntArray(einlesenArray);
77
78      int sum = 0;
79      try {
80        sum = this.summeGeraderZahlen(arr);
81        this.ausgabeBereich.setText("Summe = " + sum);
82      } catch (NegativeElementException e) {
83        this.ausgabeBereich.setText("Fehler!");
84      }
85    }
86
87    /**
88     * berechnet die Summe aller geraden Zahlen
89     *
90     * @param arr
91     * @return Summe der geraden Zahlen
92     */
93    public int summeGeraderZahlen(int[] arr) throws NegativeElementException {
94      int sum = 0;
95
96      for (int i = 0; i < arr.length; i++) {
97        if (arr[i] < 0) {
98          throw new NegativeElementException();
99        }
100       if (arr[i] % 2 == 0) {
101         sum = sum + arr[i];
102       }
103     }
104     return sum;
105   }
106 }
```

Besprechung der Präsenzaufgaben in den Übungen vom 18.01.2019 und 21.01.2019. Abgabe der Hausaufgaben bis Mittwoch, 30.01.2019, 14:00 Uhr über UniworX (siehe Folien der ersten Zentralübung).

- *Erstellen Sie zu jeder Aufgabe Klassen mit den entsprechenden Namen, die in der Aufgabe gefordert sind.*
- *Geben Sie nur die entsprechenden* `.java`*-Dateien ab. Wir benötigen* ***nicht*** *Ihre* `.class`*-Dateien.*
- *Geben Sie Java-Code nur in* `.java`*-Dateien ab. Java-Code in Bildern, PDF-Dokumenten und Text-Dateien wird nicht korrigiert.*