Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. R. Hennicker, Dr. A. Klarl

WS 2016/17
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.

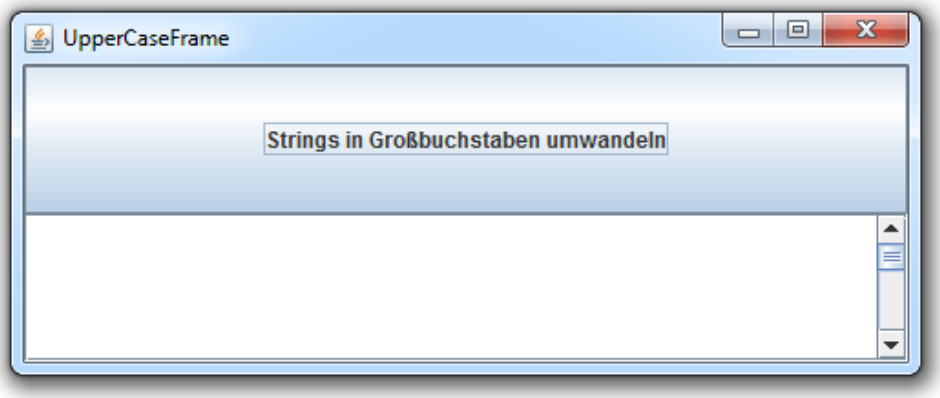
```
1 public String printUpperCase(String s) throws Exception1, Exception2 {
2   if (s == null)
3     throw new Exception1();
4   if (s.length() == 0)
5     throw new Exception2();
6
7   return s.toUpperCase();
8 }
```

Die Methode `printUpperCase` verwendet die folgenden Klassen `Exception1` und `Exception2`:

```
1 public class Exception1 extends Exception {
2 }
3
4 public class Exception2 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:

```
JScrollPane scrollPane = new JScrollPane(ausgabeBereich);
scrollPane.setHorizontalScrollBarPolicy(
        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`.

Auf der Webseite finden Sie die Implementierung der Klassen `UpperCaseFrame` und `UpperCaseFrameMain`.

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.

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

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 `Exception1` und `Exception2` um *Checked Exceptions* handelt, die abgefangen werden müssen.

Implementierung der Methode `umwandelnStrings`:

```
1 private void umwandelnStrings() {
2   String einlesenAnzahlStrings = JOptionPane
3       .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("Exception bei Versuch "
7            + (i + 1) + "\n");
8      }
9    }
10   this.ausgabeBereich.append("Fertig!\n");
11 }
```

Auf der Webseite finden Sie die Implementierung der Klasse `UpperCaseFrame`. 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      }
6      catch (Exception1 e) {
7        this.ausgabeBereich.append("text[" + i
8            + "] war null!\n");
9      }
10     catch (Exception2 e) {
11       this.ausgabeBereich.append("text[" + i
12           + "] war leer!\n");
13     }
14   }
15   this.ausgabeBereich.append("Fertig!\n");
16 }
```

Auf der Webseite finden Sie die Implementierung der Klasse `UpperCaseFrame`. 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 (Exception1 e) {
6        this.ausgabeBereich.append("text[" + i
7            + "] war null!\n");
8      } catch (Exception2 e) {
9        this.ausgabeBereich.append("text[" + i
10           + "] war leer!\n");
11     } catch (ArrayIndexOutOfBoundsException e) {
12       this.ausgabeBereich
13           .append("Fehler! Falscher Arrayzugriff!\n");
14     } finally {
15       this.ausgabeBereich.append("Nach Versuch "
16           + (i + 1) + "\n");
17     }
18   }
19   this.ausgabeBereich.append("Fertig!\n");
20 }
```

Auf der Webseite finden Sie die Implementierung der Klasse `UpperCaseFrame`. 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 = JOptionPane
3          .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    }
10   else {
11     this.ausgabeBereich.setText("Ungültige Anzahl " + num);
12   }
13 }
```

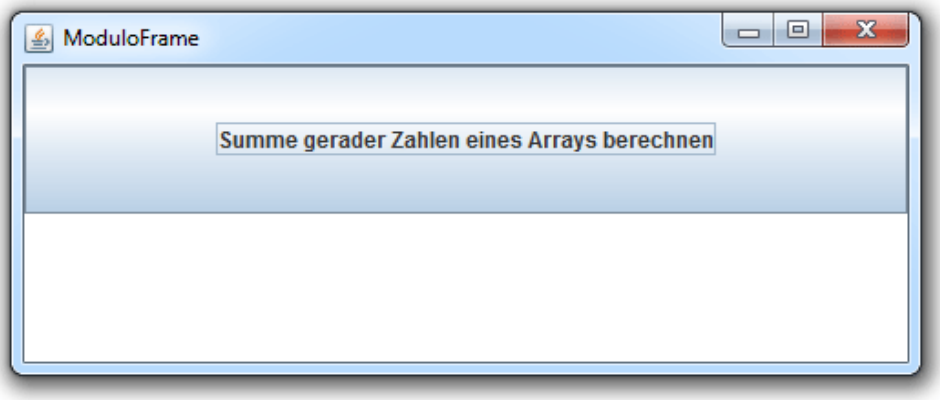
Auf der Webseite finden Sie die Implementierung der Klasse `UpperCaseFrame`. Die entsprechende Methode heißt hier `umwandelnStringsE`.

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 `ModuloFrames` 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`.

Auf der Webseite finden Sie die Implementierung der Klassen `ModuloFrame` und `ModuloFrameMain`.

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();
8      if (arr[i] % 2 == 0)
9        sum = sum + arr[i];
10   }
11   return sum;
12 }
```

Auf der Webseite finden Sie die Implementierung der Klassen `NegativeElementException` und `ModuloFrame`.

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. Auf der Vorlesungswebseite finden Sie die Klasse `Konverter.java`, mit der Sie in der Methode `konvertiereZuIntArray` einen Komma-separierten 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`:

```
1  public void summeBerechnen() {
2    String einlesenArray = JOptionPane
3        .showInputDialog("Array eingeben: ");
4    int[] arr = Konverter
5        .konvertiereZuIntArray(einlesenArray);
6
7    int sum = 0;
8    try {
9      sum = summeGeraderZahlen(arr);
10     this.ausgabeBereich.setText("Summe = " + sum);
11   } catch (NegativeElementException e) {
12     this.ausgabeBereich.setText("Fehler!");
13   }
14 }
```

Auf der Webseite finden Sie die Implementierung der Klasse `ModuloFrame`.

Aufgabe 12-3 **Klausurvorbereitung 6 und 9 ECTS** Hausaufgabe

Überlegen und formulieren Sie Fragen zu Themen aus der Vorlesung, bei denen Sie noch Probleme haben. Die am meisten gestellten Fragen werden in den Tutorien 01.02.2017 bis 08.02.2017 behandelt. Bitte geben Sie auch diese Hausaufgabe per UniWorX ab!

Besprechung der Präsenzaufgaben in den Übungen vom 18.01.2017 bis zum 25.01.2017. Abgabe der Hausaufgaben bis Mittwoch, 01.02.2017, 14:00 Uhr über UniworX (siehe Folien der ersten Zentralübung). Erstellen Sie zu jeder Aufgabe Klassen, die die Namen tragen, die in der Aufgabe gefordert sind. Geben Sie nur die entsprechenden `.java`*-Dateien ab. Wir benötigen* ***nicht*** *Ihre* `.class`*-Dateien.*