1. Aufgabe

Lernziele

Erfassung von Daten, Prüfen der Korrektheit, Konvertierung von Strings in Zahlen, Fehlerbehandlung

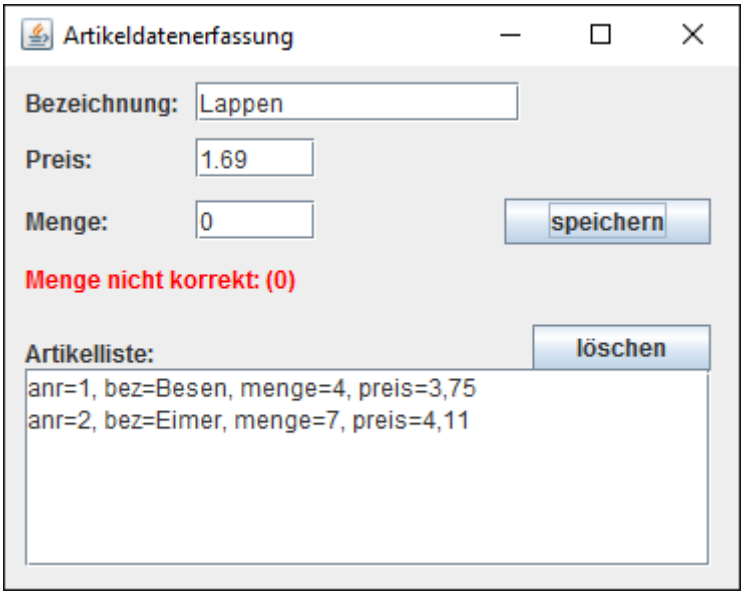
Aufgabenstellung

Erstellen Sie ein Programm zur Erfassung von Artikeldaten. Dazu gehören eine Artikelnummer, eine Bezeichnung, ein Preis und eine Menge. Die Artikelnummer wird fortlaufend gezählt, die Bezeichnung muss mindestens 3-stellig sein, der Preis und die Menge größer 0. Entwerfen Sie eine entsprechende GUI mit Eingabefeldern 2 Buttons und einer TextArea. Die Buttons haben folgende Funktionen:

Speichern: Liegen keine Fehlereingabefehler wird ein Artikel erfasst und in der TextArea dargestellt mit allen Attributen durch Kommas getrennt. Die Eingabefelder werden zurückgesetzt. Ansonsten erfolgen Fehlermeldungen.
Löschen: TextArea und Artikelnummernvergabe werden zurückgesetzt.

Überlegen Sie sich ein sinnvolles Design von Oberfläche und Programmstruktur. Verwenden Sie diesmal eine JTextArea (Hinweis: diese muss in den Viewport einer JScrollpane platziert werden, um scrollable zu sein).

GUI-Skizze:



2. Aufgabe

Lernziele

Konvertierung von Strings in numerische Datentypen, Auto-(un)boxing, Fehlerbehandlung.

Aufgabenstellung

a) Erstellen Sie einen Integer-Taschenrechner mit den Grundoperationen +, -, *, /. Es gibt zwei Textfelder zur Eingabe der Argumente. Die Ausgabe erfolgt über ein weiteres Textfeld(nur editierbar). Die jeweilige Operationsart kann über ein Listfeld gewählt werden (Default ist +). Bei Eingabefehlern oder Berechnungsfehlern sollen aussagekräftige Fehlermeldungen erfolgen. Erstellen Sie für die Durchführung der Operationen eine Funktion

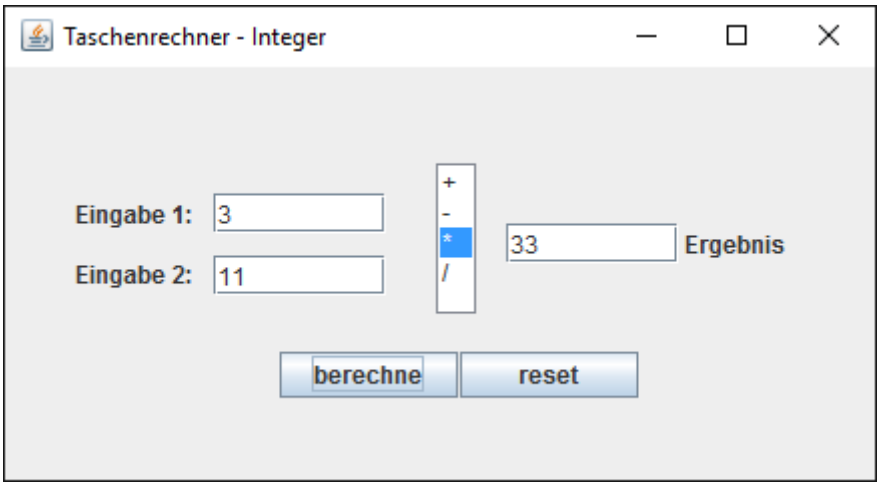
*„**public int** berechneInt(TextField arg1, TextField arg2, String operation)“*

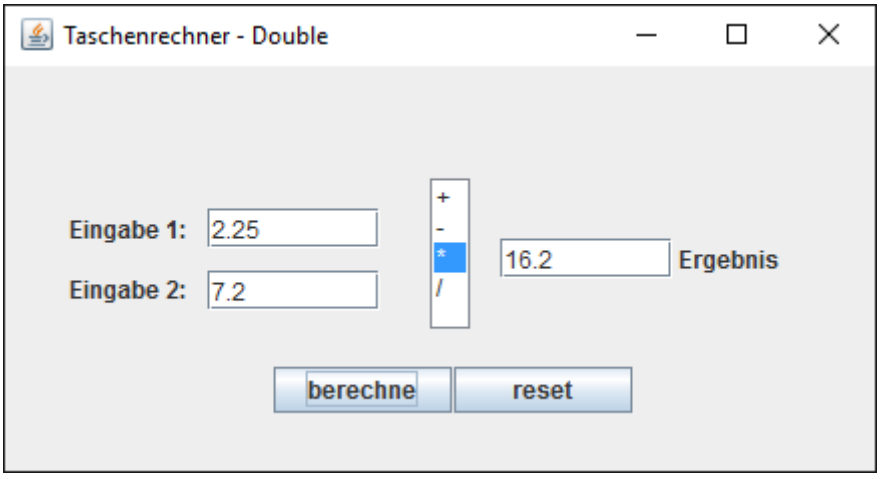
welche bei Verwendung/Aufruf eine Fehlerbehandlung erzwingt.

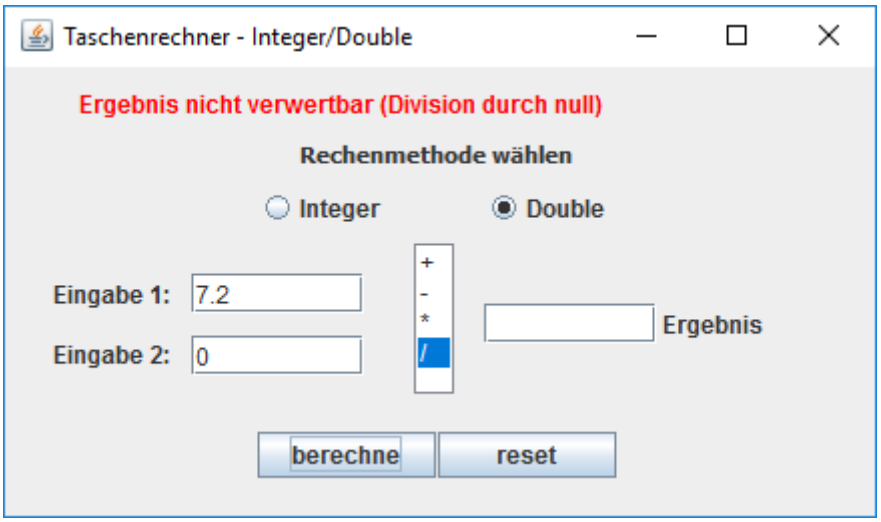
b) Erstellen Sie zusätzlich einen vergleichbaren Double-Rechner. Beide Rechner sollen auch in der Lage sein die Division durch Null bei Integer bzw. das Phänomen des Ergebnisses „INFINITY“ bei Double zu behandeln.

c) Erstellen Sie einen integrierten Rechner, bei dem Sie wählen können ob Sie mit Integer oder Double arbeiten wollen.

GUI-Skizzen:






Taschenrechner - Integer/Double
Ergebnis nicht verwertbar (Division durch null)
Rechenmethode wählen
Integer
Double
Eingabe 1:
7.2
Eingabe 2:
0
+
-
*
/
Ergebnis
berechne
reset

3. Aufgabe

Lernziel

Konvertierung von Strings in numerische Datentypen, Fehlerbehandlung, grafische Oberfläche.

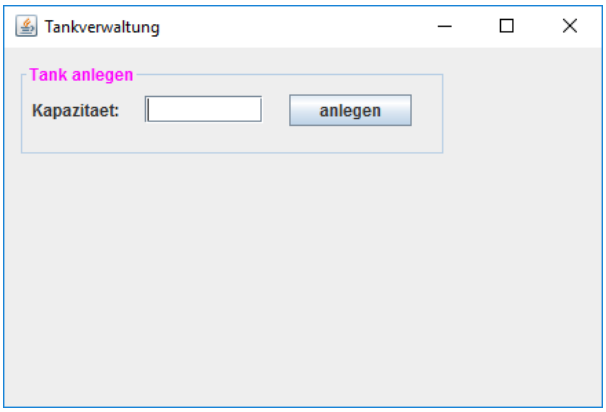
Aufgabenstellung

Erstellen Sie eine Fachkonzeptklasse **Tank**, die folgendes leistet:

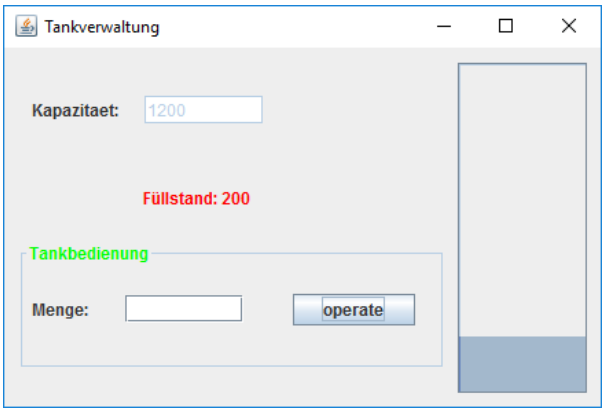
/1/	Ein Tank besitzt eine Soll-Füllhöhe (max. Kapazität) und eine Ist-Füllhöhe (in Litern).
/2/	Ein Tank kann um eine bestimmte Menge gefüllt werden.
/3/	Wird beim Füllen die Soll-Füllhöhe überschritten, erzeugt der Tank eine Tank-Exception in der auch die Überschussmenge enthalten ist.
/4/	Ein Tank kann um eine bestimmte Menge geleert werden.
/5/	Wird beim Leeren die Höhe 0 unterschritten, erzeugt der Tank eine Tank-Exception in der auch die Fehlmenge enthalten ist.

Schreiben Sie ein entsprechendes GUI-Programm um einen Tank zu betreiben. Es soll möglich sein, bestimmte Mengen einzufüllen/zu entnehmen und die jeweilige Ist-Füllhöhe anzuzeigen (numerisch und mit Progress-Bar). Unter- / Überschreitungen werden als Statusmeldung ausgegeben. Der aktuelle Füllstand soll durch eine „Progress-Bar“ reflektiert werden.
Hinweis: Die Soll-Füllhöhe des Tanks soll durch eine einmal ausführbare Funktion gesetzt werden (die danach disabled/unerreichbar wird). Dadurch werden dann die Tank-Operationen „füllen“ und „leeren“ freigegeben bzw enabled/sichtbar. Verwenden Sie JPanels für die einzelnen Bereiche. JPanels können sichtbar/unsichtbar gemacht werden und z.B. auch einen beschrifteten Rahmen haben (ist über die WindowBuilder-Properties parametriesierbar).

GUI-Skizzen:



Phase 1: der Tank muss angelegt werden



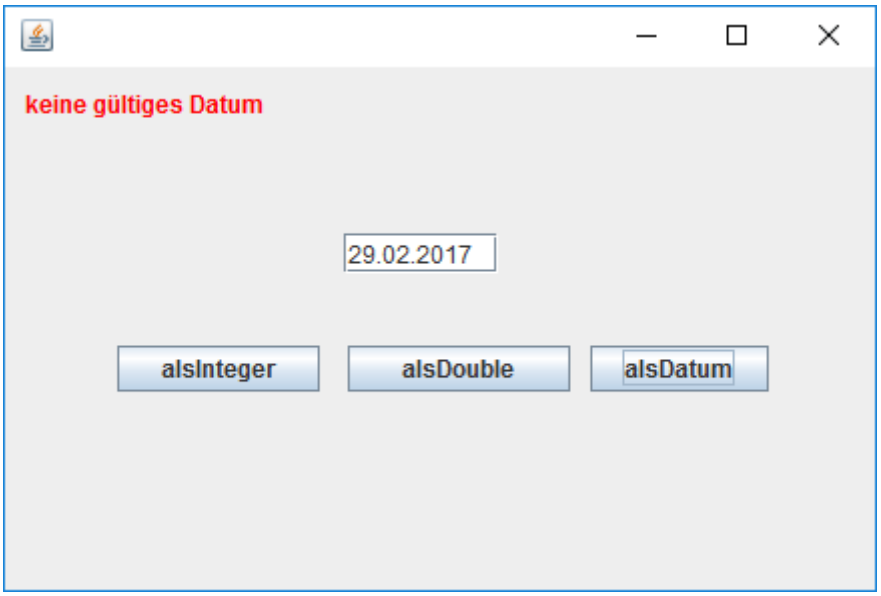
Phase 2: der Tank kann bedient werden

4. Aufgabe

Erstellen Sie eine eigene Klasse Textfeld als Unterklasse von `JTextField`. Diese Klasse soll alle Eigenschaften der Swing-Klasse besitzen und zusätzlich die folgenden Funktionen/Schnittstelle bereitstellen:

```
public Integer intZahl() ;
public Double doubleZahl() ;
public LocalDate datum() ;
```

Bei Aufruf der Funktionen soll versucht werden, den Inhalt des Textfelds gemäß des Rückgabewerts der jeweiligen Funktion zu interpretieren, um ein geeignetes Objekt zurück zu geben. Bei Fehlern soll ein Objekt der Fehlerklasse *KonvertError* (mit entsprechender Fehlerbenachrichtigung) ausgeworfen werden.
Erstellen sie ein GUI-Programm mit einem Textfeld und drei Buttons, um diese Funktionen zu testen und die Resultate geeignet zu dokumentieren/darzustellen.

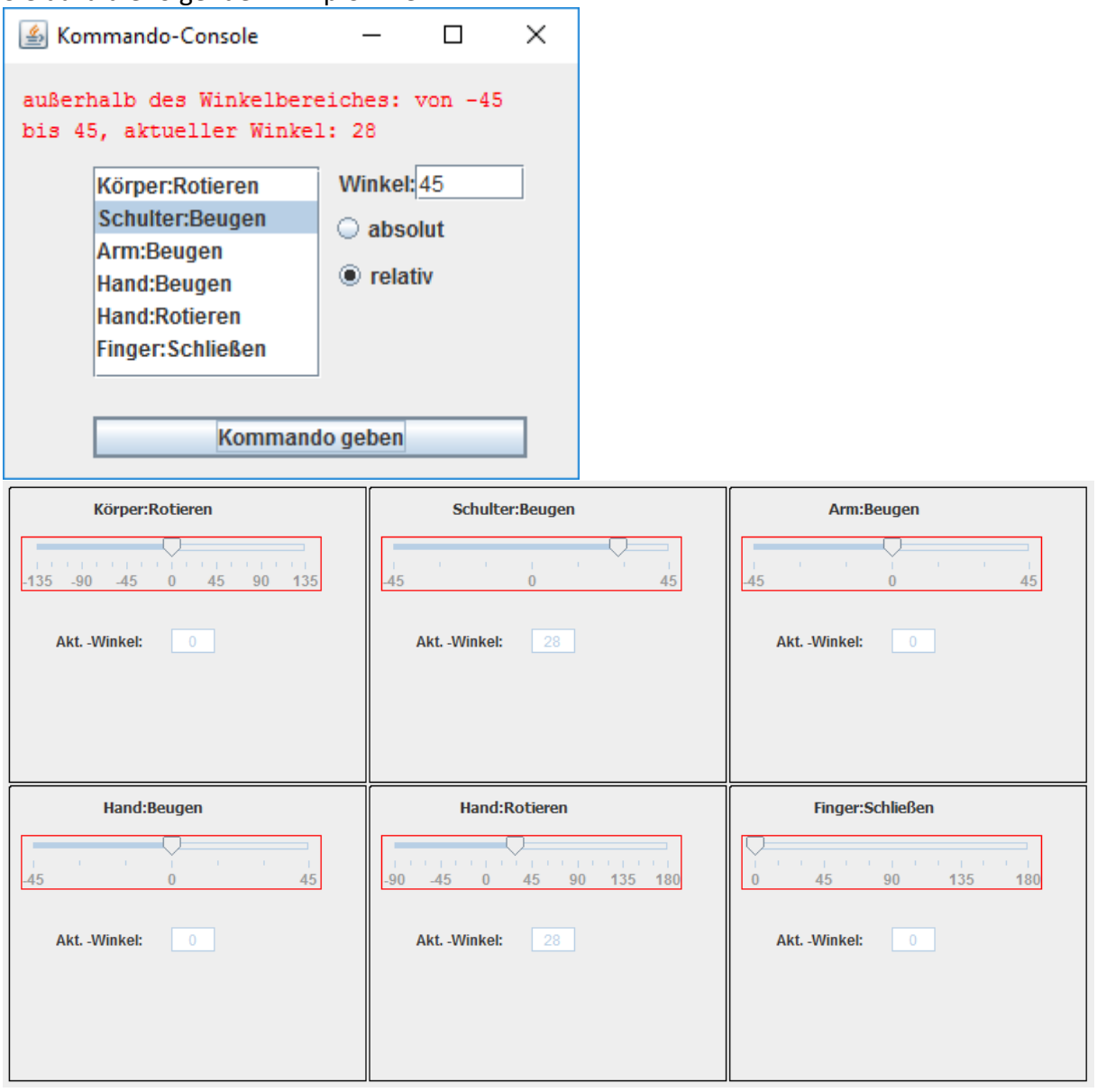


5. Aufgabe

Modifizieren Sie nun die aus der Übung 1 bekannte Roboteraufgabe. Dazu sollen zwei grafische Oberflächen erstellt werden:

- Eine zur grafisch/interaktiven Auswahl/Eingabe der Kommandos
- Eine weitere zur Darstellung des aktuellen Status der Objekte bzw. möglichen Aktionen
- Beim Start des Programms werden beide Oberflächen gleichzeitig angezeigt. Das Programm kann nur über die Kommandokonsole beendet werden. Statusmeldungen werden über die Kommandokonsole angezeigt.

Sie dazu die folgende Prinzip-Skizze:



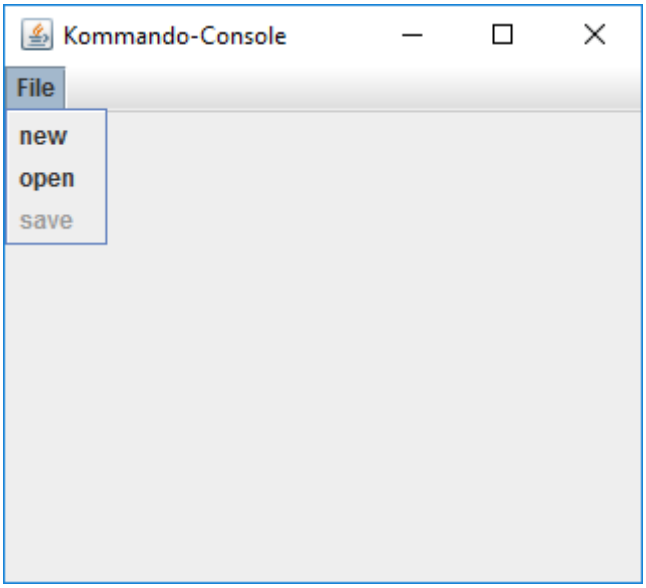
Das Programm dazu (SwingRoboter.jar) ist zur Anschauung über den Lehre-Ordner aufrufbar.

6. Aufgabe

Modifizieren Sie die Aufgabe 5 folgendermaßen:

- Die Kommando-Konsole wird durch ein Menü „File“ mit den Items „new“, „open“ und „save“ ergänzt.
- Beim Start des Programms wird nur die Kommando-Konsole mit Menü, aber nicht der Inhalt der ContentPane angezeigt. Die Menü-Items „new“ und „open“ sind enabled, „save“ ist disabled.
- Wird „new“ gewählt startet das Programm wie bisher – die ContenPane der Kommando-Konsole und das Statusfenster werden sichtbar, alle Winkel sind bei 0. Wird „open“ gewählt, können über einen File-Chooser die Kommando-Objekte eine alten Roboterstatus eingelesen, angezeigt und damit weiter gearbeitet werden. In beiden Fällen werden dann die Items „new“ und „open“ disabled und „save“ enabled.
- Mit „save“, können die aktuellen Kommando-Objekte abgespeichert werden.

Startkonfiguration:



Das vollständige Programm dazu (SwingRoboterIO.jar) ist zur Anschauung über den Lehre-Ordner aufrufbar.

Fragen zum Wiederholen/ Selbststudium/ Ausprobieren

try … catch

1. Aufgabe

Welches Ergebnis erhält man, wenn man folgendes Programm kompiliert und startet?

```
public class Fehler {
     public static void main(String[] args) {
          try {
               throw new ArithmeticException();
          } catch (ArithmeticException rechenfehler) {
               System.out.println("Rechenfehler");
          } catch (Exception fehler) {
               System.out.println("Fehler");
          } finally {
               System.out.println("finally");
          }
          System.out.println("Ende");
     }
}
```

a) Rechenfehler Fehler finally
b) Rechenfehler Fehler finally Ende
c) Rechenfehler finally Ende
d) Fehler finally Ende
e) finally Ende
f) Kompilierfehler
g) Laufzeitfehler

2. Aufgabe

Welches Ergebnis erhält man, wenn man folgendes Programm kompiliert und startet?

```
public class Fehler {
     public static void main(String[] args) {
          try {
               throw new ClassCastException();
          } catch (ArithmeticException rechenfehler) {
               System.out.println("Rechenfehler");
          } catch (Exception fehler) {
               System.out.println("Fehler");
          } finally {
               System.out.println("finally");
          }
          System.out.println("Ende");
     }
}
```

a) Rechenfehler Fehler finally
b) Rechenfehler Fehler finally Ende
c) Rechenfehler finally Ende
d) Fehler finally Ende
e) finally Ende
f) Kompilierfehler
g) Laufzeitfehler

3. Aufgabe

Welches Ergebnis erhält man, wenn man folgendes Programm kompiliert und startet?

```
public class Fehler {
     public static void main(String[] args) {
          try {
               throw new ClassCastException();
          } catch (ClassCastException fehler) {
               System.out.println("Fehler");
               throw new ArithmeticException();
          } catch (ArithmeticException rechenfehler) {
               System.out.println("Rechenfehler");
          } finally {
               System.out.println("finally");
          }
          System.out.println("Ende");
     }
}
```

a) Rechenfehler Fehler finally
b) Rechenfehler Fehler finally Ende
c) Rechenfehler finally Ende
d) Fehler finally
e) finally Ende
f) Kompilierfehler
g) Laufzeitfehler

4. Aufgabe

Welches Ergebnis erhält man, wenn man folgendes Programm kompiliert und startet?

```
public class Fehler {
     public static void main(String[] args) throws
                                        FileNotFoundException {
          try {
               throw new FileNotFoundException();
          } catch (ArithmeticException dokumentfehler) {
               System.out.println("fangen");
          } catch (ClassCastException fehler) {
               System.out.println("nochmals fangen");
          } finally {
               System.out.println("Restarbeit");
          }
          System.out.println("Ende");
     }
}
```

a) werfen nochmals fangen Restarbeit Ende
b) nochmals fangen Restarbeit Ende
c) fangen Restarbeit Ende
d) Kompilierfehler
e) Laufzeitfehler
f) Keine dieser Möglichkeiten.

5. Aufgabe

Welches Ergebnis erhält man, wenn man folgendes Programm kompiliert und startet?

```
public class Fehler {
     public static void main(String[] args) {
          try {
               Fehler fehler = new Fehler();
               fehler.ausnahme();
          } catch (Exception fehler) {
               System.out.println("Fehler!");
          } catch (FileNotFoundException fnf) {
               System.out.println("Ich fange!");
          }
     }
     public void ausnahme() throws FileNotFoundException {
          throw new FileNotFoundException();
     }
}
```

a) Ich fange!
b) Fehler!
c) Kompilierfehler
d) Laufzeitfehler
e) Keine dieser Möglichkeiten.