

# Übung 8

## Exceptions und Dateien in JAVA

1. Erzeugen Sie eine Java-Klasse **Modulo** mit einer statischen Methode/Funktion

**int modulo(int dividend, int divisor),**

die den Teilungsrest bei der ganzzahligen Division  $\frac{\text{dividend}}{\text{divisor}}$  an den Aufrufer zurückgeben soll. Dabei soll der Fall, dass bei der Berechnung ein **ArithmeticException**-Fehler auftritt (weil beispielsweise der **divisor** gleich 0 ist), abgefangen werden und durch eine entsprechende Fehlermeldung sowie die Rückgabe des **dividend** an den Aufrufer quittiert werden.

```
public class Modulo {

    static int modulo(int dividend, int divisor) {
        try {
            return dividend % divisor;
        } catch (ArithmeticException e) {
            System.out.println("Durch 0 darf man nicht teilen.");
            e.printStackTrace();
            // oder System.err.println(e.getMessage());
        }
        return dividend;
    }

    public static void main(String[] args) {
        int zahl1 = 1;
        int zahl2 = 0;
        int ergebnis = modulo(zahl1, zahl2);
        System.out.println(zahl1 + " MODULO " + zahl2
            + " = " + ergebnis);
    }
}
```

2. Erzeugen Sie eine Java-Klasse **Alarm**, die im Konstruktor eine (beliebige) Exception auslöst, wenn beim Erzeugen eines Objektes dieser Klasse im Konstruktor der String **"ein"** übergeben wird.

```
public class Alarm {  
  
    // Alle Funktionen, die eine Ausnahmebehandlung erzeugen können,  
    // müssen als "throws Exception" deklariert werden!  
    public Alarm(String s) throws Exception {  
        if(s.equals("ein")) throw new Exception("Alarm aktiviert!");  
    }  
  
}
```

**Alarm.java** enthält außerdem ein **public static void main(String[] args)**, das den Konstruktor mit einigen Test-Eingaben aufruft.

3. Kombinieren Sie die Funktionsweise von **DateiLesen.java** und **DateiSchreiben.java** in einem Programm **FileCopy.java**, das den Inhalt einer Datei vollständig in eine andere Datei kopiert. Die Namen von Quell- und Zielfeld sollen als Kommandozeilenparameter **args[0]** und **args[1]** in **main(String[] args)** übergeben werden (also Aufruf:  
**java FileCopy Quelldatei Zieldatei**  
). Fangen Sie dabei möglicherweise auftretende Fehler wie in den Beispielen mit einer entsprechenden Meldung ab.

Siehe **FileCopy.java**.

4. Zusatzaufgabe (optional): Ergänzen Sie in der Aufgabe von letzter Woche die angegebene **String/int** Umwandlungsfunktion in **Addieren.java** so, dass Eingabezeichenketten, die keiner Zahl entsprechen, mit einer entsprechenden Fehlermeldung quittiert werden.

Siehe **Addieren.java** in diesem Verzeichnis. Der relevante Teil ist dieser:

```
// Achtung: Das Umwandeln von String nach int kann schiefgehen.  
// Wir fangen dies hier mit einer Exception-Behandlung ab  
String result = "";  
try {  
    result = new String("Ergebnis: " +  
        ( Integer.valueOf(zahl1.getText()).intValue() +  
          Integer.valueOf(zahl2.getText()).intValue() ) );  
}  
catch(Exception e) {  
    result = new String("Fehler: " + e.toString());  
}
```

Im Falle eines Fehlers enthält die Ergebnis-Variable (jetzt String statt int) eine Fehlermeldung, und wird anstelle der Summe im Ergebnisfenster als Button-Text angezeigt.