Java-Grundlagenschulung
Tag 5

SHD Einzelhandelssoftware
GmbH & Co. KG
SHD

Agenda

- Exceptions
- Aufgabe 1 - Die Exceptions in der Praxis
- Dateiverarbeitung
- Aufgabe 2 - Dateiverarbeitung in der Praxis
- Aufgabe 3 - Stall Persistieren
- Das DAO Pattern
- Aufgabe 4 - Schreiben einer DAO Klasse
- Hausaufgabe

Exceptions

- Beschreiben Ausnahmen / Fehler im Programmcode
- Viele verschiedene Arten z.B.: NumberFormatException
- Mithilfe von try-catch Fehler gezielt behandeln
  (mehrere catch-Blöcke möglich, von detailliert zu grob)
- mit „throw“ Fehler werfen.
- Mit „throws“ in der Methodensignatur die Exception weiterwerfen
- Mit „finally“ können wir die „Exceptionprobleme“ aufräumen

Syntax:

```
try
{
    //code
}
catch(Exception e)
{
    //das wird im
    //Fehlerfall ausgeführt
}
finally
{
    //das wird immer!
    //ausgeführt
}
```

Aufgabe 1 - Die Exceptions in der Praxis

1. Programmiere folgende mathematische Berechnung
   55 / (33 - (11 * 3))

2. Fange den Fehler mit einem try-catch Block ab und gebe den Fehler auf der Konsole aus

3. Werfe anschließend von Hand im catch Block eine (new) RuntimeException (*throw*)

4. Schreibe eine Methode die eine Exception mit throws in der Methodensignatur an den Aufrufer weiterwirft, sodass diese beim Aufruf ebenfalls mit einem try-catch Block abgefangen werden muss

Dateiverarbeitung

File

- Die Klasse File repräsentiert eine Datei oder einen Ordner.
- Die Klasse File unterscheidet mit *isDirectory(), isFile()* zwischen Ordner und Dateien.
- Unterscheidung zwischen relativen und absoluten Pfaden beachten

PrintWriter

- Hiermit lassen sich Dateien befüllen.

FileReader in Kombination mit dem BufferdReader

- Hiermit ist es möglich Dateien auszulesen.

Aufgabe 2 - Dateiverarbeitung in der Praxis

1. Erstelle ein Objekt vom Typ File und rufe die Methode createNewFile() auf. Benutze dazu einmal einen absoluten und einmal einen relativen Pfad.

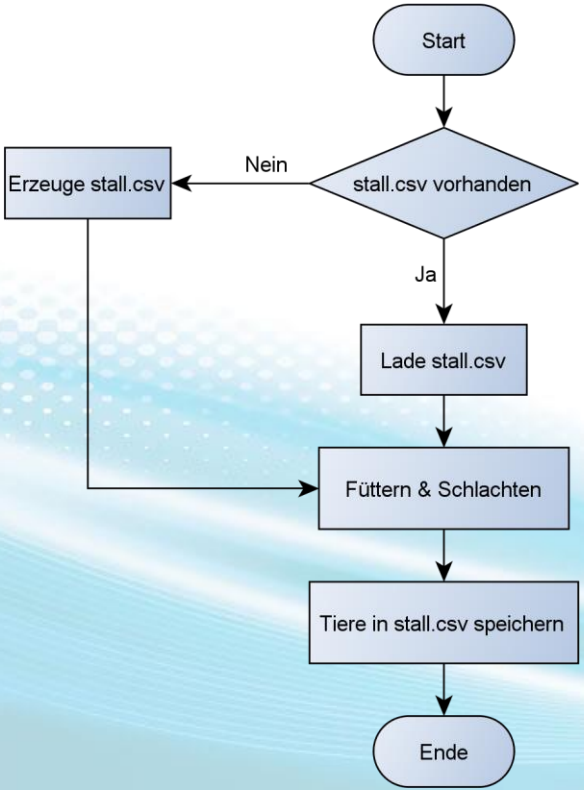
2. Verwende den PrintWriter um einen beliebigen Text in eine Datei zu schreiben.

3. Lese den Inhalt der eben erstellten Datei aus (Buffered Reader & FileReader) und gebe dir diesen auf der Konsole aus.
   => Verwende hierfür einmal die while-/ und for-Schleife

4. Schließe die Reader in dem „finally-Block“.

5. Wandle das try-catch-finally in ein try with resources (siehe internet).

Aufgabe 3 - Stall Persistieren



stall.csv

```
Name;Gewicht;Tierart
Berta;550;Kuh
Paula;320;Schwein
```

Tipp:
String zeile = „Berta1;552.36;stalltiere.Kuh“
String[] zeilenSplit = zeile.split(„;“);
String name = zeilenSplit[0];
int gewicht = Integer.parseValue(zeilenSplit[1])
String tierart = zeilenSplit[2];

Für fortgeschrittene:
stalltier = (Stalltier) Class.forName(zeilenSplit[2]).newInstance();

Das DAO Pattern

- DAO (Data Access Object)
- Eine zusätzliche Schicht im Programm welches der Persistierung dient.

- Diese Schicht enthält die komplette Logik zum speichern / persistieren.

Aufgabe 4 - Schreiben einer DAO Klasse

1. Erstelle eine neue Klasse „StallDAO“ in dem Verzeichnis „dao“.

2. Lagere die komplette Dateiverarbeitung in diese Klasse aus, sodass du zum Laden & Speichern nur einen Aufruf brauchst.

Hausaufgabe

Anstatt den Stall in eine CSV Datei zu speichern soll jetzt eine XML/JSON Datei verwendet werden.

1. Recherchiere im Internet wie man XML/JSON Code in Java generieren kann.

2. Speichere den Stall anstatt in einer CSV Datei jetzt in einer XML/JSON Datei.

3. *Für die Fleißigen*: Schreibe ein kleines Tool welches die .csv Datei in eine .xml / .json Datei konvertiert.