

---

SWE 3 – Übung 05

---

Aufwand in h: 12 h

**Beispiel 1: Flugreisen**

## Lösungsidee

Es gibt die drei Klassen *flight*, *person* und *voyage*. Verbunden sind sie miteinander über die Klasse *voyage*, welche hauptsächlich aus einer Sammlung von Personen, die Teil der Flugreise sind, und Flügen, die Teil der Flugreise sind, aufgebaut ist. Prinzipiell ist das Konzept erlaubt, dass eine Flugreise nur aus einem Flug besteht, in den Beispielen konzentriert man sich jedoch mehr auf Flugreisen mit mehreren Flügen.

Ein Objekt der Klasse *Person* besteht aus Vorname, Nachname, Geschlecht, Alter, Adresse und Kreditkartennummer. Für das Geschlecht wird ein eigener Datentyp *gender\_type* eingeführt, welcher die drei Optionen „male“, „female“ und „diverse“ zur Verfügung stellt. Da eine Adresse aus vielen kleineren Teilen besteht, wird sie ebenfalls als eigener Datentyp *adress\_type* erfasst mit den Eigenschaften „streetname“, „streetnumber“, „postcode“, „cityname“ und „country“.

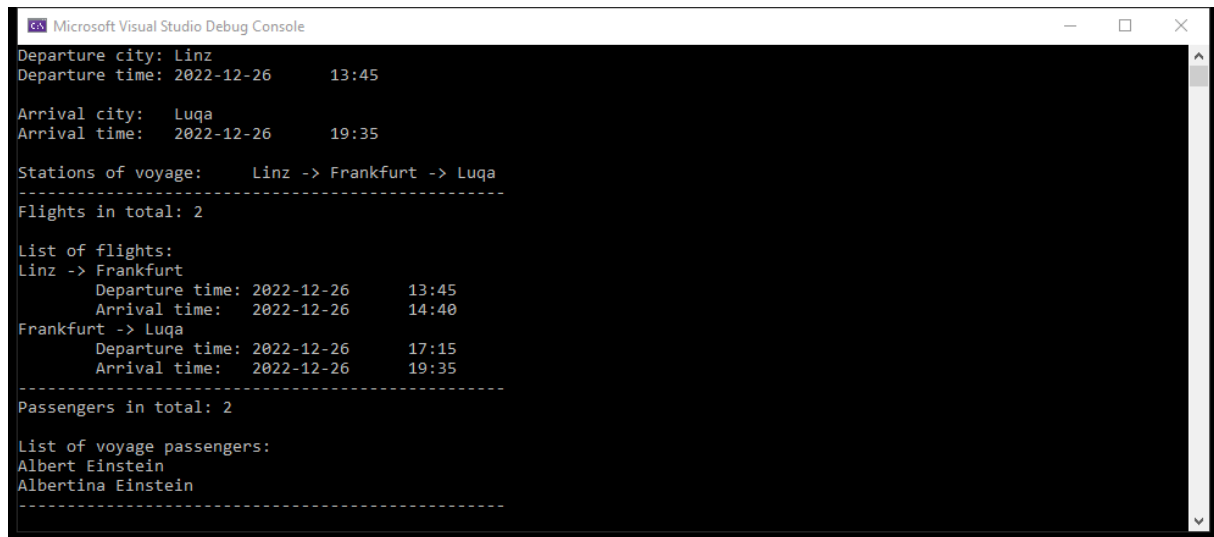
Ein Objekt der Klasse *Flug* bzw. *flight* besteht aus Flugnummer, Airline, Abflugsort, Ankunftsort, Abflugzeit, Ankunftszeit und Flugdauer. Für die Abflug- und Ankunftszeit wird ein eigener Datentyp *date\_type* mit den Komponenten „year“, „month“, „day“ und „time“ eingeführt, um den genauen Zeitpunkt mittels Datum und Uhrzeit festlegen zu können. Die Uhrzeit in *date\_type* wiederum beziehungsweise die Flugdauer werden in einem eigenen Datentyp *time\_type* erfasst, welcher aus „hour“ und „minute“ besteht. Ob die beiden Einträge „Abflugzeit“ und „Ankunftszeit“ zusammenpassen mit dem Eintrag der Flugdauer, wird im Programm nicht überprüft. Stattdessen wird davon ausgegangen, dass die eingetragenen Daten stimmig sind.

Die Klasse *Flugreise* bzw. *voyage* enthält nun sowohl Objekte der Klasse *Person* als auch *Flug*. Eine Flugreise besteht aus Datenkomponenten wie Abflugort, Abflugzeitpunkt, Ankunftsort, Ankunftszeitpunkt, Flüge (Vektor von Flügen), Anzahl der Flüge (errechnet aus der Summe eingetragener Flüge), Passagiere (Vektor von Personen) und Anzahl der Passagiere (errechnet aus der Summe erfasster Personen). Die Flüge werden direkt bei der Eintragung in ihrer zeitlichen Reihenfolge erfasst, damit das spätere Printen einer Flugreise erleichtert werden kann. Das bedeutet, dass beispielsweise eine Reise Linz→Frankfurt→Denver die Flüge in der Reihenfolge Linz→Frankfurt und Frankfurt→Denver erfasst haben muss, um ein stimmiges Ausgeben der Reise zu ermöglichen. Diese Art der Eintragung ist also ebenfalls eine Voraussetzung an User\*in, um eine korrekte Arbeitsweise des Programms versichern zu können.

Durch das Überladen des „<<“ - Operators wird eine gesammelte Ausgabe der Flugreise ermöglicht.

## Testfälle

### Test 1: Zwei Flüge und zwei Personen sind Teil einer Flugreise



```

Microsoft Visual Studio Debug Console

Departure city: Linz
Departure time: 2022-12-26      13:45

Arrival city:   Luqa
Arrival time:   2022-12-26      19:35

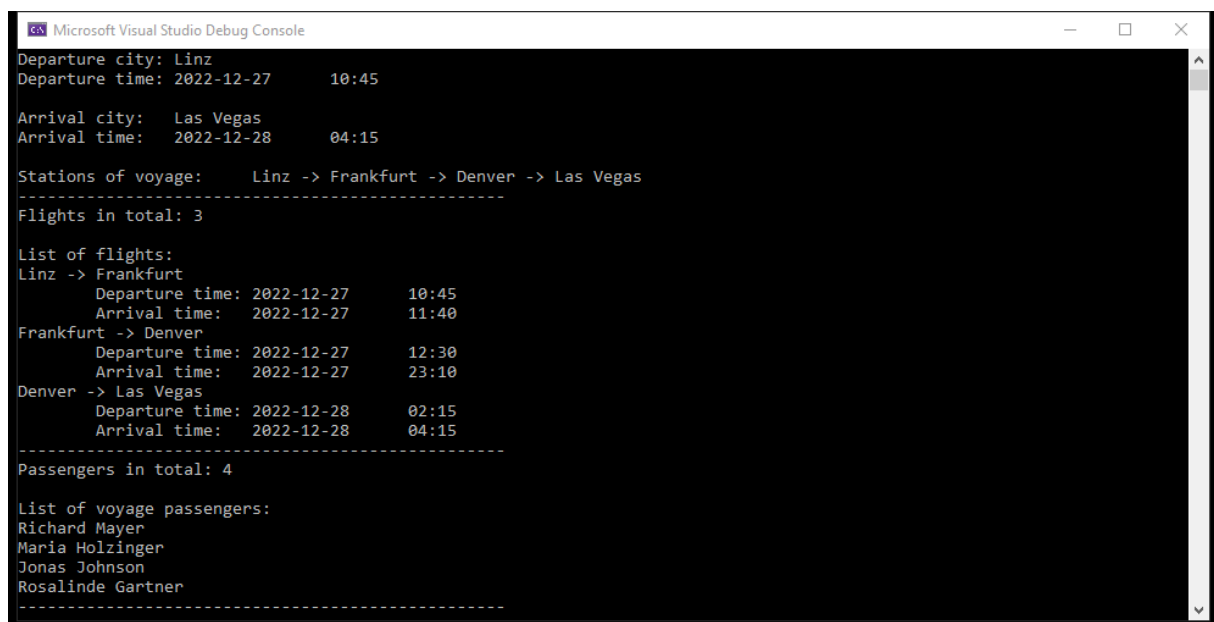
Stations of voyage:   Linz -> Frankfurt -> Luqa
-----
Flights in total: 2

List of flights:
Linz -> Frankfurt
  Departure time: 2022-12-26      13:45
  Arrival time:   2022-12-26      14:40
Frankfurt -> Luqa
  Departure time: 2022-12-26      17:15
  Arrival time:   2022-12-26      19:35
-----
Passengers in total: 2

List of voyage passengers:
Albert Einstein
Albertina Einstein
-----

```

### Test 2: Eine Flugreise mit drei Flügen (Linz → Frankfurt → Denver → Las Vegas)



```

Microsoft Visual Studio Debug Console

Departure city: Linz
Departure time: 2022-12-27      10:45

Arrival city:   Las Vegas
Arrival time:   2022-12-28      04:15

Stations of voyage:   Linz -> Frankfurt -> Denver -> Las Vegas
-----
Flights in total: 3

List of flights:
Linz -> Frankfurt
  Departure time: 2022-12-27      10:45
  Arrival time:   2022-12-27      11:40
Frankfurt -> Denver
  Departure time: 2022-12-27      12:30
  Arrival time:   2022-12-27      23:10
Denver -> Las Vegas
  Departure time: 2022-12-28      02:15
  Arrival time:   2022-12-28      04:15
-----
Passengers in total: 4

List of voyage passengers:
Richard Mayer
Maria Holzinger
Jonas Johnson
Rosalinde Gartner
-----

```

### Test 3: Eine Flugreise mit drei Flügen (Las Vegas → San Francisco → München → Linz)

```
Microsoft Visual Studio Debug Console

Departure city: Las Vegas
Departure time: 2022-12-26      03:00

Arrival city:   Linz
Arrival time:   2022-12-26      23:20

Stations of voyage:   Las Vegas -> San Francisco -> Muenchen -> Linz
-----
Flights in total: 3

List of flights:
Las Vegas -> San Francisco
    Departure time: 2022-12-26      03:00
    Arrival time:   2022-12-26      04:40
San Francisco -> Muenchen
    Departure time: 2022-12-26      06:30
    Arrival time:   2022-12-26      17:40
Muenchen -> Linz
    Departure time: 2022-12-26      20:15
    Arrival time:   2022-12-26      23:20
-----
Passengers in total: 4

List of voyage passengers:
Lisa Haslinger
Lorentz Bauer
Jakob Sauer
Bettina Baecker
-----
```

### Test 4: Flugreise mit nur einem Flug

```
Microsoft Visual Studio Debug Console

Departure city: Belgrad
Departure time: 2022-12-28      14:40

Arrival city:   Frankfurt
Arrival time:   2022-12-28      16:45

Stations of voyage:   Belgrad -> Frankfurt
-----
Flights in total: 1

List of flights:
Belgrad -> Frankfurt
    Departure time: 2022-12-28      14:40
    Arrival time:   2022-12-28      16:45
-----
Passengers in total: 4

List of voyage passengers:
Larissa Hauser
Lorenzo Mueller
Janine Suess
Marina Koch
-----
```

### Test 5: Leere Flugreise (ohne Flüge und ohne Personen)

```
Microsoft Visual Studio Debug Console

Departure city: Muenchen
Departure time: 0-00-00 00:00

Arrival city:   Linz
Arrival time:   0-00-00 00:00

Stations of voyage:
-----
Flights in total: 0

List of flights:
-----
Passengers in total: 0

List of voyage passengers:
-----
```

## Beispiel 2: Eine Herakles Aufgabe – Stücklistenverwaltung

### Lösungsidee

Die Klassen *part* und *composite\_part* richten sich in Aufbau nach den Forderungen der Angabe. Zusätzlich enthält ein Objekt der Klasse *composite\_part* einen Vektor von Part-Zeigern als privates Attribut, um die dazugehörigen Parts speichern zu können.

In der Klasse *composite\_part* sind außerdem noch die beiden Methoden „store“ und „load“ vertreten, die von der abstrakten Klasse *storable* geerbt werden. Die Angabe spricht explizit von einer „Stückliste“, die gespeichert bzw. geladen werden können soll. Lediglich ein Objekt der Klasse *composite\_part* kann als Stückliste charakterisiert werden und da man diese Stückliste ebendeshalb nicht in ein Objekt der Klasse *part* speichern können muss, wird hier bewusst darauf verzichtet, die beiden genannten Methoden in der Klasse *part* zu implementieren (obwohl selbstverständlich jedes *composite\_part* eigentlich auch ein *part* ist).

Das Format des Speicherns bzw. Wiederherstellens einer Stückliste richtet sich am Format des Hierarchy-Formatters, es wird also mithilfe von Tabs eine hierarchische Verzweigung ermöglicht bzw. eingelesen.


Das Einlesen und Auslesen an sich bedient sich der Rekursion, um die gesamte hierarchische Struktur abbilden zu können.

Die beiden Formatter *set\_formatter* und *hierarchy\_formatter* bedienen sich ebenfalls der Rekursion zur Durchwanderung eines Part-Baums. Abhängig davon, ob das aktuelle Teil dann ein einfaches *part* oder ein *composite\_part* ist, wird die Rekursion entweder abgebrochen bei einem *part* oder wieder vertieft bei einem *composite\_part*.

Der *hierarchy\_formatter* benutzt je nach Rekursionstiefe verschiedene Einrückungsstufen, um die Hierarchie abzubilden.

Der *set\_formatter* bedient sich einer Map-Konstruktion, um die Teile der untersten Rekursionsebene zu erfassen, zu zählen und schließlich auszugeben.

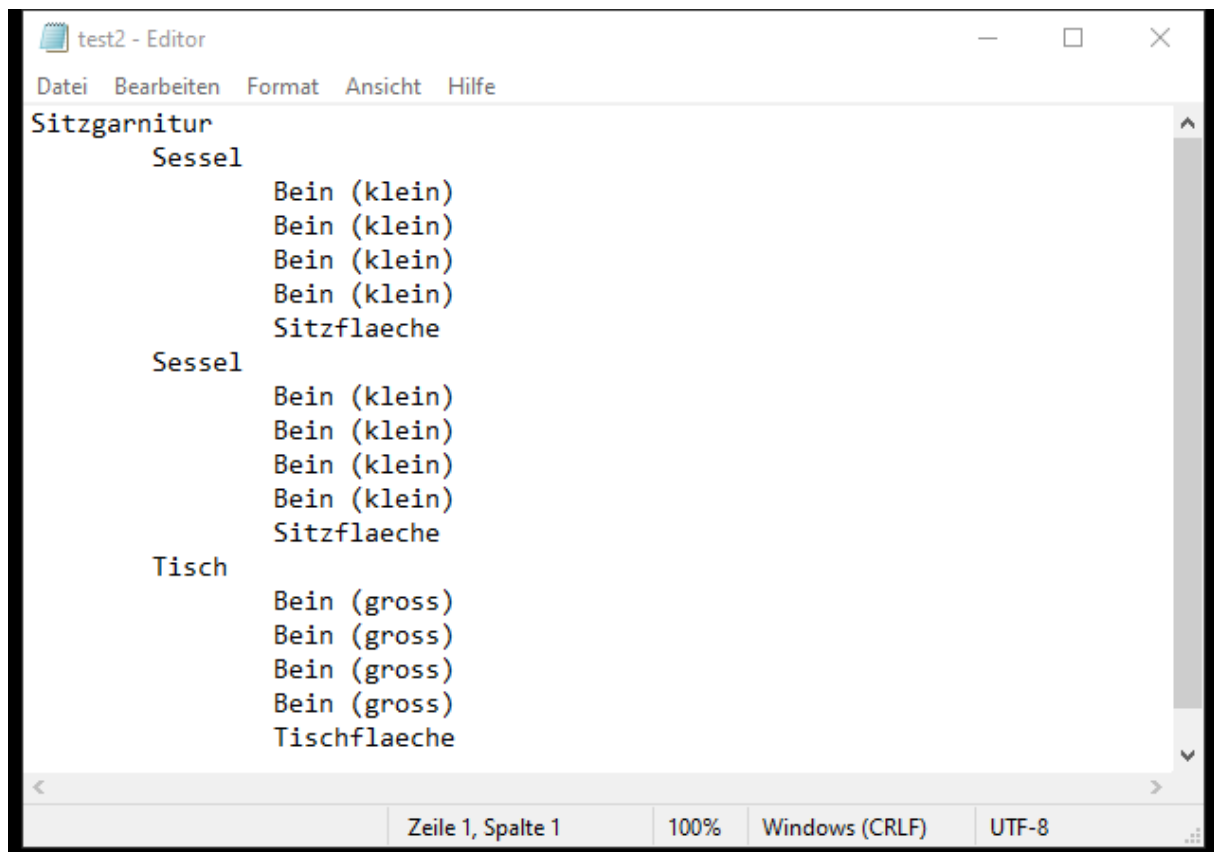
## Testfälle

**Test 1:** Hierarchy-Formatter- und Set-Formatter-Darstellung wie in der Angabe


```

Microsoft Visual Studio Debug Console

Sitzgarnitur
  Sessel
    Bein (klein)
    Bein (klein)
    Bein (klein)
    Bein (klein)
    Sitzflaeche
  Sessel
    Bein (klein)
    Bein (klein)
    Bein (klein)
    Bein (klein)
    Bein (klein)
    Sitzflaeche
  Tisch
    Bein (gross)
    Bein (gross)
    Bein (gross)
    Bein (gross)
    Tischflaeche
Sitzgarnitur:
  4 Bein (gross)
  8 Bein (klein)
  2 Sitzflaeche
  1 Tischflaeche
  
```

**Test 2:** Test von store-Funktion mit Beispiel aus Angabe


```

test2 - Editor
Datei Bearbeiten Format Ansicht Hilfe

Sitzgarnitur
  Sessel
    Bein (klein)
    Bein (klein)
    Bein (klein)
    Bein (klein)
    Sitzflaeche
  Sessel
    Bein (klein)
    Bein (klein)
    Bein (klein)
    Bein (klein)
    Sitzflaeche
  Tisch
    Bein (gross)
    Bein (gross)
    Bein (gross)
    Bein (gross)
    Tischflaeche
  
```

Zeile 1, Spalte 1    100%    Windows (CRLF)    UTF-8

### Test3: Test load-Funktion mit Beispiel aus Angabe und Ausgabe in Hierarchie und Set

```

Microsoft Visual Studio Debug Console

Sitzgarnitur
  Sessel
    Bein (klein)
    Bein (klein)
    Bein (klein)
    Bein (klein)
    Sitzflaeche
  Sessel
    Bein (klein)
    Bein (klein)
    Bein (klein)
    Bein (klein)
    Sitzflaeche
  Tisch
    Bein (gross)
    Bein (gross)
    Bein (gross)
    Bein (gross)
    Tischflaeche

Sitzgarnitur:
  4 Bein (gross)
  8 Bein (klein)
  2 Sitzflaeche
  1 Tischflaeche
  
```

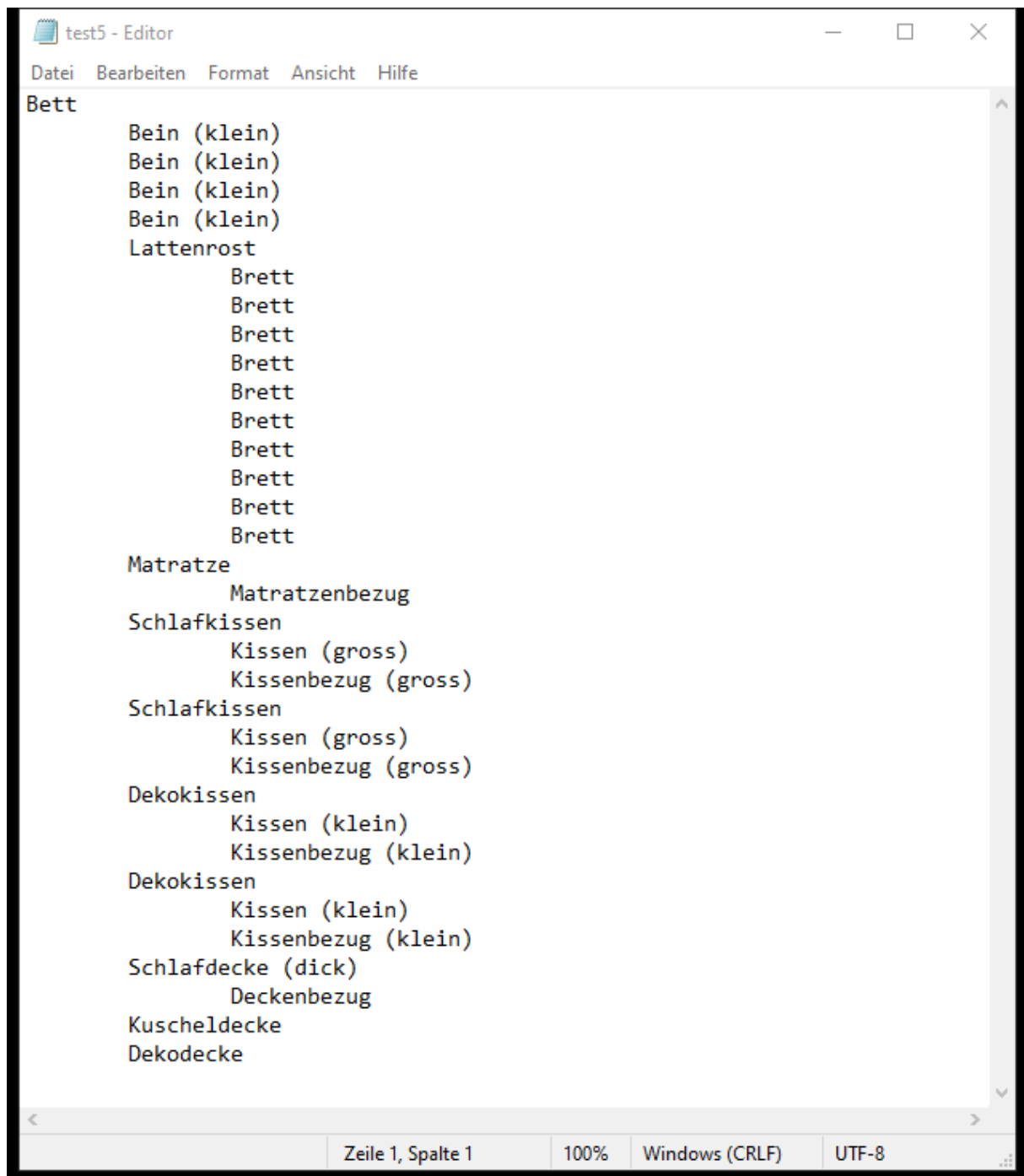
### Test4: Neues Beispiel mit Test von load-Funktion und Ausgabe in Hierarchie und Set

```

Microsoft Visual Studio Debug Console

Bett
  Bein (klein)
  Bein (klein)
  Bein (klein)
  Bein (klein)
  Lattenrost
    Brett
    Brett
    Brett
    Brett
    Brett
    Brett
    Brett
    Brett
    Brett
    Brett
    Brett
    Brett
  Matratze
    Matratzenbezug
  Schlafkissen
    Kissen (gross)
    Kissenbezug (gross)
  Schlafkissen
    Kissen (gross)
    Kissenbezug (gross)
  Dekokissen
    Kissen (klein)
    Kissenbezug (klein)
  Dekokissen
    Kissen (klein)
    Kissenbezug (klein)
  Schlafdecke (dick)
    Deckenbezug
  Kuschedecke
  Dekodecke

Bett:
  4 Bein (klein)
  10 Brett
  1 Deckenbezug
  1 Dekodecke
  2 Kissen (gross)
  2 Kissen (klein)
  2 Kissenbezug (gross)
  2 Kissenbezug (klein)
  1 Kuschedecke
  1 Matratzenbezug
  
```

**Test 5:** store-Funktion mit Beispiel aus Test 4

```
test5 - Editor
Datei Bearbeiten Format Ansicht Hilfe
Bett
    Bein (klein)
    Bein (klein)
    Bein (klein)
    Bein (klein)
    Lattenrost
        Brett
        Brett
        Brett
        Brett
        Brett
        Brett
        Brett
        Brett
        Brett
        Brett
        Brett
    Matratze
        Matratzenbezug
    Schlafkissen
        Kissen (gross)
        Kissenbezug (gross)
    Schlafkissen
        Kissen (gross)
        Kissenbezug (gross)
    Dekokissen
        Kissen (klein)
        Kissenbezug (klein)
    Dekokissen
        Kissen (klein)
        Kissenbezug (klein)
    Schlafdecke (dick)
        Deckenbezug
    Kuschedecke
    Dekodecke
Zeile 1, Spalte 1 100% Windows (CRLF) UTF-8
```

**Test6:** load-Funktion mit leerer Datei