# CLog Documentation

HHU-Programmierpraktikum SS2016 Projekt 5

## Inhaltsverzeichnis



# 1 Verwendete Strukturen

## 1.1 Wrapper-Klassen

Wrapper-Klassen enthalten in der Regel eine private Variable des entsprechenden Datentyps, welche über den Konstruktor gesetzt werden kann. Da Getter- und Setter-Methoden nicht erlaubt sind, ist es nicht möglich, den gespeicherten Wert abzufragen oder nach der Erstellung zu manipulieren. Lediglich eine Ausgabe auf der Konsole (mittels `Ausgabe`) ist durch die Methode `ausgeben()` möglich und völlig ausreichend für die hier verwendeten Zwecke.

Wenn die Wrapper-Klassen Wrapper für Elemente eines Datensatzes sind, so enthalten sie in der Regel ebenfalls eine Methode `unterstreichen (char zeichen)` zum Ausgeben einer aus `zeichen` bestehenden Zeichenkette in der Länge des durch die Methode `ausgeben()` produzierten Strings, und eine Methode `vonEingabeEinlesen()`, welche statisch ist und die für ein Objekt des entsprechenden Typs von der Konsole einliest

(den Nutzer dazu auffordert), ein Objekt des entsprechenden Typs daraus erzeugt und dieses zurückgibt.

### 1.2 „Vereinigungs"-Klassen

Ein Kernproblem ist die Restriktion von maximal 2 Attributen pro Klasse. Da recht viele Attribute gespeichert werden müssen, ist es nicht möglich, z.B. eine Klasse `Datum` zu erstellen, welche die Attribute `tag`, `monat` und `jahr` speichert.

Ich löse dieses Problem, indem ich erst eine Klasse `MonatJahr` erstelle, welche die Attribute `monat` und `jahr` speichert, und dann eine weitere Klasse `Datum`, welche die Attribute `monatJahr` und `tag` speichert, um diese Regel zu befolgen.

Etwaige Funktionsaufrufe von Methoden wie `ausgeben()` oder `unterstreichen()` (weiteres dazu später) bestehen dann aus den Aufrufen der entsprechenden Methoden bei den gespeicherten Objekten (z.B. `Datum.ausgeben()` ruft `MonatJahr.ausgeben()` und `Tag.ausgeben()` auf).

## 2 Packages

Im Programm sind die einzelnen Klassen in Packages sortiert:

- Package `data` enthält die Klassen, die einen Datensatz repräsentieren und die Datensätze verwalten:
  - `Datensatz` repräsentiert einen Datensatz
  - `DatensatzListe` speichert alle Datensätze in einer Liste, bietet Methoden zum Hinzufügen, Durchsuchen (und Ausgeben).
  - `DatensatzManager` verwaltet die Datensatzliste, also Datensätze hinzufügen, durchsuchen (und ausgeben), in eine Datei speichern und aus einer Datei lesen.
- Package `datumzeit` enthält alle Klassen, die zum dem Speichern von Datum und Uhrzeit verwendet werden:
  - `Jahr, Minute, Monat, Sekunde, Stunde` speichern die entsprechende Angabe als Integer.
  - `MonatJahr, StundeMinute` werden verwendet, um die jeweiligen Objekte zu vereinen (maximal 2 Attribute pro Klasse).
  - `Datum` stellt ein Datum dar (Vereint `MonatJahr` und `Tag`).
  - `Zeit` stellt eine Uhrzeit dar (Vereint `StundeMinute` und `Sekunde`)
- Package `eingabeausgabe` behandelt die Ein- und Ausgabe jeglicher Art, sowohl zum/vom Terminal als auch in/aus Dateien:

  - `Ausgabe` bietet Funktionen zum Ausgeben von Text in die Kommandozeile.
  - `Eingabe` bietet Funktionen zum Lesen von Text aus der Kommandozeile.
  - `DeSerializer` bietet Funktionen, um Java-Objekte in eine gegebene Datei zu speichern bzw. aus einer zu lesen.
  - `Path` repräsentiert einen Dateipfad.
- Package `fields` beinhaltet alle Felder, die in einem Datensatz gespeichert werden sollen (zusammengefügt aus gewrappten Strings und Primitives und deren zusammenführenden Klassen):
  - `DatumZeit, NameWohnort, TitelText` speichern jeweils die entsprechenden Objekte, um sie zu einem zu vereinen.
  - `NameWohnortDatumZeit` vereint `NameWohnort` und `DatumZeit` zu einem Objekt.
  - `NameWohnortDatumZeitTitelText` vereint `NameWohnortDatumZeit` und `TitelText` zu einem Objekt.
  - `Schlagworte` speichert eine `ArrayList`⟨`Schlagwort`⟩.
- Package `main` enthält lediglich die Main-Klasse, welche die Main-Methode enthält.
- Package `menues` enthält die Objekte, welche die Menüs repräsentieren. Jedes Menü-Objekt enthält dabei eine Methode `menue()`, welche die dem Menü zugeordnete Aktion ausführt (z.B. Daten Einlesen, Programm beenden etc.) und entsprechende Ein- und Ausgaben verwaltet. Die `menue()`-Methode endet, wenn eine jeweilige Aktion abgeschlossen ist.
- Package `textangaben` enthält Wrapper-Klassen und „Vereinigungsklassen“für alle weiteren Angaben eines Datensatzes (Textangaben, bis auf Datum + Zeit):
  - `Nachname, Schlagwort, Text, Titel, Vorname, Wohnort, Zeichen` speichern jeweils ein entsprechendes Element als `String` (bzw. `char` im Falle von Zeichen).
  - `Name` vereint `Vorname` und `Nachname` zu einem Objekt.

# 3 Funktionsweisen

Der Nutzer startet das Programm im Terminal mit dem Aufruf `java Clog`. Dadurch wird in der Klasse `main.Clog` die Main-Methode `public static void main(String args[])` aufgerufen. Diese Methode enthält lediglich einen Aufruf an die Methode `menue()` der Klasse `menues.Menue0`, welche die Main-Loop enthält. Darin wird für den Nutzer lesbar das Hauptmenü ausgegeben, auf eine

```
Hauptmenü:
1) Clog-Eintrag erzeugen
2) Clog ausgeben
3) Clog laden
4) Clog speichern
5) Programm beenden
6) Alle Datensätze ausgeben
```

**Abbildung 1:** Hauptmenü

Antwort gewartet, die Antwort mittels eines Switches ausgewertet und die entsprechende Subroutine in einer der anderen Menüklassen aufgerufen.

## 3.1 Generelle Funktion der Menüpunkte

Es gibt zu jedem Menüpunkt `X` eine Klasse `menues.MenueX`. Diese beinhaltet eine Methode `menue()`, welche den Programmcode enthält, der beim Auswählen des Menüpunktes ausgeführt werden soll. Ist die Methode durchlaufen, so wird ins Hauptmenü zurückgekehrt (dank der Main loop).

## 3.2 Menüpunkt 1: `Clog-Eintrag erzeugen`

Wählt der Nutzer Menüpunkt 1, so wird die Funktion `menue()` in der Klasse `menues.Menue1` aufgerufen. Diese behandelt das Erzeugen eines neuen Logeintrags. Dazu wird eine Ausgabe auf der Konsole erzeugt, welche den Nutzer über den gestarteten Vorgang informiert und die Bestätigung der Eingaben mit Enter anfordert. Danach soll ein neues `Datensatz`-Objekt gespeichert werden, und zwar als der Rückgabewert von `Datensatz.vonEingabeEinlesen()`.

Dadurch startet die Klasse selbstständig den Einlesevorgang, und lässt auch die gespeicherten Objekte den Einlesevorgang starten. Der Nutzer wird nach *Vorname, Nachname, Wohnort, Datum und Zeit, Titel, Text* und *Schlagworten* gefragt, wobei Titel und Text die einzigen Eingaben sind, bei denen mehr als ein Wort erlaubt ist. Dazu wird die Eingabe nicht nur durch einen Zeilenumbruch (Enter), sondern zusätzlich auch noch durch ein `';'` in der Eingabe beendet. Eine Eingabe von mehreren Schlagworten auf einmal führt dazu, dass jedes einzelne Wort als Schlagwort interpretiert wird.

In allen Klassen außer der `DatumZeit`, `Titel` und `Text` wird einfach der nächste String vom Eingabe-`Scanner` abgefragt und als Wert verwendet. Bei den Klassen `Titel` und `Text` wird zunächst der Delimiter des Scanners auf den spezifizierten Terminierungs-Character `';'` gewechselt, und vor sowie nach dem Einlesen etwaig übergebliebene Delimiter (Zeilenumbrüche von der vorherigen Eingabe oder das Semikolon dieser Eingabe) im Scanner übersprungen, damit diese nicht in dieser oder der nächsten Eingabe interpretiert werden.

In der Klasse `DatumZeit` wird das Datum im Format `TT.MM.JJJJ` eingelesen, die Zeit im Format `HH:MM:SS`, wobei für `TT`, `MM`, `JJJJ`, `HH`, `MM` und `SS` Integer erwartet werden. Dabei wird jeweils das Konstrukt `TT.MM.JJJJ` bzw. `HH:MM:SS` als ganzer String eingelesen und entsprechend der Trennzeichen `'.'` bzw. `':'` gesplittet. Die einzelnen „Stellen“des daraus resultierenden StringArrays werden versucht nach Integern zu parsen. Etwaige Fehlermeldungen werden direkt abgefangen, und es wird eine entsprechende Fehlermeldung ausgegeben (dabei wird zwischen Ëingabe ist keine Zahlöder Ëingabe ist nicht

vollständigünterschieden), und der Nutzer wird sofort erneut zur Eingabe aufgefordert.

Sind alle Daten erfasst, so wird der `Datensatz` dem `DatensatzManager` übergeben, um ihn in der `DatensatzListe` zu hinterlegen. Danach ist der Methodenaufruf zuende, und das Hauptmenü wird erneut ausgegeben.

## 3.3 Menüpunkt 2: `Clog ausgeben`

In diesem Menüpunkt wird der Nutzer um Eingabe eines Schlagwortes gebeten, nach dem die Logeinträge durchsucht werden sollen. Die Eingabe akzeptiert nur ein einzelnes Wort und wird erneut mit Enter bestätigt.

Anschließend wird die Methode `DatensatzManager.datensaetzeMitSchlagwortAusgeben (Schlagwort schlagwort)` aufgerufen, welche den Aufruf an die Methode `DatensatzListe.datensaetzeMitSchlagwortAusgeben (Schlagwort schlagwort)` weitergibt. Diese iteriert durch die Liste, und führt auf jedem `Schlagwort` die Methode `Datensatz.ausgebenWennEnthaeltSchlagwort (Schlagwort schlagwort)` ausführt.

Die Funktion der Methode ist sehr simpel, sie fragt die gespeicherte `ArrayList⟨ Schlagwort ⟩`, ob sie das übergebene `schlagwort` enthält, und reicht den Rückgabewert durch. Zum Vergleichen wird automatisch `Schlagwort.equals (Object anObject)` durch die `ArrayList⟨ Schlagwort ⟩` verwendet.

## 3.4 Menüpunkt 3: `Clog laden`

Mit diesem Menüpunkt wird ein gespeicherter `CLog` aus einer Datei geladen. Es wird zunächst der Nutzer nach einem Pfad gefragt, von dem aus die Datei eingelesen werden soll. Die Methode `DatensatzManager.readFromFile (Path path)` wird aufgerufen, welche das eigentliche Einlesen durch die Methode `eingabeausgabe.DeSerializer.readFromFile (Path path)` durchführen lässt. Dazu wird zunächst ein Objekt vom Typ `DeSerializer` erstellt, und mit dem Einzulesenden Datentyp parametrisiert (in diesem Fall `DatensatzListe`). Die Funktion verwendet einen `ObjectInputStream`, der auf einen `FileInputStream` gesetzt wird, zum Einlesen der serialisierten Daten in's Programm. Danach wird ein Cast in den parametrisierten Datentyp versucht und das Element zurückgegeben.

Etwaige Exceptions werden von der aufrufenden Methode `DatensatzManager.readFromFile (Path path)` behandelt. Im Falle eines Fehlers wird eine entsprechende Fehlermeldung ausgegeben (dabei wird unterschieden, ob die Datei nicht existiert, nicht gelesen werden kann oder nicht lesbar ist), und ins Hauptmenü zurückgekehrt.

## 3.5 Menüpunkt 4: `Clog speichern`

Mit diesem Menüpunkt kann der aktuell geladene `CLog` in eine Datei gespeichert werden. Dazu wird der Einfachheit halber einfach das Objekt `DatensatzListe`, welches alle Datensätze enthält, Serialisiert und mittels eines `ObjectOutputStreams`, der auf einen `FileOutputStream` gesetzt wird, zum Schreiben der serialisierten Daten verwendet. Der Pfad ist dabei vom Benutzer frei wählbar und wird zuvor abgefragt.

Schlägt das Schreiben fehl, so werden die entsprechenden Exceptions abgefangen und eine dazugehörige Fehlermeldung ausgegeben. Danach wird der Nutzer zurück ins Hauptmenü gebracht, von wo aus er die nächste Aktion tätigen kann.

## 3.6 Menüpunkt 5: `Programm beenden`

Dies ist der wohl einfachste aller Menüpunkte, in diesem Fall ruft die `Menue5.menue()` einfach einen `System.exit(0)` auf, wobei die 0 für den Exitcode einer „normal termination“steht. Das Programm wird somit sofort beendet, vorher wird zur Klarheit eine entsprechende Ausgabe in die Konsole geschrieben.

## 3.7 Menüpunkt 6: `Alle Datensätze ausgeben`

Diesen Menüpunkt habe ich zusätzlich eingeführt, da ich es einerseits für Debugging-Zwecke, andererseits auch für den Enduser für interessant halte, alle Datensätze ansehen zu können. In diesem Fall wird die Methode `DatensatzManager.ausgeben()` aufgerufen, welche wiederum die Methode `DatensatzListe.ausgeben()` aufruft, welche wiederum über die Liste aller Datensätze iteriert und für jeden Datensatz die Methode `ausgeben()` aufruft. Somit werden alle Datensätze in die Konsole geschrieben.

## 3.8 Ausgaben und Eingaben

Die Eingaben und Ausgaben dieses Programmes werden mit den Klassen `eingabeausgabe.Ausgabe` und `eingabeausgabe.Eingabe` realisiert. `Ausgabe` stellt im Prinzip einen Wrapper für `System.out` dar, `System.out` wird dabei als Variable gespeichert, und die entsprechenden Methoden (`print()`, `printline()`, ...) werden entsprechend an `System.out` weitergeleitet. Mit dieser Vorgehensweise wird der Regel „Nicht mehr als ein Punkt pro Zeile“entsprochen.

Die Eingabe ist dann schon etwas komplizierter, hierfür wird ein `Scanner` verwendet. Er wird statisch erzeugt, und die Methoden `nextString()` (Gibt den nächsten String zurück) und `nextInteger()` (Gibt den nächsten Integer zurück) werden entsprechend an den `Scanner` weitergeleitet (Methoden `next()` und `nextInt()`).

### 3.8.1 nextString (String delimiter)

Darüber hinaus wird eine Methode `nextString (String delimiter)` bereitgestellt, welche für dieses eine Einlesen den Delimiter des `Scanner`, welcher bestimmt, wonach Einträge abgetrennt werden, auf den übergebenen Delimiter wechselt. Da es Probleme beim Wechseln von Delimitern mit eventuell noch im Buffer befindlichen Delimitern gibt (diese werden beim Wechseln des Delimiters nicht weiter als Delimiter betrachtet und als normale Eingabe geparst, was zu unerwünschten Eingaben führt), werden vorher und nachher ein eventuell im Buffer befindlicher Delimiter übersprungen (`Scanner.skip()`). Dies ist ein Workaround für eine Funktion, die der `Scanner` eigentlich selbstverständlich bereitstellen sollte.

# 4 Fragen

Folgende Fragen habe ich mir während der Umsetzung der Aufgabenstellung beantwortet, da sie aus der Aufgabenstellung nicht klar hervorgingen:

- Werden bei den Feldern `Vorname, Nachname, Wohnort` mehr als ein Wort akzeptiert? - Nein
- Werden bei den Feldern `Titel, Text` mehr als ein Wort akzeptiert? - Ja (Letzteres ist vermutlich auch selbstverständlich)
- Wird bei dem Feld `Text` mehr als eine Zeile akzeptiert (Absätze im Text)? - Ja
- Wie gibt der Nutzer an, dass die Eingabe für das aktuelle Feld im Falle von `Titel` oder `Text` beendet ist? - Die Eingabe endet auf ein `';'`, danach wird mit Enter bestätigt.
- Muss das Format, in dem die Einträge gespeichert werden, von externen Programmen lesbar sein? - Nein. Ich dachte erst an eine Umsetzung desselben mittels JSON, aber es gibt keine Java-Interne JSON-Library.
- In welchem Format erfolgt die Ausgabe eines Datensatzes? - Siehe Abbildung 2.

```
Name: Peter Meier Wohnort: Düsseldorf Datum, Uhrzeit: 2.6.2016 13:37:37
========================================================================
Titel: Hallo Welt!
Ich bin ein Testeintrag!
Über diese erste Zeile hinaus enthalte ich noch eine zweite Zeile!
Dies ist die dritte und letzte Zeile.
Schlagworte: Test Eintag Das ist ein Test Banane 1337
```

**Abbildung 2:** Ausgabekonvention eines Datensatzes