

Software–Projekt 1 2013

VAK 03-BA-901.02

Architekturbeschreibung

<Five and a half Men>



Joscha Cepok	redlotus@tzi.de	2763987
Eugen Konrad	ekonrad@tzi.de	2781241
Joschka Köster	jkoester@tzi.de	2788610
Matthias Kümmel	mkuemmel@tzi.de	2790942
Olga Miloevich	halfelv@tzi.de	2586817
Arthur Niedzwiecki	artnie91@tzi.de	2698960

Inhaltsverzeichnis

1	Einführung	3
1.1	Zweck	3
1.2	Status	3
1.3	Definitionen, Akronyme und Abkürzungen	3
1.4	Referenzen	3
1.5	Übersicht über das Dokument	4
2	Globale Analyse	5
2.1	Einflussfaktoren	5
2.2	Probleme und Strategien	8
3	Konzeptionelle Sicht	13
4	Modulsicht	15
4.1	Arbeitspakete	15
4.2	Klassendiagramm	17
4.3	Quellcode und Beschreibung	23
5	Datensicht	59
6	Ausführungssicht	63
7	Zusammenhänge zwischen Anwendungsfällen und Architektur	64
8	Evolution	66

Version und Änderungsgeschichte

(bearbeitet von: EUGEN KONRAD)

Version	Datum	Änderungen
1.0	18.06.2013	Dokumentvorlage als initiale Fassung kopiert
1.1	18.06.2013	Erstellen des Sequenzdiagramm für Punkt 7
1.2	18.06.2013	Erstellen des Java-Codes
1.3	18.06.2013	Erstellen des Javadoc und einbinden des Codes
1.4	26.06.2013	Einbinden und Bearbeitung der Datensicht
1.5	27.06.2013	Einbinden der Konzeptionellen Sicht
1.6	27.06.2013	Einbinden der Problemkarten
1.7	27.06.2013	Bearbeitung und Einbinden Letzter Aufgabenpunkte
1.8	27.06.2013	Angabe der Referenzen
1.9	27.06.2013	Ausfüllen der Versionsgeschichte
2.0	27.06.2013	Entgültige Rechtschreibkorrekturen

1 Einführung

1.1 Zweck

Entfällt in SWP-1

1.2 Status

Entfällt in SWP-1

1.3 Definitionen, Akronyme und Abkürzungen

(bearbeitet von: ARTHUR NIEDZWIECKI)

Alle verwendeten Begriffe sind verständlich und Sachverhalte nachvollziehbar, Definitionen werden für dieses Dokument nicht benötigt.

1.4 Referenzen

(bearbeitet von: ARTHUR NIEDZWIECKI)

Koschke, Rainer: Architekturbeschreibung-Vorlage SWP1, SS 2013

Koschke, Rainer: 3-Hinweise-Abgabe-Architektur SWP1, SS 2013 

Koschke, Rainer: Architekturbeschreibung-Beispiel SWP1, SS 2013

(bearbeitet von:)

1.5 Übersicht über das Dokument

Entfällt in SWP-1

2 Globale Analyse



(bearbeitet von: OLGA, MILOEVICH, JOSCHA CEPOK, JOSCHKA KÖSTER, ARTHUR NIED-ZWIECKI)

Hier werden Einflussfaktoren aufgezählt und bewertet sowie Strategien zum Umgang mit interferierenden Einflussfaktoren entwickelt.

2.1 Einflussfaktoren

Einflussfaktoren	Flexibilität und Veränderlichkeit	Auswirkung
Organisation		
O1: Zeitmanagement		
O1.1: Time-To-Market; Abgabe am 19.08.13	Keine	Zeitbeschränkung des gesamten Projektes, hat bei schlechter Planung Auswirkung auf die Qualität
O1.2: Meilensteine; Abgabe von Projektteilen in Zeitabschnitten	Gering; Abhängig vom Professor und anderen gegebenen Pflichten der Studenten	Geregelter Ablauf des Projektfortschrittes, beugt Qualitätsverlust durch schlechte Zeitplanung vor



Abbildung: Einflussfaktoren1





Technische Faktoren		
T1: Kompatibilität		
T1.1: Unabhängigkeit; Die Software muss auf Windows, Linux, iOS und die App auf Android Betriebssystemen verwendbar sein	Gering; Soll die App z.B. auch auf Apple-Handgeräten ausführbar sein, muss auf weitere Programmiersprachen zugegriffen werden 	Beschränkung der verwendeten Programmiersprachen 
T2: Implementierung		
T2.1: Internetbrowser; Ein Internetbrowser muss für die Benutzung installiert sein	Mittel; Möglichkeit das System über Java-GUIs zu implementieren, das Programm und ein Java-Environment muss sich in dem Fall auf dem Rechner des Benutzers befinden 	Legt die Vorgabe für die Implementierung der Benutzeroberflächen fest.
T2.2: Internet; Eine Verbindung vom Benutzer zum Internet muss bestehen	Keine	Vereinfacht die Handhabung von Medien- und Benutzerdaten massiv
T2.3: Programmiersprachen; Der Server wird in SQL, Funktion des Systems in Java, das Design der App in XML und die Website in HTML geschrieben	Gering; 	Kompetenz in anderen Programmiersprachen wäre erforderlich

Abbildung: Einflussfaktoren2

Produktfunktionen		
P1: Performanz		
P1.1: Schnell und präzise; Die Software soll ohne unnötige Zusätze möglichst schnell arbeiten	Hoch; Auf Wunsch des Kunden können Zusatzfunktionen hinzugefügt/entfernt werden	Kann zu Zeitnot führen und dadurch zu Qualitätsverlust oder andererseits die Implementierung vereinfachen
P2: Bedienbarkeit		
P2.1: Übersichtlich; Die Oberflächen müssen benutzerfreundlich (übersichtlich) sein	Hoch; Das Layout der Oberflächen kann immer geändert werden	Technische und visuelle Kompetenz des Software-Teams vorausgesetzt
P3: Sicherheit		
P3.1: Stabiler Datenaustausch; Datentransfer zwischen Benutzern und Datenbank ist verlustfrei	Keine	Die Softwareteile müssen sorgfältig auf Schwachstellen untersucht und gesichert werden, Zeitintensiv
P3.2: Verschlüsselte Daten; Nutzerdaten und Passwörter müssen verschlüssert gesichert werden	Keine	Die Implementierung von Verschlüsselung und die Entschlüsselung von Daten kann schwierig werden
P3.3: Fremdzugriff; Einem Zugriff auf Daten von Außen wird vorgebeugt	Keine	Wie bei P3.1: Schwachstellen in der Software beheben.

Abbildung: Einflussfaktoren3

2.2 Probleme und Strategien

Hier werden einzelne Probleme, die bei der Verwirklichung der Software eine Rolle spielen können als Problemkarten beschrieben. Sie beinhalten immer zuerst den Namen des Problems, eine kurze Beschreibung und die von dem Problem oder dessen Bewältigung betroffene Einflussfaktoren. Dazu werden zu jedem Problem verschiedene Strategien erklärt, die zur Lösung des Problems führen



1. Problem: Wunsch des Kunden: Zusatzfunktionen

Der Kunde könnte sich eine erweiterte Funktionalität des Programms wünschen, welche er bis zur Abnahme erwartet.

Einflussfaktoren

1.1 Keine feststehenden Anforderungen im Vertrag vereinbart

Lösung

Strategie: Intensivierung

Entwickler intensiver und länger arbeiten lassen.

Anmerkung: Anzahl der Entwickler ist begrenzt. Entwickler können nicht Tag und Nacht durch arbeiten.

Strategie: Softwareintegration

Notwendige Module von „dritten“ Firmen kaufen

Anmerkung: Die Kosten für das Projekt müssen einen Kauf zulassen.

Strategie: Vertrag verlängern

Den Kunden über die zusätzlich anfallende Zeit informieren und die Fristen verlängern.

Anmerkung: Kunde könnte uneinsichtig sein.

Abbildung Problem 1

2.Problem: Ausfall einigen Entwickler

Die Entwickler können durch Krankheit oder andere Faktoren temporär oder permanent ausfallen.

Einflussfaktoren

1.1 Abwerbung durch andere Firma

Lösung

Strategie: Arbeiterteilung



Falls ein Ausfall permanent ist, muss der Teil des ausfallenden Entwicklers zwischen anderen verteilt werden.

Anmerkung: Der Zeitplan könnte sich sehr verzögern oder die Mitarbeiter könnten für das Problemgebiet nicht ausgebildet sein.

Strategie: Neuer Entwickler

Einen neuen Entwickler für das Projekt besorgen

Anmerkung: Die Einarbeitungszeit für den Entwickler muss zusätzlich eingeplant werden. Es gibt keine Garantie dafür, dass der Entwickler für das Gebiet die gleiche Qualifikation besitzt, wie das ausgefallene Mitglied.

Abbildung Problem 2

3.Problem: Qualitätsverlust der Software

Durch schlechte Planung/Zeitnot/Entwicklerausfall/nicht ausreichendes Testen kann das Produkt unter Qualitätsverlust leiden.

Einflussfaktoren

1.1 Ausfall eines Entwicklers

1.2 Unterschätzung des Zeitaufwands

1.3 Unzureichender Testaufwand

Lösung


Strategie: priorisierte Entwicklung

Priorisierung der Funktionen. Hauptfunktionen werden zuerst implementiert und getestet.

Anmerkung: Anzahl der Entwickler und Zeit der Entwicklung ist begrenzt. Mache Hauptfunktionen könnten nicht komplett getestet werden, da sie teilweise von nicht implementierten Unterfunktionen abhängen und man nur über vorgegebene Rümpfe testen könnte.

Strategie: Wiederverwendung

Funktionierenden Code von anderen Stellen wiederverwenden

Anmerkung:  Der Code könnte nicht genau den Anforderungen entsprechen und müsste angepasst werden.

Strategie: Regelmäßige Rücksprache mit dem Kunden

Den Kunden regelmäßig mit einbeziehen

Anmerkung: Kunde könnte keine Zeit haben

Abbildung Problem 3

4.Problem: Oberfläche ist nicht übersichtlich

Die angebotene Oberfläche ist für den Benutzer nicht geeignet.

Einflussfaktoren

1.1 Unzureichend viel Rücksprache mit dem Kunden

1.2 Schlechte Struktur

Lösung

Strategie: Testen mit zusätzlichen Ressourcen



Menschen, die keine Beziehung zum Projekt haben, bitten, die Bedienbarkeit zu bewerten.

Anmerkung: Menschen, die mit Bibliothek und Software nichts zu tun haben, können die Bedienbarkeit der Oberflächen nicht präzise bewerten.

Strategie: Testen mit Kunden


Dem Kunden nach dem Strukturieren der Oberfläche nach Kritik fragen.

Anmerkung: Ansprechpartner kann keine Zeit für den Termin finden.

Ansprechpartner kann nicht unbedingt die Bedienbarkeit für die Oberfläche bewerten.

Abbildung Problem 4

5.Problem: Sicherheit

Die implementierte Software kann durch verschiedene Faktoren unsicher sein. Durch Zeitdruck können die notwendigen Maßnahmen nicht getroffen werden und für die anschließende Wartung ist zu wenig Zeit. 


Einflussfaktoren

1.1 Nicht identifizierte Gefahrenquellen


Lösung

Strategie: Integration fremder Sicherheitssoftware

Sicherheitssoftware kaufen und integrieren.

Anmerkung: Fremde Software könnte sehr teuer sein. Eine günstige Variante von unbekannten Hersteller können unsicher oder gefährlich sein 

Strategie: Sicherer Umgang mit personenbezogenen Daten

Personenbezogene Daten sollen so viel wie nötig und so wenig wie möglich genutzt werden. 

Anmerkung: Passwörter und Benutzerdaten sollen verschlüsselt sein. Zugriff auf personenbezogene Daten soll durch Implementierung sehr beschränkt werden.

Abbildung Problem 5



6.Problem: Betriebssystem- und Hardwareanpassung

Durch die ständige Weiterentwicklung der Betriebssysteme und Hardware sind ggf. entsprechende Anpassungen notwendig. Die Anpassungen müssen entsprechend schnell vorgenommen werden.

Einflussfaktoren

1.1 Neue Standards in Betriebssystemen

Lösung

Strategie: Kapselung

Betriebssystem- und Hardware abhängige Komponenten können gekapselt werden und virtuelle Plattform für andere Komponente bilden.

Anmerkung: Man muss sich in ältere Software einarbeiten und sich über die neuen Standards informieren.

Strategie: Softwareintegration



Fremde verkapselte Module einbauen

Anmerkung: Wartung fremder Module würde bei der Firma selbst liegen, man ist darauf angewiesen, dass diese ihrer Pflicht schnell nachgehen.

Abbildung Problem 6

7.Problem: Kompatibilität

Die Software muss ggf. auf unterschiedlichen Plattformen funktionieren, dies könnte im Einzelfall nicht immer funktionieren.

Einflussfaktoren

1.1 Unklare Definition im Vertrag

Lösung

Strategie: Plattformvorgabe

Die Software für bestimmte Plattformen entwickeln und dies auch für den Nutzer kenntlich machen.

Anmerkung: Der Kunde könnte sich die Kompatibilität zu einer Plattform wünschen, welche für die Software ungeeignet ist.

Abbildung Problem 7

8.Problem: Inkompetente Mitarbeiter

Mitarbeiter überschätzt seine eigenen Fähigkeiten und ist seinen Aufgaben nicht gewachsen.

Einflussfaktoren

1.1 Falsche Selbsteinschätzung



Lösung

Strategie: Mitarbeiter vor Aufgabenverteilung auf Fähigkeiten testen

Testen, ob ein Mitarbeiter wirklich das nötige Fachwissen besitzt.

Anmerkung: Setzt voraus, dass sich jemand selbst damit auskennt und geeignet testen kann.

Abbildung Problem 8

9.Problem: Nicht Einhalten des Klassenmodell
Mitarbeiter setzt sich über die Vorgaben hinweg und hält sich nicht an die Signatur, Eingabe und Ausgabeparameter Einflussfaktoren <i>1.1 Eigensinnige, sture Mitarbeiter</i>
Lösung Strategie: Abstrakte Klassen, welche Methodensignaturen und Rückgabewerte vordefinieren Zu jeder zu implementierenden Klasse eine abstrakte Klasse vorgeben, welche das Vorgehen erklärt und vorgibt. Anmerkung: Vergrößert den Aufwand, das Klassendiagramm zu erstellen

Abbildung Problem 9

10.Problem: Beschädigung des Entwicklungsfortschritt
Hacker greifen an, Mitarbeiter sabotiert das Projekt, Schaden am Server. Einflussfaktoren <i>1.1 Unsichere Firmensoftware</i> <i>1.2 Böswillige Mitarbeiter</i> <i>1.3 Höhere Gewalt</i>
Lösung Strategie: Mehrere unabhängig voneinander gelagerte Sicherheitskopien Sowohl physische als auch Virtuelle Sicherheitskopien erstellen und diese sicher lagern. Anmerkung: Es sollten unabhängige Standorte und Firmen genutzt werden.

Abbildung Problem 10

11.Problem: Beschädigung des Datenbestandes bei Verbindungsabbruch während der Übertragung
Durch einen Verbindungsabriss werden nur teilweise Daten übertragen und die Datenbank beschädigt Einflussfaktoren <i>1.1 Instabile Verbindung</i>
Lösung Strategie: Sicheres Softwarekonzept Software so implementieren, dass unvollständige Befehle zu keiner Änderung in der Datenbank führen. Anmerkung: Beispielweise über doppelte Bestätigung.

Abbildung Problem 11

3 Konzeptionelle Sicht

(bearbeitet von: ARTHUR NIEDZWIECKI)

Hier sieht man die konzeptionelle Sicht der Datenstruktur als UML-Komponentendiagramm.

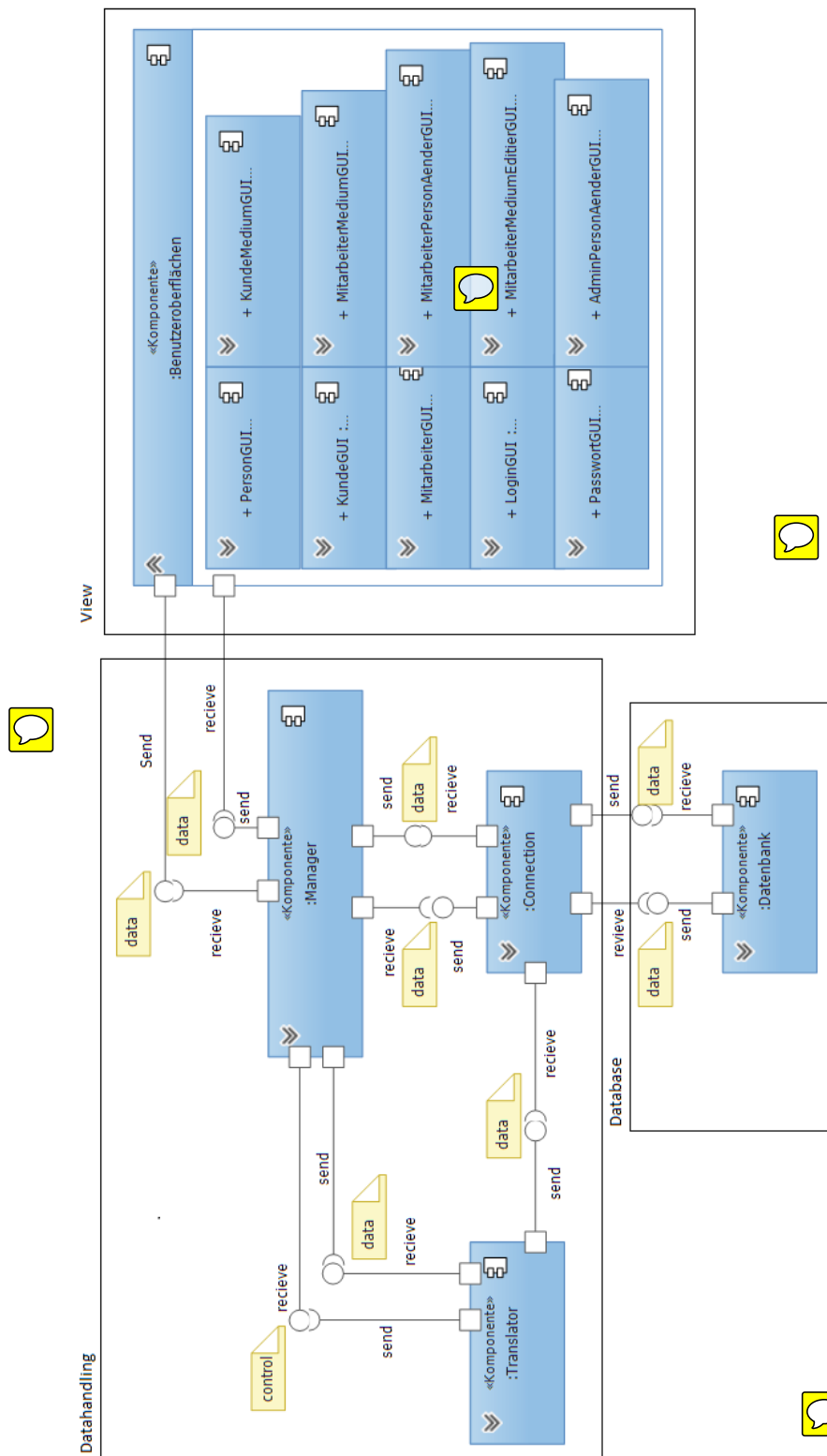


Abbildung: Konzeptionelle Sicht

4 Modulsicht

(bearbeitet von: JOSCHA CEPOK, ARTHUR NIEDZWIECKI)

4.1 Arbeitspakete

Hier sind die Arbeitspakete und die jeweils bearbeitende Person aufgeführt. Alle Arbeitspakete können parallel innerhalb einer Woche abgeschlossen werden.

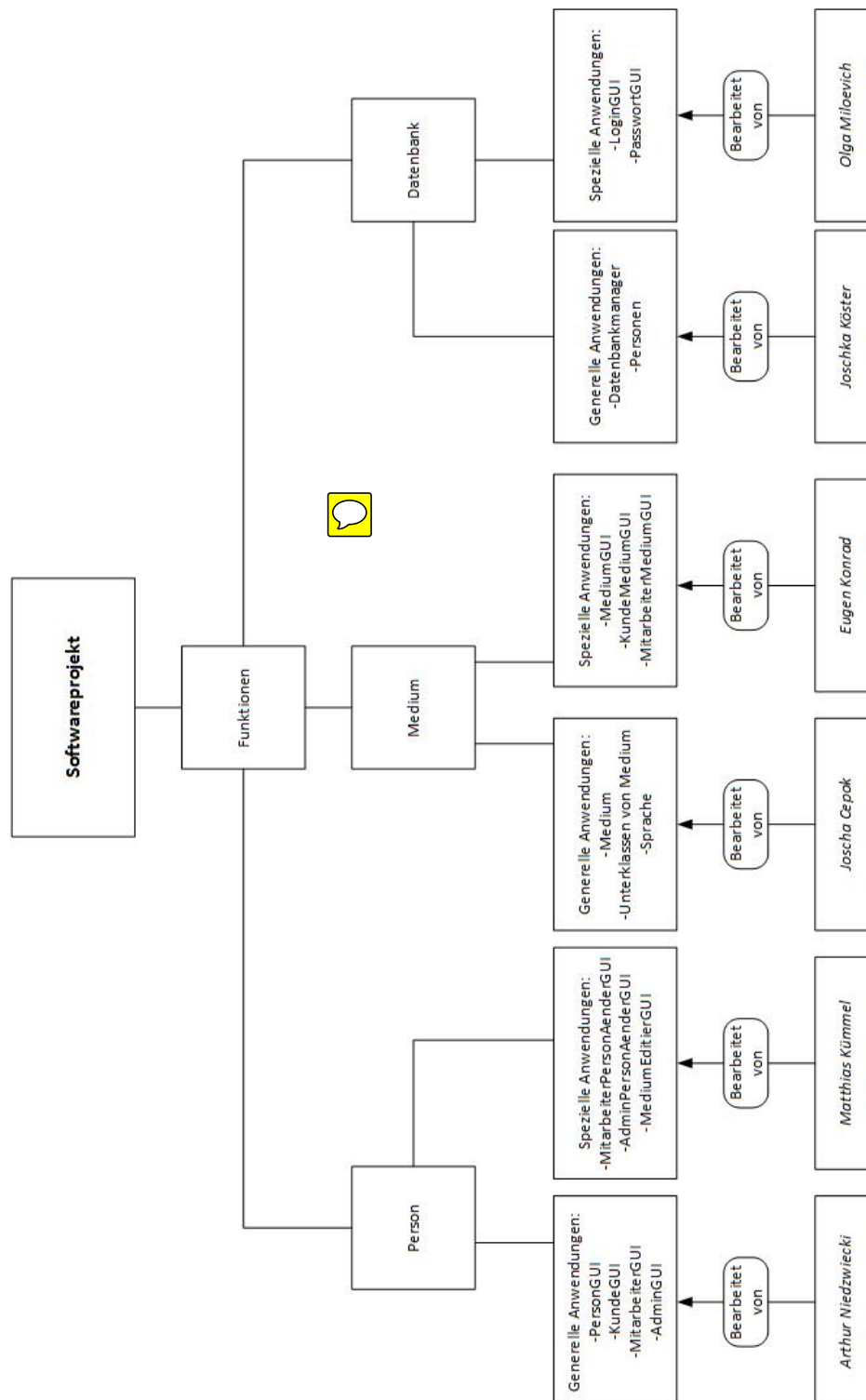
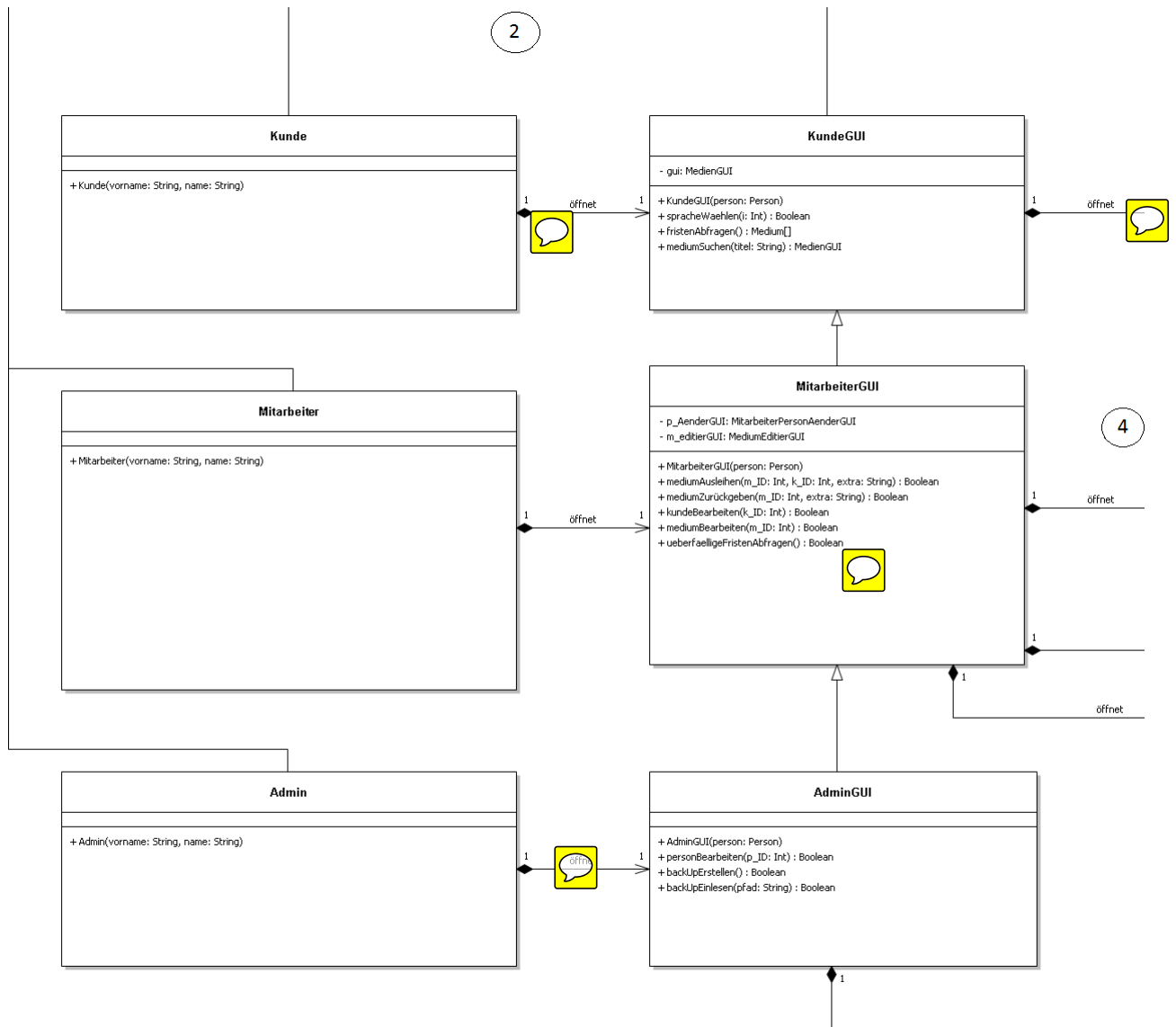
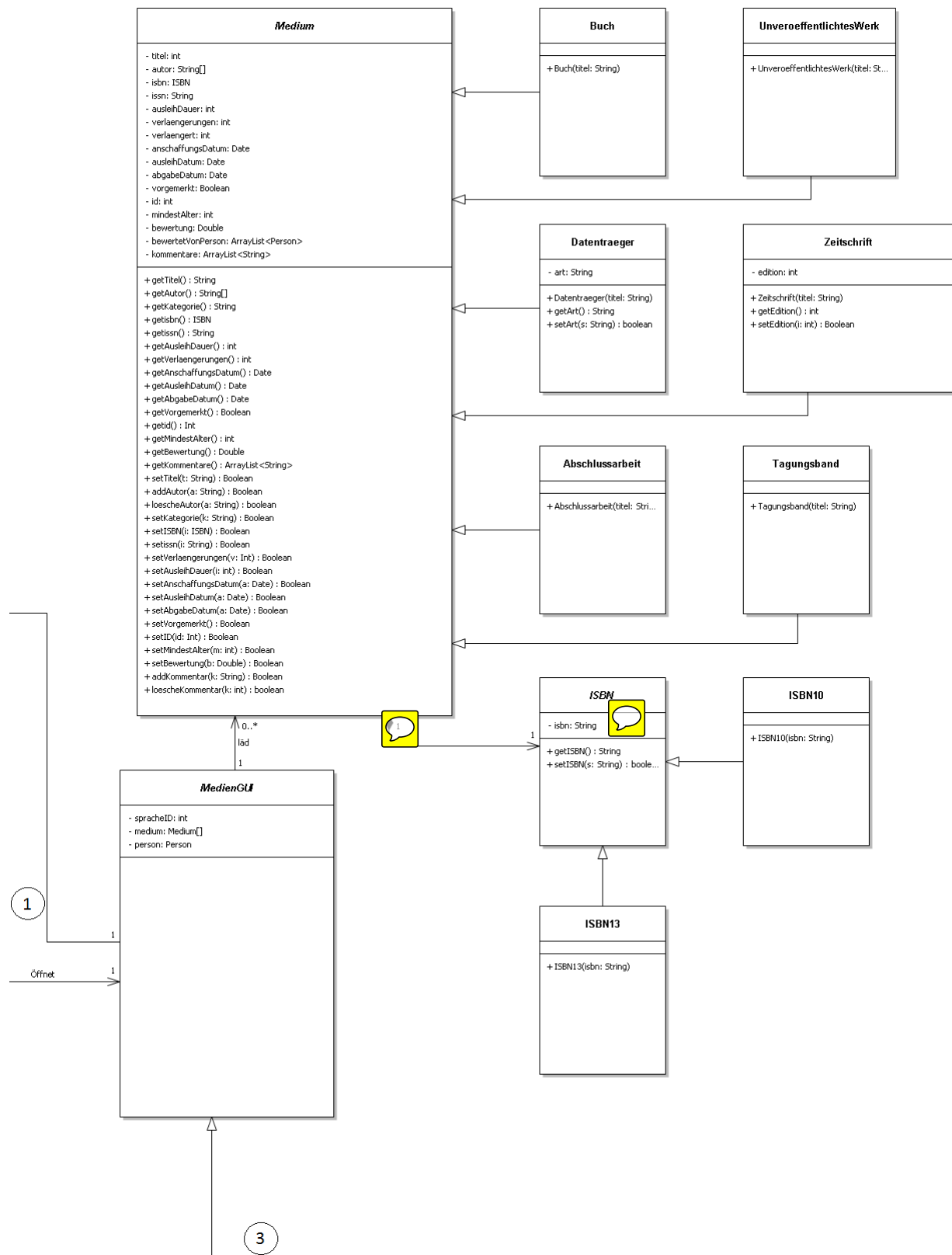


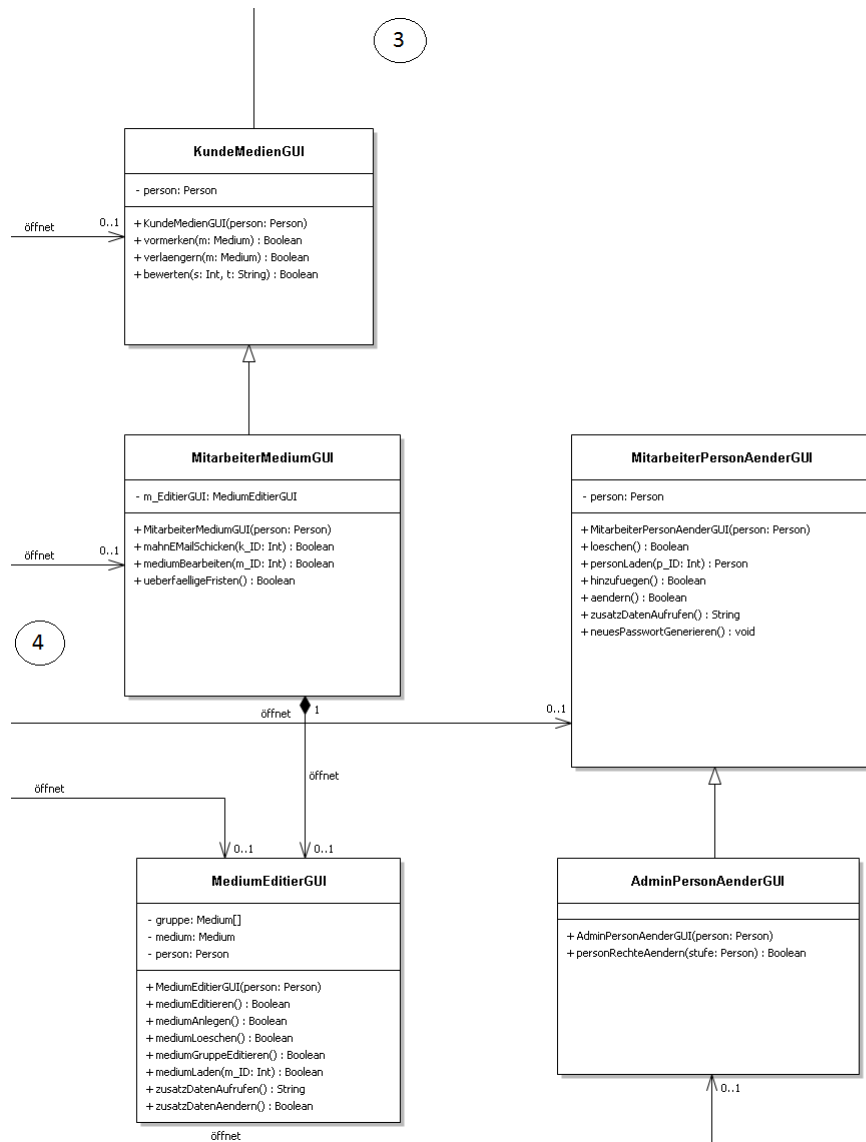
Abbildung: Arbeitspakete

4.2 Klassendiagramm

Das Klassendiagramm zur Software ist im Folgenden als UML-Klassendiagramm dargestellt. Das Diagramm ist, aufgrund der Größe auf 4 Blätter verteilt, die Verbindungen zum jeweils gegenüberliegenden Teil ist mit Nummern (1-4) markiert. Die Definitionen für die '-' und '+' Markierungen, sowie kursiv geschriebene Klassen und die Darstellung der Pfeile entsprechen den Konventionen von UML-Klassendiagrammen.







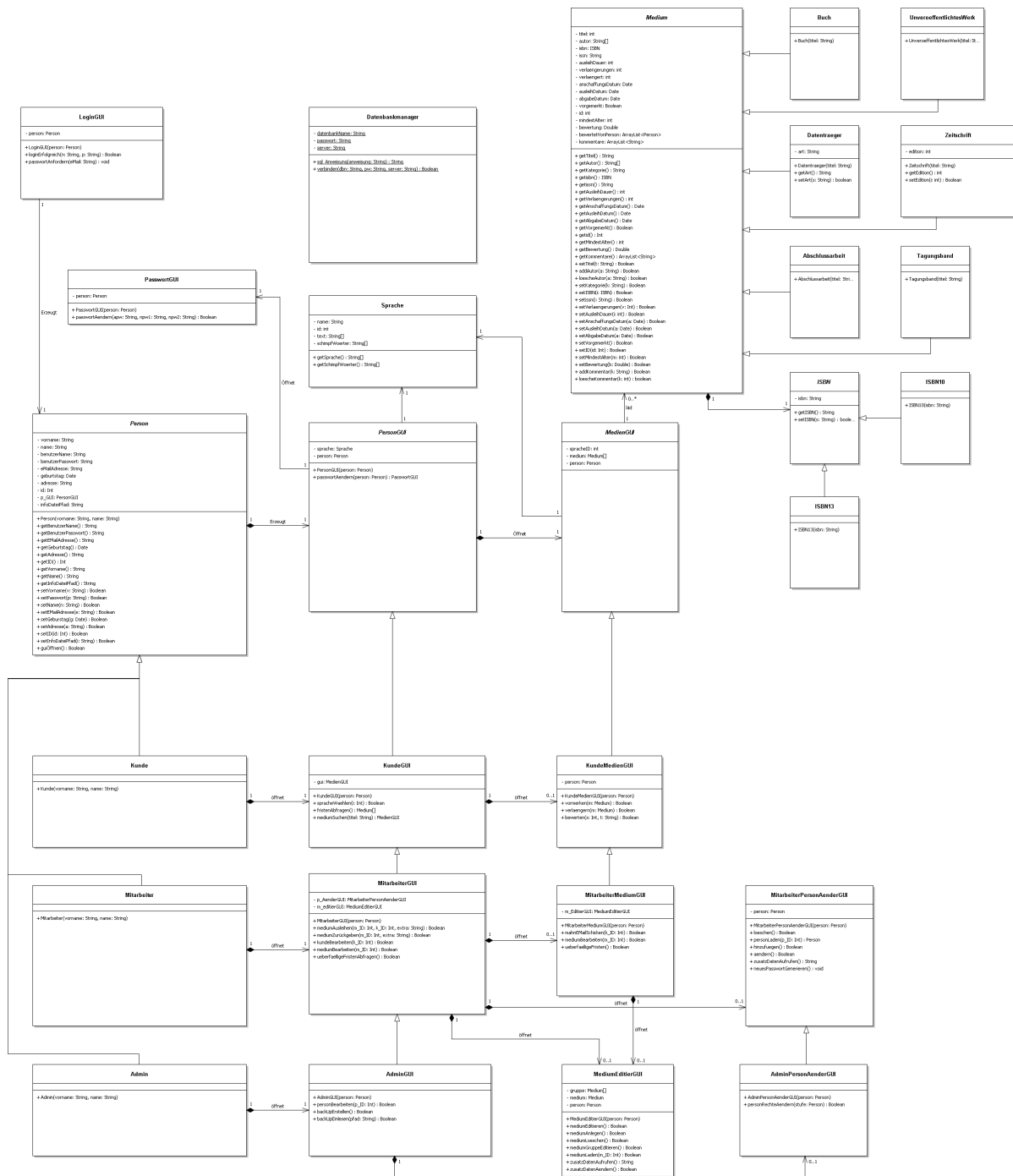




Abbildung: Klassendiagramm

4.3 Quellcode und Beschreibung

Die grobe Implementierung der Software ist hier durch die verwendeten Klassen und Methoden beschrieben. Methoden beinhalten lediglich den Kopf und einen minimalen Rumpf. Zu den Komponenten existieren in Javadoc  Erklärungen für die Funktionsweisen, sowie den Ein- und Ausgabeparametern.

Abschlussarbeit

```
/**
 *
 * @author Five_and_a_half_Men
 * Eine Unterklasse von Medium.
 */
public class Abschlussarbeit extends Medium{

    public Abschlussarbeit( String titel ){
        this.setTitel( titel );
    }
    
}
```

Admin

```
/**
 *
 * @author Five_and_a_half_Men
 * Eine Unterklasse von Person, mit den meisten Rechten aller Unterklassen von Person.
 */
public class Admin extends Person{

    public Admin( String vorname, String nachname ){
        super( vorname, nachname);
    }

}
```


AdminGUI

```
/**
 *
 * @author Five_and_a_half_Men
 * Die allgemeine Benutzeroberflaeche der Administratoren.
 */
public class AdminGUI extends MitarbeiterGUI{

    public AdminGUI( Person person ){
        super( person );
    }

    /**
     * Oeffnet die Benutzeroberflaeche AdminPersonAenderGUI fuer eine bestimmte Person.
     * @param p_ID ID der zu bearbeiteten Person.
     * @return True, bei Erfolg.
     */
    public boolean personBearbeiten( int p_ID ){
        return true;
    }

    /**
     * Erstell ein Backup der Datenbank auf dem Rechner des Admins.
     * @return True, bei Erfolg.
     */
    public boolean backUpErstellen(){
        return true;
    }

    /**
     * Liest und laed ein Backup auf die Datenbank.
     * @param pfad Der Dateipfad des Backups.
     * @return True, bei Erfolg.
     */
    public boolean backUpEinlesen( String pfad ){
        return true;
    }
}
```

AdminPersonAenderGUI

```
/**
 *
 * @author Five_and_a_half_Men
 * Die Benutzeroberflaeche , auf der eine Person vom Admin bearbeitet wird.
 * Sie erbt von MitarbeiterPersonAenderGUI.
 */
public class AdminPersonAenderGUI extends MitarbeiterPersonAenderGUI{

    public AdminPersonAenderGUI( Person person ){
        super( person );
    }

    /**
     * Aendert den Status der bearbeiteten Person.
     * @param stufe Der Rechte-Status zu dem Gewechselt wird.
     * @return True, bei Erfolg.
     */
    public boolean personRechteAendern( Person stufe ){
        return true;
    }
}
```

Buch

```
/**
 *
 * @author Five_and_a_half_Men
 * Eine Unterklasse von Medium.
 */
public class Buch extends Medium{

    public Buch( String titel ){
        this.setTitel( titel );
    }
}
```

Datenbankmanager

```
/**
 *
 * @author Five_and_a_half_Men
 * Ist fuer die Kommunikation zur Datenbank zustaending.
 */
public class Datenbankmanager {

    private static String datenbankName;
    private static String passwort;
    private static String server;

    /**
     * Gibt der Datenbank eine Anweisung um Informationen zu bekommen.
     * @param anweisung Die gegebene Anweisung, Anfrage von Information.
     * @return Die gewuenschte Information.
     */
    public static String sql_Anweisung( String anweisung ){
        return "";
    }

    /**
     * Stellt eine Verbindugn mit der Datenbak her.
     * @param dbn Name der Datenbank.
     * @param pw Passwort der Datenbank.
     * @param server Serveradresse.
     * @return True, bei Erfolg.
     */
    public static boolean verbinden( String dbn, String pw, String server ){
        return true;
    }
}
```

Datentraeger

```
/**
 *
 * @author Five_and_a_half_Men
 * Unterklasse von Medium.
 */
public class Datentraeger extends Medium{

    private String art;

    public Datentraeger( String titel ){
        this.setTitel( titel );
    }

    /**
     * Gibt die Art des Datentraegers zurueck.
     * @return Art des Datentraegers.
     */
    public String getArt(){
        return this.art;
    }

    /**
     * Setzt die Art des Datentraegers.
     * @param s Art des Datentraegers.
     * @return True, bei Erfolg.
     */
    public boolean setArt( String s ){
        return true;
    }
}
```

ISBN

```
/**
 *
 * @author Five_and_a_half_Men
 * Die Klasse der ISBN-Nummern von Medien.
 */
public abstract class ISBN {

    private String ISBN;

    /**
     * Gibt die ISBN-Nummer zurueck.
     * @return Die ISBN-Nummer
     */
    public String getISBN(){
        return this.ISBN;
    }

    /**
     * Setzt die ISBN-Nummer.
     * @param s Die gewuenschte ISBN-Nummer.
     * @return True, bei Erfolg.
     */
    public boolean setISBN( String s ){
        return true;
    }
}
```

ISBN10

```
/**
 *
 * @author Five_and_a_half_Men
 * Unterklasse von ISBN. Beinhaltet die 10-stelligen ISBN-Nummern.
 */
public class ISBN10 extends ISBN{

    public ISBN10( String isbn ){
        this.setISBN( isbn );
    }

}
```

ISBN13

```
/**
 *
 * @author Five_and_a_half_Men
 * Unterklasse von ISBN. Beinhaltet die 13-stelligen ISBN-Nummern.
 */
public class ISBN13 extends ISBN{

    public ISBN13( String isbn ){
        this.setISBN( isbn );
    }

}
```


Kunde

```
/**
 *
 * @author Five_and_a_half_Men
 * Eine Unterklasse von Person. Hat die geringsten Rechte aller Personen.
 * Entspricht dem allgemeinen Ausleiher.
 */
public class Kunde extends Person{

    public Kunde( String vorname, String nachname ){
        super(vorname, nachname);
    }

}
```

KundeGUI

```
/**
 *
 * @author Five_and_a_half_Men
 * Die Benutzeroberflaeche fuer Kunden.
 */
public class KundeGUI extends PersonGUI{

    private MedienGUI gui;

    public KundeGUI( Person person ){
        super( person );
    }

    /**
     * Setzt die Sprache der Texte auf der Benutzeroberflaeche auf die gewuenschte.
     * @param i Index der Sprache in der Liste der verfuegbaren Sprachen.
     * @return True, bei Erfolg.
     */
    public boolean spracheWaehlen( int i ){
        return true;
    }

    /**
     * Gibt die Liste der Medien mit ihren Abgabedaten zurueck.
     * @return Liste der Medien incl. Abgabedatum.
     */
    public Medium[] fristenAbfragen(){
        return new Medium[0];
    }

    /**
     * Gibt eine Liste von Medien zurueck, die nach den Suchparametern ausgewaehlt wurden.
     * @param titel Der Suchbegriff.
     * @return Die Benutzeroberflaeche MedienGUI mit den Suchergebnissen.
     */
    public MedienGUI mediumSuchen( String titel ){
        return gui;
    }
}
```

KundeMedienGUI

```
/**
 *
 * @author Five_and_a_half_Men
 * Die Benutzeroberflaeche des Kunden fuer eine Medienliste.
 */
public class KundeMedienGUI extends MedienGUI{

    private Person person;

    public KundeMedienGUI( Person person ){
        this.person = person;
    }

    /**
     * Merkt ein gewuenshtes Medium vor und setzt dessen vorgemerkt-Attribut
     * bei Erfolg auch true.
     * @param m Das gewuenschte Medium.
     * @return True, bei Erfolg.
     */
    public boolean vormerken( Medium m ){
        return true;
    }

    /**
     * Verlaengert das Abgabedatum einer ausgeliehenen Mediums.
     * @param m Das Medium.
     * @return True, bei Erfolg.
     */
    public boolean verlaengern( Medium m ){
        return true;
    }

    /**
     * Gibt eine Bewertung fuer ein Medium ab.
     * @param s Die Bewertung von 1 bis 5.
     * @param t Der Mediumtitel.
     * @return True, bei Erfolg.
     */
    public boolean bewerten( int s, String t ){
        return true;
    }
}
```

LoginGUI

```
/**
 *
 * @author Five_and_a_half_Men
 * Die Benutzeroberflaeche beim Einloggen einer Person.
 */
public class LoginGUI {

    private Person person;

    public LoginGUI( Person p ){
        this.person = p;
    }

    /**
     * Oeffnet die KundeGUI, wenn die Person ein Kunde ist ,
     * MitarbeiterGUI, wenn sie ein Mitarbeiter ist und AdminGUI,
     * wenn sie ein Admin ist. Fragt zunaechst Name und Passwort ab.
     * @param n Benutzername.
     * @param p Passwort.
     * @return True, bei Erfolg.
     */
    public boolean loginErfolgreich( String n, String p ){
        return true;
    }

    /**
     * Sendet das Passwort des Benutzers an dessen E-Mail-Adresse.
     * @param email Die E-Mail-Adresse des Benutzers.
     */
    public void passwortAnfordern( String email ){

    }

}
```

MedienGUI

```
/**
 *
 * @author Five_and_a_half_Men
 * Oberklasse aller Medien-GUIs von Kunde, Mitarbeiter und Admin.
 */
public abstract class MedienGUI {

    private int spracheID;
    private Medium[] medium;
    private Person p;

}
```

Medium

```
import java.util.ArrayList;
import java.util.Date;

/**
 *
 * @author Five-and-a-half-men
 * Ist die abstrakte Oberklasse aller Medien mit entsprechenden Attributen.
 * Von ihr erben alle Medien.
 */
public abstract class Medium {

    private String titel;
    private String[] autor = new String[3];
    private ISBN isbn;
    private String issn;
    private int ausleihdauer;
    private int verlaengerungen;
    private int verlaengert = 0;
    private Date anschaffungsDatum;
    private Date ausleihDatum;
    private Date abgabeDatum;
    private boolean vorgemerkt;
    private int id;
    private int mindestAlter;
    private double bewertung;
    private ArrayList<String> bewertetVonPerson = new ArrayList<String>();
    private ArrayList<String> kommentare = new ArrayList<String>();

    /**
     * Gibt den Titel des Mediums zurueck.
     * @return Der Titel des Mediums.
     */
    public String getTitle(){
        return this.titel;
    }

    /**
     * Gibt die Liste der Autoren/Herausgebern/Produzenten des Mediums zurueck.
     * @return Liste der Autoren/Herausgebern/Produzenten.
     */
    public String[] getAuthor(){
        return this.autor;
    }

    /**
     * Gibt die Kategorie des Mediums zurueck.
     * @return Kategorie des Mediums.
     */
    public String getKategorie(){
        return "";
    }

    /**
     * Gibt die ISBN-Nummer des Mediums zurueck.
     * @return Die ISBN-Nummer des Mediums.
     */
    public ISBN getIsbn(){ //<<<<ISBN return statt String
        return this.isbn;
    }

    /**
     * Gibt die ISSN-Nummer des Mediums zurueck.
     * @return Die ISSN-Nummer des Mediums.
     */
    public String getIssn(){
```

```
        return this.issn;
    }

    /**
     * Gibt die Ausleihdauer des Mediums zurueck.
     * @return Die Ausleihdauer des Mediums.
     */
    public int getAusleihDauer(){
        return this.ausleihdauer;
    }

    /**
     * Gibt die Anzahl der Verlaengerung des Mediums zurueck.
     * @return Die Anzahl der Verlaengerungen.
     */
    public int getVerlaengerungen(){
        return this.verlaengerungen;
    }

    /**
     * Gibt das Anschaffungsdatum des Mediums zurueck.
     * @return Das Anschaffungsdatum des Mediums.
     */
    public Date getAnschaffungsDatum(){
        return this.ananschaffungsDatum;
    }

    /**
     * Gibt das Ausleihdatum des Mediums zurueck.
     * @return Das Ausleihdatum des Mediums.
     */
    public Date getAusleihDatum(){
        return this.ausleihDatum;
    }

    /**
     * Gibt das Abgabedatum des Mediums zurueck.
     * @return Das Abgabdatum des Mediums.
     */
    public Date getAbgabeDatum(){
        return this.abgabeDatum;
    }

    /**
     * Gibt zurueck, ob das Medium vorgemerkt ist.
     * @return True, wenn das Medium vorgemerkt ist.
     */
    public boolean getVorgemerkt(){
        return this.vorgemerkt;
    }

    /**
     * Gibt die ID des Mediums zurueck.
     * @return Die ID des Mediums.
     */
    public int getId(){
        return this.id;
    }

    /**
     * Gibt das Mindestalter zurueck, das der Ausleiher fuer
     * die Ausleihe dieses Mediums erfuellen muss.
     * @return Mindestalter des Ausleihenden.
     */
    public int getMindestAlter(){
        return this.mindestAlter;
    }
}
```

```
}

/**
 * Gibt die Bewertung dieses Mediums zurueck.
 * @return Bewertung des Mediums 1 <= bewertung <= 5.
 */
public double getBewertung(){
    return this.bewertung;
}

/**
 * Gibt die abgegebenen Kommentare ueber das Medium als Liste zurueck.
 * @return Liste der Kommentare ueber das Medium.
 */
public ArrayList<String> getKommentare(){
    return this.kommentare;
}

/**
 * Setzt den Titel des Mediums.
 * @param t Titel des Mediums.
 * @return True, wenn der Titel erfolgreich geaendert wurde.
 */
public boolean setTitel( String t ){
    return true;
}

/**
 * Fuegt der Autorenliste des Mediums einen Autor hinzu.
 * @param a Name des Autors.
 * @return True, wenn das Hinzufuegen erfolgreich war.
 */
public boolean addAutor( String a ){ //addAuthor und removeAuthor statt setAuthor
    return true;
}

/**
 * Loescht aus der Autorenliste des Mediums einen Autor.
 * @param a Der zu loeschende Autor.
 * @return True, wenn der vorgang erfolgreich war.
 */
public boolean loescheAutor( String a ){
    return true;
}

/**
 * Setzt die Kategorie des Mediums.
 * @param k Kategorie, die das Medium haben soll.
 * @return True, wenn der vorgang erfolgreich war.
 */
public boolean setKategorie( String k ){
    return true;
}

/**
 * Setzt die ISBN-Nummer fuer das Medium.
 * @param i Die ISBN-Nummer.
 * @return True, bei Erfolg.
 */
public boolean setIsbn( ISBN i ){ //<<<<ISBN parameter statt string
    return true;
}

/**
 * Setzt die ISSN-Nummer fuer das Medium.
 * @param i Die ISSN-Nummer.
```



```
* @return True, bei Erfolg.
*/
public boolean setIssn( String i ){
    return true;
}

/**
 * Setzt die moegliche Dauer der Ausleihe fuer das Medium.
 * @param a Dauer der Ausleihe.
 * @return True, bei Erfolg.
 */
public boolean setAusleihDauer( int a ){ //<<<<param int statt String
    return true;
}

/**
 * Setzt die Anzahl der moeglichen Verlaengerungen fuer das Medium.
 * @param v Anzahl der moeglichen Verlaengerungen.
 * @return True, bei Erfolg.
 */
public boolean setVerlaengerungen( int v ){
    return true;
}

/**
 * Setzt das Datum der Anschaffung des Mediums.
 * @param d Datum der Anschaffung.
 * @return True, bei Erfolg.
 */
public boolean setAnschaffungDatum( Date d ){
    return true;
}

/**
 * Setzt das Ausleihdatum fuer das Medium.
 * @param a Datum der Ausleihe.
 * @return True, bei Erfolg.
 */
public boolean setAusleihDatum( Date a ){
    return true;
}

/**
 * Setzt das Datum der Rueckgabe des Mediums.
 * @param a Datum der Rueckgabe.
 * @return True, bei Erfolg.
 */
public boolean setAbgabeDatum( Date a ){
    return true;
}

/**
 * Deklariert das Medium als vorgemerkt.
 * @return True, bei Erfolg.
 */
public boolean setVorgemerkt(){
    return true;
}

/**
 * Setzt die ID des Mediums.
 * @param id Die ID des Mediums.
 * @return True, bei Erfolg.
 */
public boolean setId( int id ){
    return true;
}
```

```
    }

    /**
     * Setzt das Mindestalter fuer die Ausleihe des Mediums.
     * @param m Das Mindestalter.
     * @return True, bei Erfolg.
     */
    public boolean setMindestAlter( int m ){
        return true;
    }

    /**
     * Aktualisiert die Bewertung fuer das Medium.
     * @param i Die neu abgegebene Bewertung.
     * @return True, bei Erfolg.
     */
    public boolean setBewertung( double b ){
        return true;
    }

    /**
     * Fuegt der Liste der Kommentare fuer das Medium ein Kommentar hinzu.
     * @param k Der Kommentar.
     * @return True, bei Erfolg.
     */
    public boolean addKommentar( String k ){
        return true;
    }

    /**
     * Loescht ein Kommentar aus der Liste der Kommentare fuer das Medium.
     * @param k Der Index, an der das zu loeschende Kommentar in der Liste steht.
     * @return
     */
    public boolean loescheKommentar( int k ){
        return true;
    }
}
```

MediumEditierGUI

```
/**
 *
 * @author Five_and_a_half_Men
 * Die Benutzeroberflaeche des Mitarbeiters und Admins, auf der Medien editiert werden.
 */
public class MediumEditierGUI {

    private Medium[] gruppe;
    private Medium medium;
    private Person person;

    public MediumEditierGUI( Person person ){
        this.person = person;
    }

    /**
     * Bestaetigt den Editiervorgang.
     * @return True, bei Erfolg.
     */
    public boolean mediumEditieren(){
        return true;
    }

    /**
     * Legt ein neues Medium fuer die Datenbank an.
     * @return True, bei Erfolg.
     */
    public boolean mediumAnlegen(){
        return true;
    }

    /**
     * Entfernt das ausgewaehlte Medium aus der Datenbank.
     * @return
     */
    public boolean mediumLoeschen(){
        return true;
    }

    /**
     * Bearbeitet alle Medien der ausgewaehlten Mediengruppe.
     * @return True, bei Erfolg.
     */
    public boolean mediumGruppeEditieren(){
        return true;
    }

    /**
     * Laed ein Medium zur Bearbeitung.
     * @param m_ID ID des Mediums.
     * @return True, bei Erfolg.
     */
    public boolean mediumLaden( int m_ID ){
        return true;
    }

    /**
     * Laed die Zusatzdaten des Mediums in die Benutzeroberflaeche.
     * @return True, bei Erfolg.
     */
    public String zusatzDatenAufrufen(){
        return "";
    }

    /**
```

```
    * Legt die Veraenderung der Zusatzdaten frei.
    * @return True, bei Erfolg.
    */
    public boolean zusatzDatenAendern(){
        return true;
    }
}
```

Mitarbeiter

```
/**
 *
 * @author Five_and_a_half_Men
 * Unterklasse von Person mit mehr Rechten als Kunde, aber weniger als Admin.
 */
public class Mitarbeiter extends Person{

    public Mitarbeiter( String vorname, String name ){
        super( vorname, name );
    }

}
```

MitarbeiterGUI

```

/**
 *
 * @author Five_and_a_half_Men
 * Die Benutzeroberflaeche des Mitarbeiters.
 */
public class MitarbeiterGUI extends KundeGUI{

    private MitarbeiterPersonAenderGUI p_AenderGUI;
    private MediumEditierGUI m_editierGUI;

    public MitarbeiterGUI( Person person ){
        super( person );
    }

    /**
     * Ein Medium wird einem Kunden ausgeliehen.
     * @param m_ID ID des Mediums.
     * @param k_ID ID des Kunden.
     * @param extra Eventuelles Kommentar ueber das Medium (z.B. bei Beschaedigung)
     * @return True, bei Erfolg.
     */
    public boolean mediumAusleihen( int m_ID, int k_ID, String extra ){
        return true;
    }

    /**
     * Nimmt ein Medium aus der Liste der Ausleihliste eines Kunden
     * und gibt das Medium zum Verleih frei.
     * @param m_ID ID des Mediums
     * @param extra Eventuelles Kommentar ueber das Medium (z.B. bei Beschaedigung)
     * @return True, bei Erfolg.
     */
    public boolean mediumZurueckgeben( int m_ID, String extra ){
        return true;
    }

    /**
     * Oeffnet die MitarbeiterPersonAenderGUI um Kundendaten zu bearbeiten.
     * @param k_ID ID des Kunden.
     * @param extra Kommentar des Mitarbeiters ueber den Kunden
     * den nur andere Mitarbeiter und Admins sehen koennen.
     * @return True, bei Erfolg.
     */
    public boolean kundeBearbeiten( int k_ID ){
        return true;
    }

    /**
     * Oeffnet die MediumEditierGUI um Mediumdetails zu bearbeiten.
     * @param m_ID ID des Mediums.
     * @return True, bei Erfolg.
     */
    public boolean mediumBearbeiten( int m_ID ){
        return true;
    }

    /**
     * Oeffnet eine MitarbeiterMediumGUI mit den Medien,
     * dessen Abgabedaten ueberschritten sind.
     * @return True, bei Erfolg.
     */
    public boolean ueberfaelligeFristenAbfragen(){
        return true;
    }
}

```

MitarbeiterMediumGUI

```
/**
 *
 * @author Five_and_a_half_Men
 * Die Benutzeroberflaeche fuer Mitarbeiter , mit einer Mediumliste.
 */
public class MitarbeiterMediumGUI extends KundeMedienGUI{

    private MediumEditierGUI m_EditierGUI;

    public MitarbeiterMediumGUI( Person person ){
        super( person );
    }

    /**
     * Sendet eine Mahn-Mail an den Kunden.
     * @param k_ID ID des Kunden.
     * @return True, bei Erfolg.
     */
    public boolean mahnEMailSchicken( int k_ID ){
        return true;
    }

    /**
     * Oeffnet die MediumEditierGUI fuer ein Medium.
     * @param m_ID ID des Mediums.
     * @return True, bei Erfolg.
     */
    public boolean mediumBearbeiten( int m_ID ){
        return true;
    }

    /**
     * Oeffnet eine MediumGUI mit den Medien, deren Abgabefristen ueberfaellig sind.
     * @return True, bei Erfolg.
     */
    public boolean ueberfaelligeFristen(){
        return true;
    }
}
```

MitarbeiterPersonAenderGUI

```
/**
 *
 * @author Five_and_a_half_Men
 * Die Benutzeroberflaeche zur Bearbeitung von Personendetails.
 * Der Mitarbeiter kann lediglich Kundendetails bearbeiten.
 */
public class MitarbeiterPersonAenderGUI {

    private Person person;

    public MitarbeiterPersonAenderGUI( Person person ){
        this.person = person;
    }

    /**
     * Loescht die ausgewaehlte Person und oeffnet die
     * Startbenutzeroberflaeche der bearbeitenden Person.
     * @return True, bei Erfolg.
     */
    public boolean loeschen(){
        return true;
    }

    /**
     * Laed eine Person zu Bearbeitung.
     * @param p_ID ID der Person.
     * @return True, bei Erfolg.
     */
    public Person personLaden( int p_ID ){
        return person;
    }

    /**
     * Fuegt die ausgewaehlte Person der Datenbank hinzu und oeffnet die
     * Startbenutzeroberflaeche der bearbeitenden Person.
     * @return True, bei Erfolg.
     */
    public boolean hinzufuegen(){
        return true;
    }

    /**
     * Bestaetigt die Veraenderung der Personendaten und oeffnet die
     * Startbenutzeroberflaeche der bearbeitenden Person.
     * @return True, bei Erfolg.
     */
    public boolean aendern(){
        return true;
    }

    /**
     * Laed die Zusatzdaten der ausgewaehlten Person in die Benutzeroberflaeche.
     * @return Der Zusatztext der Person.
     */
    public String zusatzDatenAufrufen(){
        return "";
    }

    /**
     * Erstellt ein neues Passwort fuer diese Person, das vorherige wird
     * unwirksam gemacht und der Bearbeitende sieht das Passwort
     * auf der Benutzeroberflaeche.
     */
    public void neuesPasswortGenerieren(){
```



```
    }  
}
```

PasswortGUI

```
/**
 *
 * @author Five_and_a_half_Men
 * Die Benutzeroberflaeche fuer alle Personen. Hier loggt man sich ein.
 */
public class PasswortGUI {

    private Person p;

    public PasswortGUI( Person p ){
        this.p = p;
    }

    /**
     * Aendert das Passwort fuer diese Person.
     * Das Neue Passwort muss zwei mal eingegeben werden.
     * @param apw Altes Passwort.
     * @param npw1 Neues Passwort
     * @param npw2 Neues Passwort
     * @return True, bei Erfolg.
     */
    public boolean passwordAendern( String apw, String npw1, String npw2 ){
        return true;
    }
}
```

Person

```
import java.util.Date;
/**
 *
 * @author Five_and_a_half_Men
 * Die Oberklasse alle Personen. Von ihr erben Kunde, Mitarbeiter und Admin.
 */
public abstract class Person {

    private String vorname;
    private String name;
    private String benutzerName;
    private String benutzerPasswort;
    private String eMailAdresse;
    private Date geburtstag;
    private String adresse;
    private int id;
    private PersonGUI p_GUI;
    private String infoDateiPfad;

    public Person( String vorname, String name ){
        this.setVorname( vorname );
        this.setName( name );
    }

    /**
     * Gibt den Benutzernamen der Person zurueck.
     * @return Benutzernamen der Person.
     */
    public String getBenutzername(){
        return this.name;
    }

    /**
     * Gibt das Passwort der Person zurueck.
     * @return Passwort der Person.
     */
    public String getBenutzerPasswort(){
        return this.benutzerPasswort;
    }

    /**
     * Gibt die E-Mail-Adresse der Person zurueck.
     * @return E-Mail-Adresse der Person.
     */
    public String getEMailAdresse(){
        return this.eMailAdresse;
    }

    /**
     * Gibt den Genurtstag der Person zurueck.
     * @return Gebutstag der Person.
     */
    public Date getGeburtstag(){
        return this.geburtstag;
    }

    /**
     * Gibt die Adresse der Person zurueck.
     * @return Adresse der Person.
     */
    public String getAdresse(){
        return this.adresse;
    }

    /**
```

```
* Gibt die ID der Person zurueck.
* @return ID der Person.
*/
public int getID(){
    return this.id;
}

/**
 * Gibt den Vornamen der Person zurueck.
 * @return Vorname der Person.
 */
public String getVorname(){
    return this.vorname;
}

/**
 * Gibt den Nachname der Person zurueck.
 * @return Nachname der Person.
 */
public String getName(){
    return this.name;
}

/**
 * Gibt den Dateipfad in der Datenbank der Person zurueck.
 * @return Dateipfad in der Datenbank der Person.
 */
public String getInfoDateiPfad(){
    return this.infoDateiPfad;
}

/**
 * Setzt den Vornamen der Person.
 * @param v Neuer Vorname der Person.
 * @return True, bei Erfolg.
 */
public boolean setVorname( String v ){
    return true;
}

/**
 * Setzt das Passwort der Person.
 * @param p Neues Passwort.
 * @return True, bei Erfolg.
 */
public boolean setPassword( String p ){
    return true;
}

/**
 * Setzt den Nachnamen der Person.
 * @param n Neuer Nachname.
 * @return True, bei Erfolg.
 */
public boolean setName( String n ){
    return true;
}

/**
 * Setzt die E-Mail-Adresse der Person.
 * @param e Neue E-Mail-Adresse.
 * @return True, bei Erfolg.
 */
public boolean setEmailAdresse( String e ){
    return true;
}
```

```
/**
 * Setzt das Geburtsdatum der Person.
 * @param g Genurtstag der Person.
 * @return True, bei Erfolg.
 */
public boolean setGeburtstag( Date g ){
    return true;
}

/**
 * Setzt die Adresse der Person.
 * @param a Neue Adresse.
 * @return True, bei Erfolg.
 */
public boolean setAdresse( String a ){
    return true;
}

/**
 * Setzt die ID der Person.
 * @param a Neue ID.
 * @return True, bei Erfolg.
 */
public boolean setID( String a ){
    return true;
}

/**
 * Setzt den Datenpfad der Person in der Datenbank.
 * @param i Neuer Datenpfad in der Datenbank.
 * @return True, bei Erfolg.
 */
public boolean setInfoDateiPfad( String i ){
    return true;
}

/**
 * Oeffnet die KundeGUI, wenn die Person ein Kunde ist ,
 * die MitarbeiterGUI, wenn sie ein Mitarbeiter ist
 * und die AdminGUI, wenn die ein Admin ist.
 * @return True, bei Erfolg.
 */
public boolean gui.oeffnen(){
    return true;
}
}
```

PersonGUI

```
/**
 *
 * @author Five_and_a_half_Men
 * Die abstrakte Oberklasse aller Benutzeroberflaechen der Personen.
 * Von ihr erben KundeGUI, MitarbeiterGUI und AdminGUI.
 */
public abstract class PersonGUI {

    private Sprache sprache;
    private Person person;

    public PersonGUI( Person person ){
        this.person = person;
    }

    /**
     * Oeffnet die PasswortGUI fuer diese Person, sodass er sein Passwort aendern kann.
     * @param p Die Person.
     * @return Die PasswortGUI.
     */
    public PasswortGUI passwortAendern( Person person ){
        return new PasswortGUI( person );
    }

}
```

Sprache

```
/**
 *
 * @author Five_and_a_half_Men
 * Alle Komponenten der Sprache, die auf den Benutzeroberflaechen erscheint
 * sind hier beschrieben incl. der Texte, dem Namen der Sprache,
 * dem Index in der Liste der Sprachen und den Schimpfwoertern.
 */
public class Sprache {

    private String name;
    private int id;
    private String[] text;
    private String[] schimpfWoerter;

    /**
     * Gibt die Texte in der ausgewaehlten Sprache zurueck.
     * @return Liste der Texte der Benutzeroberflaechen in gewuenschter Sprache.
     */
    public String[] getSprache(){
        return text;
    }

    /**
     * Gibt die Schimpfwoerter der Sprache wieder.
     * @return Liste aller Schimpfwoerter dieser Sprache.
     */
    public String[] getSchimpfwoerter(){
        return schimpfWoerter;
    }

}
```

Tagungsband

```
/**
 *
 * @author Five_and_a_half_Men
 * Unterklasse von Medium.
 */
public class Tagungsband extends Medium{

    public Tagungsband( String titel ){
        this.setTitel( titel );
    }

}
```


Unveroeffentlicht

```
/**
 *
 * @author Five_and_a_half_Men
 * Unterklasse von Medium.
 */
public class Unveroeffentlicht extends Medium{

    public Unveroeffentlicht( String titel ){
        this.setTitel(titel);
    }

}
```

Zeitschrift

```
/**
 *
 * @author Five_and_a_half_Men
 * Unterklasse von Medium.
 */
public class Zeitschrift extends Medium{

    private int edition;

    public Zeitschrift( String titel ){
        this.setTitel( titel );
    }

    /**
     * Gibt die Edition der Zeitschrift zurueck.
     * @return Edition der Zeitschrift.
     */
    public int getEdition(){
        return this.edition;
    }

    /**
     * Setzt die Edition der Zeitschrift.
     * @param i Gewuenschte Editionsnummer.
     * @return True, bei Erfolg.
     */
    public boolean setEdition( int i ){
        return true;
    }
}
```

5 Datensicht

(bearbeitet von: JOSCHA CEPOK, ARTHUR NIEDZWIECKI)

Dieses Diagramm beschreibt den Zusammenhang zwischen den Klassen. Der Datenbankmanager ist statisch und damit von allen Klassen aus verfügbar.

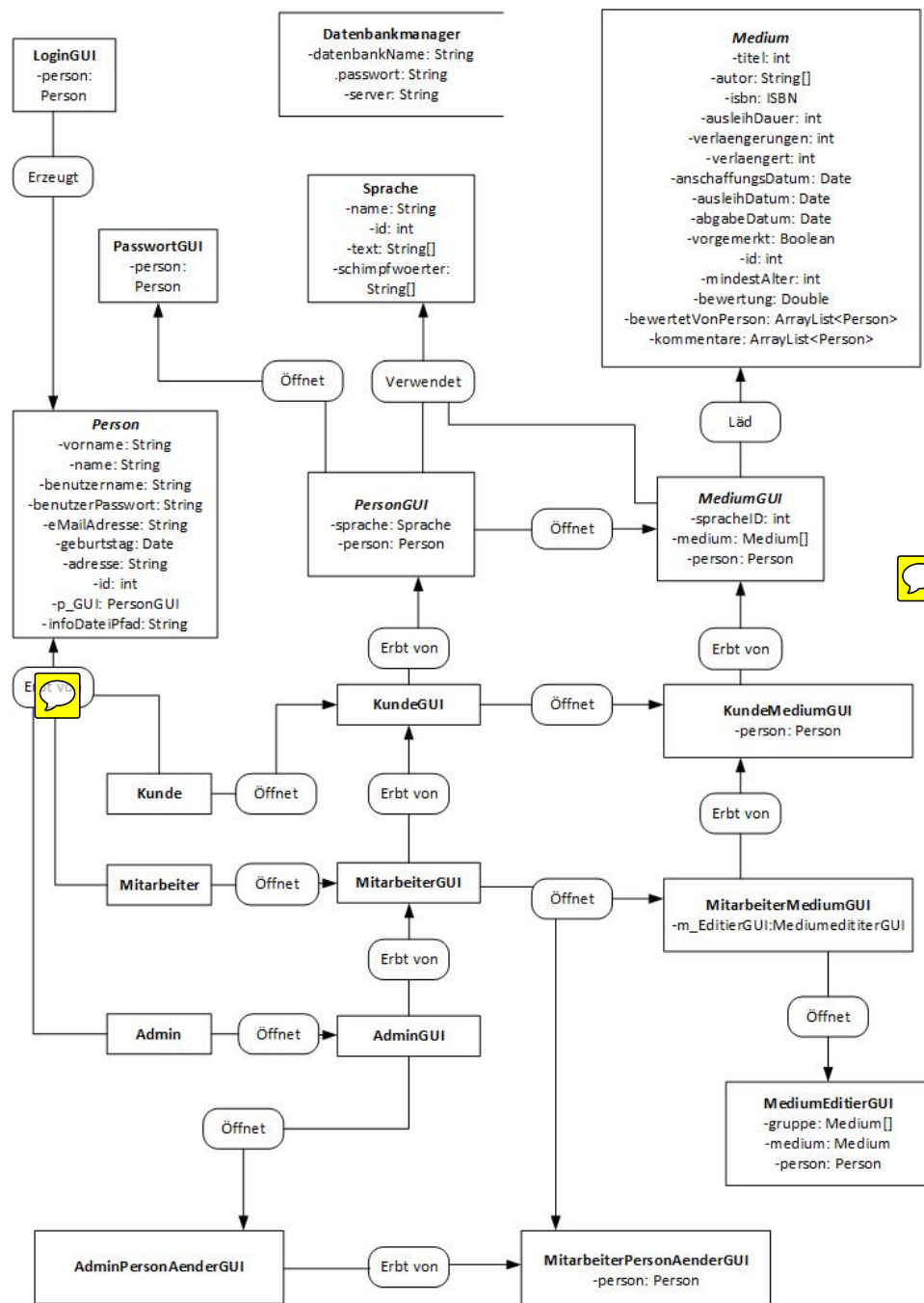


Abbildung: Datenstruktur

Medium

In der abstrakten Klasse Medium werden zunächst sämtliche Attribute, die ein Medium haben kann festgelegt, diese werden je nachdem welches Medium es tatsächlich ist entweder mit Daten gefüllt, oder auf null gesetzt. Welche Daten gesetzt werden oder nicht, wird jeweils im Konstruktor der Unterklassen festgelegt. Es gibt für jedes Attribut einen Getter und Setter. Die Getter liefern jeweils genau die Daten die abgespeichert wurden zurück. Der Setter liefern jeweils einen Boolean, ob das Überschreiben der Daten erfolgreich war, an dieser Stelle verzichten wir auf Exceptions, die spezifische Fehlermeldungen geben könnten, aus dem Boolean ist nur ableitbar, ob die Aktion erfolgreich oder nicht erfolgreich war. Die Methoden `setKommentar(String,Person)` und `setBewertung(Double,Person)` schreiben jeweils 1 Element in einen Array hinein, ob sie es hinzufügen oder verändern wird daran bewertet, ob diese Person bereits bewertet bzw. kommentiert hat, oder nicht.

Buch,Unveröffentlicht,Abschlussarbeit,Tagungsband

Diese 4 verschiedenen Medien brauchen nur die im Medium bereitgestellten Methoden und speichern die im Konstruktor geforderten Werte in die entsprechenden Attribute, alle Attribute die nicht gesetzt wurden sind automatisch null, dies ist bei Abfragen zu berücksichtigen und abzufangen, falls man auf die falschen Werte zugreift.

Datenträger

Als Datenträger bezeichnen wir zunächst erst einmal alle Medien, die einer sein könnten (CD,DVD,BlueRay,DVD, etc.) die entsprechende Art wird zunächst in einem String gespeichert. An dieser Stelle kann wenn die Art des Datenträger eine größere Rolle einnimmt der Datenträger zu einer Abstrakten Klasse gemacht werden und die entsprechenden Unterklassen dazu implementiert. Für unser Modell verzichten wir jedoch auf eine größere Unterscheidung der unterschiedlichen Datenträger.

Zeitschrift

Die Zeitschrift besitzt ein zusätzliches Attribut `edition`, welches die aktuelle Versionsnummer speichert. Zusätzlich dazu gibt es Getter und Setter für die Versionsnummer.

ISBN

Die Klasse ISBN ist eine abstrakte Oberklasse für ISBN10 und ISBN13, welche es ermöglichen soll eine ISBN von beiden Typen in Medium zu speichern ohne genauere Unterscheidungen zu treffen

ISBN10,ISBN13

ISBN10 und ISBN13 spezifizieren die Art der ISBN, die zu einem Medium gehört

Person

In der abstrakten Klasse Person werden zunächst alle für eine Person wichtigen Attribute festgehalten. Für alle Methoden gibt es Getter und Setter. Der String `infoDateiPfad` besitzt den Pfad zu einer Datei die auf dem Server liegt. In dieser Datei kann endlos viel zusätzlicher Text über eine Person gespeichert werden. Zusätzlich zu den Gettern und

Settern gibt es die Methode `guiÖffnen()`, welche für alle Unterklassen neu implementiert werden muss, damit sie die für die Person richtige GUI öffnet.

Kunde/Mitarbeiter/Admin

Diese 3 Personen besitzen jeweils keine eigenen Medien und sind nur für die strenge Unterscheidung zwischen den einzelnen Rechten notwendig.

PersonGUI

Die PersonGUI legt im Konstruktor das Layout dieser GUI fest. Der Konstruktor selbst wertet dafür wichtige Maße wie z.B. höhe/breite des Bildschirm aus und passt daraufhin das Layout dynamisch an. Die PersonGUI weiß zu welcher Person sie gehört, sowie sie die Sprache kennt auf der sie die Oberfläche anzeigt. Dies funktioniert, indem jedes Element der GUI eine ID besitzt und diese ID auf das Element im StringArray der Sprache zeigt und somit den richtigen Text lädt.

KundeGUI

Die KundeGUI erbt von der PersonGUI und fügt weitere Funktionen hinzu, diese geben der Person die Möglichkeit die Sprache mit einer aufklappbaren Leiste zu wählen, die aktuellen Fristen abzufragen, ein Medium zu suchen, sowie das Passwort zu ändern. Die Funktionen `fristenAbfragen()` und `MediumSuchen(String)` rufen beide jeweils eine Instanz der KundeMediumGUI auf. Dort wird entsprechend der Ergebnisse der Abfrage angezeigt was gesucht wurde. Die Abfragen selbst sind SQL Abfragen, welche direkt mit dem Server kommunizieren und in interpretierbare Fragmente zerlegt werden. Die Funktion `PasswortÄndern(Person)` öffnet die PasswortGUI.

MitarbeiterGUI

Die MitarbeiterGUI erbt von KundeGUI und erhält ein zusätzliches Attribut: `MitarbeiterPersonÄnderGUI`, welches es ermöglicht das ein Mitarbeiter auf eine Person zugreifen und die Datensätze ändern kann. Es gibt die Funktion `mediumAusleihen(Int,Int,String)` welche ein Medium ausleiht und anhand der ID's Einträge in die Datenbank macht. Die Funktion `mediumZurückgeben(Int,String)` nimmt in Medium zurück und trägt es bei der Person, welche es ausgeliehen hat aus. Die Funktion `kundeBearbeiten(Int)` ruft eine Instanz von `MitarbeiterPersonÄnderGUI` auf. Die Funktion `mediumBearbeiten(Int)` ruft eine Instanz von `MediumEditierGUI` auf. Die Funktion `überfälligeFristenAbfragen()` sucht in der Datenbank nach allen Medien mit überfälligen Fristen und ruft eine Instanz der `MitarbeiterMediumGUI` auf, welche sämtliche überfälligen Bücher anzeigt.

AdminGUI

Die AdminGUI erbt von MitarbeiterGUI und bietet zusätzliche Funktionalitäten. Die Funktion `personBearbeiten(Int)` ruft eine Instanz von `AdminPersonÄnderGUI` auf. Die Funktion `backUpErstellen()` erzeugt eine Textdatei, in der ein Abbild von dem aktuellen Zustand der Datenbank in einer Form, die wieder einlesbar ist. Die Funktion `backUpEinlesen(String)` liest aus einer Textdatei sämtliche Einträge aus und ÜBERSCHREIBT die aktuelle Datenbank mit sämtlichen Werten.

Datenbankmanager

Der Datenbankmanager hat 3 statische Attribute, die für die Verbindung mit der Datenbank wichtig sind und besitzt Funktionen zum Zugriff auf die Datenbank. Die Funktion `sql_Anweisung(String)` führt eine Anweisung durch und gibt ggf. wenn es eine Abfrage war den Ergebnisstring zurück. Die Funktion `verbinden(String,String,String)` verbindet mit den eingegebenen Daten mit der Datenbank und gibt bei Erfolg ein `true` zurück.

Sprache

In Sprache werden sämtliche Elemente einer Sprache als Stringarray gespeichert, sowie weitere Attribute zur Identifikation festgelegt. Diese werden denn von den einzelnen Oberflächen geladen und in die entsprechenden Felder eingefügt. Die Funktion `getSprache()` gibt sämtliche für die Oberflächen wichtigen Begriffe zurück. Die Funktion `getSchimpfWörter()` gibt einen String zurück, mit dem ein Wortfilter arbeiten kann.

MedienGUI

Die abstrakte Klasse MedienGUI legt das Layout fest. Das Layout wird anhand verschiedener Eigenschaften, wie z.B. die Höhe und Breite des Bildschirms festgelegt und dynamisch angepasst. Dazu speichert MedienGUI die Sprache sowie die aktuell angezeigten Medien ab.

KundeMedienGUI

Die KundeMedienGUI erbt von MedienGUI und besitzt folgende Funktionen. Die Funktion `vormerken(Medium)` merkt ein Medium für den Kunden vor. Die Funktion `verlängern(Medium)` verlängert sofern es möglich ist die Ausleihdauer. Die Funktion `Bewerten(Int,String)` gibt dem Kunden die Möglichkeit ein Medium zu bewerten, wenn er dieses Medium bereits einmal bewertet hat, werden die Daten aktualisiert.

MitarbeiterMediumGUI

Die MitarbeiterMediumGUI erbt von KundeMedienGUI und setzt ein neues Attribut, `MediumEditierGUI` fest. Die Funktion `mahnEMailSchicken(Int)` schickt eine Mail an den Kunden, welcher das Medium im Moment besitzt. Die Funktion `mediumBearbeiten(Int)` öffnet ein Fenster von der Instanz `MediumEditierGUI` und liest dort sämtliche Werte ein, die zu dem Medium gehören. Die Funktion `überfälligeFristen()` lädt sämtliche Medien, die überfällig sind in das Fenster.

MediumEditierGUI

Die MediumEditierGUI besitzt 2 Attribute, einmal das Medium, welches aktuell bearbeitet wird, dazu die Gruppe, dem das Medium angehört. Sie besitzt Felder in denen sämtliche Attribute eines Medium angezeigt werden und dazu folgende Funktionen: Die Funktion `mediumEditieren()` speichert sofern ein Medium ausgewählt ist alle Daten in die Datenbank ab. Die Funktion `mediumAnlegen()` fügt in die Datenbank die aktuellen Attribute mit einer neuen ID als Medium hinzu. Die Funktion `mediumLöschen()` löscht aus der Datenbank das ausgewählte Medium, sofern es nicht ausgeliehen ist. Die Funktion `mediumGruppenEditieren()` ändert für die gesamte Gruppe die Einträge in der Datenbank. Die Funktion `mediumLaden(Int)` lädt anhand der ID ein Medium aus der Datenbank. Die Funktion `zusatzDaenAufrufen()` lädt aus der entsprechenden Datei sämt-

liche Zusatzinformationen Die Funktion `zusatzDatenÄndern()` ändert die Zusatzdaten des Medium, die in einer zusätzlichen Datei gespeichert wurden.

MitarbeiterPersonÄnderGUI

Die `MitarbeiterPersonÄnderGUI` besitzt als Attribut, die Person die geändert werden soll und besitzt folgende Funktioalitäten: Die Funktion `ändern()` speichert sofern eine Person ausgewählt ist alle Daten in die Datenbank ab. Die Funktion `hinzufügen()` fügt in die Datenbank die aktuellen Attribute mit einer neuen ID als Person hinzu. Die Funktion `löschen()` löscht aus der Datenbank die ausgewählte Person, sofern es erlaubt ist. (z.B. wenn kein Medium mehr ausgeliehen ist) Die Funktion `personLaden(Int)` läd anhand der ID eine Person aus der Datenbank Die Funktion `zusatzDaenAufrufen()` läd aus der entsprechenden Datei sämtliche Zusatzinformationen Die Funktion `zusatzDatenÄndern()` ändert die Zusatzdaten der Person, die in einer zusätzlichen Datei gespeichert wurden.

AdminPersonÄnderGUI

Der `AdminPersonÄnderGUI` erbt von `MitarbeiterPersonÄnderGUI` und besitzt eine weitere Funktionalität. Die Funktion `personRechtÄndern(Person)` ändert die Rechte einer Person auf die angegebene Personenstufe.

LoginGUI

Die `LoginGUI` öffnet ein Fenster, welches folgende Funktionalitäten besitzt: Die Funktion `loginErfolgreich(String,String)` logt eine Person ein, sofern der Rückgabewert `true` ist. Die Funktion `passwortAnfordern(String)` schickt an die angegebene Email Adresse ein neues Passwort. `MitarbeiterPersonÄnderGUI`

PasswortGUI

Die `PasswortGUI` erzeugt für die übergebene Person ein neues Passwort und besitzt folgende Funktionalität. Die Funktion `passwortÄndern(String,String,String)` erzeugt sofern die Eingaben stimmen ein neues Passwort.

6 Ausführungssicht

[Entfällt in SWP-1](#)

7 Zusammenhänge zwischen Anwendungsfällen und Architektur

(bearbeitet von: MATTHIAS KÜMMEL)

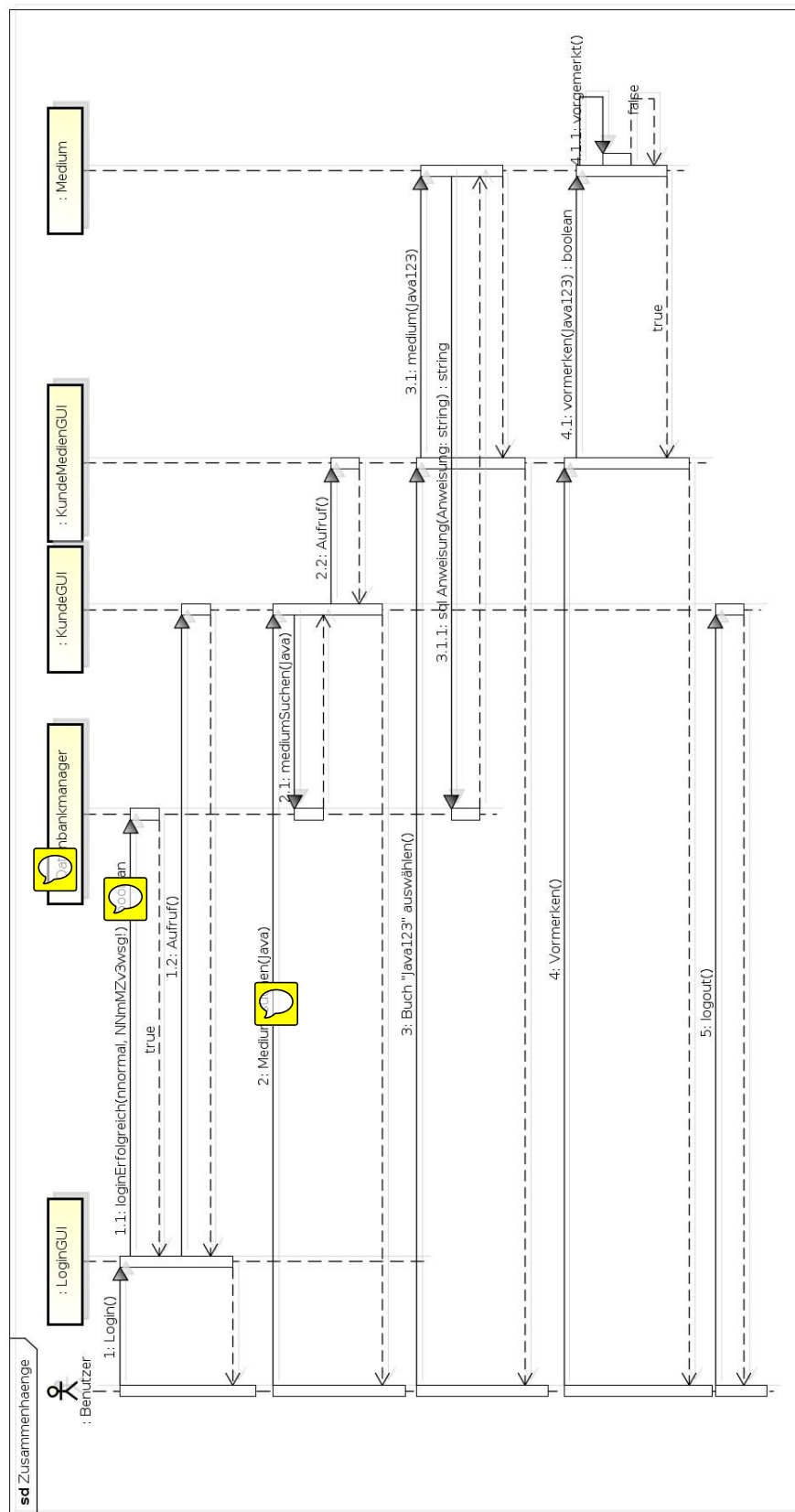


Abbildung: Sequenzdiagramme

Das Sequenzdiagramm stellt dar, wie sich ein Benutzer in das Büchereisystem einloggt, nach einem Buch sucht, dieses Vormerken lässt und sich wieder ausloggt.

Der Benutzer Norman Normal öffnet die Website und klickt auf den Login-Button. Hierzu wird das LoginGUI aufgerufen, in das der Benutzer über die Funktion *loginErfolgreich*(*n: string, p: string*) : *boolean* seinen Nutzernamen **nnnorma** und das Passwort **NNNmMZv3wsg!** eingibt. Die Eingaben werden mit dem Datenbankmanager, in welchem die Nutzerdaten gespeichert sind, abgeglichen. Stimmen die Daten mit der Datenbank überein, so wird der Benutzer zum KundenGUI weitergeleitet.

Hier durchsucht der Kunde durch einen Klick auf den Button „Medien suchen“, die Datenbank mit der Funktion *mediumSuchen*(*titel: String*) nach dem Buchtitel „Java,. Die gefundenen Bücher werden von dem Datenbankmanager als Liste im KundeMedienGUI zurückgegeben.

Der Kunde wählt das gewünschte Buch „Java123“, durch einen Klick aus, woraufhin das KundeMedienGUI den Buchtitel der Klasse Medium übergibt, welche über einen SQL-Aufruf *sql Anweisung*(*Anweisung: string*) : *string* an den Datenbankmanager dem Kunden die Daten zu dem Buch anzeigt.

Hier sieht der Kunde, dass das ausgewählte Buch derzeit verliehen ist und lässt es sich vormerken. Dazu wird geprüft, ob das Buch bereits vorgemerkt ist. Das KundeMedienGUI fragt über die Funktion *vormerken*(*Java123*) : *boolean* bei der Klasse Medium ab, ob das Buch bereits reserviert ist. Da dies noch nicht geschehen ist, wird diese Funktion mit „true“, beantwortet, wodurch das Buch für den Kunden reserviert ist.

Daraufhin loggt der Kunde sich über den Logout-Button des KundenGUI aus und wird damit vom System abgemeldet.

8 Evolution

Entfällt in SWP-1