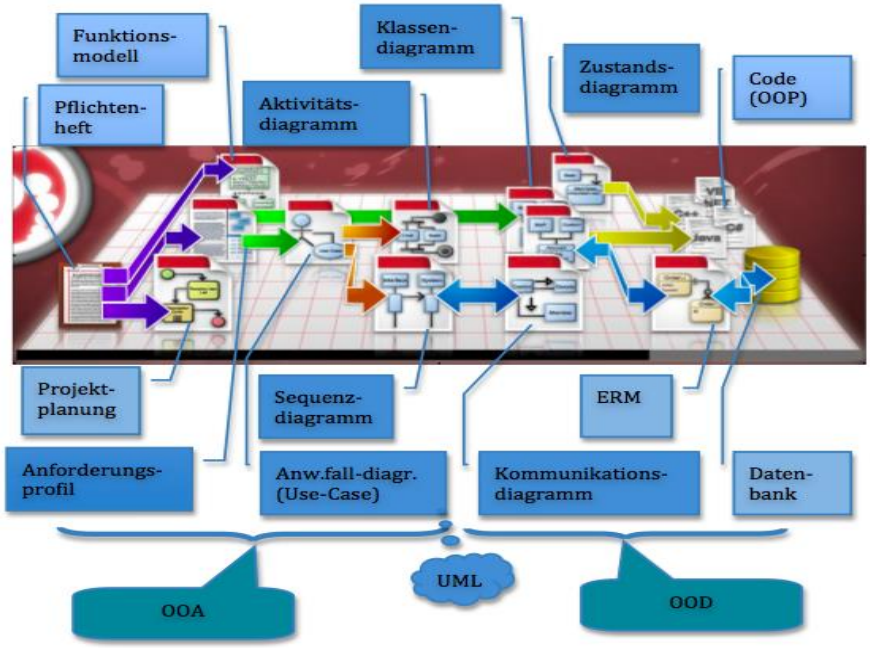
INFORMATIONS-TECHNIK
Ausstellungsstrasse 70
CH-8090 Zürich
Homepage www.tbz.ch
Telefon 044 446 96 60
Telefax 044 446 96 66
E-Mail admin.it@tbz.zh.ch



Entwurf von Anwendungen

Version 2.0

D. A. Waldvogel
28. 03. 2018

TECHNISCHE BERUFSSCHULE ZÜRICH

Inhaltsverzeichnis

1 Analyse und Entwurf

1.1 Grundlegendes

Analyse und Entwurf (auch Design) von Softwaresystemen sind umfangreiche und komplexe Themengebiete. Anhand eines Fallbeispiels sollen dabei die grundlegenden Schritte bei der Analyse und Entwurf praktisch erarbeitet werden.
Zahlreiche Methodiken sind in der Literatur beschrieben und werden in der Praxis eingesetzt. Im Folgenden lernen wir die *Verb/Substantiv-Methode* kennen und anschliessend verwenden wir die *CRC-Karten* um einen ersten Entwurf vorzunehmen.

1.2 Die Verb/Substantiv-Methode

Bei dieser Methode geht es ausschliesslich darum, die Klassen und Objekte für einen Problembereich sowie ihre Verknüpfungen und Interaktionen zu identifizieren. Die Substantive in der natürlichen Sprache beschreiben üblicherweise Dinge wie Menschen, Gebäude und so weiter. Die Verben beschreiben Aktionen wie schreiben, essen und so weiter. Aus diesen Konzepten der natürlichen Sprache können wir darauf schliessen, dass bei einem Programmierproblem die Substantive oft mit den Klassen und Objekten übereinstimmen, während die Verben oft mit den Dingen korrespondieren, die diese Objekte tun: also ihren Methoden. Wir brauche keine sehr lange Beschreibung, um diese Technik zu illustrieren. Es reicht üblicherweise eine Darstellung in ein paar Absätzen.

Das Beispiel, an dem wir diesen Entwurfsprozess verdeutlichen wollen, ist ein Buchungssystem für ein Kino.

1.3 Das Beispiel: ein Kinobuchungssystem

Wir nehmen an, dass wir uns in einer Situation befinden, in der wir eine Anwendung von Grund auf neu entwerfen müssen. Die Aufgabe ist, für ein Kinounternehmen ein System zu entwickeln, mit dem die Plätze für Kinovorstellungen gebucht werden können. Kinobesucher rufen häufig im Voraus an, um Plätze zu reservieren. Die Anwendung sollte dann in der Lage sein, die freien Plätze zu finden und für den Besucher zu reservieren

Wir nehmen an, dass wir schon einige Treffen mit den Betreibern des Kinounternehmens hatten, bei denen sie uns die Funktionalität beschrieben haben, die sie von dem System erwarten. (Die erwartete Funktionalität zu verstehen, sie zu beschreiben und diese Beschreibung mit dem Kunden abzustimmen, ist üblicherweise bereits ein Problem für sich. Auf diese Thematik gehen wir aber hier nicht weiter ein.)

Hier ist die Beschreibung, die wir für das Kinobuchungssystem formuliert haben.

Das Kinobuchungssystem sollte Platzreservierungen für mehrere Kinosäle verwalten. Jeder Kinosaal hat Plätze, die in Reihen angeordnet sind. Kinobesucher können Plätze reservieren und bekommen eine Reihen- und eine Platznummer zugewiesen. Sie können nach nebeneinanderliegenden Plätzen fragen.

Jede Platzreservierung gilt für eine bestimmte Vorstellung (also einen bestimmten Film zu einem bestimmten Zeitpunkt). Vorstellungen finden zu festgelegten Zeitpunkten an festgelegten Tagen statt und sind einem Kinosaal zugeteilt, in dem sie gezeigt werden. Das System speichert die Telefonnummer des Kinobesuchers.

Anhand dieses mehr oder weniger klar formulierten Textes können wir einen ersten Versuch vornehmen, auf Basis der Substantive und Verben Klassen und Methoden zu identifizieren.

1.4 Identifizieren von Klassen

Der erste Schritt beim Finden der Klassen besteht darin, dass wir durch die Beschreibung gehen und alle Substantive und Verben im Text markieren. Wenn wir dies tun, finden wir die folgenden Substantive und Verben (die Substantive sind in der Reihenfolge ihres Auftretens aufgeführt, die Verben sind den Substantiven zugeordnet, auf die sie sich beziehen):

Substantive:	**Verben:**
Kinobuchungssystem	verwaltet (Platzreservierung) speichert (Telefonnummer)

Übung 1.4.1 Finden Sie die Substantive und Verben und tragen Sie diese in den entsprechenden Spalten ein.

Die Substantive, die wir gefunden haben, geben uns einen ersten Hinweis auf die Klassen in unserer Anwendung. Als erste Näherung können wir eine Klasse für jedes Substantiv formulieren. Das ist kein exaktes Vorgehen – wir können später feststellen, dass wir noch zusätzlich Klassen benötigen oder einige der Substantive nicht benötigt werden. Das werden wir jedoch erst später herausfinden. Es ist auf jeden Fall wichtig, keine Substantive von Anfang an auszusortieren – wir haben noch nicht genügend Informationen, um eine fundierte Entscheidung treffen zu können.

Fast immer, wenn diese Übung mit Studenten durchgeführt wird, lassen die sofort einige Substantive weg. Zum Beispiel verzichtet einer der Studenten auf das Substantiv *Reihe* aus der obigen Beschreibung, weil es seiner Meinung nach nur eine Zahl ist, für das ein **int** reicht und keine Klasse benötigt wird. Auf dieser Stufe ist es wirklich wichtig, nicht den gleichen Fehler zu machen. Wir haben zu diesem Zeitpunkt einfach noch nicht genug Informationen, um zu entscheiden, ob Reihe ein int oder eine Klasse sein sollte. Diese Entscheidung können wir erst viel später treffen. Zurzeit gehen wir einfach nur mechanisch die Absätze durch und schreiben alle Substantive auf. Wir entscheiden noch nicht, welche Substantive „gut“ sind und welche nicht.

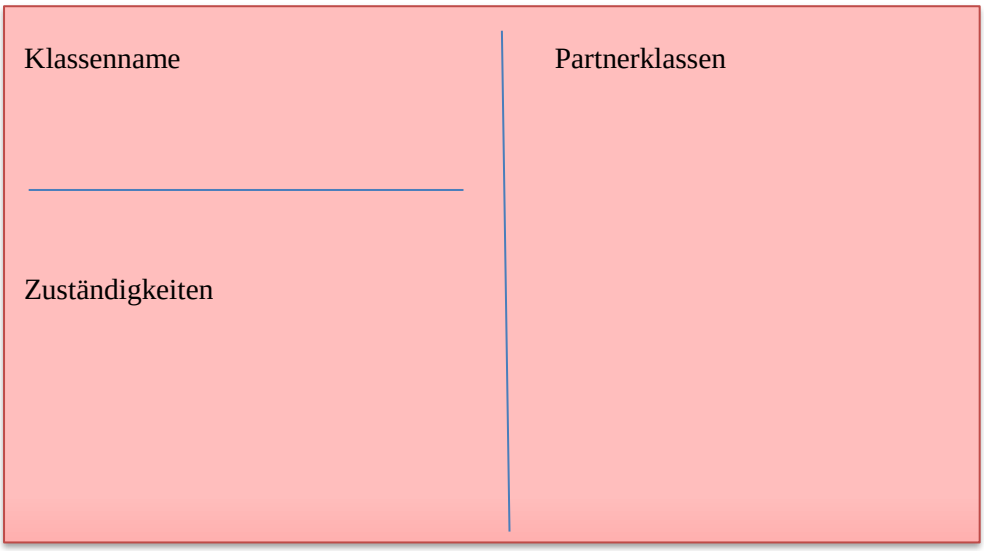
Sie haben möglicherweise festgestellt, dass alle Substantive im Singular (Einzahl) aufgeführt sind. Es ist bei Klassen typisch, dass ihre Namen im Singular und nicht im Plural formuliert sind. Beispielsweise würden wir eine Klasse immer eher **Kino** als **Kinos** nennen.

Übung 1.4.2 Gibt es gute Gründe in gewissen Fällen ein Klassenname im Plural zu definieren?

1.5 CRC-Karten

Der nächste Schritt in unserem Entwurfsprozess besteht darin, die Interaktionen zwischen unseren Klassen herauszuarbeiten. Dazu verwenden wir CRC-Karten[1]

CRC steht für Class/Responsibilities/Collaborators (Klasse/Zuständigkeiten/Partnerklassen). Die Idee besteht darin, für jede Klasse eine Pappkarte (normale Karteikarten sind gut geeignet) zu verwenden. Wichtig ist, dass tatsächlich für jede Klasse eine eigenständige, an fassbare Karte angelegt wird und nicht eine Computerdarstellung oder ein einzelnes Blatt Papier für alle. Jede Karte wird in drei Bereiche unterteilt: ein Bereich oben links, in den der Name der Klasse geschrieben wird; ein Bereich darunter, in dem die Zuständigkeiten der Klasse eingetragen werden; und ein Bereich rechts, in dem die Partnerklassen eingetragen werden (Klassen, mit denen diese Klassen kooperiert)

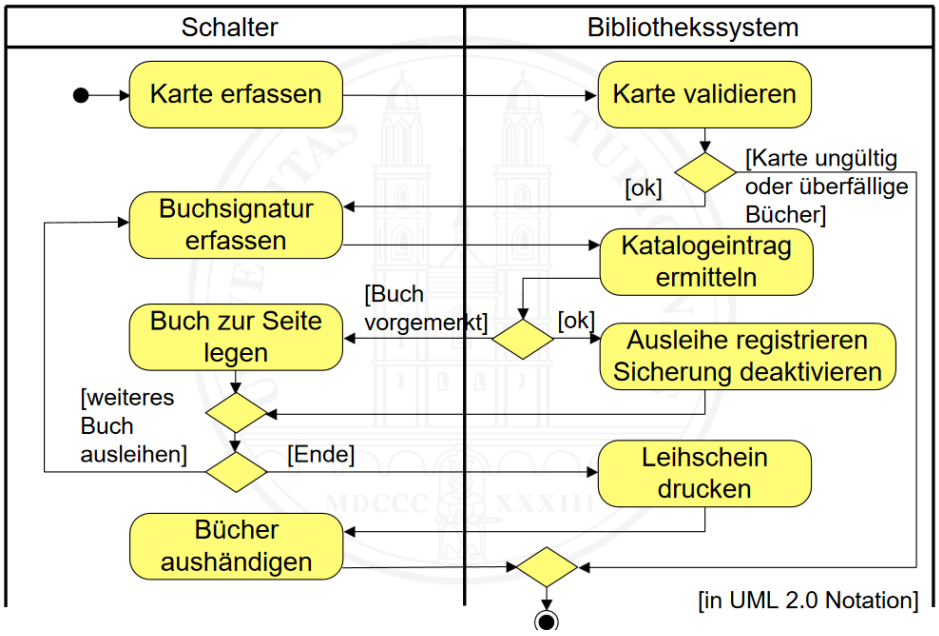


Übung 1.5.1 Erstellen Sie CRC-Karten für die Klassen im Kinobuchungssystem. An dieser Stelle müssen Sie lediglich die Klassennamen eintragen.

[1] CRC-Karten Artikel von Kent Beck: http://c2.com/doc/oopsla89/paper.html

1.6 Szenarios (Use Case)

Wir haben nun eine erste Näherung für die Klassen in unserem System und eine physische Repräsentation auf CRC-Karten. Um nun die notwendigen Interaktionen zwischen den Klassen in unserem System herauszufinden, spielen wir einige **Szenarios** durch. Ein Szenario ist ein Beispiel für eine Aktivität, die das System ausführen oder unterstützen muss.



Beispiel: Szenoriobeschreibung mit UML Aktivitätendiagrammen

Akteur(e): Benutzerin

Auslöser: Eine Benutzerin bringt ein Buch oder mehrere Bücher, das/die sie ausleihen möchte, zum Ausleiheschalter

Normalablauf:

1. Ausweiskarte der Benutzerin lesen und Angaben überprüfen
2. Signatur eines Buchs lesen und zugehörigen Katalogeintrag ermitteln
3. Ausleihe registrieren und Diebstahlsicherungsetikett deaktivieren
4. Wenn mehrere Bücher auszuleihen sind, mit den weiteren Büchern nach 2. und 3. verfahren
5. Leihschein drucken für alle ausgeliehenen Bücher
6. Der Benutzerin Bücher aushändigen, Vorgang abschließen

Beispiel: Szenariobeschreibung mit strukturiertem Text

Kinobuchungs - Szenarios in einzelnen Gruppen durchspielen.

Jedes Gruppenmitglied bekommt eine Klasse (oder eine kleine Anzahl an Klassen) zugewiesen und spielt die Rolle dieser Klasse, indem es laut ausspricht, was diese Klasse gerade tut. Während ein Szenario durchgespielt wird, halten die Teilnehmer auf den CRC-Karten alles fest, was die gerade agierende Klasse betrifft: wofür sie zuständig sein soll und mit welchen Klassen sie kooperieren muss.

Wir beginnen mit einem simplen Beispielszenario:

> Ein Kunde ruft beim Kino an und möchte zwei Plätze für *Die Verurteilten* heute Abend buchen. Der Kinoangestellte versucht nun mit dem Buchungssystem zwei Plätze zu finden und zu reservieren.

Da ein menschlicher Benutzer mit dem System interagiert (das durch die Klasse **Kinobuchungssystem** repräsentiert wird), fangen wir hier mit dem Szenario an. Als nächstes könnte Folgendes passieren:

- Der Benutzer (der Kinoangestellte) möchte alle Vorstellungen von *Die Verurteilten* finden, die heute angesetzt sind. Also können wir auf der CRC-Karte **Kinobuchungssystem** als Zuständiger eintragen: *kann Vorstellungen nach Filmtitel und Tag finden*. Wir können auch die Klasse **Vorstellung** als Partnerklasse eintragen.
- Wir müssen uns fragen: Wie findet das System die Vorstellung? Wen fragt es dazu? Eine Lösung könnte sein, dass das **Kinobuchungssystem** eine Sammlung von Vorstellungen hält. Das führt uns zu einer weiteren Klasse: der Sammlung. (Diese könnte später durch eine **ArrayList**, eine **LinkedList**, ein **HashSet** oder eine andere Sammlung implementiert werden. Diese Entscheidung können wir später treffen – momentan notieren wir nur eine **Sammlung**.) Dies ist ein Beispiel dafür, dass wir beim Durchspielen von Szenarios weitere Klassen entdecken können. Es kann immer wieder passieren, dass wir Klassen aus Implementierungsgründen hinzufügen müssen, die wir bis dahin übersehen haben. Wir fügen also bei den Zuständigkeiten auf der Karte für das **Kinobuchungssystem** ein: *speichert eine Sammlung von Vorstellungen*. Und **Sammlung** wird als Partnerklasse eingetragen.

Übung 1.6.1 Erstellen Sie eine CRC-Karte für die neu erkannte Klasse **Sammlung** und fügen Sie diese in ihr System ein.

- Wir nehmen an, dass es heute drei Vorstellungen gibt: eine um 17:30 Uhr, eine um 21:00 Uhr und eine um 23:30 Uhr. Der Angestellte teilt dem Kunden diese Zeiten mit und der Kunde wählt die Vorstellung um 21:30 Uhr. Der Angestellte möchte deshalb als Nächstes Informationen zu dieser Vorstellung abrufen (ob sie ausverkauft ist, in welchem Kinosaal sie läuft etc.) Also muss unser System in der Lage sein, die Details einer Vorstellung abzurufen und anzuzeigen. Spielen Sie dies durch. Die Person, die das Buchungssystem spielt, sollte die Person, die die Vorstellungen repräsentiert, nach den benötigten Details fragen. Daraufhin notieren Sie auf der Karte für das **Kinobuchungssystem**: *Ruft Details einer Vorstellung ab* und zeigt sie an, und auf der Karte Vorstellung: *Liefert Informationen über Kinosaal und Anzahl der freien Plätze*.
- Nehmen wir an, dass es reichlich freie Plätze gibt. Der Kunde wählt die Plätze 13 und 14 in Reihe 12. Der Angestellte nimmt die Buchung vor. Wir notieren auf der Karte **Kinobuchungssystem**: *Akzeptiert Platzreservierungen vom Benutzer*.
- Wir müssen durchspielen, wie eine Platzreservierung ablaufen soll. Eine Platzreservierung ist sicherlich an eine Vorstellung gebunden. Das **Kinobuchungssystem** sollte also die Vorstellung über die Reservierung informieren. Es delegiert das tatsächliche Durchführen der Reservierung an das Objekt

Vorstellung. Wir können also für Vorstellung notieren: *Kann Plätze reservieren*. (Sie haben vielleicht schon festgestellt, dass die Grenze zwischen Klasse und Objekt sehr leicht verschwimmt, wenn man mit der CRC-Karte umgeht; letztlich repräsentiert die Person, die eine Klasse repräsentiert, auch die Instanzen dieser Klasse. Das ist absichtlich so und im Allgemeinen auch kein Problem)

- Jetzt sind wir bei der Klasse **Vorstellung**. Sie hat die Anfrage bekommen, einen Platz zu reservieren. Wie geht sie mit dieser Anfrage um? Um Platzreservierungen vornehmen zu können, muss sie auf einer Repräsentation der Plätze in einem Kinosaal zugreifen können. Als nehmen wir an, dass eine Vorstellung eine Verbindung zu einem **Kinosaal**-Objekt hat. (Notieren Sie dies auf der Karte: *Speichert Kinosaal.* Dies Klasse wird damit auch zur Partnerklasse.) Der Kinosaal sollt vermutlich eine exakte Anzahl und Anordnung seiner Plätze kennen. (Wir können an dieser Stelle im Hinterkopf behalten oder auf einem Blatt Papier notieren, dass jede Vorstellung eine eigene Kopie eines Kinosaals haben sollte, damit eine Reservierung für eine Vorstellung nicht auch einen Platz in einer anderen Vorstellung blockiert. Darauf sollten wir achten, wenn eine **Vorstellung** erzeugt wird. Wir werden darüber später noch einmal nachdenken, wenn wir ein anderes Szenario durchspielen: *das Ansetzen neuer Vorstellungen.*) Vermutlich wird also eine Vorstellung die Reservierung wiederum an einen Kinosaal delegieren.
- Der Kinosaal hat nun die Anfrage bekommen, einen Platz zu reservieren. (notieren Sie dies auf der Karte: *Akzeptiert Reservierungsanfragen*.) Wie geht er damit um? Der Kinosaal könnte eine Sammlung seiner Plätze halten. Oder er könnte eine Sammlung von Reihen halten (und jede Reihe ist ein eigenen Objekt), und Reihen wiederum halten Plätze. Welche dieser Alternativen ist besser? Wenn wir bereits einige andere Szenarios im Auge haben, dann entscheiden wir uns vermutlich für en Ansatz mit den Reihen. Wenn ein Kunde beispielsweise vier Plätze nebeneinander in derselben Reihe reservieren möchte, dann könnten benachbarte Plätze einfacher zu finden sein, wenn sie alle in einer Reihe organisiert sind. Wir notieren also auf der Karte **Kinosaal**: *Speichert Reihen*. Und **Reihe** wird damit zur Partnerklasse.
- Wir notieren auf der Karte **Reihe**: *Speichert Sammlungen von Plätzen*. Und als neue Partnerklasse: **Platz.**
- Zurück zur Klasse **Kinosaal**. Wir haben noch nicht entschieden, wie sie mit der Reservierungsanfrage umgehen soll. Nehmen wir an, dass sie zwei Dinge tut: Sie findet die gewünschte Reihe und stellt an das **Reihe**-Objekt die Anfrage, die gewünschten Plätze zu reservieren.
- Als Nächstes notieren wir auf der Karte **Reihe**: *Akzeptiert Reservierungsanfrage für einen Platz*. Es muss dazu das entsprechende **Platz**-Objekt finden (wir können als Zuständigkeit notieren: *Kann Plätze über ihre Nummer finden*) und kann eine Reservierung dieses Platzes vornehmen. Es wird dies tun, indem es dem Platz-Objekt mitteilt, dass es nun reserviert ist.
- Wir können nun auf der Karte Platz notieren: *Akzeptiert Reservierungen*. Der Platz selbst kann sich merken, dass er reserviert ist. Wir notieren auf der Karte Platz: *Speichert Reservierungszustand (frei/reserviert*)

Übung 1.6.2 Spielen Sie dieses Szenario auf ihren Karten durch (mit einer Gruppe von Personen). Fügen Sie weitere Informationen hinzu, die Ihrer Meinung nach bisher ausgelassen wurden.

Sollte ein Platz auch speichern, wenn er reserviert hat? Es könnte den Namen des Kunden oder seine Telefonnummer speichern. Oder vielleicht sollten wir ein Kunden-Objekt erzeugen, sobald jemand eine Reservierung vornimmt, und eine Referenz auf dieses Kunden-Objekt im reservierten Platz speichern? Das sind spannende Fragen und wir werden versuchen, die beste Antwort über das Durchspielen von Szenarios zu finden.

Dies war ein erstes, einfaches Szenario. Wir müssen noch viele weitere durchspielen, bevor wir ein angemessenes Verständnis vom gewünschten System bekommen.

Das Durchspielen von Szenarios funktioniert am besten, wenn eine Gruppe von Personen um einen Tisch herumsitzt und die Karten nach Bedarf verschoben werden. Karten, die eng zusammenarbeiten, können auch räumlich enger angeordnet werden, um die Kopplung im System zu visualisieren.

Weitere durchzuspielende Szenarios würden unter anderem die folgenden sein:

- Ein Kunde möchte fünf Plätze nebeneinander reservieren. Spielen Sie möglichst exakt durch, wie diese fünf Plätze gefunden werden.
- Ein Kunde ruft an und teilt mit, dass er die Platznummern, die er gestern reservieren liess, vergessen hat. Könnten Sie die Platznummern bitte noch einmal herausfinden?
- Ein Kunde ruft an und möchte seine Reservierung stornieren. Er kann seinen Namen und die Vorstellung angeben, aber er hat die Platznummer vergessen.
- Eine Kundin, die bereits reserviert hat, ruft an. Sie möchte wissen, ob sie einen weiteren Platz direkt neben ihren bereits reservierten Plätzen bekommen kann.
- Eine Vorstellung wird abgesagt. Das Kino möchte alle Kunden anrufen, die für diese Vorstellung reserviert haben.

Die Szenarios sollten Ihnen ein gutes Verständnis von dem Teil des Systems vermitteln, das mit Platzsuche und Reservierung zu tun hat. Wir brauchen noch eine weiter Gruppe von Szenarios: solche, die mit dem Aufbau der Kinosäle und dem Aufsetzen von Vorstellungen zu tun haben. Einige mögliche wären:

- Das System soll für ein neues Kino eingerichtet werden. Das Kino hat zwei Kinosäle verschiedener Grösse. Saal A hat 26 Reihen mit jeweils 18 Plätzen. Saal B hat 32 Reihen, wobei die ersten sechs Reihen 20 Plätze haben, die nächsten 10 Reihen haben 22 Plätze und die restlichen Reihen 26 Plätze.
- Ein neuer Film kommt in die Kinos. Er wird in den nächsten zwei Wochen dreimal täglich gezeigt (um 16:40 Uhr, um 18:30 Uhr und um 20:30 Uhr). Die Vorstellungen müssen neu in das System eingefügt werden. Alle Vorstellungen laufen im Kinosaal A.

Übung 1.6.3 Spielen Sie dieses Szenario durch. Notieren Sie alle unbekannten Fragen auf einem getrennten Blatt Papier. Protokollieren Sie, welche Szenarios Sie bereits durchgespielt haben.

Übung 1.6.4 Welche weiteren Szenarios können Sie sich vorstellen? Schreiben Sie diese auf und spielen Sie sie durch.

Das Durchspielen von Szenarios erfordert einige Geduld und auch Übung. Sie sollten sich dafür genügend Zeit nehmen. Die hier genannten Szenarios erfordern mehrere Stunden.

Neulinge in dieser Technik neigen häufig zu Abkürzungen, indem sie nicht jedes Detail während des Durchspielens hinterfragen oder nicht ausreichend festhalten. Das ist gefährlich! Wir werden sehr bald dazu übergehen, dieses System in Java zu implementieren, und wenn Details offengeblieben sind, dann besteht die Gefahr, dass bei der Implementierung Entscheidungen ad hoc getroffen werden, die sich später als schlecht erweisen.

Neulinge neigen auch dazu, Szenarios zu vergessen. Wenn Teile nicht genügend durchdacht sind, bevor mit dem Entwurf und der Implementierung von Klassen begonnen wird, kann der Aufwand sehr gross werden, wenn bereits implementierte Teile wieder geändert werden müssen.

Die Szenarios gut durchzuspielen, sorgfältig alle notwendigen Schritte durchzuspielen und die Schritte ausführlich festzuhalten, bedarf einiger Übung und einer grossen Disziplin. Diese Aufgabe ist schwerer, als sie aussieht, und wichtiger, als Ihnen zurzeit klar ist.

1.7 Klassenentwurf

Jetzt ist der nächste Schritt zu machen: von den CRC-Karten zu Java-Klassen. Während der Übungen mit den CRC-Karten sollten Sie ein gutes Verständnis von der Struktur Ihrer Anwendung bekommen haben und davon, wie die Klassen zur Erfüllung der Aufgaben des Systems zusammenarbeiten. Möglicherweise hatten Sie die Situation, in denen neue Klassen eingeführt werden mussten (das gilt meist für Klassen, die interne Datenstrukturen repräsentieren), und möglicherweise hatte Sie auch Karten für Klassen, die gar nicht benutzt wurden. Wenn das der Fall war können Sie diese Karten nun entfernen.

Das Erkennen der Klassen für die Implementierung ist nun trivial: Die Karten zeigen uns den kompletten Satz an Klassen, den wir benötigen. Das Festlegen der Schnittstellen dieser Klassen (also öffentlichen Methoden) ist etwas schwieriger, aber auch dafür haben wir bereits einen wichtigen Schritt getan. Wenn das Durchspielen der Szenarios gut gelaufen ist, dann sollten die Zuständigkeiten, die für die Klassen notiert wurden, mehr oder weniger ihre öffentlichen Methoden beschreiben (und möglicherweise auch einige ihrer Datenfelder)

Übung 1.7.1 Erstellen Sie nun ein Klassendiagramm aus den CRC-Karten. Die Beziehungen zu den einzelnen Klassen schauen wir noch nicht genauer an

1.8 Klassenbeziehungen

Übung 1.8.1 Untersuchen Sie die Beziehungen zwischen den einzelnen Klassen. Verwenden Sie dazu das Merkblatt: *Beziehungen_in-UML.pdf*

1.9 UML-Diagramme

Übung 1.9.1 Erstellen Sie ein Use-Case-Diagramm des Kinobuchungssystems (KBS)

- Welche Akteure gibt es, die mit dem KBS-System arbeiten?
- Welche (Haupt-)Anwendungsfälle werden ausgelöst? 5 UC’s
- Ergänzen Sie mit «include»-Beziehungen und mind. einer «extend»-Beziehung!
- Das UC-Diagramm sollte max. 10 UC’s und max. 3 «UC-Ebene» aufweisen…

Übung 1.9.2 Erstellen Sie von: *Anfrage frei Plätze in der Vorstellung XYZ* ein Sequenzdiagramm

Übung 1.9.3 Erstellen Sie von: *Plätze reservieren* Kinobuchungssysems ein Aktivitätendiagramm

1.10 Kooperation

Traditionell werden Klassen von Einzelpersonen implementiert. Die meisten Programmierer entwickeln Ihren Quelltext allein, andere Personen werden erst einbezogen, wenn die Implementierung abgeschlossen ist und bewertet oder getestet werden soll.

In den letzten Jahren wurde verstärkt das Programmieren im Paar (pair programming) als eine Technik empfohlen, mit deren Hilfe bessere Quelltext (besser strukturiert und mit weniger Fehlern) erstellt werden kann.

Softwareentwicklung wird üblicherweise in Teams vorgenommen. Ein wohlstrukturierter objektorientierter Ansatz liefert eine gute Unterstützung für Teamarbeit, denn es erlaubt die Zerlegung einer Aufgabe in lose gekoppelte Komponenten (Klassen), die unabhängig voneinander implementiert werden können.

Obwohl die grundlegende Entwurfsarbeit im Team erfolgt ist, sollten Sie nun getrennt arbeiten. Wenn die Definition der Klassenschnittstellen und der Dokumentation gut gemacht wurde, sollten die Klassen unabhängig voneinander implementiert werden können. Klassen können also Programmierern zugewiesen werden, dieses allein oder im Paar umsetzen.

1.11 Prototyping

Anstatt eine Anwendung in einem einzigen grossen Schritt zu entwerfen und zu entwickeln, können *Prototypen* verwendet werden, um Teile eines Systems zu erkunden.

Ein Prototyp ist eine Version einer Anwendung, in der Teile diese Anwendung simuliert werden, um mit anderen Teilen experimentieren zu können. S. a. Prototyp Modul 120.

Die Funktionalität der Anwendung ist dann eventuell nicht vollständig implementiert. Stattdessen implementieren Sie Methoden, die ihre Aufgaben nur simulieren. Beispielsweise könnte bei Aufruf der Methode des Kinosystems, die einen freien Platz suchen soll, immer Platz 3, Reihe 15 geliefert werden, ohne dass wirklich eine Suche implementiert ist. Prototyping erlaubt uns, ein ausführbares (aber nicht voll funktionierendes) System schnell zu entwickeln, um Teile der Anwendung in der Praxis zu erproben.

Prototypen sind auch nützlich für einzelne Klassen, um die Entwicklung im Team zu unterstützen. Wenn verschiedene Teammitglieder an unterschiedlichen Klassen arbeiten, werden diese Klassen oft nicht gleichzeitig fertig. In einigen Fällen kann eine fehlende Klasse die Entwicklung und den Test anderer Klassen aufhalten. Dann kann es nützlich sein, für diese Klasse einen Prototyp zu schreiben.

Dieser Prototyp sollte schnell erstellt werden, damit Entwickler von Klassen, die diese Klassen benutzen müssen, bis zur Fertigstellung der vollständigen Implementierung weiterarbeiten können.
Zudem können mit Hilfe dieses Prototypen Schwierigkeiten und Probleme aufgedeckt werden, die in einer früheren Phase noch nicht bedacht wurden.

Übung 1.11.1 Umreissen Sie einen Prototyp für Ihr Kinosystem. Welche Klasse sollten zuerst implementiert werden und welche sollten vorläufig nur als Prototypen realisiert werden.

Übung 1.11.2 Implementieren Sie den Prototyp Ihres Kinosystems.

Quelle (in Teilen): Java lernen mit BlueJ, David J. Barnes, Michael Kölling 5. Auflage, PEARSON