Übungszettel UML

# Objektorientierte Analyse

Arbeite dich in die Grundprinzipien der Objektorientierten Analyse (OOA) ein.

* Ziele
  + Anforderungen eines Auftraggebers zu ermitteln und zu beschreiben
* Requirements-Engineering (Anforderungsermittlung)
  + Anforderungen an ein neues Softwareprodukt ermitteln, spezifizieren, analysieren, validieren und eine fachliche Lösung ableiten
  + verallgemeinern Konzepte der Objektorientierung
  + stellen graphische Notationen bereit
  + betten die Konzepte in einen methodischen Rahmen ein
  + führen zu einem integrierten Modell der funktionalen Anforderungen, in dem alle relevanten Systemaspekte berücksichtigt sind
* Strukturelle Modellierung
  + zur Modellierung der inneren Struktur eines Systems
  + Klassendiagramm und Objektdiagramm
  + Untersuchung der Realität -> vereinfachte Abbildung
  + im Klassendiagramm werden alle beteiligten Klassen und deren Beziehungen abgebildet
  + Im Objektdiagramm werden Objekte des Klassendiagramms zu einem bestimmten Zeitpunkt während der Ausführung dargestellt
* Dynamische Modellierung
  + Mit Anwendungsfällen (Use Cases) werden die extern beobachteten Funktionen spezifiziert, das heißt das, was ein Anwendungssystem einem Benutzer anbieten soll
  + Ein Akteur ist dabei eine außerhalb des Systems liegende Einheit, die an der Interaktion beteiligt sind -> kann ein Mensch oder ein technisches System sein (Betriebssystem, Drucker, …)

<https://de.wikipedia.org/wiki/Objektorientierte_Analyse_und_Design#Objektorientierte_Analyse>

Dokumentiere die wesentlichen Aspekte sowie deine persönlichen Lernergebnisse.

Nach der Beschäftigung mit dieser Aufgabe musst du wissen, was OOA ist, wozu man sie braucht, wie und wann man sie im Softwarenentwicklungsprozess einsetzt.

# Anforderungsanalyse und Anwendungsfälle

Arbeite dich in die Grundprinzipien der Anforderungsanalyse sowie der USE-Case-Analyse (Diagramm + textuelle Beschreibung + Mockup) ein.

* Grundlagen
  + Teil des Systementwicklungsprozesses sowie ein Teil der Business-Analyse
  + Ziel ist es, die Anforderungen des Auftraggebers an das zu entwickelnde System zu ermitteln, zu strukturieren und zu prüfen
  + Ergebnis wird meistens in einem Lastenheft dokumentiert oder bei einer agilen Softwareentwicklung resultiert daraus ein Product Backlog
* Vorgehensweisen
  + Anforderungen werden gesammelt
  + durch Analyse soll ein gemeinsames Verständnis hergestellt werden
  + Anforderungen werden textlich oder in Modellen dokumentiert, d.h. spezifiziert
  + Danach wird geprüft, ob das Ganze noch stimmig ist
  + Anforderungskriterien:
    - vollständig
    - eindeutig definiert/abgegrenzt
    - verständlich beschrieben
    - atomar
    - identifizierbar
    - einheitlich dokumentiert
    - nachprüfbar
    - rück- oder vorwärtsverfolgbar
    - konsistent
  + Strukturierung und Klassifizierung der Anforderungen
    - abhängig
    - zusammengehörig
    - rollenbezogen
  + Prüfung und Bewertung
    - korrekt
    - machbar
    - notwendig
    - priorisiert
    - nutzbar, nützlich
* Werkzeuge
  + Anwendungsfall-Modellierung: Erfassung und Darstellung der funktionalen Systemanforderungen aus Sicht von Akteuren
  + Interviewtechnik: Anwender werden in einem vorgegebenen, formalisierten Verfahren befragt
  + Dialog Design Modellierung: Struktur eines Nutzerdialogs mit Bildschirmmasken zu modellieren
  + Systemverhaltensmodelle: Anforderungen eines Systems mit einem Modell präzisieren
  + Kosten-Nutzen-Analyse bei Anforderungen: Priorisierung der Anforderungen
  + Einsatz von Kreativitätstechniken: Brainstorming, Wechsel der Perspektive, Walt Disney Methode
  + Einsatz von Beobachtungskriterien: Feldbeobachtung
* Dokumentation
  + die erhobenen Anforderungen werden in einer bestimmten Form festgehalten
  + solide Basis für Validierung von Anforderungen
  + Quelle für die Erhebung von Anforderungen
  + unterstützt das Anforderungsmanagement
  + ermöglicht genauere Schätzungen für Planung von Ressourcen
  + spezifiziert Rahmenbedingungen und Schlüsselaspekte
  + Testfälle lassen sich ableiten
  + Änderungsmanagement
  + wichtige Informationsquelle bei Systemnutzung und Systemwartung
  + unterstützt Vertragsverwaltung

**Diagramm:**

* Komponenten
  + Akteure: Nutzer, die mit einem System interagieren. Können Personen, Organisationen oder auch externe Systeme sein.
  + System: bestimmte Abfolge von Aktionen und Interaktionen zwischen Akteuren und dem System
  + Ziele: Diagramm sollte die Aktivitäten und Varianten darstellen, die zur Verwirklichung des Ziels führten
* Symbole:
  + Anwendungsfälle: Horizontal geformte Ovale, die die verschiedenen möglichen Anwendungszwecke eines Benutzers darstellen.
  + Akteure: Strichmännchen zur Darstellung der Personen, die die Anwendungsfälle tatsächlich umsetzen
  + Assoziationen: Linie zwischen Akteur und Anwendungsfällen.
  + Systemgrenzen: Kästchen. Alle Anwendungsfälle außerhalb des Kästchens liegen außerhalb der Systemgrenzen.
  + Pakete: UML-Form zur Gruppierung unterschiedlicher Elemente. Darstellung als Dateiordner.

**Textuelle Beschreibung:**

stellt zusätzliche Informationen zur Verfügung

* Vorbedingungen die für den Use-Case erfüllt sein müssen
* Nachbedingungen die erfüllt sein müssen, wenn der Use-Case fertig ist
* Hauptfluss: Szenario, das am häufigsten eintritt
* Alternative Flüsse: Szenarien die nicht so häufig sind oder nicht normal.

**Mockup:**

Einfache Zeichnungen der Benutzeroberfläche.

Vorteile:

* Fokus auf wichtige, inhaltliche und strukturelle Fragen
* kreative Prozess wird nicht durch kompliziertes Arbeiten mit einem Tool unterbrochen. Kann auch auf Papier gezeichnet werden.
* Man kann sie gemeinsam mit Kunden, Analysten und Fachbereichen erstellen und Dinge schnell verändern.

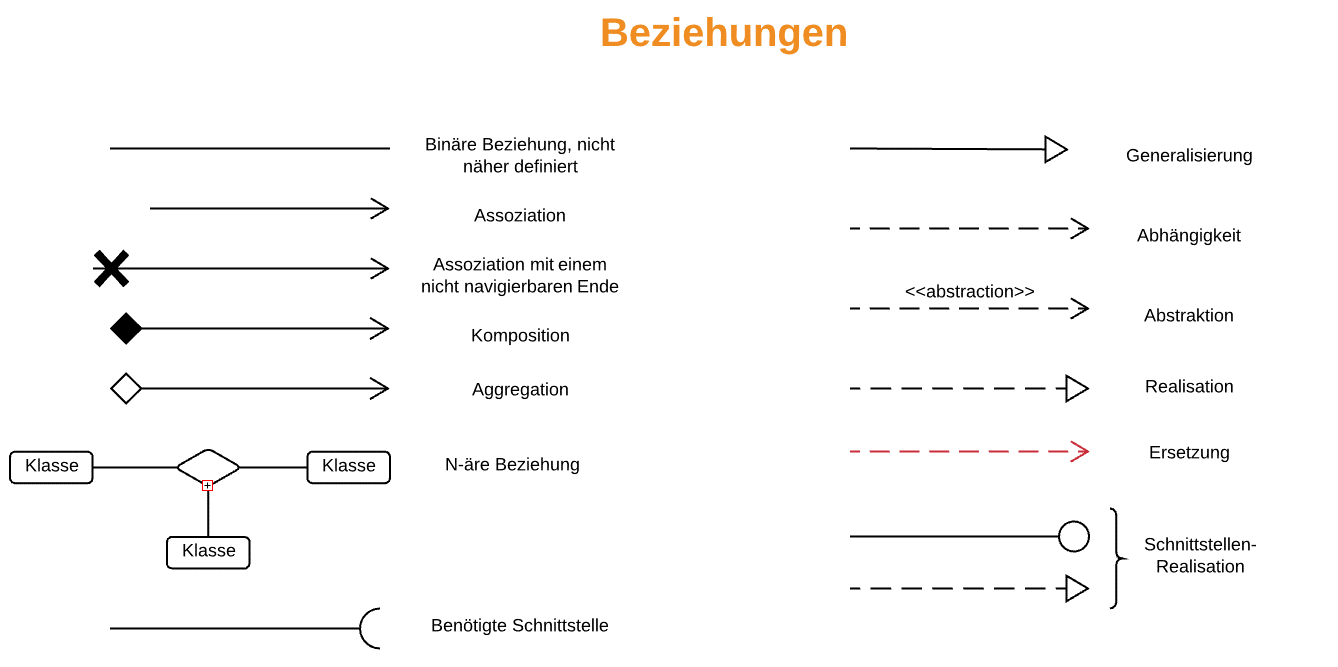
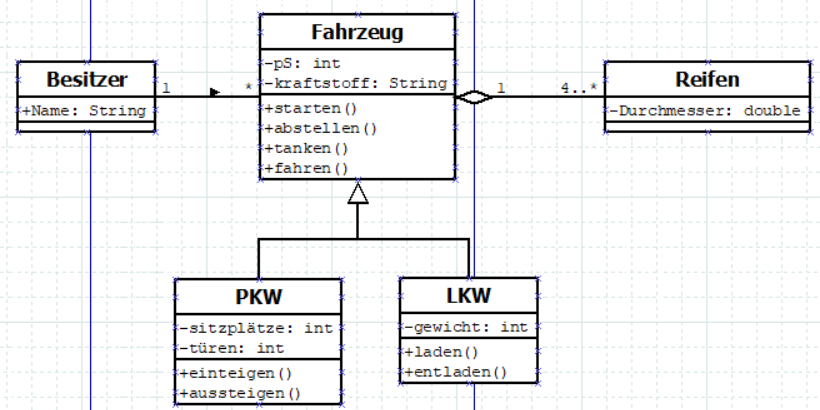
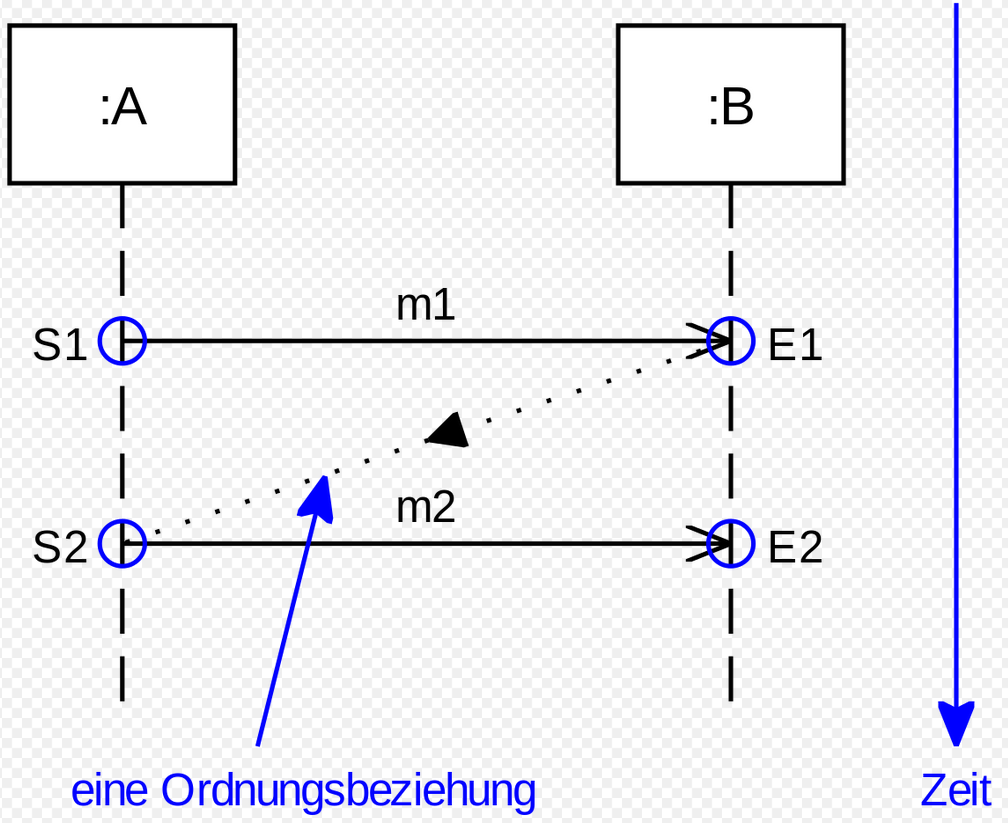
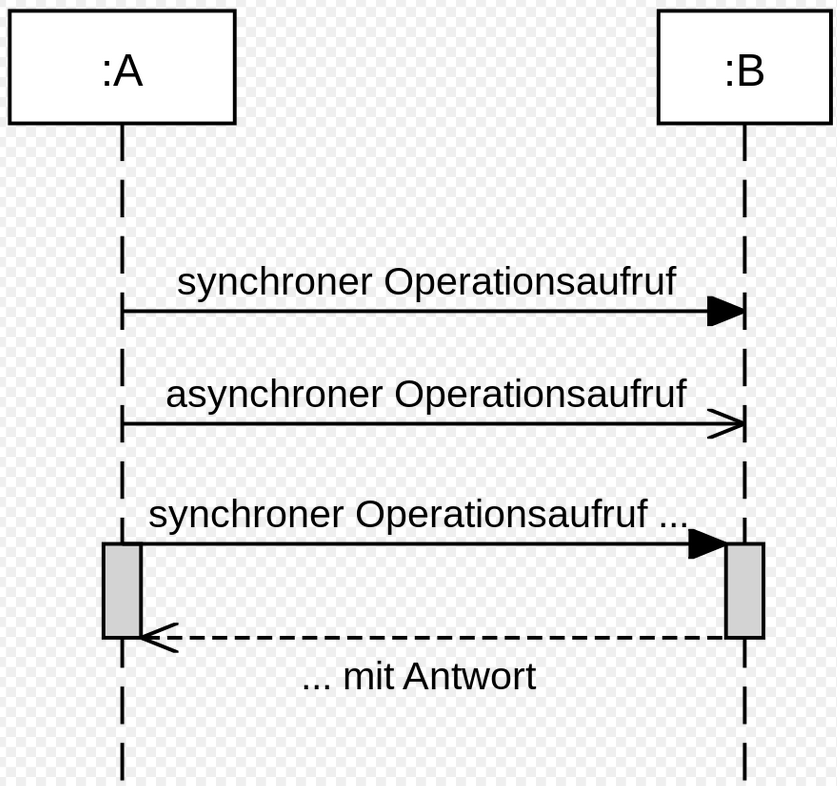
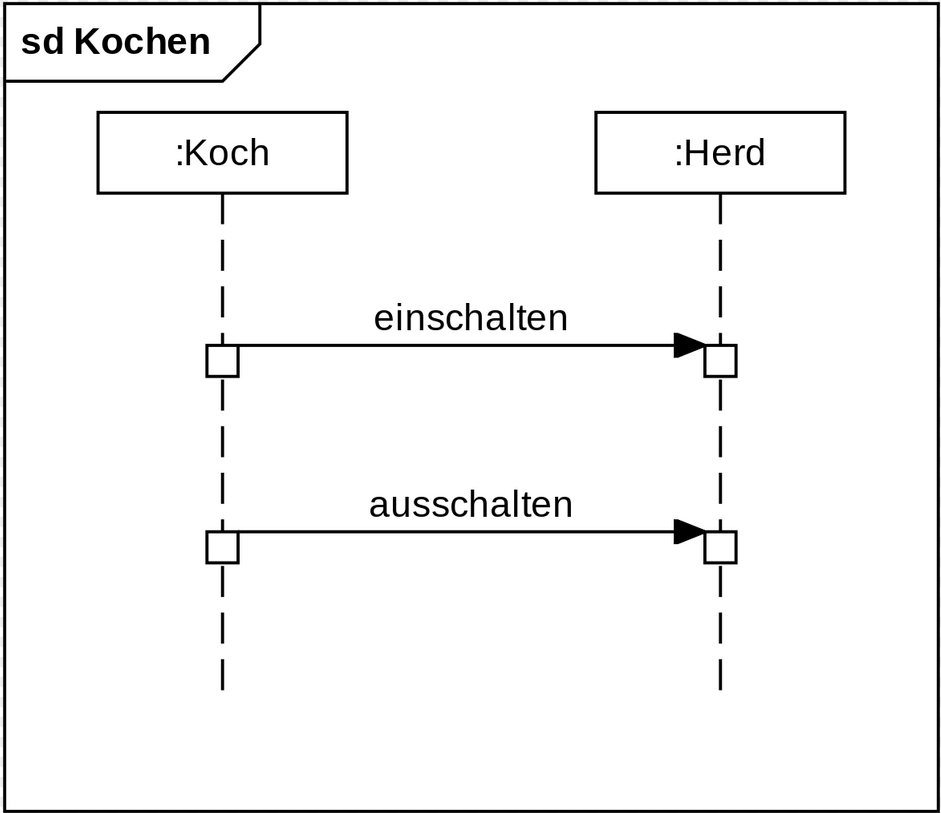
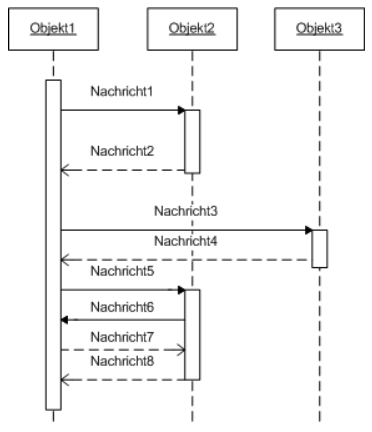
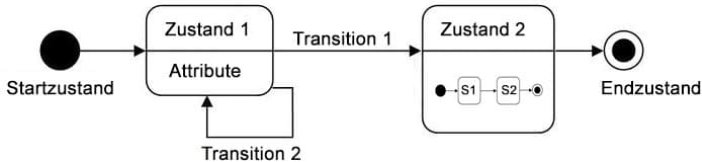
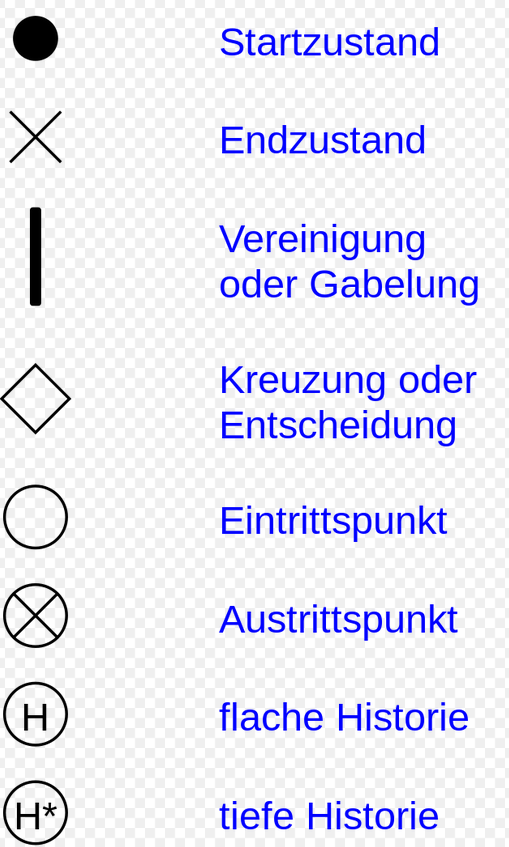
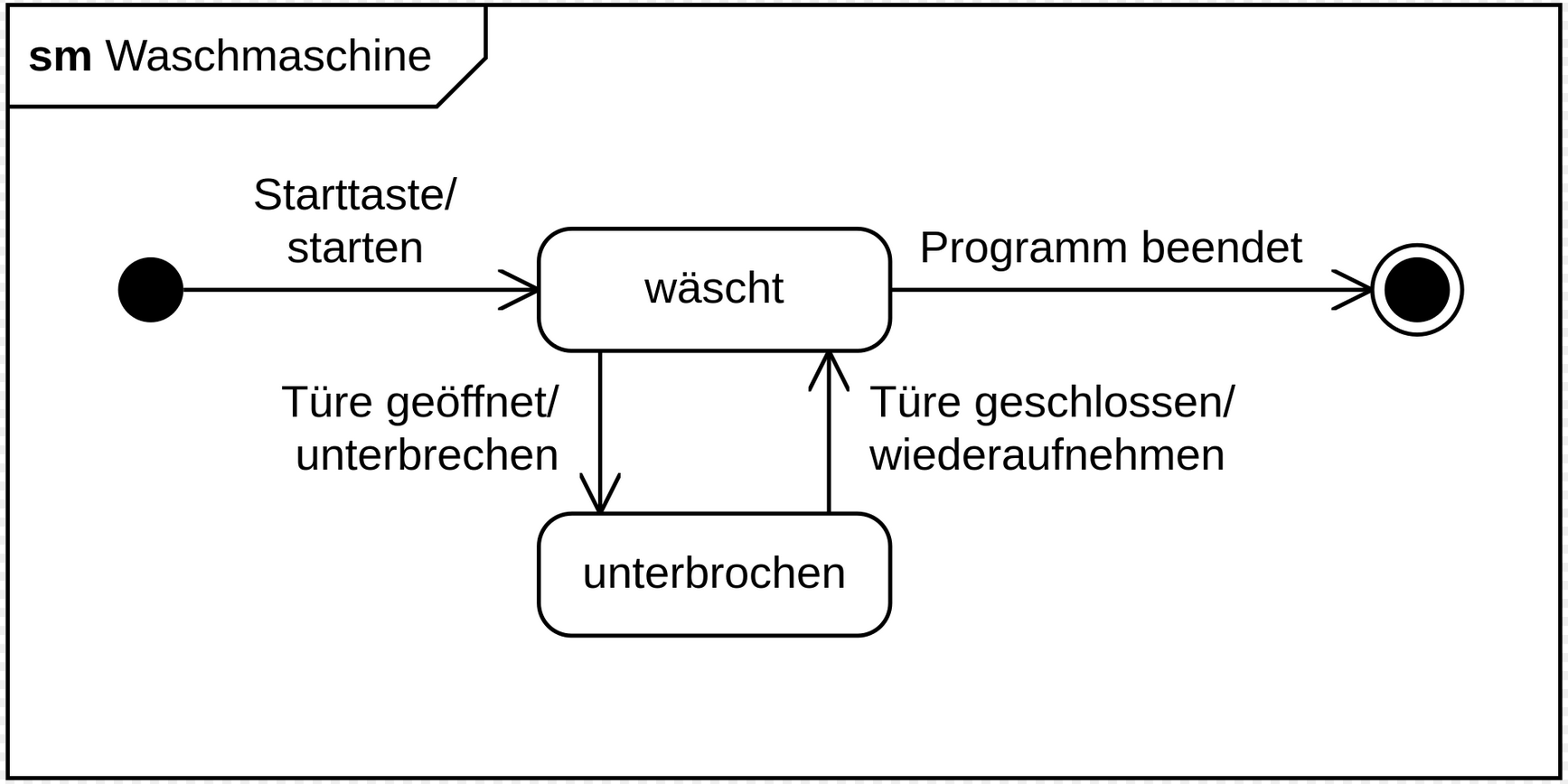
Dokumentiere die wesentlichen Aspekte sowie deine persönlichen Lernergebnisse.

Nach der Beschäftigung mit dieser Aufgabe musst du wissen, was unter Anforderungsanalyse zu verstehen ist, welche Werkzeuge man dazu verwendet und wie man diese einsetzt. Insb. musst du wissen, wie man USE-Case-Diagramme zeichnet, wie man Use-Cases genau beschreibt und welche Rolle UI-Mockups in diesem Zusammenhang spielen.

# Wichtige UML-Diagrammarten

Arbeite dich in folgende Diagrammarten ein:

* **Klassendiagramm**
  + dient zur grafischen Darstellung von Klassen, Schnittstellen sowie deren Beziehungen
  + Einsatzzweck: Objekte abstrahieren, Zusammenspiel der Klassen wird sichtbar
  + Notationsform:  
    Ein Bild, das Tisch enthält.

    Automatisch generierte Beschreibung  
    
  + Beispiel: 
* **Sequenzdiagramm**
  + ist ein Verhaltensdiagramm, welches eine Interaktion im Sinne der UML grafisch darstellt
  + beschreiben den Austausch von Nachrichten zwischen Objekten mittels Lebenslinien
  + Einsatzzweck: Darstellung eines Weges durch einen Entscheidungsbaum innerhalb eines Systemablaufes
  + Notationsform:   
    
  + Beispiel:  
    
* **Zustandsdiagramm**
  + stellt endliche Automaten grafisch dar
  + Einsatzzweck: Verhalten eines Systems oder die zulässige Nutzung der Schnittstelle eines Systems
  + Notationsform:  
    
  + Beispiel:  
    

Fasse für jede Diagrammart folgende Punkte übersichtlich zusammen:

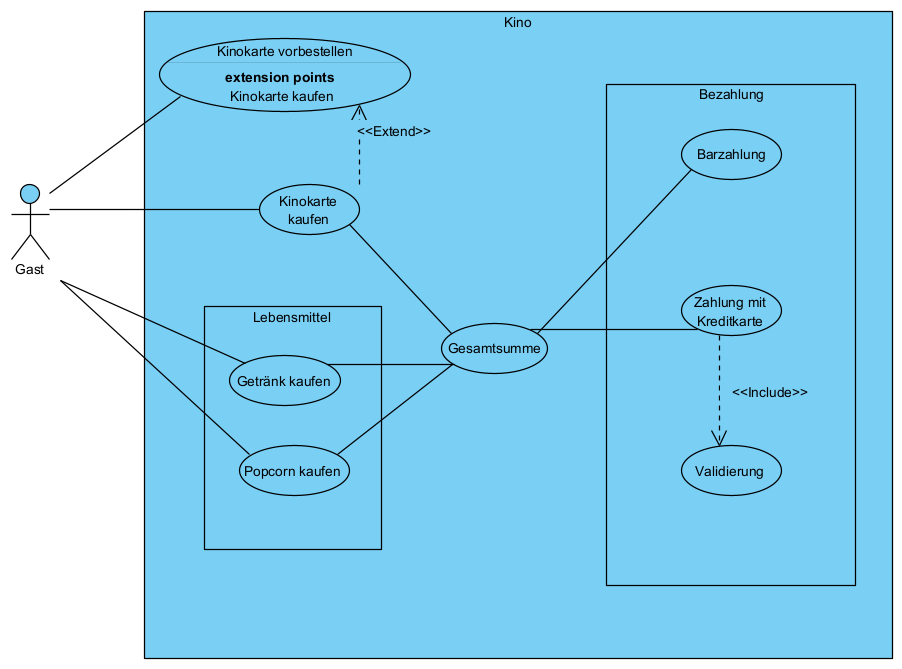
* Einsatzzweck
* Wesentliche Notationsformen
* Exemplarische Anwendung anhand eines Beispiels

Nach der Beschäftigung mit dieser Aufgabe musst du wissen, wie man die gegebenen Diagrammarten korrekt zeichnet und wann man sie braucht.

# USE-Case-Diagramm 1

Entwerfen Sie ein Anwendungsfalldiagramm zu der folgenden Beschreibung:

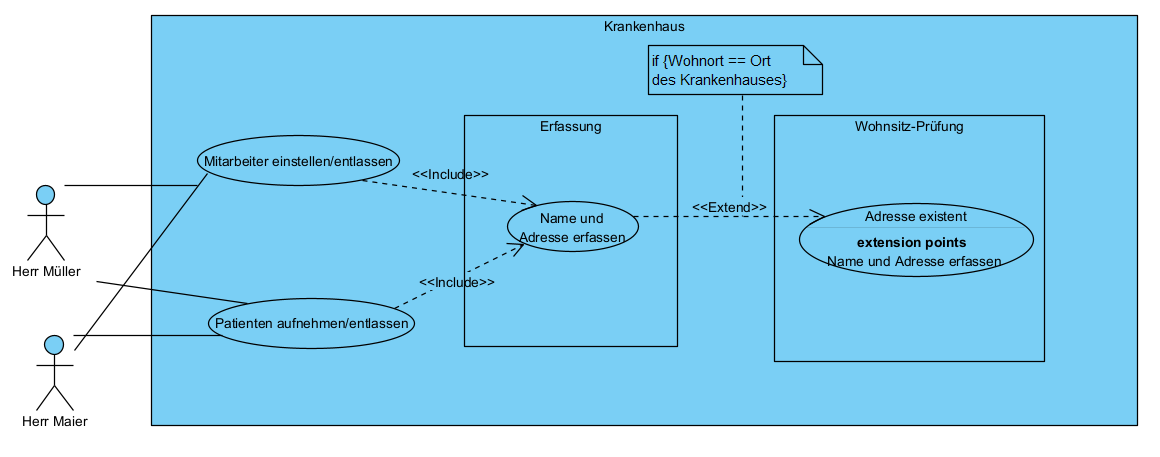
In einem Kino kann ein Gast Kinokarten an der Kasse kaufen, die vorbestellt sein könnten. Außerdem ist es möglich Popcorn und Getränke zu bestellen. Danach bezahlt der Kunde beim Kassierer die Rechnung. Es ist auch möglich mit Kreditkarte zu bezahlen, welche bei Bedarf einer automatischen Prüfung unterzogen werden kann.



# USE-Case-Diagramm 2

Es soll ein Anwendungssystem zur Unterstützung der Geschäftsprozesse in einem Krankenhaus entwickelt werden. Das System soll folgende Aufgaben erledigen:

* Herr Müller und Herr Maier seien in der Verwaltung angestellt.
* Herr Müller soll Mitarbeiter einstellen und entlassen können. Sowohl Herr Müller als auch Herr Maier kann Patienten aufnehmen und entlassen.
* Sowohl bei der Einstellung von Mitarbeitern, als auch bei der Aufnahme von Patienten müssen Name und Adresse erfasst werden. Um redundante Anwendungsfall-Beschreibungen zu verhindern wird diese Tätigkeit in einen gesonderten Anwendungsfall ausgelagert.
* Falls der einzustellende Mitarbeiter bzw. der aufzunehmende Patient seinen Wohnsitz am Ort des Krankenhauses hat, wird geprüft, ob die angegebene Adresse am Wohnort existiert. Lagern Sie auch diesen Anwendungsfall aus.



# USE-Case-Detailbeschreibungen

Priorisieren Sie die aus den vorherigen beiden Use-Case-Übungen und definieren Sie die Use-Case-Details (level, complexity, status, pre-conditions, post-conditions and assumptions, event flow etc.) für die vorherigen beiden Use-Case-Übungen. Verwenden Sie dazu eine der bereitgestellten Schablonen bzw. ein entsprechendes Software-Tool.

|  |  |
| --- | --- |
| Name | Kino besuchen |
| Ziel im Kontext | Kinokarte kaufen und bezahlen |
| Akteure | Gast |
| Trigger | Kunde geht ins Kino |
| Essenzielle Schritte | 1. Kunde kauft Kinokarte  2. Gesamtsumme berechnen  3. Barzahlung |
| Erweiterungen | 1a. Kinokarte ist vorbestellt |
| 1b. Getränk/Popcorn kaufen |
| 3a. Bezahlung mit Kreditkarte  3a1. automatische Prüfung |

|  |  |  |
| --- | --- | --- |
| Name | Personenbezogene Daten verwalten | |
| Ziel im Kontext | Mitarbeiter und Patienten verwalten | |
| Akteur | Herr Müller, Herr Maier | |
| Trigger | Mitarbeiter/Patient Statusänderung | |
| Vorbedingungen | Herr Müller und Herr Maier sind berechtigt eine Änderung vorzunehmen  Mitarbeiter oder Patient ändert seinen Status | |
| Essenzielle Schritte | Mitarbeiter | Patient |
| M1. Mitarbeiter einstellen | P1. Patient aufnehmen |
| M2. Name und Adresse erfassen | P2. Name und Adresse erfassen |
| Erweiterungen | M/P2a. Prüfen, ob Adresse existent ist | |

**Schablone A**

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

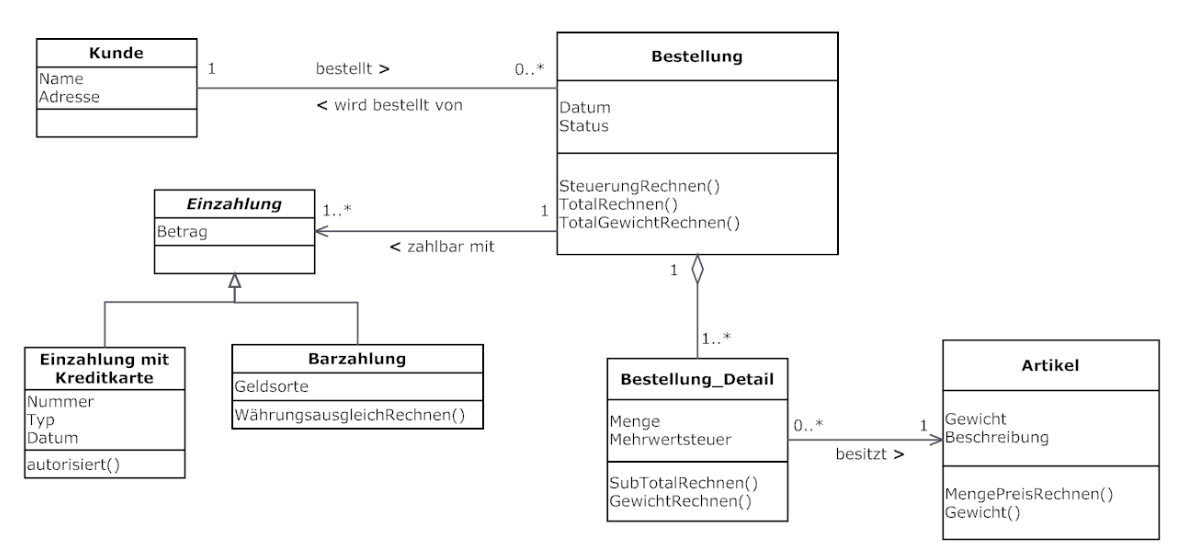
**Schablone B**

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

# Klassendiagramme 1

Gegeben ist folgendes Klassendiagramm:



Bestimmen Sie, ob die folgenden Aussagen zum Klassendiagramm richtig oder falsch sind.

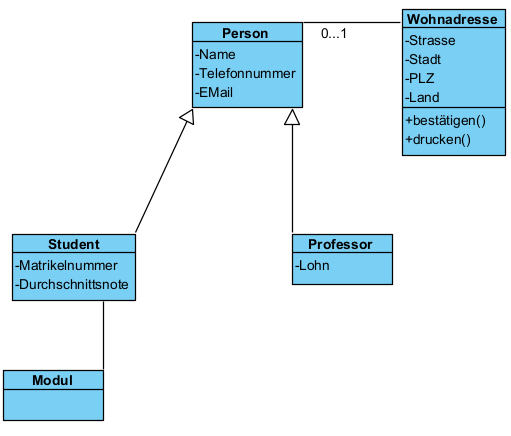
* Es kann im System Kunden geben, die nie eine Bestellung durchgeführt haben.
* Die Klasse Einzahlung ist die Oberklasse der Klasse Bestellung.
* Jedes Objekt der Klasse *Bestellung\_Detail* besitzt genau einen Artikel.
* Alle Einzahlungen mit Kreditkarte haben einen Betrag.
* Es ist möglich, dass ein Artikel keine Assoziation mit einem *Bestellung\_Detail* besitzt.
* Jedes *Bestellung\_Detail*, das Teil einer Bestellung ist, hat seinen eigenen Status und sein eigenes Datum.

# Klassendiagramme 2

Gegeben ist der folgende Sachverhalt.

Jede Person hat einen Namen, eine Telefonnummer und E-Mail. Jede Wohnadresse wird von nur einer Person bewohnt. Es kann aber sein, dass einige Wohnadressen nichtbewohnt sind. Den Wohnadressen sind je eine Strasse, eine Stadt, eine PLZ und ein Land zugeteilt. Alle Wohnadressen können bestätigt werden und als Beschriftung (für Postversand) gedruckt werden. Es gibt zwei Sorten von Personen: Student, welcher sich für ein Modul einschreiben kann und Professor, welcher einen Lohn hat. Der Student besitzt eine Matrikelnummer und eine Durchschnittsnote.

Modellieren Sie diesen Sachverhalt mit einem UML Klassendiagramm.

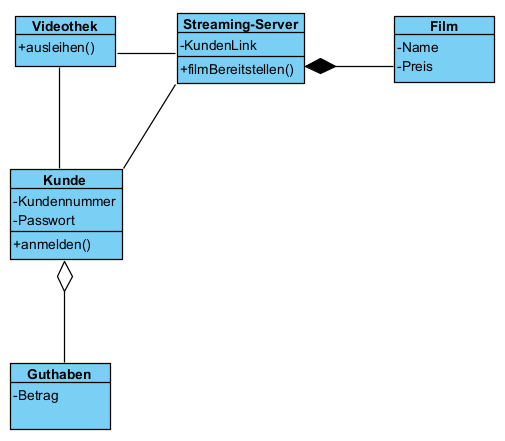


# Klassendiagramme 3

Sie haben den Auftrag, eine Online-Videothek zu realisieren. Sie haben dazu folgende Angaben erhalten:

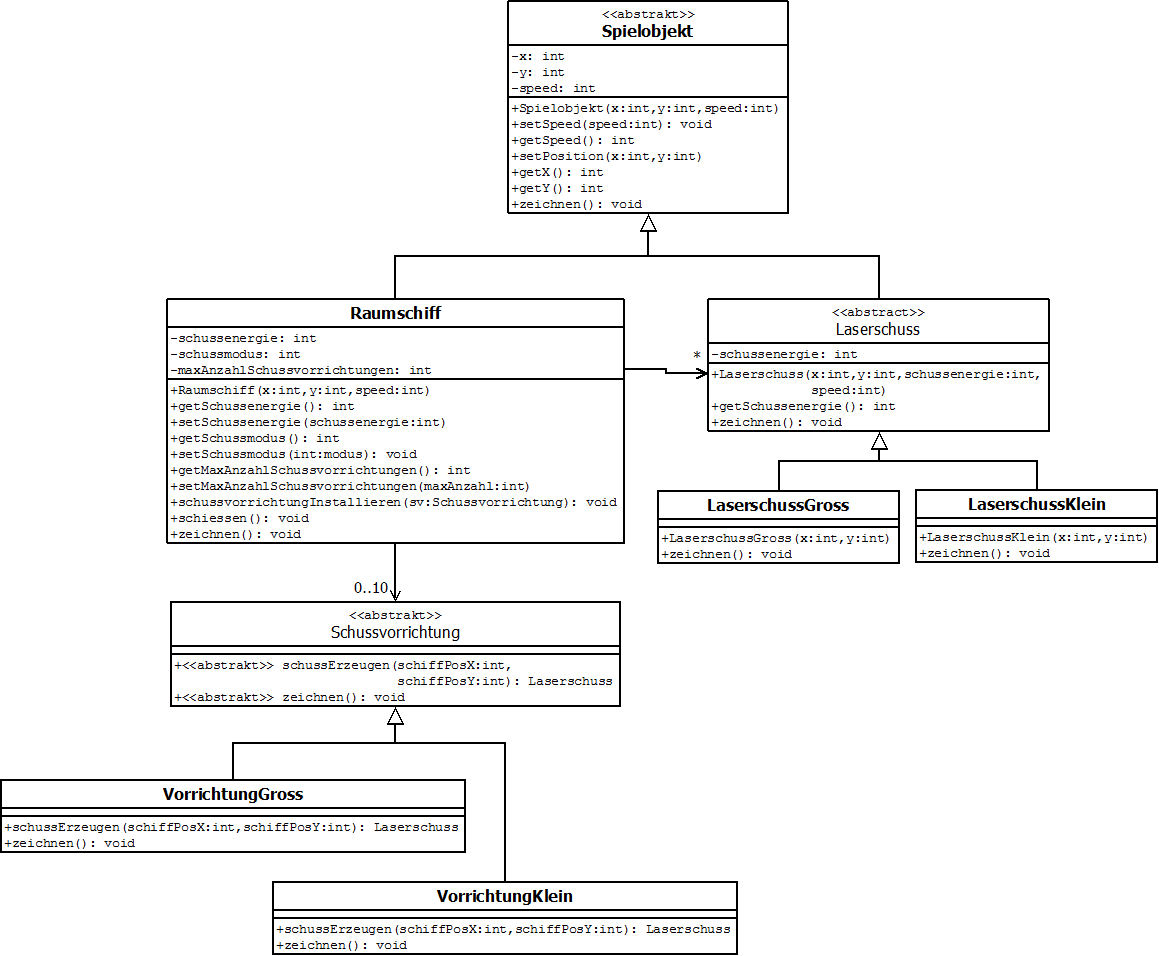
* Die Videothek unterstützt das Ausleihen von Filmen für registrierte Kunden. Dazu müssen
* Kunden sich zunächst mit ihrer Kundennummer und ihrem Passwort anmelden.
* Kunden werden zusammen mit ihrem Guthaben verwaltet.
* Filme besitzen einen individuellen Namen und Preis.
* Ein Film wird über einen Streaming-Server bereitgestellt. Der Server kann hierzu einen kundenspezifischen Link generieren.

Modellieren Sie diesen Sachverhalt anhand eines Klassendiagramms. Wählen Sie sinnvolle Operationen (mit möglichst vollständigen Signaturen) und Attribute für Ihre Klassen. Ergänzen Sie die Klassen um sinnvolle Beziehungen und deren Kardinalitäten.



# Klassendiagramme 4

Beschreiben Sie den in dem Klassendiagramm dargestellten Sachverhalt in Worten.



Ein Raumschiff ist ein Spielobjekt mit Datenfeldern und Eigenschaften des Spielobjekts. Zusätzlich hat es noch eigene Datenfelder und Eigenschaften. Ein Raumschiff kann beliebig viele Laserschüsse abgeben, diese müssen entweder groß oder klein sein. Laserschüsse sind auch Spielobjekte mit deren Datenfeldern und Eigenschaften.

Ein Raumschiff kann keine bis 10 Schussvorrichtungen haben, diese müssen wieder klein oder groß sein.

# LinSequenzdiagramm

Modellieren Sie für die Online-Videothek (siehe Aufgabe 3) die Film Ausleihen Funktion. Erstellen Sie dazu ein Sequenzdiagramm für folgenden Ablauf der Ausleihe:

* Die Videothek berechnet zuerst, ob das Guthaben des Kunden reicht, um den Film zu bezahlen.
* Reicht das Guthaben nicht aus, wird stattdessen eine Aufforderung zum Ausfüllen des Guthabens

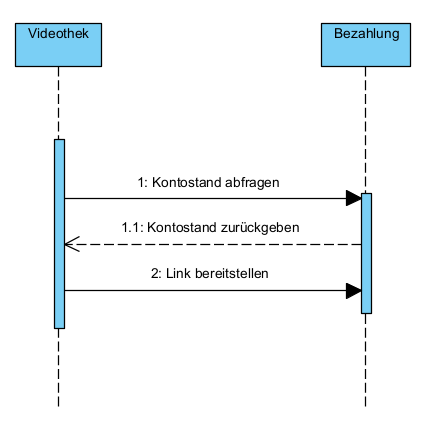
angezeigt.

* Falls das aktuelle Guthaben des Mitglieds ausreicht, veranlasst die Videothek einen Streaming-

Server einen Link für den Film zu generieren.

* Die Videothek zeigt dem Benutzer den Link an, unter dem der Film zugreifbar ist.

Gehen Sie davon aus, dass sich das Mitglied bereits auf der Seite des gewünschten Films beendet. Wählen Sie geeignete Namen für die Elemente Ihres Diagramms. Wichtig: Achten Sie darauf, dass Ihre Diagramme aus Aufgabe 3 und 5 konsistent sind. Nutzen Sie im Sequenzdiagramm nur Klassen, Operationen, etc. die im Klassendiagram aus Aufgabe 3 enthalten sind. Falls Sie im Sequenzdiagramm zusätzliche Operationen, Parameter, etc. brauchen, ergänzen Sie das Klassendiagramm entsprechend!

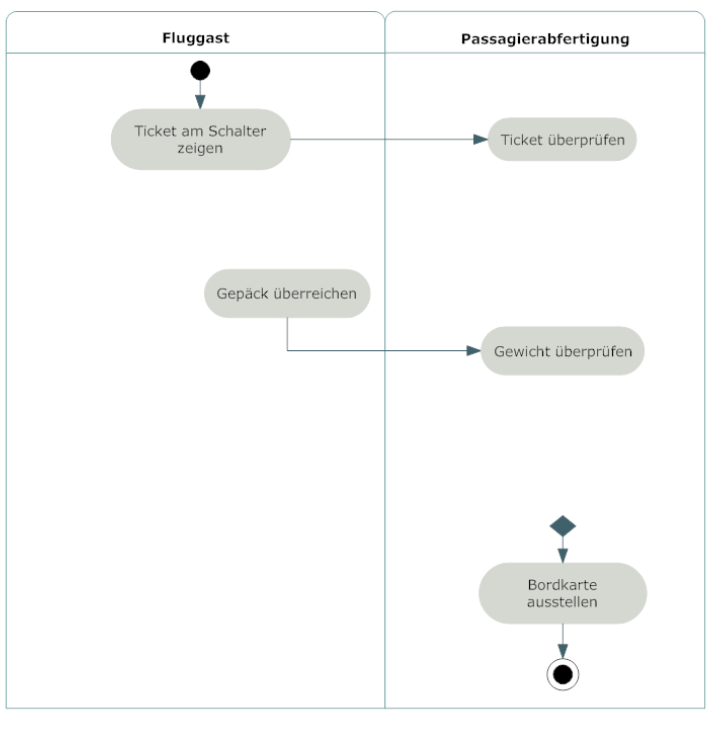


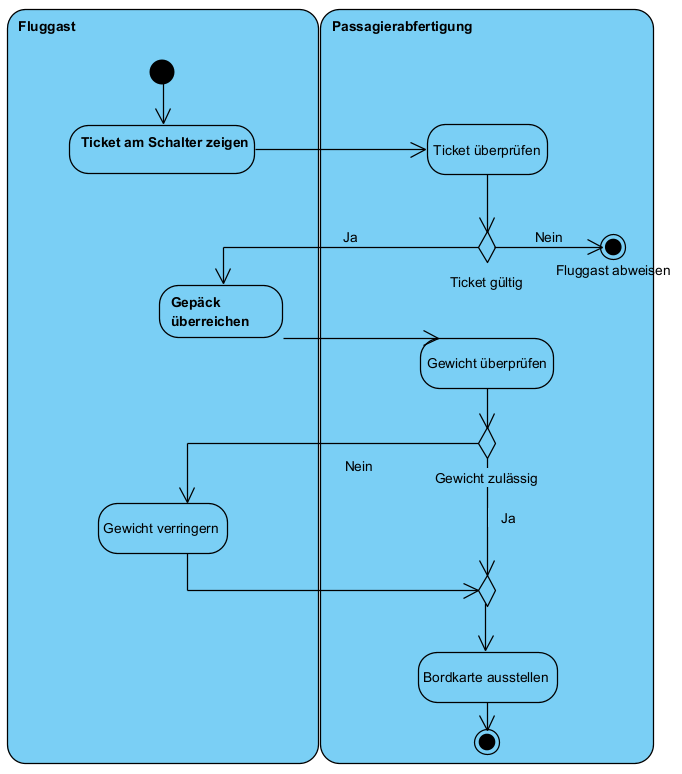
# Aktivitätsdiagramm

Gegeben ist der folgende Sachverhalt.

Ein Fluggast ist am Flughafen angekommen. Zur Überprüfung seines Tickets begibt er sich zum Schalterseiner Fluggesellschaft. Falls das Ticket in Ordnung ist, übergibt er am Schalter sein Gepäck. Falls mit dem Ticket etwas nicht stimmt, muss der Fluggast den Kundendienst konsultieren und er kann nichtmitfliegen. Das Gepäck wird zudem auf Übergewicht überprüft. Falls dem so ist, muss der Fluggast zusätzliche Kostenübernehmen. Falls aber das Gewicht in Ordnung ist, wird die Bordkarte ausgestellt.

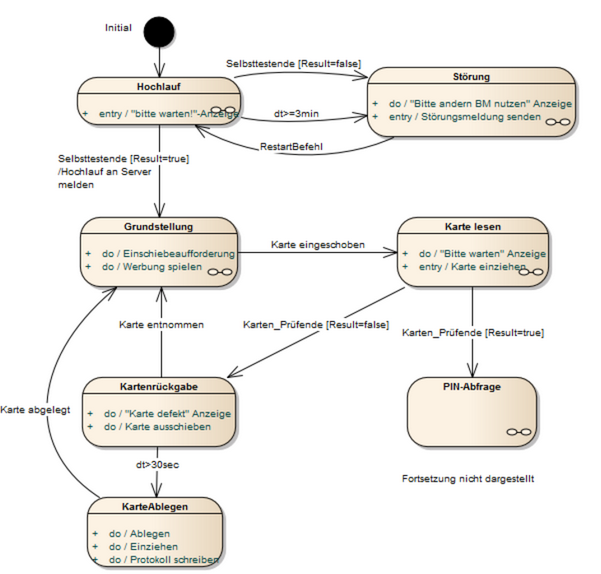
Modellieren Sie diesen Sachverhalt im folgenden unvollendeten UML-Aktivitätsdiagramm:





# Zustandsdiagramm

Erklären Sie folgendes Zustandsmodell zum Thema „Booten eines Bankomaten“.



Nach dem Einschalten führt der Bankomat einen Testlauf durch. Wenn dieser fehlschlägt oder länger als 3 Minuten dauert, wird eine Störung ausgegeben und der Bankomat wieder neugestartet. Wenn die Selbsttestung erfolgreich ist, begibt sich der Bankomat in Grundstellung und übermittelt den erfolgreichen Start an den Server. Jetzt kann eine Karte eingeschoben werden. Diese wird gelesen und wenn die Karte korrekt ist kommt die PIN-Abfrage. Bei nicht korrekter Karte erfolgt die Kartenrückgabe. Wenn die Karte innerhalb von 30 Sekunden entnommen wurde, kehrt der Bankomat in die Grundstellung zurück. Tritt dies nicht ein, wird die Karte eingezogen, abgelegt und ein Protokoll geschrieben. Anschließend geht der Bankomat wieder in die Grundstellung über.