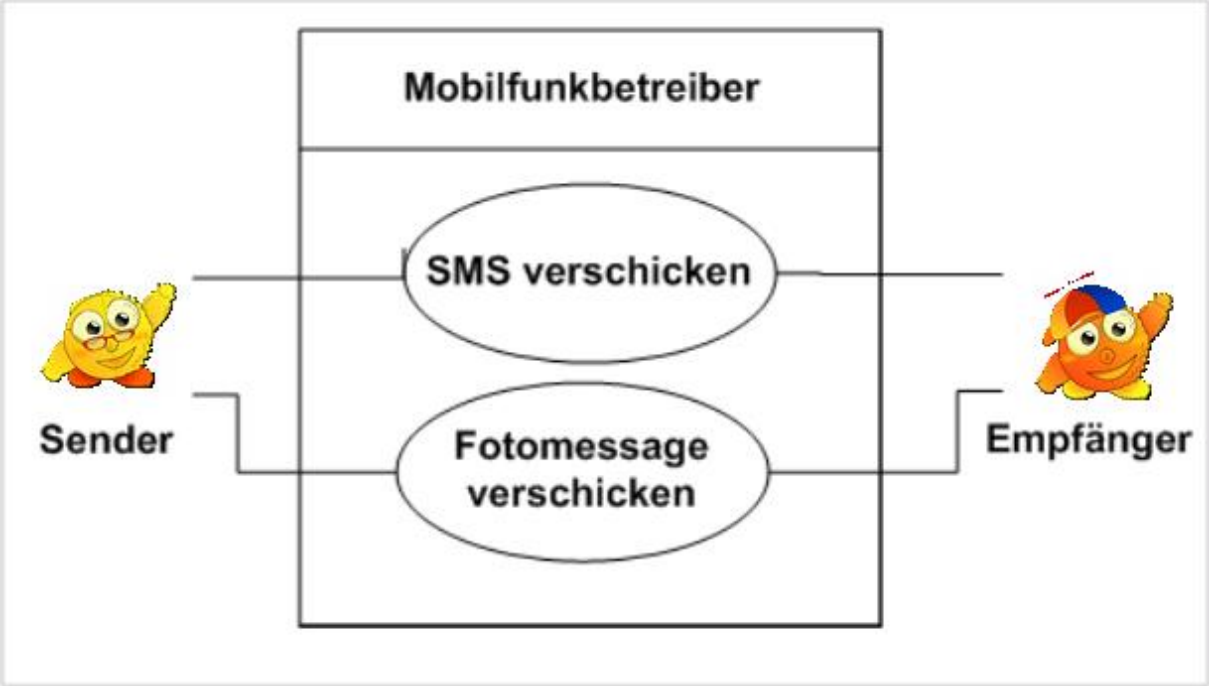
http://schuelerlabor.informatik.rwth-aachen.de

Leitprogramm: Modellieren mit UML



Verfasser: Christian Banyai, Tina Jetten

Kurz-Info:

Informatischer Inhalt: UML, Modellierung
Jahrgangsstufe: 12

Vorwissen: Objektorientierte Programmierung

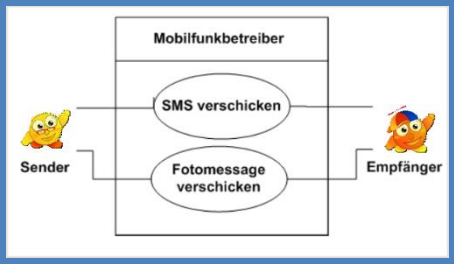
KURZINFORMATION

Titel: *Modellieren mit UML*

Schultyp: *Gymnasium*

optimale Jahrgangsstufe: *Klasse 12*

geeignete Kursart: *Grundkurs Informatik*



Themenbereich: *Modellierung, UML*

Leitideen: Das Leitprogramm kann genutzt werden, um Oberstufenschülerinnen und -schülern UML-Diagramme näherzubringen und diesbezüglich auf das Abitur vorzubereiten. Anhand eines größeren Szenarios werden die verschiedenen Diagrammtypen vorgestellt (mit dem Schwerpunkt auf Klassendiagrammen). Dazu wird das kostenlose Programm *violet* verwendet, das eine einfache Erstellung dieser Diagramme ermöglicht.

EINORDNUNG IN GESETZLICHE RAHMENBEDINGUNGEN

Lehrplan NRW:

Inhaltsfeld: Daten und ihre Strukturierung

Kompetenzbereiche: Modellieren, Argumentieren

Darstellen und Interpretieren,

Vorgaben zum Zentralabitur: Daten und ihre Strukturierung, Java

Bildungsstandards der GI:

Erstellung eines Datenmodells zu einem Realitätsausschnitt

Einbindung in den Unterricht: Das Leitprogramm kann als Einstieg in die Arbeit mit UML-Diagrammen verwendet werden.

Vorkenntnisse: Objektorientierte Programmierung

Inhaltsbeschreibung: Die Modellierung und Darstellung von Daten in (UML-)Diagrammen ist grundlegender Bestandteil der Informatik, welcher auch im Kernlehrplan verankert ist.

UML-Diagramme werden zur Veranschaulichung von Daten und Situationen genutzt, um beispielsweise die spätere Implementierung und die verständliche Darstellung von komplexeren Sachverhalten zu vereinfachen.

In diesem Leitprogramm wird dies mithilfe des Programms *violet* vermittelt. Hauptsächlich wird auf Klassendiagramme eingegangen, in den Zusatzaufgaben wird aber auch die Funktionsweise und Anwendung von Anwendungsfalldiagrammen, Objektdiagrammen und Zustandsdiagrammen eingeübt.

INHALT

EINLEITUNG

Stell dir vor, du arbeitest in einer großen Firma. Ein ganz normaler Arbeitstag beginnt: Dein Chef trägt dir auf, ein Java-Programm durchzuarbeiten. Du sollst es verstehen und dann in eine andere Programmiersprache übertragen. Ach du Schreck! Du merkst, dass das Programm ausgedruckt aus zehn DIN-A4-Seiten besteht. Da musst du jetzt durch! Als du allerdings deine Arbeit erledigt hast, denkst du darüber nach, wie man sich diese unangenehme Aufgabe hätte erleichtern können. Du schaust das Programm noch einmal durch und stellst fest, dass dir zum Verstehen des Programms auch ein paar Kernabschnitte gereicht hätten. Die genaue Ausformulierung in der Programmiersprache Java hat jedoch dazu beigetragen, dass du viel mehr lesen musstest, als du später in die andere Programmiersprache übertragen konntest. Du kommst zu dem Schluss, dass es besser gewesen wäre, wenn der Programmierer des Java-Programms die Kernpunkte seines Programms in knapper und besser verständlicher Form zusammengefasst hätte. Genau diese Art zu arbeiten wirst du mit diesem Leitprogramm erlernen. Dazu lernst du, wie die Kernpunkte eines Programms in Diagrammen zusammengefasst werden. Du wirst nach der Bearbeitung des Leitprogramms wissen, was UML-Diagramme sind und wie man diese mit Hilfe von violet selber erstellen kann.

Arbeite dieses Leitprogramm sorgfältig durch. Falls du Fragen hast und keine Lehrkraft oder andere Person fragen kannst, schreibe gerne eine E-Mail mit deinen Fragen an infosphere-support@informatik.rwth-aachen.de.

ARBEITSANLEITUNG

Die folgenden Kapitel des Leitprogramms sollst du selbstständig durcharbeiten. Die unterschiedlichen Teile der Kapitel sind durch folgende Symbole gekennzeichnet:

Lernziel

Das hast du nach dem Kapitel gelernt, wenn du erfolgreich gearbeitet hast.

Theorieteil

In diesem Teil lernst du die Grundlagen für die Aufgabenbearbeitung und die Tests.

Aufgaben

Die Übungsaufgaben erledigst du entweder am Computer oder im Heft. Anschließend kannst du deine Ergebnisse mit der Musterlösung vergleichen.

Kapiteltest

Hier kannst du testen, ob du das Kapitel, an dem du zuletzt gearbeitet hast, gut verstanden hast.

Additum

Diese zusätzlichen Aufgaben kannst du bearbeiten, wenn du zu den Schnellsten gehörst. Die Arbeitsaufträge sind schwieriger und erweitern den normalen Unterrichtsstoff.

KAPITEL 1: EINFÜHRUNG IN UML UND VIOLET

Übersicht

Damit dir und den anderen Mitarbeitenden in deiner Firma die Arbeit in Zukunft erleichtert wird, weist dein Chef Mr. Smith dich an, die Abläufe in der Auftragsabwicklung deines Betriebes graphisch darzustellen. Wahrscheinlich wirst du nun darüber nachdenken, die Abläufe mit Hilfe einer Mind-Maps in einem Zeichenprogramm darzustellen. Dieses Kapitel zeigt dir, dass es noch bessere Darstellungsmöglichkeiten gibt mit aussagekräftigeren Symbolen und Beziehungen, die expliziter dargestellt werden können.

Die Symbolsprache, die sich vor allem in der Informatik durchgesetzt hat, nennt sich **UML** (**U**nified **M**odeling **L**anguage). Sie wurde 1990 erarbeitet und 2003 zu UML 2.0 weiterentwickelt. Diese Version behandelt dieses Leitprogramm. Ein einfaches Werkzeug zur Erstellung von UML-Diagrammen ist **violet**.

Nachdem du dieses Kapitel bearbeitet hast, kannst du UML-Diagramme mit violet entwerfen und somit unter anderem eine optisch ansprechende Grafik der Abläufe in seinem Betrieb erstellen. Die dafür nötigen Fachbegriffe, die du bei der objektorientierten Programmierung kennengelernt hast, sind dir nach diesem Kapitel ebenfalls wieder geläufig.

Lernziele

Am Ende dieses Kapitels...

- kannst du erklären, was ein Klassendiagramm ist und aus welchem Grund es eingesetzt wird.
- kannst du selbstständig einfache UML-Diagramme mit violet erstellen.
- kannst du den Unterschied zwischen den Begriffen Klasse, Attribut und Operation wiedergeben.

Bevor es losgeht:

1. Installation von Java (OpenJDK)

Wenn du noch kein JDK (Java Software Development Kit) auf deinem Computer hast, solltest du Folgendes tun:

Da die Standard-Editions des JDKs von Oracle seit dem 16.04.2019 kostenpflichtig sind, empfehlen wir die Installation von OpenJDK 14.0.2 (du kannst auch eine neuere Version herunterladen, passe dann einfach die Installationsanleitung darauf an, und ersetze „14.0.2“ durch deine Version).

<u>Für Windows 10:</u>

1. Lade die **Windows/x64.zip** Version auf https://jdk.java.net/archive/herunter.
2. Entpacke die heruntergeladene Datei in *C:\Program Files\Java* . (Es sollte nun ein Pfad *C:\Program Files\Java\jdk-14.0.2* entstanden sein.)
3. Öffne Systemsteuerung → System & Sicherheit → System → Erweiterte Systemeinstellungen → Umgebungsvariablen
4. Überprüfe unter Systemvariablen, ob die Variable „*JAVA_HOME*“ existiert:
    - Falls ja: Korrigiere den Variablenwert von „*JAVA_HOME*“ durch den Button Bearbeiten...“ zu „*C:\Program Files\Java\jdk-14.0.2*“
    - Falls nein: Erstelle eine Variable durch den Button „Neu...“ mit Variablennamen „*JAVA_HOME*“ und Variablenwert „*C:\Program Files\Java\jdk-14.0.2*“

5. Ergänze, falls noch nicht vorhanden, für die Systemvariable „Path“ den Wert „*C:\Program Files\Java\jdk-14.0.2\bin*“.
6. Starte deinen Computer neu.
7. Überprüfe die Installation:
   - Öffne die „Eingabeaufforderung“ (cmd bzw. terminal).

```
C:\Users\mustermann>java -version
openjdk version "14.0.2" 2020-07-14
OpenJDK Runtime Environment (build 14.0.2+12-46)
OpenJDK 64-Bit Server VM (build 14.0.2+12-46, mixed mode, sharing)
```

```
C:\Users\mustermann>javac -version
javac 14.0.2
```

Für Linux:

1. Installation via Terminal

```
$ sudo
          apt-get install openjdk-14-jdk openjdk-14-demo openjdk-14-doc openjdk-14-jre-headless openjdk-14-source
```

2. Überprüfen der Installation via Terminal

```
$ java -version
openjdk version "14.0.1" 2020-04-14
OpenJDK Runtime Environment (build 14.0.1+7-Ubuntu-1ubuntu1)
OpenJDK 64-Bit Server VM (build 14.0.1+7-Ubuntu-1ubuntu1, mixed mode, sharing)
```

```
$ javac -version
javac 14.0.1
```

Für Mac:

1. Lade die **macOS/x64 tar.gz** Version auf https://jdk.java.net/archive/ herunter.
2. Entpacke die heruntergeladene Datei in *Library>Java>JavaVirtualMachines.*
3. Überprüfe die Installation:

```
~%  java -version
openjdk version "14.0.2" 2020-07-14
OpenJDK Runtime Environment (build 14.0.2+12-46)
OpenJDK 64-Bit Server VM (build 14.0.2+12-46, mixed mode, sharing)
```

```
~% javac -version
javac 14.0.2
```

2. Installation von violet

Gehe auf http://violet.sourceforge.net/, klicke auf **Download** und dann auf **here**. Wähle dann die Version **2.1.0** aus.

Falls du Microsoft Windows benutzt, lade **violetumleditor-2.1.0.exe** herunter, und speichere es auf deinem Computer. Zum Öffnen, doppelklicke einfach die Datei.

Falls du ein anderes Betriebssystem benutzt, lade **violetumleditor-3.0.0.jar** herunter, und speichere es auf deinem Computer. Falls ein Doppelklick auf die Datei nicht funktioniert, öffne dann das Terminal und gib „java -jar violetumleditor-2.1.0.jar“ ein.

Wenn du mit violet 3.0.0 arbeiten möchtest, ist das auch möglich, manche Funktionalitäten sind aber dann anders als hier beschrieben erreichbar.

Aufgabe 1.1

Erster Blick auf violet

Damit du die Aufgabe bearbeiten kannst, musst du dich zunächst mit violet vertraut machen. Starte dazu dein violet-Programm! Wenn du **File → New** folgst, kannst du zwischen verschiedenen Diagrammtypen auswählen.

Diese Typen stehen zur Auswahl:

1. Class diagram (Klassendiagramm)
2. Object diagram (Objektdiagramm)
3. Use case diagram (Anwendungsdiagramm)
4. State diagram (Zustandsdiagramm)
5. Activity diagram (Aktivitätsdiagramm)
6. Sequence diagram (Sequenzdiagramm)

Wähle nun unter Static View den ersten Eintrag **Class diagram** aus!

Dein Chef kommt vorbei und schaut dir über die Schulter: „Richtig! Ich möchte, dass du die Zusammenhänge in einem Klassendiagramm darstellst. Die vier wichtigen Eckpunkte einer Auftragsabwicklung sind Kundschaft, Bestellung, Rechnung und Anschrift!“

Du rufst dir noch mal die Definition einer Klasse in Erinnerung und schaust nach, wie sie in UML dargestellt wird.

Definition Klasse

In einer Klasse werden Attribute und Operationen definiert. Außerdem entsprechen alle Objekte aus dieser Klasse den in der Klasse festgelegten Eigenschaften.

Du weißt noch ungefähr, was Attribute, Operationen und Objekte sind, und beschließt, dich genauer darüber zu informieren, wenn sie in deiner Grafik vorkommen sollen. Deswegen springst du schnell weiter zur Notation einer Klasse:

Notation und Beispiel

Wie du in diesem Beispiel sehen kannst, werden Klassen als Rechtecke dargestellt. Der Klassenname wird immer fett- und großgeschrieben.

Aufgabe 1.2

Zeichne mit violet das Klassendiagramm so, wie es dein Chef sich vorstellt! Nachdem du Class diagram ausgewählt hast, öffnet sich ein Arbeitsbereich dafür. Rechts siehst du eine Toolleiste mit allen Symbolen, die du zum Erstellen eines Klassendiagramms benötigst. Das erste Symbol mit den vier lila Rechtecken ist das Auswahlwerkzeug (*Select*). Erstelle vier Klassen, indem du erst auf das *Class*-Werkzeug und dann viermal ins Zeichenfeld mit der linken Maustaste klickst. Wähle dann das Auswahlwerkzeug, und ziehe die vier Klassen mit gedrückter linker Maustaste in geeigneter Position. Wenn du nun mit der Maus doppelt auf eine Klasse klickst, kannst du der Klasse einen Namen geben. Es erscheint ein neues Fenster. Gib den Namen in dem Feld *Name* ein, und bestätige die Eingabe mit OK!

Dein Chef ist damit zufrieden, dass du die vier wichtigen Eckpunkte als Klassen dargestellt hast. Allerdings weist er dich darauf hin, dass ...

Definition Klassendiagramm

... ein Klassendiagramm beschreibt, welche Klassen existieren und in welchen Beziehungen sie zueinanderstehen. UML-Modellelemente können verschiedene Arten von Beziehungen untereinander haben. Klassen können beispielsweise Spezialisierungsbeziehungen, Assoziationen, Realisierungs- und Abhängigkeitsbeziehungen haben.

Hinweis

Eine ausführlichere Beschreibung, was unter Beziehungen in UML zu verstehen ist, folgt im nächsten Kapitel. An dieser Stelle wird daher für das erste Verständnis nur ein kleines Beispiel angegeben: Als Verbindungslinie zwischen zwei Klassen **A** und **B** kann man eine Assoziation angeben. Sie bedeutet, dass Klasse **A** auf Klasse **B** zugreifen kann. In UML wird von Navigierbarkeit gesprochen. Die Pfeilrichtung gibt an, welche Klasse auf welche andere Klasse Zugriff hat. In einem Online-Shop beispielsweise sollen verschiedene Artikel angeboten werden können. Daher werden alle Artikel, die durch die gleichnamige Klasse repräsentiert werden, in einer Klasse Angebot zusammengefasst. Dazu zeichnet man zwischen der Klasse Angebot und der Klasse Artikel eine Assoziation ein.

Aufgabe 1.3

Überlege in welcher Beziehung die Klassen zueinanderstehen! Verbinde sie sinnvoll mit Pfeilen! Verwende dazu das „*Is associated with*“-Tool in der Toolleiste!

Nachdem du den durchgezogenen Pfeil mit V-Spitze ausgewählt hast, klicke mit der linken Maustaste auf eine Klasse und ziehe den Pfeil mit gedrückter linker Maustaste auf die nächste Klasse. Du kannst die Pfeile auch beschriften, wenn du auf den Pfeil mit der linken Maustaste doppelklickst. Trage passende Bezeichnungen für die Assoziationspfeile in das Feld *Middle label* ein. Unten in dem Fenster kannst du außerdem das Design des Pfeils (*Bent style*) ändern, probiere das am besten selbst aus. Wenn du auf den Pfeil klickst, die linke Maustaste gedrückt hältst und in eine beliebige Richtung ziehst, kannst du die Form des Pfeils manuell verändern.

Aufgabe 1.4

Du hast das fertige Diagramm deinem Chef vorgelegt, doch leider gefällt es ihm noch nicht ganz. Er möchte, dass die Kundschaft an oberster Stelle steht, Bestellung links, Anschrift rechts und Rechnung zuunterst.

Ändere das Diagramm so, dass es deinem Chef gefällt! Verschiebe dazu die Klassen, indem du mit der linken Maustaste die Rechtecke anklickst, die Maustaste gedrückt hältst und die Rechtecke an die richtige Position ziehst!

Aufgabe 1.5

Welche Vorteile der Arbeit mit violet gegenüber der Darstellung mit Hilfe einer Mind-Map in einem Zeichenprogramm sind dir nach der Bearbeitung des Kapitels klar geworden?

_____

_____

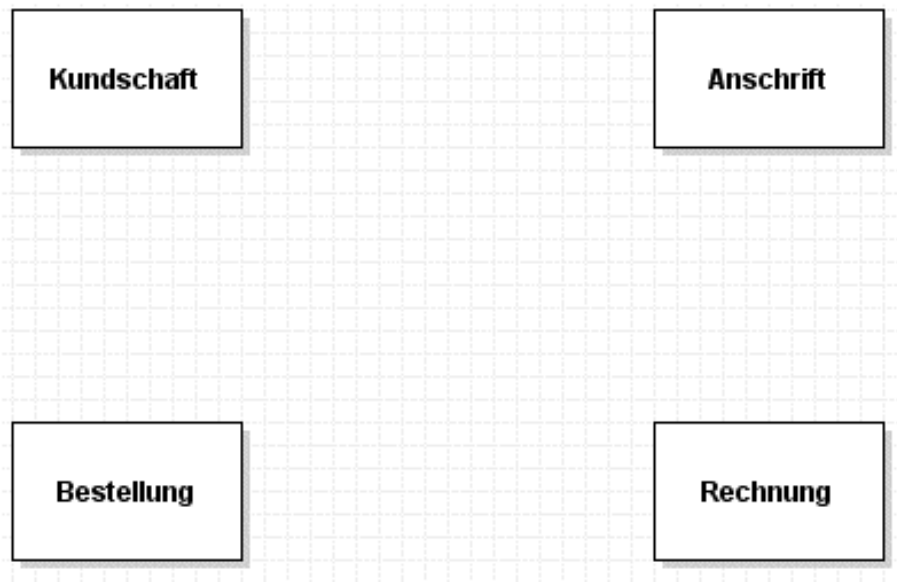
_____

_____

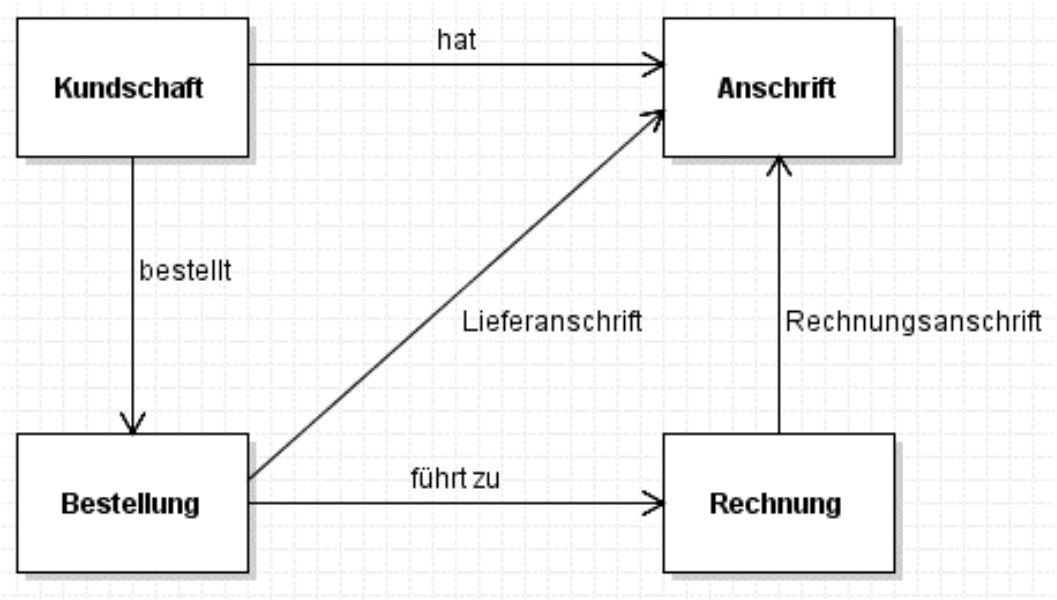
_____

Musterlösungen für Kapitel 1

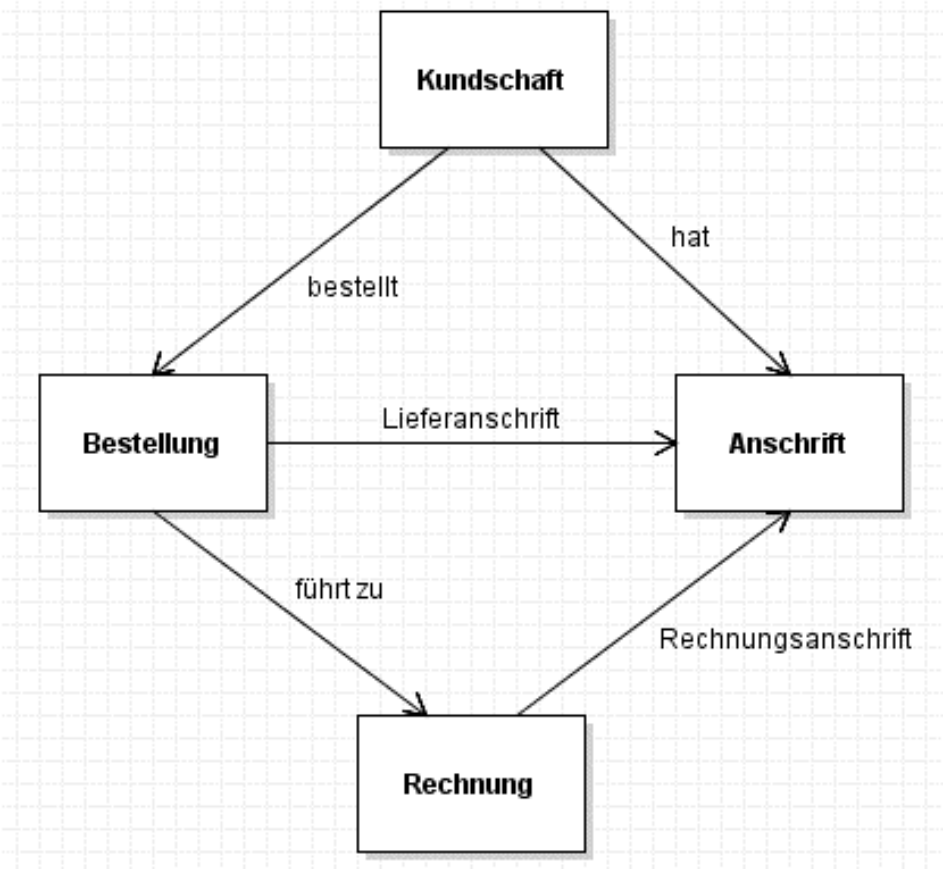
Lösung Aufgabe 1.2



Lösung Aufgabe 1.3



Lösung Aufgabe 1.4



Lösung Aufgabe 1.5

1. Durch die Vorlagen für Klassen und Pfeile geht eine Diagrammentwicklung wesentlich schneller.
2. Die Klassen sind frei verschiebbar, und die Pfeile ändern sich automatisch. Diagramme sind also schnell und einfach anpassbar.

Kapiteltest Kapitel 1

Frage 1: Nenne mindestens drei unterschiedliche Diagrammtypen, die man mit violet erstellen kann!

_____

Frage 2: Wie sieht die Notation einer Klasse in UML mit violet aus?

a)

b)

c)

d)

e)

Frage 3: Die Firma möchte ihr Warenlager aufstocken. Entwickle mit violet ein Klassendiagramm, das die Auftragsabwicklung modelliert! Das Diagramm kann aus den vier Klassen **Warenlager**, **Bestellung**, **Anschrift** und **Rechnung** bestehen. Zeichne auch die Beziehungen zwischen den Klassen, und beschrifte sie!

Frage 4: Wähle die Vorteile von violet gegenüber einer Mind-Map aus! Kreuze die richtigen Vorschläge an!

☐ Man kann Videoclips einfügen.

☐ Vorlagen für Klassen und Pfeile.

☐ Pfeile ändern sich beim Verschieben automatisch.

☐ Violet erkennt Rechtschreibfehler und korrigiert Groß- und Kleinschreibung automatisch.

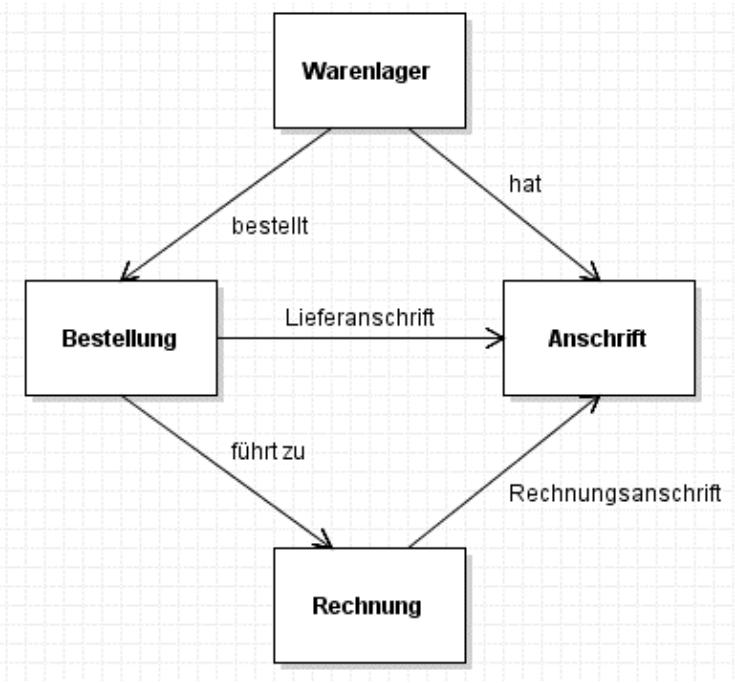
Lösung Frage 1: 3 der folgenden Diagrammtypen waren aufzuzählen:

- Class diagrams (Klassendiagramme)
- Sequence diagrams (Sequenzdiagramme)
- State diagrams (Zustandsdiagramme)
- Object diagrams (Objektdiagramme)
- Use case diagrams (Anwendungsdiagramme)
- Activity diagrams (Aktivitätsdiagramme)

Lösung Frage 2:

Antwort c) ist richtig. Klassen werden als Rechtecke dargestellt. Außerdem wird der Klassenname mit großem Anfangsbuchstaben geschrieben.

Lösung Frage 3: Eine mögliche Lösung ist das folgende Diagramm. Falls du eine andere Lösung hast, frage ggf. deine Lehrkraft, ob diese auch in Ordnung ist. Falls du keine Lehrkraft fragen kannst, kannst du uns auch eine Mail an infosphere-support@informatik.rwth-aachen.de schicken, wenn du dir unsicher bist.



Das Warenlager hat eine Anschrift, und es wird von dort aus eine Bestellung aufgegeben. Für die Bestellung wird als Anschrift eine Lieferanschrift benötigt. Die Bestellung führt dazu, dass eine Rechnung gemacht werden muss, die wiederum eine Anschrift benötigt, die Rechnungsanschrift.

Lösung Frage 4:

- ☐ Man kann Videoclips einfügen.
- ☑ Vorlagen für Klassen und Pfeile.
- ☑ Pfeile ändern sich beim Verschieben automatisch.
- ☐ Violet erkennt Rechtschreibfehler und korrigiert Groß- und Kleinschreibung automatisch.

Sehr gut, du hast deinen ersten Auftrag erfolgreich ausgeführt. Dein Chef lobt dich!

KAPITEL 2: WAS IST DER UNTERSCHIED ZWISCHEN ANALYSE UND DESIGN?

Übersicht

Ein neuer Auftrag wartet auf dich! Mr. Smith verlangt eine Erweiterung des Modells. Im folgenden Kapitel lernst du den Unterschied zwischen Analyse und Design kennen, der für die Entwicklung von Programmen beachtet werden sollte. Dazu wirst du das Modell aus dem letzten Kapitel um Attribute und Operationen erweitern. Du lernst also, was Attribute und Operationen sind und wie du sie in violet darstellen kannst.

Lernziele

Am Ende dieses Kapitels...

- kannst du den Unterschied zwischen Analyse und Design erklären.
- kannst du Attribute und Operationen definieren und wiedergeben, wie man diese mit violet zeichnet.

Dein Chef sagt zu dir: „Das war für den Anfang schon mal sehr gut. Ich möchte, dass du neben den vorhandenen Komponenten der objektorientierten Analyse nun die Komponenten des Entwurfs hinzufügst."

So ein Mist! Jetzt weißt du leider nicht, wo genau der Unterschied zwischen Analyse und Design liegt. Da du meinst, dass du wegen des vorangegangenen Lobs einen Stein im Brett hast bei deinem Chef, fragst du ihn danach. Dieser zeigt dir die folgende Abbildung im Internet und erklärt es dir bereitwillig:

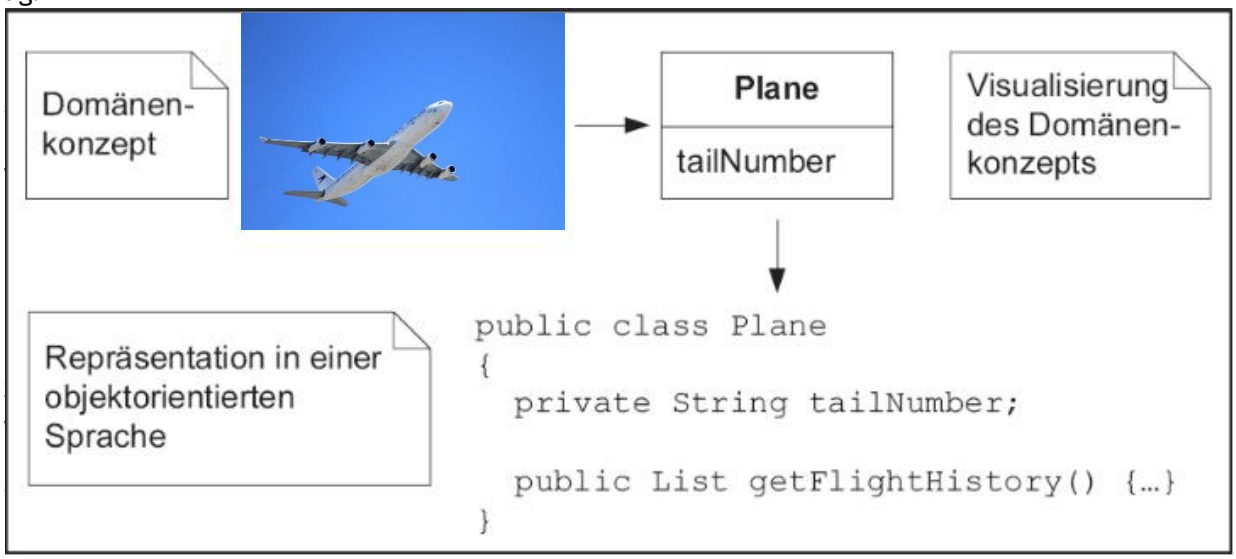


Abb. 1: Flugzeug

Aufgabe 2.1

Schreibe mit eigenen Worten auf, was unter einer objektorientierten Analyse zu verstehen ist!

_____

_____

_____

_____

Definition Design

Das auf der objektorientierten Analyse aufbauende objektorientierte Design ist die kreative Erarbeitung eines Lösungs- bzw. Bearbeitungskonzepts. Die Beziehungen zwischen den Klassen, sowie die Bezeichnung der Attribute und Operationen zeigen auf, *wie* das System die vorhandenen Anforderungen grundsätzlich erfüllt.

Aufgabe 2.2

Definiere das objektorientierte Design mit eigenen Worten!

_____

_____

_____

_____

Damit hast Du also im vorangegangenen Kapitel bereits die objektorientierten Analyseschritte geschafft. Außerdem hast du vom Lösungskonzept Design den Teilbereich der Beziehungen schon bearbeitet. Demnach musst Du noch die Attribute und Operationen in ein Schema einfügen, um das Klassendiagramm zu vervollständigen.

Aufgabe 2.3

Aber was versteht man noch mal unter Attributen und Operationen im objektorientierten Zusammenhang? Versuche dich zu erinnern, und notiere deine Überlegungen!

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Aufgabe 2.4

Du siehst ein Lämpchen in einer GUI (Graphical User Interface). Es ist ganz einfach als ein Kreis dargestellt, der die Farbe von Schwarz auf Gelb wechselt, wenn du draufklickst. Erstelle mit violet die zwei Klassen **Kreis** und **Lampe**!

Diese Klassennamen können nun durch Attribute und Operationen ergänzt werden. Um dies in violet umzusetzen, sollte man sich zunächst vergegenwärtigen, was unter Attributen und Operationen zu verstehen ist.

Speichere deine violet-Datei. Wie du siehst, wird sie mit der Endung *.class.violet.html* abgespeichert. Diese Datei kannst du dann auch wieder mit violet öffnen.

Definition Attribut:

Ein Attribut ist die „Eigenschaft“ eines Objekts in Programmen und Datenbanken, wie zum Beispiel Bezeichner, Datentyp, Wert, Erzeugungsdatum, Darstellungsmöglichkeiten des Objekts. Darstellungsmöglichkeiten des Objekts sind beispielsweise Schrifttyp, Farbe oder Größe. Das Attribut ist ein (Daten-)Element, das in jedem Objekt einer Klasse gleichermaßen enthalten ist und von jedem Objekt mit einem individuellen Wert repräsentiert wird. Im Gegensatz zu Objekten haben Attribute außerhalb des Objektes, von dem sie ein Teil sind, keine eigene Identität. Attribute sind vollständig unter der Kontrolle der Objekte, von denen sie ein Teil sind.

Definition Operation:

Operationen sind Dienstleistungen, die von einem Objekt angefordert werden können, sie werden beschrieben durch ihre Signatur (Operationsname, Parameter, ggf. Rückgabetyp).

In violet werden Klassennamen, Attribute und Operationen durch waagerechte Striche voneinander getrennt. Ganz oben steht immer der Klassenname (Kreis, Lampe), in der Mitte die Attribute und im unteren Bereich die Operationen.

Attribute und Operationen beginnen im Gegensatz zu Klassennamen mit einem kleinen Buchstaben.

Aufgabe 2.5

Ergänze die beiden Klassen **Lampe** und **Kreis** aus Aufgabe 2.4 mit Attributen und Operationen! Öffne dazu deine gespeicherte violet-Datei aus Aufgabe 2.4, und klicke mit der linken Maustaste doppelt auf eine Klasse! Trage nun die Attribute im Feld *Attributes* und die Operationen im Feld *Methods* ein!

Bei Attributen notiert man in Klassendiagrammen oft auch schon den Datentyp in der Form „name : datentyp“ oder die Signatur einer Methode. Beides kann aber auch unvollständig sein oder weggelassen werden.

Außerdem kann durch ein „+“, „-“, „#“ vor Attributen oder Operationen direkt im UML-Diagramm eingetragen werden, ob Attribut oder Operation *public*, *private* oder *protected* sind, wenn man dazu schon Informationen hat.

Lösung Aufgabe 2.1

Zur Kontrolle deines Ergebnisses noch einmal eine anders formulierte Definition:

Definition Analyse

Man versteht unter der objektorientierten Analyse den ersten Schritt des Softwareentwicklungsprozesses. In diesem Schritt wird geklärt, was das System leisten soll und alle Klassen aufgezählt, die bei der Ermittlung, Klärung und Beschreibung der Anforderungen benötigt werden.

Lösung Aufgabe 2.2

Die folgenden Eigenschaften des objektorientierten Designs sollte deine Lösung beinhalten:

- kreative Erarbeitung eines Lösungs- bzw. Bearbeitungskonzepts
- Beziehungen zwischen Klassen, Bezeichnung der Attribute und Operationen zeigen auf, *wie* das System die vorhandenen Anforderungen grundsätzlich erfüllt

Lösung Aufgabe 2.3

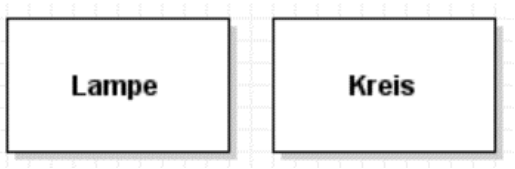
Attribut:

- ist ein (Daten-) Element
- ist eine „Eigenschaft“ eines Objekts
- in jedem Objekt einer Klasse gleichermaßen enthalten
- von jedem Objekt mit einem individuellen Wert repräsentiert

Operationen:

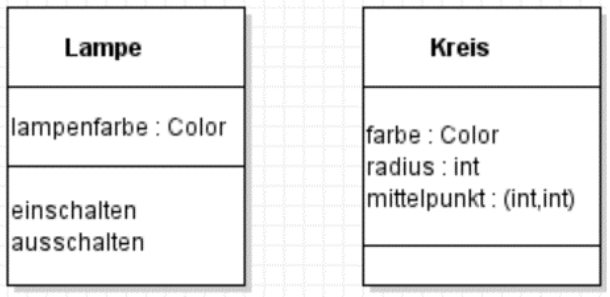
- Dienstleistungen
- können von einem Objekt angefordert werden
- durch ihre Signatur (Operationsname, Parameter, gegebenenfalls Rückgabetyp) beschrieben

Lösung Aufgabe 2.4



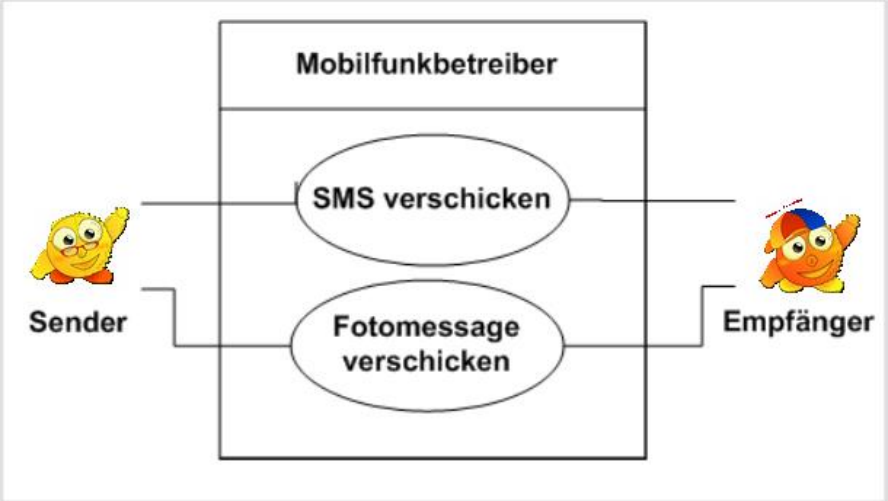
Lösung Aufgabe 2.5

So ungefähr könnte deine Lösung aussehen. Wenn du andere Attribute und Operationen gefunden hast, ist das genauso richtig. Wichtig ist, dass die Attribute in der Mitte stehen, die Operationen unten und beide mit kleinen Buchstaben anfangen.



Additum Kapitel 2

Weil du besonders schnell bei der Bearbeitung deiner Aufgaben gewesen bist, hast du nun die Chance mehr zu lernen als die anderen. Glückwunsch!



Dir ist sicherlich schon die Grafik auf der Titelseite des Leitprogramms aufgefallen. Bei diesem Diagramm handelt es sich nicht um ein Klassendiagramm, sondern um ein Anwendungsfalldiagramm (Use Case Diagram). Die Infodots (in violet sind es Strichmenschen), hier „Empfänger“ und „Sender“, stehen für Akteure, während die Informationen in den Ellipsen („SMS verschicken“, „Fotomessage verschicken“) Anwendungsfälle genannt werden. Von Bedeutung sind des Weiteren die Beziehungen zwischen den Elementen.

Das Anwendungsfalldiagramm eignet sich sehr gut für die Arbeit in deinem Betrieb, denn es bildet den Kontext und die Gliederung für eine Beschreibung, wie mit einem Geschäftsfall umgegangen wird. Mit Anwendungsfällen beschreibt man vor allem die Kommunikation mit den zukünftigen Anwendenden, den Auftraggebenden, der Fachabteilung o. Ä..

Aufgabe A.2.1

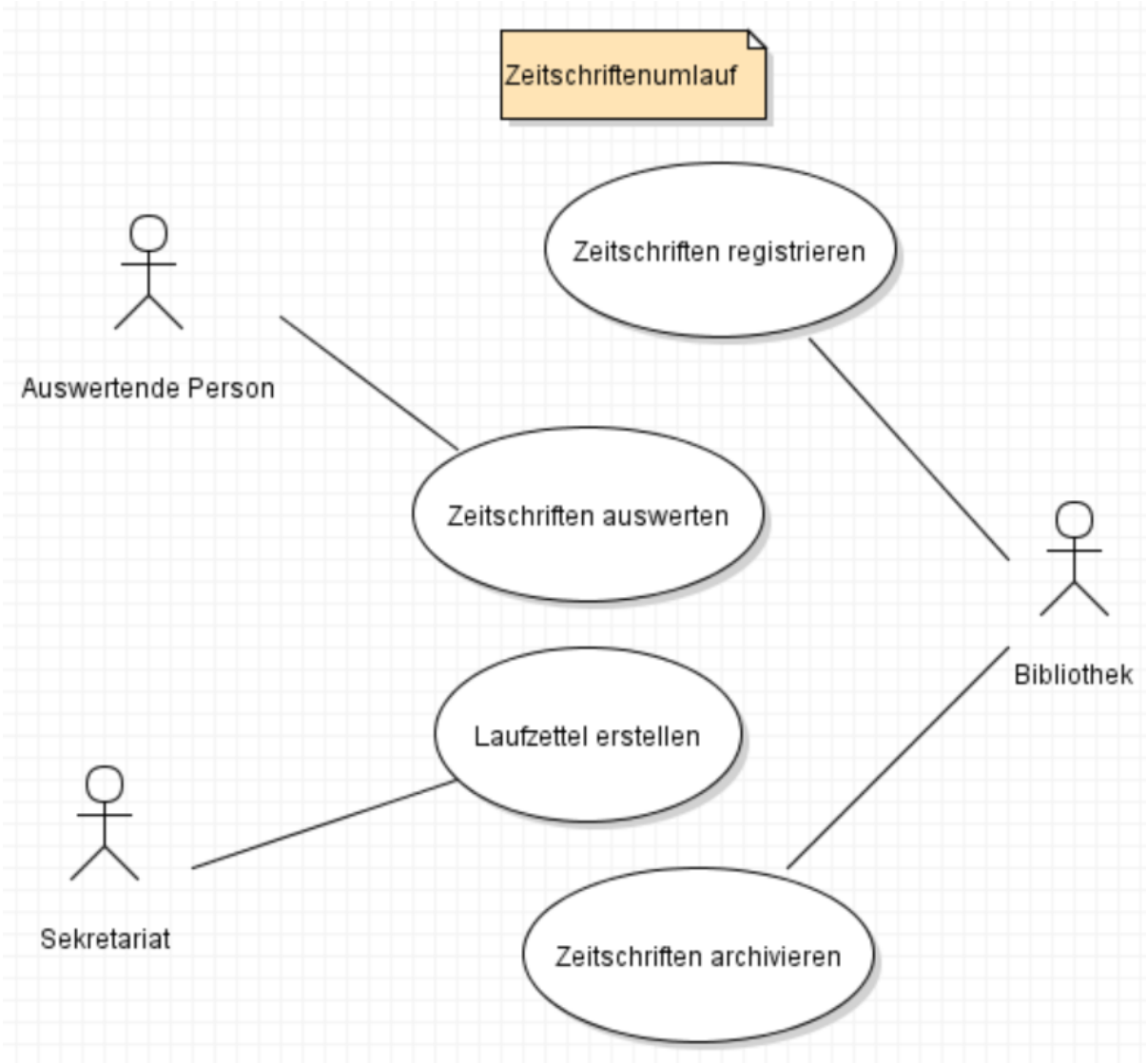
Zu Werbezwecken erstellt deine Firma eine Zeitung. Du hast nun die Aufgabe, das Anwendungsfalldiagramm zu erstellen, welches den Geschäftsprozess des Zeitschriftenumlaufs unterstützt. Beim Zeitschriftenumlauf wird jede neu erstellte Zeitschrift zunächst von der Bibliothek registriert. Dann werten die Mitarbeitenden die Zeitschrift inhaltlich aus. Welche Mitarbeitenden die Zeitschrift Korrektur gelesen haben, soll auf einem Laufzettel vermerkt werden. Dieser wird im Sekretariat erstellt. Wenn die letzte Person die Zeitschrift gelesen hat, wird diese in der Bibliothek archiviert. Dies dient zur Dokumentation des Werbeerfolgs.

Zeichne das Anwendungsfalldiagramm mit Bleistift und Papier!

Aufgabe A.2.2

Zeichne das Diagramm mit Hilfe von violet! Wähle dazu das Anwendungsfalldiagramm unter **File -> New -> Dynamic view -> Use Case Diagram**! In der Toolleiste hast du als erstes Symbol wie gewohnt das Auswahlwerkzeug. Mit dem *Actor*-Werkzeug erstellst du Akteure und mit dem „*Use case*“-Werkzeug die Anwendungsfälle. Die Akteure werden mit dem *Interaction*-Werkzeug den Anwendungsfällen zugeordnet. Einen Kasten können wir mit violet nicht erstellen. Deswegen fügst du den Diagrammnamen als Notiz mit dem *Note*-Werkzeug hinzu.

Lösung Aufgabe A.2.1/A.2.2



Kapiteltest Kapitel 2

Aufgabe 1: Definiere den Begriff objektorientiertes Design!

_____

_____

_____

_____

Aufgabe 2: Was ist unter der objektorientierten Analyse zu verstehen?

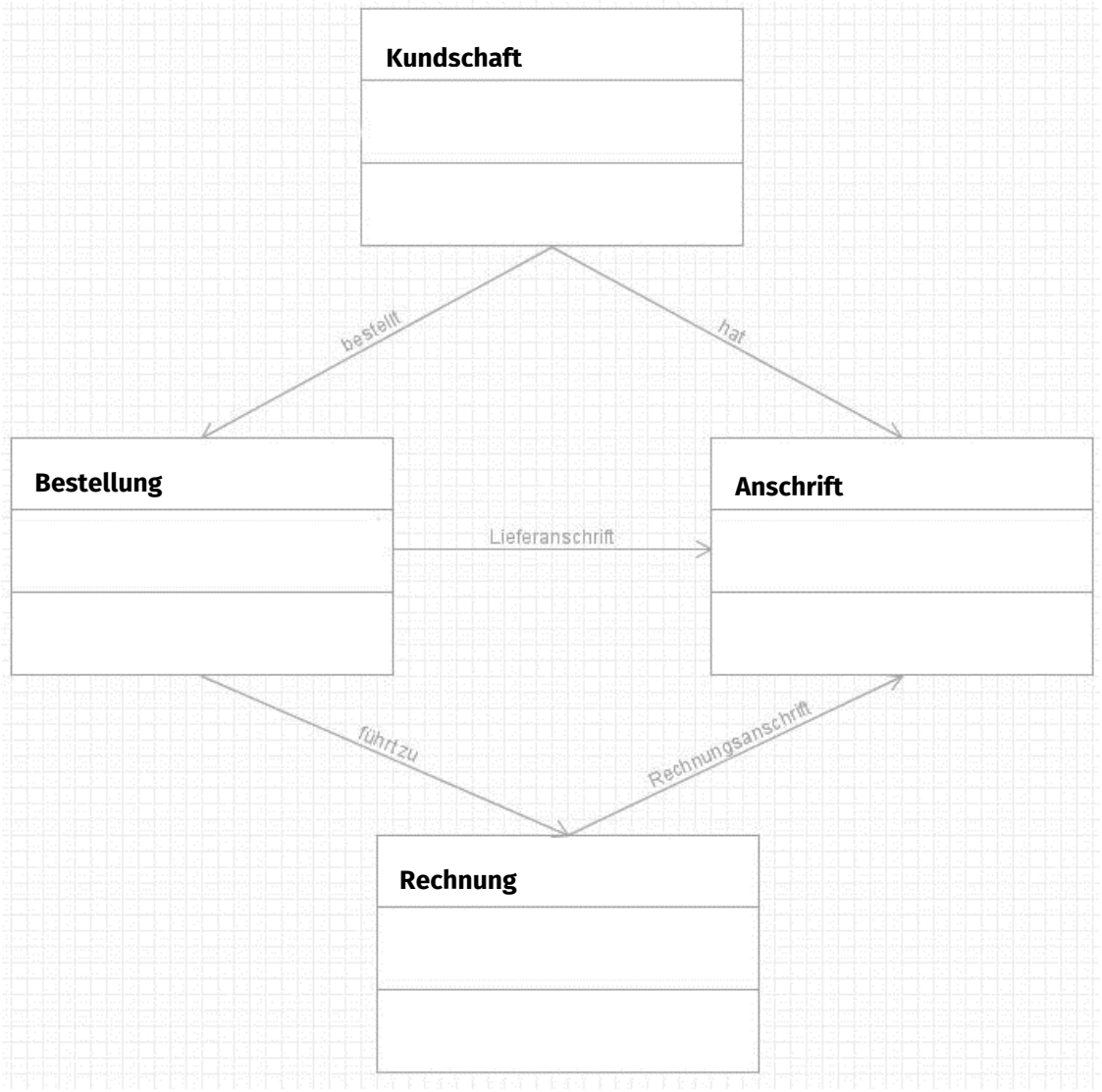
_____

_____

_____

_____

Aufgabe 3: Vervollständige das folgende Klassendiagramm, indem du geeignete Attribute und Operationen hinzufügst!



Lösung Frage 1:

- die kreative Erarbeitung eines Lösungs-bzw. Bearbeitungskonzepts
- Die Beziehungen zwischen den Klassen, sowie die Bezeichnung der Attribute und Operationen zeigen auf, wie das System die vorhandenen Anforderungen grundsätzlich erfüllt.

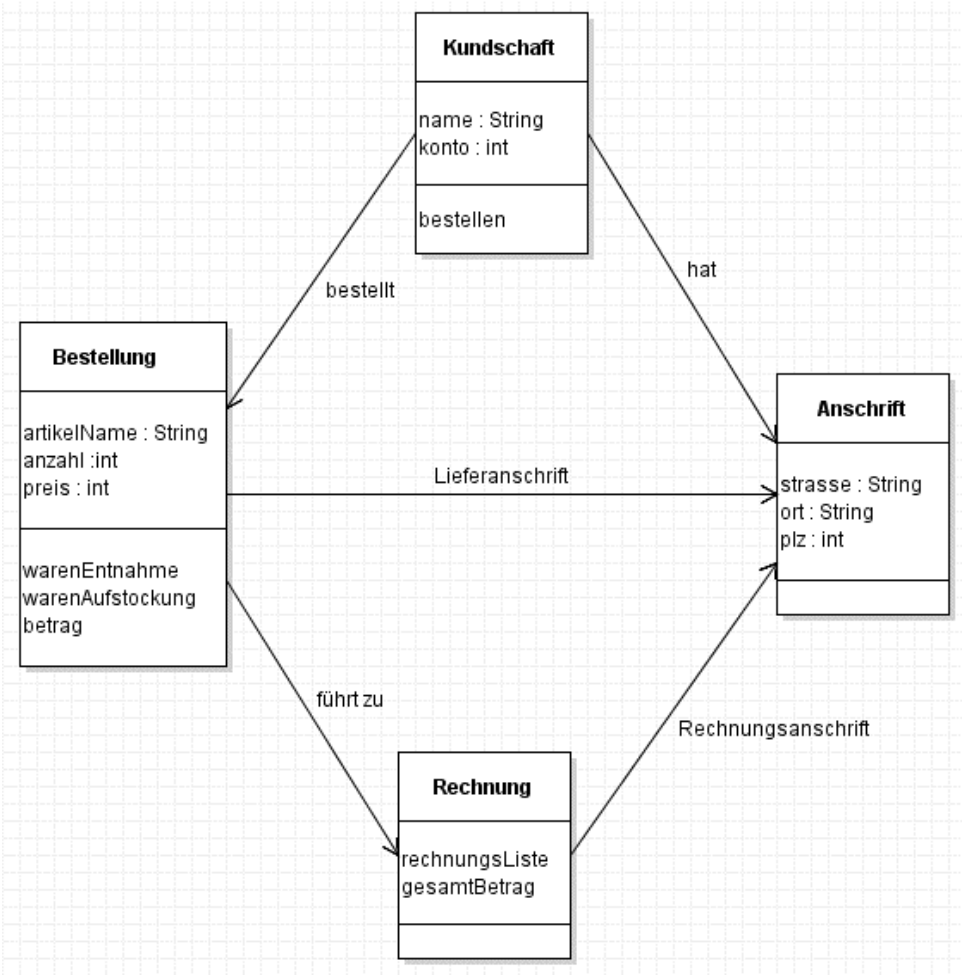
Lösung Frage 2:

- erster Schritt des Softwareentwicklungsprozesses
- wird geklärt, was das System leisten soll und
- alle Klassen werden aufgezählt, die bei der Ermittlung, Klärung und Beschreibung der Anforderungen benötigt werden

Lösung Frage 3:

Klassenname
attribute
operationen

Lösung Frage 4:



Super! Mr. Smith ist von deiner Arbeit so angetan, dass eine Gehaltserhöhung fällig ist!! Glückwunsch!

KAPITEL 3: WIE FUNKTIONIERT EINE AMPELSCHALTUNG?

Übersicht

Da du nun mehr Geld verdienst, erwarten dich auch anspruchsvollere Aufgaben. Nachdem du dich zuletzt mit den Abläufen in deiner Firma befasst hast, darfst du nun an einem konkreten Projekt teilnehmen. Deine Firma beteiligt sich an der Erstellung einer virtuellen Ampelschaltung. Dazu wirst du, neben den bereits erlernten Komponenten eines Klassendiagramms, in diesem Kapitel Beziehungselemente wie Vererbung, Aggregation und Komposition erlernen.

Abb. 2: Ampel

Lernziele

Am Ende dieses Kapitels kannst du die Beziehungselemente (Vererbung, Aggregation, Komposition) erklären und diese in violet darstellen.

Bevor du dein Projekt „Ampelschaltung“ beginnen kannst, machst du dir noch einmal das Prinzip der Vererbung klar.

Aufgabe 3.1

Fülle die Lücken aus! Folgende Begriffe könnten eingesetzt werden: Oberklasse – Attribute – Eigenschaften – Klassendiagramm – Komposition – Unterklasse – Methoden – Operationen – Klasse – Design

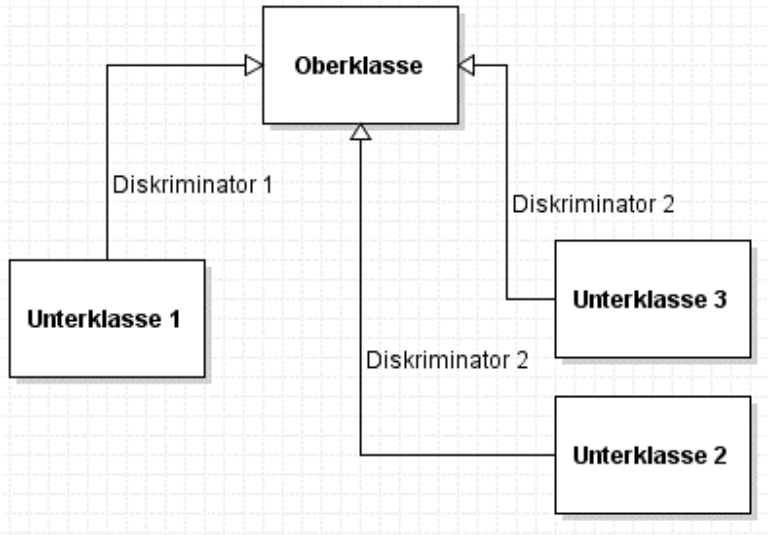
Vererbung kennst du aus dem Biologieunterricht (Stammbaum der Artverwandtschaften) oder aus dem täglichen Leben (dein Stammbaum). Diese Idee der Vererbung kann man für das Programmieren nutzen! Grundlegend für die Vererbung ist die Eltern-Kind-Relation: Es werden _____ von den Eltern auf das Kind vererbt. Was könnte man beim Programmieren vererben? Ganz einfach: Eine Klasse besteht aus _____und _____, und die können vererbt werden! Bei dieser Vererbung werden diese Informationen der _____ zugänglich für die _____ gemacht.

Aufgabe 3.2

Nun zurück zu deinem Projekt der Ampelschaltung! Wie könnte eine Vererbungsrelation der bereits besprochenen Klassen **Kreis** und **Lampe** aussehen? Unterscheide in Ober- und Unterklasse! Nun weißt du wieder, was Vererbung bedeutet. Allerdings fehlt dir zur Ausführung deiner Aufgabe noch das Wissen über die Notation von Vererbung in violet. Was für ein Glück! Da kommt deine Kollegin Regina und hilft dir auf die Sprünge:

Notation von Vererbung in violet:

Die Vererbung zwischen Unterklasse und Oberklasse stellt man mit einem durchgezogenen Pfeil mit einer unausgefüllten, dreieckigen Spitze dar. Der Pfeil kann mit dem Diskriminator beschriftet werden. Er drückt die Unterscheidungsmerkmale der Unterklasse zur Oberklasse aus:



Der Diskriminator unterscheidet Ober- und Unterklasse. Er ist demnach ein Unterscheidungsmerkmal, d.h. ein Charakteristikum. Der Diskriminator ist nicht von selbst gegeben, sondern das Ergebnis einer Modellierungsentscheidung. Zum Beispiel könnten Fahrzeuge mit Hilfe des Diskriminators *Antriebsart* untergliedert werden (Verbrennungsmaschine, Elektrisch, Pferdekraft); ebenso aber auch aufgrund des *Fortbewegungsmediums* (Luft, Wasser, Schiene, Straße). Fahrzeug ist demnach die Oberklasse, während die Begriffe, die in den Klammern aufgelistet sind, die Unterklassen darstellen.

Pfeile und die Benennungen der Pfeile hast du ja bereits gezeichnet. Nun kennst du auch die Bedeutung einer weiteren Pfeilart.

Aufgabe 3.3

So, und jetzt öffne dein violet-Programm! Zeichne eine sinnvolle Vererbungsbeziehung zwischen **Lampe** und **Kreis**!

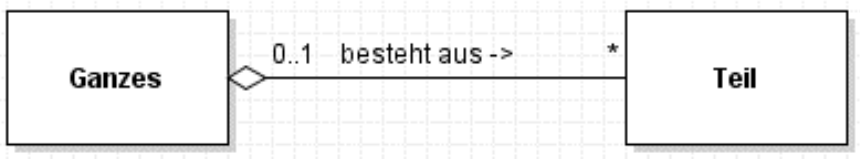
Somit hast du bereits einen Teil deiner Ampelschaltung entworfen. Um einen vollständigen ersten Entwurf zu erlangen, benötigst du allerdings noch ein paar Zusatzinformationen. Dies sind Informationen über das Konzept der Aggregation bzw. der Komposition. In der Bibliothek deiner Stadt findest du ein Buch, in dem folgendes angegeben ist:

Definition Aggregation

Eine Aggregation beschreibt, wie sich etwas Ganzes aus seinen Teilen logisch zusammensetzt. Zum Beispiel besteht eine Klasse aus mehreren Schülerinnen und Schülern. Damit wird verdeutlicht, dass die beteiligten Klassen keine gleichwertige Beziehung führen, sondern eine Ganzes-Teil-Hierarchie darstellen. Unter einer Aggregation versteht man eine Zusammensetzung eines Objektes aus einer Menge von Einzelteilen.

Notation Aggregation

Eine Aggregation wird wie eine Assoziation als Linie zwischen zwei Klassen dargestellt und zusätzlich mit einer kleinen Raute versehen. Die Raute steht auf der Seite des Aggregats, also des Ganzen. Sie symbolisiert gewissermaßen das Behälterobjekt, in dem die Einzelteile gesammelt sind. Im Übrigen gelten alle Notationskonventionen der Assoziation.



Dieses Diagramm würde man als „Kein oder ein Ganzes besteht aus keinem, einem oder mehreren Teilen“ lesen.

Definition Kardinalität

Die Kardinalität (oder auch Multiplizität bzw. Wertigkeit) gibt an, mit wie vielen anderen Objekten einer Klasse eine Beziehung eingegangen werden kann oder muss (je nach Art der Beziehung zum Beispiel Aggregation).

Die Kardinalitätsangabe wird an den beiden Enden des Pfeils eingetragen. Auf der Seite des Aggregats ist sie häufig 1, so dass sein Fehlen der Angabe standardmäßig als 1 interpretiert werden kann. Ein Teil kann gleichzeitig zu mehreren Aggregationen gehören.

Folgende Kardinalitäten sind beispielsweise möglich:

Kardinalität	Bedeutung
1	genau eins
0..1	keins oder eins
1..10	eins bis zehn
n	n-viele
*	keins, eins oder mehrere
1..*	mindestens eins

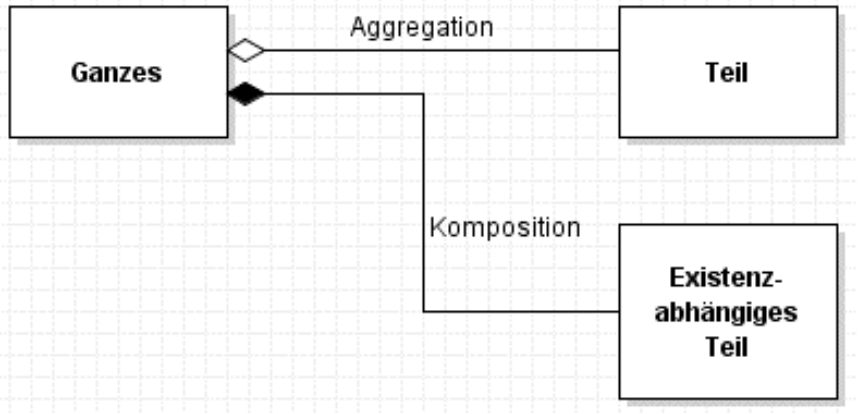
Aufgabe 3.4

Du versuchst nun, die Idee der Aggregation auf dein Firmenleben zu übertragen. Mögliche Klassen sind Unternehmen, Abteilung und Mitarbeitende. Zeichne mit violet das Modell für eine Unternehmenshierarchie, indem du den Klassen die Aggregationspfeile und die Kardinalitätsangaben hinzufügst!

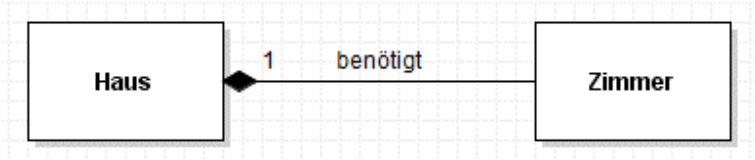
Du schlägst wieder das Buch aus der Bibliothek auf:

Definition Komposition

Eine Komposition ist eine strenge Form der Aggregation, bei der die Teile vom Ganzen existenzabhängig sind. Sie beschreibt, wie sich etwas Ganzes aus Einzelteilen zusammensetzt und diese kapselt.



Beispiel: Das Haus ist das Ganze. Damit ein Haus existieren kann, muss es mindestens ein Zimmer besitzen. Ein Zimmer kann aber auch nicht ohne Haus existieren. Demnach ist das Zimmer ein existenzabhängiger Teil eines Hauses. Wird das Haus gelöscht/abgerissen, verschwinden auch die Zimmer.



Notation

Die Komposition wird wie die Aggregation als Linie zwischen zwei Klassen gezeichnet und mit einer kleinen Raute auf der Seite des Ganzen versehen. Im Gegensatz zur Aggregation wird die Raute jedoch ausgefüllt.

Nun kannst du das Buch endlich zurückgeben, denn du weißt alles, was du für den Entwurf deiner Ampelschaltung benötigst.

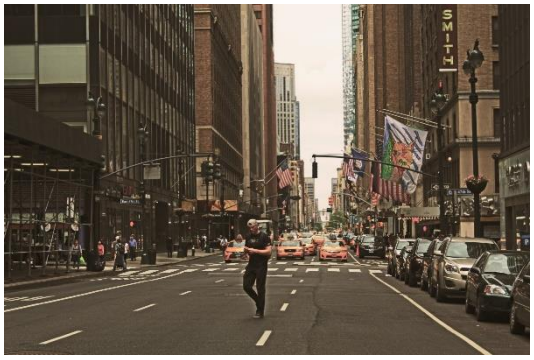


Abb. 3: Ampelkreuzung

Aufgabe 3.5

Los geht's! Vervollständige mit dem hinzugewonnenen Wissen deine Ampelschaltung! Dein Chef wird schon langsam ungeduldig! Dazu kannst du das Diagramm mit den Klassen **Kreis** und **Lampe** verwenden. Füge ihnen die Klassen **Rechteck**, **Ampel** und **Steuerung** hinzu. Zeichne zu diesen Klassen die Beziehungen, Attribute, Operationen und Kardinalitätsangaben!

Lösung Aufgabe 3.1 (Definition Vererbung)

Vererbung kennst du aus dem Biologieunterricht (Stammbaum der Artverwandtschaften) oder aus dem täglichen Leben (dein Stammbaum). Diese Idee der Vererbung kann man für das Programmieren nutzen! Grundlegend für die Vererbung ist die Eltern-Kind-Relation: Es werden Eigenschaften von den Eltern auf das Kind vererbt. Was könnte man beim Programmieren vererben? Ganz einfach: Eine Klasse besteht aus Methoden/Operationen und Eigenschaften/Attribute, und die können vererbt werden! Bei dieser Vererbung werden diese Informationen der Oberklasse zugänglich für die Unterklasse gemacht.

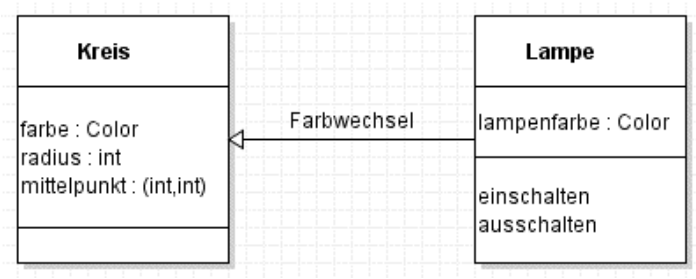
Lösung Aufgabe 3.2

- Oberklasse: **Kreis**
- Unterklasse: **Lampe**

Begründung:

Die Klasse **Lampe** gehört zur Klasse der Kreise. Eine Lampe ist der Kreis, der auch die Farbe wechseln kann. Demnach haben die Eigenschaften der Oberklasse **Kreis** eine allgemeinere Bedeutung als die Eigenschaften der Unterklasse **Lampe**. Die spezielleren Eigenschaften der Lampe sind denen des **Kreises** untergeordnet. Die Oberklasse gibt ihre Eigenschaften an die Unterklasse weiter. Die Eigenschaften werden vererbt. Die Unterklasse besitzt demnach die speziellen Eigenschaften, wie im Beispiel die jeweilige Farbe und die vererbten Eigenschaften der Oberklasse. In den Unterklassen können die Eigenschaften der Oberklasse überschrieben und erweitert, aber nicht eliminiert oder unterdrückt werden.

Lösung Aufgabe 3.3



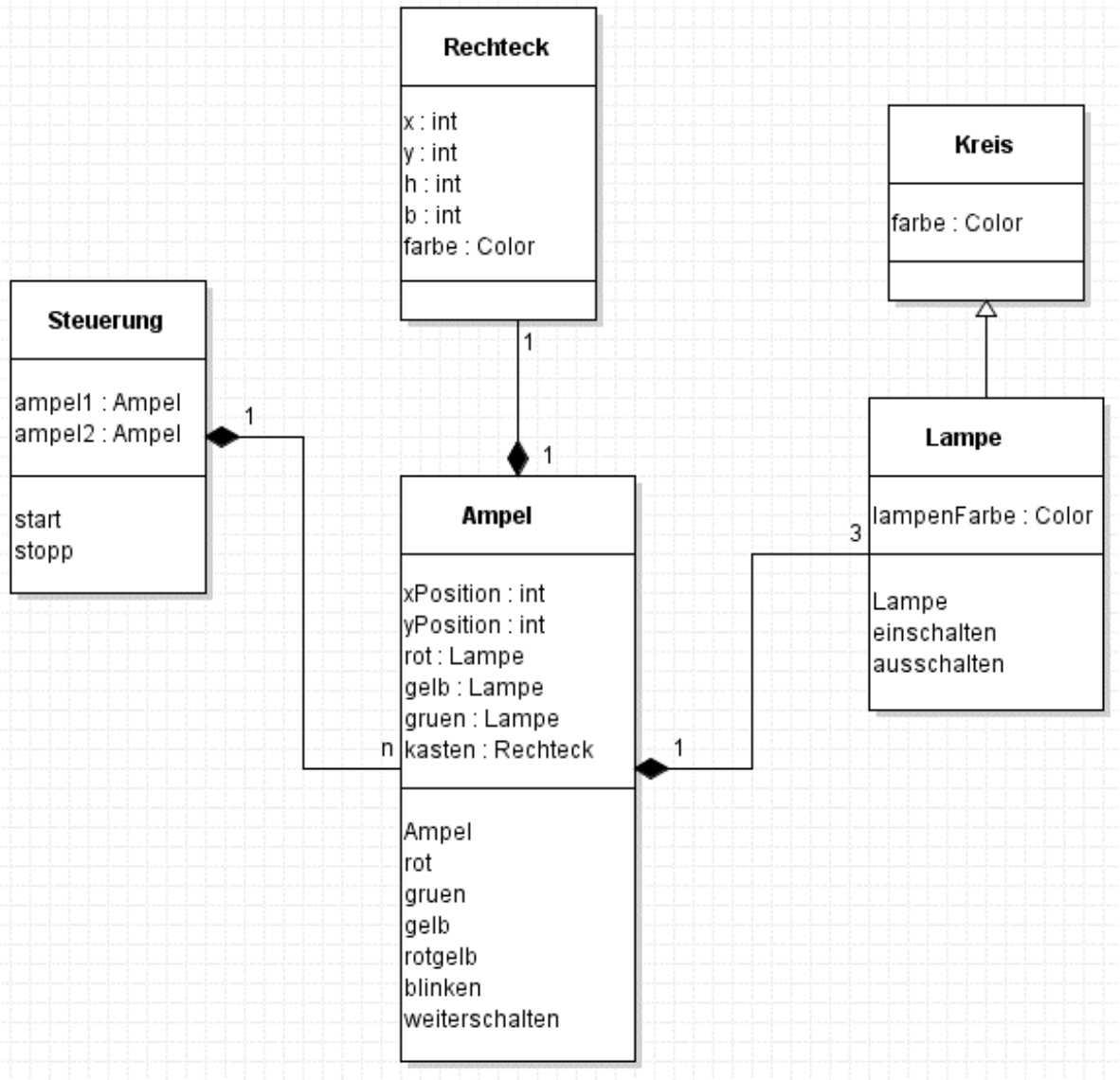
Lösung Aufgabe 3.4

Wir haben das Unternehmen in die Sektionen Unternehmen, Abteilung und Mitarbeitende unterteilt. Dann entsteht folgendes Klassendiagramm:

In einem Unternehmen gibt es also beliebig viele Abteilungen (die nur zu einem Unternehmen gehören). Jede Abteilung besteht aus beliebig vielen Mitarbeitenden, wobei jede und jeder Mitarbeitende nur zu einer Abteilung gehört.

Lösung Aufgabe 3.5

So ungefähr sollte deine Lösung aussehen. Falls du ein anderes Ergebnis hast, zeige es ggf. deiner Lehrkraft oder schreibe uns eine E-Mail, falls du keine andere Person fragen kannst (infosphere-support@informatik.rwth-aachen.de)! Vielleicht ist es ja auch gut.


Rechteck
x : int
y : int
h : int
b : int
farbe : Color
Kreis
farbe : Color
Steuerung
ampel1 : Ampel
ampel2 : Ampel
start
stopp
1
1
1
Lampe
lampenFarbe : Color
Lampe
einschalten
ausschalten
3
Ampel
xPosition : int
yPosition : int
rot : Lampe
gelb : Lampe
gruen : Lampe
kasten : Rechteck
n
1
Ampel
rot
gruen
gelb
rotgelb
blinken
weiterschalten

Additum Kapitel 3

Du hast deine Arbeiten so schnell erledigt, dass du jetzt als Weiterbildungsmaßnahme die Möglichkeit bekommst, deine Kenntnisse in UML und violet zu erweitern. Super!

Es wird dir ein weiteres Werkzeug vorgestellt:

Definition Objektdiagramm

Es hat eine ähnliche Struktur wie das Klassendiagramm. Es besitzt jedoch an Stelle von Klassen beispielhaft eine Auswahl der zu einem bestimmten Zeitpunkt existierenden Objekte mit ihren augenblicklichen Werten. Demnach ist ein Objektdiagramm sozusagen ein Schnappschuss der Objekte im System zu einem bestimmten Zeitpunkt.

Aufgabe A.3.1

Beschreibe die Unterschiede des Objektdiagramms zum Klassendiagramm!

_____
_____
_____
_____
_____

Du weißt nun, dass ein Objektdiagramm aus Objekten und nicht aus Klassen besteht. Falls du dir aber nicht sicher bist, was ein Objekt ist, lies dir die folgende Definition durch!

Definition Objekt

Das Objekt beinhaltet und speichert Informationen in Attributen, deren Anzahl und Aussehen durch den Bauplan in der Klasse festgesetzt ist. Genauso sind die verfügbaren Operationen, mit denen das Objekt mit dem Programm kommunizieren kann, in der Klasse festgelegt. Der Inhalt der Attribute kann jedoch von Objekt zu Objekt verschieden sein.

Objekte sind also Exemplare der Klasse, die im Programm agieren und nicht die Klasse selbst. Als Beispiel definiert man eine Klasse **Skatspieler mit eigener Strategie**; Skatspieler sind dann Objekte dieser Unterklasse. Hinzu kommt ein Objekt **Tisch**, welches die abgelegten Karten verwaltet und die Punkte anschreibt.

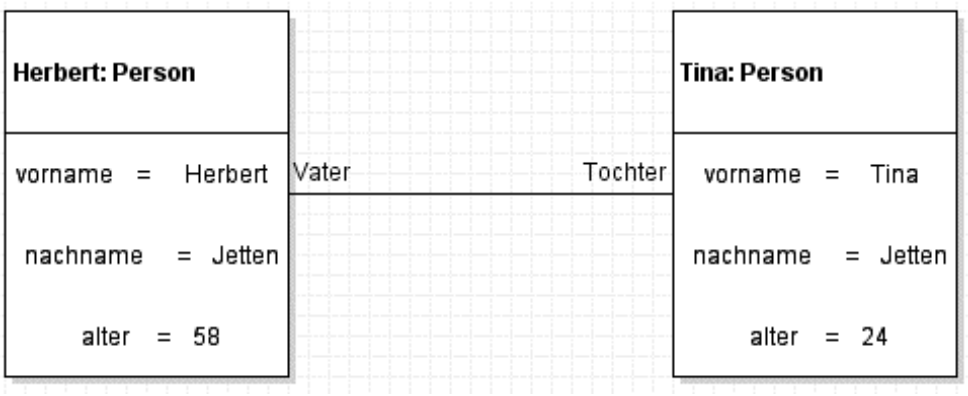
Aufgabe A.3.2

Was ist der Unterschied zwischen Klasse und Objekt?

_____
_____
_____
_____
_____

Nachdem du die Definition gelesen hast, fällt dir ein passendes Beispiel aus deiner Verwandtschaft ein. Die Verwandtschaftsbeziehungen der Familie Jetten lassen sich nämlich ganz gut als Objektdiagramm darstellen.

Du weißt, dass die Darstellung von Objekten ähnlich ist wie die Darstellung von Klassen. Nach dem Objektnamen kann man mit **: Klassenname** verdeutlichen, zu welcher Klasse das Objekt gehört. Im Beispiel ist dieser „: Person". Unter einer horizontalen Linie werden die Attribute aufgelistet mit ihren Werten. Ein Beispiel für ein Attribut ist „vorname" mit dem Wert „Herbert".



Aufgabe A.3.3

Da fällt dir ein, dass Tina auch einen Bruder namens Michael hat. Ergänze das obige Diagramm um dieses weitere Objekt! Starte dazu ein neues Objektdiagramm in violet unter **File -> New -> Static View -> Object Diagram**! Hier gibt es wieder eine neue Toolleiste. Neben dem Auswahlwerkzeug gibt es nun das *Object*-Werkzeug, mit dem du neue Objekte erstellen kannst. Attribute kannst du mit dem *Field*-Werkzeug den Objekten anfügen. Um den Attributen Namen und Werte zu geben, doppelklicke sie mit der rechten Maustaste. Mit dem „*Association between objects*"-Werkzeug kennzeichnest du die Beziehung zwischen den Objekten. Ziehe dazu die Linie von einem Objektnamen zum anderen, und beschrifte sie. **Vorsicht:** Die Linie kann auch an den Attributen angesetzt werden!

Falls du violet 3.0.0 verwendest, wundere dich nicht, wenn die erste Zeile der Attribute nicht funktioniert. Das ist ein bekannter Bug. Ignoriere die erste Zeile dann einfach, ab der zweiten sollte es funktionieren.

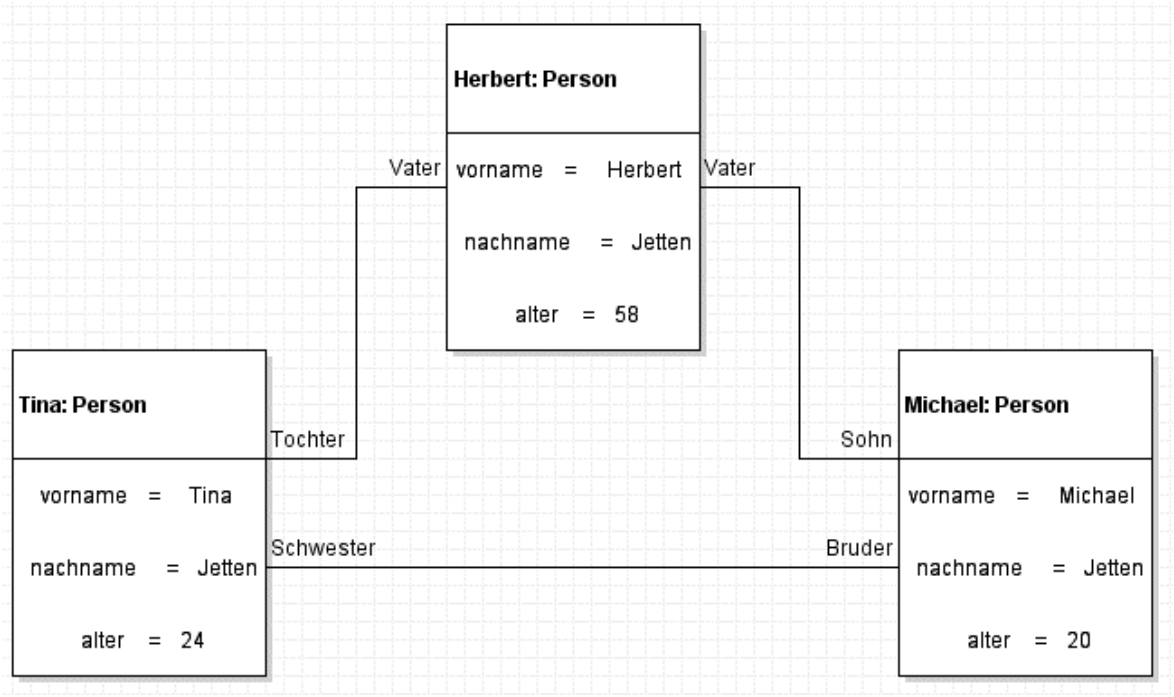
Lösung Aufgabe A.3.1

- besitzt keine Klassen
- besteht aus einer Auswahl von Objekten, die zu einem bestimmten Zeitpunkt existieren

Lösung Aufgabe A.3.2

- beinhaltet und speichert Informationen in Attributen

Lösung Aufgabe A.3.3

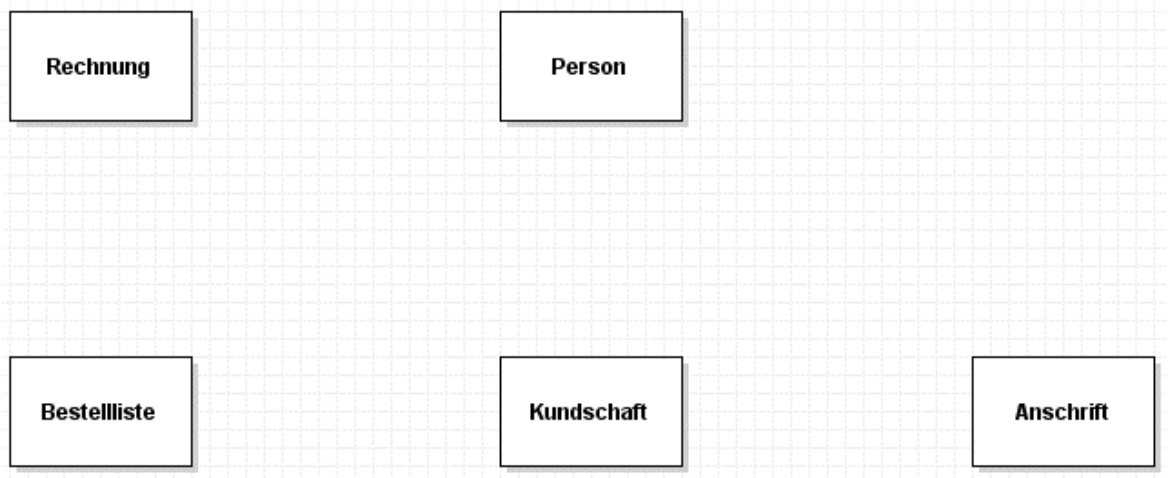


Kapiteltest Kapitel 3

Frage 1: Was versteht man unter Vererbung?

_____
_____
_____
_____
_____

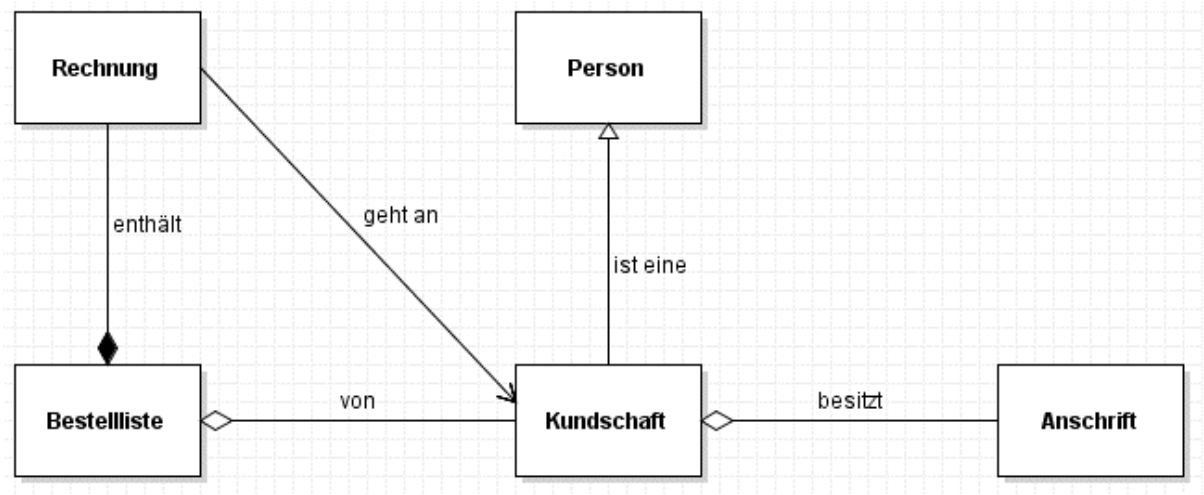
Frage 2: Dein Kollege hat die folgenden Klassen entworfen. Er hat allerdings keine Ahnung, wie er sie zueinander in Beziehung setzen kann. Hilf ihm, indem du die Kenntnisse über Beziehungselemente aus diesem Kapitel anwendest! Zeichne die passenden Beziehungspfeile!



Lösung Frage 1:

Vererbung ist ein Programmiersprachenkonzept, d.h. ein Umsetzungsmechanismus für die Relation zwischen Ober- und Unterklasse, wodurch Attribute und Operationen der Oberklasse auch den Unterklassen zugänglich gemacht werden.

Lösung Frage 2:



Herzlichen Glückwunsch! Das Resultat deiner Arbeit ist deine Beförderung!!

KAPITEL 4: WIE KANN DAS MODELL DER AMPELSCHALTUNG ERWEITERT WERDEN?

Übersicht

Du bist ziemlich stolz auf dich, denn immerhin bekleidest du nun schon die Position des stellvertretenden Chefs! Die Beförderung weckt in dir einen neuen Motivationsschub, und du möchtest dein Projekt der Ampelschaltung unbedingt vorantreiben. Demnach erarbeitest du in diesem Kapitel eine Erweiterung des Ampelmodells. Daher dient dieses Kapitel der Festigung der Kenntnisse der letzten Kapitel durch Anwendung des Gelernten.

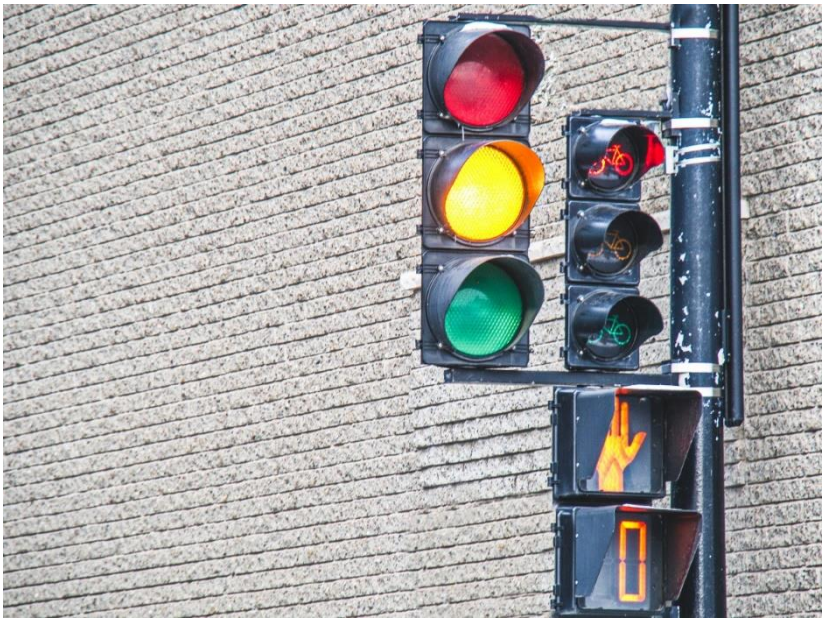


Abb. 4: Verschiedene Ampeln

Lernziele

Am Ende dieses Kapitels....

- kannst du zeigen, wie ein gegebenes Modell mit den Kenntnissen aus den vorherigen Kapiteln erweitert werden kann.
- kannst du erklären, weshalb es wichtig ist, zuerst ein Modell zu erstellen, bevor man mit der Implementierung beginnt.

Zunächst möchtest du in Erfahrung bringen, wie eine reale Ampelschaltung aufgebaut sein kann.

Aufgabe 4.1

Recherchiere im Internet auf der Seite http://de.wikipedia.org/wiki/Lichtzeichenanlage nach unterschiedlichen realen Ampelschaltungen, mit deren Hilfe das Modell aus Kapitel 3 erweitert werden kann!

_____
_____
_____
_____
_____
_____
_____

Aufgabe 4.2

Vervollständige das Modell aus Kapitel 3 um drei Erweiterungen aus Aufgabe 4.1!

(Du kannst zum Beispiel Fußgängerampel, Ausfallsicherung und Handschaltung einbauen)

Aufgabe 4.3

Um sicher zu gehen, dass dein Modell verständlich ist, fragst du am besten zwei bis drei deiner Kolleginnen und Kollegen um Rat. (Diese sind natürlich in Wirklichkeit deine Mitschülerinnen und Mitschüler!). Diskutiert über eure Ergebnisse! Fokussiert bei eurer Diskussion die Frage nach dem Sinn eines Modells wie euren Ampelmodellen in Bezug auf die spätere Implementation!

Falls du niemanden kennst, der gerade dieselben Aufgaben macht, versuche schon einmal, die Musterlösung nachzuvollziehen und mit deiner Lösung zu vergleichen.

Bemerkung:

- Ausfallsicherung sollte nicht Teil der Steuerung sein, da, wenn die Steuerung ausfällt, auch die Ausfallsicherung ausfällt.
- Handschaltung kann Teil der Steuerung oder der Ampel sein, je nachdem wie man seine Ampel konstruieren will.

Aufgabe 4.4

Trage die Vor- und Nachteile der Vorgehensweise „Erst modellieren, dann implementieren!“ in die folgende Tabelle ein:

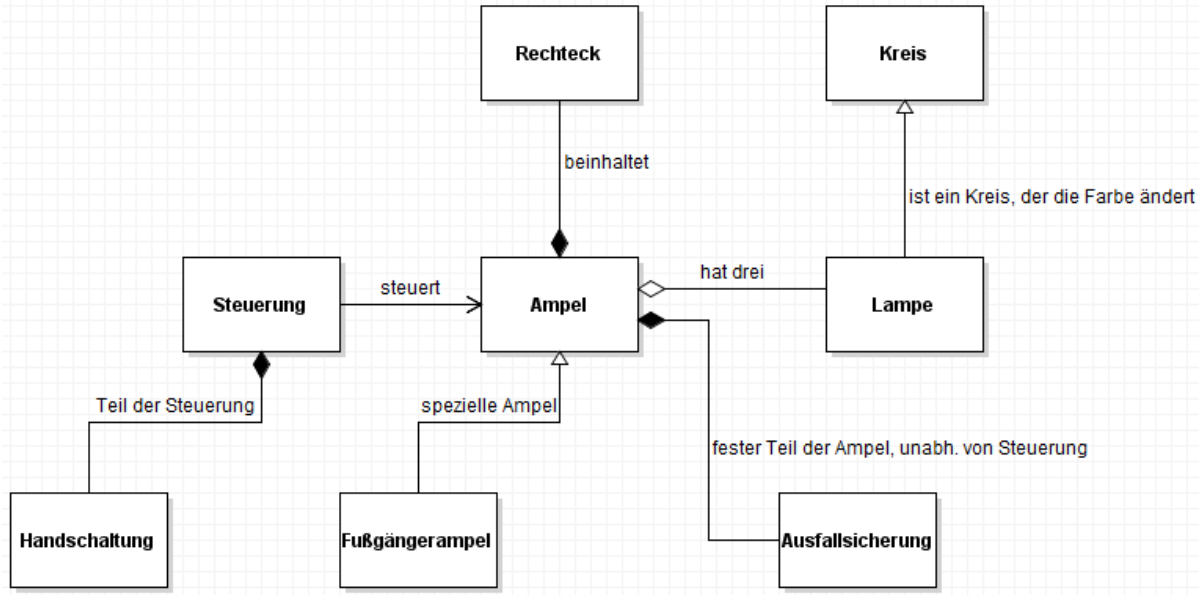
Vorteile	**Nachteile**

Lösung Aufgabe 4.1:

Wie du gesehen hast, gibt es viele Möglichkeiten zur Erweiterung des Modells, wie zum Beispiel die folgenden Innovationen:

- Fußgängerampel
- Ausfallsicherung
- Handschaltung
- ...

Lösung Aufgabe 4.2



Lösung Aufgabe 4.4

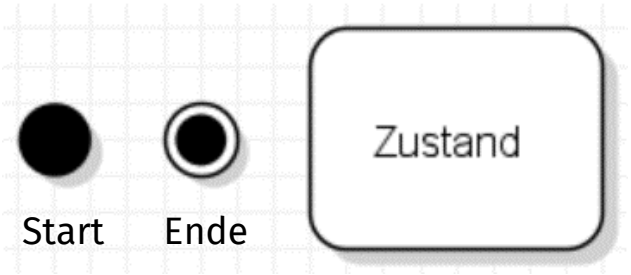
Vorteile	Nachteile
○ Mit Hilfe des Modells wird der Kern des Problems getroffen. ○ für Außenstehende daher schneller verständlich ○ nicht an einzelne Programmiersprachen gebunden ○ übertragbar	○ kostet mehr Zeit ○ kostet Flexibilität

Additum Kapitel 4

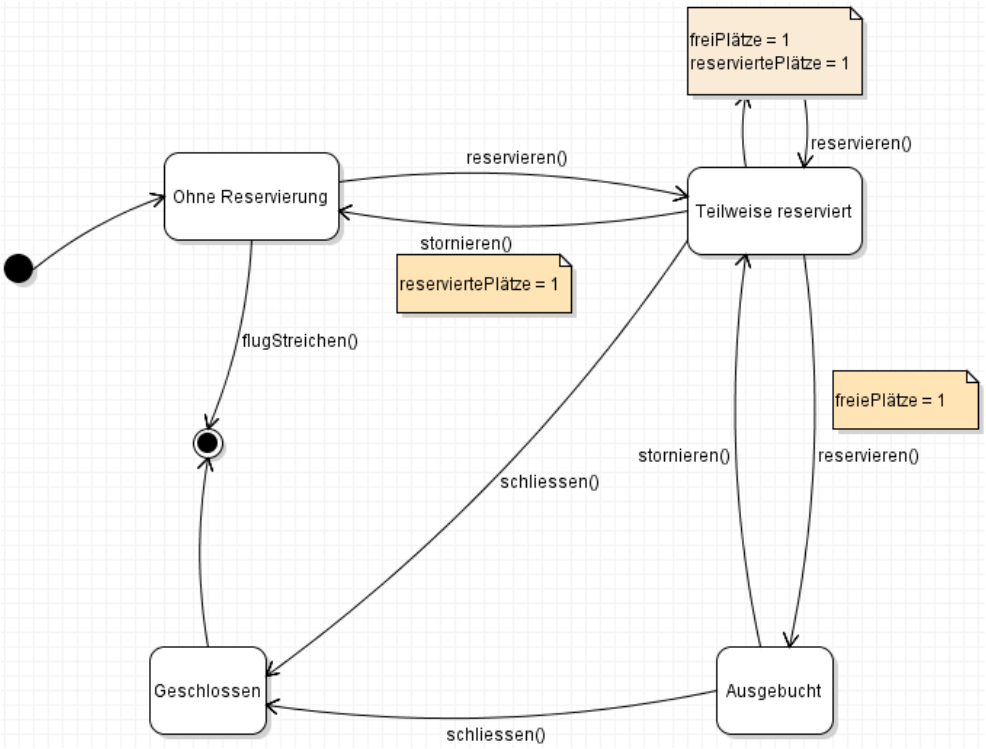
Nach dieser harten Arbeitsphase hast du dir wirklich ein paar Tage Entspannung verdient. Du planst eine Flugreise nach Gran Canaria. Während der Planung deiner Reise, stellst du gleichzeitig fest, wie viel Spaß dir die Arbeit mit Diagrammen bereitet. Du entwickelst schon Überlegungen zu Projekten nach deiner Reise und stellst dir vor, wie man die Abwicklung eines Fluges in einem Diagramm darstellen könnte. Bei deinen Internetrecherchen bezüglich dieses Themas findest du folgende Informationen:

Notation:

Zustände werden durch abgerundete Rechtecke dargestellt, in denen der Name des Zustandes steht. Um Diagramme übersichtlicher zu gestalten, können Zustände mehrfach in einem Diagramm vorhanden sein.



Das folgende Beispiel zeigt die Zustandsänderungen bei einer Flugreservierung. Wenn die Namen des Ereignisses und der Aktionsoperation übereinstimmen, ist nur die Operation aufgeführt. Bei Einrichtung des Fluges führt der Startzustand in den Zustand OhneReservierung. Bei Eintritt in diesen Zustand wird die Operation Ruecksetzen ausgeführt. Wird eine Reservierung für den Flug vorgenommen, wechselt das Objekt in den Zustand TeilweiseReserviert; mit dem Ereignis Reservieren (realisiert durch eine Operation) verbunden. In dieser Operation findet die eigentliche Reservierung statt. Und der interne Reservierungszähler wird aktualisiert. Nach Abschluss dieser Aktion befindet sich das Objekt im Zustand TeilweiseReserviert.



Jede weitere Reservierung führt zur selben Aktion. Solange noch freie Plätze vorhanden sind, bleibt das Objekt im Zustand TeilweiseReserviert. Ist nur noch ein Platz frei, wird in den Zustand Ausgebucht gewechselt. Die Stornierung von reservierten Plätzen erfolgt in ähnlicher Weise. Das Zustandsdiagramm beschreibt also, durch welche Ereignisse welche Aktionen ausgelöst werden und wann dies (und damit der Aufruf der entsprechenden Operationen) zulässig ist.

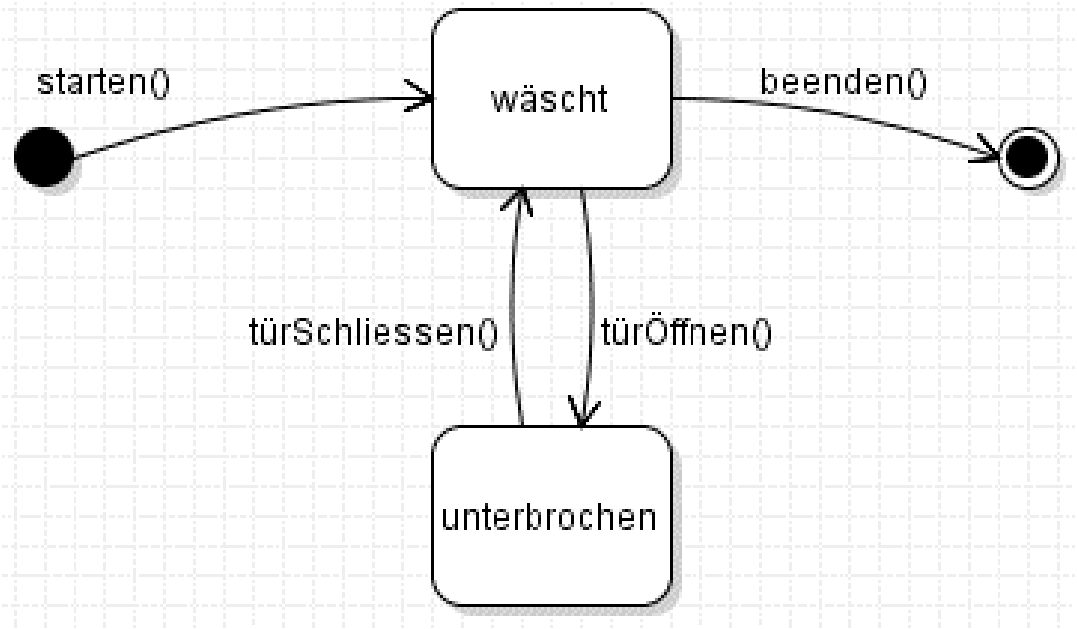
Aufgabe A.4.1

Da du auf Reisen gehst, musst du natürlich vorher deine Kleidung waschen. Als du vor der Waschmaschine sitzt, fällt dir ein, dass man den Waschvorgang als Zustandsdiagramm darstellen kann. Wichtig bei der Zustandsmodellierung ist, dass nur die bedeutenden Zustände bzw. Zustandsänderungen im Diagramm auftauchen. Die unterschiedlichen Waschphasen deiner Waschmaschine fassen wir in einem Zustand „wäscht“ zusammen. Außerdem benötigst du einen Start- und einen Endzustand. Mit einem weiteren Zustand kannst du eine Unterbrechung des Waschvorgangs darstellen.

Stelle das zugehörige Zustandsdiagramm mit violet dar!

Lösung Aufgabe A.4.1

Ungefähr wie das folgende Diagramm sollte deine Lösung aussehen:



Lernkontrolle Kapitel 4

Frage 1:

Nenne zwei Vorteile der Vorgehensweise „Erst modellieren, dann implementieren!“!

_____
_____
_____
_____
_____

Frage 2:

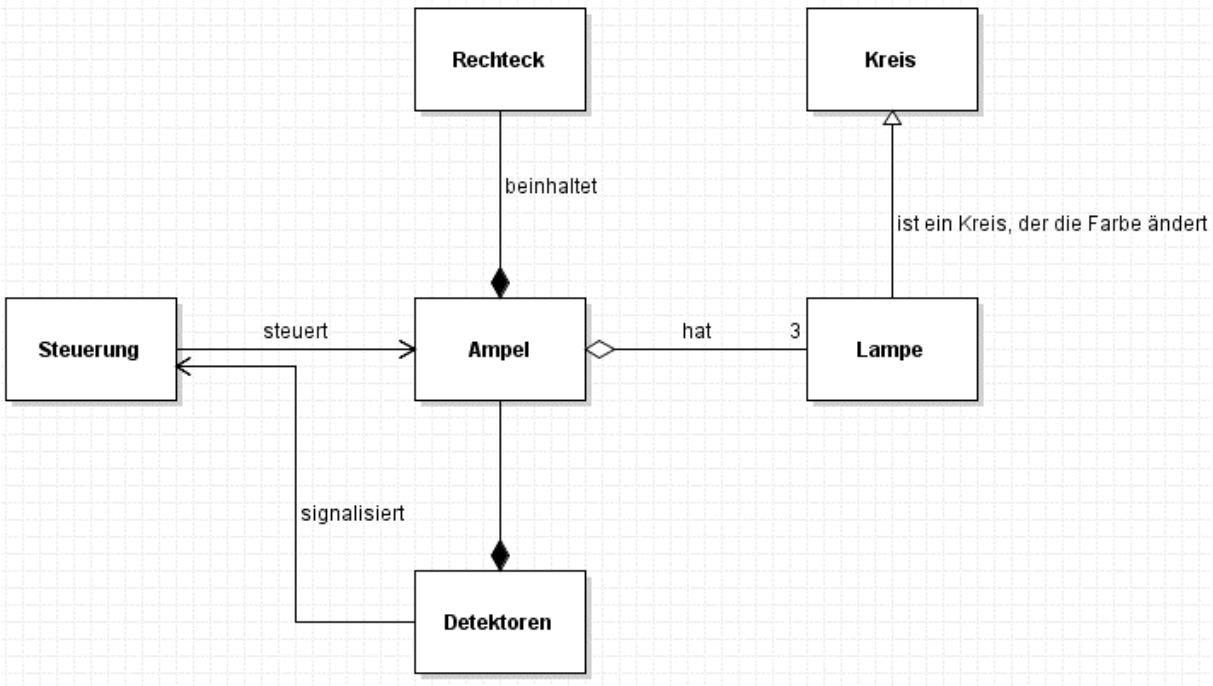
Ergänze das Modell aus Aufgabe 3 um Detektoren. Falls du nicht genau weißt, was das ist, schaue nochmals auf der Seite http://de.wikipedia.org/wiki/Lichtzeichenanlage nach!

Lösung Frage 1:

Mindestens zwei der folgenden Vorteile solltest du genannt haben:

- Mit Hilfe des Modells wird der Kern des Problems getroffen.
- für Außenstehende daher schneller verständlich
- nicht an einzelne Programmiersprachen gebunden
- übertragbar

Lösung Frage 2:



Endlich geschafft! Du erhältst eine weitere Gehaltserhöhung!

LITERATURANGABEN

Internetquellen

- http://www.mitp.de/imperia/md/content/vmi/1453/1453_kap01.pdf?PHPSESSID=259609dc36dc518f4c42c5
- UML-Editor violet, http://horstmann.com/violet/ (abgerufen am 18.02.2021)
- § 2 Objektorientierte Modellierung: Klassen und Vererbung, http://www.roro-seiten.de/info/oo/2oom.html (abgerufen am 18.02.2021)
- Spiegel, Walter: Vererbung & UML, http://www.wspiegel.de/pykurs/python14.html (abgerufen am 18.02.2021)
- Schäling, Boris: Kapitel 4. Das Klassendiagramm, in: Ders.: Der moderne Softwareentwicklungsprozess in UML, http://www.highscore.de/uml/klassendiagramm.html (abgerufen am 10.02.2021)
- Deutloff, Alexander: 4 Assoziationen in UML-Klassendiagrammen. Multiplizität, https://www.kstbb.de/informatik/oo/04/4_2_Multiplizitaet.html (abgerufen am 09.02.2021)

Literatur

- Oestereich, Bernd (2005): Die UML 2.0 Kurzreferenz für die Praxis. Kurz, bündig, ballastfrei. München/Wien: De Gruyter Oldenbourg.
- Volker, Claus und Schwill, Andreas (2003): Duden Informatik: Ein Fachlexikon für Studium und Praxis. Mannheim/Leipzig/Wien: Dudenverlag.

ABBILDUNGSVERZEICHNIS

- Abb. 1: https://pxhere.com/de/photo/1150893, CC0
- Abb. 2: https://pxhere.com/de/photo/1038830, CC0
- Abb. 3: https://pxhere.com/de/photo/1324974, CC0
- Abb. 4: https://pxhere.com/de/photo/1631158, CC0
- Arbeitssymbole , , , , , : Quelle: InfoSphere
- Alle UML-Diagramme erstellt von Christian Banyai/Tina Jetten mit Hilfe des Programmes violet (http://alexdp.free.fr/violetumleditor/page.php) und aktualisiert von Thordis Dusella