Kommunikation und Synchronisation von Prozessen

M. Jakob

Gymnasium Pegnitz

22. Januar 2017

Inhaltsverzeichnis

In diesem Abschnitt

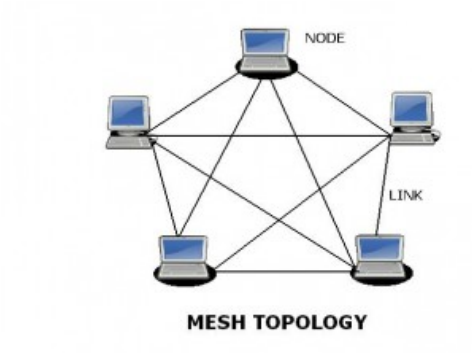
Begriffsbestimmung

Rechnertopologien

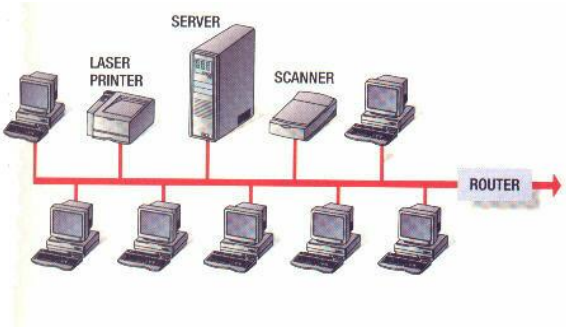
In kleinen Teileinheiten können Rechnerverbindungen direkt (point-to-point), sternförmig, busförmig oder ringförmig organisiert sein. Jede dieser Organisationsformen hat Vor- und Nachteile.

Verschiedene Topologien

point-to-point: Hierbei ist jeder Rechner mit jedem anderen verbunden.



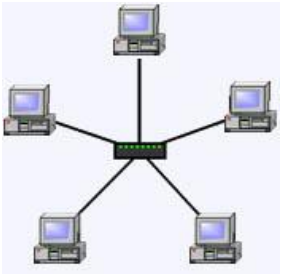
Bustopologie: Alle Netzwerkrechner sind an ein gemeinsames Übertragungsmedium angeschlossen.



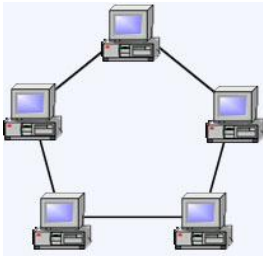
- Busanimation
  `http://thought1.org/nt100/module3/linear_bus.html`

Verschiedene Topologien

Sterntopologie: Jedes Endgerät wird direkt mit einem zentralen Knoten verbunden.



Token-Ring: Die Rechner sind ringförmig angeordnet. Jeder Rechner darf nur dann senden, wenn er ein bestimmtes Datenpaket (das Token) besitzt.

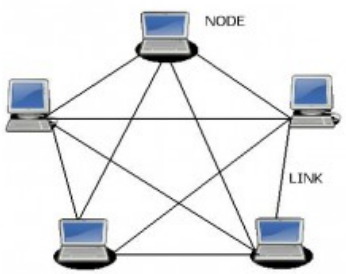


- Token-Ring bei YouTube
  `http://www.youtube.com/watch?v=50RUTSbTSR8`

Bewertungskriterien für Netzwerktopologien

- Störanfälligkeit
- Systempflege
- Kosten
- Übertragungsrate in Abhängigkeit vom Verkehr

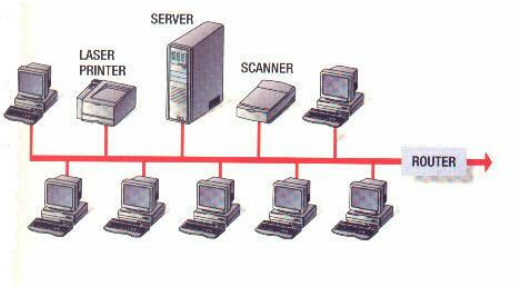
Beurteilung point-to-point



MESH TOPOLOGY

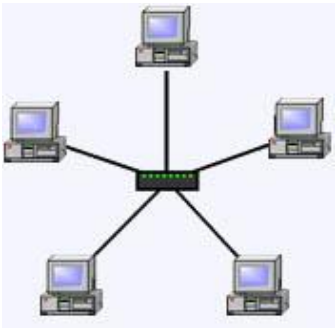
- Für n Rechner benötigt man $\frac{n(n-1)}{2}$ Verbindungen (also sau-viele).
  - nachrechnen
- Einfach aufzubauen.
- Sehr ausfallsicher, weil bei Störungen nur kleine Bereiche betroffen sind.
- Schon bei wenigen Rechnern nicht mehr administrierbar.

Beurteilung Bustopologie



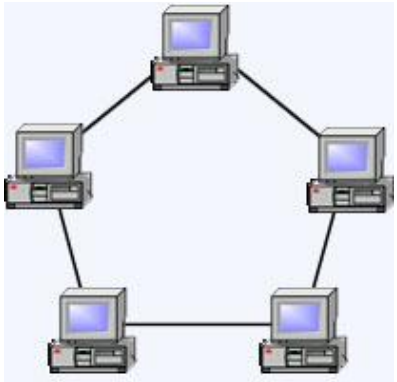
- kostengünstig und einfach
- Kollisionsmanagement nötig
- Empfangsmanagement nötig
- Sniffig möglich
- Störung der Datenleitung legt gesammte Kommunikation lahm.

Beurteilung Sterntopologie



- Kollisions- und Empfangsmanagement regelt der Zentralrechner.
- Zentralrechner ist teuer und darf nicht ausfallen.

Beurteilung Ringtopologie



- keine Kollisionen
- konstante (aber kleine) Übertragungsrate

Ausfallsicherheit

	positiv	negativ
Bus	Ausfall eines Gerätes hat für die Funktionalität des Netzwerkes keine Konsequenzen.	Störung des Übertragungsmediums an einer einzigen Stelle im Bus (defektes Kabel) blockiert den gesamten Netzstrang
Stern	Der Ausfall eines Endgerätes hat keine Auswirkung auf den Rest des Netzes	Durch Ausfall des zentralen Knotens wird Netzverkehr unmöglich
Token-Ring	Pendelbetrieb bei Unterbrechung einer Leitungsverbindung ist möglich	

Kosten und Erweiterbarkeit

	positiv	negativ
Bus	geringe Kosten, da nur geringe Kabelmengen und keine aktiven Netzwerkkomponenten nötig; leicht erweiterbar	
Stern		teurer Zentralrechner
Token-Ring	geringe Kosten, da nur geringe Kabelmengen und keine aktiven Netzwerkkomponenten nötig; leicht erweiterbar	

Sicherheit

	positiv	negativ
Bus		Datenübertragungen können leicht abgehört werden (Sniffing)
Stern		
Token-Ring		Datenübertragungen können leicht abgehört werden (Sniffing)

Übertragungsrate

	positiv	negativ
Bus		Reduktion der Übertragungsrate bei hohem Netzwerkverkehr durch Kollisionen
Stern		niedrige Übertragungsrate bei vielen Hosts, wenn nur ein Hub benutzt wird
Token-Ring	gleichzeitiges Senden wird durch Token vermieden, dadurch garantierte Übertragungsbandbreite	

Übung

- Vertiefung: Übersicht Topologien
`http://www.compu-seite.de/topologien.html`
- Ü1: S.67/3
- Ü2: Topologien im Alltagsgespräch erkennen
- Ü3: Traditionelle Kommunikationswege
`run:Arbeitsmaterial/Uebungen/TraditionelleKommunikationswege/TraditionelleKommunikationswege.pdf`

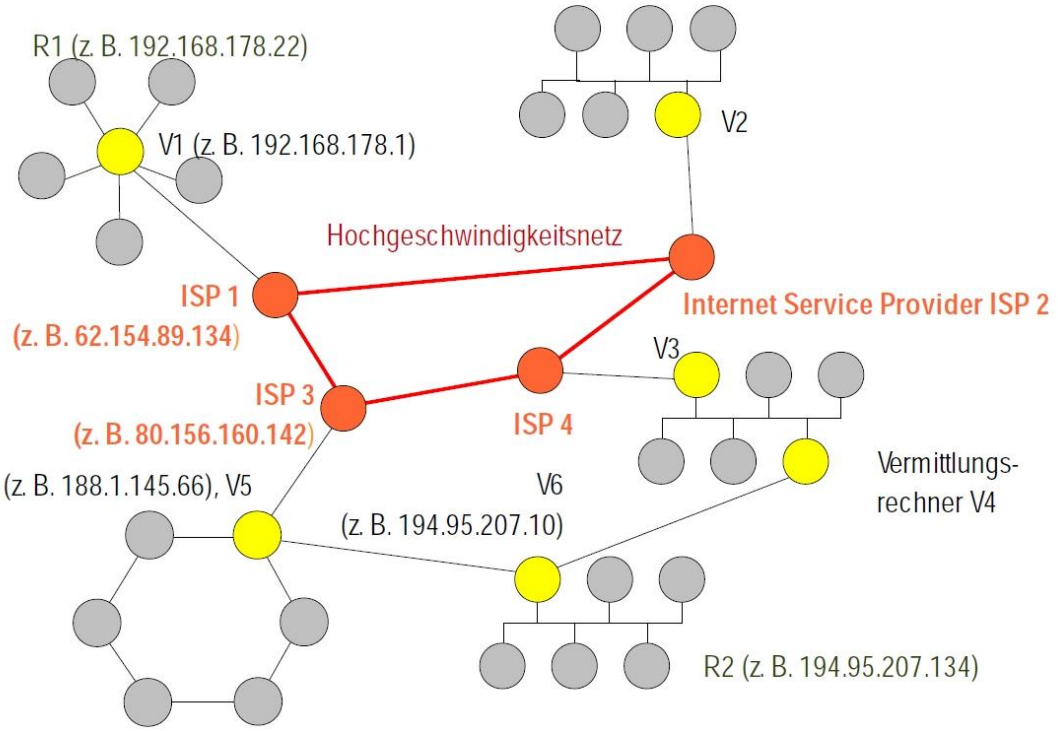
In diesem Abschnitt

Topologie des Internets

Kombinationstopologie

Um die Vorteile der unterschiedlichen Topologien beim Vernetzen von mehreren Millionen Rechnern im Internet optimal auszunutzen, verwendet man eine Kombination der verschiedenen Topologien.
Dabei werden die Teilnetze über Verbindungsrechner (Router) miteinander verbunden. Die Ausfallsicherheit wird durch alternative Datenwege sichergestellt.

Topologie des Internets



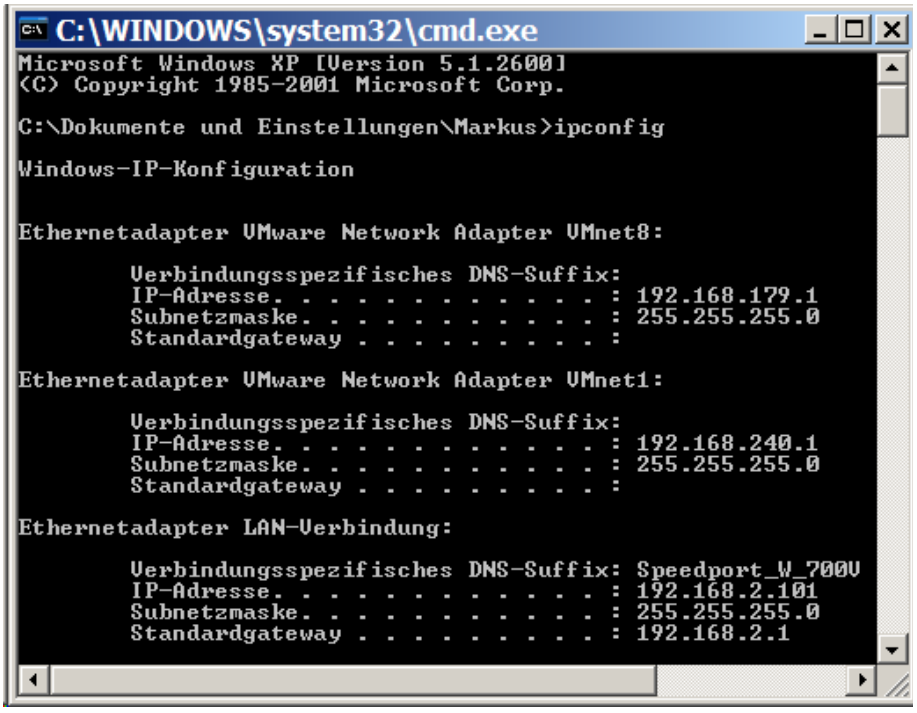
In diesem Abschnitt

Adressierung von Rechnern

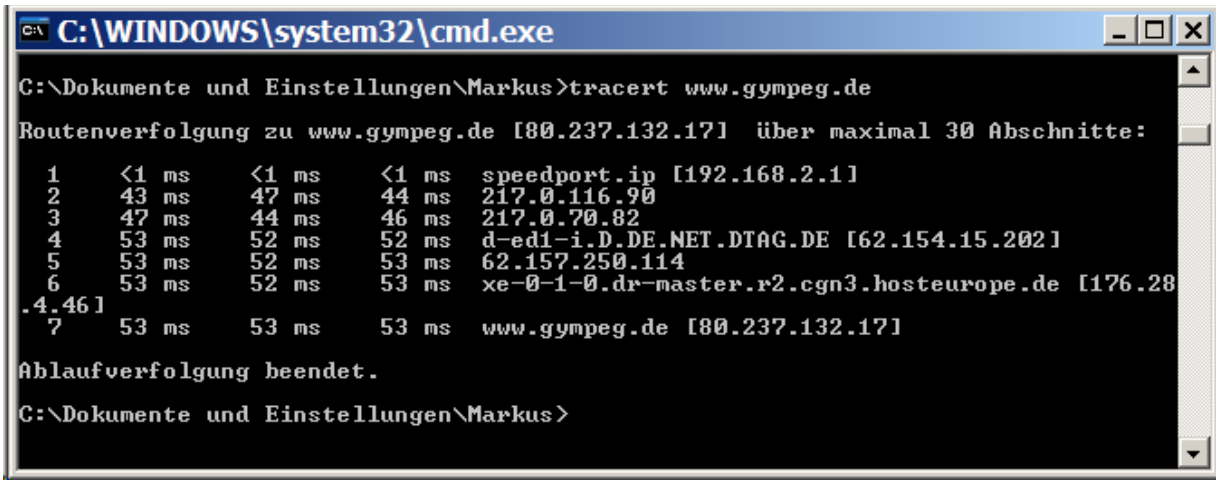
Adressen

In Netzwerken braucht jeder Rechner einen eindeutigen Bezeichner. Im Internet werden dazu die sogenannten IP-Adressen (z.B. 192.168.5.1) verwendet, die teilweise dynamisch festgelegt werden.
Wird eine Anfrage z.B. an die Adresse `www.gympeg.de` gestartet, so erfolgt zunächst eine Namensauflösung durch einen DNS-Rechner (Domain Name Server) der die Datenpakete über verschiedene Stationen an die richtige IP-Adresse (80.237.132.17) sendet.

Übersicht der Netzwerkadapter mit ipconfig abfragen



Datenweg mit tracert verfolgen



Vertiefung und Übung

- Simulation Filius

- Netzwerkzeuge bei Heise
  `http://www.heise.de/netze/Netzwerk-Tools_1386736.html`
- utrace.de
  `http://www.utrace.de/`
- virtuell Tracen
  `http://www.dnstools.ch/visual-traceroute.html`
- Funktion eines DNS
  `http://www.youtube.com/watch?v=2ZUxoi7YNgs&feature=related`

In diesem Abschnitt

Dienste des Internet

Dienst	Aufgabe
E-Mail	Versenden und Empfangen von elektronischer Post
FTP	Mit dem File Transfer Protokoll werden Dateien über das Internet ausgetauscht.
VoIP	Mit Voice over IP wird ein Dienst zur Telefonie bereitgestellt.
WWW	Erlaubt die Kommunikation eines Clientrechners unter Verwendung eines Browsers mit einem Webserver Dieser Dienst (World Wide Web) wird oft fälschlicherweise als „das Internet“ bezeichnet.
...	

- Internetdienste
`http://de.wikipedia.org/wiki/Internetdienste`

Ports

Ports

. . . dienen zur Unterscheidung mehrerer Verbindungen zwischen zwei Rechnern. Ports werden als zusätzliche Nummern realisiert, die zusätzlich zur IP-Adresse angegeben werden.

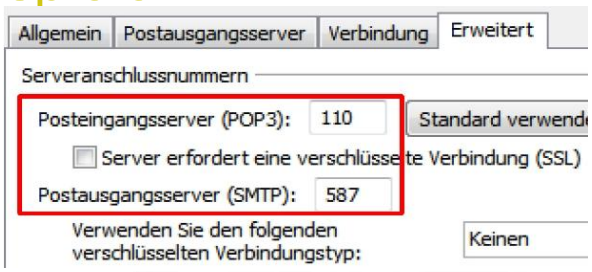
- vgl. Zimmer in Haus

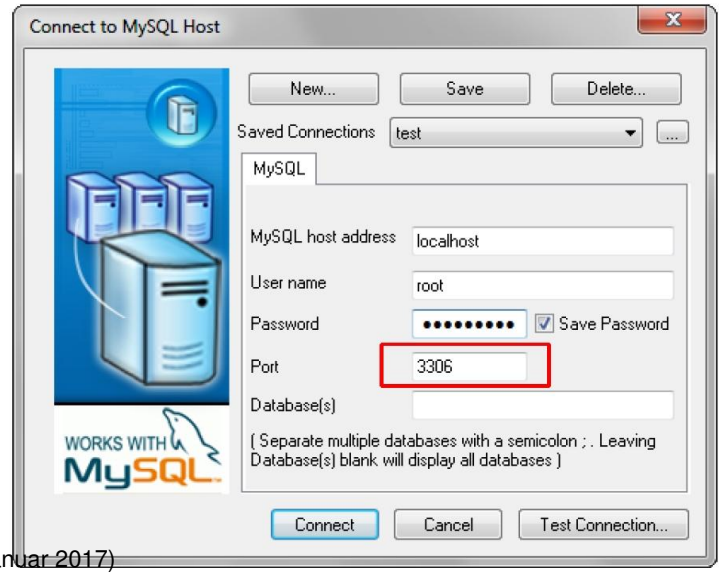
well known port numbers

Port	Dienst
20, 21	Dateitransfer (genereller Datentransfer)
25	E-Mail-Versand (smtp)
80	Webserver
110	E-Mail-Abruf (pop, imap)
443	Webserver mit verschlüsseltem Zugang
3306	Zugriff auf MySQL Datenbank (9. Klasse)

- well-known-ports
`http://de.wikipedia.org/wiki/Liste_der_standardisierten_Ports`

Beispiele





Protokolle

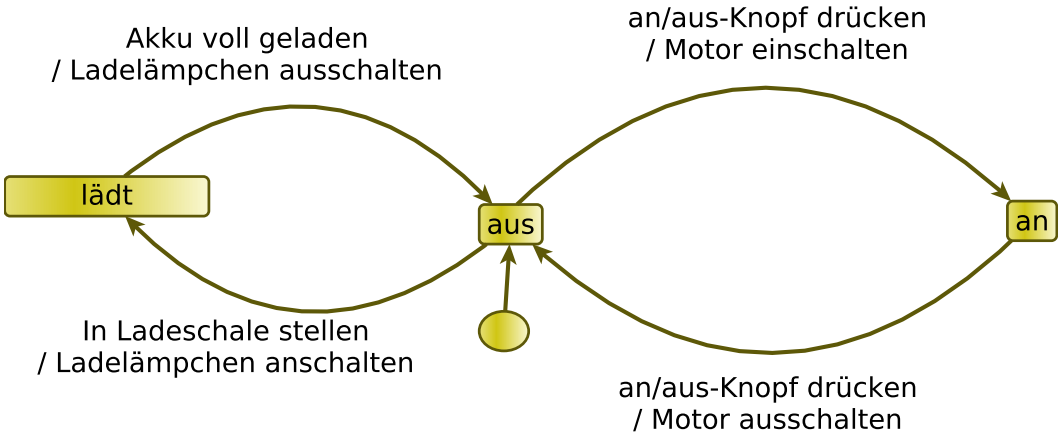
Protokolle

legen die Kommunikation zwischen zwei Parteien fest. Sie definieren

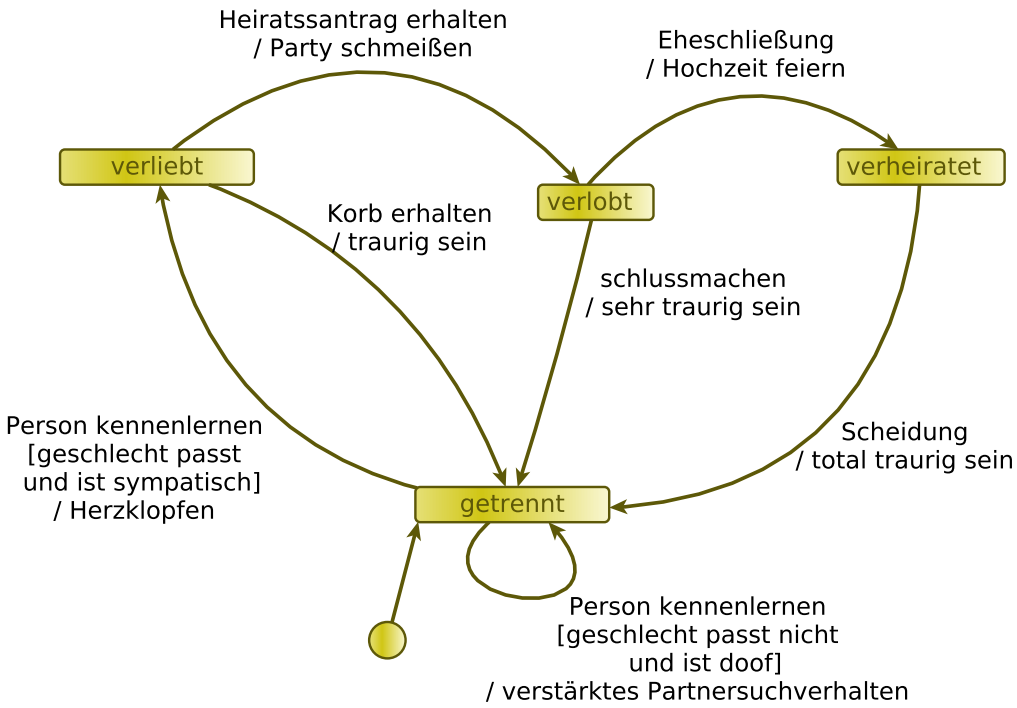
- das Format und
- die Reihenfolge der auszutauschenden Nachrichten sowie die
- Handlungen, die bei der Übertragung oder dem Empfangen einer Nachricht (oder eines anderen Ereignisses) unternommen werden.

Übersicht über das zugrunde liegende Protokoll erhält man über Sequenz- und Zustandsdiagramme.

Zustandsdiagramme



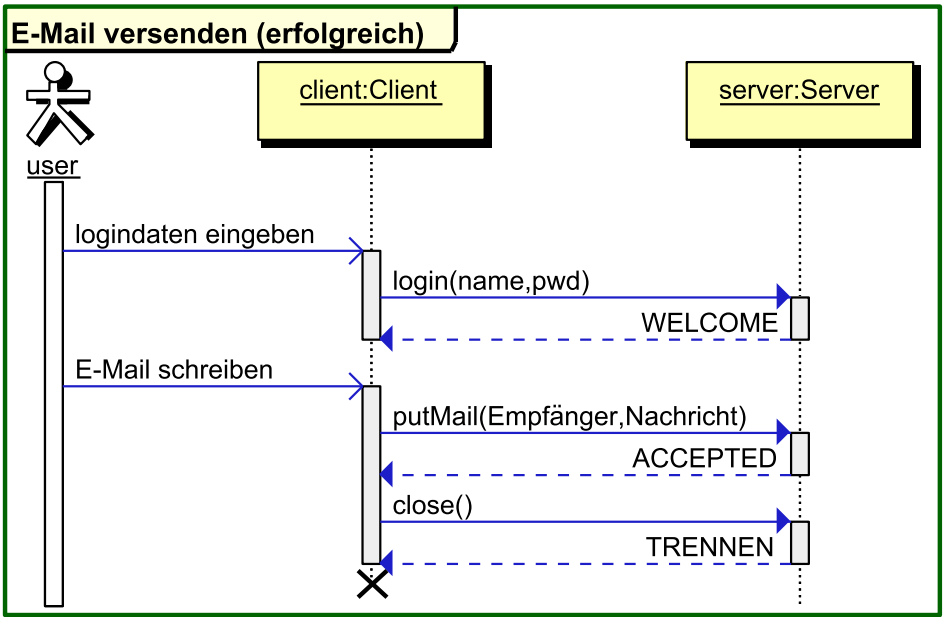
Zustandsdiagramme



Zustandsdiagrame in der UnifiedModelling Language

- Zustände werden durch abgerundete Rechtecke dargestellt.
- Ein Zustandsübergang wird durch ein (Übergangs-)Ereignis hervorgerufen und durch einen Pfeil dargestellt.
- Bei jedem Zustandsübergang kann eine Aktion ausgelöst werden. Der Zielzustand kann von einer Bedingung abhängen
- Jedes Zustandsdiagramm muss genau einen Anfangszustand haben (Symbol: ●→).
- Endzustände (Symbol: —⊙) darf es beliebig viele geben.

Sequenzdiagramm: Kommunikation beim E-Mail senden

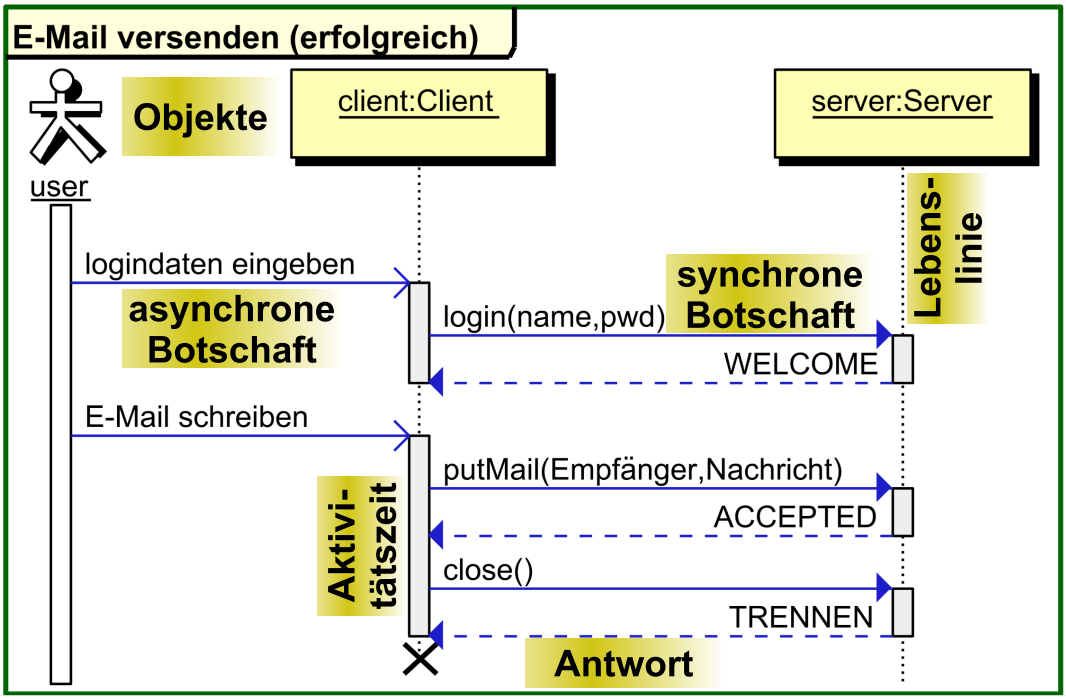


Sequenzdiagramme . . .

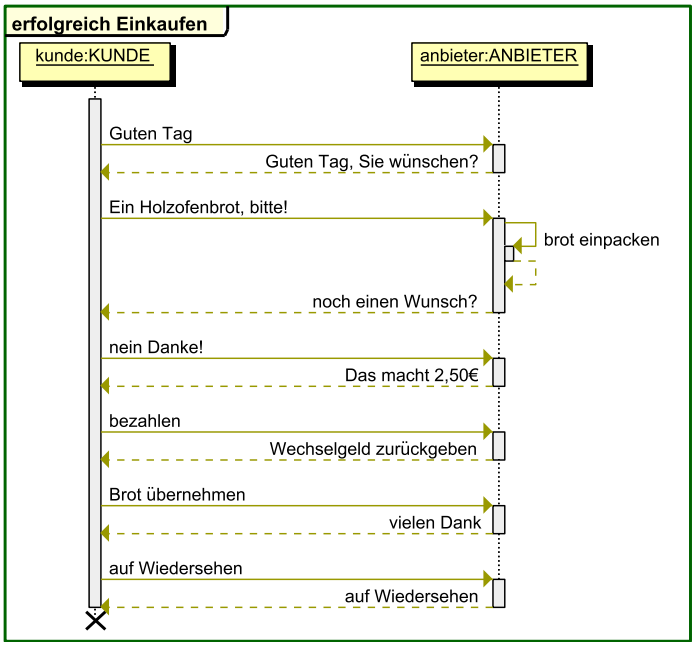
beschreiben die Kommunikation zwischen Objekten in einer bestimmten Szene.

- Die Zeitachse schreitet im Diagramm von oben nach unten fort.
- Gestrichelte senkrechte Linen stellen die Lebenszeit, die Balken die Aktivitätszeit der Objekte dar.
- Durch waagrechte Pfeile werden die ausgetauschen Botschaften dargestellt. „Initialbotschaften“ erhalten durchgehenden, Antworten gestrichelte Linien.
- Man unterscheidet
  synchrone Botschaften Der Sender ist bockiert, bis er eine Antwort erhält (geschlossener Pfeil)
  asynchrone Botschaften Der Sender darf weitere Botschaften senden (offener Pfeil)

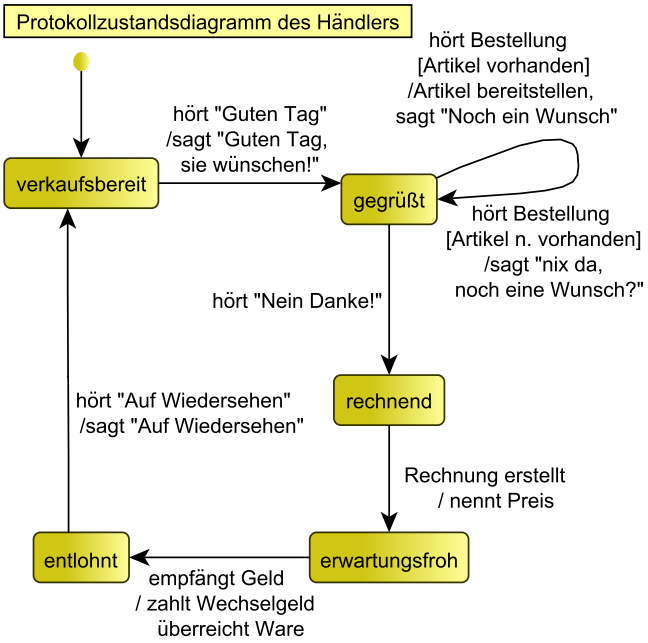
Elemente von Sequenzdiagrammen

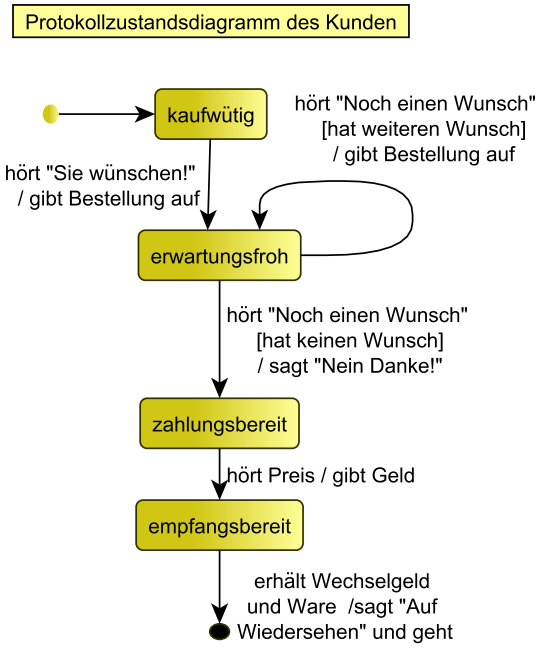


Anwendungsbeispiel: Sequenzdiagramm beim Einkaufen



Anwendungsbeispiel: Protokollzustandsdiagramm beim Einkaufen





Übungen

- Ü 2.1: E-Mail-Protokoll

(a) Erstelle die Sequenzdiagramme für einen gescheiterten E-Mail-Versand und einen erfolgreichen E-Mail-Empfang.

(b) Erstelle die Zustandsübergangsdiagramme, die den Handlungsablauf für den Client und den Server beschreiben. Verwende beim Server nur die Zustände `empfangsbereit`, `ClientRegistriert` und `verarbeitend` und beim Client nur die Zustände `Start` und `eingeloggt`.

Übungen

- Ü 2.2: Kommunikation im Unterricht

Bearbeite folgende Aufgabe nach dem Muster des Brotkaufens bzw. Einkaufens.

(a) Erstelle die Protokollzustandsdiagramme für Lehrer (Server) und Schüler (Clients) für eine typische Unterrichtsstunde die aus folgenden Elementen besteht: `Begrüßung`, `Abfrage`, `Hausaufgabenbesprechung`, `Stoffvermittlung`, `Ausaufgabenerteilung` und `Verabschiedung`.

(b) Erstelle passend zu obigem Protokollzustandsdiagramm eine Sequenzdiagramm zwischen einem Lehrer J und einem Schüler X, der nicht abgefragt wird, der aber die Hausaufgabe vortragen muss.

Übungen

- Ü 2.3: Vorfahrtsregeln

Die Vorfahrt an Kreuzungen wird bekanntlich entweder über Ampeln, Vorfahrtsschilder oder die rechts-vor-links-Regel festgelegt.

(a) Erstelle das Protokollzustandsdiagramm für die Vorfahrtsregeln. Es soll nur zwei Zustände für das Fahrzeug geben.

fahrend	Das Fahrzeug fährt auf einem Teilstück ohne Kreuzung,
anKreuzung	Das Fahrzeug befindet sich an einer Kreuzung und muss die Vorfahrtsregel beachten.

(b) Gib für folgende Situationen jeweils ein Sequenzdiagramm an, das die Kommunikation zweier Fahrzeuge an einer Kreuzung regelt:

(1) Ein Fahrzeug fährt rechts-vor-links-Kreuzung,
(2) Zwei Fahrzeuge fahren über eine rechts-vor-links-Kreuzung,
(3) Zwei Fahrzeuge fahren über eine Kreuzung mit Ampel.

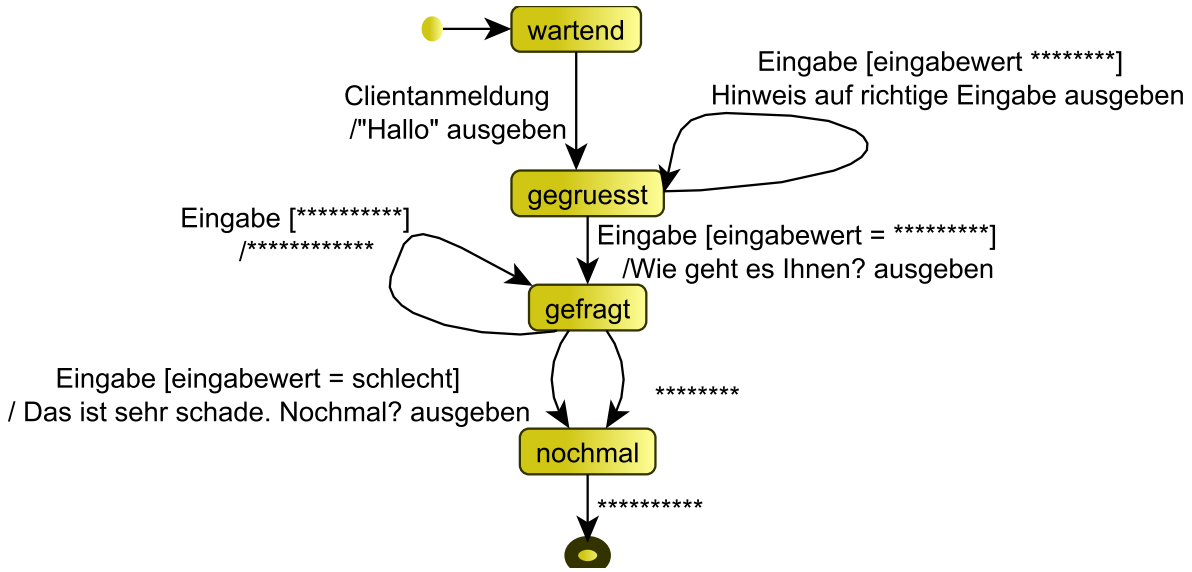
Übungen

- Ü 2.4: AB Chat-Dienst analysieren
`run:Arbeitsmaterial/Uebungen/ChatSitzung/ChatDienstAnalysieren.odt`

In diesem Abschnitt

Umsetzen einer Client-Server-Kommunikation — ZÜD Serververhalten

- Erkunde das Java-Projekt p01_wiegehts



Umsetzen einer Client-Server-Kommunikation — Übung

- Ü 2.5: Client-Server–Kommunikation
`run:Arbeitsmaterial/Uebungen/ClientServerKommunikation/ClientServerKommunikation.odt`
- Ü 2.6: Wetterauskunft
`run:Arbeitsmaterial/Uebungen/Wetterauskunft/Wetterauskunft.odt`

In diesem Abschnitt

Protokollstapel und Schichtenmodell

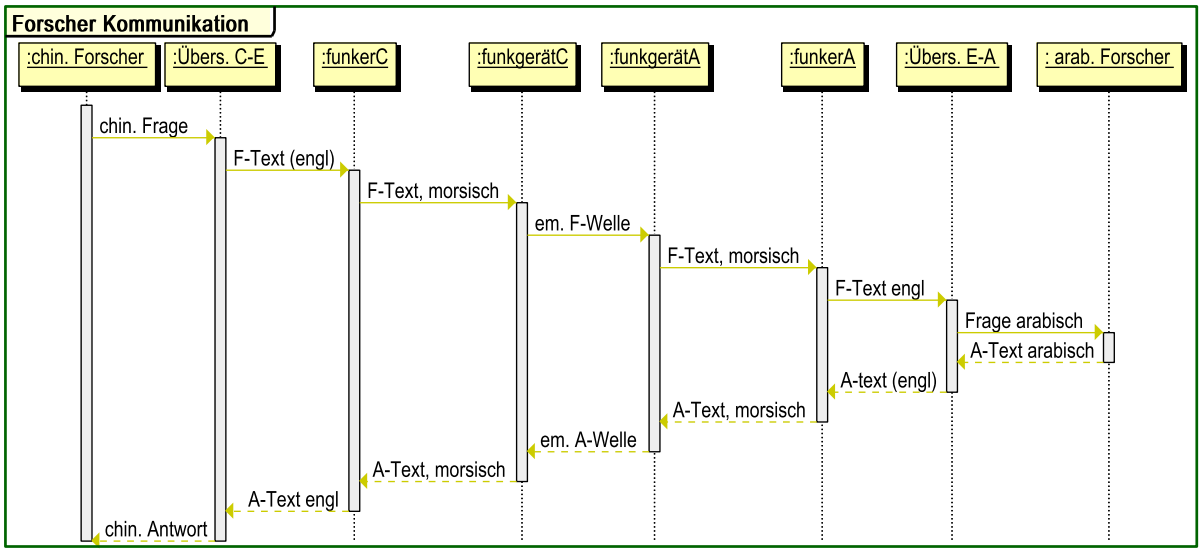
- Schichtenmodell Übersetzer
`http://www.tinohempel.de/info/info/netze/osi.htm`

Das Schichtenmodell. . .

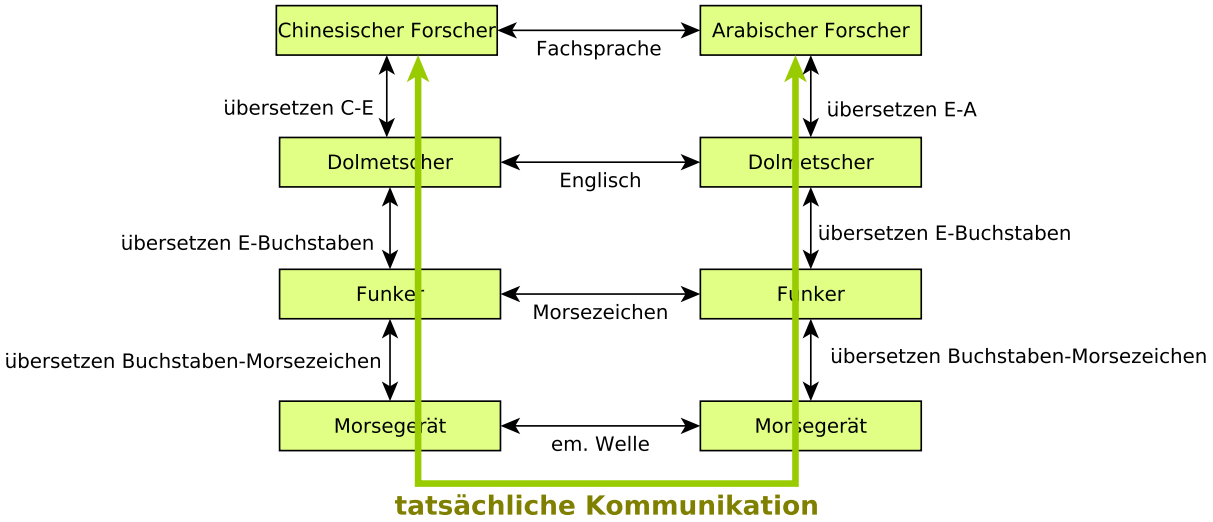
strukturiert aufwendige Arbeitsabläufe, die bei der Kommunikation zwischen Prozessen auftreten.
Jede Schicht hat ihre spezielle Aufgabe und kommuniziert nur mit der direkt darüber- oder darunterliegenden Schicht über ein schichtenspezifisches Protokoll und vorher definierte Schnittstellen.

Chinesisch-Arabischer–Forscherkommunikation

- vgl Abi 2012, IV(3b): Verlinkte Animation als Sequenzdiagramm
`http://www.tinohempel.de/info/info/netze/osi.htm`



Chinesisch-Arabischer–Forscherkommunikation



Der TCP/IP-Protokollstapel. . .

ist ein bewährtes Schichtenmodell zur Beschreibung von Kommunikation innerhalb von Rechnernetzen.

Schicht	Aufgabe	Protokolle
Anwendung	Zusammenarbeit mit Anwendungsprogrammen	http, ftp, smtp, pop, telnet
Transport	Zuverlässige Übertragung der Daten	tcp, udp
Vermittlung	Weiterleitung von Paketen und Routing	IPv4, IPv6
Netzzugang	Technik der Datenübertragung	Ethernet, Token Ring, Token Bus

Zur Veranschaulichung

- Sendung mit der Mouse
  `http://www.youtube.com/watch?v=RT16ryGtbmc&feature=related`
- IP-Routing
  `https://www-rnks.informatik.tu-cottbus.de/content/unrestricted/animations/Prinzip_der_Paketvermittlung.swf`
- Orange-Game Paketvermittlung
  `http://www.youtube.com/watch?v=VRwQd7yHwoA`
- TCP/IP-Interaktionspunkte
  `https://www-rnks.informatik.tu-cottbus.de/content/unrestricted/animations/TCP_IP_Interaktionspunkte.swf`

Das ISO/OSI Referenzmodell. . .

ist ein sehr gut gegliedertes Standardmodell eines Protokollstapels.

	OSI-Schicht	Orientierung
1	Anwendung	
2	Darstellung	anwendungsorientiert
3	Sitzung	
4	Transport	
5	Vermittlung	transportorientiert
6	Sicherung	
7	Bitübertragung	

- ISO/OSI auf YouTube
  `https://www.youtube.com/watch?v=-4-LPCUbC1w`

Übungen

- Ü 2.7: Schichtenmodell

(a) Erstelle das Schichtenmodell (Vorlage: ForscherInEinemRaum.graphml) und das Sequenzdiagramm für obige Forscherkommunikation, wenn sie sich zusammen mit ihren Übersetzern in einem Raum befinden.

(b) S. 67/5

(c) • Abi 2012-IV-3 (S. 17, Die deutsche Bundespräsidentin)
`https://mediathek.mebis.bayern.de/archiv.php?doc=display&id=BY-00012475&token=v3jewyu8pc0jn7c4qdou2p28j0pxuf8u`

In diesem Abschnitt

Problembeschreibung

- Bsp Hausbau

Nebenläufigkeit

Bei großen Projekten (Hausbau, Pizzabacken, . . .) kann man die Arbeit effizienter gestalten durch

nebenläufige Prozesse (Threads) d.h. gleichzeitiges bearbeiten unabhängiger Arbeitsschritte und einer

Synchronisation d.h. Abstimmung von Prozesse die gleiche Ressourcen nutzen oder aufeinander warten müssen.

Übung

- Ü 3.1: Begriff Nebenläufigkeit

Gib für folgende Aufgaben jeweils 12 Arbeitsschritte an und stelle grafisch dar, welche Schritte unmittelbar hintereinander oder nebenläufig stattfinden können. Bei allen drei Situationen muss es einen Zeitpunkt geben, zu dem mindestens 3 Prozesse parallel stattfinden können.

(a) Pizzabacken

(b) Autoinspektion

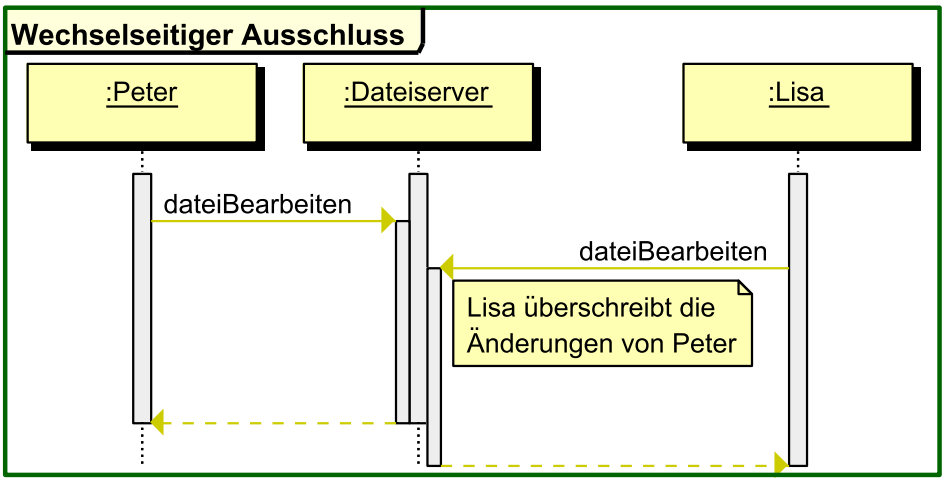
(c) Haus bauen

Synchronisationsprobleme

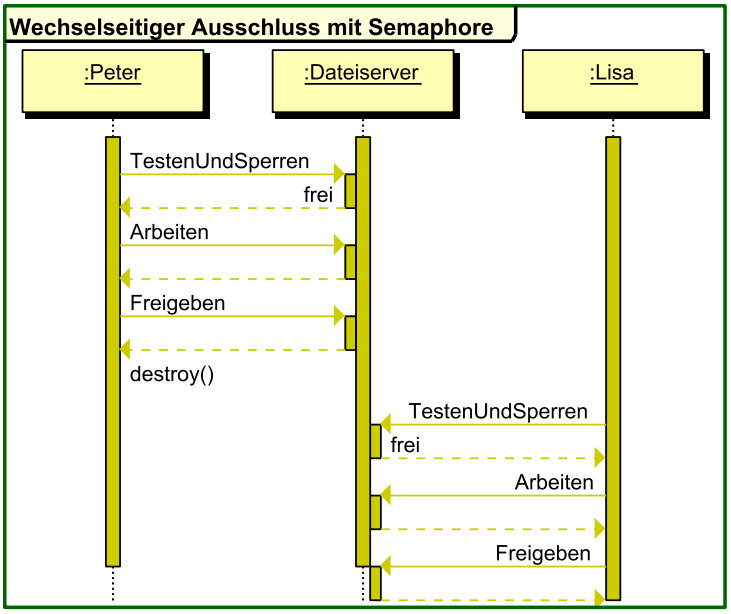
Synchronisationsprobleme

Der Abschnitt eines Projektes, bei dem verschiedene Prozesse auf gemeinsame Ressourcen zugreifen, heißt kritischer Abschitt. Die Ressource kann oft nur im wechselseitigen Ausschluss genutzt werden. Ist sie einmal zugeteilt, kann sie nicht mehr entzogen werden (ununterbrechbare Ressource).

Synchronisationsprobleme — Beispiel Dateibearbeitung



Synchronisation durch Semaphore



- vgl Buch S. 78

Ressourcenzuteilung durch Semaphore

Ressourcenzuteilung durch Semaphore

Beschränkte Ressourcen können sicher mit dem Semaphor-Prinzip verwaltet werden. Dazu muss auf die Ressource in drei Schritten zugriffen werden.

1. anfordern,auch testen genannt, und sperren (in einem Schritt)
2. nutzen
3. freigeben

Bemerkung: Die Schritte 1 und 3 können nicht durch eine Hochsprache erfolgen, sondern müssen durch Maschinenbefehle umgesetzt werden.

Vereinfachung durch Monitore

Ein Monitor

ist eine Gruppe von Methoden, die zu einem bestimmten Zeitpunkt immer nur von einem einzigen Objekt ausgeführt werden darf. Die anderen Objekte müssen warten.
Intern wird das Monitorkonzept durch Semaphore umgesetzt.

Z.B. können die beiden Methoden `lesen` und `schreiben` zu einem Monitor zusammengefasst werden, so dass immer nur ein Objekt eine dieser Aktionen ausführen kann.

Übung

- Ü 3.2: Vertiefung Semaphore

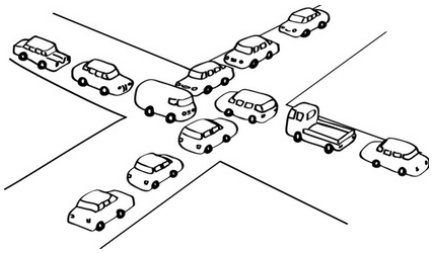
Wir nehmen an, das Anfordern und Sperren eines Dateiservers würde doch in zwei aufeinander folgenden Schritten stattfinden. Zeige durch eine Sequenzdiagramm wie oben auf der Folie `Synchronisation durch Semaphore`, dass es dadurch zu Datenverlust kommen kann.

- Ü 3.3: Buch, S. 82/1

Verklemmungen

Verklemmungen

Beim gemeinsamen Nutzen von Ressourcen kann es sein, dass zwei Prozesse gegenseitig aufeinander warten und so das Projekt nicht weiter läuft. Man sagt, es tritt eine Verklemmung auf.



Das Philosophenproblem

- Philosophenproblem durchspielen
- Philosophenproblem
  `http://users.erols.com/ziring/diningAppletDemo.html`

Übung

- Ü 3.4: Nebenläufigkeit

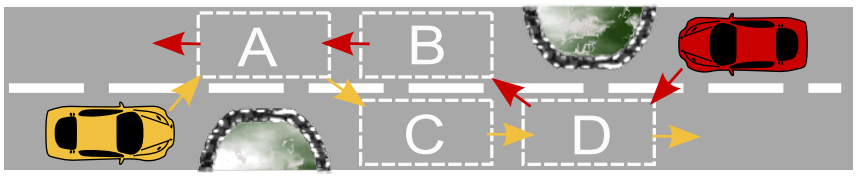
(a) Erläutere das Philophophenproblem (googeln, App zeigen). Betrachte

- SDPhilosophenProblem
`run: Arbeitsmaterial/Uebungen/Nebenlaeufigkeit/SDPhilosophenProblem.pdf`
Gib jeweils an, in welchem Zustand sich die Philosophen bei der gestrichelten Linie (guckst du) des Sequenzdiagramms befindet.

(b) Betrachte

- NebenlaufImAlltag
`run:Arbeitsmaterial/Uebungen/Nebenlaeufigkeit/NebenlaufImAlltag.pdf`
Erläutere mit Hilfe einer Tabelle bei jeder der dargestellten Sachsituationen, die Begriffe Nebenläufigkeit, kritischer Abschnitt, wechselseitiger Ausschluss, ununterbrechbare Ressource, Deadlock, anfordern, nutzen, freigeben.

- Ü 3.5: Verkehrsberuhigung (Klett S.55/6)



(a) Beschreibe genau, welche Prozesse stattfinden können und welche Ressourcen gemeinsam bzw. nicht gemeinsam benutzt werden können.

(b) Synchronisiere die beiden Prozesse, indem du angibst, welche Positionswechsel (z.B. $\rightarrow A$ oder $A \rightarrow C$) nebenläufig stattfinden können und welche nicht. Fertige dazu eine Tabelle an.

(c) Welche Verklemmung kann auftreten, wenn mehrere Autos

Übung Erzeuger-Verbraucher-Problem

- Ü 3.6: EV-Problem
http://www.doc.ic.ac.uk/~jnm/concurrency/classes/ChannelDemo/ChannelDemo.html

(a) Erforsche obigen Animation (oder andere) zum Erzeuger-Verbraucher-Problem und beschreibe, was man unter diesem Problem versteht.

(b) Nenne 10 möglichst unterschiedliche Situationen des Alltags, die auf dem Erzeuger-Verbraucher-Prinzip basieren. Benenne jeweils den Erzeuger, den Verbraucher und den Puffer.

Bedingungen für Deadlocks

Bedingungen für Deadlocks

Folgende vier Bedingungen müssen erfüllt sein, damit Verklemmungen entsthehen:

1. Benötigte Ressourcen können nur im wechselseitigen Ausschluss benutzt werden,
2. ein Prozess kann weiter Ressourcen anfordern, auch wenn er bereits andere angefordert hat,
3. belegte Ressourcen sind ununterbrechbar zugeteilt,
4. mehrere Prozesse warten auf Ressourcen, die von anderen schon belegt sind.

Übung

- Ü 3.7: Deadlocks auflösen

Weiter oben wurden vier Bedingungen formuliert, damit Verklemmungen entstehen. Gib für das Philosophenproblem und zwei weitere Verklemmungssituationen deiner Wahl an, wodurch jede der vier Bedingungen erfüllt ist, und zeige dass die Verklemmung aufgelöst werden kann, wenn man diese Bedingung aufgibt.

- Vorlage DeadlockAufloesen.odt
`run:Arbeitsmaterial/Uebungen/Nebenlaeufigkeit/DeadlocksAufloesen.odt`

In diesem Abschnitt

Konzept

Bei Programmfehlern unterscheidet man

Syntaxfehler z.B. fehlende Klammern, Tippfehler etc. Sie werden vom Compiler beim Übersetzen des Quellcodes erkannt.

Laufzeitfehler z.B. Division durch Null, Typenunverträglichkeiten, Verletzung von Arraygrenzen oder bei Zugriff auf noch nicht erzeugte Objekte.

`Laufzeitfehler`
sind schwerwiegende Fehler, die ohne Behandlung zu Abstürzen oder unerwünschten Programmzuständen führen, sie treten immer erst bei der Ausführung des Programms auf.

Konzept

`Exceptions`
Bei Laufzeitfehlern erzeugt `Java` Objekte der Klasse `Exception`, die Informationen über den Laufzeitfehler enthalten. Dabei ist die Klasse `Exception` eine Oberklasse von speziellen `Exception` wie z.B. der `NumberFormatException` oder der `ArrayIndexOutOfBoundsException`.

Übung

- Ü 3.8: Laufzeitfehler

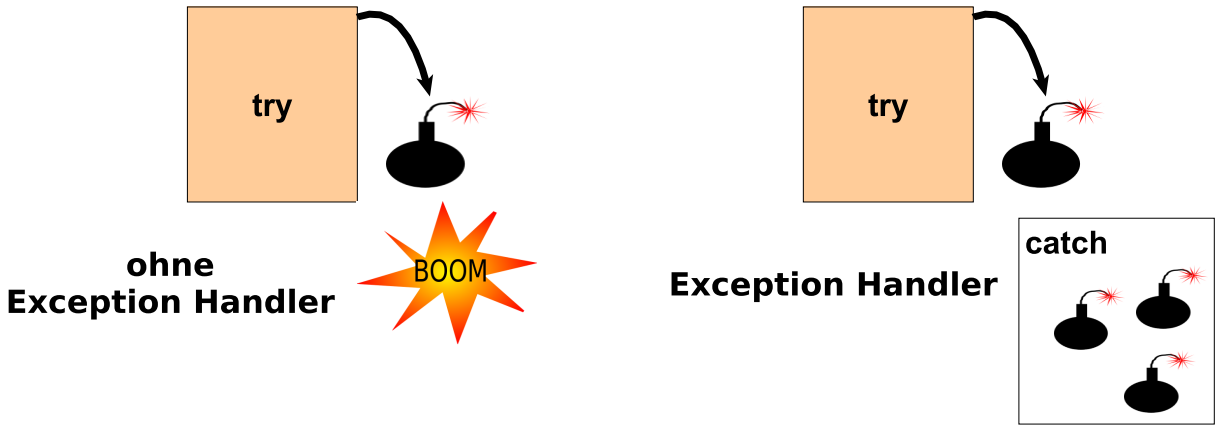
Erzeuge eine Java-Klasse `Laufzeitfehler` und implementiere dort Methoden, die folgende Laufzeitfehler erzeugen: Division durch Null, Unmögliche Typumwandlung von String zu Integer, Zugriff auf eine Arrayelement außerhalb der Arraygrenzen, Zugriff auf Objekte, die nicht existieren.
Welche der Laufzeitfehler können mit den herkömmlichen Programmiermethoden leicht abgefangen werden?
Implementiere diese Möglichkeiten.

Java-Konzept: Excecption-Handling

`Exception-Handling`

In Java können Laufzeitfehler durch einen `try-catch-[finally]-Block` abgefangen werden.

ExeptionSynchronisationsprobleme



try-catch–Syntax

```
1  try{
2      // Hier können Exceptions auftreten
3  }
4  catch( Exception e){
5      // Hier werden sie abgearbeitet
6      // Catch ist optional, sofern ein
           finally-Block folgt
7      // Es können mehrere catch-Blöcke
           hintereinander stehen, um
8      // unterschiedliche Exceptions
           abzufangen
9  }
10 finally{
11     // Wird immer durchlaufen, ist
           allerdings optional
12 }
```

Beispiel: Zahlenformat abfangen

```
1  public class ZahlenformatAbfangen{
2
3      public int stringToIntegerBOOM(String s){
4          return Integer.parseInt(s);
5      }
6
7      public int stringToInteger(String s){
8         int z;
9         try{
10           z = Integer.parseInt( s );
11           return z;
12        }
13        catch ( NumberFormatException e ){
14           System.out.println( s + " kann man nicht in
                 eine Zahl konvertieren!");
15         }
16         return -1;
17     }
18 }
```


Beispiel: Arraygrenzen abfangen

```
1  import java.io.*;
2  public class ArraygrenzenAbfangen{
3     private Object[] liste = {"a", 1, 2.3, "xyz"};
4
5     public Object gibElementNrBOOM (int nr){
6         return liste[nr];
7     }
8
9     public Object gibElementNr (int nr){
10       Object o = null;
11       try{
12          o = liste[nr];
13       }
14       catch(ArrayIndexOutOfBoundsException e){
15          System.out.println("Exception thrown  :" + e);
16       }
17       finally{
18       System.out.println("Out of the block");
19       return o;
20    }
21    }
22 }
```

Übung

- Ü 3.9: Einfache Exceptions

(a) Implementiere die oben aufgeführten Klassen zum Testen von Exceptions.

(b) Erweitere die Klasse `Laufzeitfehler` um vier Methoden, die die vier Laufzeitfehler mit einer `try-catch`-Anweisung abfangen. Gib im `catch`-Teil auch jeweils die `exception` mit aus (vergleiche `class ArraygrenzenAbfangen`). Verwende auch jedes Mal sinnvolle `finally`-Blöcke.

In diesem Abschnitt

Begriffe Threads und Prozesse

Definition

Moderne Prozessoren können mehrere Programmteile (quasi) parallel (gleichzeitig) bearbeiten. Man nennt einen solche Programmteile Prozesse.
Wir unterscheiden

- schwergewichtige Prozesse, das sind komplette Programme,
- leichtgewichtige Prozesse, sog. Threads, das sind Programmteile,
- Nebenläufige Prozesse, das sind parallele Prozesse, die auf gemeinsame Ressourcen zugreifen und deshalb eine Synchronisation erfordern.

Warum Threads?

Threads nutzen die vorhandenen Ressourcen moderner Betriebssysteme und Hardware effizienter aus. Beispielsweise durch Weiterführung der Berechnungsopperationen, während andere Threads auf Ein-/Ausgaben warten.

Beispiel

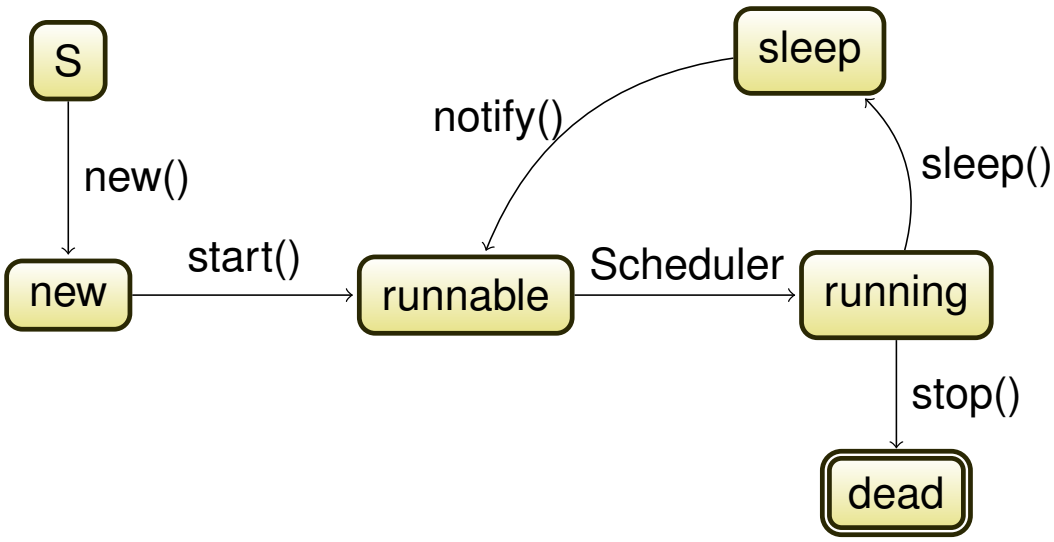
- Demo-Thread
  `http://www.doc.ic.ac.uk/~jnm/concurrency/classes/ThreadDemo/ThreadDemo.html`
- Ü 3.10: Erweitere obiges Beispiel um einen dritten Thread gleicher Art.

Allgemeine Form eines eigenen Threads

```
1  class MeinThread extends Thread {
2    // Nötige Attribute, Konstruktor, etc.
3    public void run ()
4    // Hier beginnt der Thread seine Arbeit
5      { // Arbeitsauftrag des Threads
6    }
7  }
8
9  //Erzeugen und aktivieren des Threads
10 Thread t = new MeinThread ();
11   // Das Objekt wird angelegt
12 t.start (); // Der Thread wird in die
     Verwaltung aufgenommen; sobald er an der
     Reihe ist, wird mit Ausführung der
     Methode run begonnen
```

Zustandsdiagramm der Klasse Thread

Der Zustand eines Threads kann durch verschiedene in der Klasse `Thread` implementierte Methoden geändert werden:



Bemerkungen

- Die Schnittstelle Runnable ist ein besonderer Zustand. Sie definiert als einzige Methode `run()`, die zu Beginn der Thread-Verarbeitung aufgerufen wird. Das heißt nicht, dass der Thread gestartet wird.
- Ob ein Thread tatsächlich läuft oder nicht entscheidet der JAVA-interne Scheduler.
  Daher sollte die Methode `run()` niemals direkt aufgerufen werden.
- Jeder Thread wird ausschließlich durch die Methode `start()` dem Laufzeitsystem als arbeitswillig gemeldet werden.

Methoden der Klasse Thread

Methode	Wirkung
run()	Diese Methode wird ausgeführt, wenn der Thread läuft, darf aber nicht aufgerufen werden, weil das Kommando über die Thread der Scheduler hat.
start()	aktiviert den Thread, so dass er irgendwann durch den Scheduler gestartet werden kann.
stop()	beendet und löscht den Thread.
suspend()	unterbricht den Thread und ...
resume()	... startet ihn an der gleichen Stelle wieder.
sleep(int x)	Der Thread pausiert für mindestens *x* ms und wird danach vom Scheduler wieder gestartet.

Beispiel: Zähl-Thread

```
1  public class ZaehlThread extends Thread
2  {
3      public void run(){
4          int i = 0;
5          while (true){
6              System.out.println(i++);
7              try{
8                  Thread.sleep(100);
9              }
10             catch (InterruptedException e){}
11         }
12     }
13 }
```

Beispiel eines einfachen Threads

In einer zweiten Klasse wird dann ein neuer Thread erzeugt und gestartet.

```
1  public class Demo{
2      ZaehlThread t = new ZaehlThread();
3
4      public void threadStarten(){
5          t.start();
6          try {
7              Thread.sleep(10000);
8          } catch (InterruptedException e) {}
9          t.stop();
10     }
11 }
```

Der eigene Thread t wird gestartet, läuft 10000 ms lang und wird anschließend gestoppt.

Übungen

- Ü 3.11: Threads erforschen

Erforsche die Java-Programme

- UrlaubsThread
  `run: Arbeitsmaterial/Java/Threads/UrlaubsThread/package.bluej`
- PlusMinusThread
  `run: Arbeitsmaterial/Java/Threads/PlusMinusThread/package.bluej`

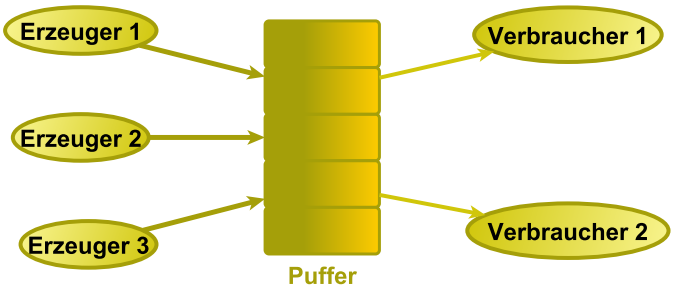
und erkläre genau, welche Prozesse hier nebenläufig stattfinden und wie man erkennt, dass wir keinen direkten Einfluss haben, welcher Thread gerade läuft.

Wiederholung: Erzeuger-Verbraucher–Problem

- Ü 3.12: EV-Problem
`http://www.doc.ic.ac.uk/~jnm/concurrency/classes/ChannelDemo/ChannelDemo.html`

Wiederholung: Erzeuger-Verbraucher–Problem

Mehrere Erzeuger können unabhängig voneinander Produkte im Pufferspeicher ablegen, die bei Bedarf vom Verbraucher abgeholt werden.



Kritische Abschnitte beim Erzeuger-Verbraucher–Problem

Kritische Abschnitte beim Erzeuger-Verbraucher–Problem. . .

- das Einfügen von Produkten in den Speicher, weil er überlaufen kann,
- das Entnehmen von Produkten aus dem Speicher, weil er leer sein kann.

Um sicherzustellen dass keine Verklemmungen auftreten, fasst man die kritischen Attribute und Methoden zu einer Gruppe zusammen, so dass nur jeweils ein Prozess auf dieser Gruppe arbeiten kann.

Monitorkonzept

Monitore...

legen Programmabschnitte fest, die immer von nur einem Prozess gleichzeitig ausgeführt werden dürfen. In objektorientierten Sprachen sind diese Programmabschnitte oft an das Ausführen von Methoden gekoppelt. So dürfen im E-V-Problem die Methoden `einfügen()` und `entnehmen()` nicht gleichzeitig ausgeführt werden. Deshalb fasst man den Pufferspeicher und diese beiden Methoden in einem Monitor zusammen. In Java erfolgt das mit dem Schlüsselwort **`synchronized`** bei den Methoden.

Übung

- Ü 3.13: Koch und Kellner
`run:Arbeitsmaterial/Uebungen/KochUndKellner/KochUndKellner.pdf`