http://www.therealgang.de/

Titel :	Grundlagen des Internet 2
Author :	David Biermann
Kategorie :	SONSTIGE-SKRIPTE

Akademie
der
Saarwirtschaft

ITG II / HTML / XML

Dozent:
Dirk Bremer

Script created by:
David Biermann

Inhaltsverzeichnis

HTML

http://www.freenet.de / index.html (Name(Ordner) der gewünschten Datei)

⇩ ⇩

Protokoll Name des Zielrechners (Webservers)

⇩ Umwandlung durch DNS-Server (Domain-Name-Service)

IP-Adresse
(z.B. 196.80.100.7)

Anforderung des Clients im http-Format

http Protokoll spezifiziert Regeln wie eine Anforderung an einem Server bzw. die Antwort des Servers „auszusehen“ hat (z.B. Aufbau der Datenpakete und deren Inhalt).

1	2	3	4	5

Gründe für die Popularität von HTML:

- Einfach (im Sinne der Erlernbarkeit)
- Textbasiert (HTML-Dokumente können unabhängig von einem Betriebssystem und einer Anwendersoftware gelesen und geschrieben werden).

Beispiel zur Formularverarbeitung:

1. Anwender gibt Daten in die Felder ein (in Browser)
2. Abschicken der Daten (im Browser)
3. Webserver empfängt Daten
4. Webserver startet Programm, das die Daten liest und weiterverarbeitet, z.B. in eine Datenbank schreibt und eine neue HTML-Datei erzeugt und diese zum Browser (Client) zurückschickt.

Im Rahmen der Formularverarbeitung werden lediglich die folgenden Schritte von HTML unterstützt:

1. Design der Eingabemarken des Formulars
2. Innerhalb des Dokuments die Dateiangaben, mit der die von Anwender eingegebenen Daten verarbeitet werden und eindeutige für die Formularobjekte vergeben.

Beachte: Jedem HTML Objekt (z.B. Formular, Tabellenfeld, Grafik ...) kann ein eindeutiger Name vergeben werden, um dieses Objekt später zu identifizieren.

Was ist ein CGI-Programm?

- ist eine ausführbare Datei, die
- in der Regel mit den Sprachen C oder Perl geschrieben wurde

⇩ wegen Schnelligkeit

⇩ wegen Verarbeitung von Zeichenketten

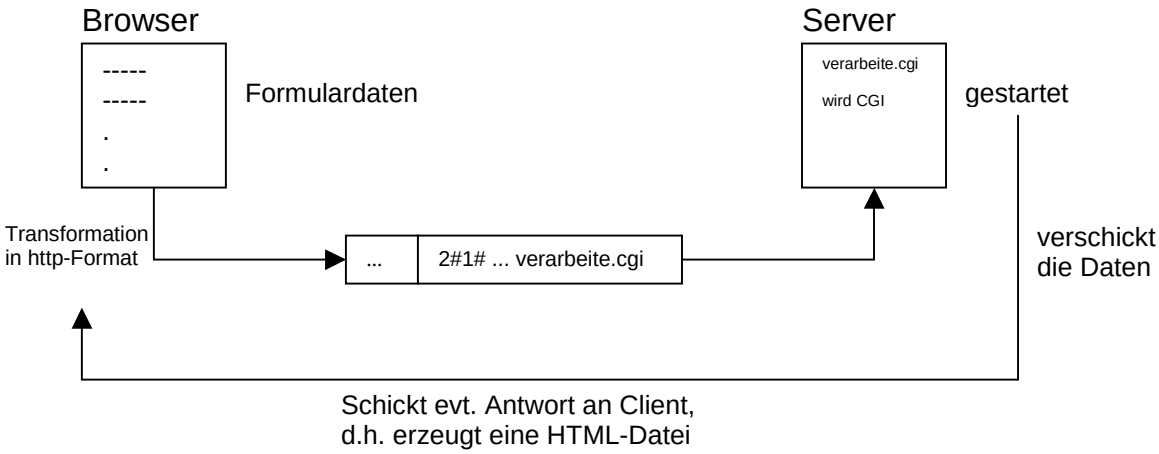
Ein Webserver besitzt eine CGI-Schnittstelle

(Empfängt z.B. Formulardaten im http-Format und startet ein entsprechendes CGI-Programm)
CGI-Programme befinden sich im Verzeichnis CGI-BIN.

Binaries: Maschinenprogramme direkt ausführbar. CGI ist eine Schnittstelle des Webservers zur Außenwelt. Programme um Formulardaten weiterzuverarbeiten, werden nicht automatisch vom Webserver bereitgestellt. Ein solches Programm vom Entwickler gemäß den Anforderungen des Formulars und dessen Verarbeitung individuell geschrieben werden. Diese wird über die CGI-Schnittstelle des Webservers zur Verfügung gestellt.

Zwischen dem HTML-Dokument, dass das Formular und dessen Layout beinhaltet und dem CGI-Programm, besteht über obigen Punkt 2 eine 1:1 Beziehung.

Werden Formulardaten lediglich auf Clientseite (Browserseite) verarbeitet, dann kann man hierzu die Sprache JavaScript (siehe später) benutzen.



Schwächen / Grenzen der Sprache HTML

1. Mangelnde Dynamik
   Mit reinem HTML können Geschäftsprozesse nur teilweise unterstützt werden. Interaktionsmöglichkeiten mit Benutzern sind nur eingeschränkt möglich.
   Bsp: Formularverarbeitung
   Mit HTML können lediglich Eingabemasken für Formulare bereit gestellt werden (ferner können für Daten Namen für die Weiterverarbeitung angegeben werden). Die Weiterverarbeitung selbst kann mit HTML nicht erfolgen, da HTML keine Programmiersprache ist und deshalb keine Verwendung von Variablen, Bedingungen, mathematischen Operationen, Schleifen, etc. direkt unterstützt. HTML dient lediglich der Darstellung von Informationen, besitzt

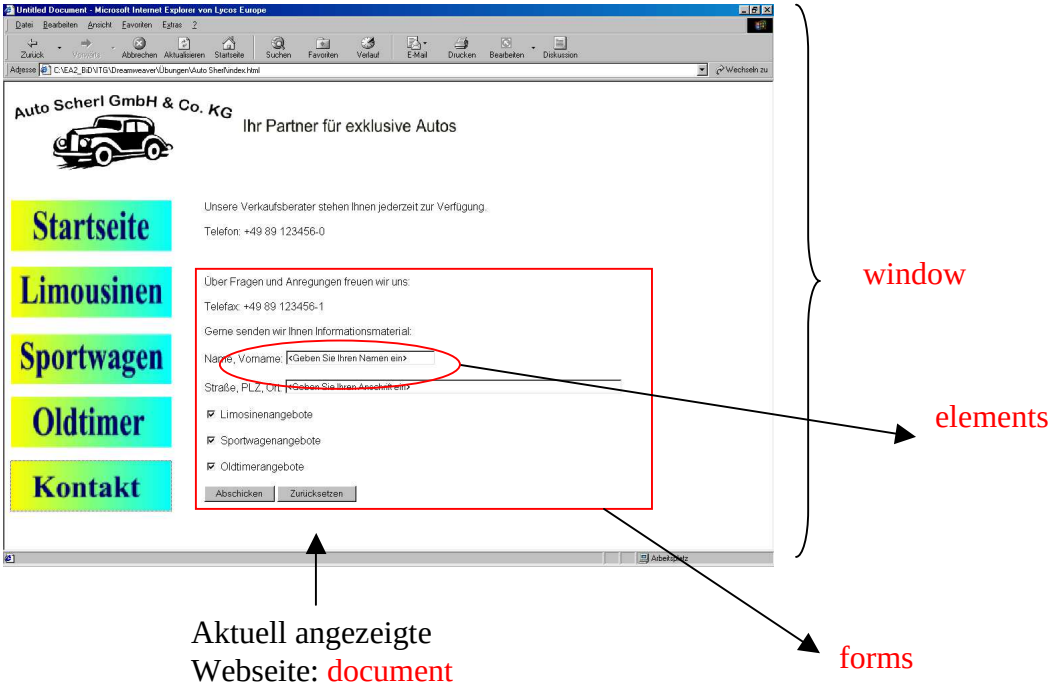
jedoch Schnittstellen um mit bestimmten Programmiersprachen erstellte Anwendungen zu integrieren (z.B. JavaScript)

2. HTML-Dokumente besitzen keine bzw. nur eine unzureichende inhaltliche Struktur
   ⇒ Die Suche nach Inhalten, die einem bestimmten Kriterium entsprechen, ist fast unmöglich
   Bsp: Ein Benutzer sucht Webseiten, die einen Audi A3 beinhalten, der weniger als 12.500 € kostet.
   Lösungsansatz:
   Um ein Dokument inhaltlich zu strukturieren, könnte es in einer Datenbank gespeichert werden.
   ↳*Problem*: Verlust der Plattformunabhängigkeit aufgrund unterschiedlicher Datenbankformate und –abfragesprachen.
   ↳*Ausweg*: Verwende XML um Inhalt plattformunabhängig zu strukturieren und mit entsprechenden Werkzeugen Inhalte nach bestimmten Kriterien zu filtern, zu selektieren und darzustellen.

Java Script

Die Objekthierarchie in JavaScript

1	**window**	(Anzeigefenster)
1.1	**frames**	(Frame-Fenster)
1.2	**document**	(Inhalt eines Fensters)
1.2.1	**all**	(HTML-Elemente der Datei)
1.2.1.1	**style**	(CSS-Attribute eines Elements)
1.2.2	**anchors**	(Verweisanker in der Datei)
1.2.3	**applets**	(Java-Applets in der Datei)
1.2.4	**forms**	(Formulare in der Datei)
1.2.4.1	**elements**	(Formularelemente in der Datei)
1.2.4.1.1	**options**	(Formular-Auswahllisten in der Datei)
1.2.5	**images**	(Grafiken in der Datei)
1.2.6	**layers**	(Layer in der Datei)
1.2.7	**links**	(Verweise in der Datei)
1.3	**event**	(Anwenderereignisse im Anzeigefenster)
1.4	**history**	(besuchte Seiten)
1.5	**location**	(URL-Adressen)
2	**Array**	(Ketten von Variablen)
3	**Boolean**	(Ja/nein-Werte)
4	**Date**	(Datum und Uhrzeit)
5	**Function**	(JavaScript-Funktionen)
6	**Math**	(Berechnungen)
7	**navigator**	(Browser-Informationen)
7.1	**mimeTypes**	(MimeType-Informationen)
7.2	**plugins**	(vorhandene Plugins)
8	**Number**	(numerische Werte)
9	**RegExp**	(reguläre Ausdrücke)
10	**Screen**	(Bildschirm-Informationen)
11	**string**	(Zeichenketten)



Arbeitsblatt 7

```
<script language="JavaScript">
      eingabe = prompt("Bitte geben Sie Ihren Namen ein!","");
      document.write(eingabe);
</script>
```

Die Methode „prompt(“-“, “-“) liest über eine Eingabemaske die Eingabe des Benutzers, speichert die Variablen „eingabe“. Der Inhalt von „eingabe“ wird über die Methode „write()“ des Objekts „document“ auf der aktuellen Seite ausgegeben.

Auslagerung einer JavaScript-Anweisung in eine js-Datei:

home_4.html

```
<script language="JavaScript" src="js_bsp03.js">
</script>
```

js_bsp03.js

```
name = prompt("Bitte geben Sie Ihren Namen ein!","");
document.write(name);
```

Funktionen in JavaScript

- Platzierung im Head der HTML-Datei
- Aufbau:

  ```
  function Name_der_Funktion(Parameter1,Parameter2,...)
  {   Anweisungen   }
  ```
- Aufruf (i.d.R. im Body der HTML-Datei) durch Angabe des Namens der Funktion (siehe Arbeitsblatt 8.2), innerhalb einer JavaScript-Umgebung oder über einen

Event-Handler (siehe Arbeitsblatt 8.3)

- **onAbort** (bei Abbruch)
- **onBlur** (beim Verlassen)
- **onChange** (bei erfolgter Änderung)
- **onClick** (beim Anklicken)
- **onDblClick** (bei doppeltem Anklicken)
- **onError** (im Fehlerfall)
- **onFocus** (beim Aktivieren)
- **onKeydown** (bei gedrückter Taste)
- **onKeypress** (bei erfolgtem Tastendruck)
- **onKeyup** (bei losgelassener Taste)
- **onLoad** (beim Laden einer Datei)
- **onMousedown** (bei gedrückter Maustaste)
- **onMousemove** (bei weiterbewegter Maus)
- **onMouseout** (beim Verlassen des Elements mit der Maus)
- **onMouseover** (beim Überfahren des Elements mit der Maus)
- **onMouseUp** (bei losgelassener Maustaste)
- **onReset** (beim Zurücksetzen des Formulars)
- **onSelect** (beim Selektieren von Text)
- **onSubmit** (beim Absenden des Formulars)
- **onUnload** (beim Verlassen der Datei)
- **javascript:** (bei Verweisen)

Passwortschutz (Arbeitsblatt 9.1)

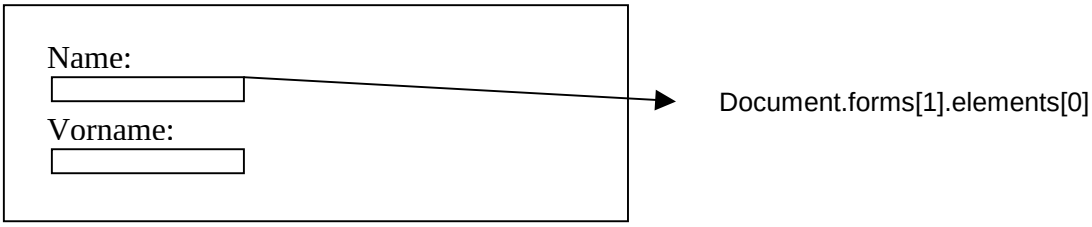
1. Anwender gibt das Kennwort über eine JavaScript-Eingabebox oder ein Formular ein.
2. Lese den eingegebenen Text, vergleiche diesen mit dem Passwort
3. a) Falls Übereinstimmung: Lade Seite
   b) Falls keine Übereinstimmung lade andere Seite (z.B. Freenet.de)

Beachte: - Platzierung des JavaScript-Bereichs an den Anfang des Bodys.
- Lagere die JavaScript-Anweisungen in eine eigene Datei aus.
- Verwende zur Passworteingabe z.B. die JavaScript-Methode „prompt“.

Inhalt von Formularfeldern testen (Arbeitsblatt 9.2)

1. Formularfelder in HTML erzeugen.
2. Die Schaltfläche „Anfrage senden“ mit dem Event-Handler „onClick“ verbinden, der z.B. die Funktion „teste_formularinhalt()“ aufruft.
3. Funktion teste_formularinhalt() schreiben. Greife über JavaScript-Objekt auf die einzelnen Formularfelder zu. Teste, ob das Feld eine leere Zeichenkette beinhaltet oder nicht.

If (document.formularname.feldname.value==““) {alert(...



Die Anzahl der Elemente eines Formulars erhält man über die Eigenschaft „length“.
→ document.forms[1].elements.length
Liefert die Anzahl der Elemente des 2. Formulars.
Die Methode forms() setzt den Cursor in das aktuelle Formularfeld.
Beachte:
HTML-Elemente wie z.B. Bilder, Tabellen, Formulare,… können über ihren Namen bzw. über ihre fortlaufende Nummer in JavaScript angesprochen werden.

Formulardaten anzeigen (Arbeitsblatt 9.3)

Clientseitig eine Antwortseite erzeugen:

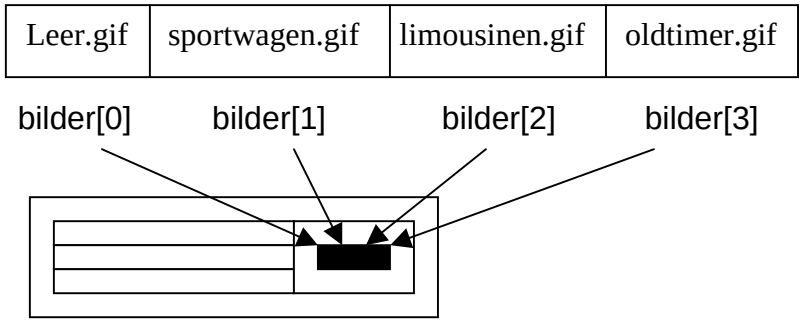
1. Anwender gibt Daten in das Formular ein.
2. Ein JavaScript-Programm liest die Formulardaten ein und erzeugt eine HTML-Datei.
   a) Lese die Eingabe des Anwenders über die Eigenschaft „value“ ein und weise sie einer Variablen zu.
   b) - Öffne neues Fenster
      - Öffne ein Dokument im neuen Fenster
      - Schreibe HTML-Markierungen und Inhalte über die Methode WriteLn() in das Dokument.

Bilder ereignisgesteuert aufrufen (Arbeitsblatt 9.5)

1. Bilder in einem Array (Feld) ablegen.
2. Erzeuge zweites Array, in dem die Bilder vom Typ „Image abgelegt werden.

} Function lade()

Funktion lade() wird beim Laden der Webseite im Body über den Event-Handler „onLoad“ gestartet.



HG-Farbe einer Tabellenzelle ereignisgesteruert verändern

(Arbeitsb.9.7)
Über die Methode „setAttribute()“ (unter document.all) können Attribute (d.h. Eigenschaften) von HTML-Elementen verändert oder festgelegt werden.

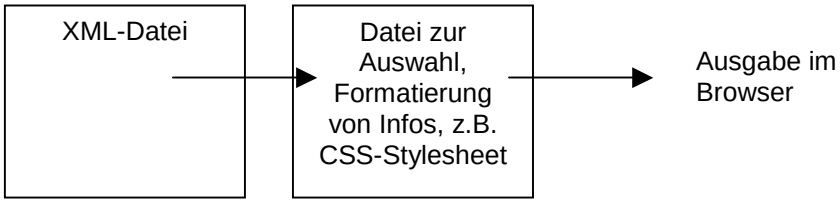
XML

Zweck:

Informationen inhaltlich zu strukturieren
→ Folge: Suche nach Inhalten, die bestimmtes Kriterium erfüllen, ist möglich
↳ Filterung, Auswahl, Verarbeitungen von Informationen hinsichtlich inhaltlicher Kriterien ist möglich
Merkmal: XML ist textbasiert (gleiche Philosophie wie HTML) und damit plattformunabhängig.

Aufbau eines XML-Dokuments:

Merke: Ein XML-Dokument beinhaltet lediglich die inhaltliche Struktur von Informationen.
Es erfolgt eine Trennung von Struktur und Layout.



Vorteile:

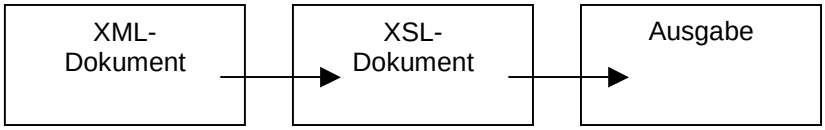
- Vermeide Datenredundanz, die XML-Datei kann man durch verschiedene Ausgabefilter laufen lassen.
- Die Filter entsprechen einer „Formatvorlage“, die auf verschiedene Dokumente angewendet werden können.
- Layoutveränderungen im Dokument (z.B. alle Preise rot färben) können in einer Anweisung verändert werden.

Cascading-Stylesheets (CSS):

- Definierte Eigenschaften eines Elements werden auf alle Unterelemente übertragen (es gibt Ausnahmen, z.B. display)
- Die Regeln eines CSS-Stylesheets beziehen sich auf Formatierung und auf alle Elemente des XML-Dokuments.

→ CSS können nicht zur Auswahl / Selektion bestimmter Daten benutzt werden.

XSL-Anweisungen zur Auswahl und Filterung von XML-Daten



Strukturierung eines Dokuments mit XML (Herdt, S.38 Übung 3)

1. Finde ein sinnvolles oberstes Element, hier z.B. SEMINAR
2. Die vorliegende Einladung besteht aus 4 Hauptpunkten. Dies sollte sich auch im XML-Dokument widerspiegeln.
   → Aus diesem Grund hat das Hauptelement „Seminar“ 4 Unterelemente: Ziel, Zeitraum, Zeitplan, Personen)

3. Das Unterelement ZIEL: Evt. existieren mehrere Dokumente mit Seminarangeboten. Möchte man z.B. über ein Programm die Ziele aller Seminarangebote, so könnte man nach allen XML-Elementen mit dem Namen ZIELNAMEN suchen lassen.
4. Das Unterelement ZEITRAUM: Ein Zeitraum wird durch Beginn und Ende festgelegt → um evt. Beginn oder Ende mehrerer Veranstaltungen zu ermitteln, trenne die Informationen BEGINN und ENDE, indem man eigene XML-Elemente für diese erzeugt. Benötigt man z.B. nur die Beginntermine, so kann man über ein Programm nach allen XML-Elementen mit dem Namen BEGINN suchen lassen.
5. Das Unterelement Zeitplan: Besteht aus den Unterelementen „Zeitüberschrift“ und „Top“ (Tagesordnungspunkt).
6. Das Unterelement Personen:

XSL-Stylesheets:

- Textbasierte Anweisungen in Form von Markierungen
- Ähnlich wie bei HTML, XML existieren Start- und Endmarkierungen
- Es existieren Dokumentkopf und Dokumentrumpf
- HTML-Elemente können zur Formatierung der ausgewählten Inhalte integriert werden
- Über <xsl: template match = „/“>
  …
  </xsl: template>
  wird das Attribut “match” festgelegt, ob sich das XSL-Stylesheet auf das gesamte XML-Dokument bezieht (dann ist Wert “/”) oder nur auf einen bestimmten Unterbaum (z.B. nur auf Unterbaum Marke
  → match=“/Autoangebot/Marke“).

Aufgaben:

1. *Schreibe alle Modellnamen aller Fahrzeuge in eine Tabellenspalte (limosinen.xml)*

   ```
   <xsl: for-each select = „AUTOANGEBOT/MARKE/AUTO“>          ①
       <xsl: value-of select = „MODELL“ />                   ②
   </xsl: for-each>
   ```

   ① Besuche im XML-Dokument alle Elemente (Knoten) mit der Beschriftung „AUTOANGEBOT/MARKE/AUTO“
   ② Gebe dann den Inhalt des Unterelements „Modell“ aus

   Ergebnis: Von allen Fahrzeugen werden die Modellnamen ausgegeben.

2. *Schreibe das 1. Modell jeder Automarke in eine Tabellenspalte*

   ```
   <xsl: for-each select = „AUTOANGEBOT/MARKE“>              ①
       <xsl: value-of select = „AUTO/MODELL“ />              ②
   </xsl: for-each>
   ```

① Besuche im XML-Dokument alle Elemente mit der Beschriftung „AUTOANGEBOT/MARKE“
② Gebe dann den Inhalt des Unterelements „Modell“ des (ersten) Unterelements „AUTO“ an.

Ergebnis: Von allen Marken wird der Modellname des jeweils ersten Fahrzeugs ausgegeben.
↓
Um den Modellnamen des jeweils zweiten Fahrzeugs auszugeben, füge in eckige Klammern den Index des XML-Elements hinzu, z.B. ②
<xsl: value-of select = „AUTO[1]/MODELL“ />

3. *Schreibe nur das 1. Modell des gesamten Autoangebots in eine Spalte*

```
<xsl: for-each select = „AUTOANGEBOT“>                    ①
    <xsl: value-of select = „MARKE/*/MODELL“ />           ②
</xsl: for-each>
```

① Besuche im XML-Dokument alle Elemente mit der Beschriftung „AUTOANGEBOT“. Da AUTOANGEBOT das Wurzelelement des Dokuments (bzw. das Element auf der obersten Ebene) darstellt, existiert nur ein solches.
② „*“ repräsentiert einen beliebigen Knoten im XML-Dokument, (d.h. ein en beliebigen Elementtyp). In diesem Fall repräsentiert * den Elementtyp „Auto“.

Ergebnis: Gehe über MARKE zum Unterelement AUTO und gebe den Inhalt des Unterelements MODELL aus.

4. *Wähle aus dem Dokument seminar.xml alle Seminarteilnehmer aus. Erzeuge eine Tabelle, die Anrede, Name und Nachname in einer Zeile für jeden Teilnehmer erfasst.*
   Lösungsweg:
   - Aufgabe4.xsl erstellen
   - Seminar_aufgabe4.xml erstellen
   - beide verbinden

Fallunterscheidungen innerhalb eines XSL-Stylesheets

Prinzip: Wähle ein XML-Element aus, falls ein XML-Element einen bestimmten Inhalt hat.

- Einfache Fallunterscheidung: (Herdt S.97 und Aufgabe5.xsl)
  <xsl:if match =“Elementname bzw. Pfad = ’Inhalt’ “>
- Komplexe Fallunterscheidung: (Herdt S.99 und Aufgabe 6)

```
<xsl:choose>
   <xsl:when match=“Elementname=’Inhalt’“>
        <!-- Anweisungen -->
  </xsl:when>
  <xsl:otherwise>
        <!-- Anweisungen werden aufgeführt falls obige Bedingung falsch ist-->
  </xsl:otherwise>
</xsl:choose>
```

Aufgaben

7. *Alle Fahrzeuge der Marke „Mercedes“*

DOM (Document Object Modell)

Idee: Zugriff (bzw. Auswahl und Verarbeitung) von XML-Elementen mit einer objektorientierten Programmiersprache. (vgl.: Zugriff auf HTML-Elemente mit JavaScript)

Vorgehensweise: (am Bsp. Von JavaScript)

- In HTML Einbindung eines JavaScript-Bereichs
- Erzeuge ein XML-Dom-Objekt, speichere dies
- Verbinde das Objekt über die Methode „load“ mit dem entsprechenden XML-Dokument
- Greife über Methoden und Eigenschaften dieses Objekts auf die Elemente des XML-Dokuments zu.

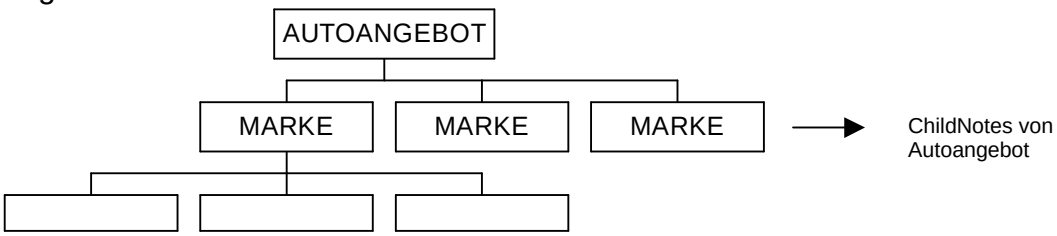
Wie erfolgt der Zugriff auf XML-Elemente? – 2 Alternativen:

1.) Durchlaufe systematisch die Baumstruktur des XML-Dokuments
2.) Suche nach bestimmten Elementnamen.

Aufgaben

8. *Erzeugen Sie die Datei DOM_limousinen_aufgabe8.html, die einen JavaScript-Bereich beinhaltet, der über DOM eine Verbindung zu limousinen_aufgabe8.xml herstellt*

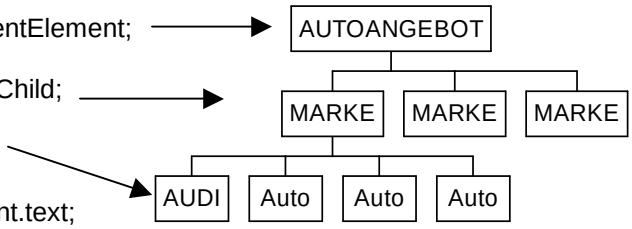
   Zugriff auf XML-Element über die Baumstruktur:



   Merke: - Objektnamen können frei definiert werden
   - Namen der Methoden u. Eigenschaften sind durch das DOM def.

9. *Welche Bezeichnung hat die erste Marke?*

```
HauptElement = XMLObjekt.documentElement;
Knoten       = HauptElement.firstChild;
KnotenElement = Knoten.firstChild;
KnotenElementText = KnotenElement.text;
document.write(KnotenElementText);
```



10. *Gebe die Informationen über alle Fahrzeuge Zeile für Zeile aus, an denen Kunden Interesse geäußert haben.*

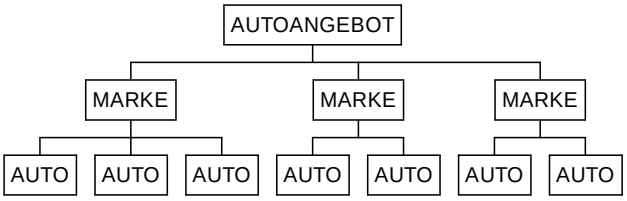
Schritte zur Lösung:

- XML.DOM-Objekt mit der Datei limousinen_aufgabe10.xml verbinden
- Wie kann man auf die benötigten Elemente zugreifen?

Verfahren:

1.) Betrachte alle Elemente vom Typ „AUTO“
2.) Untersuche für jedes Element vom Typ „AUTO“, ob das Attribut „INTERESSENT“ den Wert ja besitzt.
3.) Falls ja, so gib den Inhalt des Elements „AUTO“ an (Bemerkung: In dem Fall werden alle Unterelemente vom „AUTO“, wie z.B. Modell, Baujahr, Kilometer und Preis ausgegeben.
4.) Falls nein, tue nichts.

zu 1.)



a) Hauptelement=XMLObjekt.
documentElement

b) Betrachte alle Unterelemente vom AUTOANGEBOT
Knoten=Hauptelement.childNodes

c) Durchlaufe eine Vorschleife über die MARKE-Elemente.
For(j=0; j < Knoten.length; j++)

d) Betrachte das aktuelle Markenelement
Knotenelement=Knoten.item(j);

e) Greife auf die Unterelemente vom Typ AUTO zu:
Unterknoten=KnotenElement.
childNotes;

Beachte: Werden neue Marken in die XML-Datei hinzugefügt oder entfernt, so erfüllt der JavaScript-Code nach wie vor seinen Zweck.

```
zu 2.) for(i=1;i<Unterknoten.length;i++)        → Betrachte alle Unterelemente vom Typ Auto
       {UnterknotenElement=Unterknoten.item(j);  → Betrachte das aktuelle AUTO-Element
       UnterknotenEllementAttr =  Unterknoten    → Greife auf das erste Attribut des aktuellen
                                  Ellement.attributes (0)  AUTO-Elements zu
       if(UnterknotenElementAttr.text == „ja“)
zu 3.) {document.write(Unterknoten.Element.text);  → Gibt den gesamten Inhalt von AUTO aus.
       document.write(“<br>“);                     → Erzeugt Zeilenumbruch
       }
```

11. Schreiben Sie ein JavaScript-Programm innerhalb einer HTML-Datei, das alle Modellnamen von Fahrzeugen aus limousinen_aufgabe11.html in einer Spalte einer Tabelle ablegt.

12. Fügen Sie als Erweiterung zu Aufgabe 11 eine zweite Spalte hinzu, die die entsprechenden Preise beinhaltet.

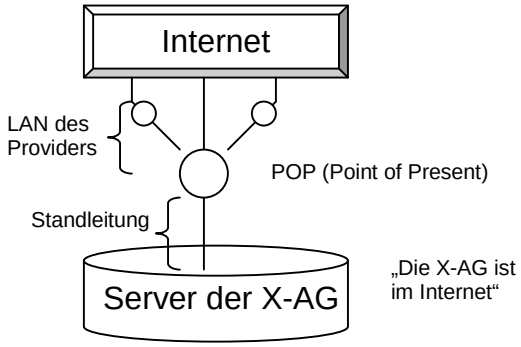
Vorgehensweise:

- Führe eine neue Variable y ein, die das Array der Preise definiert
- Gib die Inhalte des Arrays in der Spalte aus.

Grundlagen des Internet Teil II

Übertragungsmedien

- Funk / Satellit / Infrarot
- Telefonleitung
- Standleitung (permanente Verbindung → mittlere u. größere Unternehmen)
- ISDN

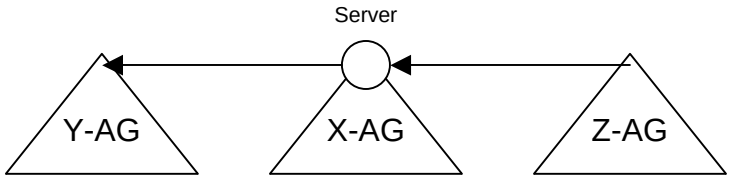


Der POP ist ein Rechner, der i.d.R. bei einem Provider steht, der selbst wiederum in Verbindung mit anderen Rechnern des Internet steht.

Auf diese Weise erhält die X-AG eine ständige (24h) Verbindung zum Internet.

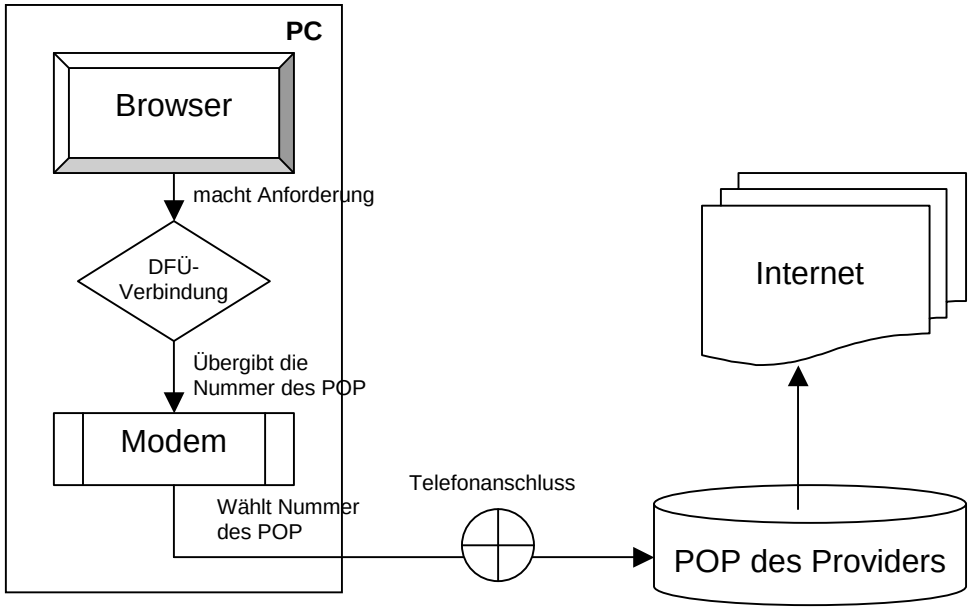
Wer bezahlt das Internet?

Prinzipiell bezahlt man nur für die Verbindung zum POP.



Schickt die Z-AG Daten zur Y-AG, so kann der Server der X-AG als Zwischenstation genutzt werden. Gebühren können hierfür von der X-Ag nicht verlangt werden.

Zugang zum Internet über Telefonleitung (z.B. Anwender von zuhause, temporär)



Installation eines Modem

- Bei einem externen Modem: Anschluss an serielle oder USB-Schnittstelle
- Verbindung zu Telefondose herstellen
- Modem über Plug&Play oder über Systemsteuerung oder über ein mitgeliefertes Installationsprogramm softwaremäßig installieren und konfigurieren (Installation von Treibern, Konfiguration von Systemressourcen, z.B. IRQ-Nummern, etc.)
- DFÜ-Verbindung einrichten, d.h. über Systemsteuerung → Netzwerk- und DFÜ-Verbindungen → Neue Verbindung erstellen → In ein Privates Netzwerk einwählen → die Daten des POP angeben → fertigstellen.

Der Browser kann über den Menüpunkt "Extras – Internetoptionen" Verbindungen zu einer DFÜ-Verbindung erhalten.

Zugang zum Internet über ISDN (z.B. Anwender von zuhause, temporär)

- Erlaubt eine schnellere Verbindung.
- Beachte: ISDN-Anschluss ist teurer als Telefonanschluss, aufgrund der höheren Geschwindigkeit und einer damit ceteris paribus geringeren Verbindungszeit zu POP kann diese bei hohem Datenvolumen zu geringeren Gesamtkosten führen.
  ceteris paribus = „alle anderen gleich", d.h. alle anderen Voraussetzungen sind identisch.

Installation einer ISDN-Karte

- ISDN-Karte einbauen
- Karte über Plug&Play oder ein beigefügtes Installationsprogramm softwaremäßig installieren und konfigurieren.
- NDIS – WAN – CAPI – Treiber und/oder CAPI-Modem installieren
- Siehe oben: DFÜ-Netzwerk einrichten und Internetzugang einrichten (siehe Seite 745 Bild 9.14).

Zugang zum Internet über ADSL (hier: T-DSL – temporär)

Übertragung der Daten erfolgt über Telefonleitung bzw. ISDN.

Voraussetzungen:

- Handelsübliche Ethernet-Karte
- Splitter
- ADSL-Modem

Anschlussvarianten:

- Variante 1: (S. 751 Bild 9.19)
  - Alle PCs erhalten über Hub Zugang zum Netz über ADSL-Anschluss
  - An allen PCs müssen die ADSL-Treiber installiert sein
  - Zu einem Zeitpunkt hat nur ein PC-Zugriff zum ADSL-Zugang
- Variante 2: (S. 751 Bild 9.20)
  - Lediglich ein ausgewählter Rechner (= Server; i.d.R. zwei Netzwerkkarten) kann sich über ADSL einwählen. Die einzelnen PCs richten in dem Fall eine Anfrage an den Server, dieser nimmt dann die Einwahl vor.
  - Der *Vorteil* dieser Variante liegt in der erhöhten Sicherheit. Der Server kann als Proxy dienen, der ...
    ... Webseiten cached
    ... eingehende und ausgehende Anfragen verweigert (→ Firewall)
  - Ferner wird die Verwaltung vereinfacht, da z.B. bestimmte

Sicherheitssoftware nur auf *einem* Rechner (dem Proxy-Server) installiert werden muss und *nicht auf allen* Clients.

- Variante 3: (S. 752 Bild 9.21)

Computersicherheit

- Was sind Viren? → S. 6/7 Herdt-Skript
- Warum sind Viren im Internet besonders gefährlich?
  Viren können sich mit exponentieller Größe im Netz verbreiten, z.B. Benutzer A empfängt einen Virus, dieser pflanzt sich über das Adressbuch von A auf die Benutzer B, C, D, ... n fort. Dieser Vorgang wiederholt sich bei B, C, D,... n jeweils.
  Nach n Schritten sind O(2“) (→ größenordnungsmäßig exponentiell viele) Benutzer befallen.
  Viren die sich auf andere Rechner im Netz verbreiten, heißen *Würmer*.
- Insbesondere Webanwendungen zeichnen sich durch Benutzerinteraktionen aus (um Geschäftsprozesse zu unterstützen). Interaktionselemente (z.B. Schaltflächen bei Formularen) könnten mit *Event-Handlern* hinterlegt werden, die Funktionen oder Programme starten, die dann das System beeinflussen.

Gruppenviren:

- Bootviren: Befallen häufig den Master Boot Record (MBR). Der MBR beinhaltet die Partitionstabelle und ausführbaren Code zum Start eines BS auf einer Partition. Verändert ein Programm (Virus) den MBR, so kann i.d.R. kein BS mehr gebootet werden. Hilfe bzw. Reparatur ist dann nur noch von „außen“ mithilfe einer bootfähigen Installations-CD oder einem anderen Bootmedium möglich.
- Dateiviren: Befinden sich in ausführbaren Programmen. Datei startet i.d.R. bei Start des Programms zunächst den Virencode.
- Makroviren: Sind häufig VBA-Anweisungen in Verbindung mit MS-Office-Anwendungen. Befinden sich in Office-Dateien.
- Scriptviren: Sind über den *Scripting Host* an ein *Betriebssystem* gebunden.

Abwehr von Viren:

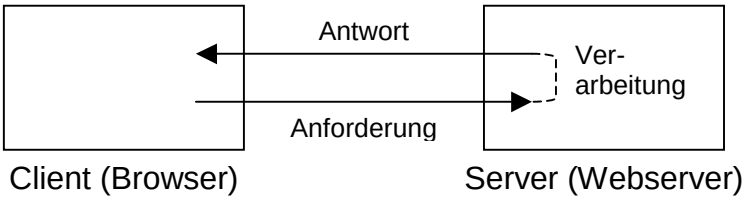
- Virenscanner: Programm, das nach Start Dateien auf Speichermedien auf Virencode überprüft.
- Sicherheitseinstellungen am Browser und am System
  - Scripting Hosts des BS deaktivieren
  - Im Browser Java, JavaScript oder ActiveX deaktivieren (Problem: Interessenkonflikt in Bezug auf Systemsicherheit und Benutzerkomfort)
- Verhalten des Benutzers:
  - Öffne nur eMails von bekannten Personen
  - Achte auf ungewöhnliche Betreffzeilen bzw. Mailgrößen
  - Mail im Zweifelsfall *löschen* oder Administrator benachrichtigen.
    → auch unter dem Befehl löschen könnte über ein Event-Handler ein Virus gestartet werden.
- Im WWW:
  - Besuche nur bekannte Seiten
  - Gebe keine persönlichen Informationen (Name, eMail, PINs, TANs, Passwörter) in Formularen ein, es sei denn es handelt sich um bekannte Seiten und der Zweck erfordert es unbedingt.

Desinfektion eines befallenen PC:

- evt. Verbindung zum Internet trennen
- ggf. den Virus mit dem Scannerprogramm entfernen
- falls dies nicht möglich ist, da z.B. der Virus bereits zu einem Schaden geführt hat (z.B. Veränderung der Partitionstabelle im MBR), Rechner über bootfähige Installationsdiskette wieder starten und z.B. die Wiederherstellungskonsole aufrufen, mit der bestimmte Systemdateien wiederhergestellt werden können.

Weitere Alternativen: z.B. über Wiederherstellungskonsole retten und System neu installieren.
Voraussetzung: Es existiert eine Notfalldiskette

Sprachen für Webanwendungen

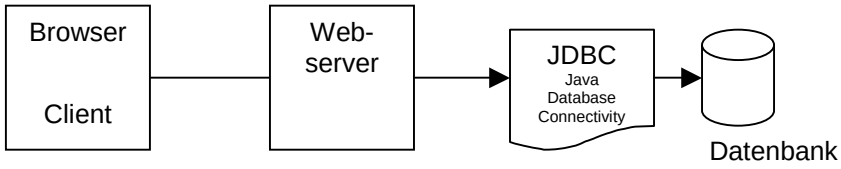


1. HTML: *Zweck*: Darstellung von Informationen
   Charakteristika: textbasiert, plattformunabhängig, leicht zu erlernen
   Nachteile/Grenzen:
   1. Reines HTML unterstützt keine Geschäftsprozesse / Interaktionen mit einem Benutzer, in denen z.B. Berechnungen durchgeführt werden → mangelnde Dynamik
   2. HTML-Dokumente besitzen keine inhaltliche Struktur. Damit verbundene Probleme:
      - Suche nach bestimmten Inhalten, die bestimmten Kriterien entsprechen, sit schwierig
      - Auswahl bestimmter Inhalte und Weiterverarbeitung schwierig
2. XML: *Zweck*: Dokumente inhaltlich zu strukturieren
   Charakteristika: textbasiert, plattformunabhängig
   Verwendung: Strukturierende Inhalte in einer XML-Datei, Suche, Auswahl, Verarbeitung von Inhalten erfolgt über Werkzeuge, wie z.B. CSS-, XSL-Steylesheets, HTML, JavaScript, etc.
3. JavaScript, VBScript:
   Zweck: Ermöglicht Interaktionen mit dem Benutzer auf Clientseite
   Charakteristika: Skriptsprachen sind „abgespeckte“ zweckbestimmte Programmiersprachen, die z.B. Zugriffe auf Festplatten aufgrund des Befehlsvorrats nicht erlauben, Variablendeklarationen können entfallen.
   JavaScript wird vornehmlich bei der Erstellung Clientseitiger Webanwendungen benutzt.
4. Java (JavaApplets):
   Erzeugung:
   1. Schreibe den Applet-Code (z.B. datei.java)
   2. Übersetze die Datei „Datei.java“ vor → datei.class

3. Binde die Datei „datei.class“ über die Markierung „applet“ mit dem Attribute „code“ in die HTML-Datei ein.

Bem.: Alle Applets werden von der Klasse „Applet“ abgeleitet.

Charakteristika: JavaApplets werden vom Browser und dessen Java Virtual Machine (JVM) ausgeführt. Das JavaApplet kann Befehle beinhalten, um Daten in eine Datenbank des Webservers zu schreiben und Daten vom Webserver zu lesen.
(JavaApplets können client- und serverseitig laufen)



= Schnittstelle, die die Anfrage innerhalb des JavaApplets in die Sprache der konkreten Datenbank und des entsprechenden Betriebssystems übersetzt.
Nachteil: Java wird vom Browser interpretiert und ist deshalb langsam.
Haupteinsatzgebiete: Webanwendungen, die clientseitig grafisch anspruchsvoll sind, Interaktionen mit dem Benutzer erfordern und die gelegentlich Informationen auf deinen Server schreiben oder von einem Server lesen.

5. PHP: *Zweck*: Wird serverseitig ausgeführt und dient vor allem dem Zugriff auf Datenbanken.
   Charakteristika: PHP wird analog einer Scriptsprache in HTML integriert. Beim Laden einer PHP-Datei wird diese nicht direkt zum Browser geschickt sondern die Befehle werden vom Server ausgeführt, z.B. werden Daten aus Datenbanken gelesen oder in sie geschrieben, wird PHP unter dem Apache-Webserver betrieben, so ist man plattformunabhängig.
6. ASP: *Beachte*: Bei ASP handelt es sich nicht um eine Sprache, sondern um eine Technologie.
   Charakteristika: ASP ist eine Erweiterung des Microsoft Webservers → Plattformunabhängigkeit
   Funktionsweise: Siehe „serverseitigesprachen_1.doc“ S. 6/7
   Aufbau einer ASP-Seite: Eine HTML-Seite die i.d.R. einen VB-Skriptbereich enthält. Im Skriptbereich können sogenannte „integrierte Objekte“ benutzt werden und von der ASP-Engine des Microsoft Webservers verarbeitet werden können.
   (siehe „serverseitigesprachen_1.doc“ S. 18/19)
7. C / Perl: *Zweck*: Ausführbare C-Programme können auf Serverseite durch die CGI-Schnittstellen gestartet werden und z.B. Formulardaten empfangen und verarbeiten.

Die Phasen der Software-Entwicklung

Problem: Heutige Webanwendungen haben hohe Komplexität, welche die Zusammenarbeit von vielen Beteiligten erfordert.
Aus diesem Grund muss die Gesamtaufgabe in mehrere Teilaufgaben zerlegt werden. Der Prozess der Softwareentwicklung vollzieht sich über mehrere Phasen.
Phasen: Siehe „skript2.doc“ und „folie2[1].doc“

Planungsphase

- Lastenheft: Beinhaltet eine fachliche Aufgabenbeschreibung, die so formuliert ist, dass Auftraggeber, Anwender und Entwickler das Problem und dessen Umfeld verstehen.
  (→ Die Grundkoordinaten sollen zwischen allen Beteiligten vereinbart werden und an alle kommuniziert werden.)
- Kalkulation: Ergebnis der Planungsphasen

Definitionsphase
Beschreibung der Geschäftsprozesse, die das zu entwickelnde System unterstützen soll (Verfeinerung der Planungsphase).
Bsp: Benutzer öffnet Webseite, betätigt Button „Belastung berechnen“. Dann öffnet sich ein Fenster, in derer die Höhe des Kredits ein gibt. Er klickt „OK“ an und die Belastung wird berechnet und angezeigt.

Entwurfsphase
Siehe Skript
Bsp: Soll z.B. die Sprache Java oder JavaScript verwendet werden?
Benötigt man ein Datenband?
Über welche Sprache / Schnittstelle greift man auf diese zu?
Soll ASP zum Einsatz kommen?
→ Entscheidungen sollen begründet werden.

- Welche Beziehungen bestehen zwischen Systemkomponenten?
- Wie kommunizieren Textsysteme?
- Ablaufbeschreibung der Programme (Alogorythmen)

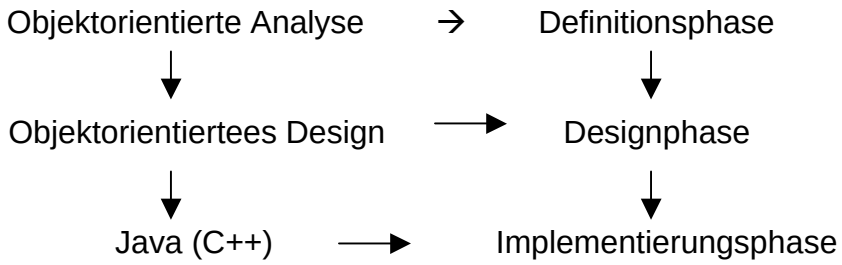
Implementierungsphase

- Erzeuge HTML-Dateien
- Schreibe z.B. CGI-Programm oder Javaskript
- Test: Dokumentation des Programms

Abnahmephase

Wartungsphase

Softwaretechnik



Abstraktion: Fokussierung auf das Wesentliche hinsichtlich der Problemstellung.

Objekt: Physischer Gegenstand oder abstrakter Gegenstand der Realität oder Phantasie.
(z.B. Baum, Hund, Mensch, Auto, Zeugnis, Beweis, ...)

Klasse: Sammlung von Objekten mit ähnlichen Eigenschaften Und Verhaltensweisen.

werden durch Methoden re-präsentiert. (Operationen)

werden durch Attribute re-präsentiert.

Botschaft: Durch eine Botschaft ruft ein Objekt ein anderes Objekt auf.

Zur Erstellung von OOA-Modellen kann die Sprache UML (Unified Modelling Language) verwendet werden.

Eine Klasse in UML-Notation:

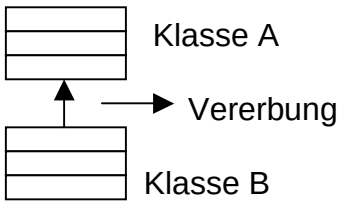
Klassenname
Attribute
Methoden

```
int y = haus40.baujahr;
```

Objektname

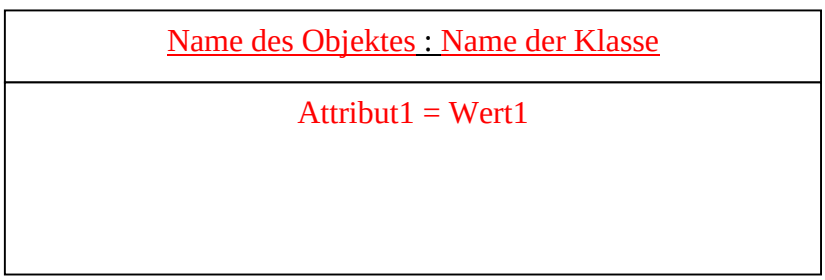
```
public class {
        String besitzer;
        String Adresse;
            .
            .
```

```
                bolean hatTiefgarage;
                int baujahr;


public Geschäftshaus (....)
{
}

public double anfragen_Verkaufspreis()
{
                return this.verkaufspreis;
}

public int anfragen_Bueroraeume()
{
                return this.bueroraeume;
}

…..main () {
  }
}

public class Einfamilienhaus (….)
{
}

public double anfragen_verkaufspreis ()
{
}
```



Klasse B erbt Attribute und Methoden von Klasse A.

UML-Notation: Objekt
z.B.



Beispiel:

Haus-Nr-40 : Einfamilienhaus
Besitzer = Meier Adresse = Saarbrücken

Der Lebenszyklus eines Objektes:

Objekt wird erzeugt (über Konstrukturoperation)
↓
Attribute werden gelesen oder geschrieben
↓
Objekt wird gelöscht

Objektoperation: Die Operation bezieht sich ein Objekt
Klassenoperation: Die Operation bezieht sich alle Objekt

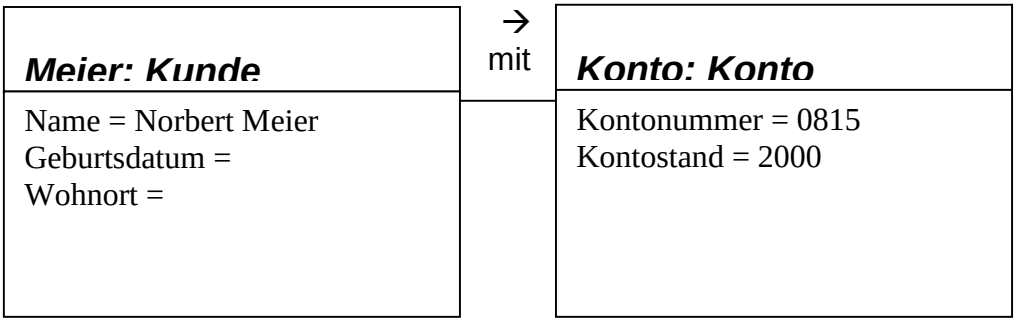
OOA →
↓
OOD
↓
Implementierung

Übungsblatt 1:

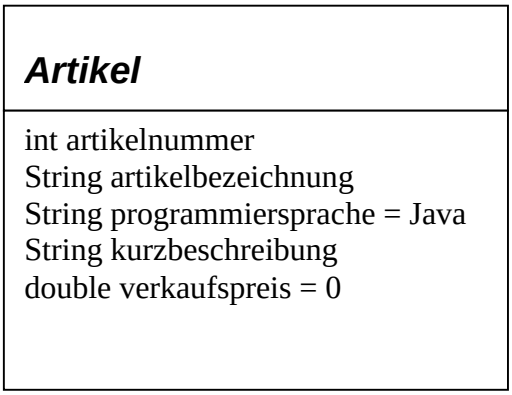
1.)

Kunde
Name Geburtsdatum Wohnort

Konto
Kontonummer Kontostand



4.)
a.)



b.)

Spezifikation des Attributs „Artikelnummer“:

1. Name : Artikelnummer
2. Typ : int
3. Anfangswert : -
4. Restriktion : nicht negative, fortlaufende Nummer
5. Klassenattribut : -
6. Abgeleitetes Attribut : -
7. Muss-Attribut : Ja
8. Schlüssel : Ja
9. not changeable : Ja
10. Wert auf Oberfläche nicht änderbar : Ja
11. Beschreibung : Gibt dem Attribut eine eindeutige Nummer

Diese Spezifikation müsste für alle Attribute der Klasse erfolgen !

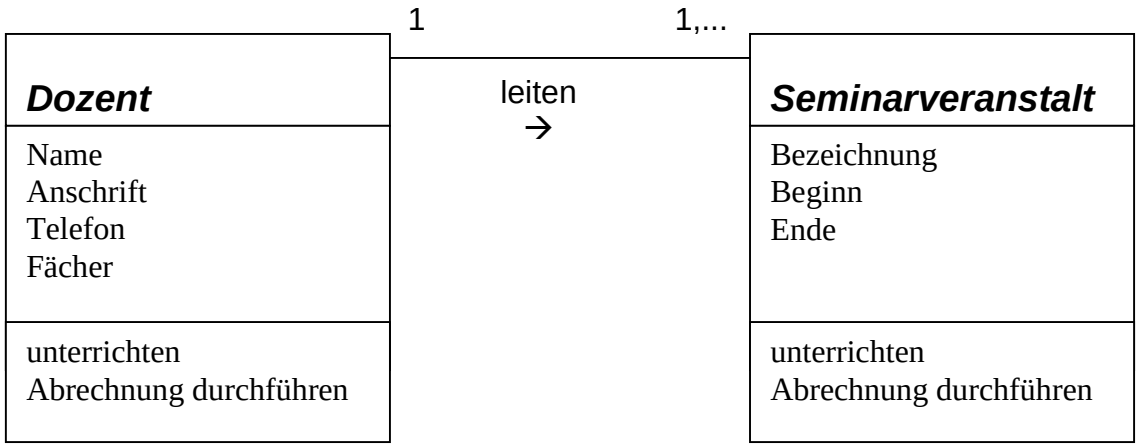
c.)

erster.Artikel : Artikel
artikelnummer = 4711 artikelbezeichnung = Diashow programmiersprache = Java kurzbeschreibung = Erlaubt Diashow auf HTML-Seiten verkaufspreis = 29,90

Übungsblatt 3:

1.)

a.)



Begründung für Klassen: Dozent und Seminarveranstaltung sind die einzigen Gegenstände in der Problemstellung, bei denen Eigenschaften und Verhalten beschrieben sind.

b.) Für Klasse Dozent:

Dozent_einstellen (=Konstruktur)
Name_lesen
Anschrift_lesen
Telefon_lesen
Fächer_lesen
} lesen

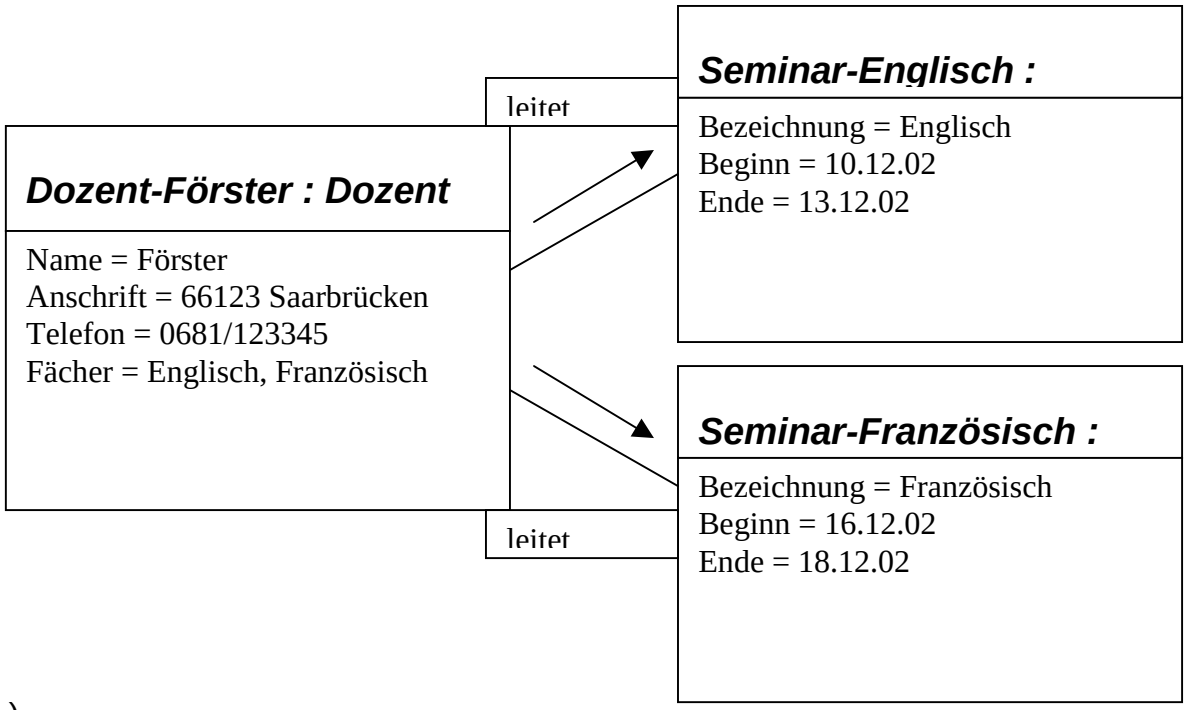
Name_verändern
.
.
.
Fächer_verändern
} schreiben

Für die Klasse <u>Seminarveranstaltung:</u>

Seminarveranstaltung_anbieten
Bezeichnung_lesen
.
.
.
Ende_lesen

Bezeichnung_verändern
.
.
.
Ende_verändern

c.)



d.)

Dozent wird eingestellt (→ Konstruktor)

↓

Dozent unterrichtet,
macht Abrechnungen,....

↓

Dozent wird entlassen oder hört auf.