

Grundkonzepte objektorientierten Programmierens mit Java

W. Mathea, M. Schillo, C. Uhrhan



Teil I: Einführung in die Objektorientierung

(aus der Java-Perspektive)

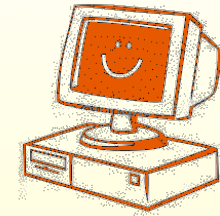
Dr. M. Schillo



Gymnasium Hermeskeil
Schwerpunkte Musik
und Informatik

Programm 1.Tag:

OO-Modellierung mit UML

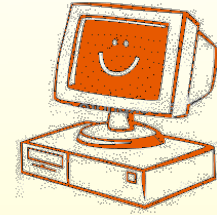


- Begriffe
 - Bauplan, Instanz, Eigenschaft, Fähigkeit
 - Klasse, Objekt, Variable, Methode
- Diagrammformen für den Unterricht
 - Anwendungsfalldiagramm, Liste der natürlichen Objekte, Klassendiagramm, Sequenzdiagramm, Aktivitätsdiagramm
 - Beispiel: Druckerwarteschlange
- Ausprobieren am Beispiel (Rollenspiel/GA)
 - Bankautomat
- ▶ Komplexe Projekte ohne Kodierung



Objektorientierung

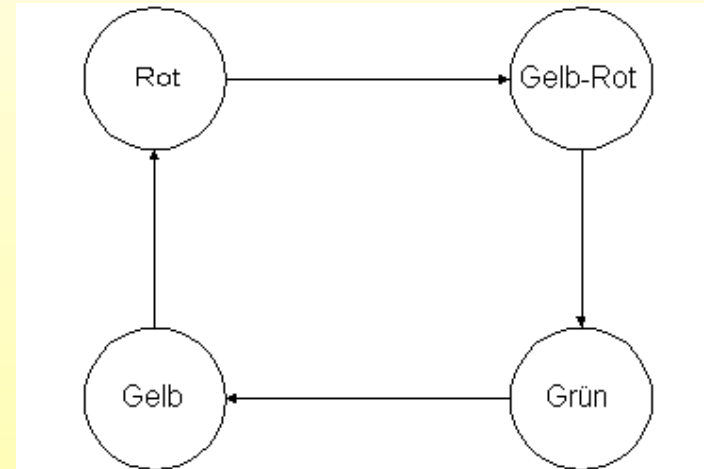
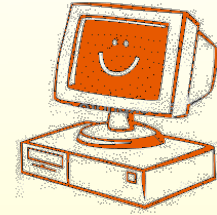
Wie verhält sich Objektorientierung zur prozeduralen Programmierung?



- Programmieren “im Kleinen” bleibt erhalten
- Prozedurorientierte Modellierung
 - Struktogramme etc.
- Objektorientierte Modellierung
 - UML



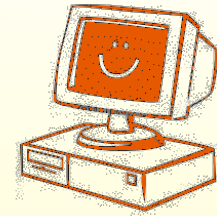
Einstiegsbeispiel



Eine Ampel kommt selten allein



Codebeispiel: Ampelklasse

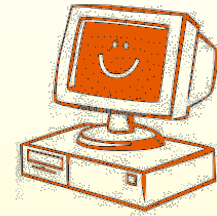


```
public class Ampel{  
    int zustand;  
  
    public void schalteWeiter(){  
        zustand = ( zustand + 1 ) % 4;  
    }  
    public void zeichneAmpel(){  
        ...  
    }  
}
```

```
Ampel ampel1 = new Ampel();  
Ampel ampel2 = new Ampel();  
...  
ampel1.schalteWeiter();  
ampel2.schalteWeiter();  
  
...
```



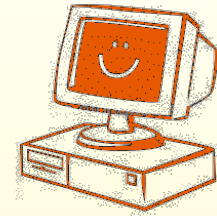
Idee der Objektorientierung



- Vorstrukturierung komplexer Systeme anhand natürlicher Objekte
- Jedes Objekt hat Fähigkeiten und Eigenschaften
- Verteilung von Lösungskompetenzen, Zuständigkeiten und Geheimnissen
- Planung des Kontrollflusses (“Meta-Algorithmus”)



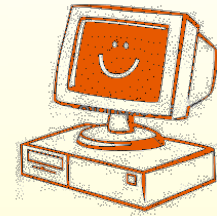
Kern der Objektorientierung



<i>Konzept</i>	<i>Effekt</i>	<i>Hintergrund</i>
Kapselung	Reduktion von Nebeneffekten	Viele Entwickler
Klassen	Reduzierung von Codelänge	Effizienz
Modellierung natürlicher Objekte	Komplexitätsreduktion	“Der Code ist die Dokumentation”



Objektorientierung: Phasenmodell



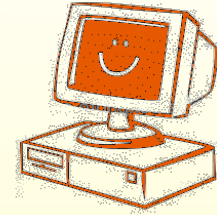
OO-Analyse → OO-Entwurf → OO-Entwicklung

OO-Modellierung

OO-Programmierung



Beispiel: Digitale Uhr



- Aufbau
 - Stunden, Minuten, Sekunden, Takt, ...
- Wie zwei Uhren (Weltuhr)?
 - den Kode kopieren und noch einmal einfügen
 - Variablen umbenennen
 - Bei Änderungen alles doppelt
- drei Uhren, vier Uhren ...?



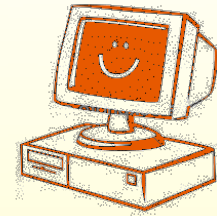
```

public class Uhr{
    int stunden, minuten, sekunden;

    public Uhr( int h, int m, int s ){
        stunden = h;
        minuten = m;
        sekunden = s;
    }
    public void zaehleWeiter(){
        sekunden = sekunden + 1;
        if ( sekunden == 60 ){
            sekunden = 0;
            minuten++;
        }

        if ( minuten == 60 ){
            minuten = 0;
            stunden++;
        }
        ...
    }
    public void zeigeUhrzeitAn(){
        System.out.println( stunden + ":" + minuten + ":" +sekunden );
    }
}

```



```

Uhr speyer = new Uhr( 14, 30, 45);
neustadt.zaehleWeiter();

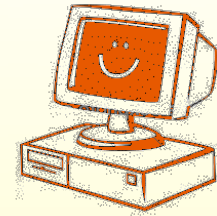
Uhr london = new Uhr( 13, 30, 45);
london.zaehleWeiter();

...

```



Begriffe der Objektorientierung



<i>Begriff</i>	<i>Beispiel</i>	<i>Erläuterung</i>
Objekt	speyer	Exemplar
Klasse	Uhr	Bauplan
Methode	zaehleWeiter()	“Prozedur”, Fähigkeiten
Objektvariable	stunden	Variable, Eigenschaften
Vererbung	AnalogUhr	Anreicherung von Eigenschaften



Bezeichnerkonventionen

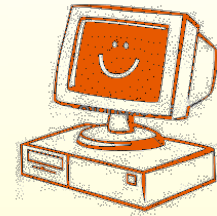


1. Sinntragende Bezeichner
 2. Konkatenation
- Klasse: `Uhr`
 - Substantiv, Großschreibung
 - Variable: `meineUhr = new Uhr()`
 - Substantiv, Kleinschreibung
 - Methode: `zaehleWeiter()`
 - Verb, Kleinschreibung

Beispiel: `meineUhr.zaehleWeiter()`



Vorteile der Implementierung mit Objektorientierung



Bessere Kapselung von:

Daten => Geheimhaltung

Zuständigkeiten => Interaktion statt Manipulation
=> Klare Schnittstellen

Komplexität => Bessere Modularisierung

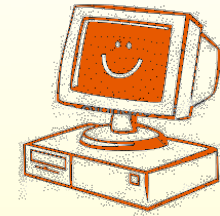
Arbeitsersparnis:

Objekte => Wiederverwendung von Code

Klassenbibliotheken => Wiederverwendung von Funktionalität



Warum Java für OOI?



- Pro
 - Plattformübergreifend
 - Einbindung ins Web -> Applets
 - Unterstützung:
 - Bibliotheken, IDEs, Literatur, Foren
 - universellere Konzepte, kompakter
- JavaScript !?!
- Kontra
 - Overhead bei Ein-/Ausgabe
 - Kein Einstieg ohne OO ?bsh?



Fragen?

UML - die Unified Modelling Language



Die wichtigsten Diagrammformen der
UML für den Unterrichtseinsatz
am Beispiel einer Druckerwarteschlange



Gymnasium Hermeskeil
Schwerpunkte Musik
und Informatik

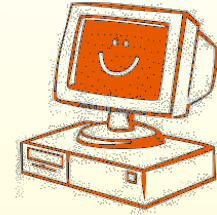
Einstieg



- Worum geht es?
 - Beispiele für UML im Unterricht
 - Denkwerkzeuge für die Modellierung
 - Allgemeinbildende Aspekte
- Worum geht es nicht?
 - Vermittlung des UML-Standards auf Niveau von Softwareentwicklern



Der Fahrplan



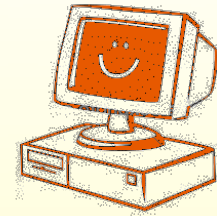
1. Druckerwarteschlange (vormittag)
2. Bankautomat (nachmittag)



- Anwendungsfalldiagramm
- Klassendiagramm
- Aktivitätsdiagramme
- Sequenzdiagramme



Anwendungsfall-Diagramm: Druckerwarteschlange



Systemgrenze

Anwendungsfälle

Anwender
Rolle

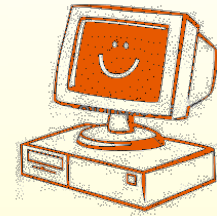
Druckauftrag
erzeugen

Druckauftrag
drucken

Warteschlangen-
inhalt ansehen



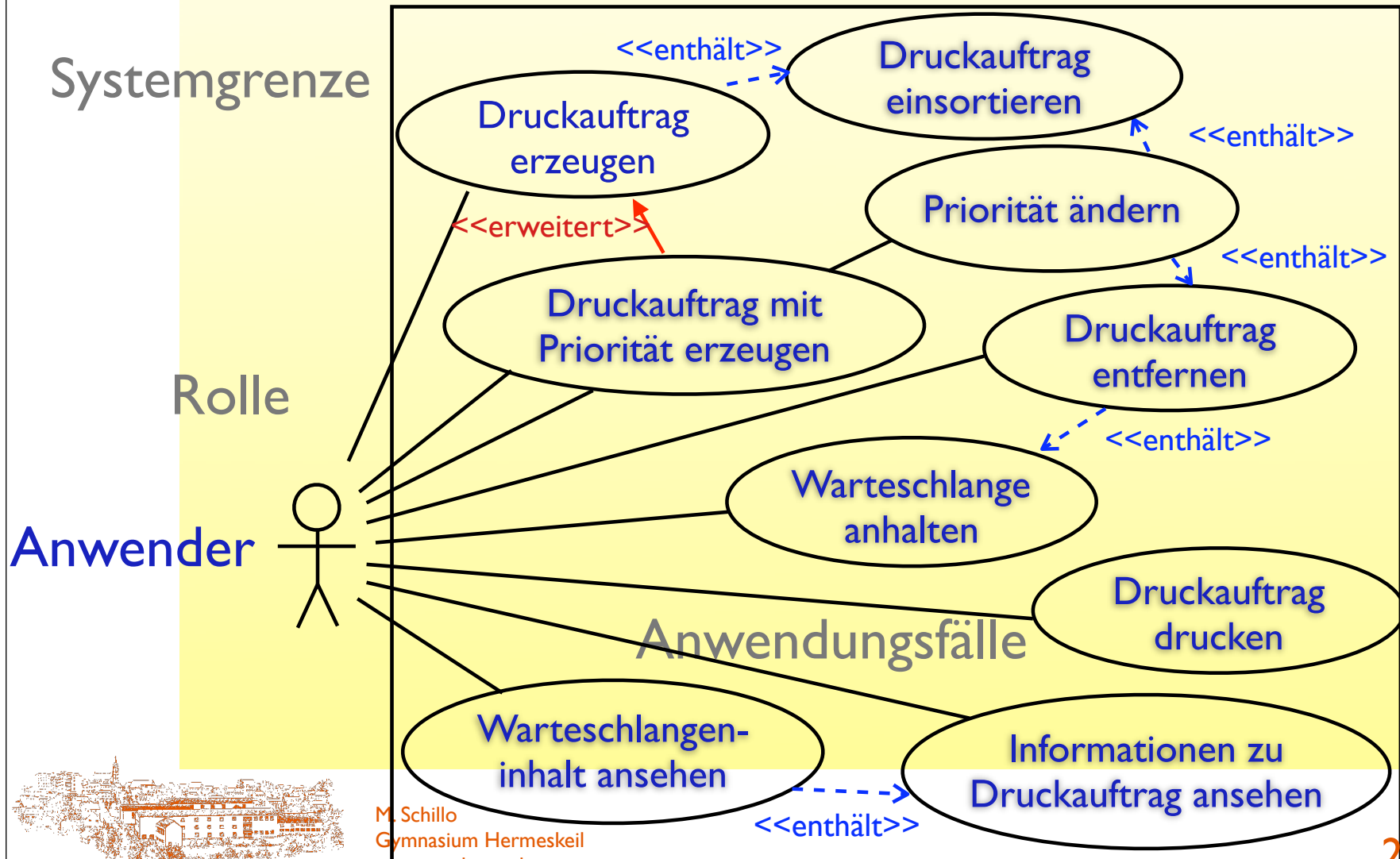
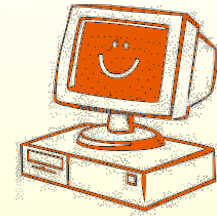
Textuelle Beschreibung der Anwendungsfälle



- Man will einen Druckauftrag erzeugen und (nach Priorität oder Reihenfolge des Einreichens) einreihen können
- Die Warteschlange muss den nächsten Druckauftrag ausgeben können
- Die Nutzer wollen die Warteschlange einsehen können
- Evtl. muss ein Druckauftrag vorgezogen werden können



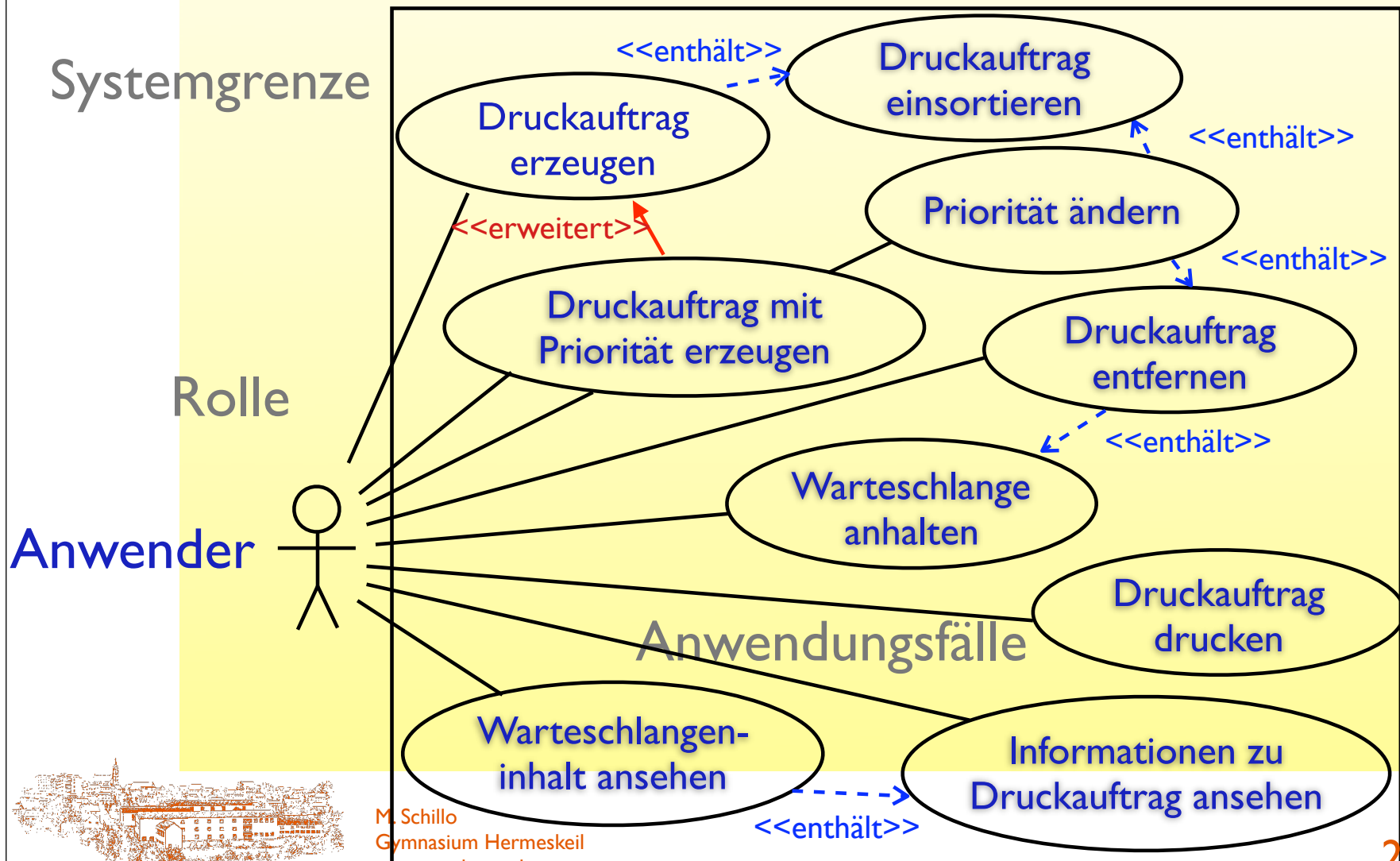
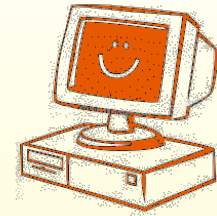
Anwendungsfall-Diagramm: Druckerwarteschlange



Rollenspiel

zu: Einsortieren eines Auftrages

Anwendungsfall-Diagramm: Druckerwarteschlange



Liste der "natürlichen Objekte"



- ~~Nutzer~~
- Warteschlange
- Druckauftrag
- ~~Papier~~
- ~~Drucker~~
- ~~Priorität~~
- ~~Papierstau~~
- ~~Dokumentname~~
- ~~Druckertreiber~~

~~Extern~~

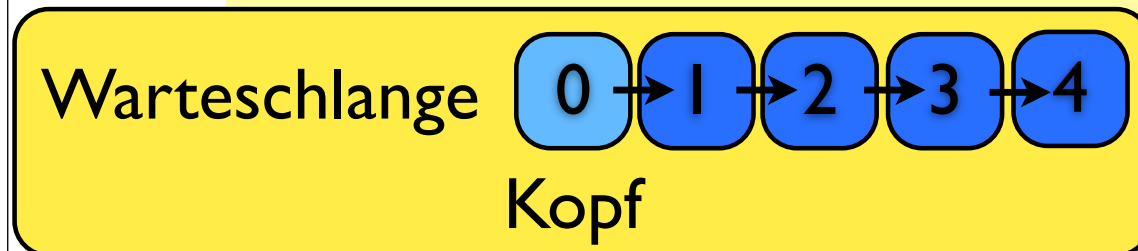
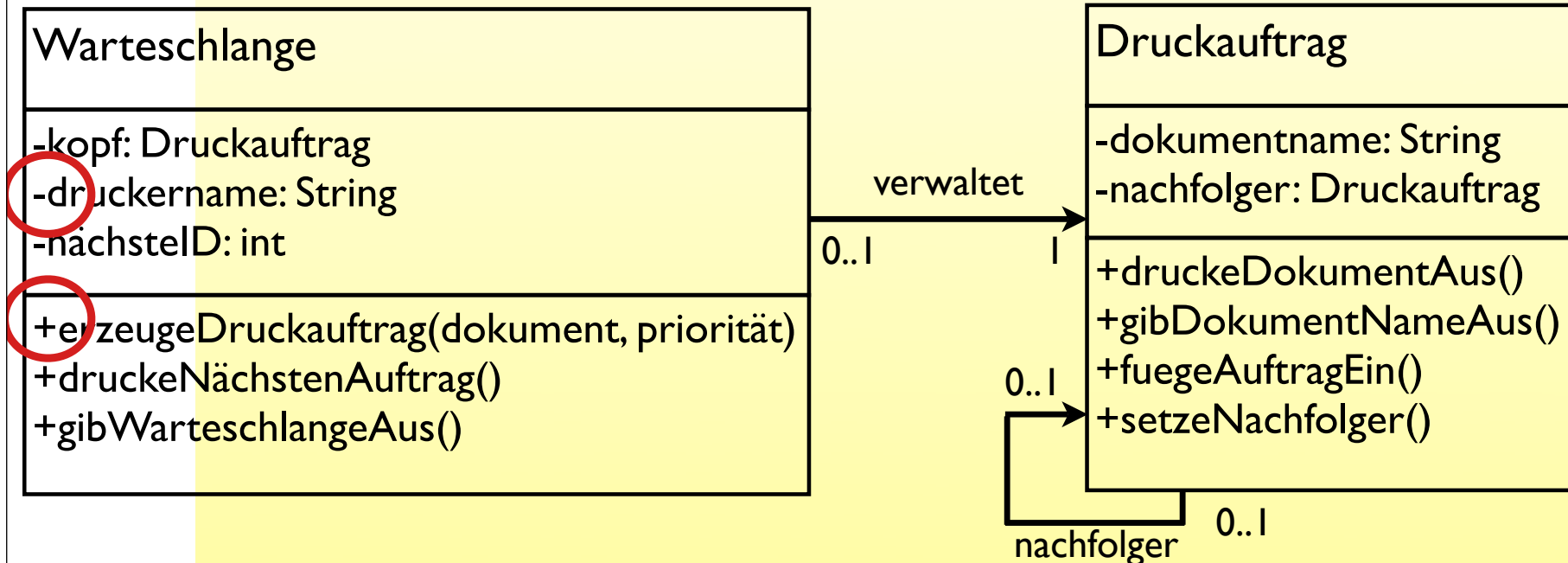
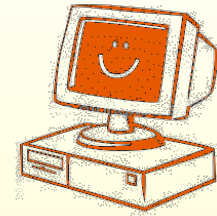
~~Elementar~~

+ Warteschlangenelement

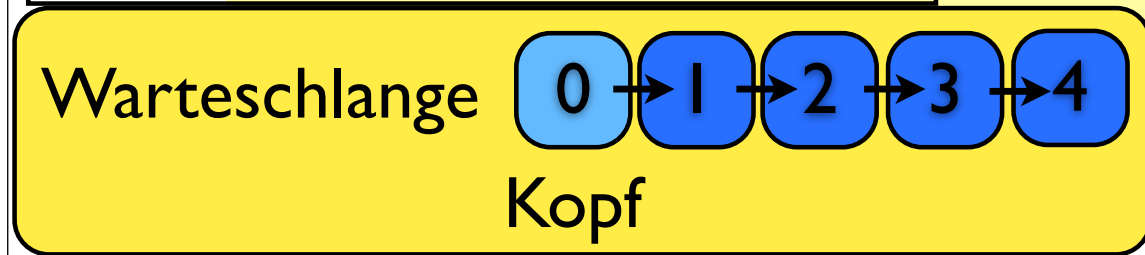
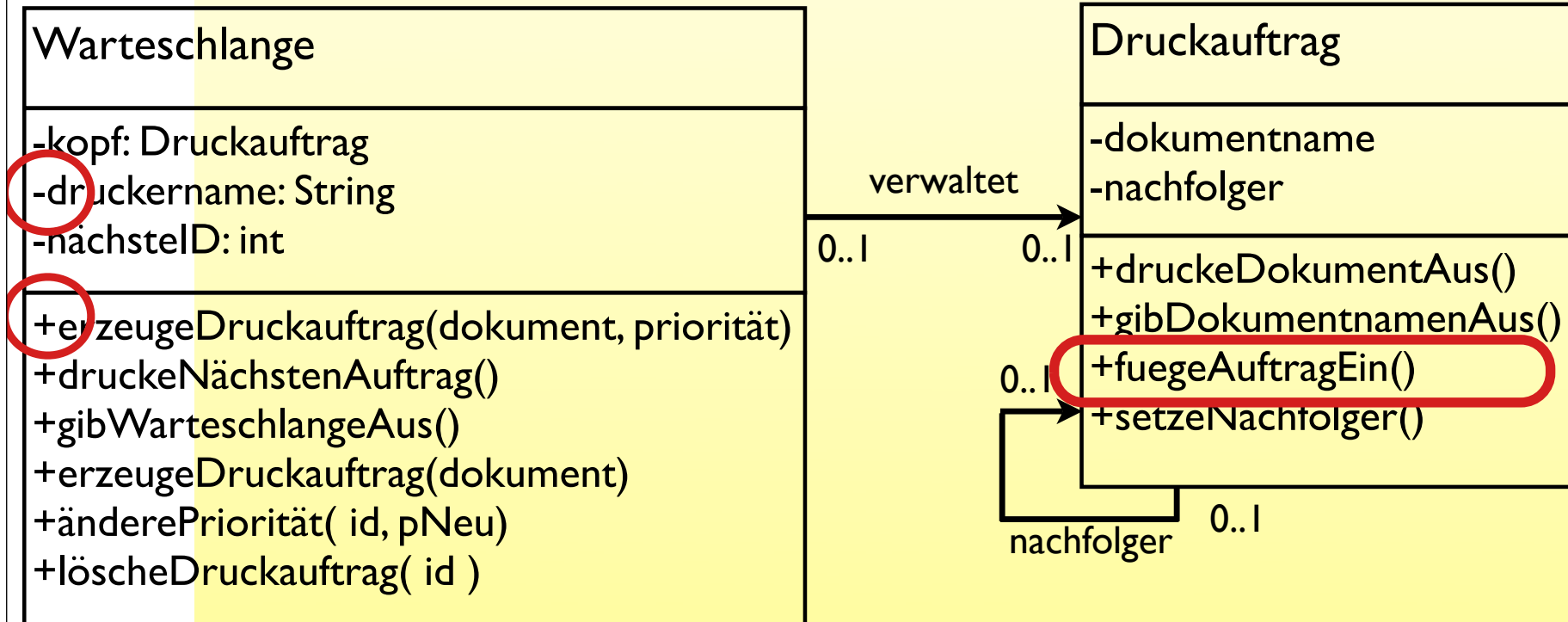
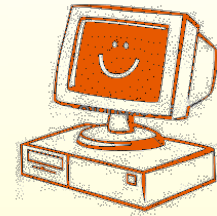
Vorbereitung des
Klassendiagramms!



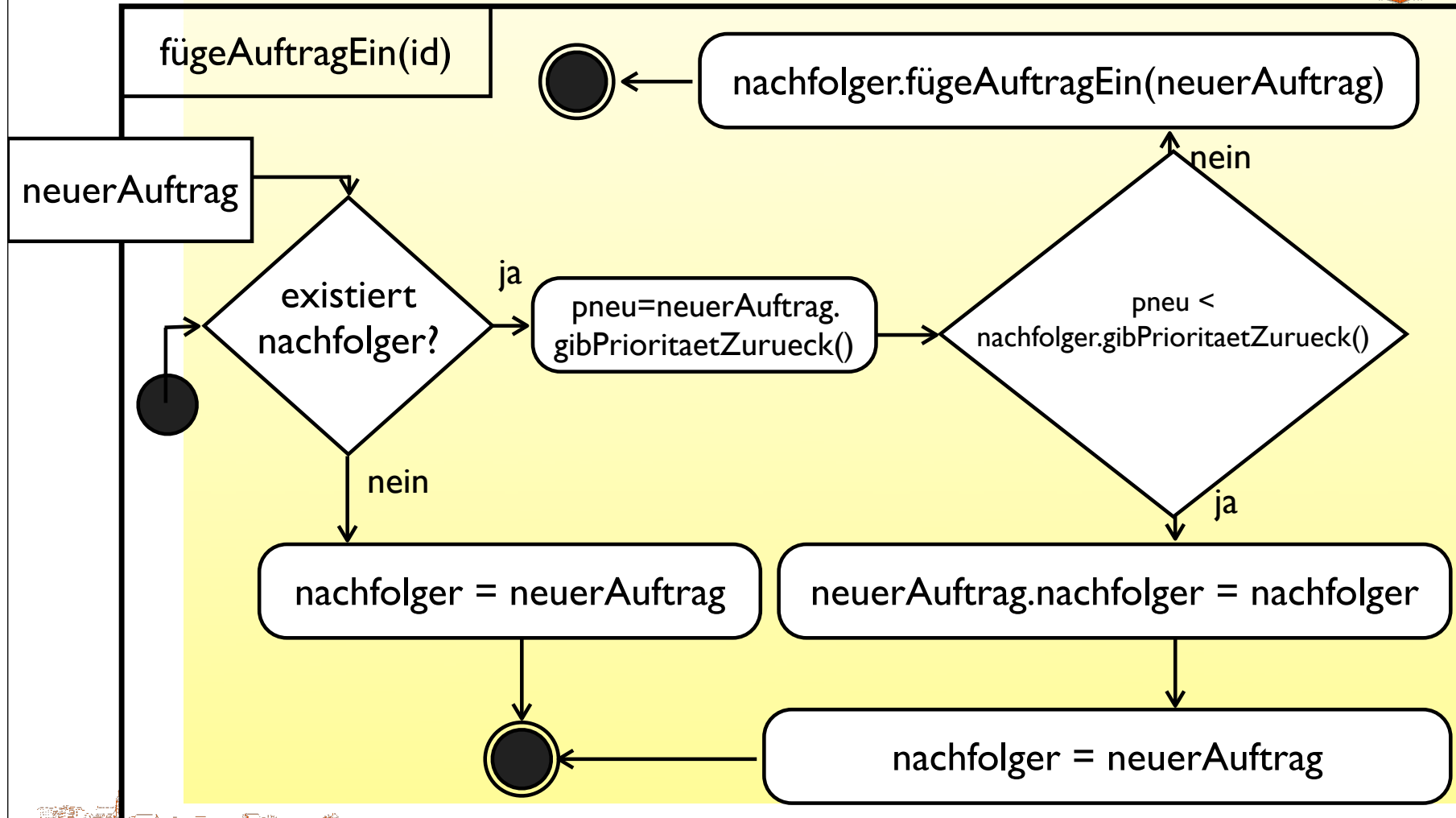
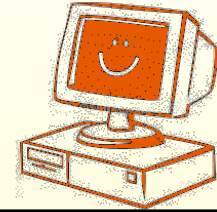
Klassendiagramm für Warteschlange und Druckauftrag



Klassendiagramm für Warteschlange und Druckauftrag (2)



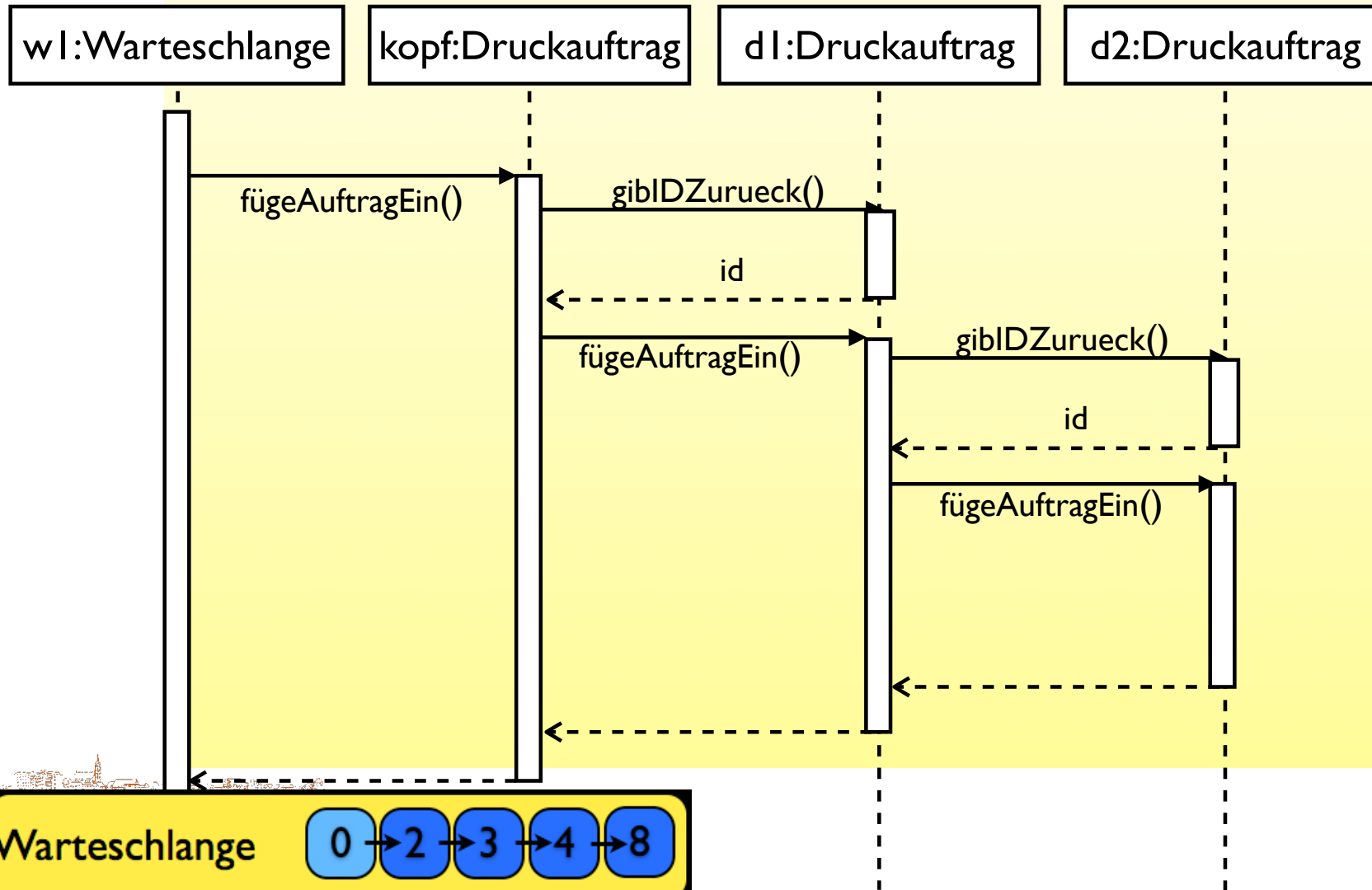
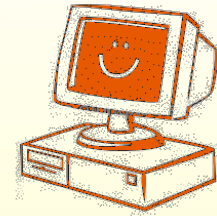
Aktivitätsdiagramm: fügeAuftragEin()



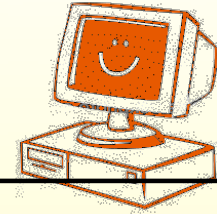
Warteschlange



Sequenzdiagramm: fügeAuftragEin()



Code: fügeAuftragEin()



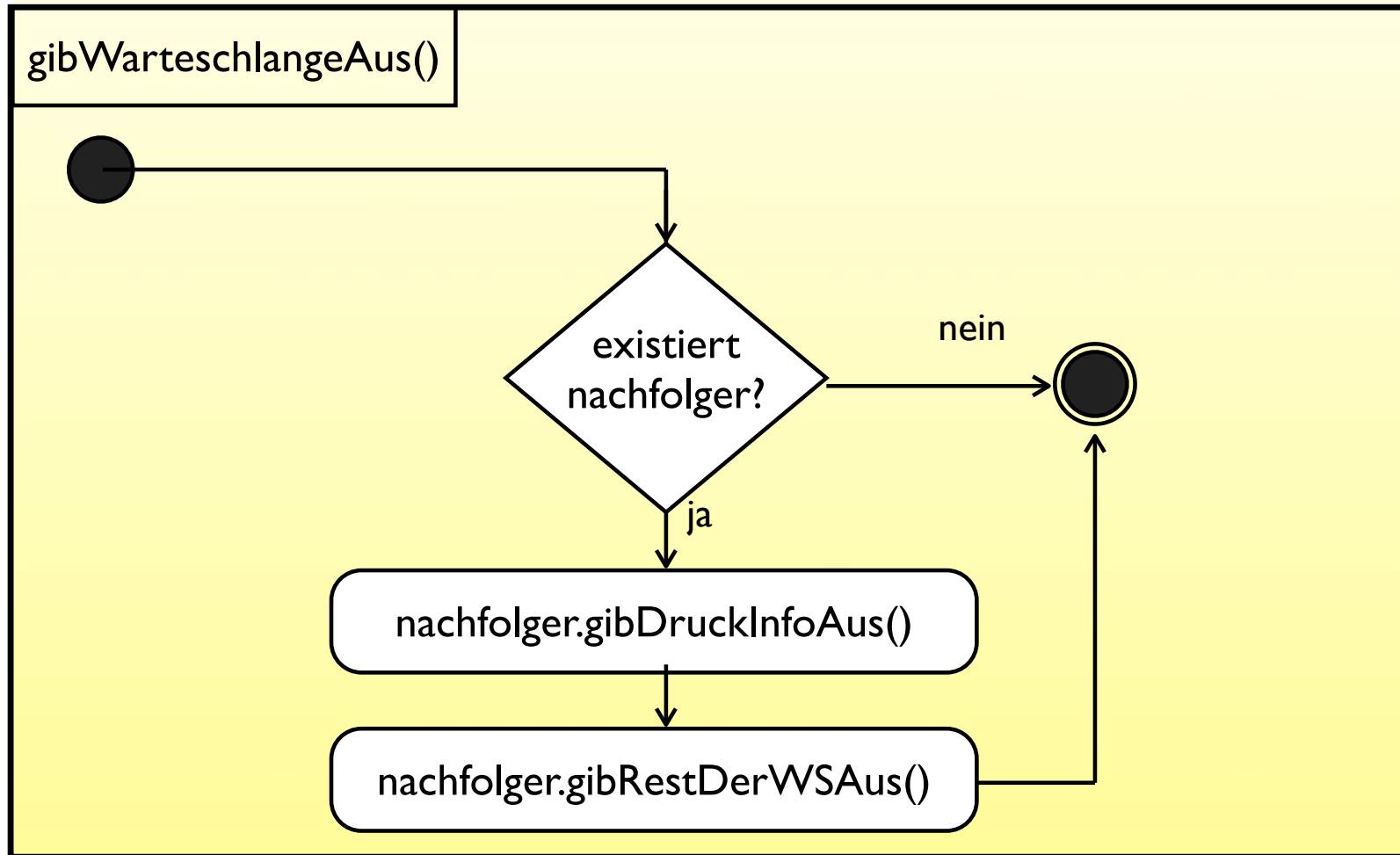
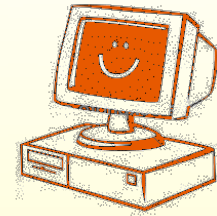
```
public class Druckauftrag{
    public void fuegeAuftragEin( Druckauftrag d ){
        if ( nachfolger == null ){
            setzeNachfolger( d );
        } else {
            if ( d.gibPrioritaetZurueck() < nachfolger.gibPrioritaetZurueck()){
                d.setzeNachfolger( nachfolger );
                setzeNachfolger( d );
            } else {
                nachfolger.fuegeAuftragEin( d );
            }
        }
    }
}
```



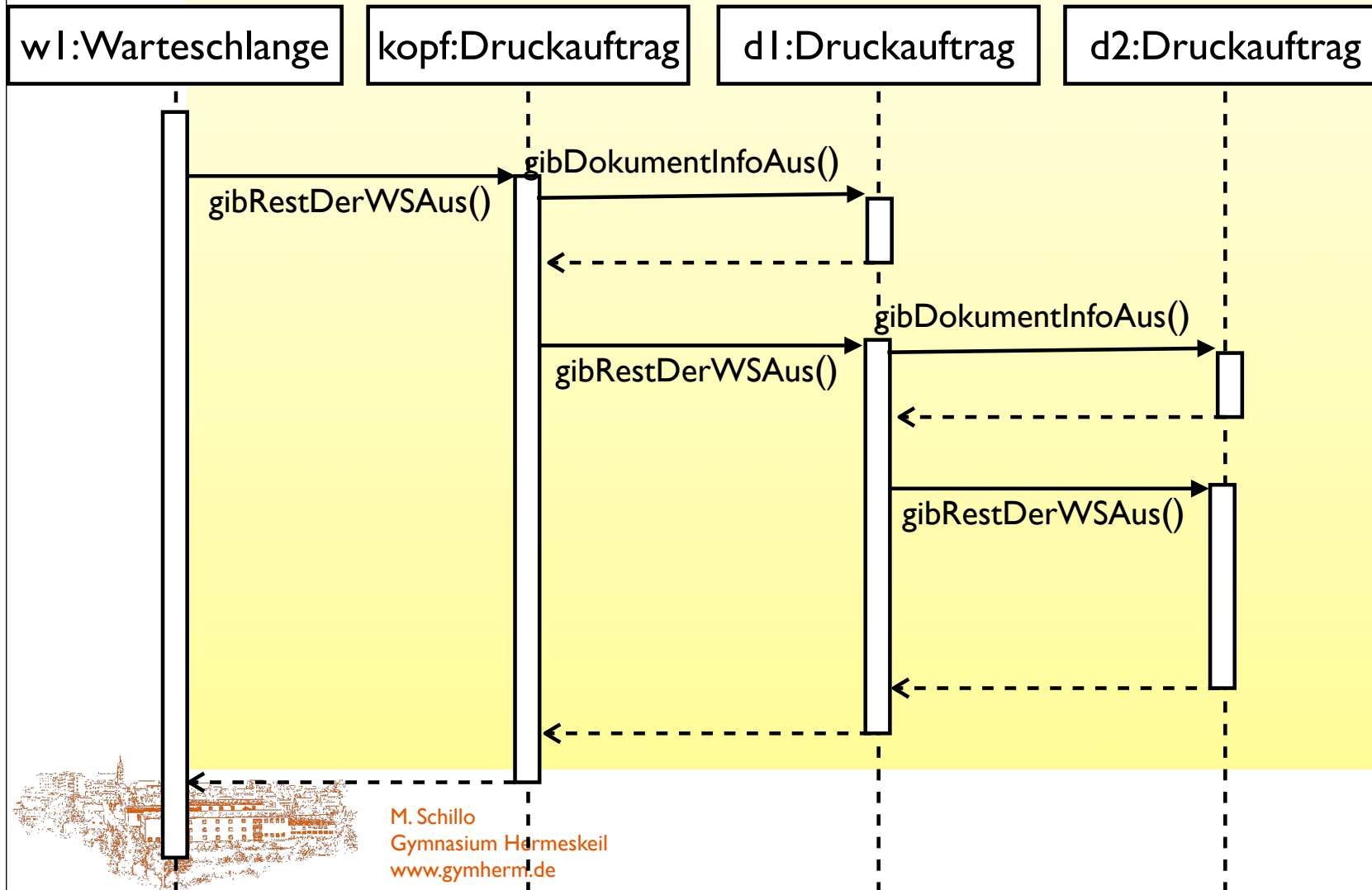
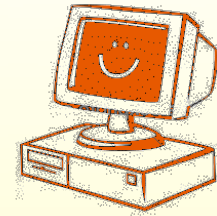
Übung

`gibWarteschlangeAus()`

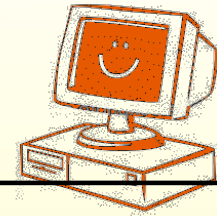
Aktivitätsdiagramm: gibWarteschlangeAus()



Sequenzdiagramm: gibWarteschlangeAus()



Code: gibWarteschlangeAus()

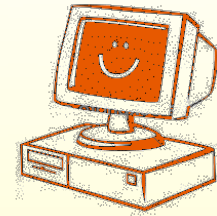


```
public class Warteschlange{  
    public void gibWarteschlangeAus(){  
        kopf.gibNachfolgerZurueck().gibAuftraegeAus();  
    }  
}
```

```
public class Druckauftrag{  
    public void gibAuftraegeAus(){  
        gibDokumentAus();  
  
        if ( nachfolger != null )  
            nachfolger.gibAuftraegeAus();  
    }  
    public void gibDokumentAus(){  
        System.out.println( id + " p: " + prioritaet + " " + dokumentname );  
    }  
}
```



Variationen



- +setze Priorität(int id)
- +aendereAnzahl(int anzahl)
- Drucken zu einem bestimmten Zeitpunkt
- Auswahl s/w oder Farbe
- Verschieben von einer Warteschlange in eine andere
- ...



Fragen?



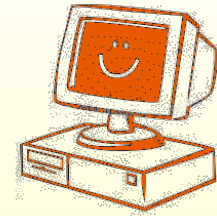
UML (2)

Die wichtigsten Diagrammformen im Überblick



Gymnasium Hermeskeil
Schwerpunkte Musik
und Informatik

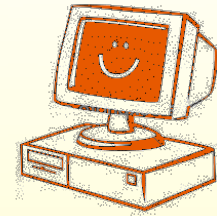
UML: Überblick



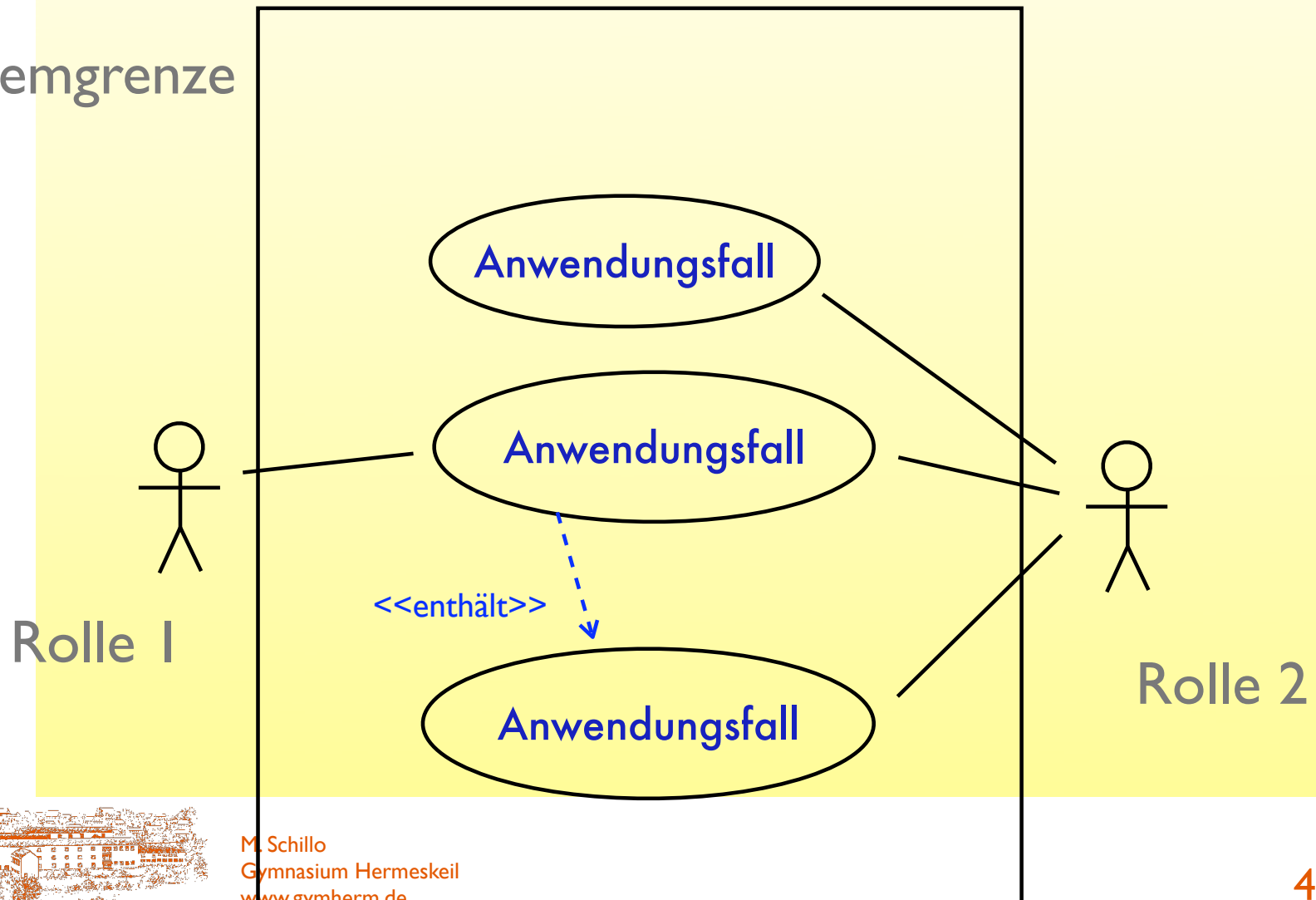
Phase	Analyse	Entwurf: Struktur	Entwurf: Verhalten
Diagramm	Anwendungsfall	Klassendiagramm	Sequenzdiagramm Aktivitätsdiagramm



Anwendungsfalldiagramm



Systemgrenze



Liste der "natürlichen Objekte", nicht in UML!



- ~~.....~~
-
-
- ~~.....~~
- ~~.....~~
- ~~.....~~
- ~~.....~~
- ~~.....~~
- ~~.....~~
- ~~.....~~

~~.....~~ Extern

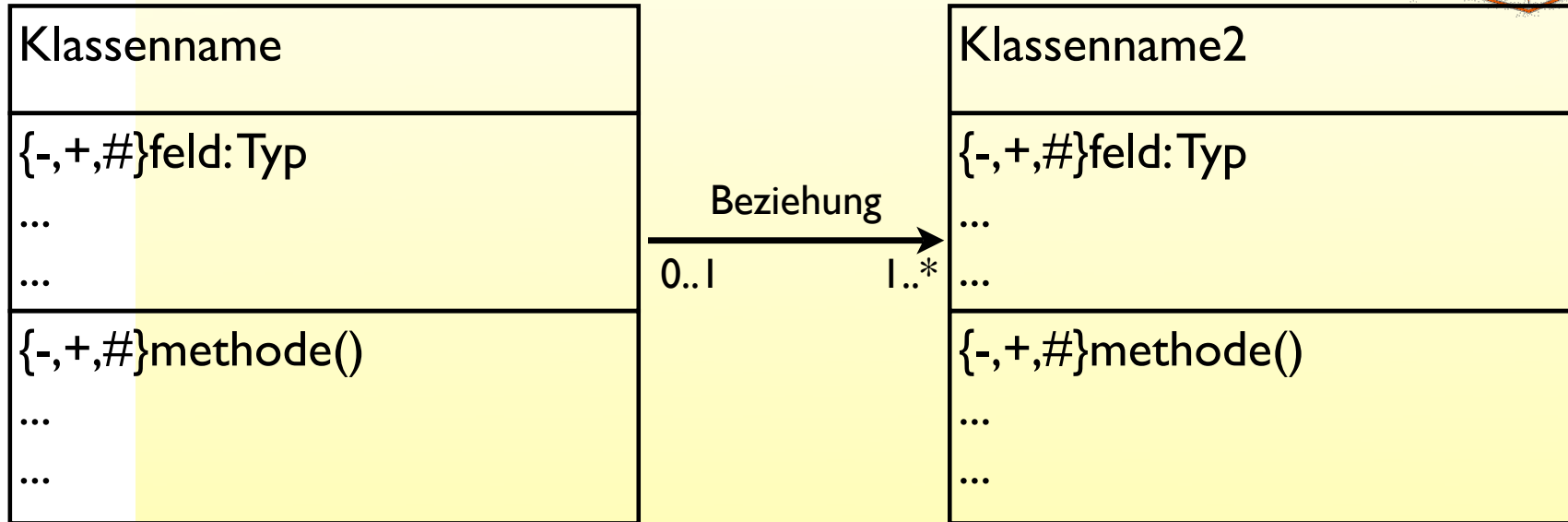
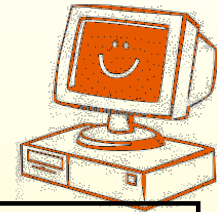
~~.....~~ Elementar

+

Vorbereitung des
Klassendiagramms!



Klassendiagramm



Sichtbarkeit:

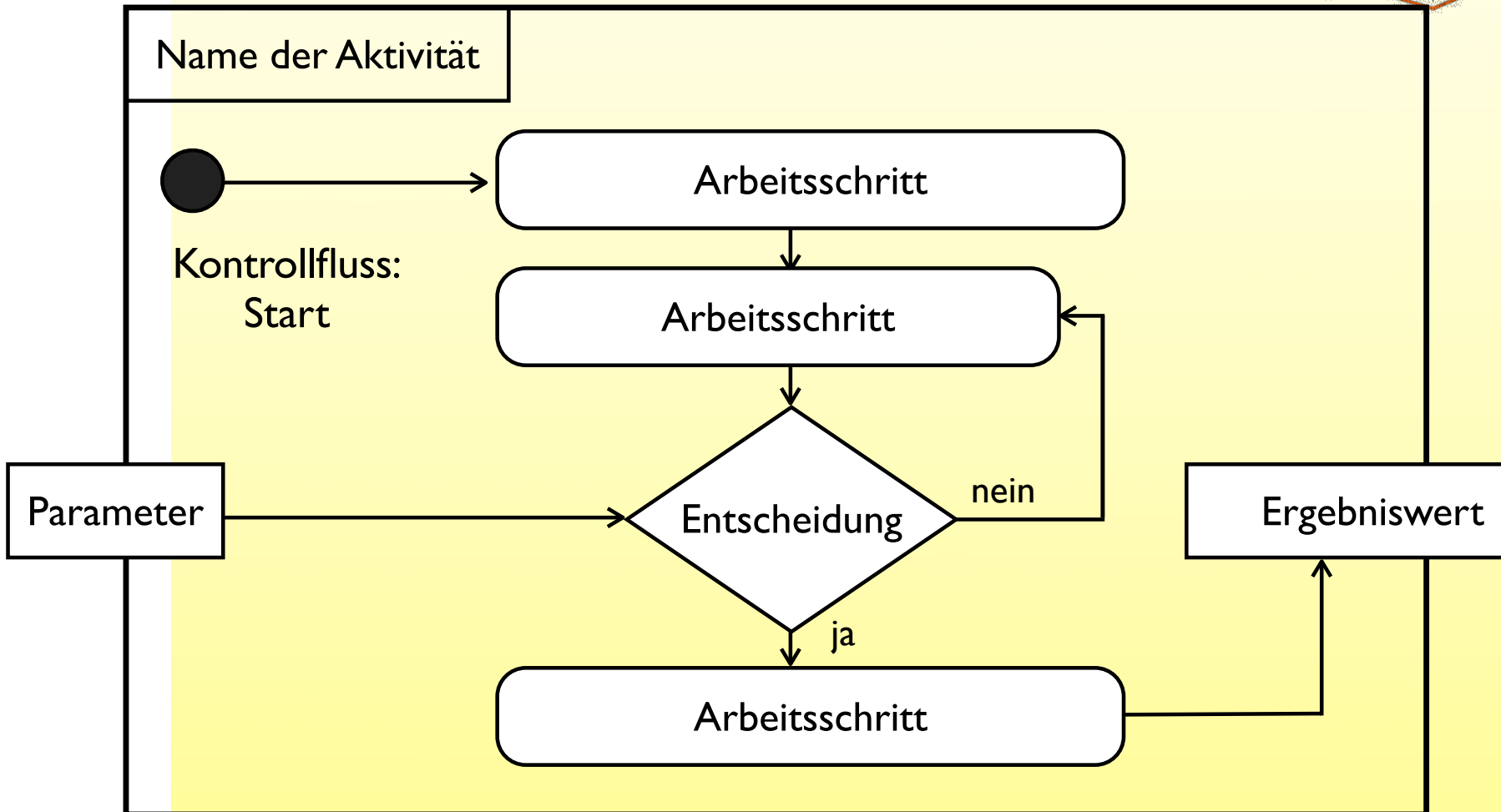
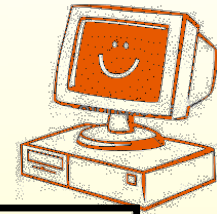
- private
- +public
- #protected

Kardinalität:

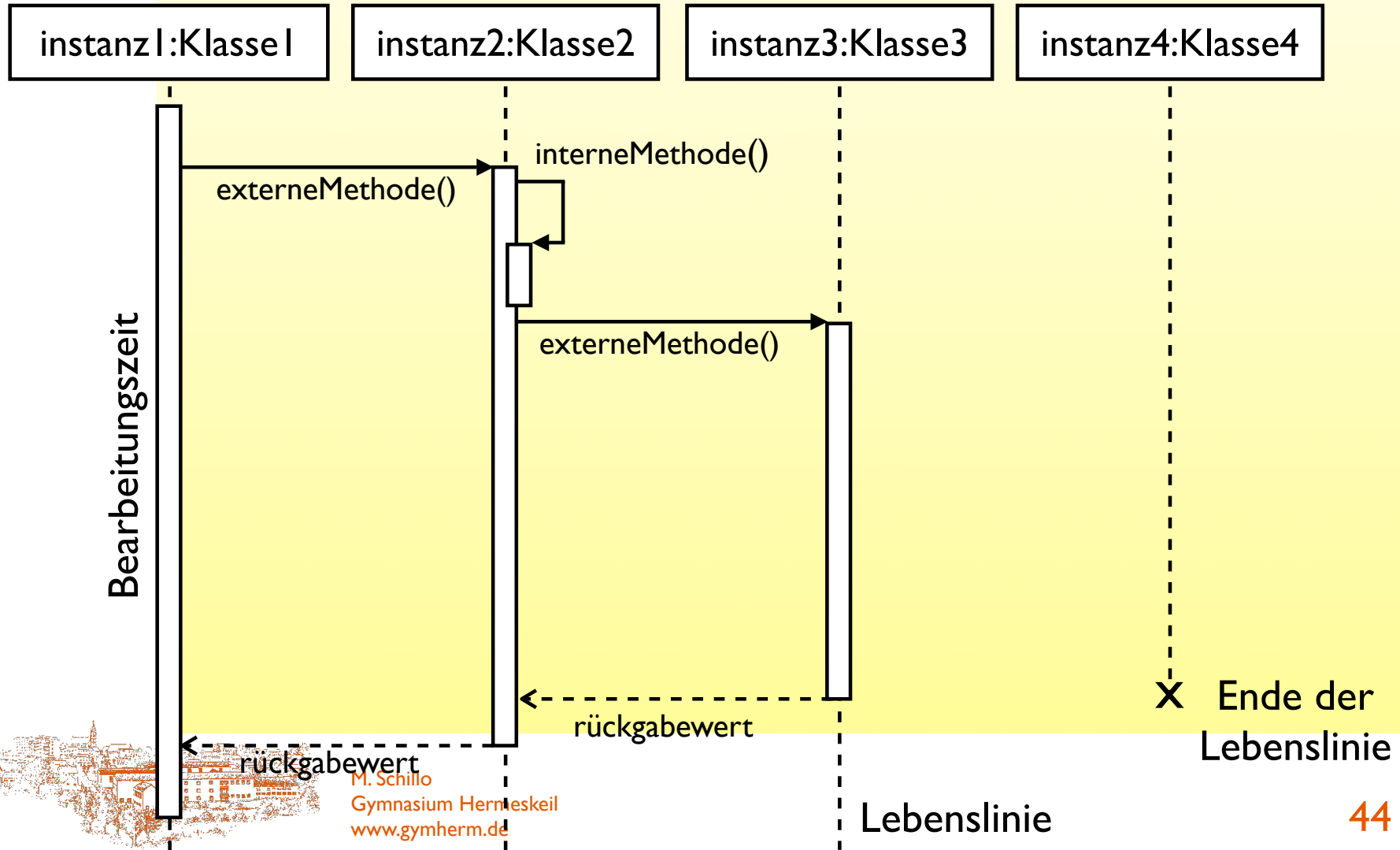
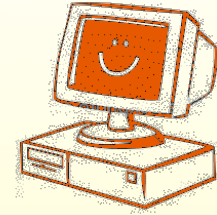
- 1:1 entfällt
- n (unbekannt, aber beschränkt)
- 0..1 (3..10 1..n)
- 1..* (1 oder mehr)



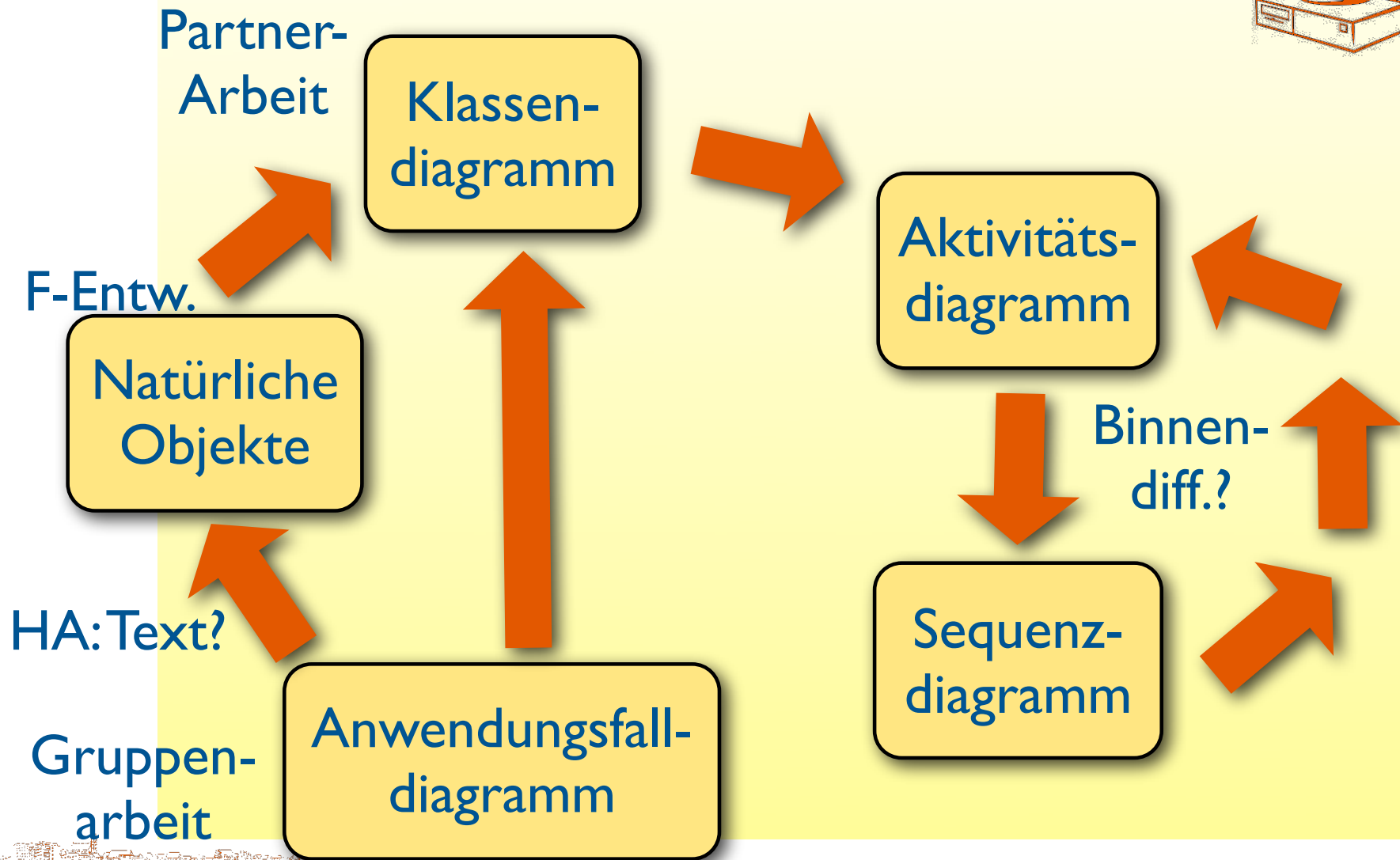
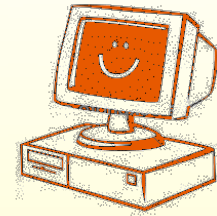
Aktivitätsdiagramm



Sequenzdiagramm



Vorschlag: Vorgehensweise für den Unterricht



Strukturelle Probleme des Unterrichtseinsatzes der OO-Modellierung



- Modellierung setzt Kodierungserfahrung voraus
 - manche Software-Objekte sind *nicht* natürlich sondern *technisch* bedingt;
 - viele *natürliche Objekte* sind nicht *relevant*
- Vorteil der Objektorientierung: bei *großen* Projekten, nicht bei kleinen
- Objektorientierung
 - setzt imperative Programmierung voraus, oder
 - *objects first*: besitzt nicht genug “Fleisch”.





Fragen?

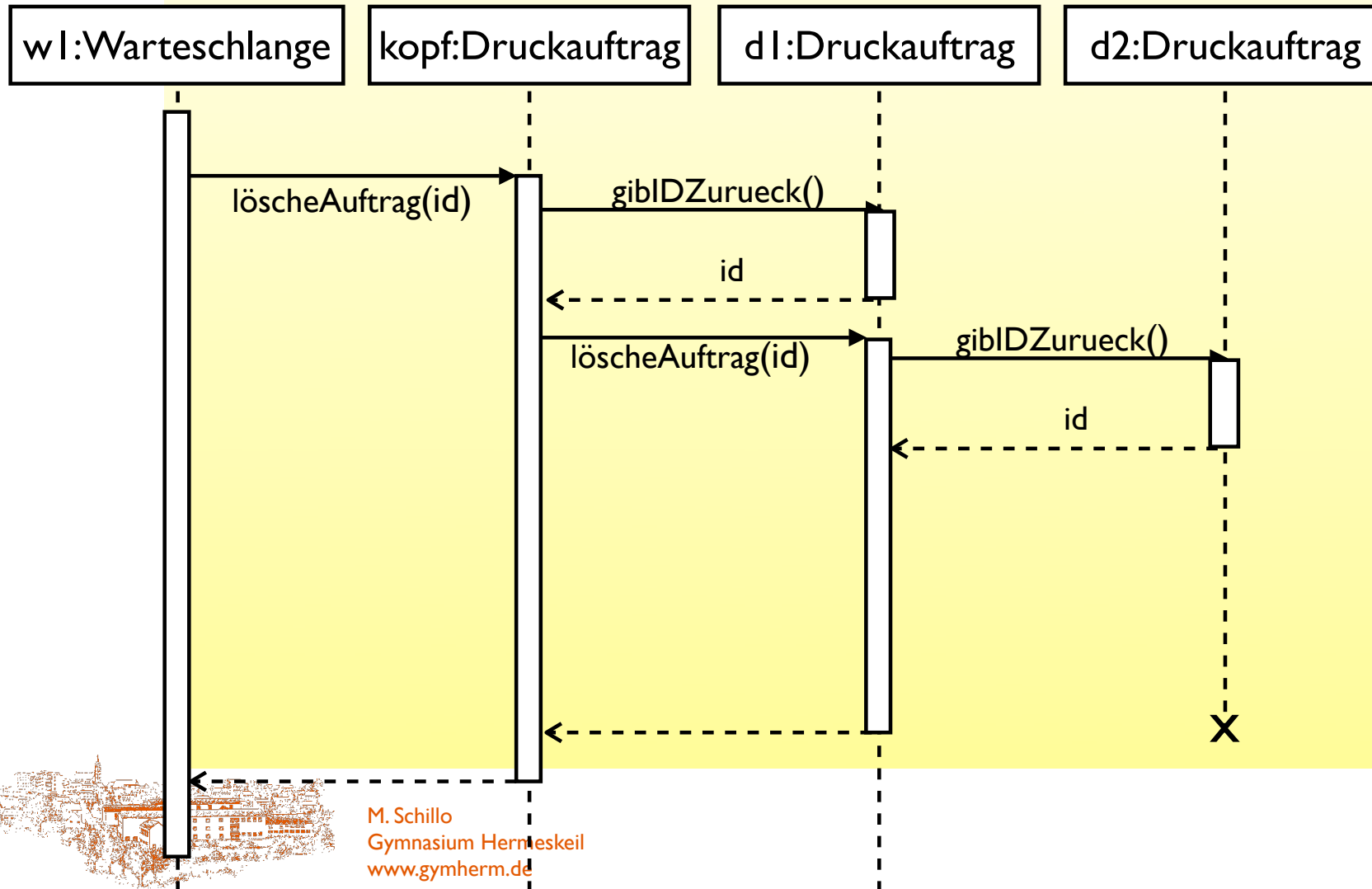
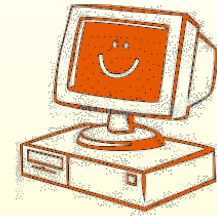


Gymnasium Hermeskeil
Schwerpunkte Musik
und Informatik

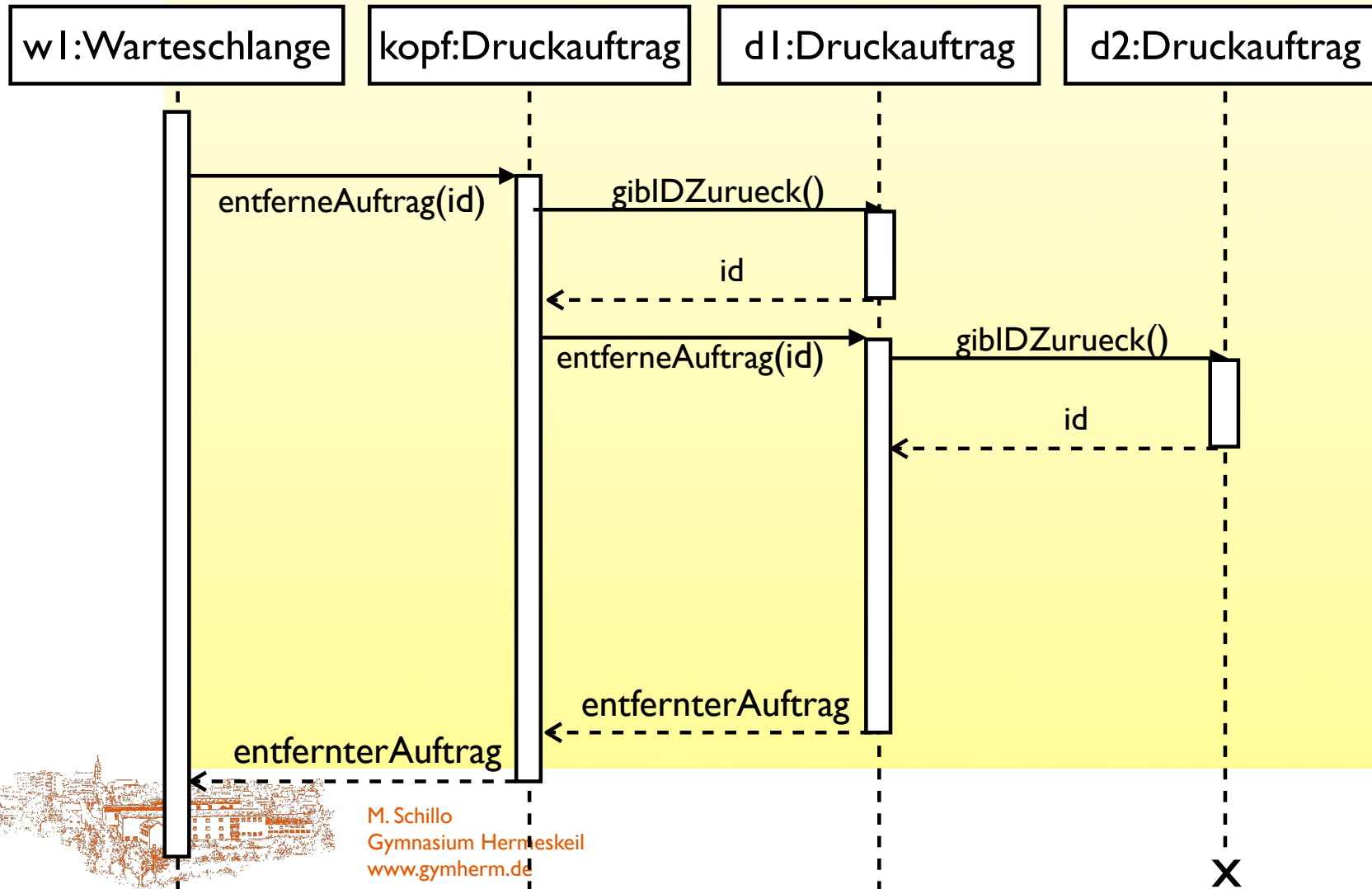
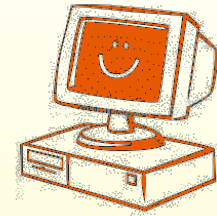
Übung

Auftrag löschen und Priorität ändern

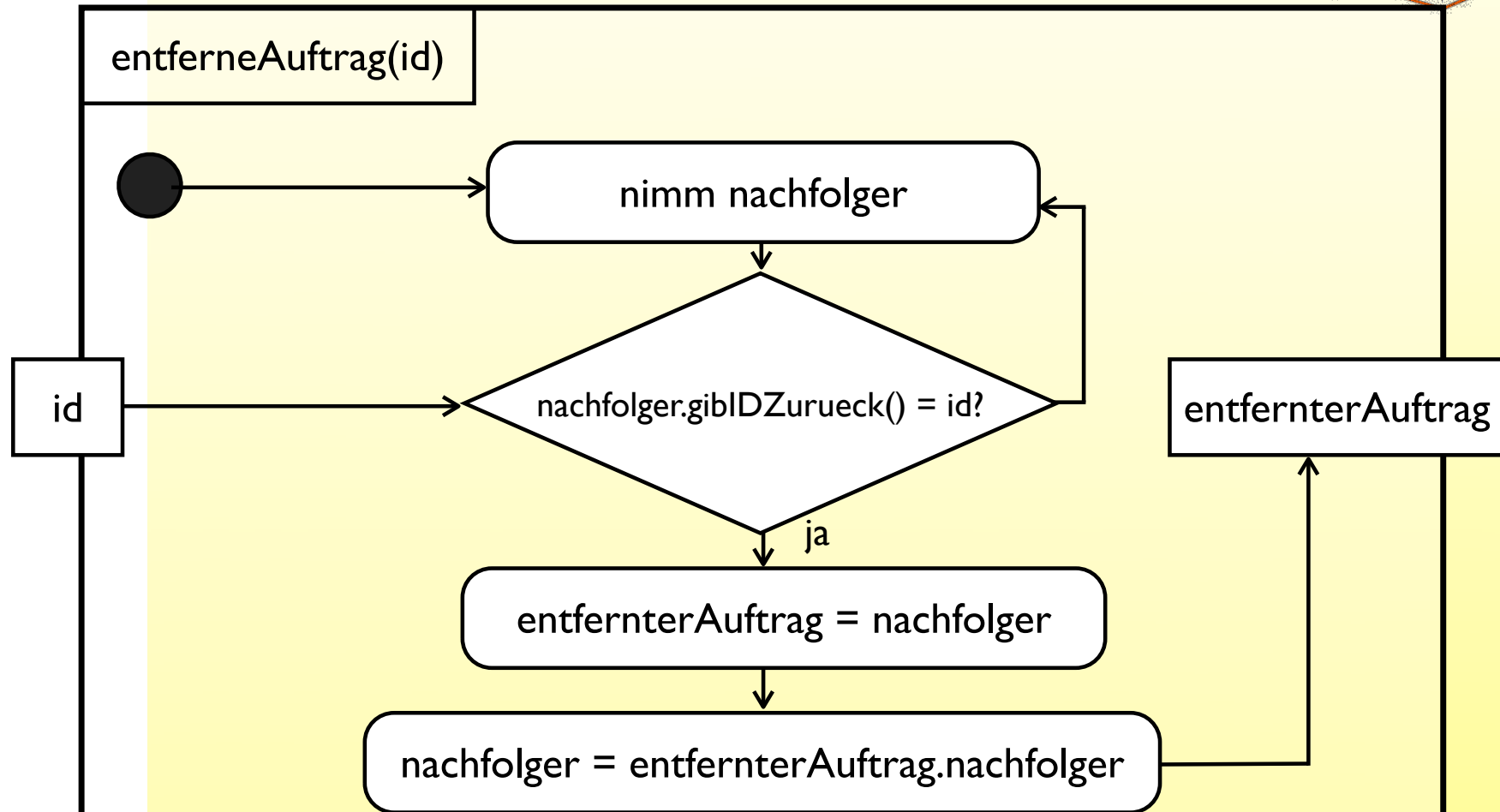
Sequenzdiagramm: löscheAuftrag(id) (I)



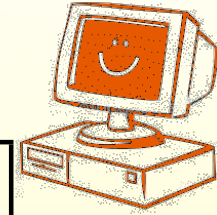
Sequenzdiagramm: entferneAuftrag(id)



Aktivitätsdiagramm: entferneAuftrag(id)



Code: entferneAuftrag(id)

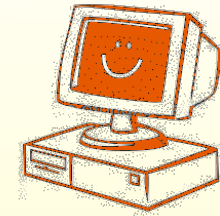


```
public class Warteschlange{  
    public void loescheAuftrag( int id ){  
        kopf.entferneAuftrag( id );  
    }  
}
```

```
public class Druckauftrag{  
    public Druckauftrag entferneAuftrag( int id ){  
        if ( nachfolger.gibIDZurueck() == id ){  
            Druckauftrag auftrag = nachfolger;  
            nachfolger = nachfolger.gibNachfolgerZurueck();  
            return auftrag;  
        }  
        return nachfolger.entferneAuftrag( id );  
    }  
}
```



Allgemeinbildende Aspekte



- Beschreibung komplexer Systeme
 - Beschreibungswerkzeuge (Interaktion, Ablauf)
 - nicht reduzieren, sondern annehmen der Komplexität
 - Textverarbeitung, SchO



OOM + Textverarbeitung



Office Datei Bearbeiten Ansicht Einfügen Format Tabelle Extras Fenster Hilfe (Geladen) So 20:20 schill

Unbenannt1 - NeoOffice Writer

Standard Helvetica 12

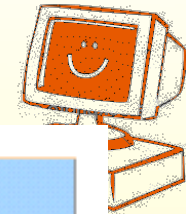
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

Lieber Kunde und Leser, falls Sie keine **Probleme** haben, diesen Blindtext schnell und zügig zu lesen, können Sie sich **glücklich** schätzen. Der verantwortliche Art Director, der Ihnen höchstwahrscheinlich gerade diesen Entwurf präsentiert, versteht sein typografisches Handwerk par excellence.

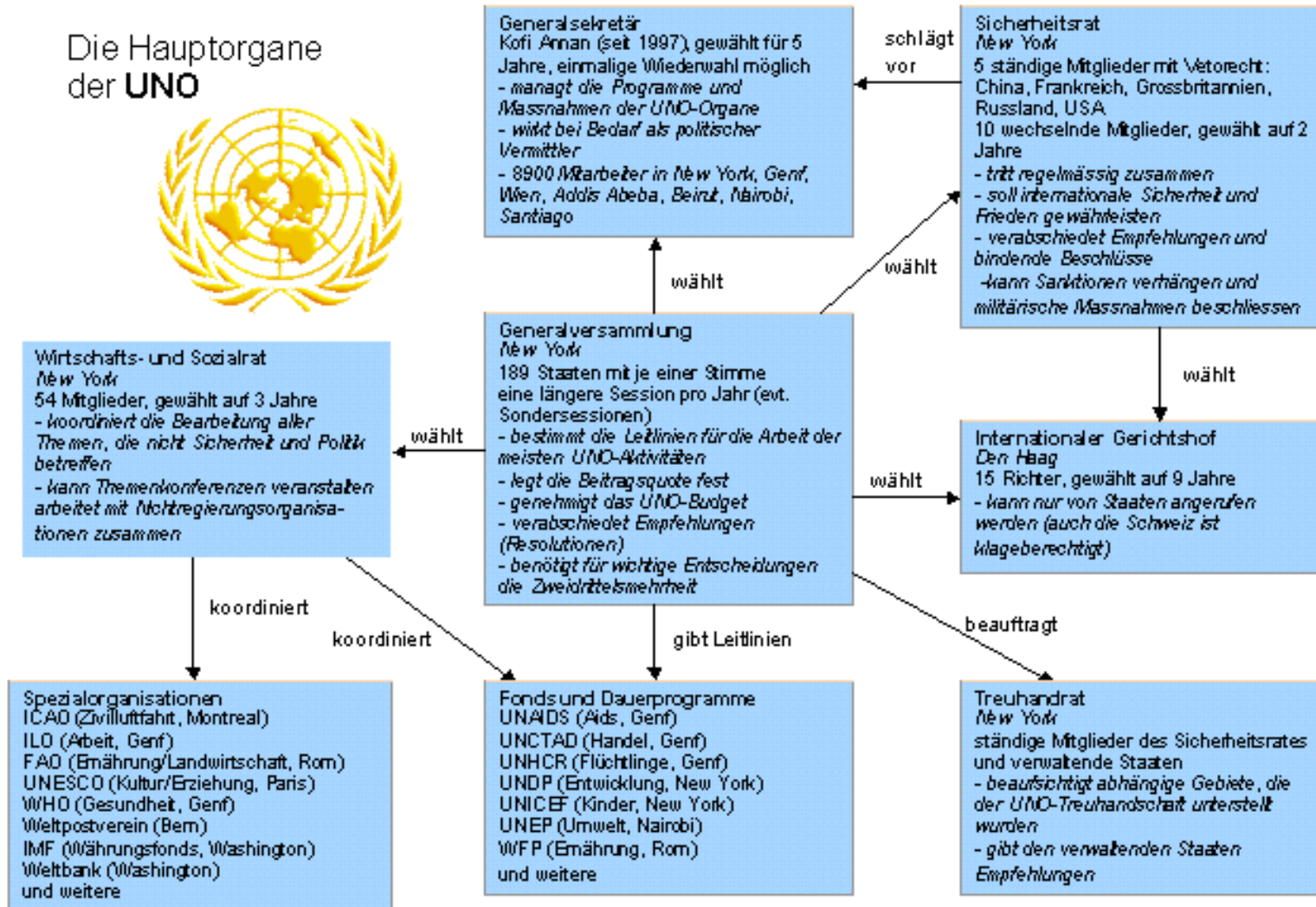
Er hat diesen Copyblock weder gestaucht, gezerrt, noch in Versalien oder gar in 6 Punkt Eurostile Outline gesetzt. Er hat ihn ganz einfach lesbar gemacht. Offenbar sogar ziemlich gut, sonst hätten Sie wohl schon einige Zeilen zuvor die Leselust verloren. Beachten Sie nur die Zeilenbreite, die er gewählt hat. Sie ist weder zu lang noch zu kurz gewählt. Der dazugehörige Zeilenabstand ist ideal. Ihre Augen haben keinerlei Probleme, vom Ende einer Zeile in die nächste zu gelangen. Um einen solchen Art Director kann man Sie beneiden. Er nutzt den ihm gewährten gestalterischen Freiraum nicht, um sich selbst darzustellen, sondern Sie. Er weiß, dass es Wichtiges über Ihr Unternehmen oder Produkt zu sagen gibt. Und dem räumt er großzügig Platz ein.

Dieser Mensch hat zweifelsohne nicht am Mäschäützets Inschtitut of Gräfick Ahts studiert. Er besitzt keine Bücher von Neville Brody oder April Greiman, und wenn doch, ordnet er sie im Regal unter Kunst ein. Statt dessen pflegt er eine liebevoll innige Beziehung zu Büchern von Tschichold und Otl Aicher. Und: Er liest sie. Sie sollten an dieser Stelle ruhig mal zu ihm rüberlächeln. Loben Sie ihn. Laden Sie ihn zum Essen ein. Denn Sie werden sicher noch viel Freude an seiner Arbeit haben.

OOM + Politische Strukturen



Die Hauptorgane der UNO



Allgemeinbildende Aspekte



- Beschreibung komplexer Systeme
 - Beschreibungswerkzeuge (Interaktion, Ablauf)
 - nicht reduzieren, sondern annehmen der Komplexität
 - Textverarbeitung, SchO
- Perspektivenwechsel
 - Nicht Kausalbeziehung
 - vgl. Hermeneutischer Zirkel



Pause



Unterrichtsmaterial “Bankautomat” und UML

Dr. Michael Schillo
Gymnasium Hermeskeil



Gymnasium Hermeskeil
Schwerpunkte Musik
und Informatik

F:\vfb\BankautomatSwing.jfm

ECKarte	<input type="text"/>	PIN	<input type="text"/>
abzuehbender Betrag	<input type="text"/>		
	<input type="button" value="Auszahlung"/>	<input type="button" value="Kontostand"/>	
neuer Kontostand	<input type="text"/>		
Status	<input type="text"/>		



vfb\Bank.automatSwing.jfm gespeichert

Applet-Ansicht: Bank...



Bankautomat (I)



Bitte wählen Sie Ihre Karte aus.

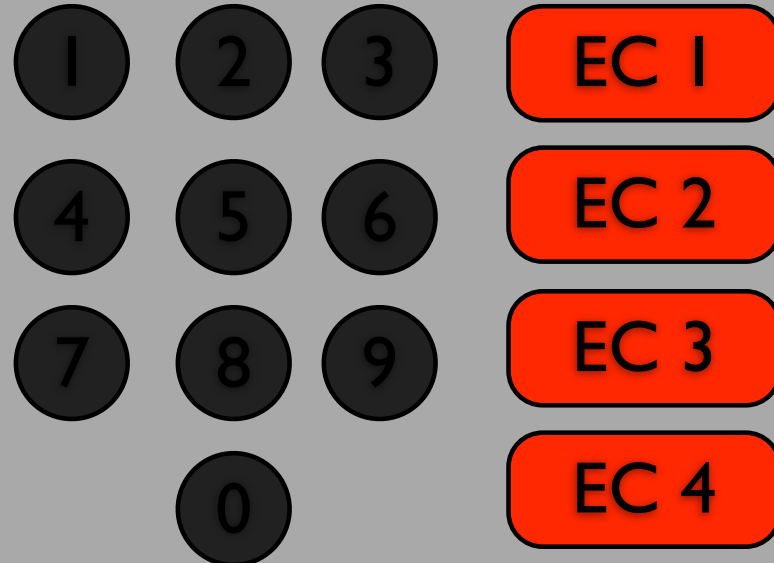
1	2	3	EC 1
4	5	6	EC 2
7	8	9	EC 3
	0		EC 4



Bankautomat (2)



Geben Sie Ihre PIN ein

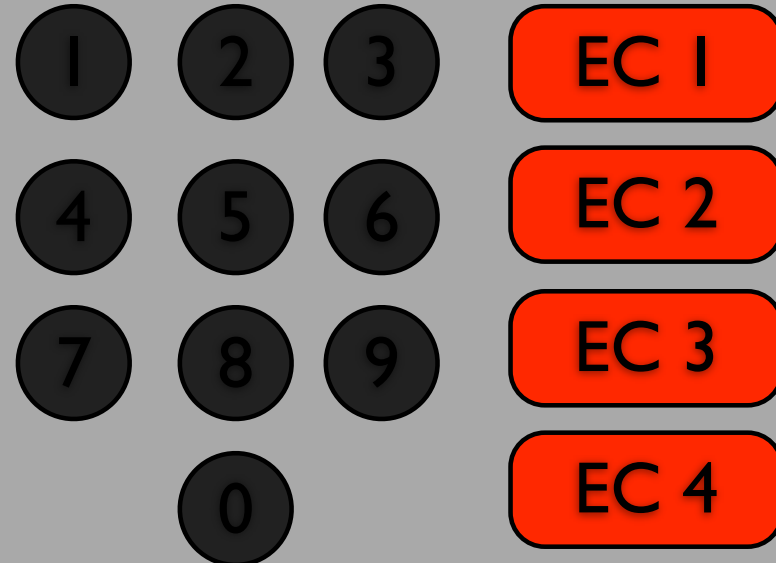


Bankautomat (3)



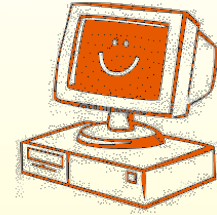
Was möchten Sie?

- (1) Kontostand einsehen
- (2) Geld abheben



AnwFallID

Aufgabe



- Fertigen Sie ein Klassendiagramm an
- Fertigen Sie Aktivitäts- und Sequenzdiagramme für folgende Abläufe an:
 - Ablaufdatum überprüfen
 - PIN prüfen
 - Saldo ausgeben
 - 50€ abheben
 - beliebigen Betrag abheben
- Dabei soll die Aktivitätsdiagramme zentrale Methoden abbilden, das Sequenzdiagramm beginnt beim Bankautomaten



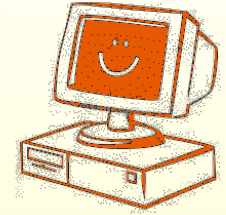


Material zum Bankautomaten



Gymnasium Hermeskeil
Schwerpunkte Musik
und Informatik

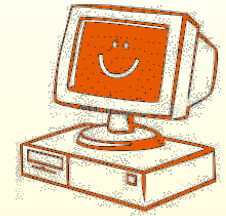
Kunde I



PIN	0815
Kartennummer	2



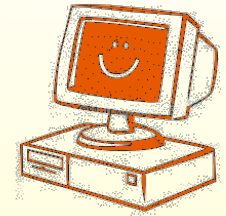
Kunde 2



PIN	
Kartenummer	



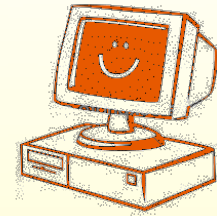
Kunde 3



PIN	
Kartennummer	



Kontoverwaltung



Die Kontoverwaltung kennt

- Konten,
- und den Sicherheitsserver

Kontoverwaltung



Sicherheitsserver

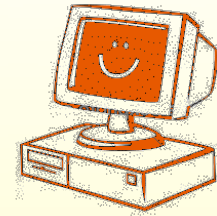


- Kennt Merkmale der EC-Karten

Sicherheitsserver



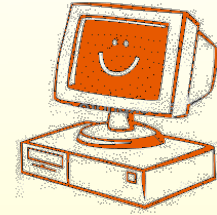
EC-Karte I



Kartenummer	I
PIN	I234
Ablaufdatum	09/2008
Kontonummer	2



EC-Karte 2



Kartennummer	2
PIN	0815
Ablaufdatum	07/2009
Kontonummer	3



EC-Karte 3



Kartenummer	3
PIN	
Ablaufdatum	
Kontonummer	I



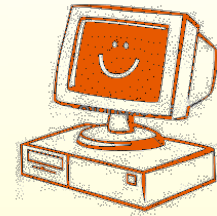
Konto I



Kontonummer	I
Kontoart	
Kontostand	
Dispositionskredit	



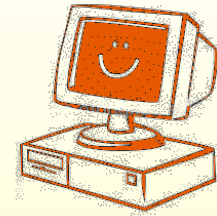
Konto 2



Kontonummer	2
Kontoart	
Kontostand	
Dispositionscredit	



Konto 3



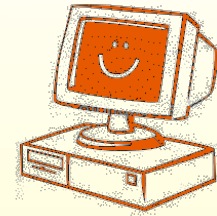
Kontonummer	3
Kontoart	Girokonto
Kontostand	500€
Dispositionscredit	1500€



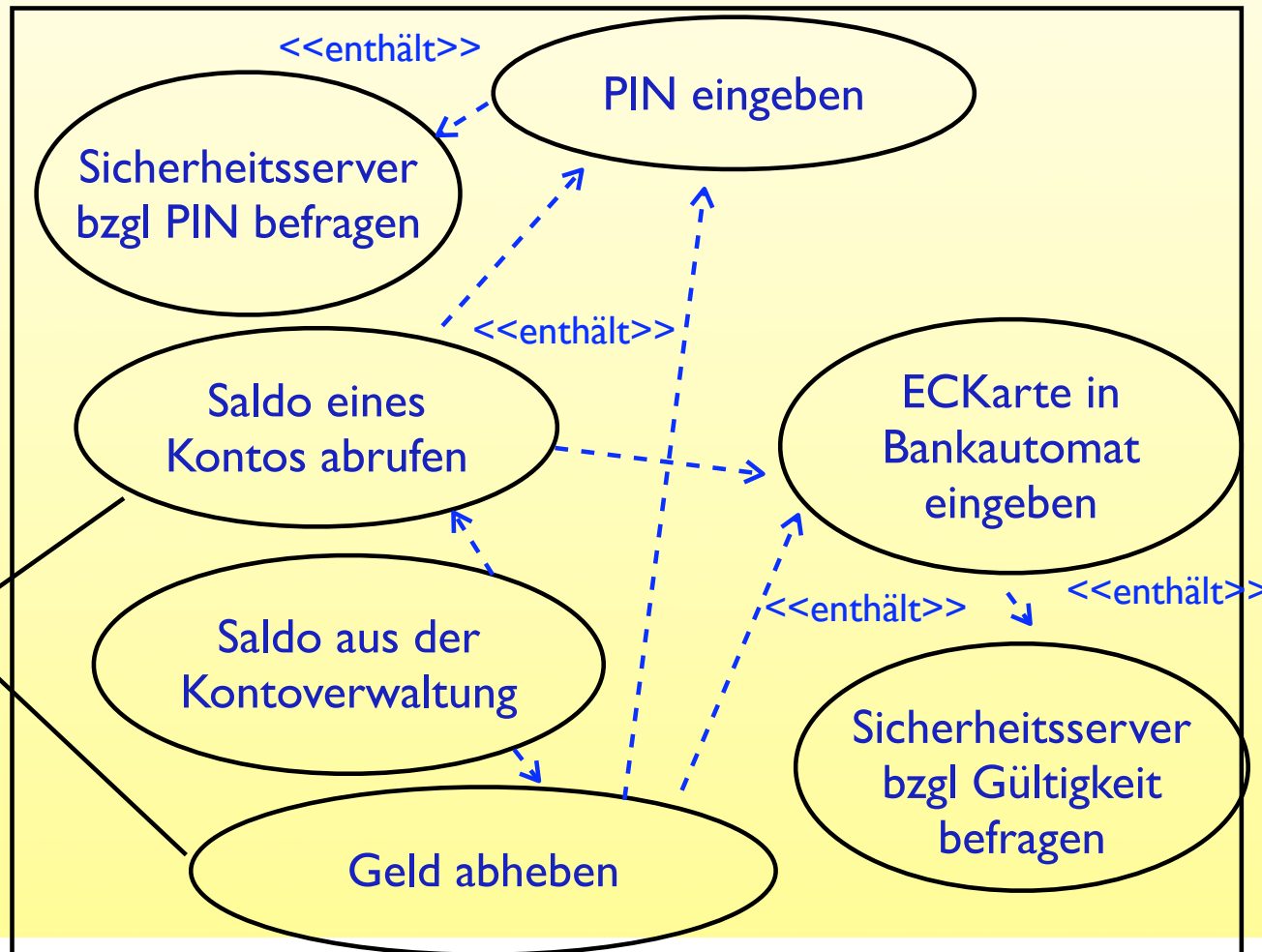
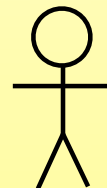
Ende

MuLö

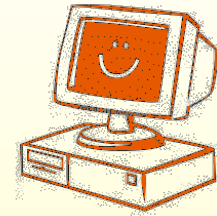
Anwendungsfall-Diagramm: Bankautomat



Kunde



Liste der "natürlichen Objekte"



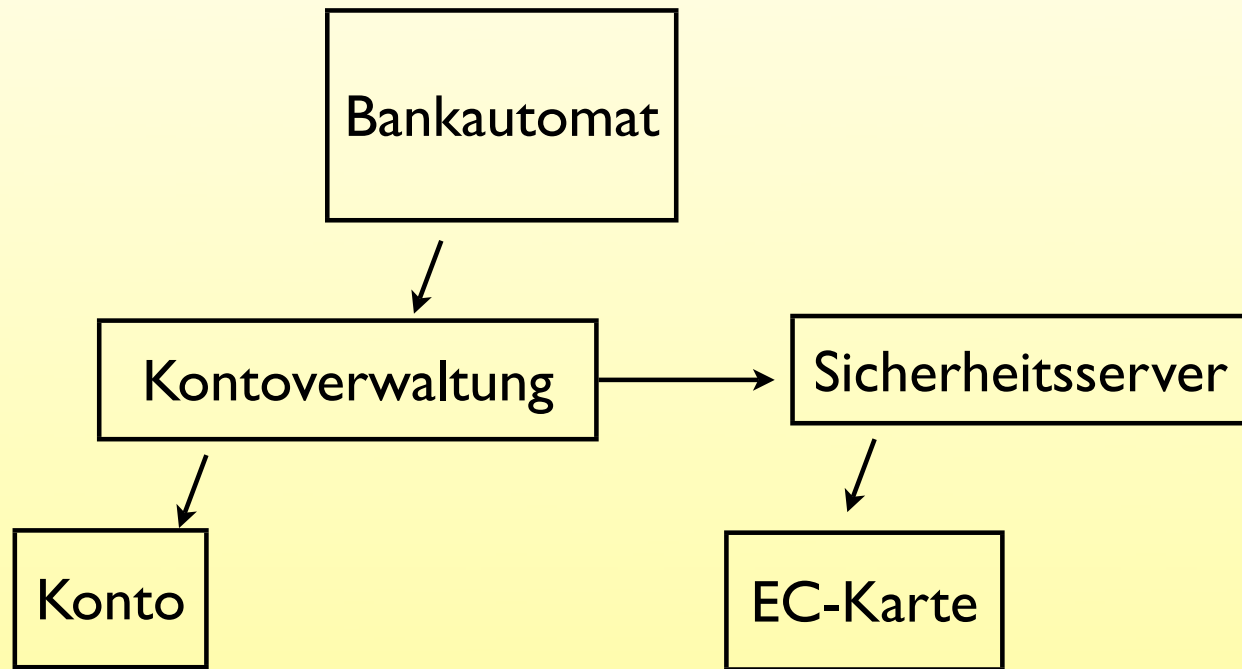
- ~~Kunde~~
- Kontoverwaltung
- Konto
- ~~Saldo~~
- Sicherheitsserver
- ~~PIN~~
- EC-Karte
- Bankautomat

~~Extern~~

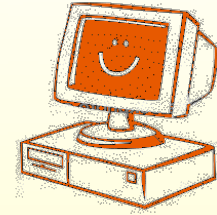
~~Elementar~~



Klassendiagramm (stark verkürzt)



Bankautomat (4)



- Der Bankautomat ist zuständig für die Interaktion mit Kunden
 - Eingabe/Ausgabe
 - Kennt die Kontoverwaltung

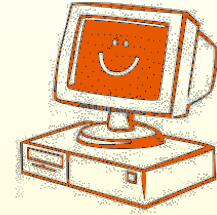
Bankautomat

kontoverwaltung: Kontoverwaltung

starteBankautomat()



Kontoverwaltung



Die Kontoverwaltung kennt

- Konten,
- den Bankautomaten
- und den Sicherheitsserver

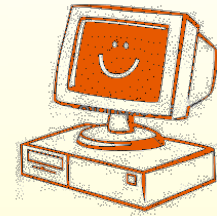
Kontoverwaltung

konto1, konto2, konto3: Konto
sicherheitsserver: Sicherheitsserver

gibKontostandZurueck()
kannGeldAusgezahltWerden
(kartennummer, betrag): boolean
istPINKorrekt(kartennummer, pin): b
erhoeheFehlversuche()
karteMussEingezogenWerden
(kartennummer): boolean



Konto

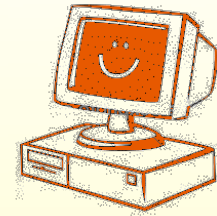


Kontonummer	
Kontoart	
Kontostand	
Dispositionscredit	

<h2>Konto</h2>
kontonummer: int kontostand: float dispositionscredit: float kontoart: int
bucheGeldAb(betrag) gibXZurueck(): float/int (4x)



Sicherheitsserver



- Kennt Merkmale der EC-Karten

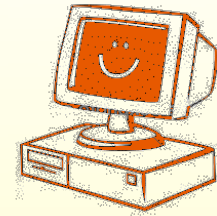
Sicherheitsserver

eckarte1, eckarte2, eckarte3: ECKarte
aktuellesJahr: int
aktuellerMonat: int

gibKontonummerZurueck(kartenummer)
istPINKorrekt(kartenummer, pin)
erhoeheFehlversuche(kartenummer)
kartelstAbgelaufen(kartenummer)



EC-Karte



Kartenummer	
PIN	
Ablaufdatum	

EC-Karte

kartenummer: int
pin: int
ablaufdatumJ: int
ablaufdatumM: int
kontonummer: int
anzahlFehlversuche: int

erhoeheFehlversuche()
gibXZurueck() 3x int

