
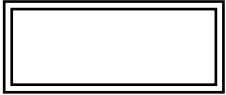
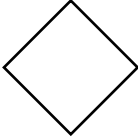
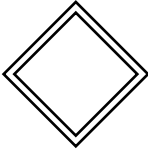


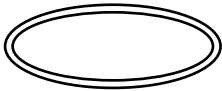
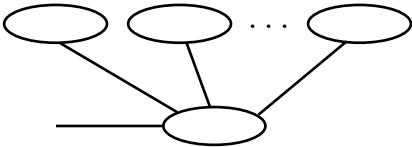

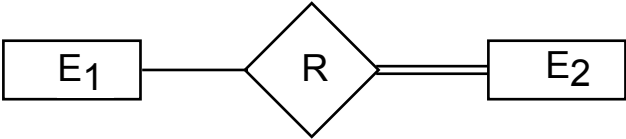

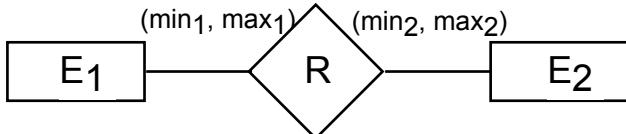
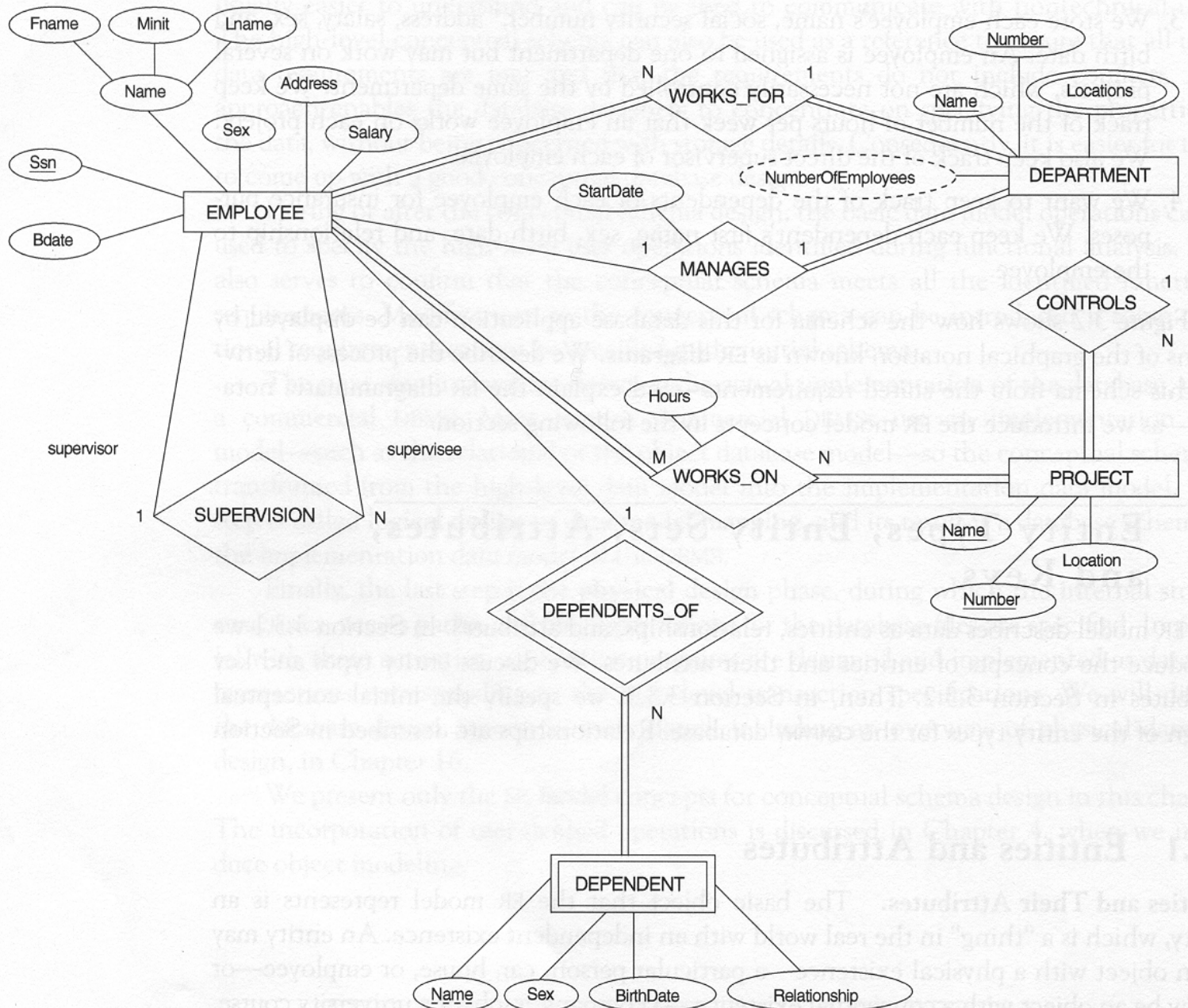


<u>Symbol</u>	<u>Bedeutung</u>
	Entity-Typ
	Abhängiger Entity-Typ
	Beziehungs-Typ (Relationship-Type)
	Identifizierender Beziehungs-Typ
	Attribut
	Schlüssel-Attribut
	Attribut, das mehrere Werte annehmen kann
	Zusammengesetztes Attribut
	Abgeleitetes Attribut
	Jedes Entity des Entity-Typs E2 muß eine Beziehung R haben
	Kardinalität
	Kardinalität mit struktureller Beschränkung

## Symbole zur Darstellung von ER-Diagrammen

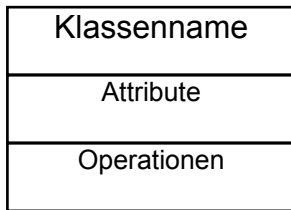
<b>Begriff</b>	<b>Informale Bedeutung</b>	<b>ER</b>	<b>EER</b>
Entity	zu repräsentierende Informationseinheit	x	x
Entity-Typ	Gruppierung von Entities mit gleichen Eigenschaften	x	x
Beziehungstyp	Gruppierung von Beziehungen zwischen Entities	x	x
Attribut	datenwertige Eigenschaft eines Entities oder einer Beziehung	x	x
funktionale Beziehung	Beziehungstyp mit Funktionseigenschaft	x	x
Schlüssel	identifizierende Eigenschaft von Entities	x	x
abhängige Entities	Entities, die nur abhängig von anderen Entities existieren können	x	
IST-Beziehung	Spezialisierung von Entity-Typen	x	
Kardinalitäten	Einschränkung von Beziehungstypen bezüglich der mehrfachen Teilnahme von Entities an der Beziehung	x	x
Optionalität	Attribute oder funktionale Beziehungen als partielle Funktionen	x	x
strukturierte Attribute	zusammengesetzte Attributwerte		x
abgeleitete Attribute	durch eine Berechnungsvorschrift berechnete Attributwerte		x
Spezialisierung	Verfeinerung eines Entity-Typen zu einem spezielleren Entity-Typ		x
Generalisierung	Zusammenfassung von Entity-Typen zu einem allgemeineren Entity-Typ		x
Partitionierung	mehrere disjunkte Spezialisierungen eines Entity-Typen		x
Aggregation	Zusammensetzung von Entities aus anderen Entities		x
objektwertige Attribute	Attribute, die Entity-Typen als Wertebereich haben		x

## Modellierungs-Elemente des ER- und des EER-Modells



## ER-Diagramm "Unternehmensdatenbank"

## Symbol



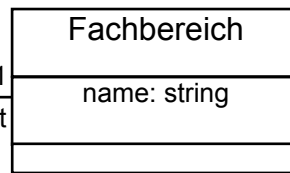
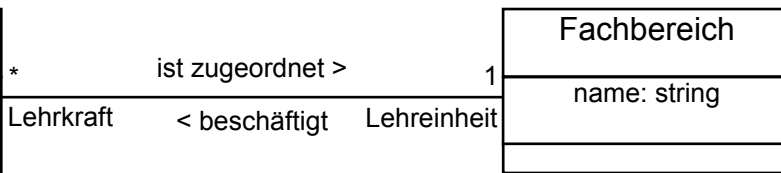
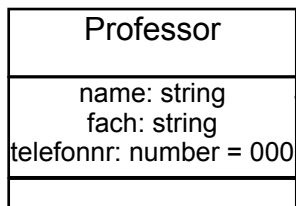
## Bedeutung

Klasse  
(*Klassenname* kursiv:  
abstrakte Klasse)

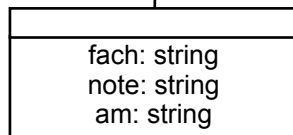
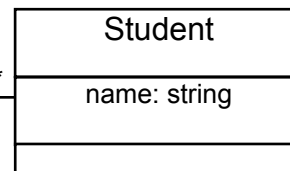
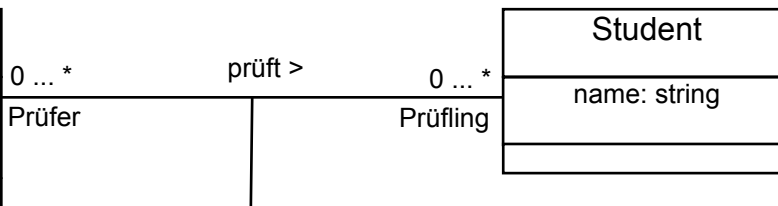
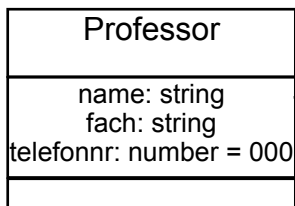
name: typ = initialer Wert { Zusicherung }

z.B.: Alter: integer = 0 { Alter > 0 and Alter < 125 }

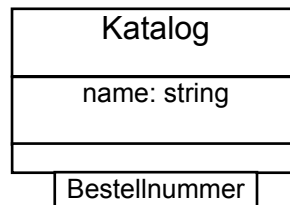
Angabe von Attributen



Beziehung  
(Assoziation)  
mit  
Beziehungsbezeichnern,  
Rollen,  
Kardinalitäten



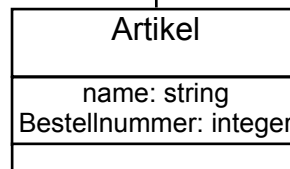
Beziehung  
mit  
Attributen



Bestellnummer

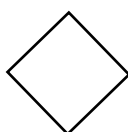
\*

\*



Beziehung  
mit  
Qualifikationsangabe

Beziehungsname

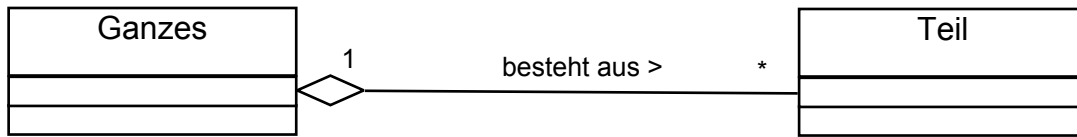


Mehrstellige Beziehung

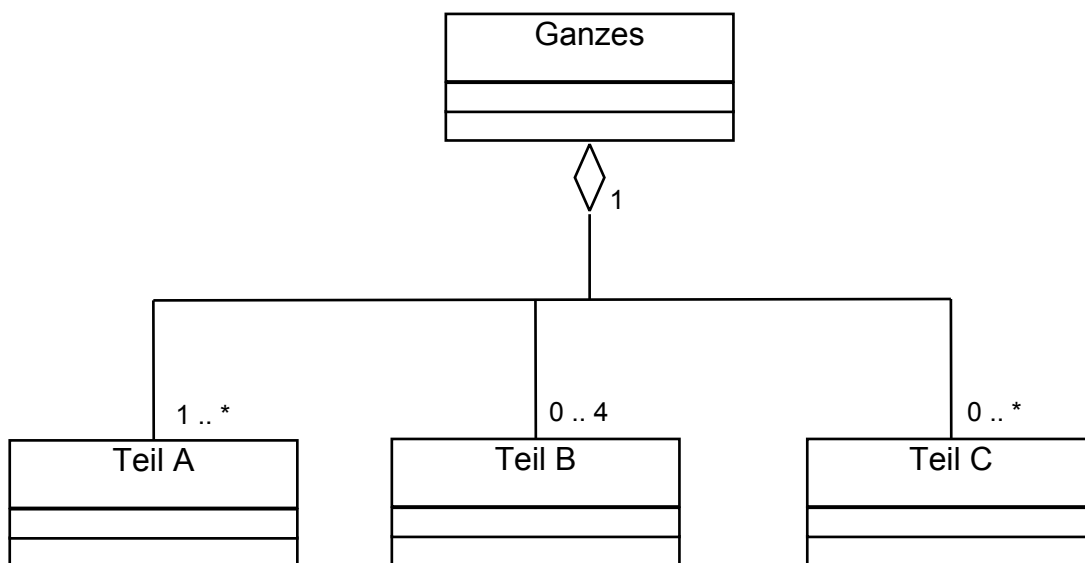
## Symbole zur Darstellung von UML-Klassendiagrammen 1(3)

## Symbol

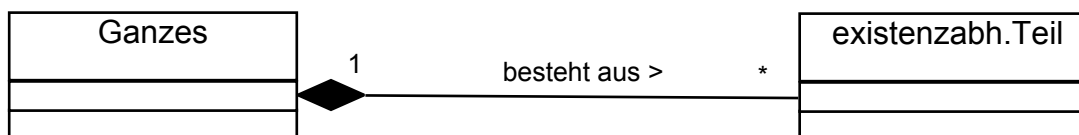
## Bedeutung



Aggregation



Aggregation

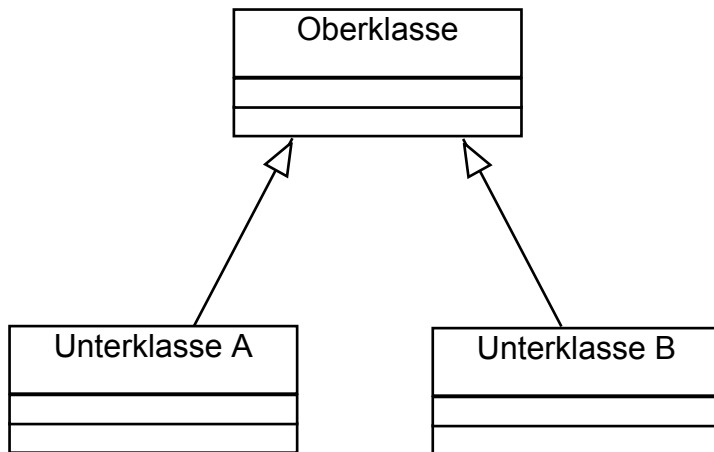


Komposition  
(Aggregation mit ab-  
hängigen Objekten)

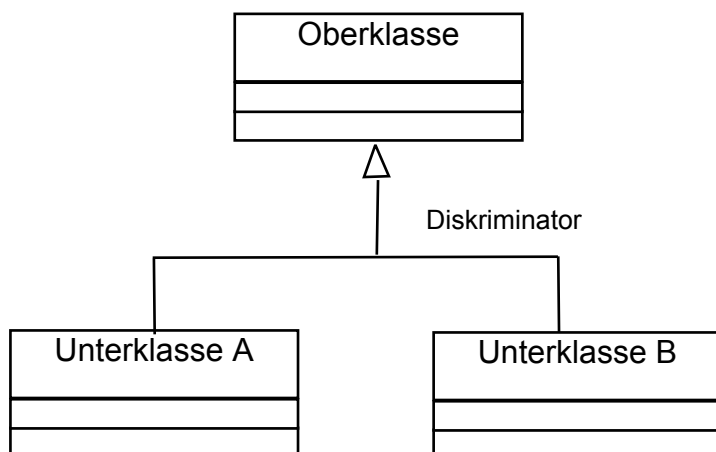
## Symbole zur Darstellung von UML-Klassendiagrammen 2(3)

## Symbol

## Bedeutung



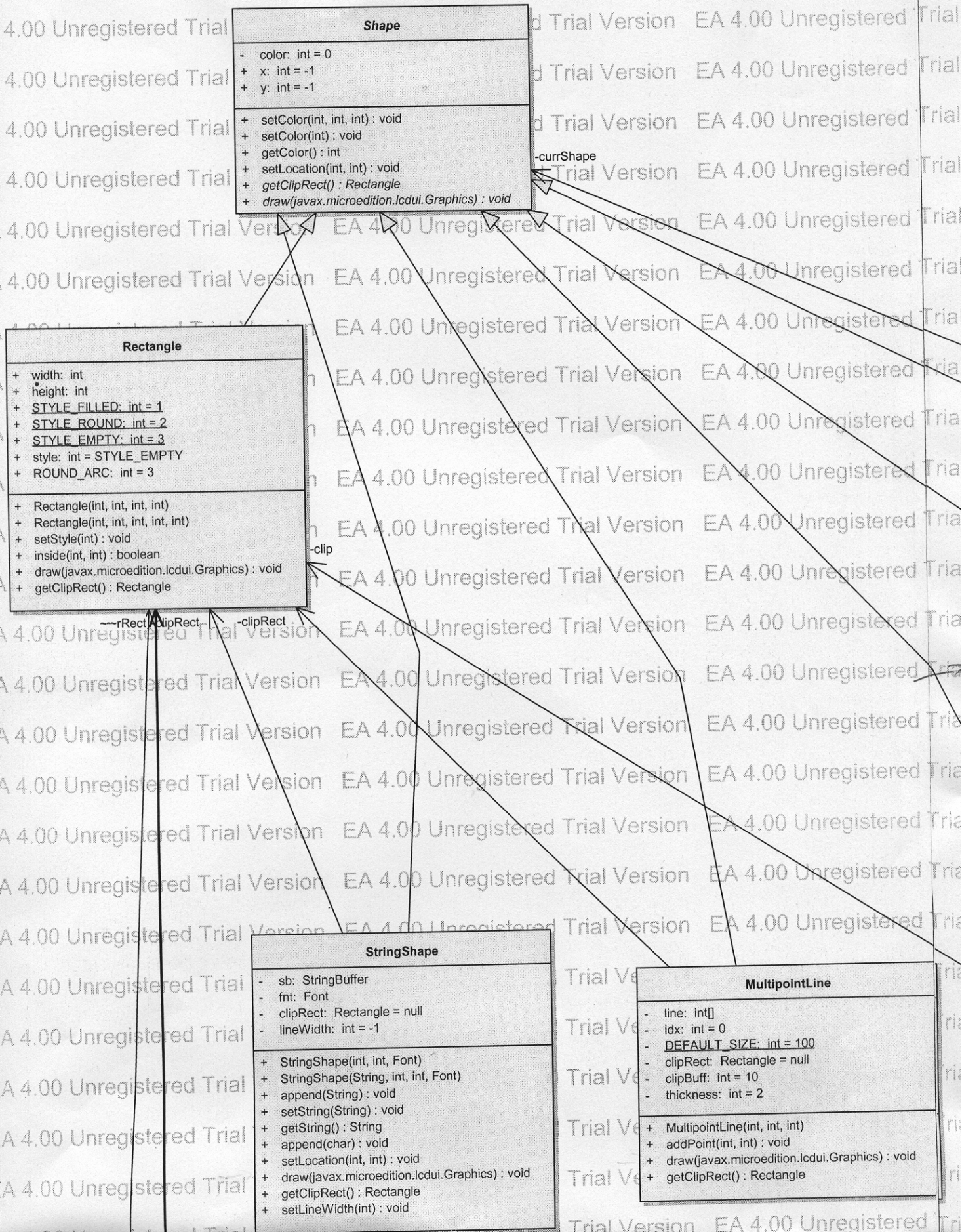
Spezialisierung



Spezialisierung mit  
Diskriminator

## Symbole zur Darstellung von UML-Klassendiagrammen 3(3)

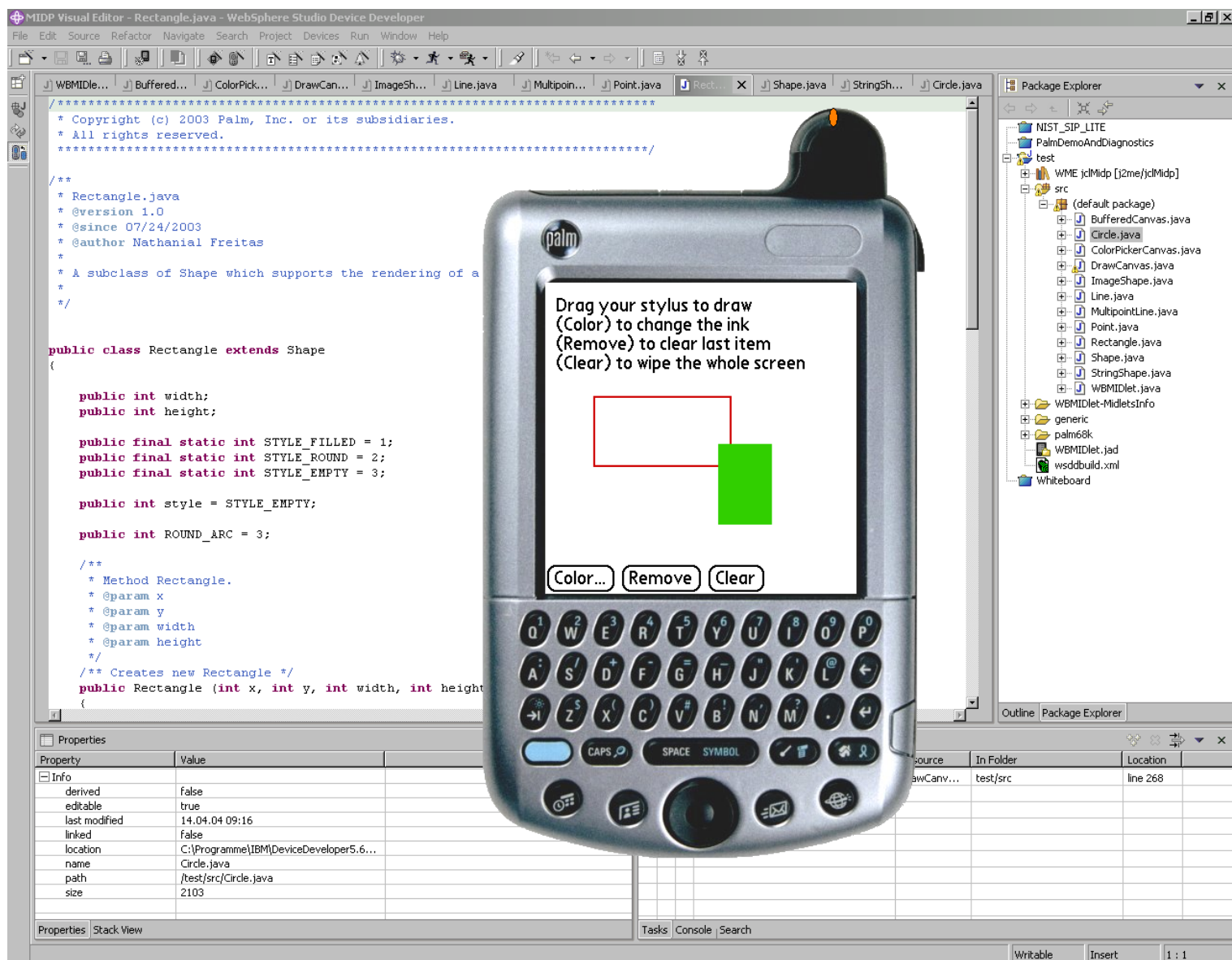




## UML-Klassendiagramm

### Beispiel: Elektronisches "Whiteboard"





## UML-Klassendiagramm

### Beispiel: Umsetzung elektronisches "Whiteboard"



MIDP Visual Editor - Shape.java - WebSphere Studio Device Developer

File Edit Source Refactor Navigate Search Project Devices Run Window Help

WBMDlet.java \*Shape.java X \*Rectangle.java

```

* The Shape abstract class which handle storage of shape color information,
* but requires the implementation of clipping and rendering duties by subclasses.
*
*/
public abstract class Shape {
    private int color = 0;
    public int x = -1;
    public int y = -1;
    /**Method setColor.
     * @param r
     * @param g
     * @param b */
    public void setColor (int r, int g, int b)
    {
        color = ((r & 0xFF) << 16) | ((g & 0xFF) << 8) | ((b & 0xFF) << 0);
    }
    /**Method setColor.
     * @param color */
    public void setColor (int color)
    {
        this.color = color;
    }
    /**Method getColor.
     * @return int */
    public int getColor ()
    {
        return color;
    }
    /**Method setLocation.
     * @param x
     * @param y */
    public void setLocation (int x, int y)
    {
        this.x = x;
        this.y = y;
    }
    /**Method getClipRect.
     * @return Rectangle*/
    public abstract Rectangle getClipRect ();
    /**Method draw.
     * @param g */
    public abstract void draw (javax.microedition.lcdui.Graphics g);
}

```

Properties

Property	Value
Info	
derived	false
editable	true
last modified	13.04.04 14:32
linked	false
location	C:\Programme\IBM\DeviceDeveloper5.6\...
name	Shape.java

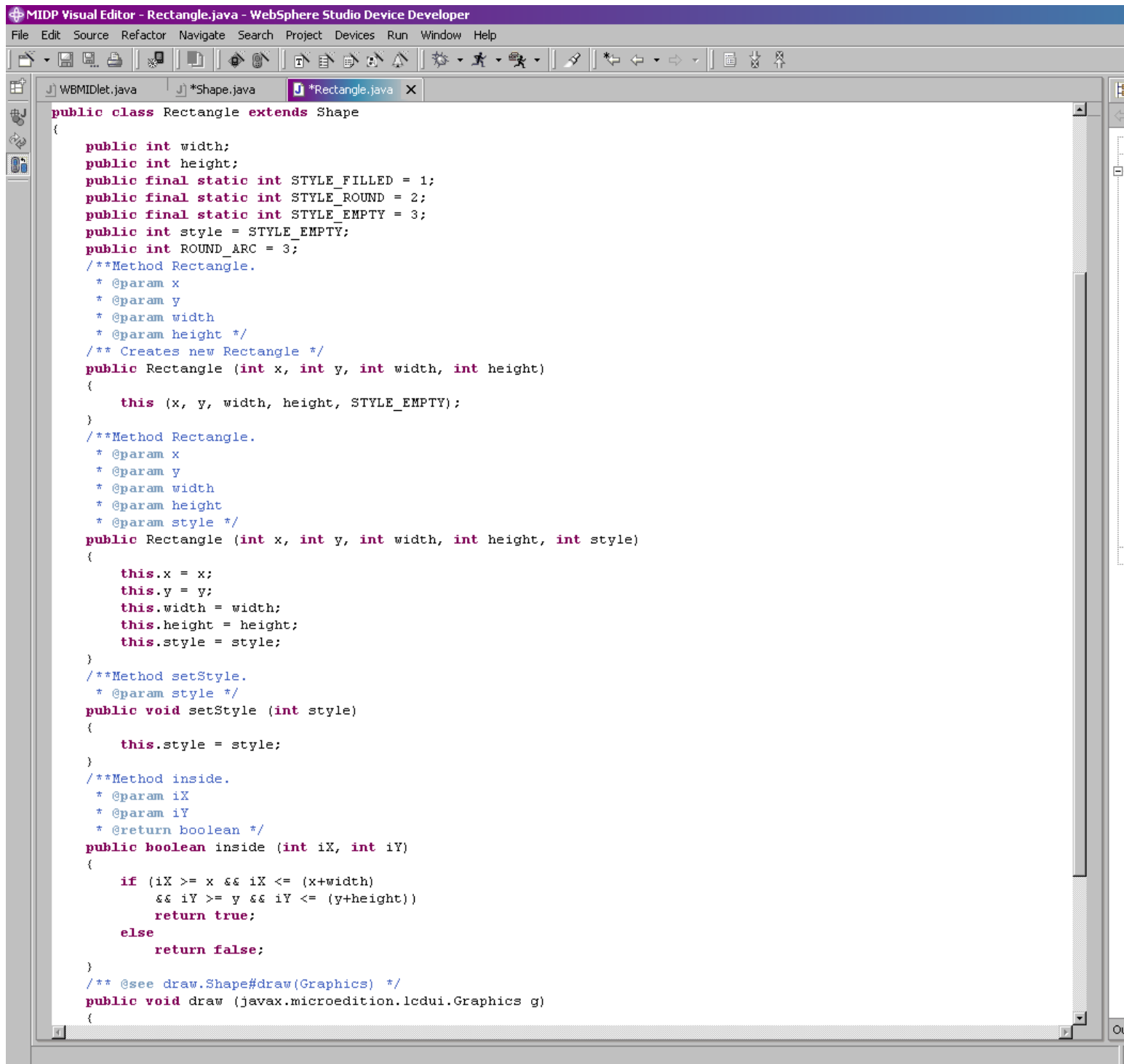
Tasks (1 item)

✓	!	Description	Resource	In Folder
		The assignment to variable currShape ...	DrawCanv...	test/src

Properties Stack View Tasks Console Search

## UML-Klassendiagramm

### Beispiel: Umsetzung elektronisches "Whiteboard"



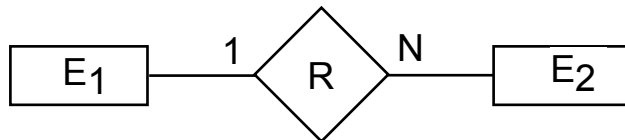
```
public class Rectangle extends Shape
{
    public int width;
    public int height;
    public final static int STYLE_FILLED = 1;
    public final static int STYLE_ROUND = 2;
    public final static int STYLE_EMPTY = 3;
    public int style = STYLE_EMPTY;
    public int ROUND_ARC = 3;
    /**Method Rectangle.
     * @param x
     * @param y
     * @param width
     * @param height */
    /** Creates new Rectangle */
    public Rectangle (int x, int y, int width, int height)
    {
        this (x, y, width, height, STYLE_EMPTY);
    }
    /**Method Rectangle.
     * @param x
     * @param y
     * @param width
     * @param height
     * @param style */
    public Rectangle (int x, int y, int width, int height, int style)
    {
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
        this.style = style;
    }
    /**Method setStyle.
     * @param style */
    public void setStyle (int style)
    {
        this.style = style;
    }
    /**Method inside.
     * @param iX
     * @param iY
     * @return boolean */
    public boolean inside (int iX, int iY)
    {
        if (iX >= x && iX <= (x+width)
            && iY >= y && iY <= (y+height))
            return true;
        else
            return false;
    }
    /** @see draw.Shape#draw(Graphics) */
    public void draw (javax.microedition.lcdui.Graphics g)
    {

```

## UML-Klassendiagramm

### Beispiel: Umsetzung elektronisches "Whiteboard"

## ER-Modell



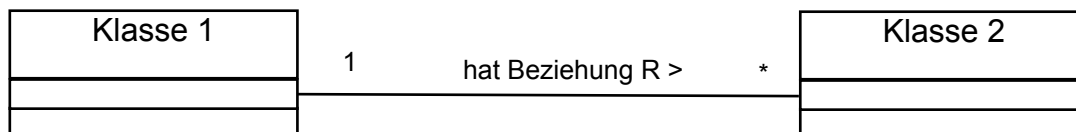
Kardinalität

Ein  $E_1$ -Entity kann mit  $N$   $E_2$ -Entities in Beziehung stehen.

Das heißt :

Ein  $E_1$ -Entity kann an  $N$  Beziehungen des Beziehungstyps  $R$  beteiligt sein.  
Ein  $E_2$ -Entity kann nur an einer Beziehung des Beziehungstyps  $R$  beteiligt sein.

## UML-Klassendiagramm



Ein Objekt der Klasse 1 kann mit beliebig vielen Objekten der Klasse 2 in Beziehung stehen.

Mögliche Kardinalitätsangaben:

1	genau 1
0,1	0 oder 1
0 .. 3	0 bis 3
0 .. *	beliebig viele (Standardannahme)
1 .. *	nicht-optionale Beziehung
0 .. 2, 6, 10 .. *	zusammengesetzte Angaben

## Zur Angabe von Kardinalitäten (Standard ER-Modell, UML-Klassendiagramm)

<b>Begriff</b>	<b>Informale Bedeutung</b>
Attribut	Spalte einer Tabelle
Wertebereich	mögliche Werte eines Attributs (auch Domäne)
Attributwert	Element eines Wertebereichs
Relationenschema	Menge von Attributen
Relation	Menge von Zeilen einer Tabelle
Tupel	Zeile einer Tabelle
Datenbankschema	Menge von Relationenschemata
Datenbank	Menge von Relationen (Basisrelationen)
Schlüssel	minimale Menge von Attributen, deren Werte ein Tupel einer Tabelle eindeutig identifizieren
Primärschlüssel	ein beim Datenbankentwurf ausgezeichneteter Schlüssel
Fremdschlüssel	Attributmenge, die in einer anderen Relation Schlüssel ist
Fremdschlüsselbedingung	alle Attributwerte des Fremdschlüssels tauchen in der anderen Relation als Werte des Schlüssels auf

## Begriffe des Relationenmodells



```

<!-- Buch-DTD -->
<!element Buch (ISBN, Titel, Verlagsname,
                Autor+, Stichwort*, Version*,
                Zusammenfassung, Buchtext?)>
<!element ISBN (#pcdata)>
<!element Titel (Haupttitel, Untertitel?)>
<!element Haupttitel (#pcdata)>
<!element Untertitel (#pcdata)>
<!element Verlagsname (#pcdata)>
<!element Autor (#pcdata)>
<!element Stichwort (#pcdata)>
<!element Version (Auflage, Jahr, Seiten, Preis)>
<!element Auflage (#pcdata)>
<!element Jahr (#pcdata)>
<!element Seiten (#pcdata)>
<!element Preis (#pcdata)>
<!element Zusammenfassung (#pcdata)>
<!element Buchtext (Kapitel*)>
<!element Kapitel (Überschrift, Textblock?, Abschnitt*)>
<!element Abschnitt ...>

```

## Beispiel für eine DTD (Document Type Definition) in XML (Extensible Markup Language)

```

<?xml version="1.0" standalone="yes"?>
<!doctype bib [
  <!element book (author+, title, year, publisher)>
  <!attlist book isbn cdata #required>
  <!element author (firstname?, lastname)>
  <!element publisher (name, address)>
  <!element name (#pcdata)>
  ...
]>

<bib>
  <book isbn="3-929821-31-1">
    <author>
      <firstname>Andreas</firstname>
      <lastname>Heuer</lastname>
    </author>
    <author>
      <firstname>Gunter</firstname>
      <lastname>Saake</lastname>
    </author>
    <title>Datenbanken: Konzepte und Sprachen</title>
    <year>2000</year>
    <publisher>
      <name>International Thomson Publishing</name>
      <address>Bonn</address>
    </publisher>
  </book>
</bib>

```

## Beispiel für ein XML-Dokument ("selbstbeschreibender Datensatz")

## **Vorgaben:**

- Relevante Entity-Typen:
  - Fakultät
  - Angestellter (Dozenten)
  - Vorlesung
  - Student
- Ein Angestellter (Dozent) kann verschiedene Vorlesungen halten.
- Eine Vorlesung wird von mehreren Studenten gehört.
- Ein Student kann verschiedene Vorlesungen hören.
- Die Vorlesungen werden durch ihre Vorlesungsnummer eindeutig identifiziert.
- Zu einer Vorlesung gehört genau eine schriftliche oder mündliche Prüfung.

## **Aufgabenstellung:**

- ER-Diagramm daraus erstellen
- Beispielhaft Attribute für die verschiedenen Entity-Typen angeben

## **Übungsaufgabe**