
II.2. Objekte, Klassen und Methoden

- 1. Grundzüge der Objektorientierung
- 2. Methoden, Unterprogramme und Parameter
- 3. Datenabstraktion
- 4. Konstruktoren
- 5. Vordefinierte Klassen

Selektoren

```
public class Rechteck {  
    //Objektattribute  
    double laenge, breite;    int strichstaerke;  
  
    //Selektoren  
    public void setLaenge (double l) {  
        laenge = l; }  
    public double getLaenge () {  
        return laenge; }  
    ...  
}
```

```
Rechteck r, s;
```

```
...
```

```
r.laenge = s.laenge + 2;
```

```
r.setLaenge (s.getLaenge() + 2);
```



Selektoren

```
public class Rechteck {  
    //Objektattribute  
    double flaeche, breite;    int strichstaerke;  
    //Selektoren  
    public void setLaenge (double l) {  
        flaeche = l * breite; }  
    public double getLaenge () {  
        return flaeche / breite; }  
    ...  
}
```

```
Rechteck r, s;
```

```
...
```

```
r.setLaenge (s.getLaenge() + 2);
```

Zugriffsspezifikationen

Einschränkung des Zugriffs auf Attribute und Methoden:

- `private:`

Komponente nur innerhalb der Klasse bekannt

- **kein Schlüsselwort:**

Komponente nur innerhalb des Pakets bekannt

- `public:`

Komponente überall bekannt

Zugriffsspezifikationen

```
public class Rechteck {  
    //Objektattribute  
    private double laenge,breite; private int strichstaerke;  
    //Selektoren  
    public void setLaenge (double l) {  
        laenge = l; }  
    public double getLaenge () {  
        return laenge; }  
    ...  
}
```

```
Rechteck r, s;
```

```
...
```

```
r.laenge = s.laenge + 2;
```

nicht mehr
möglich!

```
r.setLaenge (s.getLaenge() + 2);
```

Geheimnisprinzip (Information Hiding)

■ Client / Server - Prinzip:

- Anbieter publiziert Katalog der Dienstleistungen als öffentliche Schnittstelle.
- Kunde interessiert nur die Schnittstelle, nicht *wie* die Leistung erbracht wird.
- Bsp: API der Java-Bibliotheken

■ Abstrakter Datentyp:

- Daten (*Attribute*) und die darauf ausführbaren Operationen (*Methoden*)
- abstrakte Schnittstellendefinition nach außen (konkrete Realisierung und Implementierung bleibt verborgen)
- *Datenabstraktion* und *Datenkapselung*

■ Vorteile:

- besseres Verständnis
- leichtere Änderbarkeit
- bessere Modularisierung

Schnittstellendokumentation

Firefox

file:///Z:/Lehre/Prog11/Programme/II23_Datenabstraktion/doc/Rechteck.html

Rechteck

Class Rechteck

java.lang.Object
Rechteck

```
public class Rechteck  
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description
Rechteck ()

Method Summary

Methods

Modifier and Type	Method and Description
double	getLaenge()
void	setLaenge(double l)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Rechteck

```
public Rechteck ()
```

Method Detail

setLaenge

```
public void setLaenge(double l)
```

getLaenge

```
public double getLaenge ()
```

Methods

Modifier and Type	Method and Description
double	getLaenge ()
void	setLaenge (double l)

Method Detail

setLaenge

```
public void setLaenge (double l)
```

getLaenge

```
public double getLaenge ()
```

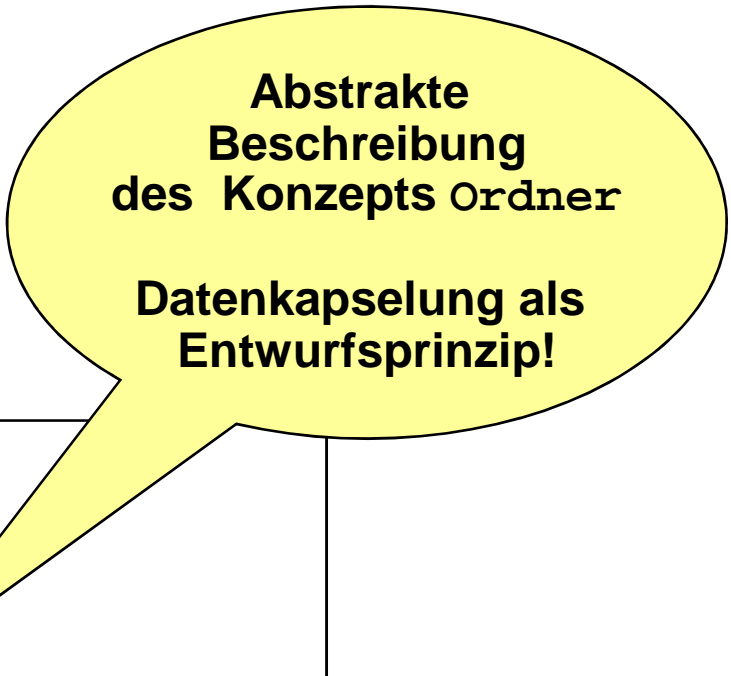
Beispiel: Datentyp Ordner

■ "Ordner " als Konzept

- *enthält* Texte
- Texte können *abgelegt* und *entnommen* werden
- ein Ordner kann *beschriftet* werden
- ein Ordner kann *leer* oder *voll* sein

■ Schnittstelle:

```
void legeTextAb (String t)
String entnehmeText ()
boolean istVoll ()
boolean istLeer ()
void beschrifte (String t)
String liesBeschriftung ()
```



Abstrakte
Beschreibung
des Konzepts Ordner

Datenkapselung als
Entwurfsprinzip!

Verwendung des Datentyps Ordner

```
Ordner o = new Ordner ();

o.beschrifte ("Kleine Gedichte");

if (o.istVoll())
    System.out.println("Ordner ist bereits voll");
else o.legeTextAb ("Herr Ribeck auf Ribeck ...");

if (o.istVoll())
    System.out.println("Ordner ist bereits voll");
else o.legeTextAb ("Von drauss vom Walde komm ich her ...");

System.out.println (o.liesBeschriftung ());
System.out.println ("-----");

if (!o.istLeer()) System.out.println (o.entnehmeText ());
if (!o.istLeer()) System.out.println (o.entnehmeText ());
```

Implementierung des Datentyps Ordner

```
/** Datentyp Ordner zur Speicherung von Texten
 * @author Juergen Giesl
 */
```

```
public class Ordner {
```

```
    private static final int maxTexte = 20;
```

```
    private String [] ordnerInhalt = new String [maxTexte];
```

```
    private int anzahlTexte = 0;
```

```
    private String beschriftung = "";
```

Es wird ein Array verwendet

```
/** @return true, falls der Ordner voll ist, sonst false
```

```
 */
```

```
public boolean istVoll () {
```

```
    return anzahlTexte == maxTexte;
```

```
}
```

```
/** @return true, falls der Ordner leer ist, sonst false
```

```
 */
```

```
public boolean istLeer () {
```

```
    return anzahlTexte == 0;}
```

Implementierung des Datentyps Ordner

```
/** @param t Text, der vorne im Ordner abgelegt wird */
public void legeTextAb (String t) {
    ordnerInhalt [anzahlTexte] = t;
    anzahlTexte ++;
}

/** Liest zuletzt eingegebenen Text und loescht ihn.
 * @return letzten abgelegten Text */
public String entnehmeText () {
    String t = ordnerInhalt [anzahlTexte-1];
    ordnerInhalt [anzahlTexte-1] = "";
    anzahlTexte --;
    return t;
}

/** @param t Beschriftung des Ordners */
public void beschrifte (String t) {
    beschriftung = t;
}

/** @return Beschriftung des Ordners */
public String liesBeschriftung () {
    return beschriftung;
}
}
```

Schnittstellendokumentation

Firefox

file:///Z:/Lehre/Prog11/Programn

Ordner

Constructor Summary

Constructors

Constructor and Description

Ordner()

Method Summary

Methods

Modifier and Type	Method and Description
void	beschrifte (java.lang.String t)
java.lang.String	entnehmeText ()
	Liest zuletzt eingegebenen Text und löscht ihn.
boolean	istLeer ()
boolean	istVoll ()
void	legeTextAb (java.lang.String t)
java.lang.String	liesBeschriftung ()

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString

Constructor Detail

Ordner

```
public Ordner()
```

Method Detail

istVoll

```
public boolean istVoll()
```

Returns:

true, falls der Ordner voll ist, sonst false

istLeer

```
public boolean istLeer()
```

Returns:

true, falls der Ordner leer ist, sonst false

legeTextAb

```
public void legeTextAb (java.lang.String t)
```

Parameters:

t - Text, der hinten im Ordner abgelegt wird

Method Detail

istVoll

```
public boolean istVoll()
```

Returns:

true, falls der Ordner voll ist, sonst false

istLeer

```
public boolean istLeer()
```

Returns:

true, falls der Ordner leer ist, sonst false

legeTextAb

```
public void legeTextAb (java.lang.String t)
```

Parameters:

t - Text, der hinten im Ordner abgelegt wird