

Einführung in die Informatik I
Programmieren in Java

Zentralübung am
6. November 2000

Andreas Harrer
Technische Universität München

Wintersemester 2000/2001

Übersicht

- ❖ **Organisatorisches**
- ❖ **Einführung in die Programmierung mit Java**
 - **Erstellen (Editieren) einer Java-Quelldatei**
 - ◆ **Aufbau einer Java-Klasse**
 - ◆ **Attribute und Attributvariable**
 - ◆ **Operationen bzw. Methoden in Java**
 - ◆ **Vererbung in Java**
 - **Übersetzen (Compilieren) einer Java-Quelldatei**
 - **Ausführen (Exekutieren) einer Java Class-Datei**
 - **Erzeugung von Objekten in Java**
- ❖ **Das UNIX-Dateisystem: ein weiteres Beispiel für das Composite-Pattern**
- ❖ **Fragen?**

Organisatorisches:

- ❖ **Vorlesungsfolien vom 23. und 24.Oktober sind als PDF-Dateien im WWW verfügbar**
- ❖ **Rechnereinführung gelungen?**
- ❖ **Tutorübungen ab dieser Woche in den ausgehängten Übungsräumen**
- ❖ **Bitte die Rechenberechtigungen bei der Rechnerbetriebsgruppe (Raum S 2034A) abholen**
- ❖ **Literaturhinweis: K. Blackburn, J. Lammers: Papierflugzeuge, Könenmann, 1998.**

Subaktivitäten während Implementation und Test (“edit-compile-execute cycle”)

❖ Codieren:

- Übersetzung des detaillierten Entwurfs mit Hilfe eines Editors in ein Quellprogramm in der ausgewählten Programmiersprache

❖ Editieren:

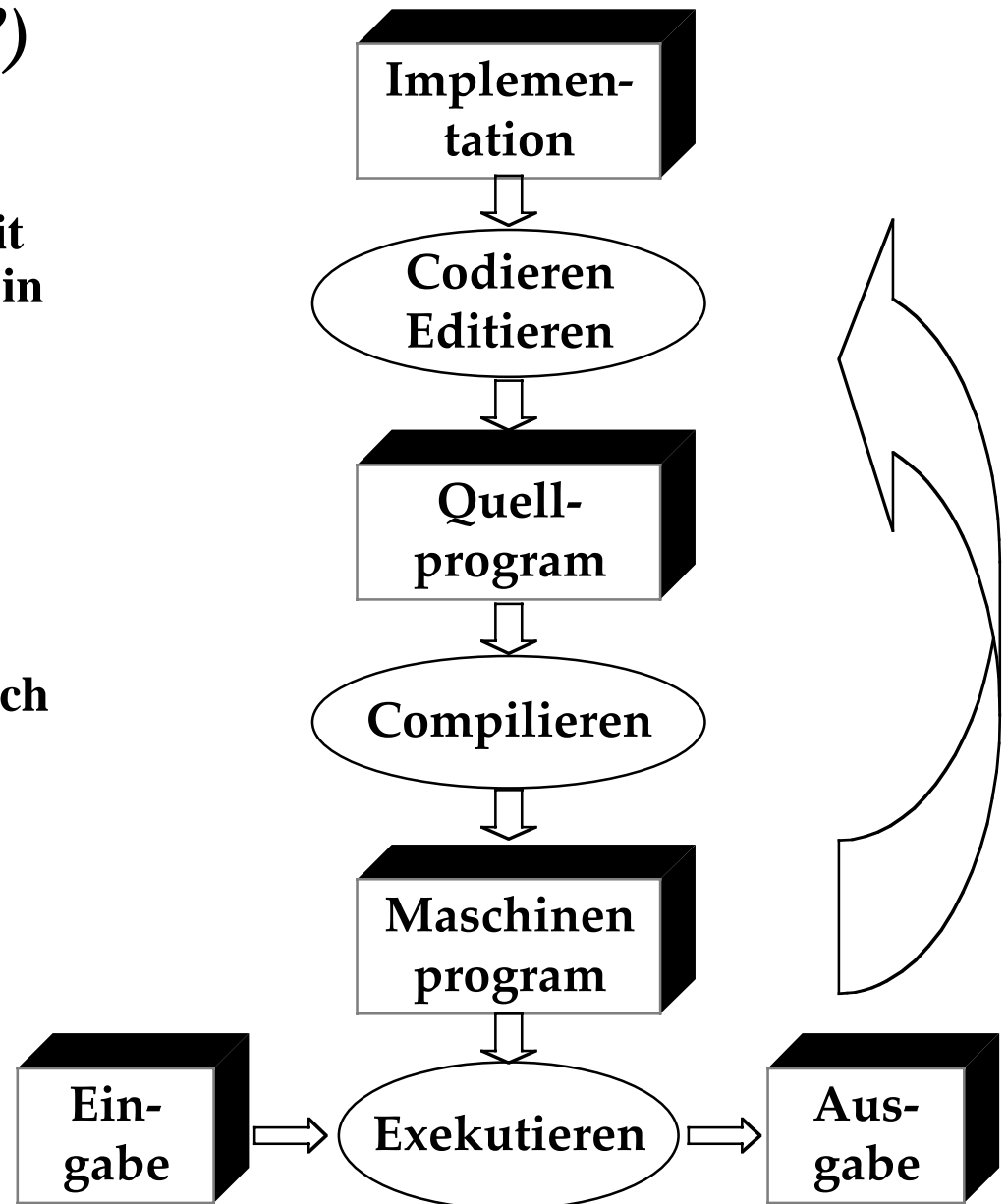
- Verbesserungen/Veränderungen am Quellprogramm

❖ Compilieren

- Transformation des Quellprogramms durch einen Übersetzer (Compiler) in ein ausführbares Programm (Maschinenprogramm)

❖ Exekutieren

- Exekution des Programs durch einen Interpreter auf einer Maschine.



Aufbau einer Java-Klasse:

```
class KlassenName {
```

Attributvariablen

Methodendefinitionen

```
}
```

KlassenName
Attribute
Operationen

Aufbau einer Java-Klasse:

```
class KlassenName {
```

```
    Variablentyp Variablenname;
```

```
    Methodendefinitionen
```

```
}
```

Aufbau einer Java-Klasse:

```
class KlassenName {
```

```
    Variablentyp Variablenname;
```

```
    Ergebnistyp Methodename (Parameterliste) {  
        Methodenrumpf  
    }
```

```
}
```

Beispiel einer Java-Klasse:

```
class Klausur {  
  
    int punktzahl;  
  
    Ergebnistyp Methodenname (Parameterliste) {  
        Methodenrumpf  
    }  
}
```

Beispiel einer Java-Klasse:

```
class Klausur {  
  
    int punktzahl;  
  
    int getPunktzahl () {  
        Methodenrumpf  
    }  
}
```

Prinzip der Informationskapselung: Auf Objekte und ihre Attribute kann nur über deren Schnittstelle zugegriffen werden; dazu verwendet man häufig so genannte ‘get’-Methoden

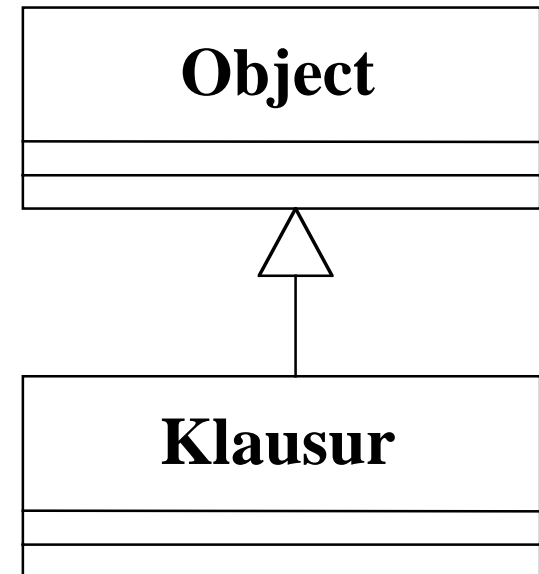
Beispiel einer Java-Klasse:

```
class Klausur {  
  
    int punktzahl;  
  
    int getPunktzahl () {  
        return punktzahl;  
    }  
}
```

Für die Rückgabe eines Operations- (bzw. Methoden)-Ergebnis wird in Java das 'return'-Schlüsselwort verwendet!

Die Vererbungs-Beziehung in Java:

```
class Klausur extends Object {  
  
    int punktzahl;  
  
    int getPunktzahl () {  
        return punktzahl;  
    }  
}
```



Für die Vererbungs-Beziehung wird in Java das ‘extends’-Schlüsselwort in der Unterklasse verwendet! Die Unterklasse erweitert also die Oberklasse (bzgl. Attributen und Operationen). Die Klasse Object ist in Java Oberklasse jeder anderen Klasse!

Konventionen für Java-Programme

- ❖ **Klassennamen beginnen mit einem Großbuchstaben**
- ❖ **Attributvariablen beginnen mit einem Kleinbuchstaben**
- ❖ **Methodennamen beginnen mit einem Kleinbuchstaben**
- ❖ **Zusammengesetzte Bezeichner werden ohne Trennzeichen geschrieben, wobei nachfolgende Teilwörter wieder mit Großbuchstaben beginnen**
- ❖ **Beispiele:**
 - **Klassen:** Klausur, Student, KellerNat
 - **Attribute:** matrikelnummer, erreichtePunktzahl
 - **Methoden:** getPunktzahl(), erzeugeVerzeichnis()
- ❖ **Jede Klasse in einer eigenen Datei**

Subaktivitäten während Implementation und Test (“edit-compile-execute cycle”)

❖ Codieren:

- Übersetzung des detaillierten Entwurfs mit Hilfe eines Editors in ein Quellprogramm in der ausgewählten Programmiersprache

❖ Editieren:

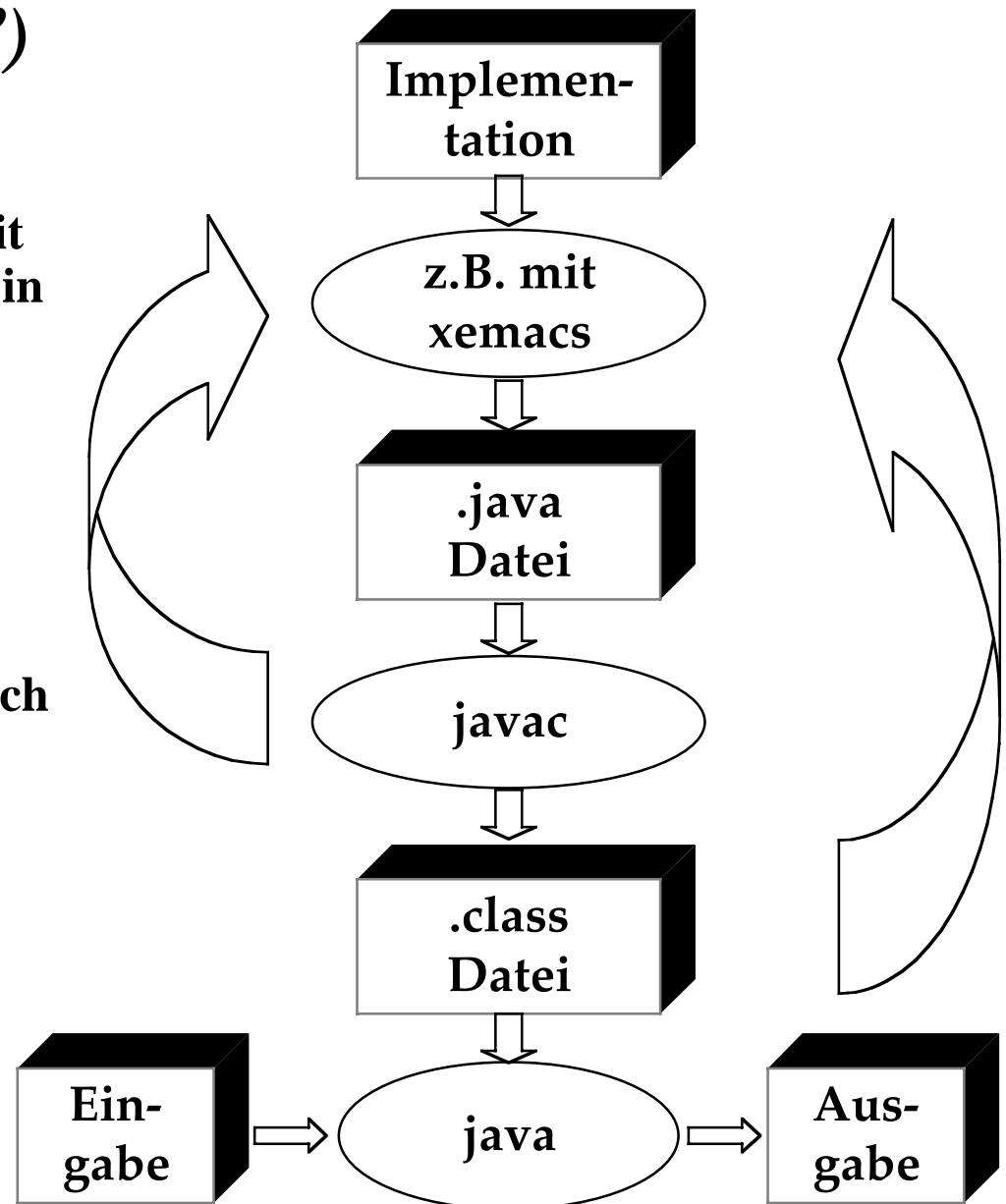
- Verbesserungen/Veränderungen am Quellprogramm

❖ Compilieren

- Transformation des Quellprogramms durch einen Übersetzer (Compiler) in ein ausführbares Programm (Maschinenprogramm)

❖ Exekutieren

- Exekution des Programs durch einen Interpreter auf einer Maschine.



Klassen und Objekte in Java:

- ❖ Die Erzeugung von Objekten einer Klasse wird häufig durch eine Operation, wie z.B. **create()** aus der Vorlesung, ermöglicht
- ❖ In Java wird diese Operation **Konstruktor** genannt
- ❖ Der Bezeichner bzw. Name dieser Operation entspricht dem Namen der Klasse

Beispiel einer Java-Klasse mit Konstruktor:

```
class Klausur {  
  
    Klausur () {}  
  
    int punktzahl;  
  
    int getPunktzahl () {  
        return punktzahl;  
    }  
}
```

Klassen und Objekte in Java:

- ❖ Die Erzeugung von Objekten einer Klasse wird häufig durch eine Operation, wie z.B. **create()** aus der Vorlesung, ermöglicht
- ❖ In Java wird diese Operation **Konstruktor** genannt
- ❖ Der Bezeichner bzw. Name dieser Operation entspricht dem Namen der Klasse

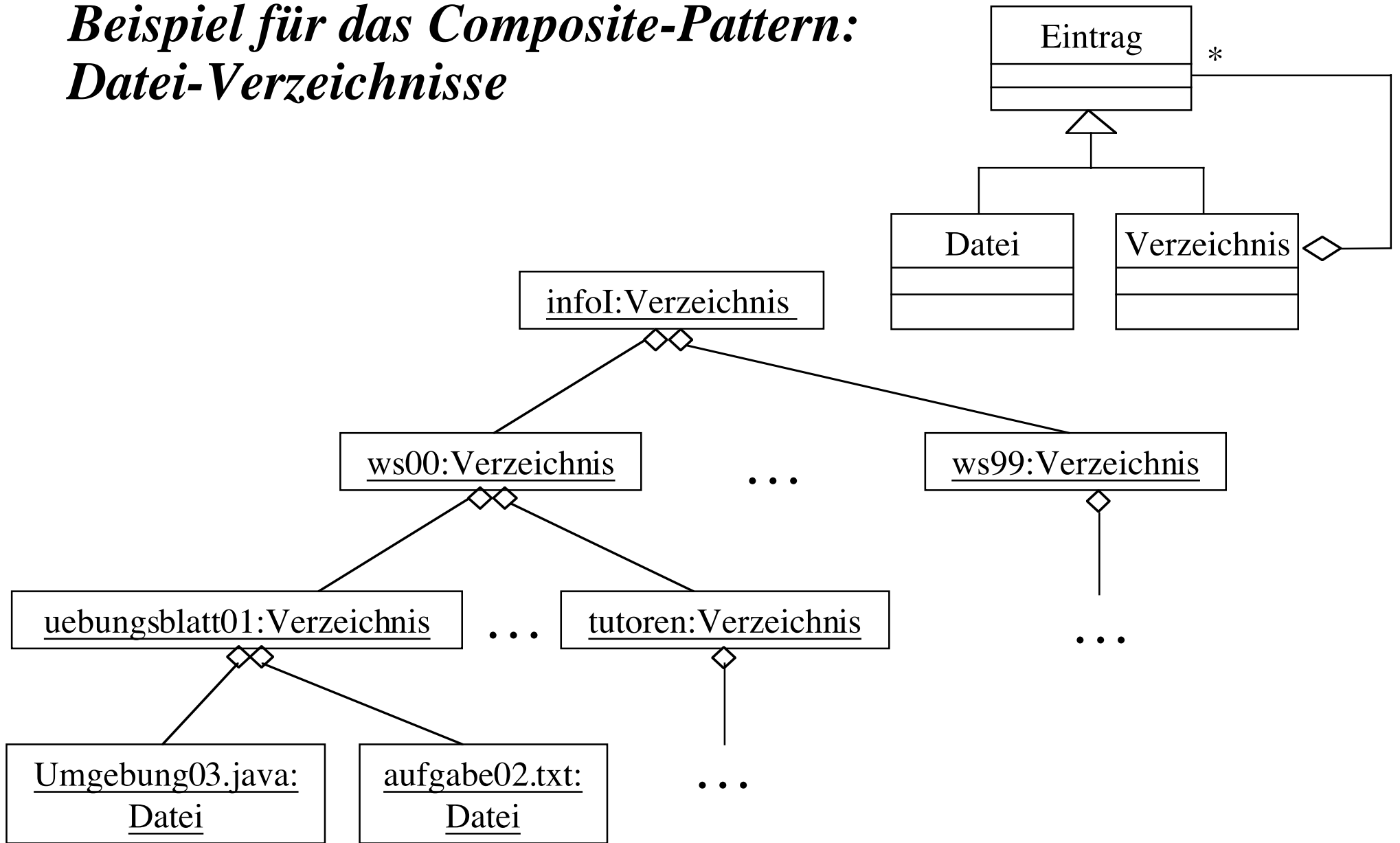
- ❖ Die Konstruktor-Operation muss nicht null-stellig sein (“der parameterlose Konstruktor”), sondern kann auch ein oder mehrere Argumente haben
- ❖ Es empfiehlt sich, in jeder Klasse auch den null-stelligen Konstruktor zu definieren!

Erzeugung von Objekten:

- ❖ Eine Instanz einer Klasse wird in Java durch die Verwendung des Schlüsselworts **new** verbunden mit einer Konstruktor-Operation erzeugt
- ❖ Dazu ein Beispiel:

```
Klausur teileKlausurAus () {  
  
    return new Klausur();  
  
}
```

Beispiel für das Composite-Pattern: Datei-Verzeichnisse



Das war's von meiner Seite. Noch Fragen?

**Vielen Dank für die
Aufmerksamkeit!**