

# OOP und JAVA

Informatik AG

07.11.2011 – 20H30

# Interfaces

- **Abstrakte** Klassen/Methoden (Wiederholung)
- Vom Problem zur Lösung
- Was ist ein **Interface**?
- Unterschied **Interface/Abstrakte Klasse**
- Implementierung

# Vom Problem zur Lösung

- Was ist, wenn eine Klasse Eigenschaften von mehreren Klassen vererben soll?
- Problem: Keine **Mehrfachvererbung** in Java (Grund: viele Schwierigkeiten)
- Mehrfachvererbung kann manchmal praktisch sein.

# Was ist ein Interface?

- Lösung: **Interface**

- Definition:

*Ein Interface in Java ist die Möglichkeit, zu einer oder mehreren Klassen bzw. von einem oder mehreren anderen Interfaces eine Schnittstelle als neue, komplexe Signatur zu definieren und anzuwenden.*

*R. Dumke, Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik*

# Was ist ein Interface?

- **Definition:**

*Interfaces (deutsch "Schnittstellen") erlauben es, ein Konzept ähnlich der Mehrfachvererbung in Java zu verwenden. Sie entsprechen in etwa abstrakten Klassen, in denen keine Objektvariablen und ausschließlich abstrakte Methoden enthalten sind. Es handelt sich bei Interfaces jedoch nicht um Klassen*

*Uni Köln - Philosophische Fakultät - Institut für Linguistik*

# Was ist ein Interface?

- Ein Interface ist ein ***Versprechen***, dass gewisse Methoden implementiert sind.
- Ein Interface dient ***nur*** dazu anzugeben, dass eine Klasse gewisse Methoden implementiert.
- Jede Klasse, die das Interface implementiert, muss dessen Methoden auch tatsächlich definieren, sonst meldet der Compiler einen Fehler.

# Unterschied Interface/Abstrakte Klasse

- Ein Interface ist keine „Klasse“.
- In einem Interface müssen alle Methoden als „abstract“ deklariert werden.
- Eine abstrakte Klasse kann normale Methoden haben (nicht alle Methoden müssen „abstract“ sein).

# Implementierung

- **Java-Syntax:**

```
interface EinInterface {  
    . . .  
    . . .  
    . . .  
}
```

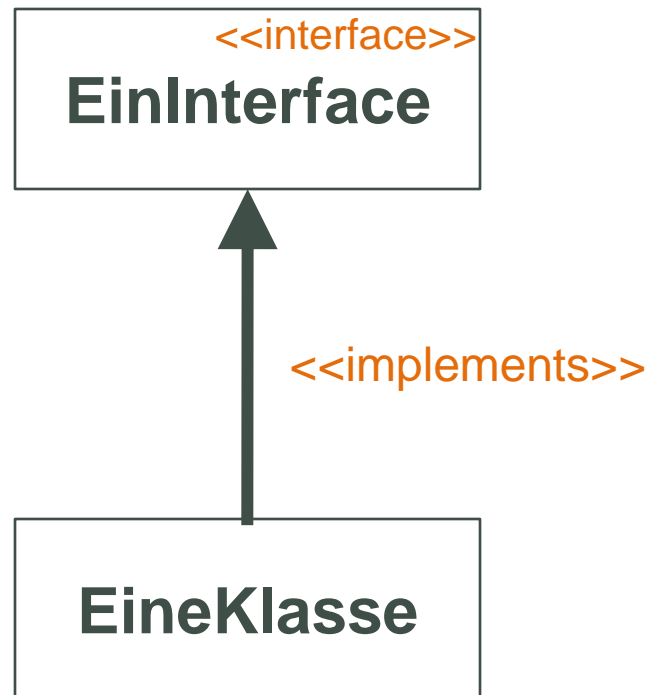
# Implementierung

- **Java-Syntax:**

```
class EineKlasse implements EinInterface {  
    . . .  
    . . .  
    . . .  
}
```

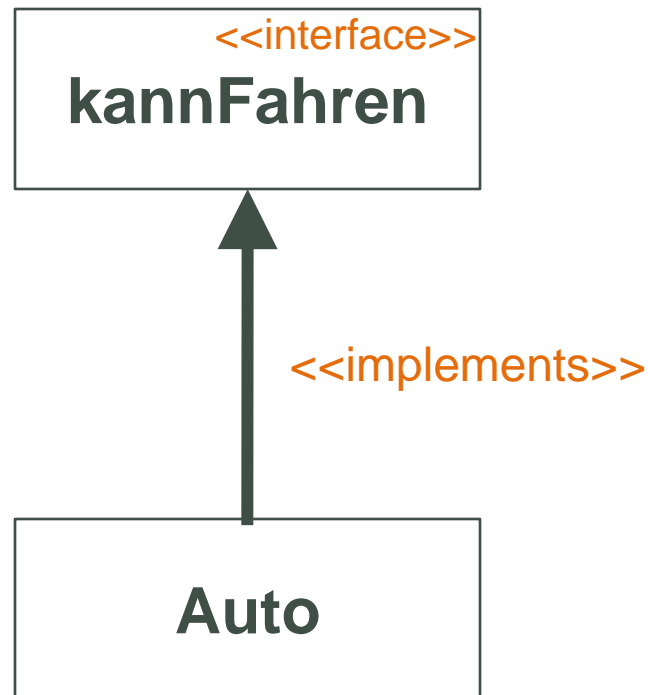
# Implementierung

- UML:



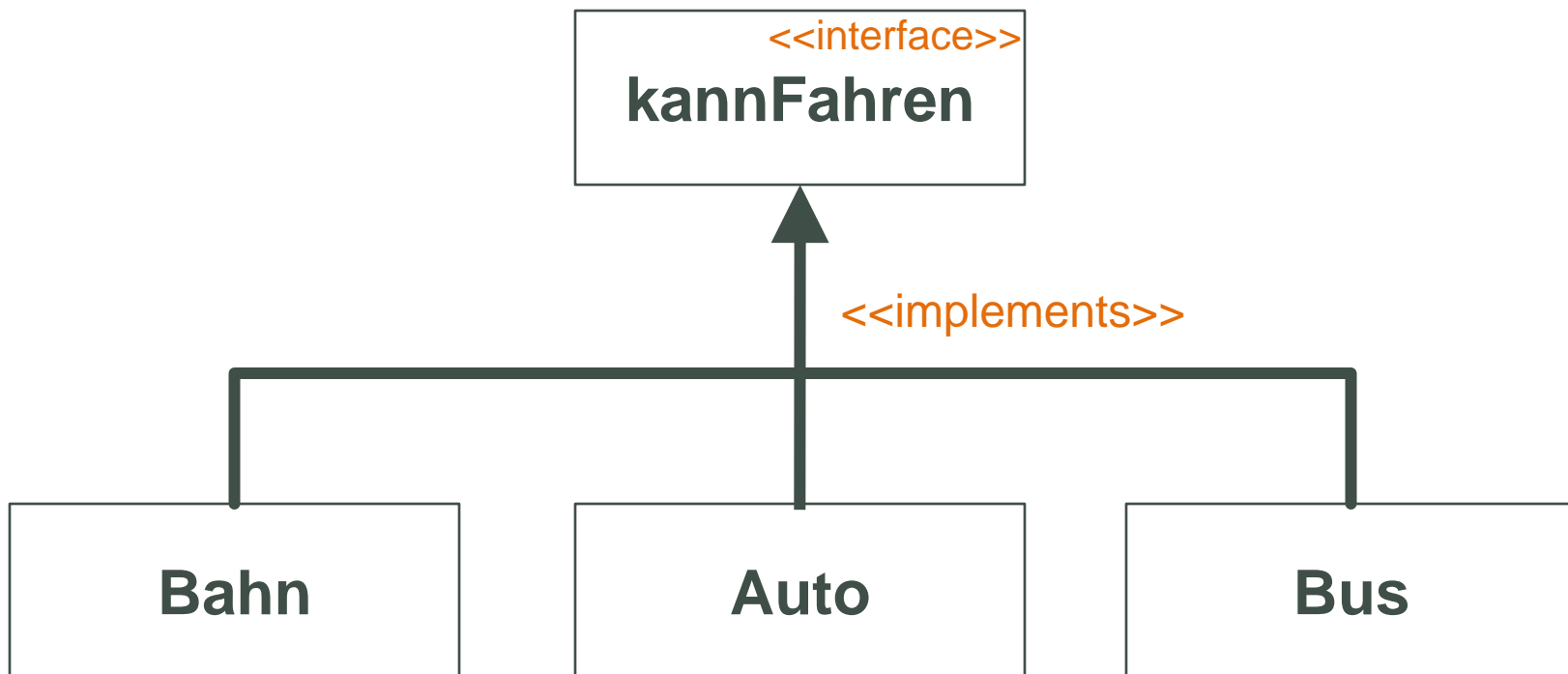
# Implementierung

- **Beispiel:**



# Implementierung

- **Beispiel:**



# Implementierung

```
interface kannFahren {  
    ...  
    void fahre(int geschwindigkeit);  
    //Methoden sind automatisch „public“  
    //Keine Implementierung  
}
```

```
class Auto implements kannFahren {  
    private int geschwindigkeit;  
    ...  
    void fahre(int geschwindigkeit) {  
        ....  
        ....  
    }  
}
```

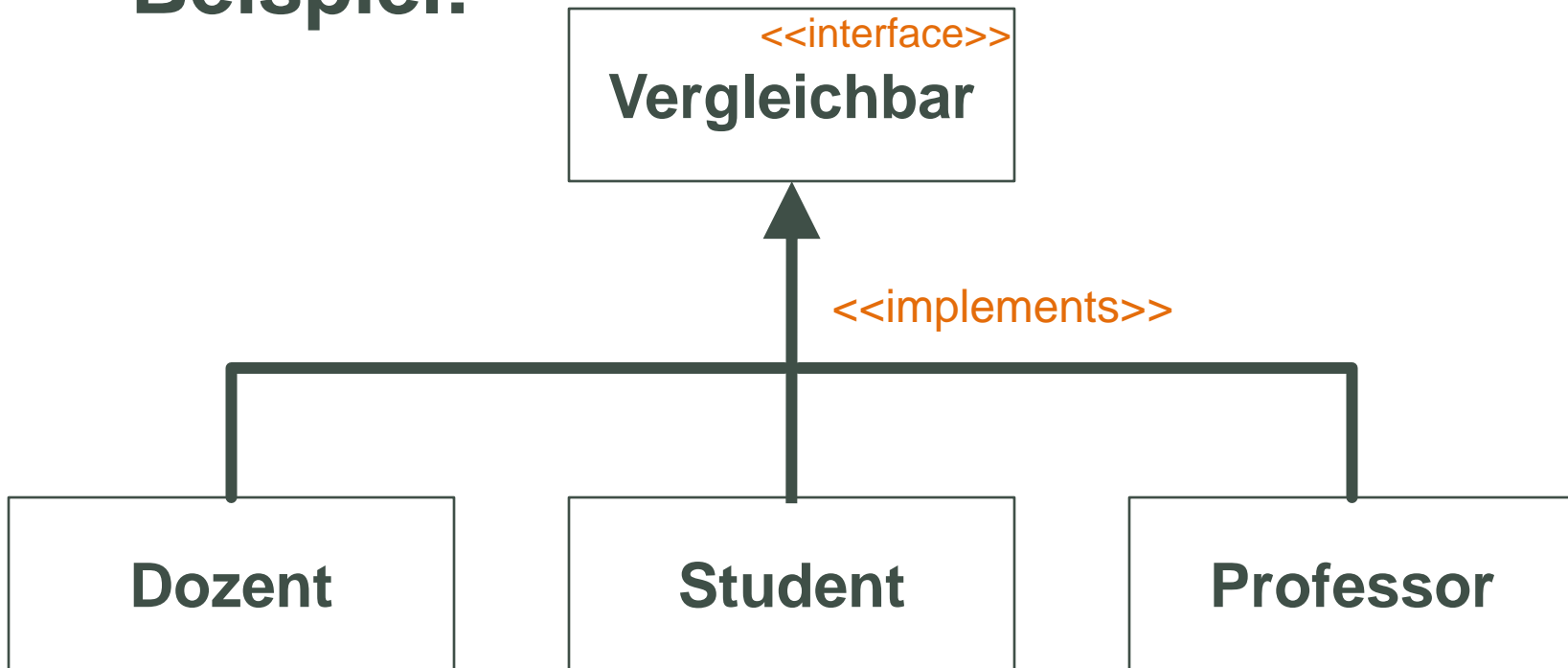
# Implementierung

- Interfaces in der API:
  - Z.B: `java.lang.Comparable`

Method Summary	
int	<a href="#">compareTo</a> ( <u>I</u> o) Compares this object with the specified object for order.

# Implementierung

- **Beispiel:**



# Implementierung

```
interface Vergleichbar {  
    ...  
    int vergleiche();  
    //Die Methode vergleiche() muss in  
    //den Klassen Student, Dozent,  
    //Professor implementiert werden  
}
```

# Fragen? Informatik AG



[dayzine.de/forum](http://dayzine.de/forum)

Skype: *informatikag*