Objektorientierte Programmierung

Studiengang Medieninformatik

Hans-Werner Lang

Hochschule Flensburg

Vorlesung 3
29.03.2017

Was bisher geschah...

Hochschule
Flensburg
University of
Applied Sciences

- Klassen und Objekte
- Attribute und Methoden
- Klassen ableiten mit `extends`
- Vererbung von Attributen und Methoden
- Methoden überschreiben
- Methoden überladen
- Konstruktoren überladen
- Schlüsselworte `this` und `super`

Polymorphie

Hochschule Flensburg
University of Applied Sciences

Objekte einer abgeleiteten Klasse lassen sich als Objekte der Basisklasse ansprechen.

Das Objekt *myframe* wird als Objekt der Basisklasse *JFrame* deklariert. Erzeugt wird aber ein Objekt der abgeleiteten Klasse *MyFrame*.

```
JFrame myframe;
myframe=new MyFrame();
```

Damit sind auf das Objekt *myframe* nur Methoden anwendbar, die auch in der Basisklasse vorhanden sind.

```
myframe.setSize(500);    // nicht anwendbar (nur 1 Parameter)
myframe.setTitle("This is MyFrame");   // funktioniert
```

Das Verrückte: Es wird nicht die Implementation von *setTitle* der Basisklasse ausgeführt, sondern die (überschriebene) Implementation von *setTitle* der abgeleiteten Klasse. Dies wird als Polymorphie bezeichnet.

Polymorphie

Hochschule Flensburg
University of Applied Sciences

Beispiel: Test von verschiedenen Sortierverfahren

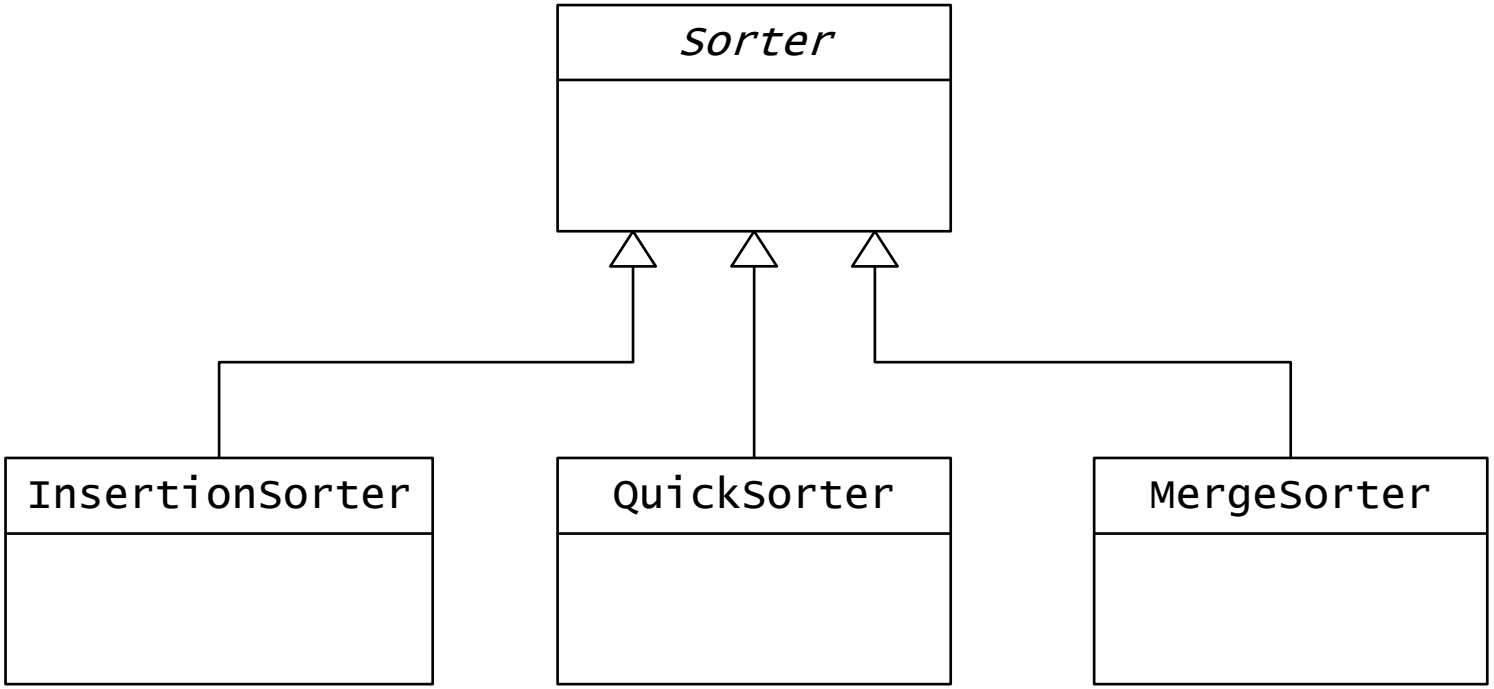
Objekte von abgeleiteten Klassen werden als Objekte der Basisklasse *Sorter* angesprochen.

```
Sorter[] s=new Sorter[3];    // Array von 3 Sortierern
s[0]=new InsertionSorter();
s[1]=new QuickSorter();
s[2]=new MergeSorter();
for (int i=0; i<3; i++)
{
    copy(a, b);       // Array a nach b kopieren
    s[i].sort(b);     // sortiert mit dem jeweiligen Sortierverfahren
    testIfSorted(b);
}
```

Die Basisklasse muss die Methode *sort* enthalten.

Klassendiagramm

Hochschule Flensburg
University of Applied Sciences



Abstrakte Klasse

Hochschule Flensburg
University of Applied Sciences

Die Basisklasse *Sorter* muss die Methode *sort* enthalten.

Aber wie soll *sort* in der Basisklasse *Sorter* implementiert werden?

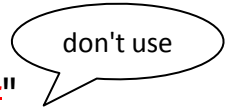
Antwort: Gar nicht.

```
public abstract class Sorter
{
    public abstract void sort(int[] a);   // keine Implementierung
}
```

Die Methode *sort* wird als abstrakte Methode deklariert. Dies heißt, dass sie erst in einer abgeleiteten Klasse implementiert wird.

Die Basisklasse *Sorter* ist als abstrakte Klasse gekennzeichnet. Damit können wir von *Sorter* direkt keine Objekte erzeugen, sondern nur von davon abgeleiteten Klassen.

"Die Klasse *Sorter* ist nicht ~~instanziierbar~~"



Konkrete Klasse

Hochschule Flensburg
University of Applied Sciences

Ein konkretes Sortierverfahren enthält eine Implementierung der Methode *sort*.

```
public class InsertionSorter extends Sorter
{
    public void sort(int[] a)
    {
        insertionSort(a);
    }

    private void insertionSort(int[] a)
    {
        // ... hier die Implementierung des Sortierverfahrens
    }
}
```

Abstrakte Klasse

Hochschule Flensburg
University of Applied Sciences

Abstrakte Klassen können auch Attribute enthalten.

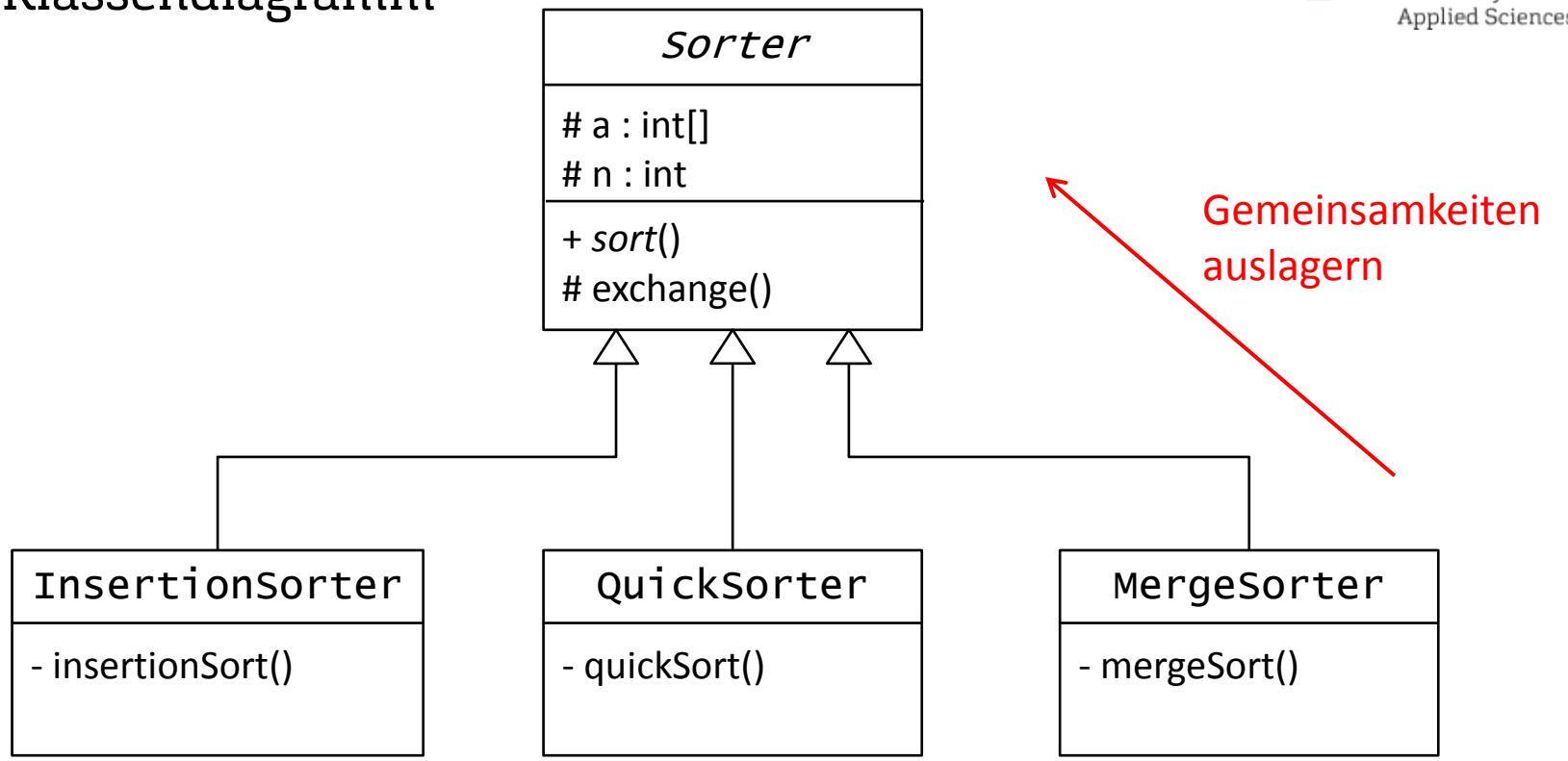
Und sie können auch nicht-abstrakte Methoden, also implementierte Methoden enthalten.

```
public abstract class Sorter
{
    protected int[] a;      // zu sortierende Datenfolge
    protected int n;        // deren Länge

    public abstract void sort(int[] a);    // keine Implementierung

    // vertauscht Datenelemente a[i] und a[j]
    protected void exchange(int i, int j)
    {
        int h=a[i]; a[i]=a[j]; a[j]=h;
    }
}
```

Klassendiagramm



Konkrete Klasse

Neue Implementierung nach Auslagern der Attribute a und n in die Basisklasse *Sorter*.

```
public class InsertionSorter extends Sorter
{
    public void sort(int[] a)
    {
        this.a=a;
        n=a.length;
        insertionSort();
    }

    private void insertionSort()
    {
        // ... hier die Implementierung des Sortierverfahrens
    }
}
```

Abstrakte Klasse

Hochschule Flensburg
University of Applied Sciences

Kann eine abstrakte Klasse einen Konstruktor enthalten?

Ja! Dieser kann mit `super` in einem Konstruktor der abgeleiteten Klasse aufgerufen werden (bzw. der Standardkonstruktor wird automatisch aufgerufen).

```
public abstract class Sorter
{
    public boolean ascending;   // aufsteigend sortieren

    public Sorter(boolean asc)
    {
        ascending=asc;
    }

    public Sorter()
    {
        this(false);          // standardmäßig absteigend sortieren
    }
    ...
}
```

Interface

Hochschule Flensburg
University of Applied Sciences

Eine abstrakte Klasse, die auf Attribute und implementierte Methoden völlig verzichtet, ist ein Interface.

```
public interface Sorter
{
    public void sort(int[] a);    // keine Implementierung
}
```

Der Modifizierer `abstract` kann weggelassen werden.

Außer abstrakten Methoden sind in Interfaces auch Konstanten zulässig:

```
public final static int MAX_LENGTH=1000;
```

Interface implementieren

Hochschule Flensburg
University of Applied Sciences

Eine Klasse implementiert ein Interface, indem alle abstrakten Methoden des Interfaces implementiert werden.

```
public class InsertionSorter implements Sorter
{
    public void sort(int[] a)
    {
        insertionSort(a);     // Implementierung
    }

    private void insertionSort(int[] a)
    {
        // ... hier die Implementierung des Sortierverfahrens
    }
}
```

Klassen erweitern, Interfaces implementieren

Eine Klasse kann nur eine einzige Basisklasse direkt erweitern, aber mehrere Interfaces implementieren:

```
public class PaintPanel extends JPanel implements MouseListener, Observer
{
    ...

}
```

Zusammenfassung

Hochschule Flensburg
University of Applied Sciences

- Polymorphie: `JFrame myframe=new MyFrame();`
- Polymorphie: Objekte verschiedener Klassen werden als Objekte einer gemeinsamen Oberklasse oder eines gemeinsamen Interfaces angesprochen, verhalten sich aber individuell je nach ihrer konkreten Implementierung
- Abstrakte Klasse: Basisklasse für konkrete Klassen, von denen dann Objekte erzeugt werden
- Abstrakte Klasse: Enthält Attribute, abstrakte Methoden und implementierte Methoden, sogar möglicherweise Konstruktoren
- Interface: Enthält nur abstrakte Methoden, allenfalls noch Konstanten
- Eine Klasse kann höchstens eine Basisklasse erweitern, aber mehrere Interfaces implementieren

Beim nächsten Mal:

- GUI-Steuerelemente
- Grafik (Punkte, Linien, Formen)