Abschnitt 5.4

Subtypen und Vererbung

Subtypen und Vererbung

Dieser Abschnitt erläutert die Konzepte der Subtypbildung und Vererbung.

Überblick:

- Klassifizieren von Objekten
- Subtypen und Schnittstellentypen
- Vererbung

Unterabschnitt 5.4.1

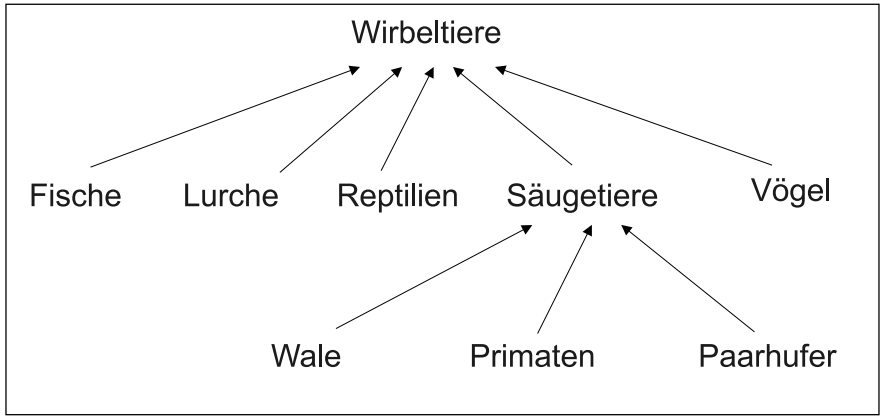
Klassifizieren von Objekten

Begriffsklärung: (Klassifikation)

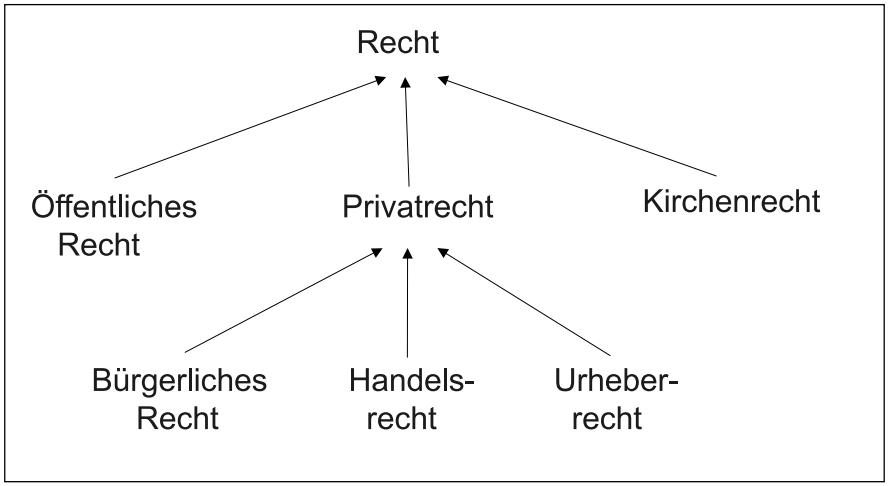
Klassifikation ist eine zentrale Grundlage der objektorientierten Modellierung und Programmierung.

Klassifizieren ist eine allgemeine Technik, mit der Wissen über Begriffe, Dinge und deren Eigenschaften hierarchisch strukturiert wird. Das Ergebnis nennen wir eine ***Klassifikation***.

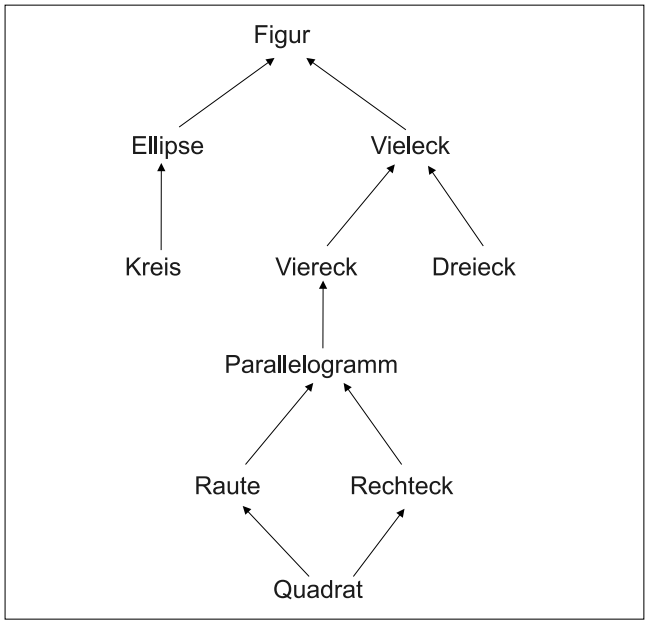
Beispiele: (Klassifikationen)

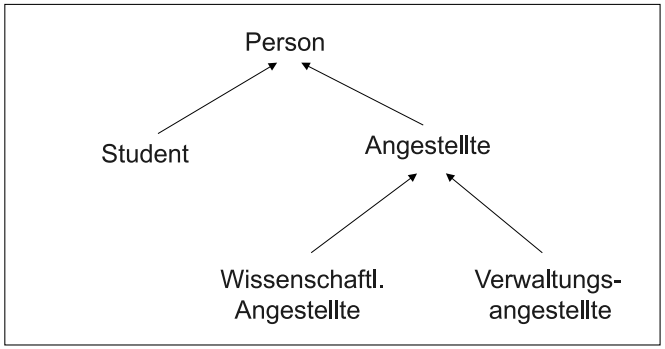


→ _ ist_ein _ (Vogel ist ein Wirbeltier)

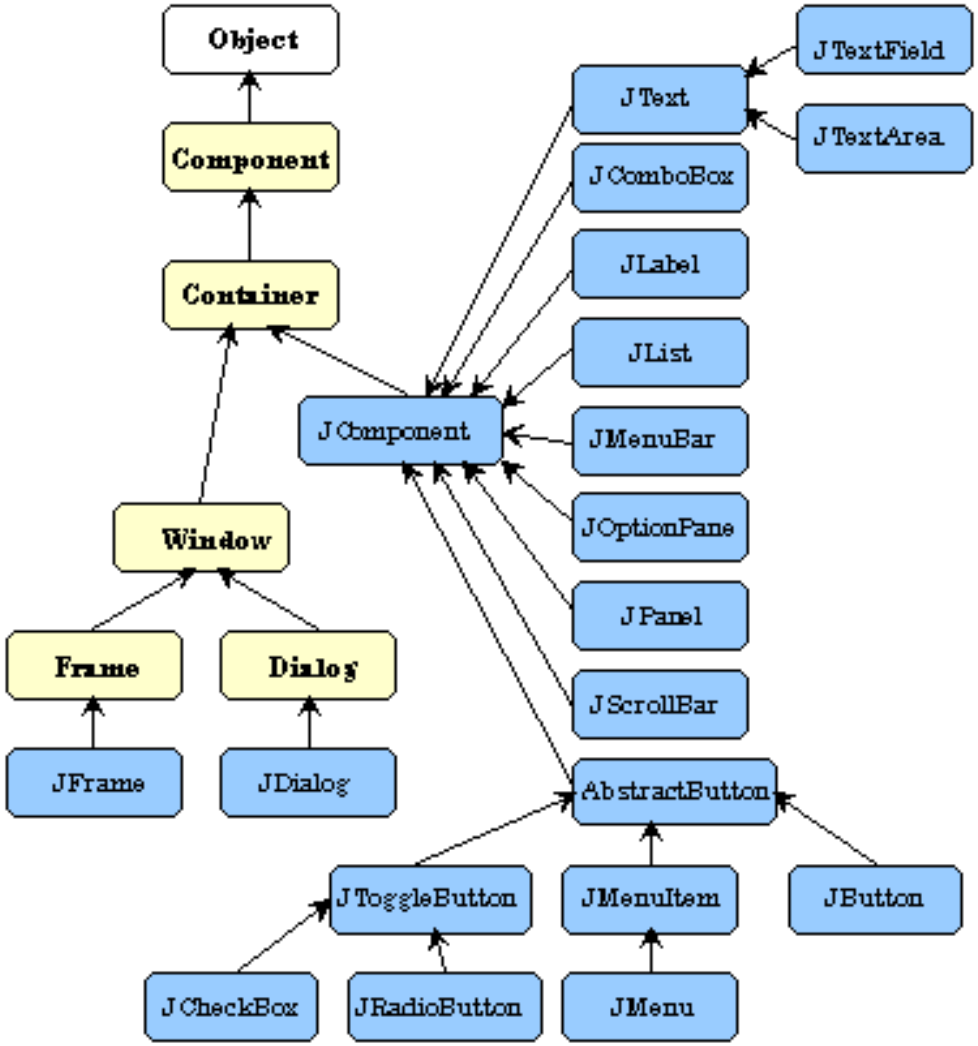


Beispiele: (Klassifikationen) (2)





Beispiele: (Klassifikationen) (3)



Bemerkung:

- Beobachtungen zu Klassifikationen:
  - Sie können sich auf Objekte oder Gebiete beziehen.
  - Sie können baumartig oder DAG-artig sein.
  - Objektklassifikationen begründen ist-ein-Beziehungen.
  - Es gibt abstrakte Klassen (ohne ”eigene” Objekte) und nicht abstrakte Klassen
- Üblicherweise stehen die allgemeineren Begriffe oben, die spezielleren unten.

Ziel:

Anwendung der Klassifikationstechnik auf Objekte in der Software-Entwicklung.

Klassifikation in der Softwaretechnik:

Objekte lassen sich nach ihren Eigenschaften klassifizieren:

- Alle Objekte mit ähnlichen Eigenschaften werden zu einer Klasse zusammen gefasst.
- Die Klassen werden hierarchisch geordnet.

Klassifikation beruht auf:

- Schnittstellen der Klassen/Objekte
- Verhalten/Eigenschaften der Objekte

Klassifikation in der Softwaretechnik: (2)

Genauer:

1. Syntaktisch: Subklassenobjekte haben im Allg. größere Schnittstellen als Superklassenobjekte (Auswirkung auf Programmiersprache)

2. Semantisch: Subklassenobjekte bieten mindestens die Eigenschaften der Superklassenobjekte.

Zentraler Aspekt der OO-Programmentwicklung: Entwurf und Realisierung von Klassen- bzw. Typhierarchien.

Abstraktion/Generalisierung

Begriffsklärung: (***Abstraktion***)

.. das Heraussondern des unter einem bestimmten Gesichtspunkt Wesentlichen vom Unwesentlichen. [Meyers großes Taschenlexikon]

Abstraktion geht also vom Speziellen zum Allgemeinen.

Ansatz:

- Für unterschiedliche Objekte bzw. Typen mit gemeinsamen Eigenschaften soll Software entwickelt werden.

  Beispiele:

  - Komponenten von Fenstersystemen (Menues, Schaltflächen, Textfelder, ...)
  - Ein-/Ausgabeschnittstellen (Dateien, Netze, ...)

- Erarbeite einen abstrakteren Typ, der die gemeinsamen Eigenschaften zusammenfasst und eine entsprechende Schnittstelle bereitstellt (Verkleinern der Schnittstelle).

- Programme, die sich auf die Schnittstelle des abstrakteren Typs abstützen, arbeiten für alle Objekte mit spezielleren Schnittstellen.

Beispiel: (Gemeinsame Eigenschaften)

Wir betrachten zwei Klassen mit gemeinsamen Eigenschaften:

```
class Student {
    String name;
    int    matNr;
    ...
    void drucken() {
        System.out.println( name );
        System.out.println( matNr );
        ...
    }
}
```

Beispiel: (Gemeinsame Eigenschaften) (2)

```
class Professor {
    String name;
    int    telNr;
    ...
    void drucken() {
        System.out.println( name );
        System.out.println( telNr );
        ...
    }
}
```

Beispiel: (Gemeinsame Eigenschaften) (3)

Anforderung:
Alle Personendaten sollen gedruckt werden.

Abstraktion:

- Entwickle einen Typ `Person`, der die Nachricht drucken versteht.
- Formuliere das Drucken der Personendaten im Typ `Person`

```
Person[]  p =  new Person[4];
p[0]  =  new Student(...);
p[1]  =  new Professor(...);
...
for( i=0; i<p.length; i++ ) {
   p[i].drucken();
}
```

ungleiche Typen

dynamisches Binden

Beispiel: (Gemeinsame Eigenschaften) (4)

Deklaration des Typs `Person` in Java:

```
interface Person {
    void drucken();
}
```

Anpassen der Typen `Student` und `Professor`:

```
class Student implements Person {
    ... // wie oben
}
```

```
class Professor implements Person {
    ... // wie oben
}
```

Spezialisierung

Begriffsklärung: (Spezialisierung)

Spezialisierung bedeutet hier das Hinzufügen speziellerer Eigenschaften zu einem Gegenstand oder das Verfeinern eines Begriffs durch Einführen weiterer Merkmale (z.B. berufliche Spezialisierung).

Spezialisierung geht also vom Allgemeinen zum Speziellen.

Ansatz:

- Existierende Objekte bzw. Typen sollen zusätzliche Anforderungen erfüllen.

  Beispiele:

  - spezielle Komponenten für eine graphische Bedienoberfläche
  - Anpassung eines Buchführungssystems an die speziellen Anforderungen einer Firma

- Erweitere die existierenden Typen (zusätzliche Attribute & Methoden, Anpassen von Methoden). Im Allg. vergrößern sich dabei die Schnittstellen.
- Existierende Programme für die allgemeineren Typen arbeiten auch mit den spezielleren Typen.

Ansatz: (2)

Programmtechnische Mittel zur Spezialisierung:

- Hinzufügen von Attributen
- Hinzufügen von Methoden
- Anpassen, Erweitern bzw. Implementieren von Supertyp-Methoden:
  - Überschreiben
  - Anwenden überschriebener Methoden

Beispiel: (Spezialisierung)

Wir spezialisieren die Klasse Frame des AWT:

```
package memoframe;

import java.awt.* ;

class MemoFrame  extends Frame  {
  private Color letzterHintergrund;

  public void einstellenLetztenHintergrund() {
    setBackground( letzterHintergrund );
  }
  public void setBackground( Color c ) {
    letzterHintergrund = getBackground();
    super.setBackground( c );
  }
}
```

Beispiel: (Spezialisierung) (2)

```
package memoframe;

public class TestMemoFrame {
  public static void main(String[] args) {
    MemoFrame f = new MemoFrame();
    f.setLocation( 200, 200 );
    f.setSize( 300, 200 );
    f.setVisible( true );

    f.setBackground( Color.red );
    f.update( f.getGraphics() );
    try{  Thread.sleep(4000); }
    catch( Exception e ){}
```

Beispiel: (Spezialisierung) (3)

```
        f.setBackground( Color.green );
        f.update( f.getGraphics() );
        try{  Thread.sleep(4000); }
        catch( Exception e ){}

        f.einstellenLetztenHintergrund();
        f.update( f.getGraphics() );
        try{  Thread.sleep(4000); }
        catch( Exception e ){}

        System.exit( 0 );
    }
}
```

Bemerkung:

- Eine genaue Kenntnis der zu spezialisierenden Klasse ist meist nicht nötig. Die ererbten Eigenschaften kann man über die Methoden ansprechen.
- Überschreibende Methoden können die überschriebene Methode nutzen.
- Zwei Aspekte werden demonstriert:
  - Subtypbeziehung: Ein MemoFrame-Objekt ist ein Frame-Objekt.
  - Vererbung: Ein MemoFrame-Objekt erbt den größten Teil seiner Implementierung von der Klasse Frame.

Zusammenfassung zu 5.3.1

- Jedes Objekt hat Schnittstelle aus Attributen und Methoden.
- Objekte werden entsprechend ihrer Schnittstelle klassifiziert.
- Allgemeinere Objekte haben kleinere Schnittstelle als speziellere Objekte.
- Abstraktion/Generalisierung erlaubt es, Typen zu deklarieren, die die relevante Gemeinsamkeiten anderer Typen ausdrücken.
- Spezialisierung erlaubt es, Typen zu deklarieren, die die Funktionalität existierender Typen erweitern.
- Entwurf geeigneter Klassenhierarchien ist ein zentraler Aspekt des objektorientierten Entwurfs bzw. der objektorientierten Programmierung. Dabei sind Abstraktion und Spezialisierung sinnvoll zu kombinieren.