Objektorientierte Programmierung: Vererbung und Polymorphie

Vererbung

Warum soll überhaupt etwas vererbt werden?
Die Idee dazu kommt aus der Biologie. Kinder erben von den Eltern bzw. von einem Elternteil, d.h. eine "Kind"-Klasse erbt Eigenschaften und Methoden von ihrer "Eltern"-Klasse.

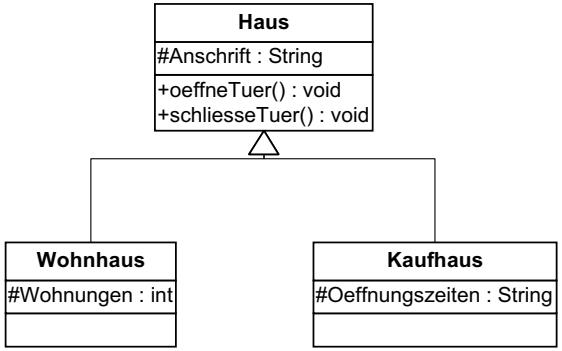
Vererbung in der Objekt-Orientierten Programmierung erlaubt bessere Modellierungsmöglichkeiten und bessere Wiederverwertbarkeit. Vererbung wird insbesondere beim Design von Bibliotheken für grafische Benutzeroberflächen (GUI) verwendet.

Die "Eltern"-Klasse heißt Superklasse, Oberklasse oder Basisklasse. Die "Kind"-Klasse heißt Subklasse, Unterklasse oder abgeleitete Klasse.

Wenn man mit Vererbung arbeitet, sind die folgenden Begriffe wichtig:

- Spezialisierung:
  Bei der Spezialisierung werden allgemeine Eigenschaften einer Klasse (Oberklasse) in Unterklassen konkretisiert und erweitert. Die Unterklassen erben die Eigenschaften der Oberklasse, passen sie ihren speziellen Anforderungen an und ergänzen sie um weitere Eigenschaften, Methoden oder Beziehungen.
- Generalisierung:
  Bei der Generalisierung werden Klassen mit ähnlichen Eigenschaften zu einer Oberklasse zusammengefasst. Gemeinsame Eigenschaften, Methoden und Assoziationen der Unterklassen werden in der Oberklasse definiert und an die Unterklassen vererbt.

Vererbung in UML:



Bemerkungen:

- Zur Vereinfachungen sind die entsprechenden Konstruktoren nicht dargestellt.
- # steht für protected
- Von der Unterklasse aus sind alle Attribute und alle Methoden der Oberklasse, die protected oder public eingestuft sind, zugreifbar. Auf private Elemente der Oberklasse können Unterklassen nicht zugreifen.
- Die Vererbung entspricht einer "... ist ein(e) ..."-Relation.
- Vererbung wird in UML mit Hilfe eines Pfeiles von Unter- zur Oberklasse dargestellt.

Vererbung in Java:

```
public class Haus{

  protected String Anschrift;

  public void oeffneTuer(){ }
  public void schliesseTuer(){ }
}

public class Wohnhaus extends Haus{

   protected int Wohnungen;
}

public class Kaufhaus extends Haus{
  protected String Oeffnungszeiten;
}
```

Bemerkungen:

- Jede Klasse befindet sich in einer eigenen Datei mit entsprechenden Namen.
- Die Sichtbarkeit protected bedeutet, dass auf eine Eigenschaft oder eine Methode nur die Unterklassen einer Klasse zugreifen können, aber keine anderen Klassen.
- Wir haben jetzt eine sogenannte *Objekthierarchie* gebildet. Von einer Superklasse Haus wurden mehrere Subklassen *abgeleitet*.

Aufgaben:

1. Ergänze in den oben dargestellten Klassen die Konstruktoren und Ausgaben. Schreibe ein Testprogramm, dass die Klassen verwendet.
2. Modelliere eine Objekthierarchie mit Vererbung in UML, die Autos, Motorräder, Fahrräder, Busse, LKW beschreibt. Finde passende Oberklassen.

Überschreiben von Methoden

Beim Überschreiben einer Methode ist die Neudefinition einer Methode aus der Oberklasse (in abgewandelter Weise) in der Unterklasse.
So kann man z.B. in der Klasse Kaufhaus, die Methoden oeffneTuer und schliesseTuer neu definieren („*überschreiben*“), so dass die Türen nur innerhalb der Öffnungszeiten benutzt werden können. Die Methode aus der Oberklasse ist dann von außerhalb nicht mehr verfügbar.

Aufgabe:

3. Modelliere geometrische Objekte. Gehe von einer Superklasse GeometrischeFigur aus. Die Klasse hat eine Methode namens `print` ohne Parameter und Rückgabewert. Leite von ihr die Klassen Dreieck, Rechteck, Kreis ab. Füge in jeder Klasse sinnvolle Eigenschaften hinzu und überschreibe in jeder Klasse die Methode print, so dass sie jeweils sinnvolle Ausgaben liefert. Definiere außerdem eine Klasse Punkt mit Eigenschaften x und y vom Typ int, die von den anderen Klassen verwendet wird.