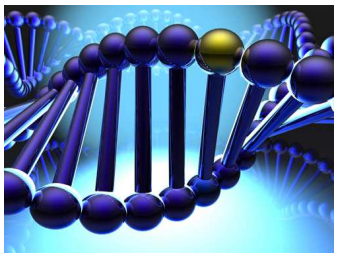
Vererbung und Polymorphie



Beschreibung
In diesem Kapitel besprechen wir Begriffe wie Vererbung, den Aufruf polymorphe Methoden, Methoden überschreiben und die Konstruktoren.

Inhaltsverzeichnis

Quellen: DoMe Beispiel aus BlueJ

1 Falsche Programmierung

1.1 Medienverwaltung falsch aufgebaut

In unserem ersten Beispiel wollen wir eine Medienverwaltung erweitern und dabei feststellen, wie durch falsche Programmierung die Wartbarkeit und Erweiterbarkeit eines Programmes verschlechtert wird. Das Programm besteht aus einer handvoll Klassen. Für jedes Medium gibt es eine Klasse. Folgende Klassen sind anfangs im Programm zu finden:

`CD`	Klasse für das Medium CD.
`Video`	Klasse für das Medium Video.
`Database`	Klasse für die Verwaltung der Medien.

Die Medienklassen haben zusätzlich je eine print Methode, die alle Attribute der Klasse ausgibt.

1.1.1 Aufgabe – Klasse Book

Erweitern Sie die Medienverwaltung mit einer Klasse Book.

`Book`	Klasse für das Medium Buch.

Die Klasse soll folgende Attribute besitzen:

```
private String title;
private String author;
private String isbn;
private int pages;
private boolean gotIt;
private String comment;
```

1.1.2 Aufgabe – Klasse Database erweitern

Erweitern Sie die Klasse `Database` so, dass sie das neue Medium Buch aufnehmen kann.

```
private ArrayList<Book> books;
```

Deklarieren Sie hierfür eine Liste für Bücher und instanzieren Sie diese im Konstruktor

```
books = new ArrayList<Book>();
```

Programmieren Sie eine Methode, über welche Bücher hinzugefügt werden können.

```
public void addBook(Book book) {
     books.add(book);
}
```

Erweitern Sie die Methode `printList`, mit einer Schleife, welche alle Bücher der entsprechenden List ausgibt.

```
// print list of Books
for (Book book : books) {
      book.print();
      System.out.println();
}
```

Nun müssen wir noch ein Buch Objekt erzeugen und dieses in der Verwaltung speichern.

```
Book bh1 = new Book("Das Parfüm");
…
db.addBook( bh1 );
```

Feststellung:
Was fällt Ihnen auf, wenn Sie die `Database` Klasse genauer betrachten und die Änderungen resp. die Erweiterungen die Sie machen mussten hinterfragen. Stellen Sie sich dabei vor, Sie müssten noch eine Klasse Musikkassette und eine Klasse DVD hinzufügen.

Notieren Sie hier Ihre Feststellungen:

Notizen

2 Lösung mit Vererbung

2.1 Analyse der falschen Lösung

Sie haben vielleicht festgestellt, dass Sie für die Erweiterung des Programmes sehr viele Attribute für die Klasse Book programmieren mussten. Die meisten Attribute sind bei allen Medienklassen vorhanden. Insbesondere sind auch die Getter und Setter ebenfalls mehrfach vorhanden.

Weiter haben Sie bei der Klasse `Database` Erweiterungen vornehmen müssen, die sehr viel Fleissarbeit erfordern. Zudem besteht eine grosse Wahrscheinlichkeit, dass bei der Erweiterung Fehler geschehen. Vor allem ist es möglich, dass Sie in der `Database` Klasse bereits getesteten Code ändern oder sogar in diesem Fehler einbauen.

Viele dieser Arbeiten erschweren die Wartbarkeit und die Erweiterbarkeit des Programmes. Das folgende Bild zeigt die **`Database`** Klasse mit und ohne Vererbung:

Database Klasse ohne Vererbung

```
public class Database {

	private ArrayList<CD> cds;
	private ArrayList<Video> videos;

	public Database() {
		cds = new ArrayList<CD>();
		videos = new ArrayList<Video>();
	}

	public void addCD(CD cd) {
		cds.add(cd);
	}

	public void addVideo(Video video) {
		videos.add(video);
	}

	public void printList() {
		// print list of CDs
		for (CD cd : cds) {
			cd.print();
		}

		// print list of videos
		for (Video video : videos) {
			video.print();
		}
	}
}
```

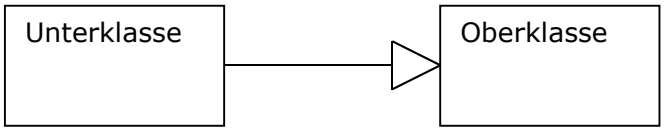
Database Klasse mit Vererbung

```
public class Database {

	private ArrayList<Medium> medien;

	public Database() {
		medien = new ArrayList<Medium>();
	}

	public void addItem(Medium medium) {
		medien.add(medium);
	}

	public void printList() {
		for (Medium medium : medien) {
			medium.print();
		}
	}
}
```

Wir wollen bei einer Erweiterung nur die neue Klasse programmieren und hinzufügen müssen. Der Rest des Programmes, beispielsweise die `Database` Klasse, sollte nicht angefasst werden.

2.2 Schrittweise zu einer besseren Lösung

Die wohl wichtigste Eigenschaft in der Objekt Orientierten Programmierung ist die Vererbung von Klassen. Klassen können von anderen Klasse erben. Dabei werden ihnen die Attribute und die Methoden vererbt.



Als ersten Schritt wollen wir eine Klasse erstellen, die alle gemeinsamen Attribute der Medienklassen vereint (generalisiert). Von dieser Klasse können dann die jetzigen und zukünftigen Medienklassen erben. Dies bedeutet, dass sie alle Attribute und Methoden der Oberklasse zur Verfügung haben und nur ihre speziellen Attribute selber einführen müssen.

In Java programmieren wir die Vererbung wie folgt:

```
public class CD extends Medium {
}
```

2.2.1 Aufgabe – neue Oberklasse

Programmieren Sie die Klasse **Media**. Die Klasse **Media** soll drei Eigenschaften besitzen, die für alle Unterklassen gelten sollen. Die Eigenschaften sind:

```
private String title;
private boolean gotIt;
private String comment;
```

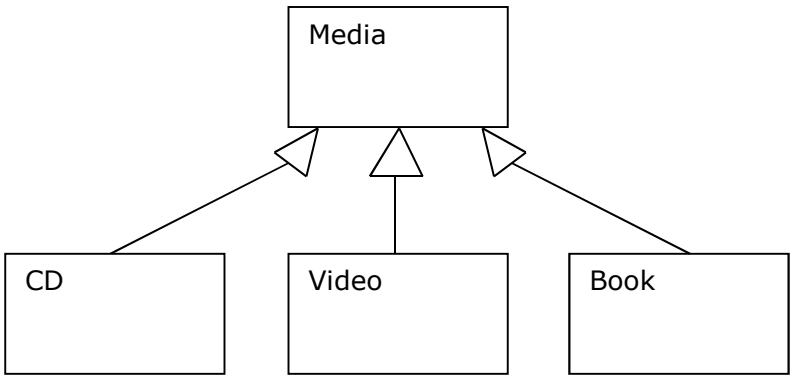
Die Klasse **Media** darf nur einen Konstruktor implementieren, welcher die Eigenschaften initialisiert und wie folgt aussehen muss:

```
public Media(String title, boolean gotIt, String comment) {
      this.title = title;
      this.gotIt = gotIt;
      this.comment = comment;
}
```

2.2.2 Aufgabe - Vererbung

Löschen Sie die gemeinsamen Attribute aus den Medienklassen **CD**, **Video** und **Book**. Programmieren Sie die Klassen so, dass sie von der Klasse **Media** erben.

Die nächste Abbildung zeigt die Situation (ohne Eigenschaften) grafisch:



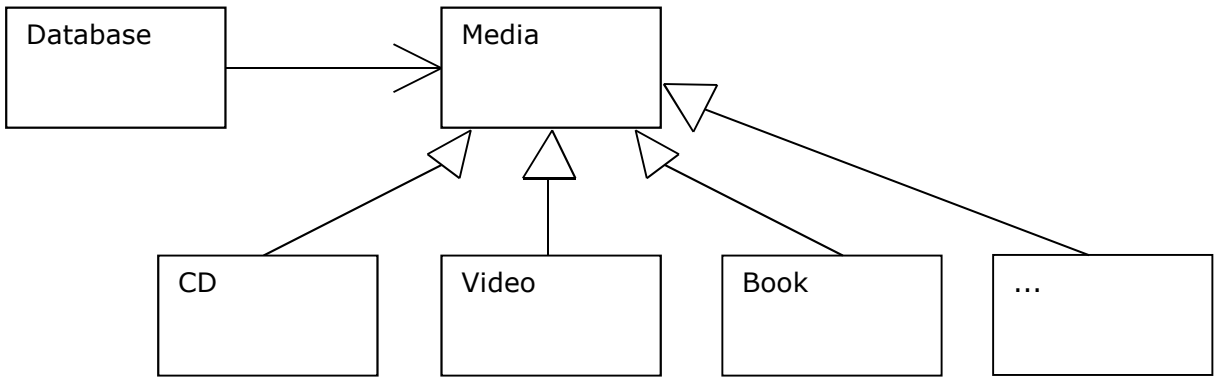
2.2.3 Aufgabe - Polymorphie

Bauen Sie die Klasse **Database** so um, dass sie nur noch mit der Klasse **Media** zu tun hat und nichts mehr von den spezifischen Medienklassen weiss. Die Klasse Database darf die Unterklassen nicht kennen.

```
public class Database {
	private ArrayList<Media> media;

	public Database() {
		media = new ArrayList<Media>();
	}
	public void addMedium(Medium medium) {
		media.add(medium);
	}
	public void printList() {
		for (Medium medium : media) {
			medium.print();
		}
	}
}
```

2.3 Endprodukt und Rahmenbedingungen

Die Klasse **Database** arbeitet nur noch mit der Klasse Media und hat keine Kenntnisse der Unterklassen. Dies ermöglicht ein späteres Erweitern des Programmes ohne die Klasse anpassen zu müssen. Theoretisch kann eine Klasse auch dynamisch, bei laufendem Programm, hinzugefügt werden.



2.3.1 Rahmenbedingungen zur Ausgabe

2.3.2 Einfache Ausgabe der Eigenschaften

Programmieren Sie die Ausgabe und das Zusammenspiel der `print` Methoden so, dass die folgende Ausgabe erzeugt wird. Zeigen Sie Ihre Lösung der Lehrperson.
Ausgabe:

```
**** My Media Library****
CD: Jones Norah (68 mins)
    tracks: 13
```

2.3.3 Verfeinerte Ausgaben der Eigenschaften

Programmieren Sie eine verfeinerte Ausgabe der Eigenschaften, sodass folgendes Resultat entsteht. Zeigen Sie Ihre Lösung der Lehrperson.

Ausgabe:

```
**** My Media Library****
CD: Jones Norah (68 mins)
    Fall
    tracks: 13
    excellent album!
```

2.4 Hinweise zu den Konstruktoren

Beim Lösen der Aufgaben werden Sie auf die eine oder andere Hürde stossen. Beispielsweise stellen die Formulierung des Konstruktors der Klasse Media, wie auch die Programmierung der Konstruktoren in den Unterklassen, solche Hürde dar.

Suchen Sie nach Lösungen, um die Struktur und den Aufbau des Programmes richtig zu bauen. Hierfür finden Sie in den folgenden Links diverse Erklärungen.

Links mit Hinweisen auf die Vererbung und die Handhabung von Konstruktoren:
http://openbook.galileodesign.de/javainsel5/javainsel06_000.htm#Xxx999378
http://openbook.galileocomputing.de/javainsel/javainsel_05_008.html → Kapitel 5.8.6
http://www.scalingbits.com/java/javakurs1/vererbung/java

http://www.youtube.com/watch?v=APhFL3Rv7B0

Notizen

2.5 Hinweise für das Methoden überschreiben

Bei der Ausgabe der einzelnen Eingenschaften müssen Sie nun ebenfalls eine geeignete Struktur resp. ein geeignetes Zusammenspiel zwischen Oberklasse und Unterklassen finden. Hierbei ist darauf zu achten, dass jede Klasse sich nur für die eigenen Eigenschaften kümmern muss.

http://openbook.galileocomputing.de/javainsel9/javainsel_05_010.htm → Kapitel 5.10
http://docs.oracle.com/javase/tutorial/java/IandI/override.html
http://docs.oracle.com/javase/tutorial/java/IandI/QandE/inherit-questions.html

Notizen

3 Übungen

Lesen Sie aus Kapitel 11 die Seiten 348 bis 357. Diese erklären die Vererbung in Java.
Beantworten Sie die folgenden Fragen und programmieren Sie die Übungen.
Wir wollen später in den Übungen eine Objekthierarchie mit Lernenden, Lehrpersonen und Schulklassen erstellen.

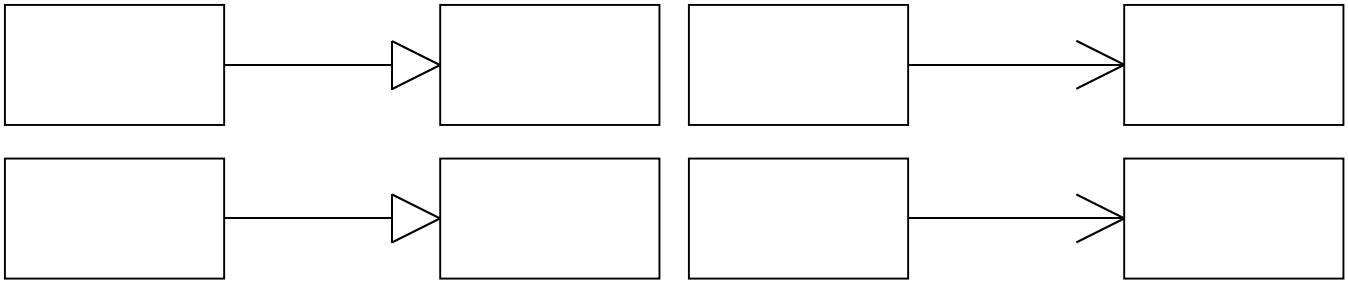
3.1 Aufgabe – Lesen

3.2 Aufgabe – Prinzip „Vererbung" verstehen

Nach der Lektüre: Wie könnte man das Prinzip Vererbung in einem Satz beschreiben?

3.3 Aufgabe – Symbole verstehen

Machen Sie zwei Beispiele für jede Diagrammsituation aus dem Bereich Schule, Lernende etc.



3.4 Aufgabe – Methoden überschreiben

Welche Kriterien müssen erfüllt sein, damit eine Methode als „überschrieben" gilt?

3.5 Aufgabe – Person, Student und Teacher

Erstellen Sie ein neues Projekt „SchoolOfJava".

Programmieren Sie für dieses Beispiel die folgenden Klassen mit ihren Beziehungen:

Klasse	Verantwortung
Person Name, Vorname, Geburtsdatum, Email	hält alle Informationen zu einer Person
Student SchoolClass, Matriculation Number	hält alle spezifischen Informationen zu einem Lernenden
Teacher Roomnumber	hält alle spezifischen Informationen zu einer Lehrperson
SchoolClass StudentList, ClassRepresentativ, MainTeacher	hält alle Informationen zu einer Klasse

3.6 Aufgabe – Fächer

Erweitern Sie die vorangegange Aufgabe mit einer Klasse Fächer und überlegen Sie sich die Beziehung zwischen der Klasse und den Fächern.

Klasse	Verantwortung
Subject (Fach) Name, NoOfLessions, Level	hält alle Informationen zu einem Fach

Seien Sie kreativ!

Notizen:

3.7 Aufgabe – Beziehungen

Erstellen Sie ein Hauptprogramm zur Aufgabe, welches die Objekte instanziert und zu einer funktionierenden Objekthierarchie zusammenbaut.

Nehmen Sie für die Klasse(n) und Lernenden echte Beispiele.

3.8 Aufgabe - Ausgabe

Überlegen Sie sich mit dem Prinzip der Methoden-Überschreibung wie Sie die Ausgabe steuern könnten.