Ergänzungsmaterial zum Lehrplannavigator

Allgemeine Objektorientierung

Möchte man sich mit objektorientierter Programmierung im Kontext eines Kurses der Einführungsphase befassen, ist es sinnvoll, sich zunächst einige grundlegende Gedanken zum objektorientierten Vorgehen in der Informatik zu machen. Objektorientierung bedeutet in diesem Kontext weit mehr als nur ein bestimmter Programmierstil bzw. eine spezielle Art und Weise, Quellcode zu produzieren. Vielmehr handelt es sich um eine umfassende Sicht- und Herangehensweise bei der Bearbeitung eines Problems oder eines Gegenstandsbereichs.

Im Grundsatz geht es darum, in einer Phase der Analyse und Modellierung einen realen Gegenstands- bzw. Problembereich mittels Abstraktion und Zusammenfassung in ein informatisches Modell zu überführen. Dieses Modell muss in seinem Komplexitätsgrad gegenüber der Wirklichkeit maßgeblich reduziert sein, ohne die Struktur des realen Gegenstandsbereiches bezogen auf die mit der Modellierung verbundene Aufgabe in unzulässiger Weise zu verzerren. Erst wenn eine solche Modellierung vorliegt, wird sie in Quellcode umgesetzt, welcher anschließend getestet werden kann.

Um sich mit grundlegenden Modellierungsaufgaben dieser Art befassen zu können, müssen zunächst einige Grundbegriffe geklärt werden. Dazu gehören in einem ersten Schritt die Begriffe *Objekt*, *Klasse*, *Attribut* und *Methode*. Da es sich hierbei um allgemeine Begriffe der Objektorientierung handelt, die in keiner speziellen Beziehung zu einer bestimmten Programmiersprache oder einem didaktischen Konzept stehen, sollten sie nicht nur programmiersprachenunabhängig, sondern auch komplett rechnerfrei behandelt werden.

Lernziele

Die Schülerinnen und Schüler sollen …

- … das Prinzip der Zerlegung eines Problembereichs in Objekte als Grundgedanken der Objektorientierung kennen lernen.
- … die Unterscheidung zwischen Objekt und Klasse erläutern können.
- … die Begriffe Methode und Attribut erläutern können.
- … einfache Modellierungen in Form von Klassendefinitionen und Beziehungsdiagrammen erstellen können.
- … das didaktisch vereinfachte Wasserfallmodell der Softwareentwicklung kennen lernen.

1 Sequenzskizze

Zur Verdeutlichung dieser Grundbegriffe gibt es eine Reihe geeigneter Kontexte, wobei der Grundgedanke immer darin besteht, im Gegenstandsbereich eine Ansammlung verschiedener Objekte zu sehen. Ein Bahnhof besteht aus *Gebäuden*, *Menschen* und *Zügen*, ein Ampelkreuzung aus *Fahrbahnen*, *Ampelanlagen* und *Fahrzeugen* und ein Schule aus *Lehrern*, *Schülern* und *Räumen*. Dabei zeigt sich bereits an diesen einfachen Beispielen der Charakter der Modellbildung, der maßgeblich darin besteht, in sinnvoller Weise zu abstrahieren. Welche Abstraktion dabei als sinnvoll zu betrachten ist, hängt von der Aufgabe ab, zu deren Lösung die Modellierung erstellt wird.

Betrachten wir im Folgenden einen einfachen Gegenstandsbereich. Auf einer grünen Wiese steht ein einzelner Baum, in dessen Ästen sich eine Reihe von Vögeln niedergelassen haben. Nehmen wir weiter an, dass eine graphische Simulation dieser Szene erstellt werden soll. Unsere erste Aufgabe besteht nun darin, Objekte im Gegenstandsbereich zu identifizieren.

Der Baum besteht aus einem Stamm, Ästen, Blättern und natürlich sind da die verschiedenen Vögel. Auf den Boden sind Grashalme. All diese Objekte könnten sich in unserer Modellierung wiederfinden, insbesondere an den Vogelobjekten lassen sich die Grundprinzipien der Objektorientierung aber gut verdeutlichen. Dabei sollten wir uns an dieser Stelle nicht allein

auf die Anforderungen dieser einfachen Aufgabe beschränken, sondern den Vogel auf eine solche Art und Weise modellieren, dass er auch in anderen ähnlichen Simulationsprojekten wiederverwendet werden kann.

Abbildung 1: Verschiedene Vögel [1]

Betrachtet man die in Abbildung 1 dargestellten Vögel, so fällt schnell auf, dass einzelne Vögel zwar sehr unterschiedlich sein können, prinzipiell aber immer vergleichbar sind. Es handelt sich durchgehend um flugfähige Tiere mit Federn, zwei Beinen, zwei Flügeln und einem Schnabel. In mancherlei Hinsicht unterscheiden sie sich aber auch. So ist ein Vogel womöglich etwas größer als ein anderer, hat eine andere Färbung des Gefieders oder ein anderes Geschlecht als sein Nachbar. Möchte man die drei Vögel in ihrer Individualität darstellen, so könnte das wie folgt aussehen.

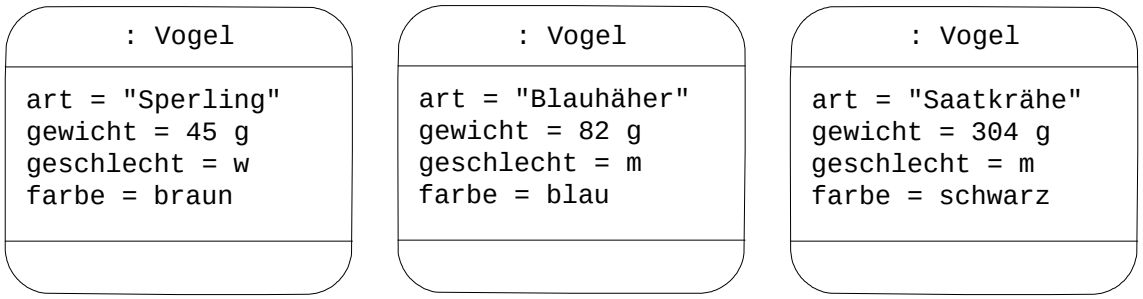

Abbildung 2: Objektdiagramm zu drei Vögeln

Alle Vögel sind im informatischen Sinne *Objekte* eines bestimmten *Objekttyps* bzw. einer bestimmten *Objektklasse*. Durch ihre Zugehörigkeit zu einem Objekttyp, in diesem Falle dem Typ *Vogel*, ist z.B. geklärt, dass es sich um Individuen mit Flügeln, Federn, Beinen und Schnabel handelt. Was jeden Vogel einzigartig macht, ist nicht mit dem Hinweis auf seine Typzugehörigkeit geklärt, sondern muss in jedem Vogelobjekt separat gespeichert werden.

[1] Quellenangabe: Wikipedia Foundation Inc.; URL 1: http://de.wikipedia.org/wiki/Haussperling; URL 2: http://de.wikipedia.org/wiki/Blauhäher; URL 3: http://de.wikipedia.org/wiki/Saatkrähe; Abgerufen: Dezember 2012.

Vogelobjekte verfügen also über *Attribute*, die unterschiedliche Werte haben können . In diesem Fall sind das die Attribute `art`, `gewicht`, `geschlecht` und `farbe`.

Dabei werden nicht alle Unterschiede, die im Gegenstandsbereich vorhanden sind, im Modell auch abgebildet. Das informatische Modell stellt eine Vereinfachung dar, die abhängig von der informatischen Aufgabenstellung von Aspekten der Wirklichkeit abstrahiert. Oder anders ausgedrückt: möchte man z.B. eine graphische Simulation der Vögel erstellen, so ist es nicht notwendig, den Blutdruck einzelner Vögel in den Objekten zu verwalten. Er mag sich von Vogel zu Vogel zwar unterscheiden, ist für die Simulation aber ohne Relevanz.

Um die Modellierung zu vervollständigen, sollten die Objekte vom Typ *Vogel* noch um Methoden ergänzt werden, d.h. um Aufträge, die ein Vogelobjekt ausführen kann und um Anfragen, bei deren Aufruf das Vogelobjekt eine Auskunft erteilt. Das kann z.B. eine Auskunft über die Belegung eines Attributs sein.

Methoden und Attribute – nicht jedoch deren Belegung – sind für alle Objekte vom Typ *Vogel* gleich, so dass es Sinn macht, von der Klasse *Vogel* als eine Art *Bauplan* für Vogelobjekte zu sprechen.

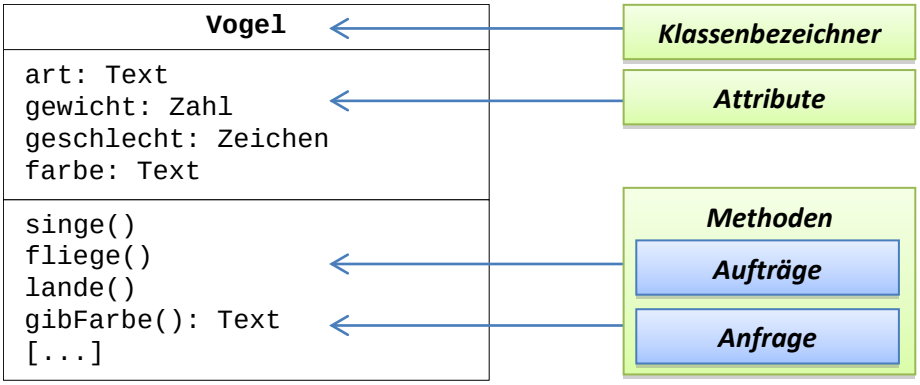

Abbildung 3: Klassendiagramm Vogel

Dieser Bauplan ist es, der letztlich programmiert wird. Aus ihm werden dann beliebig viele Objekte vom Typ *Vogel* erstellt.

Während die Modellierung einer Vogelklasse auf diese Weise sinnvoll erscheint, kann es andere Kontexte geben, bei der eine Klasse sich maßgeblich dadurch auszeichnet, dass sie aus anderen Klassen zusammengesetzt ist. Diese *Komposition* ist die einfachste Form der Klassenbeziehung und sollte bereits am Anfang thematisiert werden. So könnte man sich z.B. eine Klasse *Schwarm* vorstellen, wobei ein *Schwarm* sich aus Objekten vom Typ *Vogel* zusammensetzt. Der Schwarm selbst könnte dann wieder über Attribute und Methoden verfügen.

Spätestens an dieser Stelle sollte ein weiterer Aspekt der Objektorientierung angesprochen werden, der darin besteht, dass der Prozess der objektorientierten Bearbeitung eines Gegenstandsbereichs nicht sofort mit der Modellierung von Objekten und Klassen beginnen sollte. Der erste Schritt ist stets eine *Analyse* des Problembereichs. Mit anderen Worten: ein gewisses Grundverständnis der Funktionsweise eines Scharms und des Verhaltens sowie des Aufbaus eines Vogels muss vorhanden sein, um eine angemessene Modellierung durchzuführen. Während das in vielen Beispielkontexten, die in der Schule bearbeitet werden, noch recht einfach ist, nimmt diese Analysephase bei größeren Projekten immer mehr Raum ein.

Von einem eigentlichen Modellierungsprozess im informatischen Sinnen sind wir aber noch immer entfernt. Dazu müsste die Modellbildung zielgerichteter sein. Diese Zielsetzung bestimmt, von welchen Aspekten des Gegenstandsbereiches abstrahiert werden kann, ohne die Funktionsfähigkeit des zu erstellenden Softwareproduktes in Frage zu stellen. Bisher wurde nur gesagt, dass eine graphische Simulation erstellt werden soll, nicht aber, was sie alles leisten soll.

Schulische Projekte im Bereich der objektorientierten Modellierung und Programmierung sollten daher dem aus der Fachwissenschaft bekannten *Wasserfallmodell* der Softwareentwicklung in didaktisch reduzierter Form folgen.

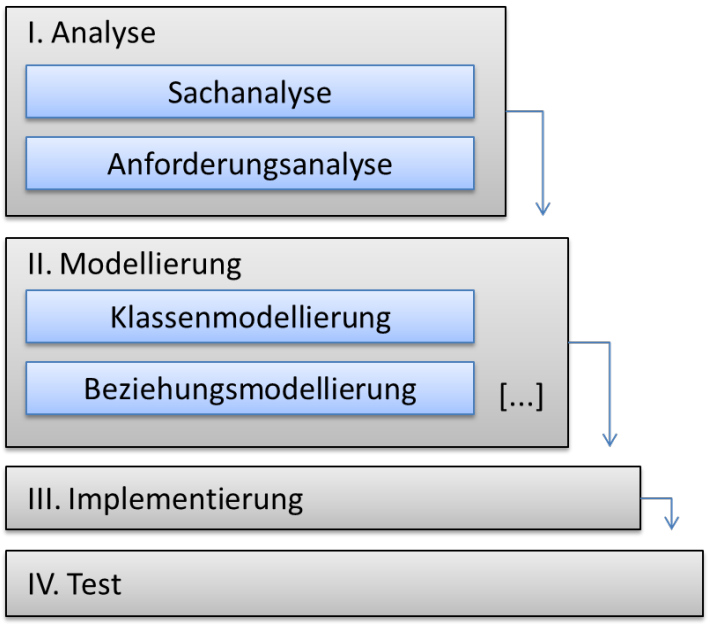

Abbildung 4: vereinfachtes Wasserfallmodell

1. ***Analyse***

   Gegenstandsbereich und Aufgabenstellung werden von den Schülerinnen und Schülern erfasst bzw. erarbeitet. Dazu gehört auch ein Anforderungsprofil für das zu erstellende Produkt.

2. ***Modellierung***

   Klassen und deren Beziehungen werden unter Berücksichtigung der Analysephase entworfen. Je nach Anforderungen des Projektes kann das ein einfaches Beziehungsdiagramm oder auch eine komplexere Modellierung bestehend aus *Entwurfsdiagramm* und *Implementationsdiagramm* sein.

3. **Implementierung**

   Die Modellierung wird unter Verwendung einer geeigneten didaktischen Umgebung in Quellcode umgesetzt. Dabei wird in vielen Fällen mit einem vom Kursleiter erstellten bzw. überarbeiteten Prototyp gearbeitet, der von den Kursteilnehmern ergänzt werden soll. Auf diese Weise werden zeitaufwändige Implementationsarbeiten ohne didaktischen Wert vermieden.

4. **Test**

   Der implementierte Quellcode wird hinsichtlich der in der Analysephase erarbeiteten Anforderungen getestet. Bei einfacheren Projekten geschieht dies nur implizit. Bei komplexeren Projekten können hier auch Qualitätskriterien wie z.B. das Laufzeitverhalten des Programms berücksichtigt werden.

2 Vertiefung

Soll die allgemeine Idee der Objektorientierung vertieft werden, so können beliebig viele Modellierungsaufgaben analog zum obigen Vogelbeispiel mit den Schülerinnen und Schülern bearbeitet werden. Natürlich können hier auch komplexere Kontexte zum Zuge kommen, bei denen eine angemessene Analyse erforderlich ist. Ein geeignetes Beispiel könnte ein Automotor sein, der im Kontext einer physikalisch realistischen Rennsimulation modelliert werden soll. Seine optische Erscheinung ist diesmal gegenstandslos. Nicht allen Schülerinnen und Schülern wird auf Anhieb klar sein, über welche Attribute und Methoden eine entsprechende Klasse verfügen sollte. Eine Anregung könnten hier die Karten eines klassischen Autoquartetts geben.

Da diese erste Sequenz aber lediglich den Grundgedanken der Objektorientierung zum Thema hat, sollte eine zeitaufwändige Vertiefung in der Regel nicht notwendig sein.