Ergänzungsmaterial zum Lehrplannavigator

Vererbung

In dieser Sequenz werden die wichtigsten Spielarten der Vererbung anhand von drei Projekten vorgestellt. Den Anfang macht das Projekt *Schneemannsimulation*, in dem die Vererbung als Spezialisierung im Sinne einer einfachen Erweiterung einer Oberklasse vorgestellt wird.

Darauf folgt das Projekt *Flummisimulation*, welches das Verständnis von Vererbung um den Aspekt der *späten Bindung* erweitert. Es werden also Dienste der Oberklasse überschrieben.

Zum Abschluss wird am Beispiel *Tannenbaumsimulation* das Prinzip der *abstrakten Klasse* eingeführt.

Im Einzelnen sollen die folgenden Lernziele erreicht werden.

Lernziele

Die Schülerinnen und Schüler sollen …

- … das Prinzip von **Ober**- und **Unterklasse** erfasst haben.
- … das Prinzip der Vererbung am Beispiel von **Spezialisierung** und **Generalisierung** kennen lernen.
- … selbst modellierte **Unterklassen in Java** umsetzen können.
- … das Prinzip der **späten Bindung** im Kontext des Überschreibens von Methoden erfasst haben.
- … eigene **abstrakte Klassen** und abstrakte Methoden verwenden können.[1]
- … das Prinzip der **dynamischen Referenzierung** und Objektselektion erfasst haben.[2]

[1] Dieses Lernziel geht über die Vorgaben des Kernlehrplans Informatik hinaus und muss daher nicht zwingend erfüllt werden.
[2] Ebenfalls ein Lernziel das über die Vorgaben des Kernlehrplans hinaus geht.

1 Sequenzskizze

Der Einstieg in dieses Modul erfolgt mit Hilfe der Simulation eines einzelnen Schneemanns. Dieser kann entweder als Übung von den Schülerinnen und Schülern selbst erstellt oder aber als Prototyp vom Lehrernden vorgegeben werden.

1.1 Das Projekt *Schneemannsimulation*

Von der Modellierung her ist entscheidend, dass der Schneemann als eigene Klasse realisiert wird, so dass ggf. auch mehrere Schneemänner in einer Szene erstellt werden können. Der Schneemann selbst besteht aus drei Kugeln, zwei Augen, einer Nase und hat ggf. Knöpfe auf der Brust. Die *Abbildung 1* zeigt einen einfachen Schneemann ohne Knöpfe.

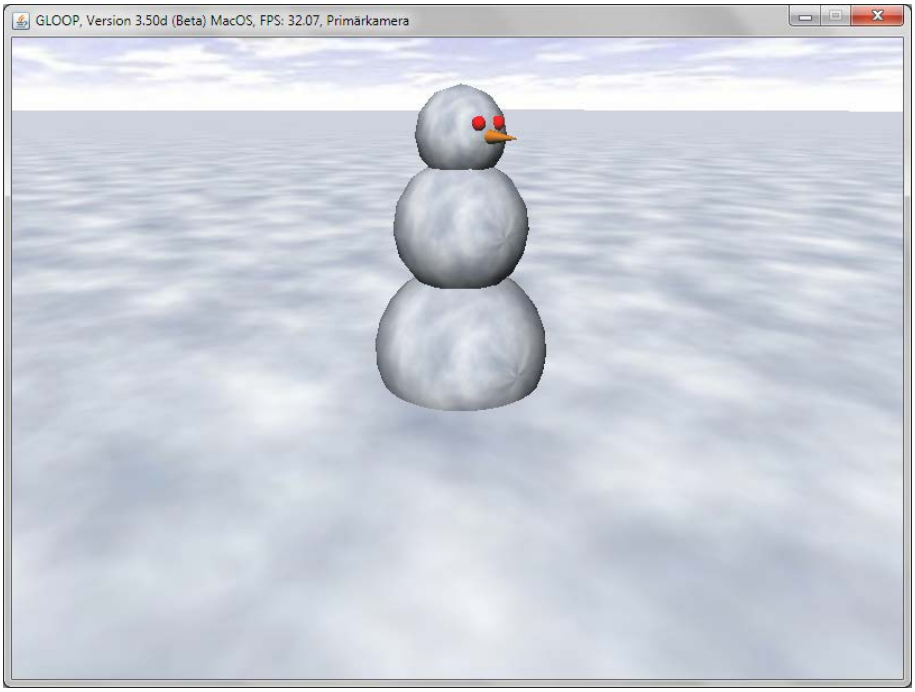

Abbildung 1: Ein einfacher Schneemann als Oberklasse

Da der Schneemann über einen Konstruktor verfügt, der ihn an unterschiedliche Stellen in der Szene positionieren kann, ist es ohne Probleme möglich, auch mehrere Schneemänner zu realisieren.

Sinn dieses Projektes ist es nun, eine solche Gruppe von Schneemännern zu erstellen, die jedoch nicht alle ganz identisch aussehen. Vielmehr soll eine internationale Schneemanngruppe erstellt werden, wobei sich die Schneemänner aus unterschiedlichen Erdteilen durch ihre Kopfbedeckung unterscheiden.

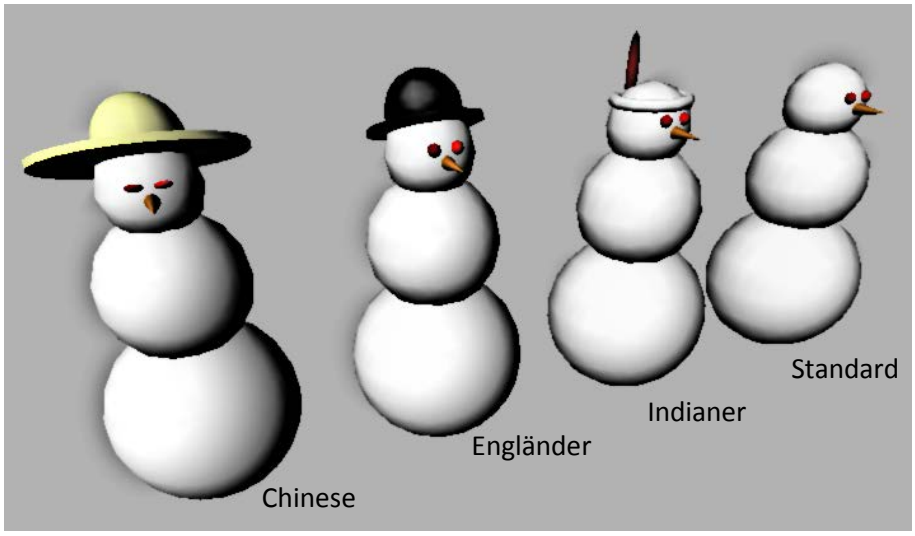

Abbildung 2: Internationale Schneemänner durch Vererbung

Die Realisierung dieser Schneemänner kann nun mit Hilfe der Vererbung geschehen, indem im Konstruktor der Unterklasse die jeweilige Kopfbedeckung ergänzt wird. Es wird die folgende Modellierung vorausgesetzt.

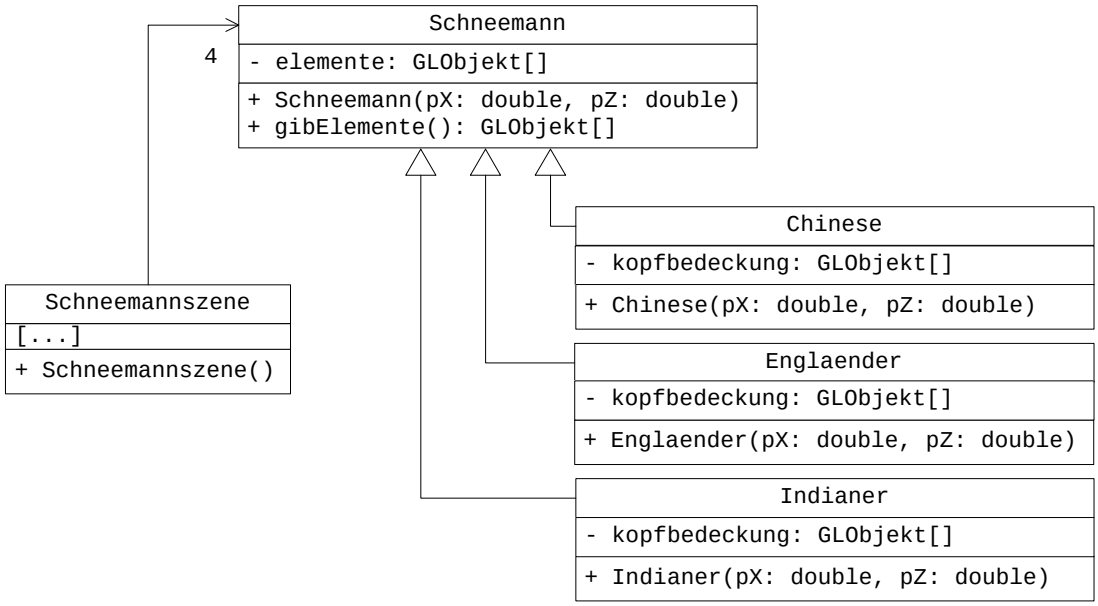

Abbildung 3: Modellierung der Schneemänner

Möchte man diese Modellierung mit den Schülerinnen und Schülern zusammen erarbeiten, so bietet es sich an, diese Aufgabe zunächst ohne den Mechanismus der Vererbung zu planen. Man kommt dann zu dem Ergebnis, dass verschiedene Schneemanntypen verschiedene Klassen mit großen Redundanzen sind oder aber, dass eine Kopfbedeckung eine eigene Klasse sein sollte. Eine andere Realisierung stellt die oben gegebene Modellierung dar. Jeder Schneemann verfügt über ein Feld von Typ `GLObjekt`, in dem seine Elemente gespeichert werden. Jede Unterklasse von `Schneemann` hat ein weiteres Feld, das seine Kopfbedeckung

beinhaltet. Da das Feld `elemente` der Oberklasse Schneemann `privat` ist und zumindest für den Chinesenschneemann ein Zugriff auf die Augen möglich sein muss, hat die Klasse `Schneemann` eine Methode `gibElemente`, die das Feld zurückliefert und es somit manipulierbar macht. Der Chinesenschneemann kann so seine Augen zu Schlitzen skalieren.

Zu berücksichtigen ist, dass an dieser Stelle die Vererbung gleich in zweifacher Hinsicht verwendet wird. Zum einen erben die speziellen Schneemänner von ihrer Oberklasse und zum anderen wird die Klasse `GLObjekt` als Oberklasse für alle in der Szene zu sehenden Elemente verwendet. Sowohl Kugeln als auch Zylinder, Kegel oder Toren können also in einem gemeinsamen Feld verwaltet werden.

Hat man die Vererbung auf diese Weise thematisiert, kann ein komplexeres Projekt in Angriff genommen werden, bei dem es notwendig wird, eine Methode zu überschreiben. Das Prinzip des *late-bindings* lässt sich gut am Projekt *Flummisimulation* verdeutlichen.

1.2 Das Projekt *Flummisimulation*

Das Projekt beginnt mit einem einfachen Prototyp. Dieser Prototyp lässt einen einzelnen Ball zwischen zwei Hindernissen hin- und herticken. Er folgt der in *Abbildung 4* gezeigten Modellierung.

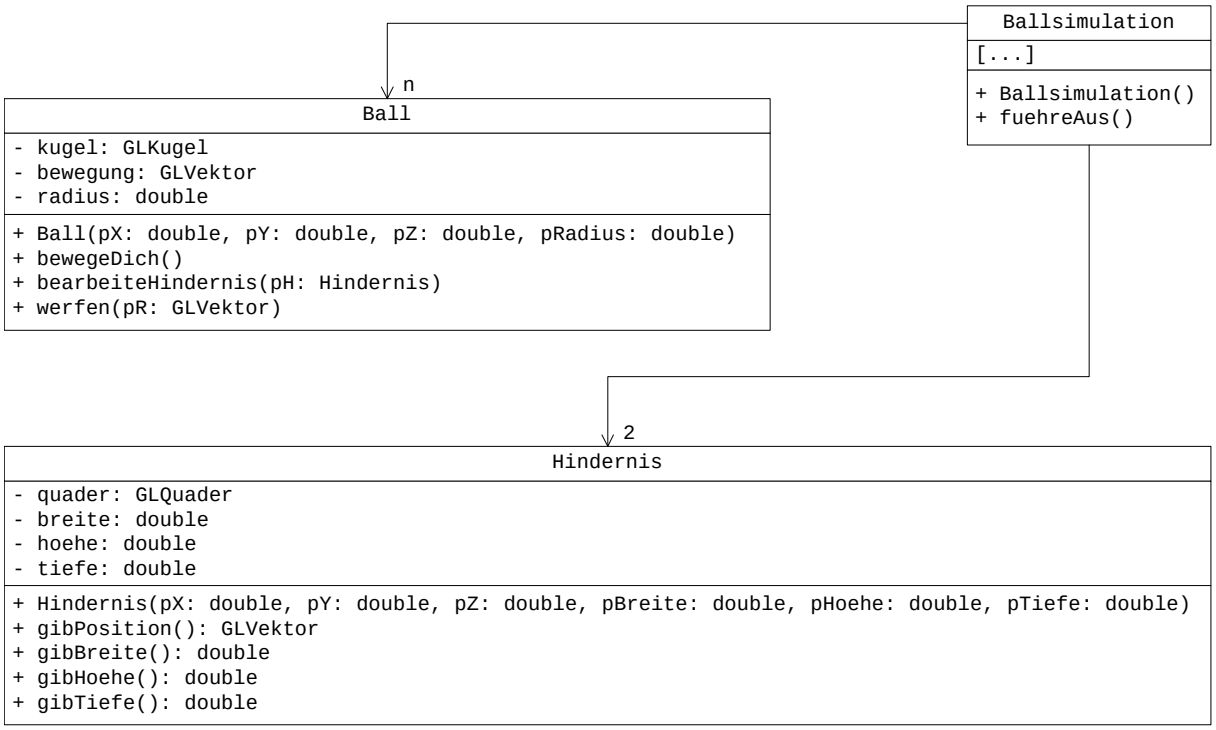

Abbildung 4: Modellierung der Flummisimulation (Prototyp)

Dieser Prototyp kann nun erweitert werden, indem spezielle Varianten des Balls ergänzt werden. Dazu wird jeweils eine Unterklasse von `Ball` erstellt. Da die unterschiedlichen Ballvarianten sich nicht nur im Aussehen unterscheiden sollen, muss auch die Bewegung des Balls variiert werden. Um den jeweiligen Ball zu bewegen, wird die Methode `bewegeDich` in einer Animationsschleife der Klasse `Ballsimulation` aufgerufen. Je nachdem wie diese Methode realisiert ist, können unterschiedliche Bewegungsvarianten realisiert werden. Folgende Modellierung setzt verschiedene Ballvarianten um.

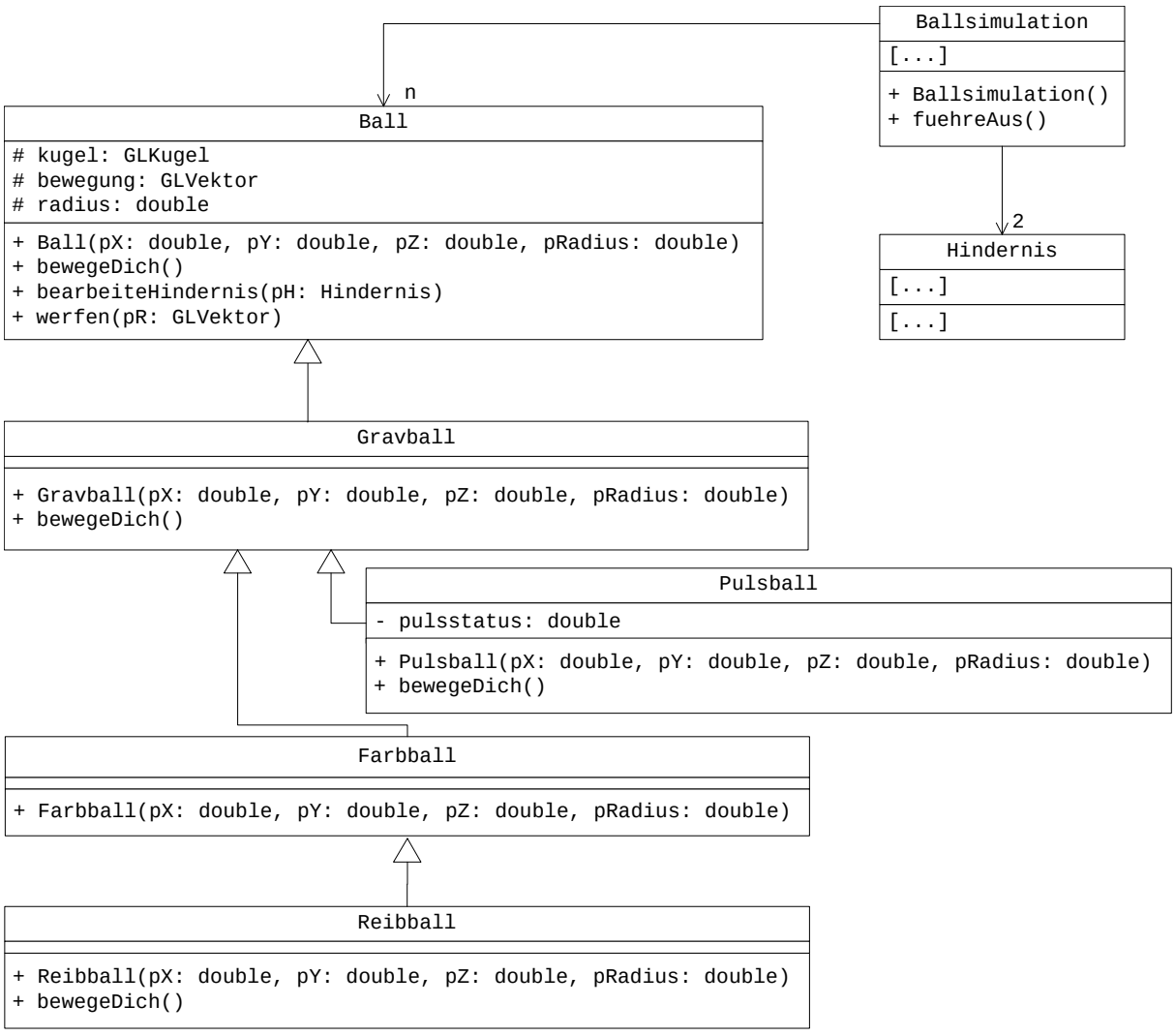

Abbildung 5: Modellierung der Flummisimulation (Modelllösung)

Die Klasse `Gravball` steht für einen Ball, der durch die Gravitation angezogen wird und vom Boden der Szene abtickt. Ein `Farbball` hat eine Zufallsfarbe und ein `Reibball` wird mit der Zeit langsamer, so dass er liegen bleibt. Der `Pulsball` verändert während der Animation seine Größe, er pulsiert. Alle Attribute der Klasse `Ball` werden als `protected` deklariert, damit ein Zugriff aus den Unterklassen möglich ist.

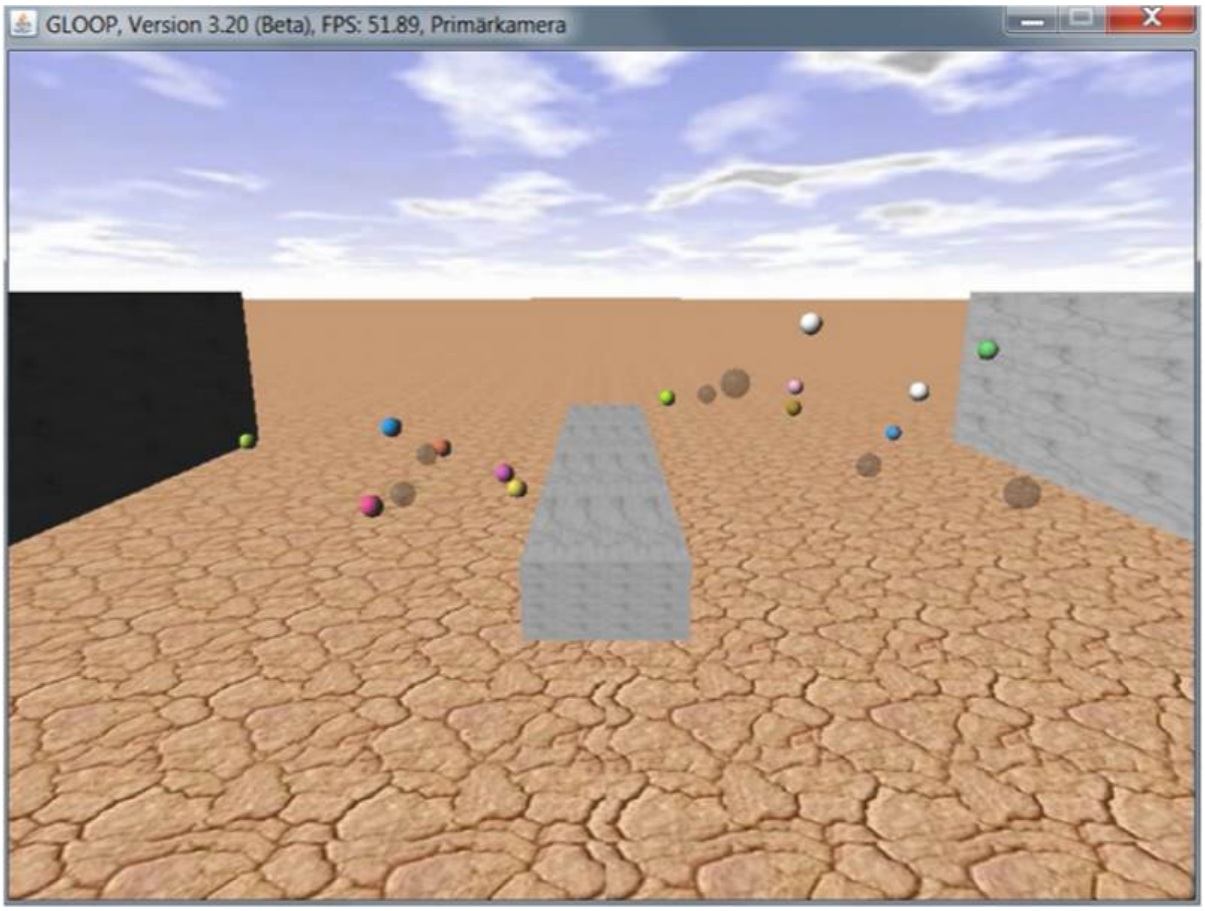

Abbildung 5: Flummisimulation mit verschiedenen Bällen

Alle Bälle können mit Hilfe eines Feldes vom Typ `Ball` in der Klasse `Ballsimulation` verwaltet werde. Da die Methode `bewegeDich` zum Teil überschrieben wird, sind sie nicht nur in ihrem Aussehen, sondern auch in ihrem Verhalten unterschiedlich.

Die hier aufgeführten Unterklassen von `Ball` sind exemplarisch zu verstehen. Natürlich sind noch weiteren Varianten möglich. Darüber hinaus könnten unterschiedliche Arten von Hindernissen erstellt werden, die sich bei einer Kollision mit einem Ball unterschiedlich verhalten.

2 Vertiefung: Das Projekt Weihnachtsbaumsimulation

Um tiefer in das Prinzip der Vererbung einzusteigen, wenden wir uns einer Simulation eines Weihnachtsbaums zu. Dabei soll ein Tannenbaum mit verschiedenen Arten von Schmuckstücken versehen werden, die durch unterschiedliche geometrische Objekte dargestellt werden. Es gibt Kugeln, Päckchen in der Form von Würfeln und Zuckerringe in Form von Toren.

Wir beginnen mit einem Prototyp, der bereits mit Kugeln geschmückt werden kann. Die Klasse `Kugel` hat dabei zwei Methoden.

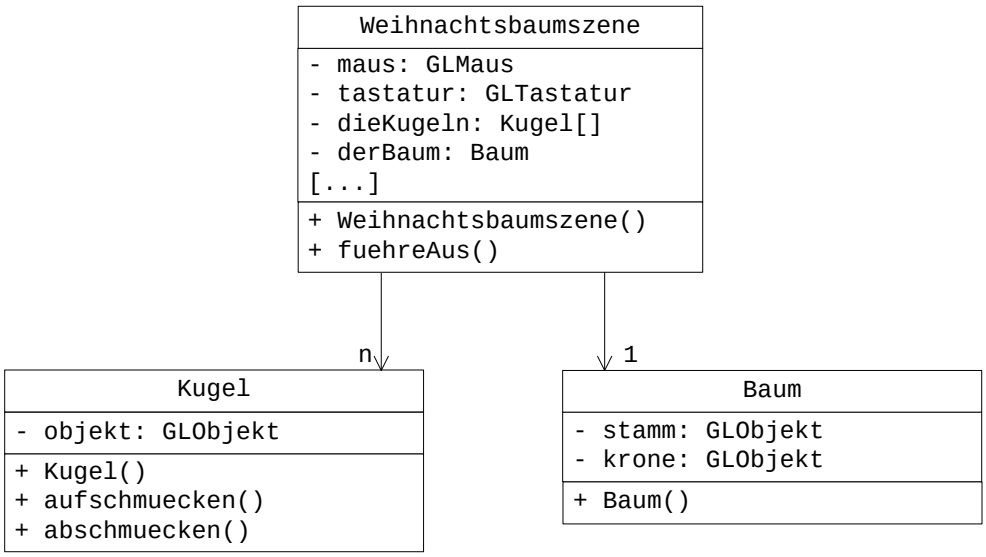

Abbildung 6: Modellierung I der Weihnachtsbaumsimulation (Prototyp)

Die Methode `abschmuecken` setzt die Y-Koordinate der Kugel auf `0`, so dass die Kugel nicht mehr am Baum ist, sondern auf dem Boden liegt. Die Kugel wird also abgeschmückt. Die Methode `aufschmuecken` positioniert die Kugel wieder an einer zufälligen Stelle auf den Baum. Sie wird also aufgeschmückt.

Der Prototyp soll auf dem Hintergrund des Prinzips der Vererbung so erweitert werden, dass die anderen Arten der Schmuckstücke ebenfalls eingebaut werden.

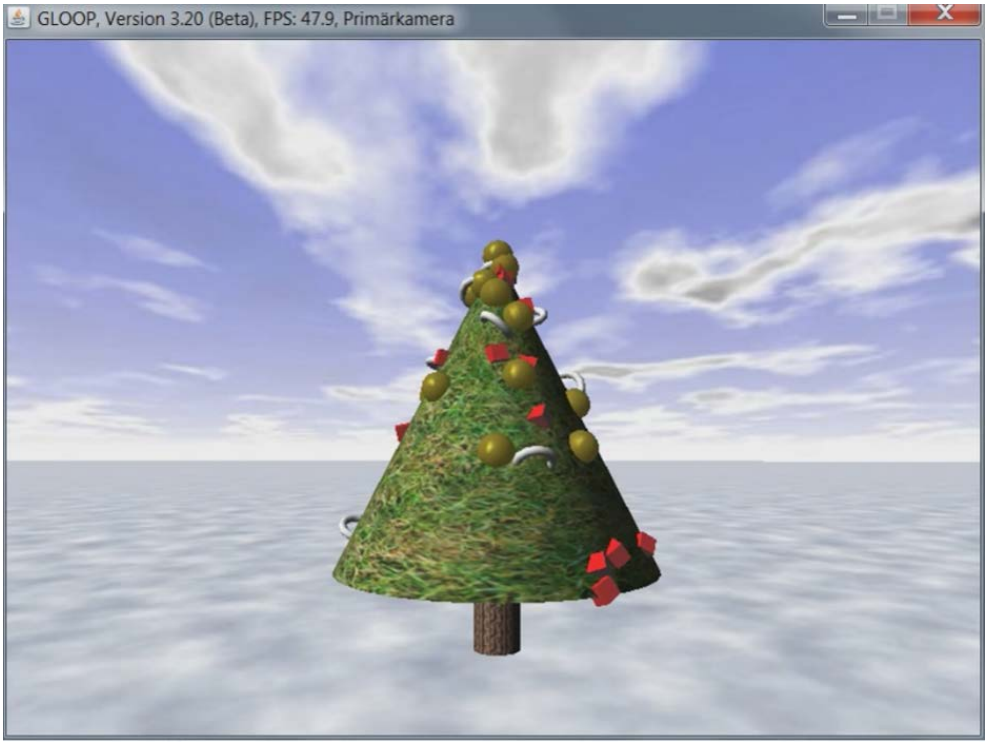

Abbildung 7: Weihnachtsbaumsimulation mit verschiedenen Schmuckstücken

Da alle Schmuckstücke über die Funktion des Auf- und Abschmückens verfügen sollen, liegt es nahe, dass diese Methoden in einer gemeinsamen Oberklasse realisiert werden.

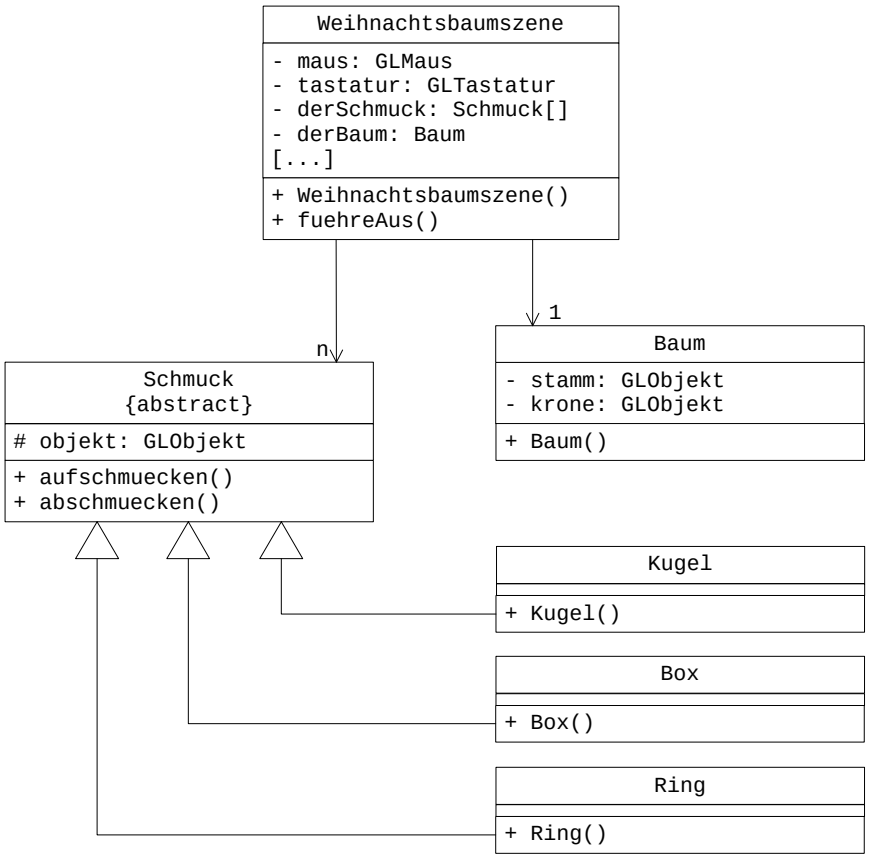

Abbildung 8: Modellierung der Weihnachtsbaumsimulation mit abstrakter Klasse

Die in *Abbildung 8* zeigt eine Modellierung in der die Klasse `Schmuck` bereits als abstrakt gekennzeichnet ist. Kursteilnehmer, die diese Aufgabe bearbeiten, werden das zunächst nicht so modellieren, sicherlich aber feststellen können, dass es keinen Sinn macht, die Klasse `Schmuck` zu instanziieren, da im Konstruktor der Klasse noch nicht entschieden werden kann, welches Objekt zur Repräsentation des Schmuckstücks erstellt werden soll. Sie beinhaltet also den Quellcode zum Auf- und Abschmücken, hat aber kein optisches Erscheinungsbild, so dass ein Objekt keinen Sinn macht. Die Referenz `objekt` der Klasse `Schmuck` kann als `protected` deklariert werden, damit in den Unterklassen ein Zugriff möglich ist. Alternativ kann eine Methode zum Setzen dieser Referenz hinzugefügt werden.

Der nächste Schritt besteht darin, dass den Schmuckstücken eine zusätzliche Animation hinzugefügt wird, die durch einen Mausklick angestoßen werden kann. Hierzu wird eine neue Methode in die (abstrakte) Klasse `Schmuck` eingeführt, die erst in den Unterklassen realisiert werden soll. Diese Methode ist abstrakt, da in der Klasse `Schmuck` lediglich die Signatur definiert werden kann. Die Modellierung sieht dann wie in *Abbildung 9* gezeigt aus.[3]

[3] Abstrakte Methoden sind im Kernlehrplan Informatik erst in der Qualifikationsphase vorgesehen. Daher kann hier die Klasse `Schmuck` auch als nicht-abstrakte Klasse mit der leeren Methode `spezialAktion()` realisiert werden.

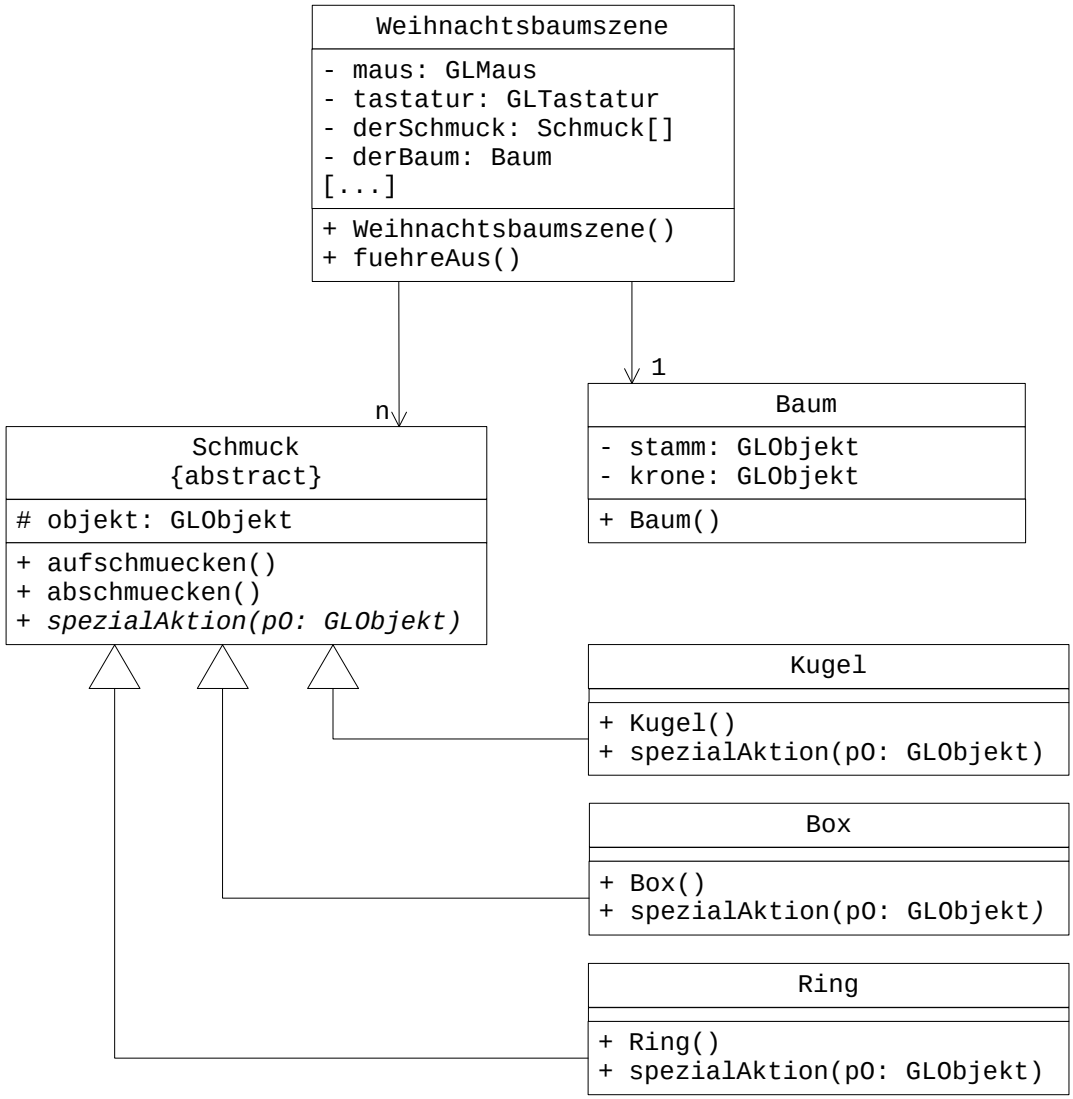

Abbildung 9: Modellierung der Weihnachtsbaumsimulation mit abstrakter Methode

In der Animationsschleife der Klasse `Weihnachtsbaumszene` wird mittels der GLOOP-Objektselektion das von der Maus angeklickte `GLObjekt` ermittelt und im Parameter der Methode `spezialAktion` an jedes Schmuckstück übergeben. In dieser Methode entscheidet jedes Schmuckstück, ob das übergebene `GLObjekt` jenes ist, mit dem es sich selbst optisch repräsentiert. Ist das der Fall, muss die jeweilige Spezialaktion in Form einer Animation durchgeführt werden. Die Hauptschleife der Klasse `Weihnachtsbaumszene` sieht dann wie folgt aus:

```
while (!tastatur.esc()){
  if (tastatur.oben())
    for (int i=0; i<50; i++) derSchmuck[i].aufschmuecken();
  if (tastatur.unten())
    for (int i=0; i<50; i++) derSchmuck [i].abschmuecken();

  GLObjekt aktuell = null;
  if (maus.linksklick()){
    aktuell = Sys.gibObjekt(maus.gibX(),maus.gibY());
    for (int i=0; i<50; i++) derSchmuck [i].spezialAktion(aktuell);
  }
  Sys.warte();
}
```

Welche Animation die jeweiligen Schmuckstücke beim Anklicken durchführen, ist für die Modellierung nicht von Relevanz. Ein Ring könnte sich einmal um sich selbst drehen, ein Päckchen die Farbe wechseln oder auch pulsieren. Eine Kugel könnte vom Baum fallen.

3 Materialien

1. P10_Schneemannszene_Vererbung
2. P11_Ballsimulation_Vererbung
3. P12_Weihnachtsbaum_Vererbung