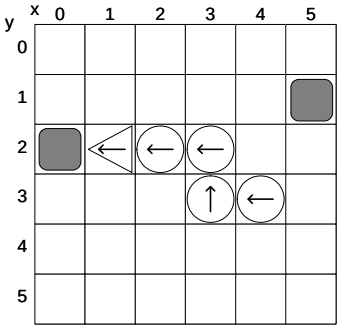
Klassendiagramme

Klassendiagramme beschreiben die vorhandenen Klassen mit ihren Attributen und Methoden sowie die Beziehungen der Klassen untereinander.

Im Folgenden soll eine Variante eines Computerspiels (Darstellung: Schlange auf einem Spielfeld) modelliert werden:



Klassen

Klassen werden durch Rechtecke dargestellt, die entweder nur den Namen der Klasse tragen oder zusätzlich auch Attribute und / oder Methoden enthalten. Attribute und Methoden können zusätzliche Angaben zu Parametern und Sichtbarkeit (`public (+)`, `private (-)`) besitzen.

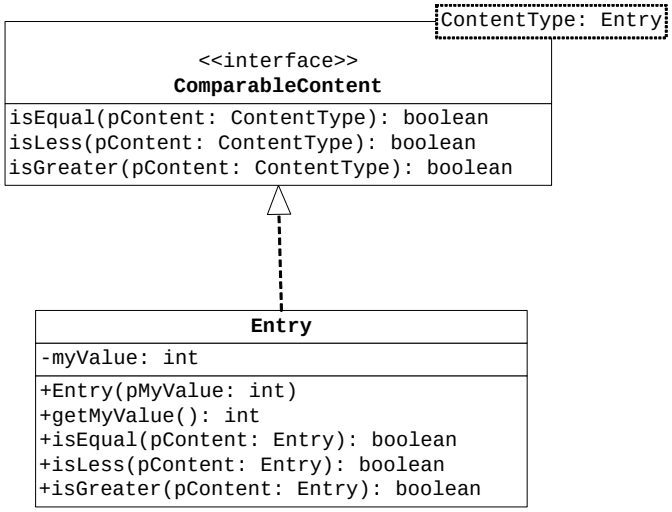
Wurmglied
-xKoordinate: int -yKoordinate: int -richtung: int
+Wurmglied(pX: int, pY: int, pRichtung: int) +rueckeVor() +gibRichtung(): int +gibXKoordinate(): int +gibYKoordinate(): int +setzeRichtung(pRichtung: int) +gibNaechstesX(): int +gibNaechstesY(): int

Bei **abstrakten Klassen**, also Klassen, von denen kein Objekt erzeugt werden kann, wird unter den Klassennamen im Diagramm `{abstract}` geschrieben. Abstrakte Methoden, also Methoden, für die keine Implementierungen angegeben werden und die nicht aufgerufen werden können, werden in Kursivschrift dargestellt. Bei einer handschriftlichen Darstellung werden sie mit einer Wellenlinie unterschlängelt.

Element **{abstract}**
+istGleich(pElement: Element): boolean *+istKleiner(pElement: Element) boolean* +istGroesser(pElement: Element): boolean

Bei diesem Beispiel sind die Methoden `istGleich` und `istKleiner` abstrakt. Die Methode `istGroesser` ist mit Hilfe der abstrakten Methoden implementiert.

Ein **Interface** (Schnittstelle) enthält nur die Spezifikationen von Methoden, nicht aber deren Implementation. Die Implementation einer Schnittstelle wird durch eine gestrichelte Linie mit geschlossener, nicht ausgefüllter Pfeilspitze dargestellt.



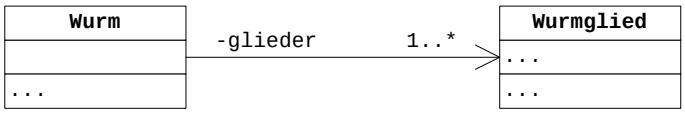
Beziehungen zwischen Klassen

Assoziation

Eine gerichtete Assoziation von einer Klasse A zu einer Klasse B modelliert, dass Objekte der Klasse B von einem Objekt der Klasse A verwaltet werden bzw. verwaltet werden können. Bei einer Assoziation kann man angeben, wie viele Objekte der Klasse B von einem Objekt der Klasse A verwaltet werden bzw. verwaltet werden können. Die Zahl nennt man **Multiplizität**.

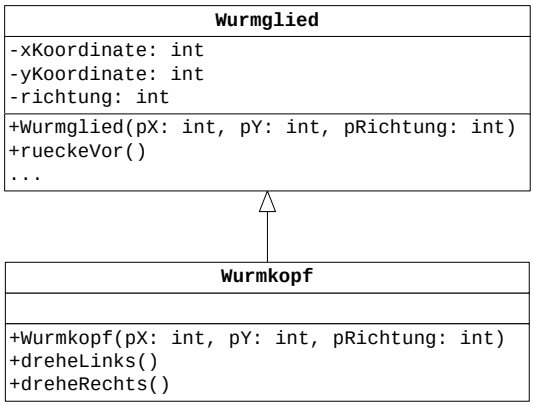
Mögliche Multiplizitäten:

- `1` genau ein assoziiertes Objekt
- `0..1` kein oder ein assoziiertes Objekt
- `0..*` beliebig viele assoziierte Objekte
- `1..*` mindestens ein, beliebig viele assoziierte Objekte



Ein Objekt der Klasse `Wurm` verwaltet mindestens ein Objekt der Klasse `Wurmglied`.

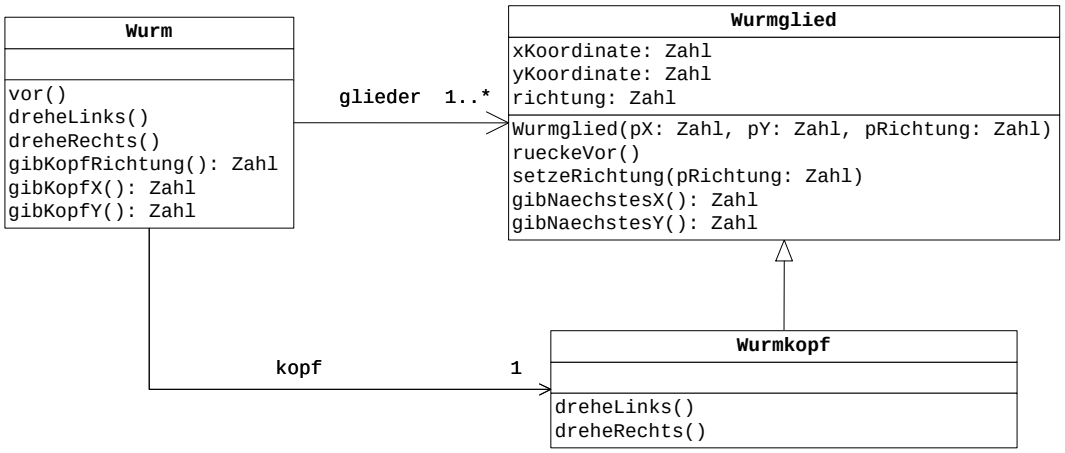
Vererbung
Die Vererbung beschreibt die Beziehung zwischen einer allgemeineren Klasse (Oberklasse) und einer spezialisierten Klasse (Unterklasse). Der Unterklasse stehen alle öffentlichen Attribute und alle öffentlichen Methoden der Oberklasse zur Verfügung. In der Unterklasse können Attribute und Methoden ergänzt oder auch Methoden der Oberklasse überschrieben werden.



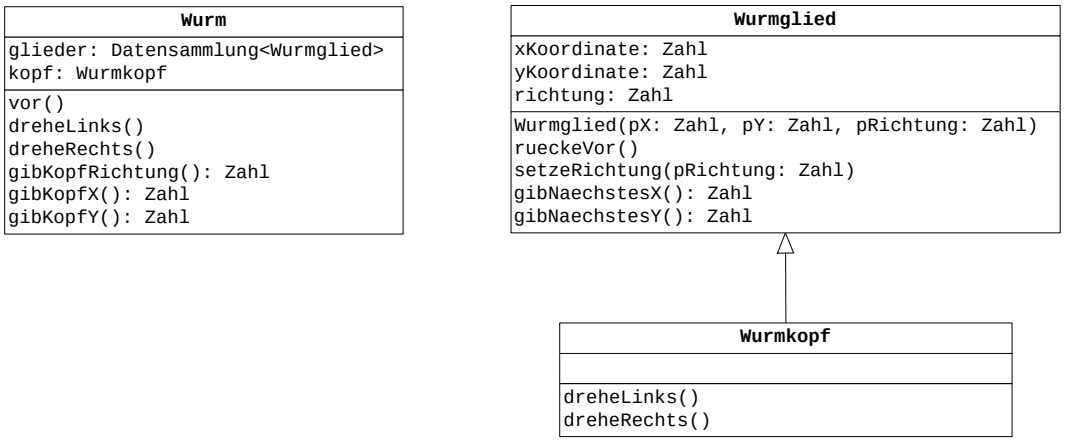
Die Klasse `Wurmkopf` spezialisiert hier die Klasse `Wurmglied`.

Entwurfsdiagramm
Bei einem Entwurf werden die in der Auftragssituation vorkommenden Objekte identifiziert und ihren Klassen zugeordnet.
Das Entwurfsdiagramm enthält Klassen und ihre Beziehungen mit Multiplizitäten. Als Beziehungen können Vererbung und gerichtete Assoziationen gekennzeichnet werden. Gegebenenfalls werden wesentliche Attribute und / oder Methoden angegeben. Die Darstellung ist programmiersprachenunabhängig ohne Angabe eines konkreten Datentyps, es werden lediglich `Zahl`, `Text`, `Wahrheitswert` und `Datensammlung<·>` unterschieden. Bei der Datensammlung steht in Klammer der Datentyp oder die Klassenbezeichnung der Elemente, die dort verwaltet werden. Anfragen werden durch den Datentyp des Rückgabewertes gekennzeichnet.

Version 1: Assoziationspfeile mit Multiplizitäten



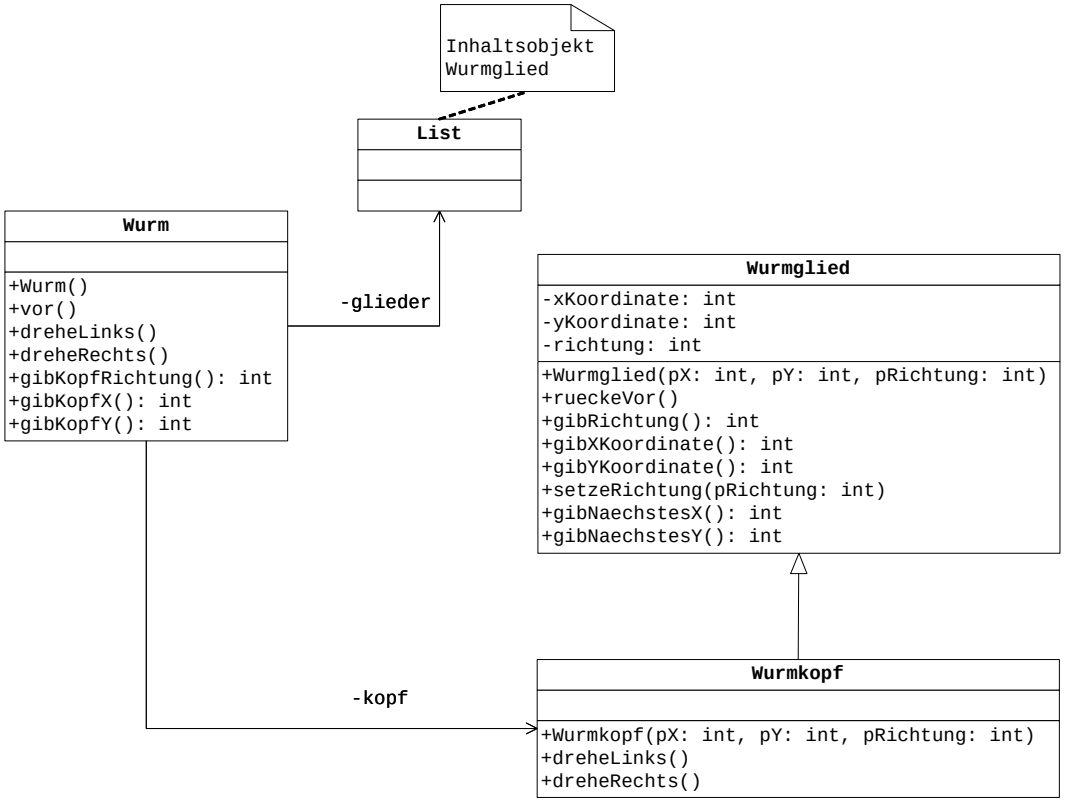
Version 2: Attribute mit Datensammlung



Beide Darstellungen drücken den gleichen Sachverhalt aus.

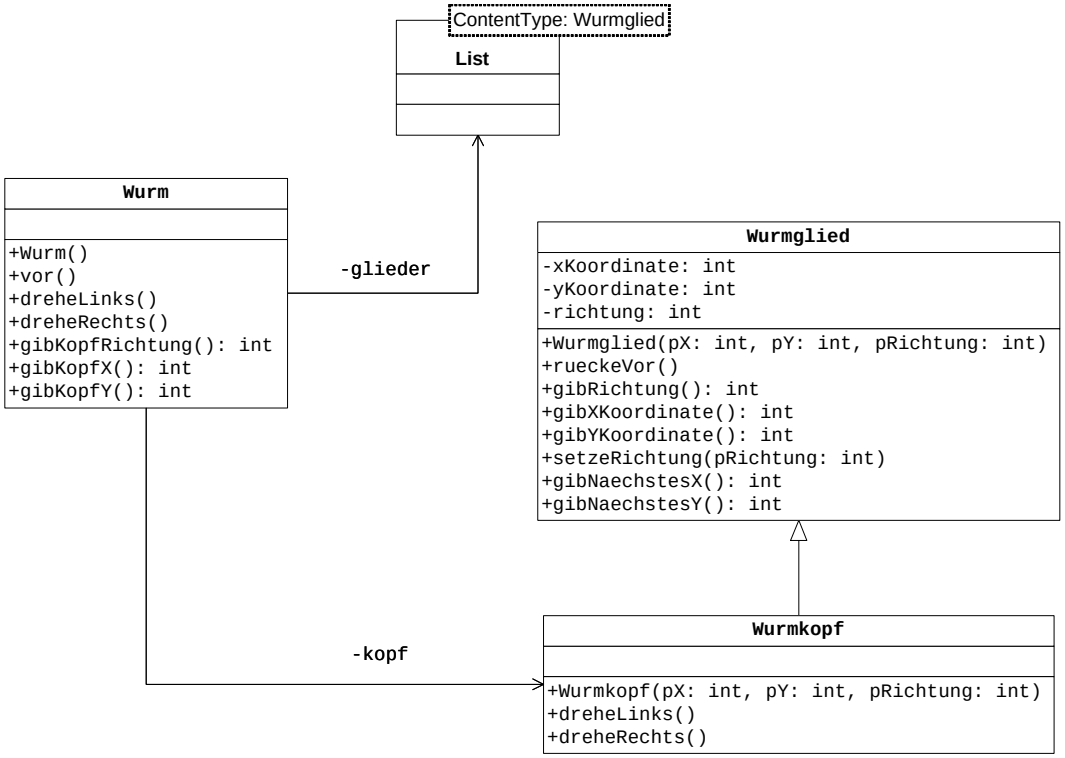
Implementationsdiagramm
Ein Implementationsdiagramm ergibt sich durch Präzisierung eines Entwurfsdiagramms und orientiert sich stärker an der verwendeten Programmiersprache. Für die im Entwurfsdiagramm angegebenen Datensammlungen werden konkrete Datenstrukturen gewählt, deren Inhaltstypen in Form von Kommentaren oder als Parameter angegeben werden. Die Attribute werden mit den in der Programmiersprache (hier Java) verfügbaren Datentypen versehen und die Methoden mit Parametern einschließlich ihrer Datentypen.
Bei den für das Zentralabitur dokumentierten Klassen (`List, BinaryTree, …`) wird auf die Angabe der Attribute und der Methoden verzichtet.

Beispiel für ein Implementationsdiagramm mit Assoziationen und Vererbung
Version 1: Die Klasse `List` verwaltet allgemein Inhaltsobjekte vom Typ `Object`.



Erläuterung:
Bezeichner der Attribute, durch die die Assoziationen realisiert werden, stehen an den Pfeilen.
Objekte der Klasse `List` verwalten Objekte der Klasse `Object`. Der Kommentar an der Klasse `List` verdeutlicht, welche Inhaltsobjekte in der Klasse `List` hier verwendet werden sollen.

Beispiel für ein Implementationsdiagramm mit Assoziationen und Vererbung Version 2: Die Klasse `List` ist eine generische Klasse (Template).



Erläuterung:
Bezeichner der Attribute, durch die die Assoziationen realisiert werden, stehen an den Pfeilen.
Bei der Klasse `List` handelt es sich um eine generische Klasse, über Parameter wird der Datentyp der Inhaltsobjekte angegeben. In der Klasse `List` werden Objekte der Klasse `Wurmglied` verwaltet.