Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. M. Wirsing, M. Barth, F. Birzele, G. Zhang

Übungen zu Einführung in die Informatik (Lösungsvorschlag)

Aufgabe 7-1 **Lernfragen** (keine Abgabe)

a) Vergleichen Sie Arrays und die Klasse `ArrayList` in einer kleinen Gegenüberstellung.

Lösung

Array ist eine Datenstruktur, die eine feste Länge in ihrer Lebensdauer hat. Die Länge eines Arrays wird beim Initialisieren festgelegt und bleibt in der gesamten Lebensdauer des Arrays unverändert. Arrays werden verwendet, um Reihungen zu implementieren, deren maximale Länge a priori festgestellt werden kann. Das ist beispielsweise dann der Fall, wenn in einem UML-Klassendiagramm die Obergrenze der Multiplizität eines Assoziationsenden nicht `*` ist. Auf einzelne Elemente eines Arrays wird mittels der eckigen Klammern zugegriffen. Die Länge eines Arrays `a` gibt der Ausdruck `a.length` zurück.

Arrayliste ist eine spezielle Liste. Liste ist eine Datenstruktur, deren Länge im Laufe der Zeit varieren kann (mehr dazu im weiteren Verlauf des Semesters). Arrylisten können verwendet werden, um Assoziationen mit Multiplizitäten, deren Obergrenze `*` ist, zu implementieren. Auf die Elementen, die in einer Arrayliste enthalten sind, kann mit Hilfe der Methoden `get`, `set` usw. zugegriffen werden. Durch `add` und `remove` verändert sich die Länge einer Arrayliste.

b) Was ist schwache bzw. starke Aggregation? Worauf muss man bei der Implementierung von starker Aggregation achten?

Lösung

Aggregation ist eine spezielle Assoziation und repräsentiert die *Ganzes-Teile*-Beziehung. Es gibt zwei Arten von Aggregation: bei einer *schwachen* Aggregation ist die Lebensdauer des Teils nicht durch die des Ganzen beschränkt, bei einer *starken* Aggregation hingegen kann die Lebensdauer der Teile höchstens nicht die des Ganzen überschreiten.

Bei der Implementierung einer starken Aggregation ist darauf zu achten:

1. dass die Teile-Objekte nur von dem Ganzes-Objekt erzeugt werden (Bei einer starken Aggregation kann ein Teil-Objekt zu einem Zeitpunkt nur einem Ganzes-Objekt gehören) und
2. dass deren Referenzen nicht nach außen hergegeben werden.

Wenn einem Ganzes-Objekte maximal m Teile-Objekte gehören können, wobei m a priori feststeht, kann man eine starke Aggregation implementieren, indem man einen Konstruktor für die Ganzes-Klasse programmiert, der alle m Teile-Objekte des neu erzeugten Objektes mit erzeugt. Wenn das allerdings nicht möglich (wenn z.B. m nicht a priori feststeht) oder nicht wünschenswert ist (Beim Gewinnen eines neuen Kunden will man beispielsweise nicht alle Konten, die er im Laufe der Zeit haben wird, sofort anlegen), dann sollte eine Methode implementiert werden, welche die Teile-Objekte erzeugt. Siehe z.B. die Methode `createAccount` in der Vorlesung.

Wird die Aggregation mit `ArrayList` implementiert, so hat der Konstruktor der Ganzes-Klasse—unabhängig vom Aggregationstyp—das `ArrayList`-Objekt zu initialisieren.

c) Was sind Multiplizitat und Navigierbarkeit im Zusammenhang mit einer Assoziation? Welche Auswirkung haben sie auf die Implementierung?

Lösung

Mit Multiplzität wird bei einer Assoziation modelliert, wieviele Objekte der einen Klasse mit wievielen Objekten der anderen Klasse assoziert sind. Implementierung verschiedener Multiplizitäten siehe Vorlesungsfolie 10.

Mit Navigierbarkeit einer Assoziation modelliert man die Sichtbarkeit der Assoziation in den beteiligten Klassen. Wenn eine Assoziation zwischen den Klassen `C` und `D` in Richtung `C -> D` navigierbar ist, dann ist die Assoziation für die Klasse Klasse `C` sichtbar und muss in `C` implementiert werden. Wenn eine Assoziation in Richtung `D -> C` nicht navigierbar ist, ist sie in der Klasse `D` nicht sichtbar und muss auch nicht implementiert werden.

Hinweis Die Beschreibung von UML-Elementen hier ist stark vereinfacht. Für detailierte Beschreibung sei auf die UML-Spezifikation `http://www.omg.org/cgi-bin/doc?formal/05-07-04` verwiesen, die jedoch den Rahmen der Vorlesung bei weitem sprengt.