Kurs 1793 “Software Engineering I - Grundkonzepte der OOSE”

Klausur am 2.8.2003

Sommersemester 2003
Hinweise zur Bearbeitung der Klausur zum Kurs 1793
“Software Engineering I - Grundkonzepte der OOSE”

Wir begrüßen Sie zur Klausur "Software Engineering I". Bitte lesen Sie sich diese Hinweise vollständig und aufmerksam durch, bevor Sie mit der Bearbeitung der Aufgaben beginnen.

1. Prüfen Sie bitte die Vollständigkeit Ihrer Unterlagen. Ihr Klausurexemplar umfaßt:

   - 2 Deckblätter,
   - diese Hinweise zur Bearbeitung,
   - 1 Formblatt für eine Bescheinigung für das Finanzamt,
   - 5 Aufgaben (Seite 3 - Seite 12)

2. Füllen Sie, **bevor** Sie mit der Bearbeitung der Aufgaben beginnen, folgende Seiten des Klausurexemplars aus:

   a) **Beide** Deckblätter mit Name, Anschrift sowie Matrikelnummer. **Markieren Sie vor der Abgabe auf beiden Deckblättern die von Ihnen bearbeiteten Aufgaben**.
   b) Falls Sie eine Teilnahmebescheinigung für das Finanzamt wünschen, füllen Sie bitte das entsprechende Formblatt aus.

   Nur wenn Sie beide Deckblätter vollständig ausgefüllt haben, können wir Ihre Klausur korrigieren!

3. Streichen Sie ungültige Lösungen deutlich durch.

4. Schreiben Sie bitte auf jedes beschriebene Blatt oben links Ihren Namen und oben rechts Ihre Matrikelnummer. Wenn Sie weitere eigene Blätter benutzt haben, heften Sie auch diese, mit Namen und Matrikelnummer versehen, an Ihr Klausurexemplar.

   Lose Blätter, insbesondere ohne Name und Matrikelnummer, werden nicht bewertet!

5. Neben Schreibgerät und unbeschriebenen Konzeptpapier sind **Kurseinheiten, Einsendeaufgaben** sowie deren **Musterlösungen** als Unterlagen zugelassen.

6. Mit **Bleistift** geschriebene oder gezeichnete Lösungen werden **nicht korrigiert**. Verwenden Sie bitte nur Füller oder Kugelschreiber.

7. Es sind maximal 100 Punkte erreichbar.

Wir wünschen Ihnen bei der Bearbeitung der Klausur viel Erfolg!

Kurs 1793 "Software Engineering I - Grundkonzepte der OOSE"

Klausur am 2.8.2003

Aufgabe 1 (*20 + 7 + 6 = 33 Punkte*)

In dieser Aufgabe sollen Sie ein Domänen-Klassendiagramm, ein Objektdiagramm und eine Anwendungsfallbeschreibung erstellen.

Gegeben sind folgende Anforderungen an ein Online-Auktionssystem:

"Unsere Kunden können in der Rolle eines Verkäufers einen Artikel zur Versteigerung anbieten, indem sie eine Artikelbeschreibung heraufladen und einen Mindestgebotspreis und einen Versteigerungs-Endzeitpunkt festlegen. Andere Kunden können in der Rolle eines Käufers dann Gebote abgeben, die jedoch das Mindestgebot nicht unterschreiten dürfen und mindestens 50 Cent über früheren Geboten liegen müssen. Bei jedem Gebot wird Datum und Uhrzeit festgehalten. Der Bieter, der bis zum Versteigerungs-Endzeitpunkt am meisten geboten hat, erhält den angebotenen Artikel. Der Verkäufer kann den Artikel alternativ oder zusätzlich auch zu einem sog. Sofortkaufpreis anbieten. Bietet ein Käufer den Sofortkaufpreis, wenn noch keine Gebote vorliegen, so kommt der Kauf sofort zustande, ohne dass weitere Gebote berücksichtigt werden. Nach dem Abschluss einer Versteigerung oder eines Sofortkaufs erhalten der Verkäufer und alle Bieter eine Benachrichtigung, ob die Versteigerung für sie erfolgreich war. Wenn der ersteigerte Artikel versandt und bezahlt wurde (dieser Vorgang wird vom System nicht unterstützt), können Verkäufer und Käufer sich gegenseitig bewerten, d. h. jeder gibt dem anderen eine eine Note (positiv, neutral, negativ). Die *Bewertungszahl* eines Kunden ist die Anzahl der positiven abzüglich der Anzahl der negativen Bewertungen. Das Online-Auktionssystem muss alle Transaktionen (Gebote, Sofortkäufe usw.) genau protokollieren. Aus dem Grund ist es auch erforderlich, dass jeder Kunde sich mit Name, Adresse und E-Mail-Adresse sowie einem Pseudonym anmeldet."

Aufgabenstellung:

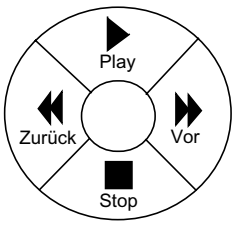
a) Erstellen Sie ein redundanzfreies Klassendiagramm für die beschriebene Anwendungsdomäne. Versuchen Sie, möglichst wenige Klassen zu definieren, und geben Sie zu jeder Beziehung die Multiplizitäten an. Verteilen Sie mindestens 15 sinnvolle Attribute auf Ihre Klassen, darunter mindestens zwei abgeleitete Attribute.

b) Skizzieren Sie ein gültiges Beispiel-Objektdiagramm, das zu jeder Ihrer Klassen aus a) mindestens eine Instanz enthält. Es reicht, wenn Sie zu jedem Objekt einen einzigen Attributwert angeben.

c) Erstellen Sie eine Anwendungsfallbeschreibung für den Anwendungsfall "Auktion beenden" des Akteurs "Systemuhr". Beachten Sie bitte auch den Fall, dass zum Versteigungs-Endzeitpunkt evtl. keine Gebote vorliegen.

Aufgabe 2 (*14 Punkte*)

In dieser Aufgabe sollen Sie die Steuerung eines portablen CD-Spielers mit Hilfe eines Zustandsdiagramms modellieren. Er besitzt 4 Tasten zur Steuerung:



- Play (zur Wiedergabe der CD)
- Stop (zum Stoppen und Ausschalten)
- Vor (für schnellen Vorlauf oder zum Anwählen des nächsten Titels)
- Zurück (für schnellen Rücklauf oder zum Anwählen des vorigen Titels)

Zur Vereinfachung wird die Taste "Zurück" in dieser Aufgabe vernachlässigt, sie arbeitet vollkommen analog zur Taste "Vor".

Im ausgeschalteten Zustand kann der CD-Spieler durch Drücken von "Play" eingeschaltet werden und beginnt dann automatisch mit dem Abspielen der CD. Drückt man dann die Taste "Vor", beginnt der CD-Spieler mit einem schnellen Vorlauf, d. h. die CD wird deutlich beschleunigt abgespielt. Lässt man die Taste nach weniger als 0,5 Sekunden los, wird dies als "kurz drücken" interpretiert und die Wiedergabe mit dem nächsten Titel fortgesetzt. Hält man die Taste mindestens 0,5 Sekunden, wird dies als "lang drücken" interpretiert und der schnelle Vorlauf fortgesetzt, bis die Taste wieder losgelassen wird. Nach dem Loslassen wird an der erreichten Stelle die normale Wiedergabe fortgesetzt. Durch kurzes Drücken auf "Stop" kann die Wiedergabe unterbrochen werden; anschließend kann sie mit "Play" fortgesetzt werden. Wenn die Wiedergabe gestoppt ist, bewirkt die Vor-Taste immer einen Sprung zum nächsten Titel, ein schneller Vorlauf ist dann nicht möglich. Wird "Stop" mindestens 0,5 Sekunden lang gedrückt, schaltet sich das Gerät aus.

Aufgabenstellung:

Erstellen Sie für die Steuerung des CD-Spielers ein Zustandsdiagramm.

Hinweise:

- Modellieren Sie einen Zustandswechsel für den Fall, dass "Vor" oder "Stop" 0,5 Sekunden lang gedrückt worden sind. Das Ereignis für diesen Zustandswechsel lautet "0,5 s vergangen".
- Geben Sie zu jedem Zustandswechsel ein Ereignis und mindestens eine Aktion an, soweit möglich.
- Beachten Sie, dass auch das Loslassen einer Taste ein Ereignis sein kann.
- Folgende Aktionen stehen zur Verfügung: einschalten(), wiedergabe(), vorlauf(), nächsterTitel(), stoppen(), ausschalten().

Aufgabe 3 (*2 + 6 + 1 + 1 + 16 + 5 = 31 Punkte*)

In dieser Aufgabe soll Entwurfsmuster "Iterator" an einem Beispiel untersucht werden.

Gegeben ist die Klassenspezifikation der Klassen HashSet, HashIterator, Array und Integer in Abb. 1 und Abb. 2:

```
class HashSet
   responsibilities:
      Verwaltet eine Menge von Objekten unter Verwendung einer Hashtable-Datenstruktur.
   features:
      attributes:
         Array hashtable;
            comment: "Das Hashtable, das die gespeicherten Elemente enthält. Nicht verwendete Einträge
                      dieses Arrays haben den Wert null.“
      operations:
         public put (in element: Object);
            comment: "Fügt das Element element der Menge hinzu.“
         public contains (in element:Object): boolean
            comment: "Der Rückgabewert ist true, wenn element in der Menge enthalten ist.“
         public iterator(): Iterator
            comment: "Liefert einen Iterator, der über die Elemente der Menge iteriert."
         public toString(): String
            comment: "Liefert eine Stringdarstellung der Menge, indem es bei jedem Element der Menge die
               Operation toString():String aufruft und die Ergebnisse konkateniert."
end HashSet.


class HashSetIterator
   responsibilities:
      Iteriert über die Elemente eines HashSet.
   inv: (index = -1) or (next() = hashtable.get(index))
   features:
      attributes:
         Array hashtable;
            comment: "Eine Referenz auf das Hashtable des HashSet, über den iteriert werden soll.“
         int size;
            comment: "Die Größe des Hashtable (einschließlich nicht verwendeter Elemente)“
         int index;
            comment: "Enthält die Indexnummer des Arrayeintrag, der beim Iterieren als nächstes
               zurückgegeben wird, bzw. -1, wenn das komplette Array durchlaufen wurde (hasNext() == false)“
      operations:
         public hasNext():boolean
            comment: "Liefert true, wenn index ungleich -1.“
         public next():Object
            pre: index ungleich -1;
            comment: "Liefert das Element an der Stelle index im Array hashtable.
               Setzt die Variable index unter Verwendung der Operation findNextElement auf das nächste
               besetzte Element des Hashtables bzw. auf -1, falls es keines mehr gibt.“
         setHashtable (in hashtable: Array);
            comment: "Weist dem Iterator ein zu durchlaufendes Hashtable-Array zu. Initialisiert die Variable size()
               mit dem Ergebnis des Aufruf der size()-Operation des übergebenen Arrays. Setzt index auf das
               erste besetzte Element des Hashtables.“
         findNextElement()
            pre: index ungleich -1;
            comment: "Lässt die Variable index auf das nächste verwendete Element im Hashtable zeigen.
               Setzt index auf -1, wenn der Hashtable komplett durchlaufen wurde.“
         setIndex(int index)
            comment: "Setzt die Instanzvariable index auf den angegebenen Wert“
end HashSetIterator.
```

Abb. 1 Textuelle Spezifikation der Klassen HashSet und HashSetIterator

```
class Array
   responsibilities:
      Realisiert eine Array-Datenstruktur.
   features:
      operations:
         public Object get(int index):Object
            comment: "Liefert das Element an der Stelle index.“
         public size():int
            comment: "Liefert die Größe des Arrays, d.h. die Anzahl seiner Speicherplätze.“
         ...
end Array.


class Integer
   responsibilities:
      Realisiert eine ganze Zahl.
   features:
      operations:
         public toString(): String
            comment: "Liefert eine Stringdarstellung der Integer-Zahl."
         ...
end Array.
```

Abb. 2 Textuelle Spezifikationen der Klassen Array und Integer

0	1	2
null	527	469

hashtable.size() = 3

Abb. 3 Beispiel-Inhalt des Arrays hashtable

Die Klasse HashSet bietet neben den für Mengen typischen Operationen put() und contains() noch die Operation iterator(), die einen Iterator zum Durchlaufen der Menge liefert. Die Operation toString() liefert eine Stringdarstellung der Menge, indem sie unter Verwendung eines Iterators jedes Element der Menge durchläuft, auf jedem Element die Operation toString() aufruft und die Ergebnisse konkateniert.

Aufgabenstellung:

a) Beschreiben Sie in einem Satz, was ein Iterator ist.

b) Ergänzen Sie das Sequenzdiagramm in Abb. 4, das den Ablauf der Operation toString() der Klasse HashSet beschreibt. Gehen Sie dabei von dem Fall aus, dass die Instanzvariable hashtable den in Abb. 3 skizzierten Inhalt hat, also genau 2 gültige Elemente enthält. Der interne Ablauf der Operationen setHashtable(), hasNext() und next() sowie Aufrufe an nicht dargestellte Objekte brauchen an dieser Stelle noch nicht dargestellt zu werden.

Die Klasse HashSetIterator sei so implementiert, dass die Instanzvariable index immer auf das Element des Arrays hashtable zeigt, das beim nächsten Aufruf von next() zurückgegeben würde. Wenn

die Menge komplett durchlaufen wurde, soll index = -1 gelten. Die Operationen setHashtable(), hasNext() und next() dürfen diese Invariante als Vorbedingung voraussetzen und müssen sie als Nachbedingung sicherstellen. Zur Implementierung stützen sie sich auf die nichtöffentlichen Operationen findNextElement() und setIndex() ab. Der Zugriff auf die Variable hashtable erfolgt über die Array-Operationen size() und get().

Gehen Sie bitte auch in den folgenden Teilaufgaben davon aus, dass die Variable hashtable den in Abb. 3 angegebenen Inhalt hat.

c) Welchen Wert hat die Variable index nach Beendigung der Operation setHashtable()?

d) Wie oft muss die Operation setHashtable() die Operation findNextElement() aufrufen?

e) Geben Sie in Abb. 5 ein verfeinertes Sequenzdiagramm für die Operation toString() an, das auch die internen Abläufe der Operationen setHashtable(), hasNext() und next() darstellt.

Hinweis: Alle in den Abb. 1 und 2 angegebenen Operationen der Klassen HashSetIterator und Array sollten in Ihrem Sequenzdiagramm vorkommen!

f) Was würde bei der hier skizzierten Implementierung passieren, wenn dem HashSet mit Hilfe der Operation put() weitere Elemente hinzugefügt werden, nachdem ein Iterator erzeugt, aber nur teilweise durchlaufen wurde? Wird der Iterator die neu hinzugefügten Elemente liefern oder nicht?

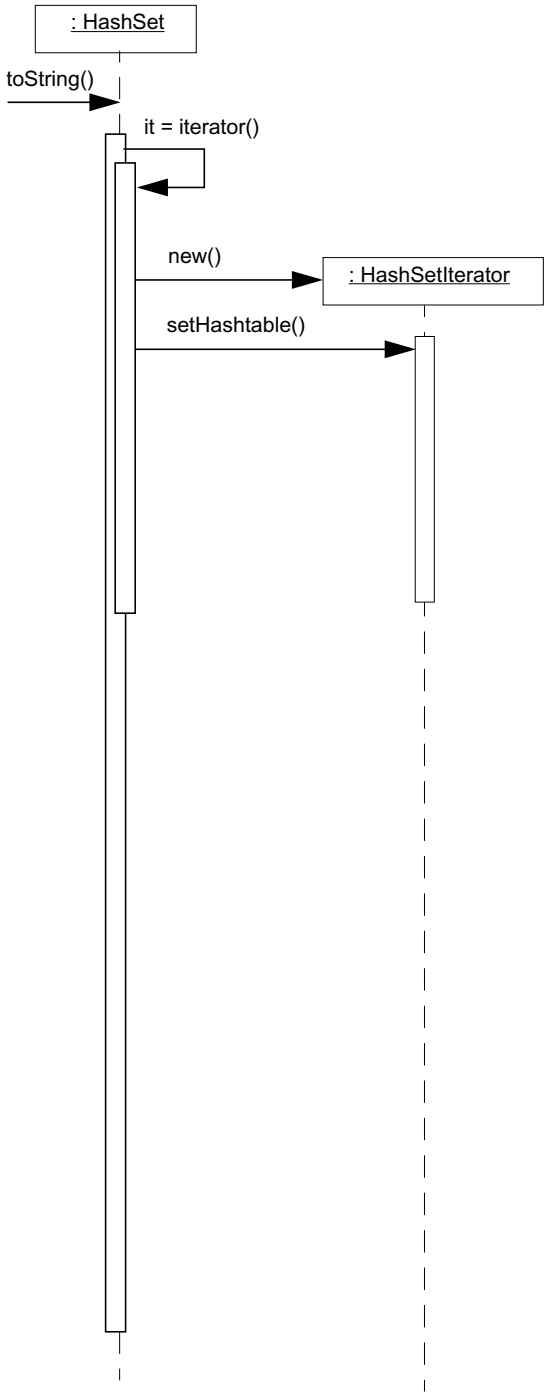


Abb. 4: Sequenzdiagramm der Operation toString() der Klasse HashSet

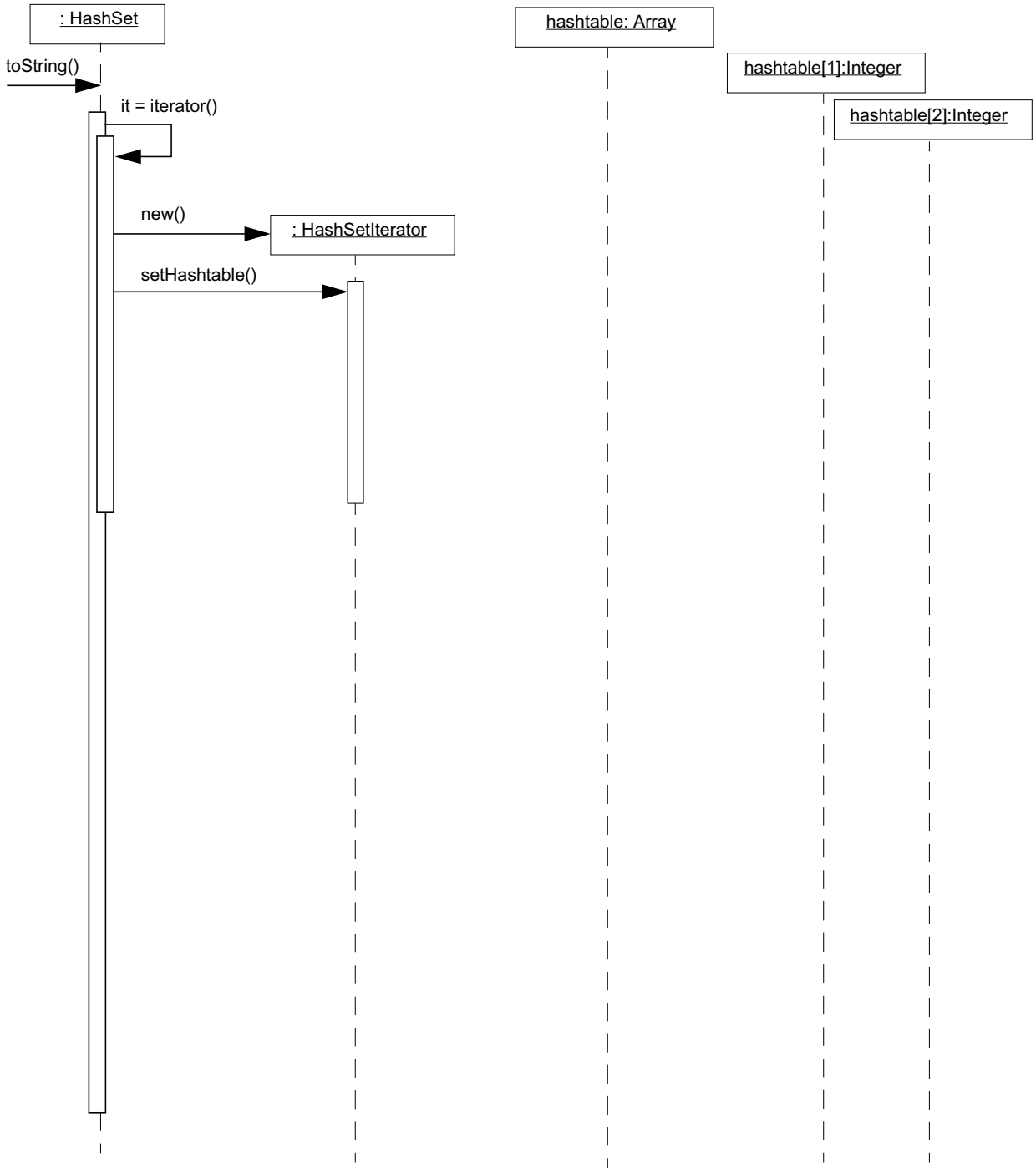


Abb. 5: Verfeinertes Sequenzdiagramm der Operation toString() der Klasse HashSet

Aufgabe 4 (*10 Punkte*)

Gegeben ist das Grobentwurfs-Klassendiagramm in Abb. 6.

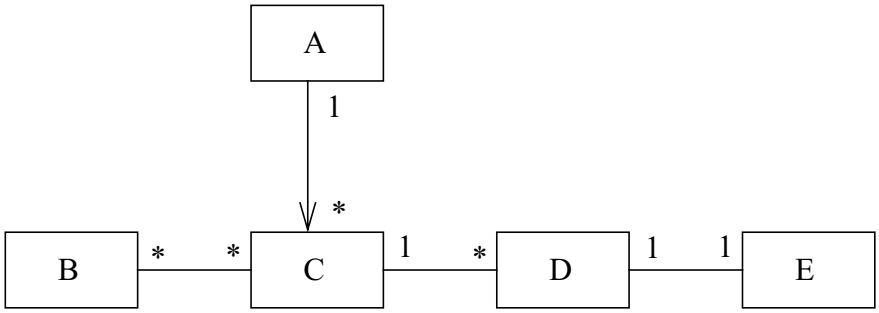


Abb. 6 Das zu transformierende Klassendiagramm

Aufgabenstellung

Erstellen Sie, ausgehend von Abb. 6, ein Feinentwurfs-Klassendiagramm durch Anwendung der im Kurstext (Kap. 48) beschriebenen Transformationen.

- Geben Sie bitte alle Multiplizitäten an.
- Definieren Sie Ordnerklassen für die Klassen A und B.

Hinweis: Assoziationen, die sich direkt auf ein Attribut abbilden lassen, sowie von Ordner- und Relationenklassen ausgehende Assoziationen müssen nicht weiter transformiert werden.

Aufgabe 5 (*4 + 1 + 1 + 1 + 3 + 2 = 12 Punkte*)

Laut Kurstext ist eine Klasse eine Echte Ganzes-Klasse, wenn es Situationen gibt, in denen zu ihren Instanzen nicht über Verbindungen von Instanzen einer anderen Klasse aus hin navigiert werden kann.

In dieser Aufgabe soll zu dem Klassendiagramm in Abb. 7 unter Zuhilfsnahme des Objektdiagramms in Abb. 8 die Menge der Echten Ganzes-Klassen bestimmt werden.

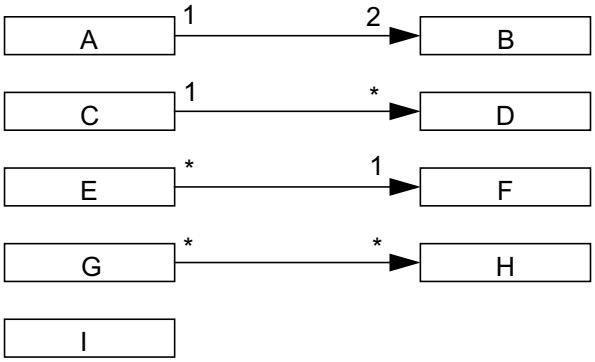


Abb. 7 Klassendiagramm

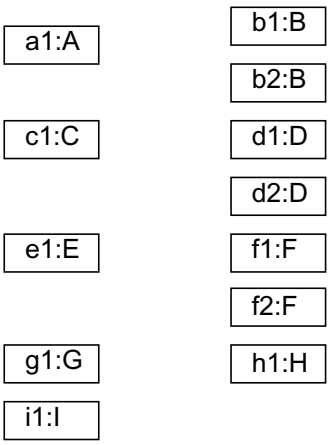


Abb. 8: Zugehöriges Objektdiagramm

Aufgabenstellung

a) Zeichnen Sie in dem Objektdiagramm in Abb. 8 Objektverbindungen ein, und zwar möglichst wenige, jedoch so viele, dass die im Klassendiagramm angegebenen Multiplizitäten erfüllt sind.

b) Welche Objekte in Abb. 8 sind von Instanzen anderer Klassen aus erreichbar?

c) Von welchen Klassen ist jede Instanz erreichbar?

d) Welche Klassen sind Echte Ganzes-Klassen?

e) Streichen Sie bitte im folgenen Satz die nichtzutreffenden Teile:

Eine Klasse K ist **keine**(!) Echte Ganzes-Klasse, wenn es
eine/keine *(Nichtzutreffendes streichen)*
Klasse K’ und eine Assoziation zwischen den Klassen K und K’ gibt, die
von K nach K’ navigierbar / von K’ nach K navigierbar *(Nichtzutreffendes streichen)*
ist und bei der die untere Grenze der Multiplizität
am K-Ende / am K’-Ende *(Nichtzutreffendes streichen)*
größer Null ist.

f) Was haben Echte Ganzes-Klassen mit Ordnerklassen zu tun?