Hochschule Aalen
Fakultät Elektronik und Informatik

Donnerstag, 12. Juli 2012

Klausur

Objektorientierte Modellierung

SS 2012

Prüfer: Prof. Dr. R. Dietrich

Aufgabe 1. *Statische Modellierung für einen Warenhauskonzern* **(68 *Punkte*)**

Es soll Software erstellt werden, die die Geschäftsprozesse in einem Warenhauskonzern unterstützen.

(1.1) *Statisches Analysemodell – Klassendiagramm* (38 *Punkte*)

Erstellen Sie ein statisches Analysemodell (Klassendiagramm) für den Warenhauskonzern. Die im Folgenden beschriebenen Sachverhalte müssen berücksichtigt werden (sofern Sie für das statische Modell relevant sind):

> Zu dem Warenhauskonzern gehören verschiedene Vertriebsketten wie z.B. „Pfennigfuchser", „gut&günstig", „Quality". Jede Kette hat Filialen in vielen Städten, in manchen Städten gibt es sogar mehrere Filialen einer Kette. Von den Städten, in denen es Filialen gibt, werden der Name und die Stadtkategorie (Klein-, Mittel-, Großstadt oder Metropole) registriert. Von den Filialen eine eindeutige Kennung sowie die Stadt und die Straße, in der sie sich befinden.
>
> Jede der Ketten hat ein festgelegtes Angebot an Produkten. Für Produkte werden eine Bezeichnung registriert und ein Preis. Es gibt zwei spezielle Produkttypen: Lebensmittel, bei denen ein Mindesthaltbarkeitsdatum registriert wird, und Elektroartikel. Bei Elektroartikeln wird registriert, ob das Produkt ein CE-Zeichen hat und, falls das Produkt technisch geprüft wurde, welcher TÜV das Produkt geprüft hat (z.B. „TÜV-Rheinland", „TÜV-Süd").
>
> Um das Kaufverhalten der Kunden und die Wirksamkeit von Werbekampagnen zu erforschen, gibt der Konzern Rabattkarten aus. Die Rabattkarte hat einen eindeutigen Code aus Ziffern und Buchstaben und ist auf einen Kunden registriert. Von dem Kunden werden Name und Adresse sowie optional eine E-Mail-Adresse registriert. Setzt ein Kunde die Rabattkarte bei einem Einkauf ein, bekommt er 2% Rabatt.
>
> Jeder Einkauf wird mit allen gekauften Produkten, dem Zeitpunkt des Einkaufs, der Filiale und – sofern eine eingesetzt wurde – mit der Rabattkarte registriert. Einkäufe mit Rabattkarte können somit einem Kunden zugeordnet werden.

Dabei ist folgendes **zu beachten**:

- Sämtliche in Klassen verwendeten Attribute sind genau zu spezifizieren (Datentypen, gegebenenfalls Multiplizitäten und Eigenschaftswerte). Falls die Spezifikation der Attribute außerhalb der Klassen erfolgt, müssen im Klassendiagramm keine Attributtypen angegeben werden.
- Klassenattribute und abgeleitete Attribute sind entsprechend darzustellen.
- Multiplizitäten sind auf beiden Seiten einer Assoziation jeweils möglichst genau anzugeben.
- Operationen soll das Modell zunächst nicht enthalten.
- Die in Abbildung 1 unten dargestellten Datentypen können ohne weitere Spezifikation benutzt werden. Wenn weitere Datentypen benötigt werden, sind diese zu spezifizieren.
- Wenn Sie der Meinung sind, dass ein Sachverhalt nicht genau genug im Text beschrieben ist, dann formulieren Sie in Ihrer Lösung informell, wie Sie den Sachverhalt verstehen, und berücksichtigen Sie Ihre Aussage entsprechend in Ihrem Modell.

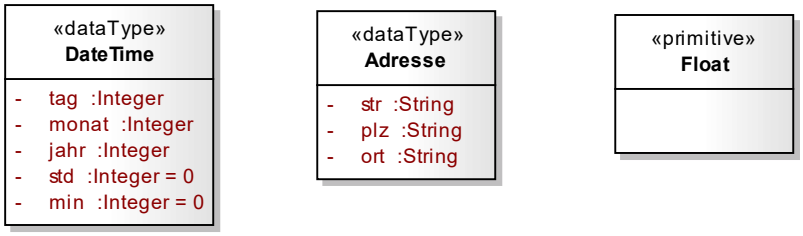

Abbildung 1: Datentypen (der Datentyp *DateTime* kann sowohl zur Modellierung eines Datums als auch eines Zeitpunkts verwendet werden).

(1.2) *Objektdiagramm* (30 *Punkte*)

Erstellen Sie ein Objektdiagramm, welches folgendes Szenario darstellt:

> Der Kunde „Müller“ kaufte gestern (11. Juli 2012) um 11:00 Uhr in der Stuttgarter Filiale der Kette „Pfennigfuchser“ einen Fernseher mit CE-Zeichen ohne Rabattkarte. Derselbe Kunde „Müller“ kaufte heute Morgen (12. Juli 2012) um 09:00 Uhr in der Aalener Filiale der Kette „Pfennigfuchser“ ein Buch „Objektorientierte Modellierung“. Dabei setzte er seine Rabattkarte mit dem Code „Muel1234567“ ein.

Das Objektdiagramm soll Objekte mit Attributwerten und Verknüpfungen („Links“) darstellen. Stellen Sie nur solche Attributwerte in Objekten dar, die aus dem obigen Text hervorgehen.

Aufgabe 2. *Dynamische Modellierung für eine Lagerverwaltung* **(*28 Punkte*)**

Der Warenhauskonzern unterhält ein zentrales Lager, in dem alle Produkte für alle Vertriebsketten bevorratet werden. Dieses Lager wird durch eine eigene Software unterstützt. Abbildung 2 zeigt einen Ausschnitt aus dem statischen Analysemodell dieser Software.

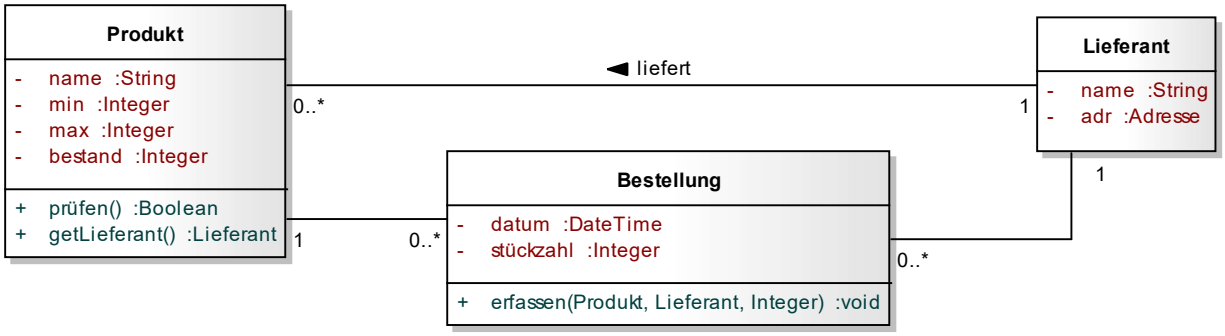

Abbildung 2: Ausschnitt aus dem statischen Analysemodell der Lagerhaltungssoftware.

(2.1) *Sequenzdiagramm* (*14 Punkte*)

Nach jeder Entnahme eines Produkts aus dem Lager muss ein Lagerverwalter eine Prüfung veranlassen, ob noch eine ausreichende Stückzahl dieses Produkts im Lager vorhanden ist und gegebenenfalls eine Bestellung veranlassen, die das Produkt bis zum Maximalbestand wieder auffüllt. Im Folgenden ist dieser Anwendungsfall beschrieben:

Anwendungsfall: Produkt prüfen

Akteur: Lagerverwalter

Ziel: Es sind ausreichend Waren eines Produkts im Lager vorhanden

Beschreibung:

1. Es wird der Minimale Bestand des Produkts bestimmt
2. Es wird der aktuelle Bestand des Produkts ermittelt

Falls der aktuelle Bestand kleiner ist, als der Minimale Bestand, wird

3. der Maximalbestand des Produkts ermittelt

4. Der Lieferant des Produkts ermittelt
5. Eine neue Bestellung erfasst
6. Die erforderliche Stückzahl wird eingetragen
7. Das aktuelle Datum wird eingetragen
8. Die Verknüpfung zum Produkt wird hergestellt
9. Die Verknüpfung zum Lieferanten des Produkts wird hergestellt
10. Das Resultat der Prüfung ist "false"

Alternativen:

Falls der aktuelle Bestand größer ist als der Minimalbestand

3. Das Resultat "true" wird zurückgegeben

4 – 9 entfallen

Erstellen Sie ein Sequenzdiagramm für das Szenario *der Minimalbestand ist unterschritten und es muss eine Bestellung erzeugt werden*. Vervollständigen Sie dazu das begonnene Sequenzdiagramm auf der letzten Seite, trennen Sie das Blatt von den Aufgabenblättern und heften Sie es am Schluss mit Ihren anderen Lösungen zusammen.

(2.2) *Zustandsautomat* (*14 Punkte*)

Beschreiben Sie den Lebenszyklus eines Objekts der Klasse Produkt aus Abbildung 2 als Zustandsautomat.

Ein Produkt ist zunächst in ausreichender Anzahl vorhanden. Nach jeder Entnahme von Stücken wird der Bestand entsprechend reduziert und es muss geprüft werden, ob eine gewisse Anzahl nachbestellt werden muss. Gegebenenfalls wird auf die Lieferung gewartet, und nach deren Ankunft die neue Ware eingelagert. Auch wenn auf eine Lieferung gewartet wird, können weiter Produkte entnommen werden, solange noch genügend restliche vorhanden sind.

Wählen Sie geeignete qualitative Zustände für Ihren Automaten.

Sie können die folgenden Ereignisse verwenden:

- *Entnahme(n):* Es sind *n* Stück des Produkts entnommen worden. (Es ist zu beachten, dass nur *n* Stück entnommen werden können, falls auch mindestens *n* Stück vorhanden sind!)
- *Einlagerung(n):* Es wurden *n* Stück des Produkts geliefert und eingelagert.

Zur Modellierung von Aktivitäten steht die Operation *prüfen():Boolean* der Klasse Produkt zur Verfügung (vgl. 2.1). Die Operation liefert das Ergebnis *true*, falls noch ausreichend Produkte vorhanden sind und *false*, falls welche nachbestellt werden mussten.

Außerdem können in Guard-Bedingungen und Effekten die Attribute der Klasse Produkt (z.B. *bestand*) und die Attribute der Ereignisse (z.B. die Anzahl *n* der entnommenen Produkte) verwendet werden.

Aufgabe 3. *Implementierung eines Sensornetzes zur Wetterbeobachtung* **(*46 Punkte*)**

Ein Netz von Sensoren für die Wetterbeobachtung soll durch Software unterstütz werden. Abbildung 3 zeigt ein Entwurfsmodell für diese Software.

Das Netz besteht aus einer Menge von Sensoren in verschiedenen Städten. Ein Sensor ist entweder ein Luftdrucksensor (*DruckSensor*) oder ein Temperatursensor (*TempSensor*). Die Sensoren sind in der Lage die aktuellen Messwerte zu bestimmen (`DruckSensor::getDruck()` bzw. `TempSensor::getTemp()` – diese Operationen greifen über geeignete Schnittstellen auf die Hardware der Sensoren zu.)

Die Sensoren können ihre aktuellen Messwerte auf Standardausgabe ausgeben (`print()`-Operation der verschiedenen Sensor-Klassen). Die Operation `Sensornetz::print()` ruft die `print()`-Funktionen aller Sensoren des Netzes auf und gibt so alle aktuellen Messwerte des ganzen Netzes aus.

Die Operationen `Sensornetz::neuerDruckSensor(stadt:string)` bzw. `Sensornetz::neuerTempSensor(stadt:string)` fügen einen neuen Druck- bzw. Temperatur-Sensor in der Stadt `stadt` in das Netzwerk ein.

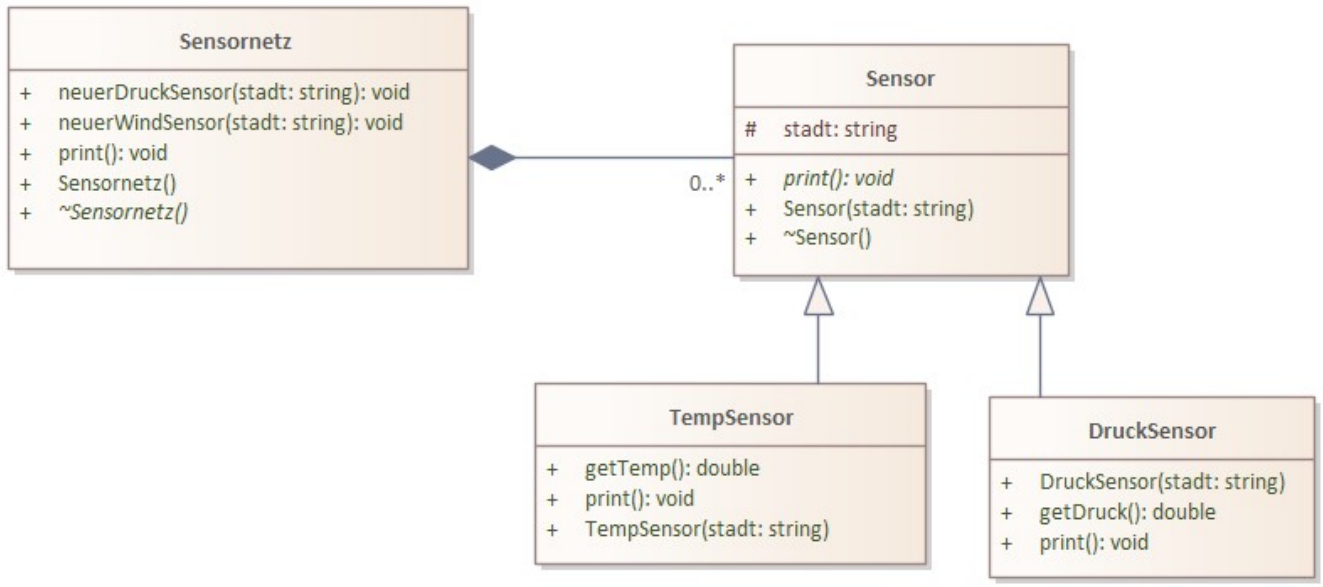

Abbildung 3. Entwurfsmodell für ein Sensornetz.

(3.1) *Klassen Sensor und TempSensor* (*23 Punkte*)

Schreiben Sie Klassendefinitionen für die Klassen *Sensor* und *TempSensor* sowie Methodendefinitionen für sämtliche Methoden dieser Klassen außer `TempSensor::getTemp()`.

(3.2) *Klasse Sensornetz* (*23 Punkte*)

Schreiben Sie eine Klassendefinition für die Klasse *Sensornetz* sowie Methodendefinitionen für sämtliche Methoden der Klasse.

In Aufgabe (3.1) und (3.2) ist folgendes zu beachten:

- Die Klasse *Drucksensor* kann als korrekt implementiert vorausgesetzt und verwendet werden.
- Ebenso kann die Methode `TempSensor::getTemp()` als korrekt implementiert vorausgesetzt und verwendet werden.
- Die Druckausgage der *print()*-Operationen soll so gestaltet werden, dass eine Aufruf der Methode `Sensornetz::print()` eine Ausgabe wie im folgenden Beispiel erzeugt:

```
Luftdruck in Aalen: 978 hPa
Temperatur in Aalen: 25 Grad
Temperatur in Stuttgart: 27 Grad
Luftdruck in München: 990 hPa
Temperatur in München 24 Grad
```

Arbeitsblatt **Matrikelnummer:** _____

Bitte abtrennen, Matrikelnummer eintragen und zu Ihren Lösungen dazu heften!