ulm university universität
uulm

Institut für Numerische Mathematik
Dr. Andreas F. Borchert und Dr. Michael C. Lehn

10. November 2016
Blatt 3

Objektorientierte Programmierung mit C++ (WS 2016/2017)

Abgabe bis zum 17. November 2016, 16:00 Uhr

Lernziele:

- Erstellung von UML-Klassendiagrammen.

Aufgabe 4: UML-Klassendiagramm für Brettspiele

Entwerfen Sie ein Klassendiagramm für eine Verwaltung von Brettspielpartien. Ein Brettspiel im Sinne dieser Aufgabe ist ein offenes Spiel für zwei Spieler, die alternierend einen Spielzug bestimmen. Ein Brettspiel läuft entweder noch oder ist bereits beendet und wurde dann von einem der beiden Spieler gewonnen oder als unentschieden gewertet. Brettspiele sind offen, d.h. die gesamte Spielsituation lässt sich anhand der abgegebenen Spielzüge bestimmen. Typische Brettspiele in diesem Sinne sind Go, Schach oder Nim-Spiele.

Für das Klassendiagram müssen Sie sich nicht auf ein konkretes Brettspiel festlegen. Dies bleibt noch abstrakt. Darzustellen ist eine Verwaltung für beliebig viele Partien und Spieler. Bei jeder Partie gibt es einen ersten und zweiten Spieler und beliebig viele Spielzüge. Jeder Spielzug kann von jedem beliebigen Spieler (nicht nur an einer Partie beteiligte Spieler) kommentiert werden.

Alle Beziehungen sind mitsamt ihren Komplexitätsgraden und den Navigationsrichtungen darzustellen. Konkret ist dabei zu berücksichtigen,

- dass Sie ausgehend von der Verwaltung nach Partien und Spielern suchen können,
- dass Sie für jeden Spieler sämtliche Partien, an denen er beteiligt war, ermitteln können und
- dass für jede Partie die beteiligten Spieler und die Spielzüge abgerufen werden können.

Zu jedem Spielzug sollten all die zugehörigen Kommentare abrufbar sein und bei jedem Kommentar der zugehörige Spieler, der diesen hinzugefügt hat.

Bei jeder der Klassen ist zumindest für jede der unterstützten Navigationsrichtung mindestens eine passend erscheinende Methode zu benennen. Sie müssen aber keine der Datenstrukturen modellieren, noch die Signaturen irgendwelcher Methoden hinzufügen noch Vollständigkeit bei der Nennung der Methoden erreichen.

Sie dürfen das Klassendiagramm auf beliebige Weise anfertigen. Am Ende sollte aber eine PDF-Datei dabei entstehen, die Sie mit *submit* einreichen können:

```
thales$ submit cpp 4 classdiag.pdf
```

Hinweise:

Auch wenn es Ihnen vollkommen frei steht, wie Sie die Diagramme anfertigen, wird empfohlen, das MetaUML-Paket zu verwenden. Die zugehörige Dokumentation lässt sich unter `https://github.com/ogheorghies/MetaUML/releases/download/v0.2.5/metauml_manual_0.2.5.pdf` herunterladen.

Unter *department.mp* ist ein Beispiel hierfür zu finden. So wird beispielsweise mit MetaUML eine Klasse definiert:

```
Class.Department("Department")
    (
        "name"
    )
    (
        "+Department()",
        "+getEmplyees()",
        "+getHead()",
        "+setHead()",
        "+addEmployee()",
        "+remEmployee()",
    );
```

Klassendeklarationen beginnen hier mit „Class.“, gefolgt von dem Namen der Klasse. Dann folgen drei Klammerpaare. Innerhalb des ersten Klammerpaares wird der Klassenname genannt (als Zeichenkette), beim zweiten Klammerpaar werden die Variablenfelder genannt und beim dritten Klammerpaar die Methoden.

Wenn die Klassen alle definiert sind, kann darüber nachgedacht werden, ob diese in geeigneter Form im Diagramm zu arrangieren sind. Wenn hier nichts angegeben wird, werden diese von MetaUML irgendwie arrangiert – das sieht dann nicht unbedingt sehr schön aus. Mit

distance = 1.2 (**xpart** *Employee.e* − **xpart** *Employee.w*);
Department.nw = *Employee.ne* + (*distance*, 0);

wird in MetaPost ein lineares Gleichungssystem definiert. *Department* bezeichnet hier die jeweilige Klassenbox und *Department.nw* die Nordwest-Ecke der entsprechenden Box. Die beiden Gleichungen sorgen dafür, dass die beiden Klassenboxen etwas weiter voneinander entfernt sind, als die Klassenboxen jeweils breit sind. Alles, was diese Gleichungen offenlassen, wird dann durch MetaUML festgelegt. Mit

drawObjects(*Employee*, *Department*);

werden dann die beiden genannten Objekte gezeichnet. Es lohnt sich, dies am Anfang einmal zu testen, bevor die Beziehungen hinzukommen.

Die Beziehungen sind etwas komplizierter, da hier jeweils auch noch die Beschriftungen anzubringen sind:

pair *memberOf.w*, *memberOf.e*, *memberOf.c*;
memberOf.w = 0.5[*Employee.ne*, *Employee.e*];
memberOf.e = *memberOf.w* + (*distance*, 0);
memberOf.c = 0.5[*memberOf.w*, *memberOf.e*];
link(*associationUni*)(*memberOf.e* −− *memberOf.w*);
item(*iAssoc*)("1..∗")(*obj.ne* = *memberOf.e*);
item(*iAssoc*)("∗")(*obj.nw* = *memberOf.w*);
item(*iAssoc*)("memberOf")(*obj.s* = *memberOf.c*);

Mit **pair** werden Punkte in MetaPost deklariert. (Einfache skalare Werte – wie im obigen Beispiel *distance* – benötigen keine Deklaration.) Für die Beziehung, die wir definieren wollen, werden hier drei Punkte definiert: die beiden Endpunkte und der Mittelpunkt, jeweils mit einem Suffix, der die Himmelsrichtung angibt wie beispielsweise „e“ für *east*. Mit *link* wird die Beziehung dargestellt, wobei hier *associationUni* angibt, dass es sich um eine unidirektionale Beziehung handelt. Die in den Klammern angegebene Anweisung *memberOf.e* −− *memberOf.w* zeichnet eine Linie zwischen den angegebenen Punkten. Mit Hilfe von *item* lassen sich dann noch die Beschriftungen hinzufügen, wobei diese durch eine virtuelle Box namens *obj* repräsentiert wird, die in den Klammern jeweils zu positionieren ist.

Wenn Sie Ihre MetaPost-Datei in PDF verwandeln wollen, dann empfiehlt sich die Nutzung dieses *Makefile*.

Viel Erfolg!