Christian-Albrechts-Universität zu Kiel
Sommersemester 2000
Fortgeschrittenenpraktikum (Java)

Übung 6:

Ausgabetermin: 19. Juni 2000

Abgabe: 3. Juli 2000

Aufgabe 1: [Netzwerkprogrammierung]

Schreiben Sie einen Server als Applikation, wenn Sie wollen auch als Applet, der auf Verbindungsaufnahmen wartet und an jeden Klienten zunächst einen Willkommensgruß und dann immer wieder — in Abständen von sagen wir 10 Sekunden — jeweils zwei Textzeilen liefert, die einen kurzen Text und Datum und Uhrzeit enthalten. Dafür verwenden Sie das Standard-Format der aktuellen Uhrzeit (siehe `(new Date()).toString()`).

Schreiben Sie einen Klienten (als Applikation reicht wieder), der mit diesem Server Verbindung aufnimmt und alle Informationen, die vom Server gesendet werden, Zeile für Zeile auf die Standard-Ausgabe ausgibt.

Für diese Übung müssen Sie die Hostnamen der verwendeten Rechner (innerhalb des lokalen Netzes) kennen und sich auf eine Portnummer einigen.

Aufgabe 2: [Hand-in]

Schreiben Sie ein Programm zur Aufgabenabgabe über das Netzwerk. Es sollen genau genommen zwei Programme entwickelt werden: ein Server, der die Aufgaben der Gruppen empfängt und verwaltet, und ein Client der die Serien zusammenpackt (wie in einer der vorherigen Aufgaben) und an den Server übermittelt.

Der Client soll dazu für jede zu verwaltende Gruppe ein Directory anlegen, die Datenstruktur soll nach der Übermittlung folgendermaßen aussehen:

- `gruppe12/serie01/`
  Dieses Verzeichnis enthält die von Gruppe 12 übermittelten Daten zu Serie 1, und zwar entpackt. Das Verzeichnis soll nach der Eingabe nicht mehr durch erneute Abgabe dieser Serie verändert werden können.
- `gruppe12/abgabe.txt`
  In dieser Datei soll festgehalten werden, wann welche Lösung der Gruppe abgegeben wurde.

Aufgabe 3: [Erweiterung von Schnittstellen]

In dieser Aufgabe geht es um die Erweiterung von *Schnittstellen* mittels `extends`, etwas, was wir bisher noch nicht aktiv genutzt haben.

In der vorliegenden Aufgabe gehen wir zurück zu den *booleschen Ausdrücken* des letzten Übungszettels. Um es nicht zu kompliziert zu machen, verwenden wir als Ausgangspunkt für's erste nicht die komplexe Version aus der letzten Aufgabe von Zettel 5, sondern die aus Aufgabe 2.[1] Die booleschen Ausdrücke sollen nachträglich um *Negation* erweitert werden, sodaß folgende Grammatik realisiert wird:

$$expr_B ::= Const \quad | \quad expr_B \vee expr_B \quad | \quad expr_B \wedge expr_B \quad | \quad \neg expr_B \tag{1}$$

Wenn wir uns anschauen, was wir bereits haben, scheint klar, was zu tun ist: Wir erweitern die abstrakte Klasse `B_Expr_D` einfach um eine weitere Unterklasse für die Negation, erweitern ebenso das Interface `B_ExpressionVisitor` und die konkreten Visitorenklassen. Abbildung 1 zeigt die Situation. Die in dieser Aufgabe geforderten Teile sind mit gepunkteten Pfeilen dargestellt.[2]

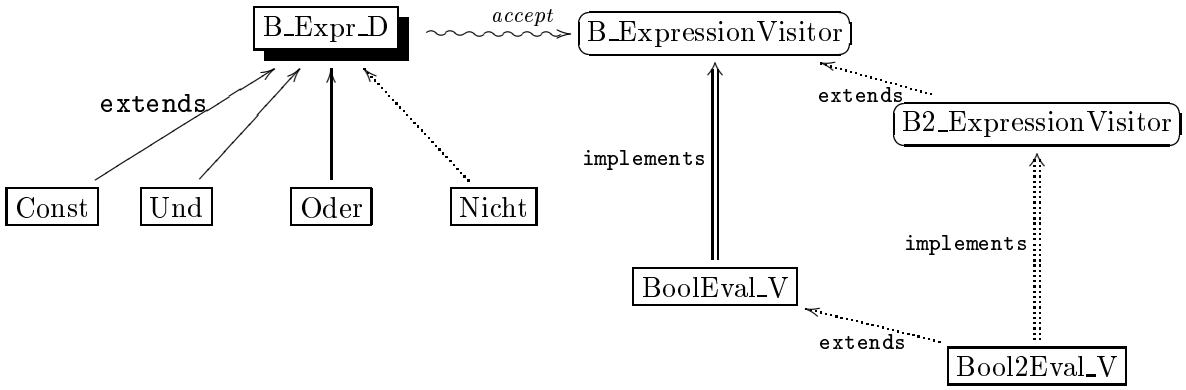

Abbildung 1: Boolesche Ausdrücke

[1] Sobald alle Gruppen ihre Vorschläge abgegeben haben, lege ich meinen Vorschlag auf das Netz.

[2] Die Form der Kästen und Pfeile ist, soweit ich weiß, aus keiner der verschiedenen graphischen OO-Entwurfsmethodiken entlehnt. Das Bild soll nur zur Veranschaulichung dienen.