Universität Koblenz-Landau WS '03/04
Institut für Informatik
Prof. Dr. Bernhard Beckert · www.uni-koblenz.de/~beckert
Vladimir Klebanov · www.uni-koblenz.de/~vladimir

Praktikum

Entwicklung objektorientierter Software mit formalen Methoden

Aufgabenblatt 1

Aufgabe 1

Installieren Sie das KeY-System. KeY ist eine Erweiterung des Borland Together Control-Center. (Für dieses Übungsblatt brauchen Sie allerdings nur die Standard-Funktionalität von TogetherCC.)

- Loggen Sie sich (hier im Poolraum) ein; X11 muß laufen
- Stellen Sie sicher, daß Sie die Konfigurationspakete `JAVA` und `TOGETHER` verwenden. Unter `http://www.uni-koblenz.de/~admin/Info/cshrc.html` finden Sie allgemeine Informationen zu Konfigurationspaketen.
- Laden Sie von der Praktikums-Webseite

  `http://www.uni-koblenz.de/~vladimir/teaching/lab/ws0304/`

  die Bytecode-Distribution des KeY-Systems herunter.
- Entpacken Sie das heruntergeladene Archiv: `tar xvfz KeY_vxxx.tgz`
- Starten sie die Installationsroutine: `java -jar setup.jar`. Die erforderlichen Bibliotheken befinden sich in `/home/vladimir/keylib`. TogetherCC befindet sich in `/usr/local/Together`
- Starten sie das KeY-System.

Wenn Sie sich das System zu Hause installieren wollen, wenden Sie sich an die Praktikumsbetreuer.

Aufgabe 2

Um Ihnen das Arbeiten in der Gruppe zu erleichtern, sollen Sie das Revisionskontrollsystem CVS verwenden. Dieses bietet neben vielen weiteren Vorteilen, wie dem Rücksetzen auf einen älteren Stand oder dem Verfolgen von Änderungen, unter anderem die Möglichkeit, parallel an verschiedenen Aufgaben zu arbeiten und sie im Anschluss wieder zu einer zu verbinden.

Zunächst muss dazu die Datenbasis des Projektes eingerichtet werden. Als erstes setzen Sie die Umgebungsvariable `CVSROOT` auf den Wert `/home/vladimir/PRAKTCVS<Gruppe-Nr>`. In der (T)C-Shell geben Sie dafür

```
setenv CVSROOT /home/vladimir/PRAKTCVS<Gruppe-Nr>
```

ein. Wenn Sie mit CVS arbeiten, muß diese Variable stets gesetzt sein, da sie die Lage der CVS-Datenbasis ("Repository") angibt. Führen Sie desweiteren folgende Kommandos aus:

```
cvs init
mkdir praktikum
cd praktikum
mkdir -p uebungsblaetter/1/
mkdir -p projekt/diagramme
mkdir -p projekt/src
mkdir -p projekt/bin
```

Stellen Sie nun sicher, dass Sie sich im Unterverzeichnis `praktikum` befinden und führen Sie folgenden Befehl aus:

```
cvs import -m "Initialer Check-In" praktikum keyprkt<nr> start
```

import veranlasst CVS, alle Verzeichnisse und Dateien, die es unterhalb der aktuellen Position findet, in das Repository aufzunehmen.

-m spezifiziert einen Kommentar, der kurz die Änderungen beschreibt.

praktikum wird das neue Top-Level Verzeichnis

start ist eine initiale Markierung ("Tag"), unter dem sich der soeben abgelegte Stand wieder zurückholen läßt.

Verlassen Sie das Verzeichnis `praktikum` und legen Sie zwei neue Verzeichnisse, zum Beispiel `mueller` und `meier`, an. Wechseln Sie nach `mueller` und führen Sie folgenden Befehl aus:

```
cvs co praktikum
```

ein anschließendes `ls praktikum/` sollte folgendes Bild liefern:

```
CVS projekt uebungsblaetter
```

Wechseln Sie in das Unterverzeichnis `praktikum` und legen Sie dort eine Textdatei `Gruppe<nr>.txt` an, in der Sie die Namen der Mitglieder Ihrer Gruppe ablegen.

Als nächstes Markieren Sie sie als hinzuzufügende Datei mit dem Befehl:

```
cvs add Gruppe<nr>.txt
```

Beachten Sie bitte, dass die bisherigen Änderungen, noch nicht vom Repository übernommen wurden, dafür ist ein zusätzlicher Arbeitsschritt, das sogenannte Einchecken, notwendig.

Führen Sie vor jedem Einchecken zunächst eine Aktualisierung `cvs update` Ihres Arbeitsverzeichnis (also z.B. `mueller`) durch. Somit ist sichergestellt, dass vor dem Einchecken, von anderen Mitgliedern der Gruppe, veränderte Dateien berücksichtigt werden und Sie nicht versehentlich, deren Änderungen eliminieren. Ein abschließendes

```
cvs commit -m "das wurde geaendert"
```

legt die Änderungen im Repository mit dem übergebenen Kommentar (`-m`) ab.

Wechseln Sie ins Verzeichnis `meier`. Sofern Sie im vorherigen Arbeitschritt wie oben noch keine Version ausgecheckt haben, machen Sie das jetzt, andernfalls reicht auch ein einfaches `cvs update`. Anschließend sollte sie auch hier in dem entsprechenden Unterverzeichnis die Datei `Gruppe<nr>.txt` vorfinden. Spielen Sie nun ein bisschen mit CVS herum, indem Sie z.B. in `meier` und `mueller` die Datei `Gruppe<nr>.txt` an der gleichen (und/oder verschiedenen) Stellen abändern und Sie sie dann nacheinander einchecken.

CVS erlaubt es Markierungen ("Tags") zu setzen, unter denen sich bestimmte Versionen, abrufen lassen. Dafür stellen Sie zunächst sicher, dass alle Dateien in Ihrem Arbeitsverzeichnis in der vorliegenden Version, bereits im Repository vorhanden sind (also schon einmal eingecheckt wurden). Eine Markierung setzen Sie mit:

`cvs tag` <name>

Damit lässt sich dann über `cvs co -r` <*name*> `praktikum` genau diese Version wieder auschecken.

Eine sehr empfehlenswerte (und frei verfügbare) Anleitung für CVS ist "Open Source Development With CVS" von Karl Fogel, zu finden unter:

`http://cvsbook.red-bean.com/`

Aufgabe 3

Die Bibliothek einer Informatik-Fakultät besitzt Bücher und Zeitschriften, die an Kunden der Bibliothek, darunter Studenten, Mitarbeiter und Professoren, ausgeliehen werden können. Je nach Personengruppe können die Bücher und Zeitschriften unterschiedlich lange ausgeliehen werden. Wenn etwas ausgeliehen wird, wird das Datum der Ausleihe notiert, um später die Leihfrist überprüfen zu können. Ist die Leihfrist überschritten, wird der Ausleiher ermahnt.

Beschreiben Sie dieses Szenario möglichst genau mit der UML. Bitte beschreiben Sie möglichst nicht mehr als oben angegeben. Modellieren Sie mithilfe von TogetherCC, das sie in der ersten Aufgabe installiert haben.

(a) Erstellen Sie ein UML-Use-Case-Diagramm

(b) Erstellen Sie ein UML-Klassendiagramm.

(c) Erstellen Sie ein UML-Sequenzendiagramm.

(d) Erstellen Sie ein UML-Zustandsmaschinen-Diagramm, das den Zustand eines ausgeliehenen Gegenstands beschreibt. Sie können hier auch mitmodellieren, dass man Bücher und Zeitschriften vormerken und die Ausleihfrist verlängern kann.

Aufgabe 4

In dem Buch *A Programming Approach to Computability* von Kfoury, Moll und Arbib (Springer Verlag 1982) findet sich auf S. 19f die folgende Definition einer einfachen (künstlichen) Programmiersprache:

> We define a **while**-program to be a finite (possibly empty) sequence of *statements*, separated by semicolons, and preceded by **begin** and followed by **end**:
>
> $$\textbf{begin}\ S_1; S_2; \ldots; S_n\ \textbf{end}$$
>
> where $S_1, S_2, \ldots, S_n$ are arbitrary statements and $n \geq 0$.
>
> Statements are defined inductively. The "assignment statements" form the basis step, while the induction step involves the formation of "**while** statements" and "compound statements".
>
> In **while**-programs, the values assigned to variables will be non-negative integers. An *assignment statement* has one of the following forms:

1. $X := 0$
2. $X := succ(Y)$
3. $X := pred(Y)$

where *succ* is the "successor" function and *pred* is the "predecessor" function. [...]

X and Y in the above instructions stand for arbitrary variable names, possibly identical. [...]

A **while** statement has the following form:

$$\textbf{while } X \neq Y \textbf{ do } S \quad ,$$

which means "while the value of variable X is not equal to the value of variable Y, repeat the action specified by statement S. [...]

Finally, a "compound statement" has the form:

$$\textbf{begin } S_1; S_2; \ldots; S_m \textbf{ end}$$

where $S_1, S_2, \ldots, S_n$ are arbitrary statements and $m \geq 0$. A compound statement such as the one given here simply means that we have to carry out the actions prescribed by $S_1, S_2, \ldots,$ and S_n in succession and in the given order.

Geben sie ein UML Klassendiagramm an, das die *abstrakte Syntax* der **while**-Sprache modelliert. D.h. die Schlüsselwörter **begin**, **end**, etc. und die Semikolons brauchen nicht modelliert zu werden. Auch die Semantik braucht nicht beachtet zu werden. Das Klassendiagramm soll lediglich die strukturellen Beziehungen der syntaktischen Elemente "program", "statement", "variable", etc. widerspiegeln. Benützen Sie das Programm Together-CC.

Abgabe bis — noch offen
Es muss pro Gruppe nur *eine* Lösung abgegeben werden.
Die Abgabe der Übungsblätter erfolgt mit dem CVS System. Dazu legen Sie die abzugebenden Dateien im CVS ab und markieren die abzugebende Version der Dateien mit "`LoesungsBlatt`<*nr*>" wie in Aufgabe 2 beschrieben. Die Lösungen sollten vorzugsweise im dazugehörigen Unterverzeichnis `uebeungsblaetter/nr/` vorzufinden sein, zumindest aber ein Hinweis auf den Ort der Lösungen.
Einige Aufgaben verlangen eine schriftliche Bearbeitung, diese ist dann je nach Komplexität als ASCII, html, ps- oder pdf-Dokument abzugeben. Auf *keinen* Fall im MS Word `doc`-Format.

Materialien
`http://www.uni-koblenz.de/~vladimir/teaching/lab/ws0304/`

Bernhard Beckert: Zi. MB 221, Tel. 287-2775, Email: `beckert@uni-koblenz.de`
Vladimir Klebanov: Zi. MB 230, Tel. 287-2781, Email: `vladimir@uni-koblenz.de`