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

Handout 6: Prototyp

Ausgabetermin: 8. Mai 2000

Die Programmieraufgabe dieses Semesters ist die Entwicklung und Implementierung eines *graphischen Werkzeugs* zur Modellierung, und Untersuchung reaktiver, paralleler Systeme.

Die Aufgabe wird in verschiedene Teile zerlegt werden, wobei jede der Arbeitsgruppen für einen Teil verantwortlich sein wird. Aufgaben werden beispielsweise Teile einer Oberfläche sein, die graphische Eingabe, Anbindung an andere Tools, Implementierung einer Zwischensprache, ein Parser, Speichern der Modelle im Dateisystem und — je nach Anzahl der Gruppen — anderes mehr.

Wie angekündigt, zerfällt das Praktikum in zwei Teile:

1. Implementieren eines *Prototyps* und
2. Entwickeln und Implementieren des gesamten PEST.

Für den ersten Teil der Aufgabe sind für die 8-stündigen Teilnehmer) *2 Wochen* vorgesehen. Er dient unter anderem zum Aufwärmen, allerdings ist ein lauffähiger Prototyp Bestandteil für den 8-stündigen Schein, nicht alleine die gewählte Teilaufgabe am Ende. Mit Aufwärmen, beziehungsweise Prototyping ist Folgendes gemeint: die Phase dient dazu, sich mit der etwas größeren Aufgabe vertraut zu machen. Am Ende der der ersten Phase sollte man neben einem vorführbaren Prototyp Folgendes erreicht haben:

- Beherrschen der Programmierumgebung,[1] das ist ohnehin offensichtlich.
- Falls notwendig, Vertrautheit im Umgang mit Java, insbesondere soll man nach dem ersten Teil die Verwendung von *Paketen* und der unterschiedlichen *Sichtbarkeitsstufen* zur Modularisierung und Strukturierung größerer Aufgaben beherrschen, denn die eigentliche Aufgabe wird, neben der feineren Gliederung in Klassen, in einzelne Pakete unterteilt werden.
- grobes Bild von der Gesamtaufgabe (es wird dazu noch eine Besprechung geben).

Der erste Teil des Praktikums wird durch eine *Projektbesprechung* abgeschlossen. Das Ergebnis der Besprechung wird die Unterteilung der Aufgabe sein.

Aufgabe 1: [Prototyp]

Für den *Prototyp* nehmen wir uns den Teil der Aufgabe vor, der am schnellsten zumindest sichtbare Ergebnisse bringt.[2] In unserem Fall ist das der graphische Teil, also die

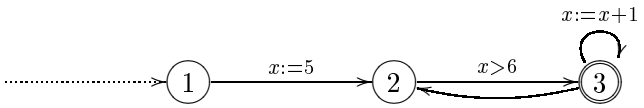
[1] Ausgenommen CVS, damit starten wir erst im zweiten Teil.
[2] Das ist ein Grundgedanke des Prototypingansatzes bei der Softwareentwicklung.

Herstellung einer prototypischen *graphischen Oberfläche*. Diese Aufgabe ist für jeder der Gruppen die selbe.

Was die Gestaltung der Oberfläche betrifft (Knöpfe, Menues, etcetc.), sind der Phantasie keine Grenzen gesetzt. Da es sich um einen Prototyp handelt, kann es sich dabei teilweise um noch nicht funktionale Interaktionsmöglichkeiten handeln. Aber bitte im Auge behalten, daß Phantasie kein Synonym für Ergonomie ist...

Auch wenn die Gestaltung der Oberfläche ist in der ersten Phase relativ freigestellt ist, sollen folgende Interaktionsmöglichkeiten *funktionsfähig* sein.

Graphische Eingabe Viele Editoren heutzutage bieten — ob zum Vorteil des Benutzers sei dahingestellt— die Möglichkeit, graphisch zu arbeiten (UML, SDL, ...) Wie wollen einfache reaktive Programme als *Zustands-Übergangsdiagramme* darstellen, für den Prototyp stelle man sich eine einfache automatenartige Notation vor:



Das Diagramm ist ein unverbindlicher Vorschlag, was die Gestaltung betrifft, haben Sie Freiheit.

Der Teil beinhaltet das Zeichnen von

- Zuständen
- Übergängen
- Anfangszuständen.

Übergänge und Zustände sollen beschriftet sein. In dem Bild sind die Zustände mit Zahlen beschriftet, es sollen auch Identifier möglich sein. Es sollen mehrere derartige Transitionsdiagramme (z.B. nebeneinander) gezeichnet werden können.[3] Eine Nichtlösung ist, dem Benutzer ein Kritzelapplet zur Verfügung zu stellen, mit anderen Worten, die genannten graphischen Elemente sollen in irgendeiner Form als eigenständige graphische Elemente auf die Oberfläche zu bekommen sein.

Speichern Es soll möglich sein, das Ergebnis in eine vom Benutzer angegebene Datei zu speichern. Diese Dateien sollen zur weiteren Bearbeitung in den Editor eingelesen werden können.

Sonstiges Daneben soll Ihr Lösungsvorschlag folgende eher formale Anforderungen erfüllen:

- Das Tool soll stand-alone laufen können, also nicht (nur) als ein Applet im Browser
- Es soll aus mindestens zwei Java-*Paketen* bestehen. Überlegen sie sich dazu eine sinnvolle Unterteilung.

[3] eine Aufwendigere Lösung wäre, für jedes ein einzelnes Fenster aufzumachen.

- „Vervollständigen“ Sie die Oberfläche mit weiteren Vorschlägen für die Interaktion mit dem Werkzeug. Diese Teile brauchen nicht funktional sein.

Wer will, kann auch folgendermaßen vorgehen und (bei geschickter Unterteilung) Arbeit sparen: es ist möglich, daß sich *je zwei Arbeitsgruppen* von *8stündigen Teilnehmer* zusammenarbeiten, geschickterweise wird man dann jede Gruppe ein Paket programmieren lassen. Der Zusammenschluß zu 4-er Gruppen für den Prototyp ist optional.

Anmerkung Die genannten Aufgaben lassen sich natürlich auf vielfältige Weise lösen. Eine gute Lösung bedenkt, daß der erste Prototyp nicht alles sein wird, was wir machen werden. Sie könnten beispielsweise bedenken, daß man am Ende nicht nur Zeichnen will, sondern beispielsweise auch selektiv löschen, daß man allgemein eine Zeichnung ändern können will. Wie geeignet ist die Lösung in Hinblick auf die Tatsache, daß das Tool nicht nur zeichnen soll, sondern beispielsweise auch die Automaten simulieren?

Für diese Aufgabe sind speziell `java.awt` und auch `java.awt.events` aus der Javabibliothek relevant (siehe auch Kapitel 7 und 8 aus [**?**], bzw. Lektion 8 und 9 aus dem Java-Praktikum des letzten Semesters).

Arbeitsstil. Gehen Sie die Entwicklung des Prototypen *systematisch* an. Bedenken Sie, daß die Gruppenarbeit bei der Entwicklung von PEST von Ihnen vor allem *Selbständigkeit und Verläßlichkeit* erfordert. Am besten stellen Sie sich bei der Entwicklung des Prototypen darauf ein. Dazu gehen Sie bitte folgendermassen vor:

1. Machen Sie einen Plan! Gliedern Sie die Aufgabe in Teilaufgaben und machen Sie einen Terminplan dazu. Schreiben Sie es auf!
2. Lösen Sie die Teilaufgaben und erfüllen Sie dabei den Terminplan.

Hier ist ein nicht verbindlicher Vorschlag zum Plan.

Milestone 1. Machen Sie sich u.a. folgende Fragen klar: Welche Features soll der Editor unterstützen? Welche Datentypen brauchen Sie um die graphischen Objekte intern darstellen zu können? Wie werden die Diagramme gespeichert gespeichert?

Resultat. Definitionen von zwei Paketen, die die Datentypen definieren und Methoden auflisten, die die Funktionalität zur Verfügung stellen. Diese Methoden sind *Stubs:* Funktionen mit Namen, Parametern und Rückgabetyp, aber ohne Inhalt, der die Funktionalität zur Verfügung stellt.

Zeit. 5 Tage

Milestone 2. Programmieren Sie den Prototypen. Dazu ergänzen Sie die Stubs aus dem ersten Punkt mit Code, der die Funktionalität implementiert.

Resultat. Kompilierfähiger Javacode des Prototypen.

Zeit. 5 Tage

Milestone 3. Testen Sie den Prototypen.

Resultat. Fertiger Prototyp.

Zeit. 5 Tage

Bitte bedenken Sie, daß Sie während der Gruppenarbeit selbst ähnlichen Plan erstellen müssen und daß wir nach dem gleichen Muster vorgehen werden. Zum Beispiel, werden Sie aufgefordert, Stubs zur Verfügung zur stellen, die die Schnittstelle zur anderen Gruppen darstellen. Es wird von Ihnen erwartet, daß Sie die Stubs rechtzeitig implementieren, damit noch Zeit für den Test des Gesamtsystems bleibt.