Christian-Albrechts-Universität zu Kiel
Institut für Informatik
Lehrstuhl für Programmiersprachen und Übersetzerkonstruktion
Prof. Dr. M. Hanus, F. Reck

Institut für Informatik

14. Übung zur Vorlesung „Programmierung“

keine Abgabe

Hinweis: In der Woche vom 08. bis zum 12.2. werden zu den Übungsterminen freiwillige Fragestunden angeboten. Konkrete Fragen sollten Sie möglichst vorab per Mail an Ihren Übungsgruppenleiter richten.

Hinweise zur Klausur:

- Wir werden Ihnen einen Sitzplatz zuteilen, auf dem ein **personalisiertes Deckblatt** für Sie bereit liegen wird. Bitte unterschreiben Sie den oberen Teil dieses Deckblatts und legen Sie es zusammen mit Ihrem **Personal- und Studienausweis** an Ihren Platz. Die Verteilung der Sitzplätze erfolgt alphabetisch nach Nachnamen.
- Anschließend werden wir die Aufgabenzettel **verdeckt** verteilen. Diese dürfen erst umgedreht werden, wenn alle Zettel verteilt sind. Nachdem die Zettel umgedreht wurden, haben Sie **90 Minuten** Zeit, die Klausur zu bearbeiten.
- Während der Klausur werden wir die **Anwesenheit** anhand Ihrer ausgelegten Deckblätter und Ausweise kontrollieren.
- **Elektronische Geräte** jeglicher Art (z.B. Laptops, Mobiltelefone oder iPods) und schriftliche Unterlagen (wie Bücher oder Ihre Vorlesungsmitschrift) dürfen **nicht** zur Klausur mitgebracht werden.
- Bitte bringen Sie genügend **unbeschriebene Blätter** zur Bearbeitung der Klausuraufgaben mit. Beginnen Sie **jede Aufgabe auf einem neuen Zettel** und beschreiben Sie jedes Blatt nur einseitig!
- Während der **letzten halben Stunde** der Bearbeitungszeit dürfen Sie den Raum nicht mehr verlassen – auch nicht, wenn Sie abgeben.
- Bei der Abgabe werden Ihre Zettel mit dem Deckblatt **zusammengetackert**. Den Aufgabenzettel dürfen Sie behalten.
- Die Einsichtnahme in die korrigierten Klausuren findet am Mittwoch den 3. März zwischen 9 und 12 Uhr im Raum 715 des Hochhauses statt.

Aufgaben zu Java für die vorlesungsfreie Zeit

Aufgabe 1

Implementieren Sie eine Klasse `Liste<T>`. Eine Liste besitzt die Merkmale `first` und `rest`, wobei `first` den Typ `T` und `rest` den Typ `Liste<T>` besitzt. Die leere Liste soll durch `null` repräsentiert werden. Der Aufruf

```
new Liste<String>("eins",new Liste<String>("zwei", new Liste<String>("drei",null)))
```

soll dann die Liste *(“eins” “zwei” “drei”)* erzeugen.
Fügen Sie Ihrer Listenimplementierung die Prozeduren `set_first` und `set_rest` hinzu, die die Merkmale `first` bzw. `rest` eines Objekts vom Typ `Liste<T>` auf neue Werte setzen.

Aufgabe 2

Verwenden Sie Ihre Listen aus Aufgabe 1, um die aus der Vorlesung bekannte Warteschlange in Java zu implementieren. In Scheme wurde die Warteschlange wie folgt definiert:

```
(define-struct ws (anfang ende))
```

Diese Struktur können Sie nun als Klasse `Warteschlange<T>` umsetzen, wobei die beiden Komponenten der Struktur zu Merkmalen der Klasse mit dem Typ `Liste<T>` werden.
Um eine neue Warteschlange zu erzeugen, wurde in Scheme der Konstruktor `konstr-warteschlange` verwendet. Wird in einer Java-Klasse nicht explizit ein Konstruktor angegeben, erhält sie automatisch einen Konstruktor ohne Argumente, der alle Merkmale mit Standardwerten vorbelegt. Konstruktoren mit Argumenten müssen explizit angegeben werden. Da die leere Liste durch `null` repräsentiert wird, muss für die Klasse Warteschlange kein Konstruktor definiert werden.
Nun werden noch die Operationen auf Warteschlangen benötigt:

- `leere-warteschlange?`
- `anfang`
- `hinzufuegen-warteschlange!`
- `entfernen-warteschlange!`

Diese können als Methoden der Klasse `Warteschlange` umgesetzt werden. Beachten Sie, dass die Zeichen `-`, `?` und `!` in Java-Bezeichnern nicht vorkommen dürfen.

Aufgabe 3

Nun können Sie den aus der Vorlesung bekannten Simulator für digitale Schaltkreise in Java implementieren. Dazu muss zunächst überlegt werden, welche Klassen benötigt werden. Ein guter Anhaltspunkt für benötigte Klassen sind die Nomen, die in einer Programmbeschreibung vorkommen. Da hier ein Scheme-Programm umgesetzt werden soll, sollten Sie sich die Datenstrukturen, die mit `define-struct` erzeugt werden und die Objekte mit Nachrichtenweitergabe genauer ansehen. Letztere lassen sich besonders direkt in Java-Klassen übersetzen.
Ihre Klasse `Draht` sollte ein Merkmal enthalten, das in einer Liste alle Gatter aufnehmen soll, die Signale von einem konkreten Draht verarbeiten. Um dies zu ermöglichen, müssen alle Gatter den *gleichen* Typ haben. Dazu müssen alle verschiedenen Gatter von einer Klasse wie z.B.

```
public class Gatter {
    public void verarbeite_signal(){}
}
```

erben und die Methode `verarbeite_signal` überschreiben. Da auch Sonden Signale von Drähten verarbeiten, sollten Sie auch diese zu einer Unterklasse von `Gatter` machen. Die Liste in der Klasse `Draht` sollte also den Typ `Liste<Gatter>` haben.
An vielen Stellen im Scheme-Programm wird die Konstante `der-plan` verwendet. Um diese Konstante in Java nachzubilden, können Sie sich eine Klasse `Der_Plan` definieren, deren Merkmale und Methoden statisch sind. Diese Klasse sollte eine geordnete Liste von Zeitsegmenten enthalten, die wiederum jeweils eine Warteschlange mit Vorgängen enthalten. In Scheme konnten diese Vorgänge direkt als Prozeduren umgesetzt werden. In Java werden Prozeduren höherer Ordnung jedoch nicht unterstützt. Um das Verhalten dieser Prozeduren nachzubilden, können Sie sich eine Klasse `Vorgang` definieren, die die nötigen Informationen enthält, um einen solchen Vorgang auszuführen. In diesem Fall sind dies der Draht, dessen Wert gesetzt werden soll, und der entsprechende Wert. Zusätzlich sollte diese Klasse eine Methode `ausfuehren` enthalten, die den Draht dann tatsächlich auf den neuen Wert setzt.
Viele Prozeduren in der Scheme-Version sind global definiert. Für diese sollten Sie sich überlegen, zu welcher Klasse sie am besten passen. Zum Beispiel passt die Prozedur `set-signal!` am besten zur Klasse `Draht`. Auf diese Weise kann auch der Parameter `draht` eingespart werden.
Um Ihre Implementierung zu testen, definieren Sie sich eine Klasse `Main` mit einer `main`-Methode, in der Sie einen Schaltkreis aus Drähten, Gattern und Sonden aufbauen und die Simulation starten.