Übungen zu Softwareentwicklung III, Funktionale Programmierung

Blatt 10, Woche 12

CLOS und Methodenkombination

Leonie Dreschler-Fischer

WS 2012/2013

Ausgabe: Freitag, 11.1.2013,

Abgabe der Lösungen: bis Montag, 21.1.2013, 12:00 Uhr per email bei den Übungsgruppenleitern.

Ziel: Die Übungsaufgaben auf diesem Blatt dienen dazu, Mehrfachvererbung und Methodenkombination kennenzulernen. Teil 2 des Übungsblattes enthält typische Klausuraufgaben, mit denen Sie sich auf die Modulabschlußklausur vorbereiten können.

Bearbeitungsdauer: Die Bearbeitung sollte insgesamt nicht länger als 5 Stunden dauern.

Homepage:
http://kogs-www.informatik.uni-hamburg.de/~dreschle/teaching/Uebungen_Se_III/Uebungen_Se_III.html

Bitte denken Sie daran, auf den von Ihnen eingereichten Lösungsvorschlägen *Ihren Namen und die Matrikelnummer, den Namen der Übungsgruppenleiterin / des Übungsgruppenleiters* und *Wochentag und Uhrzeit der Übungsgruppe* anzugeben, damit wir ihre Ausarbeitungen eindeutig zuordnen können.

1 CLOS und generische Funktionen

1.1 Definition von Klassen

Bearbeitungszeit 1,5 Std., 15 Pnkt.

Definieren Sie geeignete Klassen in CLOS, um Literaturbeiträge repräsentieren zu können. Eine wissenschaftliche Veröffentlichung hat als Attribute mindestens

- einen eindeutigen Schlüssel,
- die Namen der Autorinnen und Autoren,
- das Erscheinungsjahr,
- sowie den Titel der Veröffentlichung.

Je nach Art der Veröffentlichung sind weitere Angaben nötig, um die Arbeit zitieren zu können:

Bücher: Für ein Buch werden zusätzlich Angaben zum Verlag, dem Verlagsort, der Reihe und der Seriennummer in der Reihe benötigt.

Sammelbände: Zusätzlich zu den Buchangaben: Name des Herausgebers, Seitenangeben zum Artikel im Buch.

Zeitschriftenartikel: Der Name der Zeitschrift, Nummer des Bandes, Nummer des Heftes, eventuell der Erscheinungsmonat.

Erzeugen Sie Objekte für die folgende Bibliographie:

1. Nessie (1790). Mein Leben im Loch Ness: Verfolgt als Ungeheuer, Band 1 der Reihe: Die besondere Biographie. Minority-Verlag, Inverness.

   Ein Beispiel für ein Buch

2. Prefect, F. (1979). Mostly harmless – some observertions concerning the third planet of the solar sytem. In Adams, D., editor, The Hitchhiker's Guide to the Galaxy, volume 5 of "*Travel in Style*". Galactic Press, Vega-System, 3^{rd} planet, 1500 edition, p. 500.

   Ein Beispiel für einen Sammelband

3. Wells, H. G. (3200). Zeitmaschinen leicht gemacht. *Heimwerkerpraxis für Anfänger,* 500(3).

   Ein Beispiel für einen Zeitschriftenartikel

1.2 Generische Funktionen und Methoden

Definieren Sie eine generische Funktion „cite“, die ein Literaturbeitrag-Objekt als Argument erhält und für diesen Beitrag einen String mit dem korrekten Zitat erzeugt: Beispiel:

```
(cite Nessie-2000) ⟶
 "Nessie (1790). Mein Leben im Loch Ness: Verfolgt als Ungeheuer,
Band 1 der Reihe: Die besondere Biographie.
 Minority-Verlag, Fantasien."
```

Implementieren Sie geeignete Methoden für die generische Funktion „cite“ und erproben Sie diese an den obigen Beispielen.

2 CLOS und Vererbung

2.1 Definition von Klassen

Bearbeitungszeit 1/2 Std., 5 Pnkt.

1. Definieren Sie als CLOS-Klasse eine Klasse von Fahrzeugen und spezialisieren Sie diese Klassen für unterschiedliche Medien, in denen sich die Fahrzeuge bewegen: Bodengebundene Landfahrzeuge, schwimmfähige Wasserfahrzeuge, flugfähige Luftfahrzeuge.

2. Definieren Sie Klassen von Mehrzweckfahrzeugen:

   (a) Ein Amphibienfahrzeug, das sich zu Wasser und zu Lande bewegen kann.

   (b) Ein Batman-Superfahrzeug, das sich zu Wasser, zu Lande und in der Luft bewegen kann.

2.2 Definition von Operationen, Methodenkombination

Bearbeitungszeit 1 Std., 9 Pnkt.

Die Klasse Fahrzeug soll folgende Operationen bieten:

1. Abfrage des Mediums, in dem sich das Fahrzeug bewegt,

2. Abfrage der Maximalgeschwindigkeit,

3. Abfrage der Zuladung (Tragfähigkeit)

4. Abfrage des Verbrauchs pro 100 km

5. Abfrage der der Passagierzahl

Spezifizieren Sie generische Funktionen als Signatur für die Operationen der Klasse Fahrzeug und diskutieren Sie, welche Methodenkombination für die jeweilige Operation sinnvoll erscheint.

3 Zur Wiederholung und Klausurvorbereitung

(Bearbeitungszeit 2 Std.)

1. Zu welchen Werten evaluieren die folgenden Racket-Ausdrücke? 5 Zusatzpnkt.

   (a) (**max** (**min** 2 (− 2 3)))

   (b) ‘(+ ,(− 2 4) 2)

   (c) (car '(Alle meine Entchen))

   (d) (cdr '(schwimmen auf (dem See)))

   (e) (**cons** 'Listen '(sind einfach))

   (f) (**cons** 'Paare 'auch)

   (g)
   ```
   (equal?
     (list 'Racket 'Prolog 'Java)
     '(Racket Prolog Java))
   ```

   (h)
   ```
   (eq?
    (list 'Racket 'Prolog 'Java)
    (cons 'Racket '(Prolog Java)))
   ```

   (i)
   ```
   (map
     (lambda (x) (* x x x))
     '(1 2 3))
   ```

   (j) (filter odd? '(1 2 3 4 5))

   (k) ((curry **min** 6) 2)

   (l) ((curry = 2) 2)

2. Zur Syntax von Racket: Gegeben seien die folgenden Definitionen: 5 Zusatzpnkt.

```
(define *a* 10)
(define *b* '*a*)
(define (merke x)(lambda () x))
(define (test x)
 (let ((x (+ x *a*))))
 (+ x 2))
```

Haben die folgenden Ausdrücke einen wohldefinierten Wert und zu welchem Wert evaluieren sie?

1. `*a*`
2. `(+ *a* *b*)`
3. `(+ (eval *a*) (eval *b*))`
4. `(and (> *a* 10) (> *b* 3))`
5. `(or (> *a* 10) (/ *a* 0))`
6. `(+ 2 (merke 3))`
7. `(+ 2 ((merke 3)))`
8. `(test 4)`

3. Geben Sie geeignete Racket-Ausdrücke an, um die folgenden Werte zu berechnen: 2 Zusatzpnkt.

   (a) $3 \times 4 + 5 \times 6$

   (b) $\sqrt{1 - \sin^2(x)}$

4. Definieren Sie die folgenden beiden Funktionen c und mytan als Racket-Funktionen: 2 Zusatzpnkt.

$$\begin{aligned} \mathrm{c}(a,b) &= \sqrt{a^2+b^2}, \quad a,b \in R \\ \mathrm{mytan}(\alpha) &= \frac{\sin\alpha}{\sqrt{1-\sin^2\alpha}}, \quad -\frac{\pi}{2} < \alpha < \frac{\pi}{2} \end{aligned}$$

5. Wandeln Sie die folgenden Ausdrücke in die Präfixnotation von Racket um: 2 Zusatzpnkt.

   (a) $1 + 4/2 - 1$

   (b) $\frac{2-\frac{1+3}{3+2*3}}{\sqrt{3}}$

6. Wandeln Sie den folgenden Ausdruck in Infixnotation um: `(* (+ 1 2 3) (- 2 3 (- 2 1)))` — 1 Zusatzpnkt.

7. Definieren Sie eine *rekursive* Funktion `(laengen xss)` in Racket, die folgendes leistet:
   Gegeben sei eine Liste von Listen. Bilden Sie diese ab auf die Liste der Längen der Unterlisten. — 4 Pnkt.

   Beispiel: `(laengen (1 3 4) ( Auto Bus) () (3 4 5 6))` $\longrightarrow$ `(3 2 0 4)`.

   Geben Sie die Funktion in zwei Varianten an: einmal linear rekursiv ohne Akkumulator und einmal endrekursiv. Woran erkennen Sie eine endrekursive Funktion?

8. Ein abstrakter Datentyp für Längenangaben: — 10 Pnkt.

   Für viele Anwendungen ist es notwendig, nicht nur den numerischen Wert einer physikalischen Größe zu kennen sondern auch die Einheit, in der dieser Wert angegeben ist.

   Implementieren Sie die Zugriffsfunktionen für einen Datentyp „laenge“:

   Eine Längenangabe sei repräsentiert als Liste mit zwei Elementen, nämlich dem numerischen Wert und der Einheit, beispielsweise `(0.5 m)`, `(3 cm)`, `(6.3 km)`, `(6 inch)` usw.

   (a) Definieren Sie eine Konstruktorfunktion `(make-length value unit)`, die aus einem Wert und einer Längeneinheit ein Element vom Typ Längenangabe erzeugt:
   `(make-length 3 'cm)` $\longrightarrow$ `(3 cm)`

   (b) Definieren Sie Selektorfunktionen für den Zugriff auf den Wert einer Längenangabe `(value-of-length len)`
   und die Längeneinheit `(unit-of-length len)`: `(value-of-length (make-length 3 'cm))` $\longrightarrow$ `3`
   `(unit-of-length (make-length 3 'cm))` $\longrightarrow$ `cm`

   (c) Definieren Sie eine Skalierungsfunktion `(scale-length len fac)`, mit der Längenangaben um einen bestimmtem Faktor vergrößert (verkleinert) werden können:
   `(value-of-length (scale-length (make-length 3 'cm) 2))` $\longrightarrow$ `6`

(d) Gegeben sei eine Tabelle `*conversiontable*` mit Umrechnungsfaktoren für Längenangaben. Diese Tabelle sei als Assoziationsliste organisiert, die zu jeder Längeneinheit den Umrechnungsfaktor gegenüber der Angabe in Metern enthält:

```
(define *conversiontable* ;
   '( ;  (unit . factor)
      (m . 1)
      (cm . 0.01)
      (mm . 0.001)
      (km . 1000)
      (inch . 0.0254)
      (feet . 0.3048)
      (yard . 0.9144)))
```

Definieren Sie die folgenden Operationen auf Längenangaben unter Verwendung der Funktionen Konstruktoren und Selektoren `make-length, unit-of-length, value-of-length, scale-length`:

- Auslesen des Umrechnungsfaktors aus der Tabelle `(factor unit)`:
  `(factor 'mm)` $\longrightarrow$ `0.001`

- Normalisierung (Wandlung in Meterangaben) `(length->meter len)`: `(length->meter '(3 cm)))` $\longrightarrow$ `(0.03 m)`

- Vergleich zweier Längenangaben `(length< len1 len2)` und `(length= len1 len2)`:
  `(length< '(3 cm) '(6 km))` $\longrightarrow$ `#t`
  `(length= '(300 cm) '(3 m))` $\longrightarrow$ `#t`

- Addition und Subtraktion zweier Längenangaben `(length+ len1 len2)`:
  `(length+ '(3 cm) '(1 km))` $\longrightarrow$ `(1000.03 m)`
  `(length- '(1.001 km) '(1 m))` $\longrightarrow$ `(1000.0 m)`

(e) Gegeben sei eine Liste von Längenangaben:
`( (6 km) (2 feet) (1 cm) (3 inch))`.
Verwenden Sie Funktionen höherer Ordnung (`reduce, map, filter, iterate, curry`...), um funktionale Ausdrücke für folgende Werte zu schreiben:

- Die Abbildung der Liste auf eine Liste, die die Längenangaben in Metern enthaelt,

- eine Liste aller Längen, die kürzer als 1m sind,
- die Summe der Längenangaben in Metern,
- die Längenangabe mit der kürzesten Länge in der Liste.

Erreichbare Punkte: 43

Erreichbare Zusatzunkte: 17