



Hausaufgabe 5 - Entwurfsrezept zur Funktionsdefinition

Im Zeitungskiosk an der Ecke ist nun auch eine Postfiliale. Die Kunden bringen Gegenstände, die sie mit der Post verschicken wollen, und der Besitzer berät die Kunden, welche Art von Versand am billigsten ist. Das hängt von Breite, Höhe, Tiefe und Gewicht des Gegenstandes ab, der verschickt werden soll. Es gibt verschiedene Formate (Standard, Kompakt, ...). Mehr Informationen zu den Formaten finden Sie auf der Webseite der Post:

- Deutsche Post (http://www.deutschepost.de/de/b/brief_postkarte.html)

Die Tochter des Kioskbesitzers studiert Informatik in Tübingen und möchte ein Programm schreiben, um die Kunden zu beraten welches das günstigste Format für den Versand eines Briefes ist. Sie hat bereits nach dem Top-Down-Prinzip angefangen zu programmieren. Die Datei mit dem bisherigen Programm und der Wunschliste finden Sie auch im Forum zum herunterladen.

Es soll eine Funktion `cheapest-format` programmiert werden, die aus Länge, Breite, Höhe und Gewicht einer Sendung die billigste Versandart berechnet, mit der diese Sendung verschickt werden kann. Falls keine der Versandarten funktionieren würde, soll die Funktion `cheapest-format` stattdessen `#false` zurückgeben.

Bearbeiten Sie dieses Hausaufgabenblatt **alleine**. Stellen Sie dabei sicher, dass sie die gesetzten Lernziele erreichen.

Verwenden Sie als Grundlage zur Bearbeitung dieser Hausaufgabe die Datei `post.rkt`, die Ihnen im Forum zur Verfügung gestellt wird.

Viel Erfolg!

Lernziele

Durch diese Hausaufgabe lernen Sie

- ✓ Funktionen für Summentypen systematisch mit Hilfe des Entwurfsrezepts zur Funktionsdefinition zu implementieren,
- ✓ Programme nach dem Top-Down-Prinzip aus mehreren Funktionen zusammenzusetzen,
- ✓ und das Don't-Repeat-Yourself-Prinzip in der Praxis anzuwenden.

Nach dieser Übung sollen Sie in der Lage sein, sicher mit Summentypen zu programmieren sowie auch komplexe Aufgabenstellungen zu bearbeiten, indem Sie die Aufgabenstellung in Teilaufgaben zerlegen und dann eine Funktion nach der anderen implementieren. Außerdem haben Sie erste Erfahrungen mit der nachträglichen Änderung von Programmen gesammelt und über den Zusammenhang zwischen der Wartbarkeit eines Programmes und seiner Organisation in mehrere Funktionen reflektiert.

Aufgabe 1: Templates für Summentypen

Die Tochter des Kioskbesitzers hat schon einige Datendefinitionen programmiert. Folgende Datendefinitionen sind nützlich, um die Argumente von `cheapest-format` zu beschreiben:

```
; Length is a Number.  
; interp. length of a letter in cm.  
  
; Width is a Number.  
; interp. width of a letter in cm.  
  
; Height is a number.  
; interp. height of a letter in cm.  
  
; Weight is a Number.  
; interp. weight of a letter in g.
```

Bewertungsschema (Gesamtpunktzahl 10 pt)

- ☐ 5pt Zwei Templates sind angegeben.
- ☐ 5pt Die Templates passen zu den Datendefinitionen.

Und folgende Datendefinitionen sind nützlich, um das Ergebnis von `cheapest-format` zu beschreiben:

```
; Format is one of:  
; - "Standard"  
; - "Kompakt"  
; - "Gross"  
; - "Maxi"  
; interp. the products offered by Deutsche Post  
  
; MaybeFormat is one of:  
; - Format  
; - #false  
; interp. the best Format choice, or #false if no Format would work
```

1. Schreiben Sie ein Template für Funktionen, die `Format` verarbeiten.
2. Schreiben Sie ein Template für Funktionen, die `MaybeFormat` verarbeiten.

Aufgabe 2: Aufgabenstellung analysieren und Testfälle schreiben

Nach dem Top-Down-Prinzip hat die Tochter des Kioskbesitzers zunächst die erste Funktion `cheapest-format` fertiggestellt. Die Implementierung der Funktion finden Sie in der Datei `post.rkt`, die Sie im Forum herunterladen können.

Die Funktion `format-possible?`, welche in der Definition von `cheapest-format` verwendet wird, hat die Tochter des Kioskbesitzers auf die Wunschliste geschrieben. Leider hat sie aber die Tests für `cheapest-format` vergessen. Schreiben Sie ausreichend Testfälle für `cheapest-format`.

Bewertungsschema (Gesamtpunktzahl 15 pt)

- ☐ 3pt Testfälle angegeben.
- ☐ 7pt Die Testfälle passen zur Aufgabenstellung (2pt) und decken die Feinheiten der Portoberechnung größtenteils ab (5pt).
- ☐ 5pt Die Testfälle decken die Feinheiten der Portoberechnung vollständig ab.

Aufgabe 3: Top-Down-Prinzip und Summentypen

Als nächstes hat die Tochter des Kioskbesitzers die Funktion `format-possible?` von der Wunschliste programmiert.

Um die weitere Wunschliste zu schreiben, hat Sie auch zusätzliche Datendefinitionen programmiert. Die folgenden Datendefinitionen sind nützlich, um die Signaturen von `in-range?` und `below-maximum` anzugeben:

```
; Maximum is a Number.  
; interp. the upper bound of something  
  
; Minimum is a Number.  
; interp. the lower bound of something
```

Manche Formate müssen ein Verhältnis von Länge zu Breite von mindestens 1.4 haben, bei anderen Formaten ist das Verhältnis egal. Die folgenden Datendefinitionen sind nützlich, um die Signaturen von `aspect-ratio` und `above-minimum?` anzugeben, so dass man `#false` benutzen kann, wenn ein Format keine Anforderungen an das minimale Verhältnis von Länge zu Breite hat:

```
; AspectRatio is a Number.  
; interp. ratio of length to width  
  
; MaybeMinimum is one of:  
; - #false  
; - Minimum  
; interp. the lower bound of something, or  
;      #false if there's no lower bound  
  
(define (maybe-minimum-template maybe-minimum)  
  (if (false? maybe-minimum)  
      ...  
      (... maybe-minimum ...)))
```

Bewertungsschema (Gesamtpunktzahl 45 pt)

- ☐ 5pt Wunschliste vollständig bearbeitet.
- ☐ 10pt Tests angegeben (2pt) für alle Funktionen (3pt), passend zur Aufgabenstellung (2pt) und decken alle Fälle der Argumente und Ergebnisse aller Funktionen ab (3pt).
- ☐ 10pt Funktionsimplementierungen angegeben (2pt) für alle Funktionen (3pt), passend zur Aufgabenstellung (2pt) und weitgehend korrekt (3pt).
- ☐ 5pt Hohe Qualität des Programmes insgesamt (lesbare Formatierung, Wahl geeigneter Namen, ...)
- ☐ 10pt Die Implementierung der Funktionen folgt dem Entwurfsrezept und geeignete Refactorings wurden verwendet.
- ☐ 5pt Das Programm insgesamt funktioniert korrekt.

Hier ist die Wunschliste:

in-range?

```
; Number Minimum Maximum -> Boolean  
; Check whether a number is between a minimum and a maximum
```

above-minimum?

```
; Number MaybeMinimum -> Boolean  
; Check whether a number is above an optional minimum
```

below-maximum?

```
; Number Maximum -> Boolean  
; Check whether a number is below a maximum
```

min-length

```
; Format -> Minimum  
; return minimum length of a format
```

max-length

```
; Format -> Maximum  
; return maximum length of a format
```

min-width

```
; Format -> Minimum  
; return minimum width of a format
```

max-width

```
; Format -> Maximum  
; return maximum width of a format
```

max-height

```
; Format -> Maximum  
; return maximum height of a format
```

max-weight

```
; Format -> Maximum  
; return maximum weight of a format
```

aspect-ratio

```
; Length Width -> AspectRatio  
; compute aspect ratio of letter
```

min-aspect-ratio

```
; Format -> MaybeMinimum  
; return optional minimum aspect ratio of format
```

Implementieren Sie alle Funktionen von der Wunschliste, so dass die Funktion `cheapest-format` wie gewünscht funktioniert. Sie können weitere Funktionen auf die Wunschliste dazuschreiben und dann implementieren, wenn das für die Bearbeitung der Aufgabe sinnvoll ist.

Achtung: Ändern Sie die Definitionen von `format-possible?` und `cheapest-format` **nicht**.

Aufgabe 4: Wartung eines Programmes

Dem Kioskbesitzer fällt auf, dass das fertige Programm zur Portoberechnung keine Postkarten unterstützt und bittet Sie, die Unterstützung für Postkarten nachträglich hinzuzufügen.

1. Erweitern Sie die Datendefinition für `Format` um eine Variante `"Postkarte"` und **ändern** Sie Ihre Lösung aus Aufgabe 3, so dass `cheapest-format` auch Postkarten korrekt unterstützt. Dafür können Sie Funktionen auf die Wunschliste dazuschreiben und dann implementieren, wenn das für die Bearbeitung der Aufgabe sinnvoll ist.
2. Beschreiben Sie in einigen Sätzen, welche Änderungen Sie zum nachträglichen Hinzufügen von Postkarten in der vorherigen Teilaufgabe durchführen mussten. Gehen Sie dabei auch auf folgende Fragen ein:
 - Wieviele Funktionen mussten Sie ändern?
 - Hat die Änderung gerigen oder großen Aufwand erfordert?
 - Hat die vorgegebene Organisation des Programmes in Funktionen den Aufwand für die Änderung verringert oder vergrößert?
3. Stellen Sie sich vor, sie sollten folgende Änderungen durchführen:
 - a. Kompaktbriefe dürfen doppelt so viel wiegen wie bisher.
 - b. Standardbriefe sind doppelt so teuer wie bisher.
 - c. Ein neues Format `"Medium"` wird zwischen `"Kompakt"` und `"Gross"` eingeführt.

Beschreiben Sie in einigen Sätzen, welche dieser Änderungen von der vorgegebenen Organisation des Programmes in Funktionen mit geringem oder großen Aufwand ermöglicht werden. Begründen Sie Ihre Antwort.

4. Schlagen Sie eine alternative Organisation des Programmes in Funktionen vor, die alle Änderungen aus der vorherigen Teilaufgabe mit geringem Aufwand unterstützt. Skizzieren Sie die wichtigsten Funktionen und schreiben Sie die Wunschliste für die anderen Funktionen. Begründen Sie in wenigen Sätzen, wieso die von Ihnen vorgeschlagene Organisation des Programmes alle Änderungen unterstützt.

Bewertungsschema (Gesamtpunktzahl 30 pt)

- | | |
|-------------------------------|---|
| ☐ 5pt | Programm für Postkarten erweitert (2pt) und weitgehend korrekt (3pt). |
| ☐ 5pt | Das Programm insgesamt funktioniert korrekt für Postkarten und die anderen Formate. |
| ☐ 5pt | Nötige Änderungen für Postkarten-Erweiterung sinnvoll beschrieben (2pt) und dabei auf die drei Fragen eingegangen (je 1pt). |
| ☐ 5pt | Änderungen für Teilaufgabe 3 sinnvoll beschrieben (2pt) und begründet (3pt). |
| ☐ 10pt | Alternative Organisation vorgeschlagen (2pt), sinnvoll (3pt) und begründet (5pt). |