**Informatik I**

Forum: https://forum-db.informatik.uni-tuebingen.de/c/ws1617-info1
Abgabestatus/Feedback: https://handin-db.informatik.uni-tuebingen.de

Übungsblatt 5 (21.11.2016)

Abgabe: Freitag 25.11.2016, 14:00 Uhr

## Sprachebene „Die Macht der Abstraktion — Anfänger"

1. [10 Punkte] (*Abgabe:* `Blatt05-A1-game`)

   Wir betrachten einen Ausschnitt aus einem Computerspiel mit folgenden Komponenten:

   - Eine *Spielfigur* (`character`) ist charakterisiert durch einen *Namen*, einen ganzzahligen *Gesundheitszustand* zwischen 0 und 100 sowie eine *Position* auf einem zweidimensionalen Spielfeld.
   - Eine *Bombe* (`bomb`) zeichnet sich durch ihren *Detonationsradius* ("blast radius") sowie einen *Schadenswert* aus.
   - Im Spiel werden Figuren durch den Abwurf von *Bomben* traktiert. Eine Bombe wird an einer bestimmten Position auf dem zweidimensionalen Spielfeld abgeworfen. Ein *Bombenangriff* (`attack`) besteht also aus einer *Position* und einer *Bombe*.

   Eine Spielfigur wird von einem Bombenangriff getroffen, wenn ihr Abstand zur Bombe kleiner als der Detonationsradius ist. Wird eine Spielfigur getroffen, verringert sich ihr Gesundheitszustand (siehe unten).

   Der Schaden, den eine Bombe anrichtet, verringert sich mit zunehmender Entfernung vom Detonationsort. Für eine Spielfigur mit Distanz $d$ zu einer Bombe mit Detonationsradius $r$ ($d < r$) und Schadenswert $s$ reduziert sich der Gesundheitszustand der Spielfigur um den folgenden Wert:

   $$(1 - \frac{d}{r}) * s$$

   Eine Spielfigur, die sich nicht innerhalb des Detonationsradius befindet, wird nicht beeinträchtigt. Beachtet außerdem, dass der Gesundheitszustand einer Spielfigur nicht kleiner als 0 werden soll.

   Implemetiert den Ausschnitt des Computerspiels, indem ihr wie folgt vorgeht:

   (a) Schreibt Daten- und Recorddefinitionen für $x/y$-Positionen in der zweidimensionalen Ebene (Signatur `position`), Spielfiguren (Signatur `character`), Bomben (Signatur `bomb`) und Bombenabwürfe (Signatur `attack`).

   (b) Definiert eine Prozedur `(: euclidean-distance (position position -> real))`, die die euklidische Distanz $d$ zwischen zwei Positionen $(x_1, y_1)$ und $(x_2, y_2)$, mittels der folgenden Formel berechnet:

   $$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

   (c) Schreibt eine Prozedur `drop-bomb` mit der Signatur

   ```
   (: drop-bomb (character attack -> character))
   ```

   die die Auswirkungen eines Bombenabwurfs auf eine Spielfigur nach obiger Beschreibung berechnet.

   **Hinweis**: Benutzt bei der Berechnung des neuen Gesundheitszustands die eingebaute Prozedur `(: round (real -> integer))`, um Fliesskommazahlen in ganze Zahlen zu runden.

**Achtet im Rahmen aller Teilaufgaben darauf, eigenständige und wiederkehrende Teilprobleme in Funktionen auszulagern!**

2. [8 Punkte] (*Abgabe:* `Blatt05-A2-balance`)

   Flächen unterschiedlicher geometrischer Formen (*shapes*) über den optischen Eindruck zu vergleichen ist ein schwieriges Unterfangen. So fällt es zum Beispiel schwer zu entscheiden, ob die Fläche des unten stehenden Kreises größer, kleiner oder gleich der des benachbarten Dreiecks ist. Um die Verhältnisse von Flächeninhalten verschiedener Formen besser einschätzen zu können, sollt ihr ein Programm schreiben, das diese durch den Einsatz einer Waage visualisiert. Dabei dient der Flächeninhalt einer Form als Gewicht.

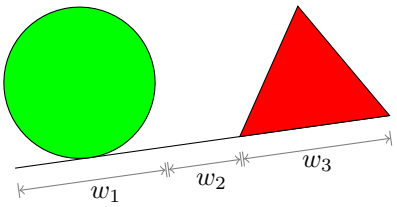


Abbildung 1: Beispiel: Vergleich des Flächeninhalt eines Kreises und eines Dreiecks mittels einer Waage (die grau gezeichneten Maße dienen nur der Illustration und müssen nicht gezeichnet werden).

   (a) Formuliert Recorddefinitionen für die drei möglichen geometrischen Formen Dreieck, Kreis und Rechteck. Diese sollen wie folgt beschrieben werden:
   - Ein *Kreis* durch seinen Radius.
   - Ein *Rechteck* durch seine Breite und Höhe.
   - Ein *Dreieck* duch seine Seitenlänge (es handelt sich um ein gleichseitiges Dreieck).

   Außerdem wird *jeder* Form zusätzlich eine Farbe zugeordnet, in der sie später gezeichnet werden soll. Hinweis: Farben können über einen `string` angegeben werden. So zeichnet z.B. `(circle 100 "solid" "green")` einen grünen Kreis[1].

   (b) Definiert eine Signatur `shape`, die alle möglichen Formen umfasst. Hinweis: Arbeitet hierbei mit einer `mixed`-Signatur.

   (c) Schreibt eine Prozedur `shape-area` mit der Signatur `(shape -> real)`, die für eine beliebige Form deren Flächeninhalt berechnet.

   (d) Um später die Waage zeichnen zu können, benötigen wir eine Prozedur

```
(: scale-length (image image real -> real))
```

   die die Länge des Balkens der Waage berechnet. Diese ergibt sich als Summe der Breiten $w_1$, und $w_3$ der übergebenen Bilder sowie des übergebenen Abstands $w_2$ (siehe Abb. 1).

   (e) Schreibt eine Prozedur

```
(: draw-shape (shape -> image))
```

   die eine beliebige Form zeichnet. Verwendet dazu die jeweiligen Zeichenfunktionen für Rechtecke, Kreise und gleichseitige Dreiecke aus dem `image2`-Teachpack (siehe Hinweise auf Seite 3).

   (f) Das Zeichnen der Waage übernimmt die Prozedur

```
(: draw-scale (shape shape -> image))
```

   Diese bekommt zwei Formen übergeben, die sie mit einem festen Abstand von $w_2 = 40px$ nebeneinander positioniert. Dabei sollen die Formen an ihrer Unterseite ausgerichtet werden. Außerdem platziert `draw-scale` die gezeichneten Formen auf der Waage.
   **Hinweis:** Der Abstand zwischen den Formen kann sehr einfach durch ein leeres Bild mit $40px$ Breite und $0px$ Höhe realisiert werden, das mittels der Prozedur `(: empty-scene (real real -> image))` erzeugt werden kann. Die Prozedur `(: beside/align (string image image image -> image))` kann mit `"bottom"` im ersten Argument genutzt werden, um die Bilder nebeneinanderzulegen.

   (g) Schließlich muss sich die Waage nach den auf sie einwirkenden Kräften ausrichten. Der Rotationswinkel $\alpha$ wird aus den Flächen $A_1$ und $A_2$ der beiden Formen mit Hilfe der folgenden Formel berechnet:

$$\alpha = 90 \times \begin{cases} 1 - \frac{A_2}{A_1} & \text{, falls } A_1 > A_2 \\ -1 + \frac{A_1}{A_2} & \text{, sonst} \end{cases}$$

   Schreibt eine Prozedur `(: rotation-angle (real real -> real))`, die für gegebene $A_1$ und $A_2$ den Rotationswinkel $\alpha$ bestimmt und fügt anschließend in `draw-scale` die Rotation um $\alpha$ hinzu.

[1] Eine Liste mit vordefinierten Farben findet ihr unter https://docs.racket-lang.org/draw/color-database___.html

**Hinweise:**

(a) Ladet das Teachpacket `image2.ss` in DrRacket, um mit Bildern arbeiten zu können (*Sprache* → *Teachpack hinzufügen*).

(b) Die Dokumentation der Zeichenfunktionen findet ihr unter

http://docs.racket-lang.org/teachpack/2htdpimage.html

(c) Verwendet insbesondere die Zeichenfunktionen `circle`, `rectangle`, `triangle`, sowie `line`, `beside/align`, `above`, `empty-scene` und `rotate`.

(d) Da die Namen `circle`, `rectangle` und `triangle` bereits im Teachpack `image2` definiert sind, müsst ihr andere Namen für eure Formen verwenden.

(e) Bitte fühlt euch durch die Aufgabenbeschreibung nicht eingeschränkt. Gerne dürft ihr auch weitere Elemente des Szenarios, wie z. B. einen schönen Unterbau für die Waage realisieren.

3. [2 Punkte] (*Abgabe:* `Blatt05-A3-dayofweek`)

Ihr wisst, dass `(one-of` $x_1$ $x_2$ ... $x_n$`)` exakt die Werte $x_1$, $x_2$, ..., $x_n$ erlaubt und daher nützlich ist, um sehr spezfische Signaturen zu formulieren. Dennoch: `one-of` ist lediglich **syntaktischer Zucker** und kann damit durch andere Konstrukte aus Racket gleichwertig ersetzt werden.

Ersetzt in der Signatur der Funktion `wochentag-index` die Vorkommen von `(one-of ...)` gleichwertig (natürlich *ohne* `one-of` zu nutzen). Hinweis: Erinnert euch an `predicate`-Signaturen.

```
(define Mo "Montag")
(define Di "Dienstag")
(define Mi "Mittwoch")
(define Do "Donnerstag")
(define Fr "Freitag")
(define Sa "Samstag")
(define So "Sonntag")

; Ermittelt die Nummer eines Wochentages innerhalb einer Woche.
; Die Nummerierung beginnt mit 0 und nimmt Montag als ersten Tag der Woche an.
(: wochentag-index ((one-of Mo Di Mi Do Fr Sa So) -> (one-of 0 1 2 3 4 5 6)))
(define wochentag-index (lambda (tag) (cond  ((string=? tag Mo) 0)
                                             ((string=? tag Di) 1)
                                             ((string=? tag Mi) 2)
                                             ((string=? tag Do) 3)
                                             ((string=? tag Fr) 4)
                                             ((string=? tag Sa) 5)
                                             ((string=? tag So) 6)
                                             )))
```