Übungen Programmieren in Clojure Serie 9

1. Datum in Clojure

Der Clojure-Reader erzeugt aus einer Eingabe der Form `#inst"2017-09-01"` ein Objekt vom Typ `java.util.Date`. In dieser Aufgabe entwickeln Sie einige Funktionen, mit denen man mit Datumsangaben arbeiten kann – auf Basis von `java.util.Date` sowie `java.util.Calendar`.

(a) Schreiben Sie eine Funktion, die ein Datum in der Zeitzone GMT aus Werten für Jahr, Monat und Tag erzeugt.

```
(defn date
  "Constructs a date in the GMT timezone from the given year month and
      day"
  [year month day]
  )
```

Folgendes sollte damit möglich sein:

```
(def d1 #inst"2017-09-03")
(def d2 (date 2017 9 3))
(= d1 d2)
; => true
```

(b) Eine Funktion `today`, die das aktuelle Datum in der Zeitzone GMT erzeugt.

(c) Eine Funktion `add` mit der man Daten manipulieren kann. Folgende beispielhafte Aufrufe sollen möglich sein:

```
(add (today) :years 1)
(add (today) :months 3)
(add (today) :days 134)
(add (today) :years 1 :months 1 :days 10)
```

(d) Entwickeln Sie die Funktionen `after?` und `before?` mit

```
(defn after?
  "Is date1 after date2?"
  [date1 date2]
  )
(defn before?
  "Is date1 before date2?"
  [date1 date2]
  )
```

(e) Entwickeln Sie eine Funktion, die ein Datum als String formatiert, wobei die Formatangaben wie in `java.text.SimpleDateFormat` vorgegeben werden können.
Beispiele:

```
(date-string (today) "dd.MM.yyyy")
; => "04.09.2013"

(date-string (today) "yyyy-MM-dd")
; => "2013-09-04"
```

2. Währungen und Geldbeträge in Clojure

Entwerfen und realisieren Sie `defrecord`s für Währungen und Geldbeträge sowie Funktionen, so dass man Folgendes tun kann:

(a) Währungen definieren gemäß ISO-4217. Der letzte Parameter gibt an, wieviele Nachkommastellen ein Betrag in der Währung haben kann. Beim Euro sind das zum Beispiel 2, beim Yen 0. Folgendes soll man mit den entwickelten Funktionen zum Beispiel machen können:

```
(def euro (currency "EUR" "Euro" "Cent" 2))
(def dollar (currency "USD" "Dollar" "Cent" 2))
(def yen (currency "JPY" "Yen" "" 0))
(def pound (currency "GBP" "Pound" "Pence" 2))
```

(b) Beträge in Währungen definieren, etwa so:

```
(def a1 (amount euro 100.00M))
(def a2 (amount yen 1000M))
```

(c) Definieren Sie Funktionen `(plus a1 a2)`, `(minus a1 a2)`, `(multiply a factor)` `(divide a divisor)`, mit denen man mit Beträgen rechnen kann. Bei `plus` und `minus` können Sie als Vorbedingung verlangen, dass die beiden Beträge dieselbe Währung haben.

(d) Schreiben Sie eine Funktion, die Beträge in eine andere Währung konvertiert, so dass man z.B. Folgendes berechnen kann:

```
(convert (amount euro 100.00M) dollar 1.12M)
```

3. Komplexe Zahlen mit `deftype` und `defrecord`

- Definieren Sie einen Datentyp für komplexe Zahlen mit `deftype`. Was passiert, wenn Sie so definierte komplexe Zahlen vergleichen?
- Definieren Sie einen Datentyp für komplexe Zahlen mit `defrecord`. Was passiert, wenn Sie so definierte komplexe Zahlen vergleichen?