Heinrich-Heine-University Düsseldorf
Computer Science Department
Software Engineering and Programming Languages
Philipp Körner
Björn Ebbinghaus

**Functional Programming – ST 2024**
**Reading Guide 02: clojure.spec**

**Timeline:** This unit should be completed by 29.04.2024.

**Note:** Please make sure you are using a recent version of Clojure again (after finishing the last unit).

## 1 Material

- Getting Clojure, chapter 15
- Material/repl2022-vertiefung/src/repl/14_spec.clj
- Clojure Guide: spec `https://clojure.org/guides/spec`
- Rich Hickey: clojure.spec `https://vimeo.com/195711510`
- Stuart Halloway: Agility & Robustness: clojure.spec `https://www.youtube.com/watch?v=VNTQ-M_uSo8`

## 2 Learning Outcomes

After completing this unit you should be able to

- discuss advantages and disadvantages of the clojure.spec library.
- read specs and give example values that meet the given spec.
- write specs yourself.

## 3 Highlights

- spec: `def`, `fdef`, `valid?`,
- spec regex: `cat`, `alt`, `+`, `*`, `?`, `&`, `spec`
- combining specs: `and`, `or`, `map-of`, `coll-of`, `keys`
- conformed/unformed values

# 4 Exercises

**Exercise 2.1 (Specs)**

a) Specify a spec for a player in a card game, who should look something like this:

```
{:cards/name "Philipp"
 :cards/hand [[3 :clubs] [:ace :spades]]}
```

Any arbitrary string is allowed as a name. A non-empty sequence of 2-tuples is associated under `:cards/hand`. The first entry is the value of the card, the second is the suit.

b) Write a function `discard`, which receives a set of players as well as a name and a playing card. If this playing card is currently in the hand of the named player, it is discarded.

c) Annotate `discard` with specs, which ensure that if called correctly, a set of players is returned by the function.

d) Is it possible to detect a problem with generated tests?

Note: you can limit the size of the generated values with
`(stest/check ‘discard {:clojure.spec.test.check/opts {:max-elements 4 :max-size 3}})`
Otherwise, testing can take a very long time..

**Exercise 2.2 (Specs II)**

In exercise 9.2, a test.check generator was used to generate users with identifiers. Write a spec for a user.

A user is represented by a map containing their first- and last name and an identifier. The identifier consists of the first two characters of the first name, the first three characters of the last name, and three digits. An example of a valid user is
`{:first-name "John", :last-name "Wayne", :id "joway142"}`.
Some invalid values are:

- `42`
- `{:first-name "John", :last-name "Wayne", :id "cowboystud69"} ; incorrect identifier`
- `{:first-name "John", :surname "Wayne", :id "joway142"} ; :surname is invalid`

# Questions

If you have any questions, please contact Philipp Körner (`p.koerner@hhu.de`).