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

Functional Programming – WT 2023 / 2024
Reading Guide 1: Overview and Tooling

Note

The learning objectives indicate what is expected of you in the exam. In principle it does not matter which material you consult to achieve the learning objective. The deadline allows you to meaningfully participate in the exercises.

Solving the exercises is voluntary. The exercises in this guide present tools and as such are elementary for subsequent course material. In addition these tools are an immense help for solving the exercises.

You are free to proceed to the next unit, as soon as you complete this one.

Timeline: This unit should be completed by 16.10.2023.

1 Material

- Learning Videos:
  - Organisation: `https://mediathek.hhu.de/watch/621b14d4-07e1-4426-9e61-98e811e9d2e3`
  - Introduction and Context: `https://mediathek.hhu.de/watch/63c2815b-5e93-48e7-89cc-8ff1872a2`
  - Tooling: `https://mediathek.hhu.de/watch/be5a7801-9e93-4262-8fa8-e60fe41db4e1`
- Leiningen (Clojure Project Management Tool — install it) `https://leiningen.org/`
- REPL: 00-syntax.clj (Note: You will find all `.clj` files in the ILIAS under Material/repl.zip in the zipped src/repl/ folder. The point is that you gain the ability to play around with these REPL sessions this week.)
- Optional:
  - Clojure Newbie Guide (overview of Clojure-related resources — give it a glance) `http://www.clojurenewbieguide.com/`
  - The Joy of Clojure, Chapter 1 - Clojure Philosophy (Do not be discouraged if you do not understand everything from this chapter immediately. The book is not especially suited for beginners, but gives a comprehensive overview of the topics we will focus on.)
  - Janet A. Carr — Mindset Shifts for Functional Programming (with Clojure) `https://blog.janetacarr.com/mindset-shifts-for-functional-programming-with-clojure/` (This short article remains relevant for the next few weeks and you will understand more once the course progresses. By the end of unit 6, you should understand the entire text.)

2 Learning Outcomes

After completing this unit you should be able to

- name the characteristics of functional programming.
- describe the downsides of object-oriented programming.
- use an editor with a REPL-plugin and paredit.
- read, write and understand fundamental Clojure syntax.

3 Exercises

Exercise 1.1 (Leiningen)
Install Leiningen (`https://codeberg.org/leiningen/leiningen`). Please use the installation instructions provided on the linked website to ensure you are not an outdated version of Clojure.

Find out how to generate a new project and familiarise yourself with the directory structure. Start a REPL-session as well.

Exercise 1.2 (Paredit)
Paredit is an important aid in programming in a LISP. A paredit-plugin exists for most editors. Make sure that your editor of choice has such a plugin and install or activate it.

Paredit helps you by keeping parentheses balanced. In order to work efficiently you should become acquainted with the keyboard shortcuts that allow for the following manipulations (copy & paste or retyping are prohibited!):

a) `(foo (bar) baz)` ⟶ `(foo (bar baz))` *(Right Slurp)*

b) `(foo (bar baz))` ⟶ `(foo bar (baz))` *(Left Barf)*

c) `(foo bar (baz))` ⟶ `(foo (bar baz))` *(Left Slurp)*

d) `(foo (bar baz))` ⟶ `(foo (bar) baz)` *(Right Barf)*

e) `(foo) (bar)` ⟶ `(foo bar)` *(Join)*

f) `(foo bar)` ⟶ `(foo) (bar)` *(Split)*

g) `foo` ⟶ `(foo)` *(Wrap)*

h) `(foo (bar baz))` ⟶ `(foo bar baz)` *(Splice)*

i) `(foo (bar baz))` ⟶ `(foo)` *(Delete / Killing)*

Exercise 1.3 (REPL-Plugin)
Many editors provide plugins to connect to open REPL sessions and execute code within their context.

Find out if such a plugin exists for your chosen editor, start a REPL session and evaluate the following code (You do not have to understand the code at this point!)

```
(def my-seq
    (lazy-cat [0 1] (map + (rest my-seq) my-seq)))

(take 10 my-seq)
```

Exercise 1.4 (4clojure Exercise Unlocks — Recommended!)
After completing this unit, you gained the knowledge to solve the following 4clojure exercises:

- elementary: 1–3, 162

You can find them in the ILIAS system in the “Material/exercises.zip” file or, alternatively, at `https://4clojure.oxal.org/`. Your task is to make the assertion succeed in the corresponding file. You may only change the definition of the solution, i.e., you should fill in the dots in the form of `(def solution ...)` or `(defn solution [...] ...)`.

Questions

If you have any questions, please contact Philipp Körner (`p.koerner@hhu.de`) or post it to the Rocket.Chat group.