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

Functional Programming – WT 2023 / 2024
Reading Guide 13: Polymorphism

1 Material

- Alternatives:
  - Clojure for the Brave and True, chapter 13
  - 06_polymorphism.clj
- Reference:
  - Clojure Reference: Multimethods `https://clojure.org/reference/multimethods`
  - Clojure Reference: Protocols `https://clojure.org/reference/protocols`
- Learning Videos:
  - Multimethoden: `https://mediathek.hhu.de/watch/2de0f875-d2aa-4062-84a6-fbcde4856edd`
  - Protokolle: `https://mediathek.hhu.de/watch/e3f13cc5-ae7b-460a-abed-094acb34e679`

Timeline: This unit should be completed by 22.01.2024.

2 Learning Outcomes

After completing this unit you should be able to

- explain how multimethods and protocols solve the expression problem.
- use multimethods and protocols.
- compare the expressiveness of multimethods and protocols.
- recall and compare the performance of multimethods and protocols.

3 Highlights

- Multimethods
- Protocols
- Functions / Macros: `defmulti`, `defmethod`, `defprotocol`, `extend-protocol`, `extend-type`, `extend`

4 Exercises

Exercise 13.1 (Marsrover)
In this task you are asked to implement a simplified version of the Marsrover-kata. For this, you are to write the control software for a robot that has landed on Mars. The robot has already scanned a rectangular area for this purpose, which unfortunately is surrounded by obstacles so that the robot cannot leave it. The requirements are as follows:

- A surface area is given, e.g.

```
[["x" "x" "x" "x"]
 ["x" " " " " "x"]
 ["x" " " " " "x"]
 ["x" "x" "x" "x"]]
```

  The data structure is a sequence of sequences that specifies the map with obstacles. The first sequence (or line) is the northernmost line of the surface, the leftmost entry corresponds to the westernmost coordinate. Entries that are the string ”x” are obstacles, strings with whitespace are free (accessible) fields.

- An initial position of the rover and its orientation (`:north`, `:south`, `:east`, `:west`) is given.
- The signature for initialization is thus: `(init! x y orientation surface)`
- Implement commands to move the rover forwards (f) and backwards (b).
- Implement command to rotate the rover by 90 degrees left and right (l, r).
- The rover is controlled via a string of commands (e.g. ”flffr”).
- The state of the rover is manipulated by `(execute! string-sequence)`.
- If there is an obstacle in the way, the rover should abort further execution of the command sequence and report the position of the obstacle.
- The function `rover-status` should return the tuple `[x y direction]` with current information about the rover.
- Your solution must allow the addition of further commands, *without the need to modify existing code*.

Note: Use multimethods.

Exercise 13.2 (Operator Overloading)
Define a new protocol `Multipliable` with a single function `mult` that takes two arguments. The function `mult` shall dispatch on the type of the first argument and “multiply” it with the second arguments. Calls and results may, for example, look like the following:

```
user=> (mult 3 4)
12
user=> (mult "foo" 4)
"foofoofoofoo"
user=> (mult '(1 2 3) 4)
(1 2 3 1 2 3 1 2 3 1 2 3)
```

Implement the protocol for at least these three types.

Questions

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