Functional Programming with Scala: Class 3

John Nestor
47 Degrees, Persist Software

uw@persist.com

Oct 31, 2016

Notes

- Homework
  - Submit only the src directory
  - Submit as .tag.gz file
- Videos on Canvas

Review

- FSet
- Present solution(s)
- Code review and discussion

Lecture

Debugging

- println (or logging) (on flag or remove after debugging)
- IDE: breakpoints, examine values, breakpoint, catch exceptions
- exceptions and stack traces
  - Handled by main or Throwable .getStackTrace, .printStackTrace
- Unit tests (test first?)
- Debugging functional code (keep most of code pure functional/immutable)
  - Repeatable
  - No side effects and less need to do mocking
  - Lack of vars, imperative loops and less recursion mean fewer errors
- Use REPL and worksheets to try out and write throw away code
- Special problem with async code: futures and actors (how to trace/single step?)

Worksheets

- How to use
- Patterns
- Type classes

Attaching Operations to ADTs

- Tree1 - OO with op in each case Object/Class
- Tree2 - OO with op in parent trait
- Tree3 - external operation with pattern matching
- Tree4 - extend objects/classes with implicit type classes

Assignment 3

Assign 3: Simplify

- Goals
  - Use an ADT: expressions
  - Task simplify expression
  - `a*3-a*2 => a*(3-2) => a*1 => a`
    - use pattern matching
    - run the unit test to make sure it works
- Look at code skeleton (Simplify) and unit test (TestSimplify)
- ??? throws exception, replace with your code
- must pass unit test

End