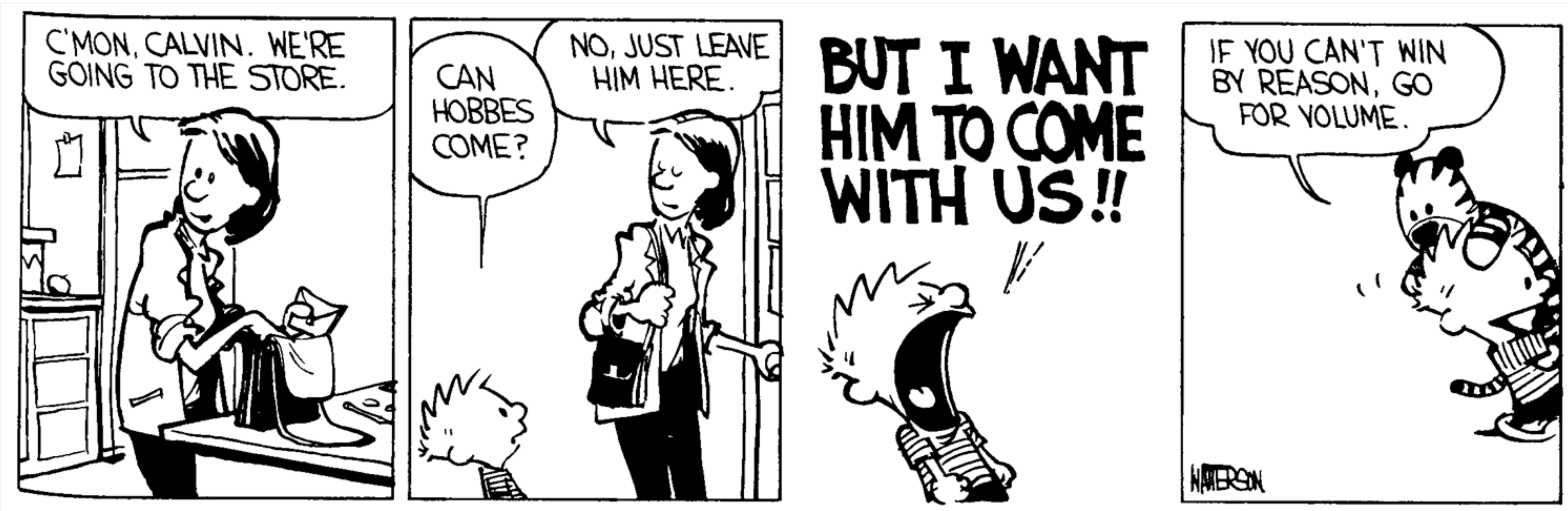
Meeting 02: Expressions



Announcements

- Syllabus Quiz due tomorrow (Friday) 6pm (with the 24-hour grace period for when “stuff happens”)
- Recitation sections are lab sections — bring your laptop!
  - This Week: Getting your development environment set up!
  - Future Weeks (in general): Getting hands-on help to *finish* assignments for the 6pm deadline — in addition to office hours during the week

Announcements

- Use Piazza for all course communication.
  - Energetically engage in discussions with your classmates to help each other on the learning activities — for your Class Participation score.
  - For course administrative things (e.g., grade issues, GitHub access issues), use private messages on Piazza to “Instructors” instead of email.
- Add notebook links to PPPL Notes (e.g., Expressions)

- Laptop section is the last row

Today

- Experience from 3155 Alumni
  - Luis Olivas (Workday)
  - Andrew Arnopoulous (Fitbod)
- Getting Your Money's Worth
- Chapter 3 Expressions
  - 3.1 Is a Program Executed or Evaluated?: imperative versus functional computation is a false dichotomy.
  - 3.2 Basic Values, Types, and Expressions: A Scala crash course.

Your Questions So Far?

- On the Syllabus?
- On Getting Your Money's Worth or why study PL?
- On Expectations and Finding Success?

Your Questions So Far?

Syllabus Policies

Requirements

Distinguishes between learning activities:

- Reading
- Quizzes and Exercises (25%)
- Lab Assignments (30%)
- Participation (5%)

And summative assessments:

- In-Class Exams (20%)
- Final Exam (20%)

Highlights

- Integrity of the Course Materials
- Collaboration Policy
- Redo Policy
- No late assignments but one drop
- No make-up exams (unless emergency or special accommodation)
- Special accommodation requests (religious observances) within first four weeks
- Regrades/redo requests within one week

Is a Program Executed or Evaluated?

The imperative versus functional schism?

Imperative

- Step by step based (Gregory)
- Mutation — Memory
  - Register
  - Stack
  - Heap

Instructions

Hardware

Java VM

VM

Executing statements for its effects on memory

Functional

- Algorithms — Mathy (Joseph)
- First-Class Functions (Gregory)
  - ↳ "Functions are Values"
- Immutability (Aaron)

Evaluating expressions = rewriting expressions until we obtain a value

Expression Rewriting

Arithmetic

$$(1+1)+(1+1) \to 2+(1+1)$$
$$\to 2+2$$
$$\to \underbrace{4}_{=\text{value}}$$

Pure Expressions and Order of Evaluation

Being effect-free or *pure* has advantages by being independent of how a machine evaluates expressions (i.e., called *referential transparency*).

left-to-right evaluation

$$
\begin{aligned}
(1+1)+(1+1) &\ \cancel{\longrightarrow \quad 2+(1+1)} \\
&\longrightarrow \quad 2+2 \\
&\longrightarrow \quad 4
\end{aligned}
$$

distributed

parallel

MapReduce

Basic Values, Types, and Expressions

Values and Types

Examples?

Expressions

An *expression* can be a literal or consist of operations that await to be evaluated. For example, here are some expected expressions:

Static Type Checking

We can check that an expression has the expected type as follows:

Static Type Checking

Static Type Checking

Type checking works by making sure all operations in all sub-expressions have the “expected type”.

Static Type Checking

Run-Time Errors

Assertions

Assertions

Unit

Since the unit value `()` itself is uninteresting and usually associated with side-effecting expression, the printer in the above simply chooses to not print unit values.

Operators and Methods

Evaluation Step

0-or-More Steps

Evaluation