Correct Programming

Table of contents

See csse2002[1] for course logistics. Lecture by Dr. Brae Webb.

> Correctness is the prime quality. If a system does not do what it is supposed to do, everything else about it […] matters little.
>
> — Bertrand Meyer, *Object-Oriented Software Construction*, §1.2 p.4, 1997

Today's outline

1. Specification — what does it mean for a program to be correct?
2. Deriving preconditions — propagating a postcondition backward through code
3. Loop invariants — reasoning about (and designing) loops
4. Pragmatic considerations — when is this worth the effort?

All content is in java-hoare-logic[2] and java-loop-invariants[3].

Applied class

See week12-tutorial-lambdas-and-streams[4] — more practice with lambdas and streams, extending java-lambdas-and-streams[5] (Week 11).

[1] courses/csse2002/csse2002.pdf
[2] courses/csse2002/java-hoare-logic.pdf
[3] courses/csse2002/java-loop-invariants.pdf
[4] courses/csse2002/week12-tutorial-lambdas-and-streams.pdf
[5] courses/csse2002/java-lambdas-and-streams.pdf

Practical

See week12-lab-generics[6] — more practice with bounded generics and wildcards, extending java-generics[7] (Week 9).

Next steps

- CSSE3100 — Reasoning About Programs
- Floyd, 1967 — *Assigning Meanings to Programs*
- Hoare, 1969 — introduced the triple $\{P\}\, S\, \{Q\}$
- Dijkstra, 1975 — weakest-precondition reasoning

[6] courses/csse2002/week12-lab-generics.pdf
[7] courses/csse2002/java-generics.pdf