Correct Programming

Table of contents

Outline 1
Applied class 2
Practical 2
Next steps 2

See csse2002[1] for course logistics.

> **Not examinable** — confirmed in 2026-05-28-course-summary-and-exam-preparation[2]. Worth knowing, but don't spend revision time here.
>
> 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

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[3] and java-loop-invariants[4].

[1] courses/csse2002/csse2002.pdf
[2] courses/csse2002/2026-05-28-course-summary-and-exam-preparation.pdf
[3] courses/csse2002/java-hoare-logic.pdf
[4] courses/csse2002/java-loop-invariants.pdf

Applied class

See week12-tutorial-lambdas-and-streams[5] — 22 tasks on lambdas and streams, extending java-lambdas-and-streams[6] (Week 11). Nothing about correctness: §1 is the three λ-function forms, the standard `java.util.function` interfaces (`Function`, `BiFunction`, `UnaryOperator`, `Predicate`, `Supplier`, `Consumer`, `BiConsumer`), method references, effectively-final capture and `andThen` composition; §2 is `filter`/`map`/`mapToInt`/`reduce`/`sum`/`forEach` and converting a loop into a pipeline.

Practical

See week12-lab-generics[7] — 6 tasks on generics, extending java-generics[8] (Week 9), built on a `Bus`/`Passenger` class hierarchy supplied as `provided.zip`. §1 rewrites a non-generic `Bus` (which stores `Object` and forces casts) into `Bus<T>`, then bounds it to `T extends Passenger` and subclasses it as `NoPetsBus<T extends Person>`; §2 adds a `BusStop` class whose methods need `? extends`, `? super`, and a generic method — the first two also asking for a `@require`/`@ensure` Javadoc comment.

Next steps

Where this material goes next. Treat the list as background reading:

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

CSSE3610/7610 is the follow-on course for concurrent and distributed algorithms (see java-loop-invariants[9]).

[5] courses/csse2002/week12-tutorial-lambdas-and-streams.pdf
[6] courses/csse2002/java-lambdas-and-streams.pdf
[7] courses/csse2002/week12-lab-generics.pdf
[8] courses/csse2002/java-generics.pdf
[9] courses/csse2002/java-loop-invariants.pdf