Object Oriented Programming I

Table of contents

See csse2002[1] for staff, assessment, and course logistics.

Today's outline

1. Objects, Classes, & Object-Oriented Programming
2. Encapsulation
3. Inheritance
4. Abstraction
5. Polymorphism (next week)

Why OOP?

A single bank account example, grown in stages, motivates the four pillars below.

Stage 1 — one global balance. A single static `bankBalance` shared by the whole "bank" — fine for one account, but there's no way to represent more than one account or customer at once.

Stage 2 — parallel arrays of accounts. Adding more accounts naively means adding a parallel array for every piece of account data (`accountNumbers`, `balances`, …) plus a manually-maintained `accountCount`, and a `findAccountIndex` helper to keep them in sync by position.

[1] courses/csse2002/csse2002.pdf

Stage 3 — parallel arrays of account *holders* too. Adding a holder's name/id alongside each account means yet more parallel arrays (`holderIds`, `holderNames`) and another manually-maintained `linkedHolderIds` array to associate accounts with holders — the bookkeeping to keep every array in sync by index grows with every new piece of data.

This is the motivation for OOP: instead of scattering related data across parallel arrays and free functions, **bundle a variable and the functions that act on it together** — e.g. instead of a free function `f(x)`, a method that belongs to `x`. This bundling is exactly what a **class** does.

The four pillars of object orientation

1. **Encapsulation** — see java-encapsulation[2]
2. **Inheritance** — see java-inheritance[3]
3. **Abstraction** — see java-abstraction[4]
4. **Polymorphism** — next week

[2] courses/csse2002/java-encapsulation.pdf
[3] courses/csse2002/java-inheritance.pdf
[4] courses/csse2002/java-abstraction.pdf