

Exercise walk-through

4.3

Composition and inheritance

University Computer Science

Composition and inheritance: is-a vs has-a

- 1 **Chain** A Member is a Person: the derived constructor calls the base first, so base state exists before any derived line runs.
- 2 **Delegate** A Library has a Catalog: five copies added, two issued, available asks the parts and counts three - swap the part, keep the whole.
- 3 **Fork** An override that copies the base body goes stale the day the base gains a line; delegate to super and subtract - the bonus survives.

Class relationship diagram

is-a inherits upward, has-a owns inward



hollow head = inheritance · filled diamond = composition · ask: can the whole exist without the part?

Two connection kinds: **inheritance** says is-a (hollow triangle to the base); **composition** says has-a (filled diamond to the part).

Exercises 2.1-2.2

2.1 A `Member(Person)` is constructed. State which `__init__` body completes first and why. [3]

2.2 Choose the connection: a `Library` and its `Catalog`. Name the relation and the arrow that draws it. [3]

Exercises 2.1-2.2

2.1 A `Member(Person)` is constructed. State which `__init__` body completes first and why. [3]

2.1

- `Person.__init__` completes first ✓.
- The derived constructor opens with the `super` call, chaining construction base-first ✓.

2.2 Choose the connection: a `Library` and its `Catalog`. Name the relation and the arrow that draws it. [3]

2.2

- Composition (has-a) ✓.
- Drawn as a filled diamond at the whole ✓.

Exercise 2.3

2.3 A subclass override duplicates the base body instead of delegating. State the one-line repair. [3]

Exercise 2.3

2.3 A subclass override duplicates the base body instead of delegating. State the one-line repair. [3]

2.3

- Delegate: call the base method and subtract – e.g. `return super().total() - 10` ✓.

Exercise 3.1

3.1 A system has: **Person**, **Member** (a person with a card and a loan count), **Library**, and **Catalog** (the list of copies a library owns).

State which pairs join by inheritance and which by composition, with a reason each. Write the **Member** constructor so base state exists before any derived line runs, naming the call that guarantees it. State what the library's **available** count reads after 5 copies are added and 2 issued, and which object answers the question. [8]

Exercise 3.1

3.1 A system has: **Person**, **Member** (a person with a card and a loan count), **Library**, and **Catalog** (the list of copies a library owns).

State which pairs join by inheritance and which by composition, with a reason each. Write the **Member** constructor so base state exists before any derived line runs, naming the call that guarantees it. State what the library's **available** count reads after 5 copies are added and 2 issued, and which object answers the question. [8]

3.1

- **Member–Person**: inheritance – a member is a person ✓.
- **Library–Catalog**: composition – the library owns its catalog ✓.
- Constructor opens with the **super** call before touching derived state ✓.
- **available** reads 3: 5 added, 2 out ✓.
- Each **Copy** answers for itself; the catalog aggregates – the whole forwards the question to its parts ✓.

Exercise 3.2

3.2 A base `Receipt.total()` sums items. `DiscountReceipt` overrides it by copying the summing code and subtracting 10. The base then gains a loyalty bonus of 2, and items total 7.

Compute what each class's `total` now returns, and the value the discount receipt should return. Name the hazard and write the one-line repair, explaining why the repair survives future base changes.

[7]

Exercise 3.2

3.2 A base `Receipt.total()` sums items. `DiscountReceipt` overrides it by copying the summing code and subtracting 10. The base then gains a loyalty bonus of 2, and items total 7.

Compute what each class's `total` now returns, and the value the discount receipt should return. Name the hazard and write the one-line repair, explaining why the repair survives future base changes. [7]

3.2

- Base returns $7 + 2 = 9$ ✓.
- The copied fork returns $7 - 10 = -3$, losing the bonus ✓.
- Should return $9 - 10 = -1$ ✓.
- The hazard is the fragile base class; repair: delegate via the base call and subtract ✓. Delegation re-uses whatever the base becomes, so future base lines flow through.