4.3 Composition and inheritance

Name: _____ Class: _____ Date: _____ **Total: 24 marks**

KEY WORDS inheritance • composition

Objective

Choose composition or inheritance with a stated reason, trace initialisation chains, and repair a fragile base.

1 Key knowledge and worked examples

■ Base first

`Member.__init__` opens with `super().__init__(name)`, so `Person.__init__` completes before any `Member` line runs ✓. Skip the call and the base attribute does not exist when the derived class needs it ✓.

■ Worked example: the stale fork

A base `total` gains a loyalty line; the subclass that copied the summing code silently loses the bonus, while the subclass that wrote `super().total() - 10` keeps it ✓. Override by delegating, never by duplicating ✓.

2 Practice

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]

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

3 Exam-style questions

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]

3.2 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]