UNIVERSITY COMPUTER SCIENCE • EXERCISE SHEET

4.2 Encapsulation

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

KEY WORDS encapsulation • invariant

Objective

Protect invariants behind an interface, and explain what a leak costs.

1 Key knowledge and worked examples

■ Validate at the gate

The invariant: a copy is out to exactly one borrower, or not out. `issue` refuses when already out; `give_back` refuses when not out – two guards, and the invariant is provable ✓. Check-first means a failed call leaves the object unchanged ✓.

■ Worked example: the truthy trap

Writing the private flag to `'y'` from outside gives a value that is truthy but not `True`: one caller shelving by truth reads out, another reading by identity reads available – two readers disagree about one object, a state no method could produce ✓.

2 Practice

2.1 A caller runs `issue()` twice on the same copy. State what the second call does and what the object reads afterwards. [3]

2.2 A method ends with `return self._loans`. State the risk and the fix. [3]

2.3 Name the invariant a `Loan` class protects when its day count must stay between 0 and 30. [3]

3 Exam-style questions

3.1 Design a `Savings` account class whose invariant is: balance is never negative and never above a fixed cap of 500.

Write the constructor check and the `deposit` method, stating every input each refuses and what the balance reads after each refusal. Explain why check-first means no rollback code is needed, and give the one-line cost argument for the wall. [8]

3.2 A teammate bypasses the methods and writes a truthy-but-not-True value into the account's private flag.

Explain how two callers can now disagree about the same object, naming the two readings. State what the repair cost scales with and what the validation cost was, and explain why exposing behaviour beats get-and-set pairs for everything. [7]