4.5 Exceptions and error handling

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

KEY WORDS exceptions • hierarchy

Objective

Choose exceptions deliberately, catch precisely, and guarantee cleanup on every path.

1 Key knowledge and worked examples

The hierarchy

`LoanError` at the base, `RenewalError` beneath it, `LimitError` beneath that: a parent handler catches the child, and the caller picks the grain ✓. `renew(Loan(20), 15)` raises with message `20+15 exceeds 30`; the loan still reads 20 ✓.

Worked example: cleanup on both paths

A lock is taken, the call inside `try` raises, the handler runs, and `finally` releases: the lock reads released ✓. On the success path `finally` runs just the same – cleanup that must happen goes in `finally` ✓.

2 Practice

2.1 `LimitError` is raised. State which of `except RenewalError` and `except LoanError` catch it, and why both can. [3]

2.2 State when the `finally` block runs, naming both paths. [3]

2.3 A lookup miss that the caller expects every run: choose a status value or an exception, and justify in one line. [3]

3 Exam-style questions

3.1 A renewal rule: the day count must never exceed 30. The hierarchy is `LoanError` with child `RenewalError` and grandchild `LimitError`; the guard checks before changing anything.

A loan of 20 days is renewed by 15. State what raises, the exact message, and the loan's day count afterwards. State which handlers would catch it and which grain a caller should pick. Explain why no rollback code is needed, and write the guarantee the `finally` block gives when a lock guards the renewal. [8]

3.2 Two designs report failure: one returns −1; one raises.

A caller forgets to check the status value and doubles the result. State what prints and what that demonstrates about status values. State what an exception does that a return cannot, naming the mechanism that carries it outward. Give the one-line rule for choosing between them. [7]