

Exercise walk-through

4.5

Exceptions and error handling

University Computer Science

Exceptions and error handling: loud, typed, cleaned up

- 1 **Hierarchy** Loan error at the base, renewal and limit beneath: a parent except catches the child, and the caller picks the grain.
- 2 **Unwind** A raise at 20 plus 15 over 30 travels out frame by frame, running each finally, until a handler matches - the loan still reads 20 after.
- 3 **Cleanup** A lock released in finally comes back False on both paths; a returned minus one is ignorable, an exception is not.

Exception flow

the raise travels down the stack until caught



handler chosen by type: a parent except also catches this child — state left as it was before the call

Raise in **renew** unwinds **issue** and **main** until a matching handler; finally runs on the way out, on every path.

Exercises 2.1-2.2

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]

Exercises 2.1-2.2

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

2.1

- Both catch it ✓.
 - Handlers match by type through the hierarchy: `LimitError` is a subclass of each ✓.
-

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

2.2

- On every path ✓.
- After the handler on the failing path; straight after the body on the success path ✓.

Exercise 2.3

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

Exercise 2.3

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

2.3

- Status value ✓.
- Expected, caller-adjacent outcome – the caller acts on it in the normal flow ✓.

Exercise 3.1

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]

Exercise 3.1

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]

3.1

- `LimitError` raises ✓.
- Message: `20+15 exceeds 30`; the loan still reads 20 ✓.
- `except RenewalError` or `except LoanError` both catch; the caller picks the grain it can act on – typically the mid-level renewal error ✓.
- No rollback: check-first changed nothing on the failing path ✓.
- `finally` guarantees the lock is released on both paths ✓.

Exercise 3.2

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]

Exercise 3.2

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]

3.2

- Prints -2 : the ignored miss spreads as nonsense ✓.
- An exception cannot be ignored: it unwinds frame by frame, running each **finally**, until a handler matches – or stops the program loudly ✓.
- Rule: status values for expected, caller-adjacent outcomes; exceptions for broken contracts ✓.