

"Lab 4: Exception Handling"

Replace printed failures with custom checked and unchecked exceptions

Lab 4 of 9

Branch: week5-exceptions

About 2 to 3 hours

Guide: LMS Guide, Step 4

Objective

Make the LMS **signal** failures with exceptions instead of printing them, so that callers (and next month, the GUI) can decide what to do. You will create a checked and an unchecked custom exception and handle both with `try/catch/finally`.

Before you start

- Lab 3 merged. Lecture 5 notes read.
- Create a new package `lms.exception`.

Git: start and submit

```
git checkout main
git pull origin main
git checkout -b week5-exceptions
```

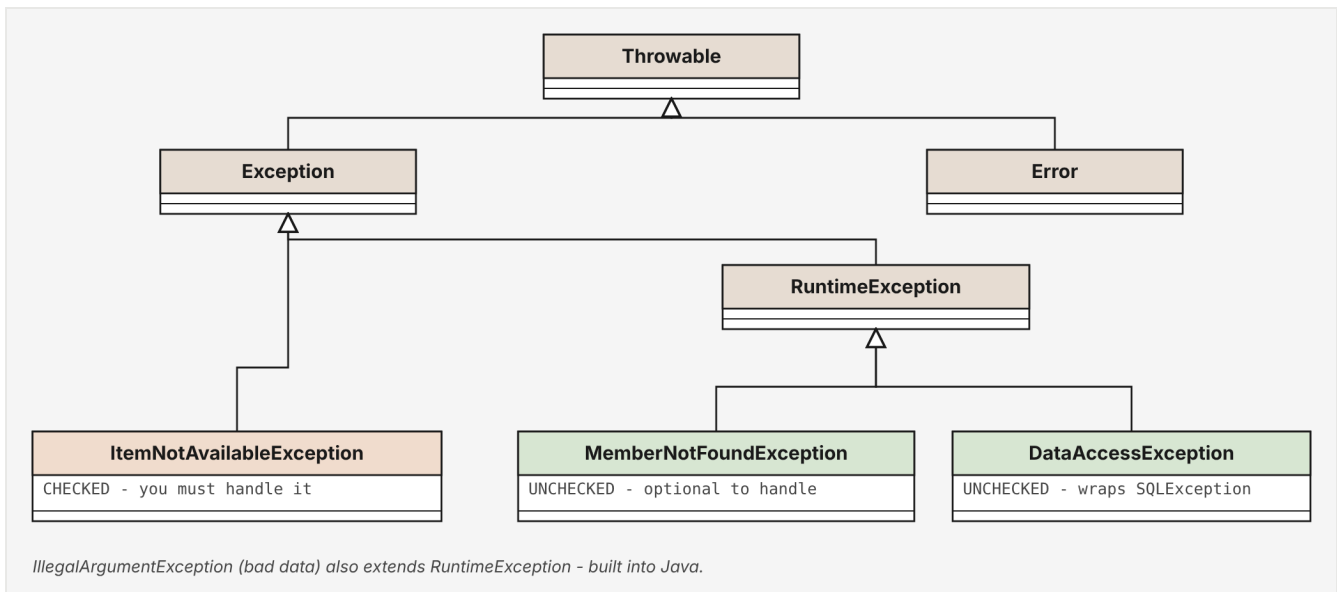
Commit small and often (see the *Git Support Guide*). When you finish:

```
git push -u origin week5-exceptions
```

Open a Pull Request titled **Week 5: Exception Handling**, fill in the template, and request a review from your lecturer.

Design first (10 minutes)

Add the two exception classes to your UML (`docs/uml-week5.png`) under `Exception` and `RuntimeException`:



Question to answer in your PR: *for each of the three failures (already on loan, unknown member, blank title) say checked or unchecked, and why.*

Tasks

Task 1: the exceptions

```

package lms.exception;
public class ItemNotAvailableException extends Exception {           // checked
    public ItemNotAvailableException(String message) { super(message); }
}
  
```

```

package lms.exception;
public class MemberNotFoundException extends RuntimeException {      // unchecked
    private final String memberId;
    public MemberNotFoundException(String memberId) {
        super("No member with id '" + memberId + "'");
        this.memberId = memberId;
    }
    public String getMemberId() { return memberId; }
}
  
```

Task 2: throw instead of print

- `Loanable`: `void loan(Member)` throws `ItemNotAvailableException`; and `void returnItem()` throws `ItemNotAvailableException`;
- `LibraryItem.loan`: if already borrowed, throw `new ItemNotAvailableException(title + " is already on loan.")`; if the member is at the limit, throw `"<name> has reached the limit of <n> items."`.
- `LibraryItem.returnItem`: if it is not on loan, throw `"<title> is not on loan."`.
- **Fix a bug while you are here**: returning an item must give the member their slot back. Add a `private Member borrower` to `LibraryItem` (set in `loan`, cleared in `returnItem`) and `Member.decrementBorrowedCount()`.

Task 3: find and borrow in `Library`

```
public Member findMember(String memberId) // throws MemberNotFoundException
public LibraryItem findItem(String itemId) throws ItemNotAvailableException // "No item with id
'X404'"
public void borrowItem(String itemId, String memberId) throws ItemNotAvailableException
public void returnItem(String itemId) throws ItemNotAvailableException
```

`lend(Loanable, Member)` now also declares `throws ItemNotAvailableException`. Fix the compile errors this causes; each one is a place where the compiler forces you to decide.

Task 4: `Main` with eight scenarios

Write a helper `attempt(Library, itemId, memberId)` that wraps `borrowItem` in:

```
try { ... }
catch (ItemNotAvailableException e) { System.out.println("Cannot borrow: " + e.getMessage()); }
catch (MemberNotFoundException e) { System.out.println("Unknown member: " + e.getMemberId()); }
finally { System.out.println("    (attempt finished: item " + itemId + ", member " + memberId + ")"); }
}
```

Then demonstrate: (1) success, (2) already on loan, (3) unknown member, (4) unknown item, (5) returning something not on loan, (6) creating `new Book("B002", " ", "Someone")` and catching `IllegalArgumentException`, (7) the student's limit of 3, (8) a return that frees a slot so that a borrow works again.

Expected output

```
--- 2. Already on loan (checked exception) ---
Cannot borrow: Clean Code is already on loan.
    (attempt finished: item B001, member M002)
--- 3. Unknown member (unchecked exception) ---
Unknown member: M999
    (attempt finished: item D001, member M999)
--- 4. Unknown item ---
Cannot borrow: No item with id 'X404'
    (attempt finished: item X404, member M001)
--- 5. Returning something not on loan ---
Cannot return: Introduction to Algorithms is not on loan.
--- 6. Bad data (IllegalArgumentException) ---
Invalid book: Title cannot be blank
--- 8. Return frees a slot ---
Clean Code returned.
Head First Java loaned to Nadeesha Perera
    (attempt finished: item B004, member M001)
```

Checkpoints

Tick each before moving on. If a box stays empty, ask for help *now*; later steps build on it.

- ☐ No `System.out.println("... already on loan")` remains in `loan()`
- ☐ The `finally` line prints in **every** attempt
- ☐ Unknown member is *not* declared in any `throws` clause, yet is caught in `Main`
- ☐ Scenario 8 proves `returnItem` frees a slot
- ☐ `docs/uml-week5.png` committed

Experiments: break it on purpose

Learning happens when you predict, run, and compare. Write your **prediction** first, then paste the real compiler message or output into your PR description.

1. Remove the `try/catch` around the first borrow call. Paste the compiler message (*unreported exception*).
2. Change `MemberNotFoundException` to `extends Exception`. List **every** place that no longer compiles.
3. Put `catch (Exception e)` *before* `catch (ItemNotAvailableException e)`. What does the compiler say?
4. Make `findMember` return `null` instead of throwing, and borrow with an unknown member. What happens and where? Which design would you rather debug at 2 a.m.?

Stretch tasks (optional, no marks lost if skipped)

- Add `FineNotPaidException` (checked) and a rule: members with a `fine0wed > 0` cannot borrow.
- Add a constructor `ItemNotAvailableException(String message, Throwable cause)` and use it when wrapping another exception.
- Print the stack trace of an uncaught exception on purpose and write down what each line means.

Common pitfalls

Symptom	Fix
<code>unreported exception ItemNotAvailableException; must be caught or declared to be thrown</code>	Add <code>throws</code> to your method or catch it, whichever fits where you can handle it
<code>exception X is never thrown in body of corresponding try statement</code>	You catch a checked exception that nothing inside the <code>try</code> can throw
Message is always <code>null</code>	Subclass constructor forgot <code>super(message)</code>
Empty <code>catch</code> block	Never swallow exceptions; at least print the message

What to submit

- ☐ PR **Week 5: Exception Handling** from branch `week5-exceptions`
- ☐ `docs/uml-week5.png`, `src/lms/exception/*`
- ☐ PR description: the checked-vs-unchecked table for the three failures; results of experiments 1 to 4
- ☐ Two review comments on a classmate's PR

Next: Week 6 is review, Week 7 is the in-class midterm. No lab. Keep your repo clean.

Marking guide

Your PR is marked on four criteria. Each is scored 4 (excellent) to 1 (needs work) and weighted as shown.

Criterion (weight)	4	3	2	1
Works (40%)	All tasks run, expected output matches	Minor differences	Partly works	Does not compile or run
OOP design (30%)	Concepts of this week used correctly and deliberately	Mostly correct	Concepts present but misused	Concepts missing
Code quality (15%)	Clear names, small methods, no dead code, comments explain <i>why</i>	Mostly clean	Messy in places	Hard to read
Git and PR (15%)	Exact branch/PR name, small clear commits, full PR description, UML in docs/	One slip	Several slips	One giant commit or direct to <code>main</code>

Check Before you open the PR, read your own diff on GitHub as if you were the reviewer. Most deductions are things you would have spotted yourself.