

"Lab 1: Classes, Objects and Encapsulation"

Build Book, Member and Library with private fields and validated setters

Lab 1 of 9

Branch: week2-encapsulation

About 2 to 3 hours

Guide: LMS Guide, Step 1

Objective

Model the first three real-world things of the Library Management System as Java classes whose data is **hidden** and **protected by validation**. You finish with a console program that adds books and members, lists them, and rejects bad input.

Before you start

- Lecture 2 notes read; Git Support Guide checklist complete.
- Your repository `lms-<index>` exists with `.gitignore` and `README.md` on `main`.
- JDK 21 works: `javac -version`.

Git: start and submit

```
git checkout main
git pull origin main
git checkout -b week2-encapsulation
```

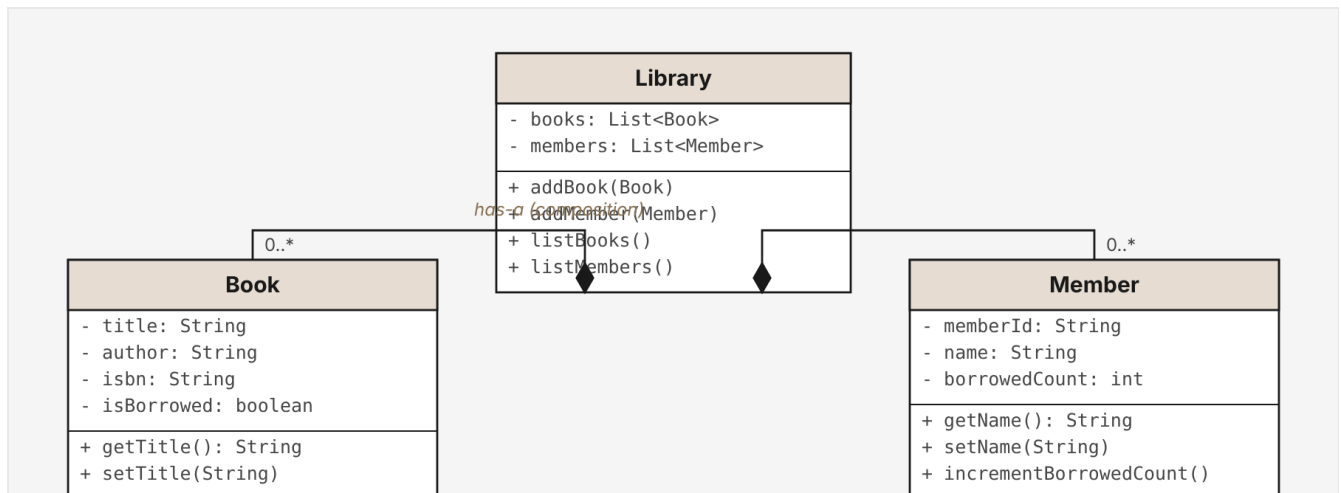
Commit small and often (see the *Git Support Guide*). When you finish:

```
git push -u origin week2-encapsulation
```

Open a Pull Request titled **Week 2: Classes, Objects & Encapsulation**, fill in the template, and request a review from your lecturer.

Design first (10 minutes)

Draw this on paper or in a tool (draw.io is fine) and save a photo or export as `docs/uml-week2.png`:



Question to answer in your PR: *which fields have no setter, and why?*

Tasks

Task 1: project structure

Create exactly this layout inside your repository:

```
src/lms/Main.java
src/lms/model/Book.java
src/lms/model/Member.java
src/lms/service/Library.java
```

Every file starts with its `package` line, for example `package lms.model;`.

Task 2: the `Book` class

- Private fields: `title`, `author`, `isbn` (all `String`), `isBorrowed` (`boolean`).
- A constructor `Book(String title, String author, String isbn)` that **calls the setters** (not direct assignment) and sets `isBorrowed = false`.
- Getters for all four fields (`isBorrowed()` for the `boolean`).
- Setters for `title`, `author`, `isbn`: each throws `IllegalArgumentException("Title cannot be blank")` (and similarly for `author` / `isbn`) if the value is `null` or blank. Use `value == null || value.isBlank()`.
- `setBorrowed(boolean)` with no validation.
- `toString()` returning, for example, `Book{title='Clean Code', author='Robert C. Martin', isbn='978-0132350884', borrowed=false}`.

Task 3: the `Member` class

- Private fields: `memberId`, `name` (`String`), `borrowedCount` (`int`).
- Constructor `Member(String memberId, String name)` that uses setters and starts `borrowedCount` at 0.
- Validated setters for `memberId` and `name` (blank is rejected).
- `setBorrowedCount(int)` rejects negative numbers with `IllegalArgumentException`.
- `incrementBorrowedCount()` that **calls** `setBorrowedCount(this.borrowedCount + 1)`.
- `toString()` like `Member{id='M001', name='Nadeesha Perera', borrowedCount=0}`.

Task 4: the `Library` class

`Library` has books and members (composition):

```
private final List<Book> books = new ArrayList<>();
private final List<Member> members = new ArrayList<>();
```

Provide `addBook`, `addMember`, `listBooks`, `listMembers` (print a heading line, then one `toString()` per line) and getters for both lists.

Task 5: the test driver

In `Main.main`: create a `Library`; add three books and two members; call both `list...` methods; then try to add `new Book("", "Nobody", "000-0000000000")` inside a `try/catch` (`IllegalArgumentException`) and print `Rejected invalid book:` followed by the message.

Task 6: compile and run

```
mkdir -p out
javac -d out $(find src -name "*.java")
java -cp out lms.Main
```

(On Windows PowerShell use `javac -d out (Get-ChildItem -Recurse -Filter *.java src | % FullName) ;` or just run from your IDE.)

Expected output

```
--- Books in the library ---
Book{title='Clean Code', author='Robert C. Martin', isbn='978-0132350884', borrowed=false}
Book{title='Effective Java', author='Joshua Bloch', isbn='978-0134685991', borrowed=false}
Book{title='The Pragmatic Programmer', author='Hunt & Thomas', isbn='978-0135957059',
borrowed=false}
--- Registered members ---
Member{id='M001', name='Nadeesha Perera', borrowedCount=0}
Member{id='M002', name='Kasun Silva', borrowedCount=0}
Rejected invalid book: Title cannot be blank
```

Your titles may differ. The **format** and the final *Rejected* line must match.

Checkpoints

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

- ☐ Every field in all three classes is `private`
- ☐ Each constructor calls its setters
- ☐ Blank title is rejected with the exact message `Title cannot be blank`
- ☐ A negative `borrowedCount` is rejected
- ☐ `Main` shows the rejection without crashing
- ☐ `docs/uml-week2.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. Change `private String title;` to `public String title;`. In `Main`, write `book.title = ""`; . *Predict*: does it compile? Does anything stop it? Now put it back to `private` and read the compiler error from the same line.
2. In the `Book` constructor, replace `setTitle(title)` with `this.title = title`. Create `new Book("", "A", "1")`. What changed, and why does that make the constructor the *weak link*?
3. Remove `this.` from `this.title = title` inside `setTitle`. Run. What is `getTitle()` now?

Stretch tasks (optional, no marks lost if skipped)

- Add `Library.findBookByIsbn(String)` that returns the `Book` or `null`. In `Main`, print whether it was found.
- Make `getBooks()` return `Collections.unmodifiableList(books)`. Try `library.getBooks().add(...)` from `Main` and describe what happens. Why is this better encapsulation?
- Add `equals` and `hashCode` to `Book` based on `isbn`.

Common pitfalls

Symptom	Fix
<code>class Book</code> is public, should be declared in a file named <code>Book.java</code>	File name must equal the public class name
<code>package lms.model</code> does not exist	Folder names must match package names; compile from the project root with <code>-d out</code>
cannot find symbol on <code>Book</code> in <code>Library</code>	Add <code>import lms.model.Book;</code>
Prints <code>null</code> for the title	Missing <code>this.</code> in a setter
<code>.class</code> files show up in <code>git status</code>	<code>.gitignore</code> must contain <code>out/</code> and <code>*.class</code>

What to submit

- ☐ PR **Week 2: Classes, Objects & Encapsulation** from branch `week2-encapsulation`
- ☐ `src/lms/...` as above, `docs/uml-week2.png`
- ☐ PR description: what you built, how to run it, answers to the UML question and to experiments 1 to 3
- ☐ Output pasted from your own run

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.