

"Lab 7: Generics, Collections and Streams"

Write Repository, use Maps and Optional, and add search and a loan report with streams

Lab 7 of 9

Branch: week10-generics-streams

About 2 to 3 hours

Guide: LMS Guide, Step 7

Objective

Replace the two plain lists in `Library` with one **generic** `Repository<T>`, then use **streams** to add a search box and a loan report to the GUI.

Before you start

- Lab 6 merged. Lecture 10 notes read.

Git: start and submit

```
git checkout main
git pull origin main
git checkout -b week10-generics-streams
```

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

```
git push -u origin week10-generics-streams
```

Open a Pull Request titled **Week 10: Generics, Collections & Streams**, fill in the template, and request a review from your lecturer.

Design first (10 minutes)

Add `Repository<T>` to your UML as a generic class with two users (docs/uml-week10.png):

```

classDiagram
    class Repository {
        +T items
        +T members
        +add(T)
        +findById(id)
        +search(Predicate)
    }
    class Library {
        +Repository<LibraryItem> items
        +Repository<Member> members
    }
    Repository <|-- Library

```

Question to answer in your PR: *what does `Repository<Member>` stop you from doing that `Repository<Object>` would allow?*

Tasks

Task 1: `Repository<T>` in `lms.service`

Backed by a `LinkedHashMap<String, T>` and a `Function<T, String> idOf`. Methods: `add(T)` (duplicate id throws `IllegalArgumentException`), `Optional<T> findById(String)`, `boolean exists(String)`, `List<T>`

`findAll()` (return a **copy**), `List<T> search(Predicate<T>)` (use a stream), `int size()`.

Task 2: use it in **Library**

```
private final Repository<LibraryItem> items = new Repository<>(LibraryItem::getId);
private final Repository<Member> members = new Repository<>(Member::getMemberId);
```

Update every method that looped over the old lists. `findMember` becomes

```
members.findById(id).orElseThrow(() -> new MemberNotFoundException(id)).
```

Task 3: three stream methods

- `search(String text)`: case-insensitive match against title, id, or (for a **Book**) author, via `items.search(...)`.
- `Map<String, Long> loansPerMember()`: filter borrowed, `Collectors.groupingBy(borrower name, TreeMap::new, Collectors.counting())`.
- `List<LibraryItem> availableItems()`: not borrowed, sorted by title.

Task 4: controller and GUI

- Controller: `search(String)` (blank means all) and `loanReport()` that returns text such as **Items currently on loan:** then **Name:** `n` lines and **Available:** `n`.
- GUI: a **Search** field and button at the top (Enter also searches) that refresh the table with `controller.search(...)`; a **Loan Report** button showing the text in a dialog.

Task 5: one query of your own

Write a stream method of your choice, such as titles of all borrowed books sorted alphabetically, and use it somewhere visible.

Expected output

After borrowing two items, **Loan Report** shows each member and the count:

The screenshot shows the 'Library Management System' GUI. On the left, there's a 'New item' form with fields for Type (set to 'Book'), ID, Title, and Detail, and an 'Add Item' button. At the top, there's a 'Search:' text box and a 'Search' button. Below the search bar is a table with columns: ID, Title, Type, Detail, and Status. The table contains four rows: B001 (Clean Code, Book, Robert C. Martin, On loan: Nadeesha Perera), B002 (Refactoring, Book, Martin Fowler, Available), D001 (Introduction to Algorithms, DVD, 120 min, On loan: Mr. Kasun Silva), and G001 (Issue 42, Available). A modal dialog box is open in the center, titled 'Items currently on loan:', showing 'Mr. Kasun Silva: 1' and 'Nadeesha Perera: 1', with 'Available: 2' and an 'OK' button. At the bottom, there's a status bar showing 'D001 borrowed by M002', a 'Member:' dropdown set to 'M002 - Mr. Kasun Silva', and buttons for 'Borrow', 'Return', and 'Loan Report'.

ID	Title	Type	Detail	Status
B001	Clean Code	Book	Robert C. Martin	On loan: Nadeesha Perera
B002	Refactoring	Book	Martin Fowler	Available
D001	Introduction to Algorithms	DVD	120 min	On loan: Mr. Kasun Silva
G001	Issue 42			Available

Searching `martin` matches by author:

ID	Title	Type	Detail	Status
B001	Clean Code	Book	Robert C. Martin	On loan: Nadeesha Perera
B002	Refactoring	Book	Martin Fowler	Available

Checkpoints

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

- ☐ No `List<LibraryItem>` or `List<Member>` field remains in `Library`
- ☐ Adding an item with an existing id gives a clear error
- ☐ Search is case-insensitive and also matches authors
- ☐ The report counts per member correctly after a return
- ☐ Your own stream query is visible in the app

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. Try `items.add(new Student("M9", "X"));`. Paste the compiler message.
2. Change `LinkedHashMap` to `HashMap`. Does the table order change? Why does that matter to users?
3. Replace `.orElseThrow(...)` with `.get()` and look up an unknown member. What happens?
4. Rewrite `availableItems()` as a plain `for` loop. Which version is easier to read? Which to change?

Stretch tasks (optional, no marks lost if skipped)

- Add `Repository.remove(String id)` and a *Delete* button.
- `Library.mostPopularTitle()` using `collect(groupingBy(...))`.
- Add a `Set<String>` of unique authors using `stream().map(...).collect(toCollection(TreeSet::new))`.

Common pitfalls

Symptom	Fix
<code>incompatible types: Object cannot be converted to ...</code>	You used the raw type <code>Repository</code> ; write <code>Repository<LibraryItem></code>
<code>local variables referenced from a lambda must be final</code>	Copy the value into a <code>final</code> variable first
Report shows the same member twice	Group by a stable key (id or name), not the object's <code>toString</code>
<code>UnsupportedOperationException</code> when modifying a result	<code>.toList()</code> returns an unmodifiable list; copy it if you need to edit

What to submit

- ☐ PR **Week 10: Generics, Collections & Streams** from branch `week10-generics-streams`
- ☐ `docs/uml-week10.png`, screenshots of search and report
- ☐ PR description: the generics answer; your own query; experiment results

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 <code>docs/</code>	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.