COMP3004 D1

Team 22

Team Members:
Mohammad Makkawi 101311993
Van Pham 101320140
Zachary Kushner 101313951
Michael Boyer 101305518

Date submitted 16 Nov 2025

Professor Dr. Christine Laurendeau

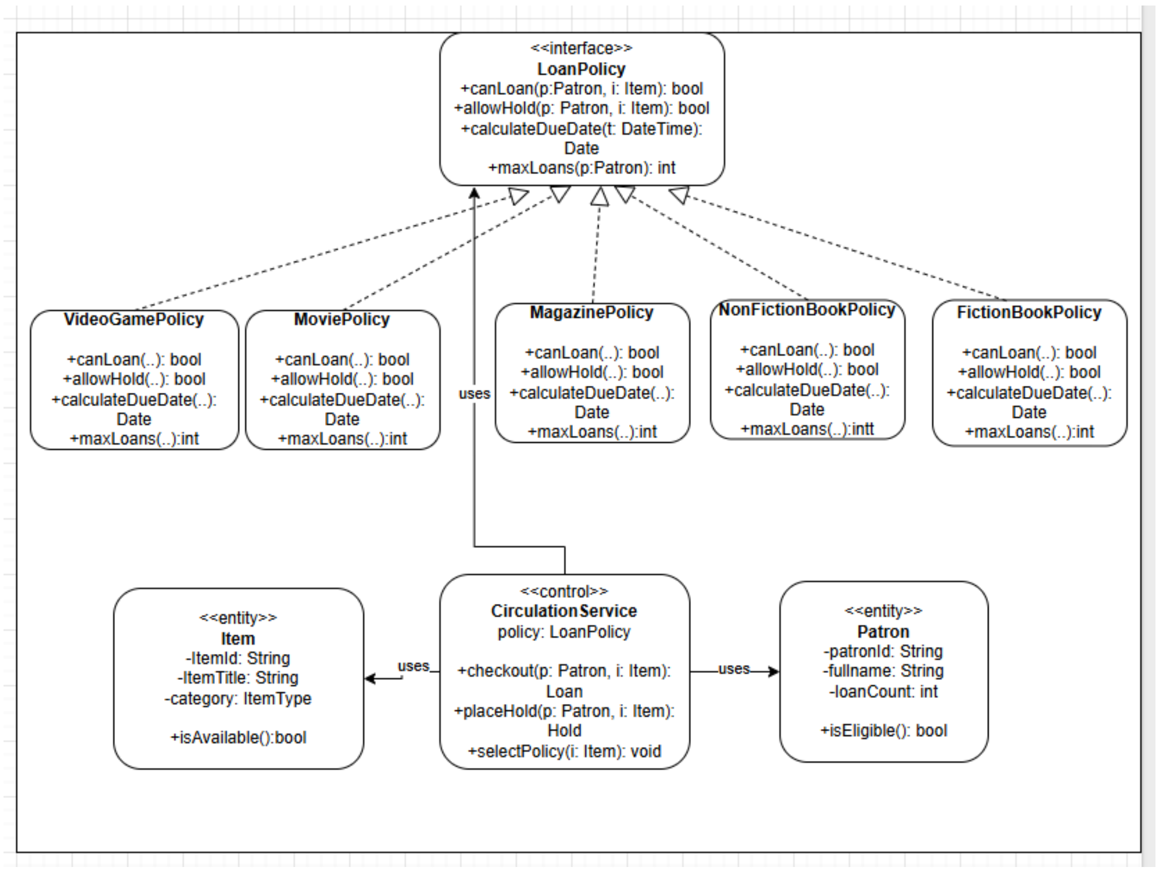

Figure 1: Strategy Pattern Used to Encapsulate Circulation Policies in HinLibs

Pattern Selection

We select the Strategy pattern to organize circulation policy behavior in HinLIBS. Strategy is a behavioral GoF pattern that summarizes algorithms and provides the client with the option to select one of the interchangeable implementations at runtime. We design CirculationService to pass over all loan-rule decisions to the LoanPolicy interface. Each item type gives its own rule set, which we defined with a concrete policy class (FictionBookPolicy, NonFictionBookPolicy, MagazinePolicy, MoviePolicy, VideoGamePolicy). This matches the UML diagram, where all five policies realize the LoanPolicy interface.

Multiplicity justification:

CirculationService uses one active LoanPolicy at runtime : 1
LoanPolicy has five concrete implementations available: 1..5
CirculationService interacts with many Patrons and Items : 1..*

Problem Fit

Without the Strategy pattern, the CirculationService would need long if/else chains to figure out loan limits, due-dates, and hold rules for every item type. This creates tight coupling, mixes workflow logic with rule logic, and makes the system harder to extend or update when policies change. By applying the Strategy pattern, all circulation rules are moved into separate LoanPolicy classes. Each policy class keeps its own rule behavior, and the controller simply delegates the decision-making to the appropriate policy for the item being borrowed. This keeps the CirculationService clean, readable, and focused on the overall workflow instead of the details of each rule.

Multiplicity justification:

An item corresponds to exactly one policy : 1..1
The controller handles many interactions overtime : 1..*

Design Principles

The design satisfies the fundamental design principles of the course:

-Low coupling: The CirculationService model depends only on the LoanPolicy interface instead of actual classes.

-High cohesion: A policy class only has the logic for each of its item classes.
- Information hiding: Rule logic is contained in the policy classes and not exposed inside the controller or entities.
-KISS: The controller does not have complex branching logic and remains simple.
Design for Change / Open-Closed Principle: When adding an item type or modifiying a rule, only one policy class is changed, and existing code stays the same

Multiplicity Justification:
One interface : many strategies 1..*
One controller : one selected strategy at runtime 1..1

UML & Architectural Context

UML class diagram is based on the original notation and uses the <<interface>> stereotype of the class diagram, realization arrows for concrete policy classes as described in UML, and "uses" dependencies. In the subsystem context, CirculationService is the Control object: in this case, Patron and Item are Entity objects. LoanPolicy and its concrete implementations are solution-domain classes of the circulation subsystem. The design fits within the application logic layer of an N tier system.

Multiplicity justification:
CirculationService : Patron= 1..*
CirculationService : Item = 1..*
LoanPolicy: concrete policies= 1..5

Scope & D1 Defaults

This design represents the vertical slice model we expect from D1:

- a maximum of three active loans per patron,

- a fixed 14-day loan period,

-FIFO holds, and no fines or suspensions.

These defaults appear at the bottom of the UML diagram and are enforced in the concrete LoanPolicy strategies. During checkout and hold placement, for example, CirculationService questions the relevant policy to ensure decisions follow the required rules.

Multiplicity justification:
A patron may have 0..3 active loans
An Item may have 0..* holds.