OCL Constraints Document

Overview

This document outlines the Object Constraint Language (OCL) constraints for the Library Management System (LMS) to enforce key business rules. These constraints ensure consistency, validity of data, and adherence to defined business rules.

---

1. Ensuring Valid Input Values

Context: User

Invariant (inv): A user’s age must be greater than 0.

```
context User
inv: self.age > 0
```

Context: Book

Invariant (inv): The total number of copies of a book must be a non-negative value.

```
context Book
inv: self.totalCopies >= 0
```

Context: BorrowingTransaction

Invariant (inv): The borrowing period (due date - borrow date) must be at least one day.

```
context BorrowingTransaction
inv: self.dueDate > self.borrowDate
```

---

2. Guaranteeing Uniqueness

Context: User

Invariant (inv): No two users can have the same email address.

context User
inv: User.allInstances()->forAll(u1, u2 | u1 <> u2 implies u1.email <> u2.email)

Context: Book

Invariant (inv): Each book must have a unique ISBN.

context Book
inv: Book.allInstances()->forAll(b1, b2 | b1 <> b2 implies b1.ISBN <> b2.ISBN)

Context: Administrator

Invariant (inv): No two administrators can have the same email address.

context Administrator
inv: Administrator.allInstances()->forAll(a1, a2 | a1 <> a2 implies a1.email <> a2.email)

Context: LibraryAnnouncements

Invariant (inv): Each announcement's title must be unique.

context LibraryAnnouncements
inv: LibraryAnnouncements.allInstances()->forAll(a1, a2 | a1 <> a2 implies a1.title <> a2.title)

---

3. Limiting Data

Context: BorrowingTransaction

Invariant (inv): A user cannot borrow more books than the maximum allowed limit (e.g., 5 books).

context User
inv: self.borrowingHistory->size() <= self.maxAllowedBooks

Context: Book

Invariant (inv): An order for borrowing books cannot exceed the available stock.

context Book
inv: self.availabilityStatus = true implies self.currentBorrower->isEmpty()

---

4. Additional Constraints

Context: Membership

Invariant (inv): A user cannot borrow books if their membership has expired.

context User
inv: self.registrationDate.addYears(1) >= today implies self.canBorrow = true

---

Context: Book Reservation

Invariant (inv): A book can only be reserved if no copies are currently available.

context Book
inv: self.availabilityStatus = false

Context: Fine Calculation

Invariant (inv): If a book is returned after the due date, a fine must be applied based on the daily rate.

context BorrowingTransaction
inv: self.returnDate > self.dueDate implies self.lateFee = (self.returnDate - self.dueDate) * self.dailyFineRate

---

Context: Administrator Role Management

Invariant (inv): Each administrator's role must belong to a predefined set (e.g., Manager, Staff).

context Administrator
inv: self.role->includes(self.role)

Context: Loan Duration Limits

Invariant (inv): Borrowing transactions must respect maximum loan duration policies (e.g., no book may be borrowed for more than 30 days).

context BorrowingTransaction
inv: self.dueDate <= self.borrowDate.addDays(30)

Context: Overlapping Transactions

Invariant (inv): A user should not have overlapping active borrowing transactions for the same book.

context User
inv: self.borrowingHistory->select(b | b.isReturned = false)->forAll(b1, b2 | b1 <> b2 implies b1.bookId <> b2.bookId)

---

5. Preconditions and Postconditions

Context: BorrowingTransaction::borrowBook()

Precondition (pre): The user must have an active membership and the book must be available.

pre: self.user.canBorrow = true and self.book.availabilityStatus = true

Postcondition (post): The book is marked as unavailable, and the borrowing transaction is recorded.

post: self.book.availabilityStatus = false and self.borrowingHistory->includes(self)

Context: BorrowingTransaction::returnBook()

Precondition (pre): The book must be currently borrowed by the user.

pre: self.book.currentBorrower = self.user

Postcondition (post): The book is marked as available, and the borrowing transaction is updated as returned.

post: self.book.availabilityStatus = true and self.isReturned = true

---

6. Derivations (derive)

Context: User::maxAllowedBooks

Derivation (derive): The maximum number of books a user can borrow is derived from their membership type.

derive: if self.membershipType = "Premium" then 10 else 5 endif

---

Context: BorrowingTransaction::lateFee

Derivation (derive): The late fee is derived based on the number of days overdue and the daily fine rate.

derive: if self.returnDate > self.dueDate then (self.returnDate - self.dueDate) * self.dailyFineRate else 0 endif

---