Library

**static** **final** **int** ***LOAN\_LIMIT*** = 2;

Sets the maximum number of books a patron can borrow at one time

**static** **final** **int** ***LOAN\_PERIOD*** = 2;

Sets the number of days for which a book may be borrowed

**static** **final** **double** ***FINE\_PER\_DAY*** = 1.0;

Sets the fine per day if a book is overdue

**static** **final** **double** ***MAX\_FINES\_OWED*** = 1.0;

Sets the maximum amount of fines a patron can owe before borrowing restrictions are enforced

**static** **final** **double** ***DAMAGE\_FEE*** = 2.0;

Sets the fine if a book is damaged

IPatron addPatron(String lastName, String firstName, String email, **long** phoneNo);

**Preconditions:** lastName, firstName, email, not null or blank, phoneNo positive (long) integer

**Postconditions:** valid borrowing enabled patron returned, patron added to patron list,

IBook addBook(String author, String title, String callNumber);

**Preconditions:** author, title, callNo, not null or blank

**Postconditions:** valid available book returned, book added to catalog,

List<IPatron> getPatronList();

**Preconditions:** none

**Postconditions:** list of patrons returned, (empty if no patrons)

List<IBook> getBookList();

**Preconditions:** none

**Postconditions:** list of books returned, (empty if no books)

List<ILoan> getCurrentLoansList();

**Preconditions:** none

**Postconditions:** list of current loans returned, (empty if no loans)

IPatron getPatronById(**int** patronId);

**Preconditions:** patronId integer

**Postconditions:** returns correct patron if patronId exists, otherwise null

IBook getBookById(**int** bookId);

**Preconditions:** bookId integer

**Postconditions:** returns correct book if bookId exists, otherwise null

ILoan getCurrentLoanByBookId(**int** bookId);

**Preconditions:** bookId integer

**Postconditions:** returns current loan associated with book if bookId exists and book is ON\_LOAN, otherwise null

**boolean** patronCanBorrow(IPatron patron);

**Preconditions:** valid patron

Applies following business rules to determine if patron can borrow:

1. A patron may not borrow further books if they have overdue loans
2. A patron may not borrow if they have already borrowed the maximum number of books permissible.
3. A patron may not borrow if they owe more than a threshold value of fines payable

**Postconditions:** returns false if any business rules apply, otherwise true

**boolean** patronWillReachLoanMax(IPatron patron, **int** numberOfPendingLoans);

**Preconditions:** valid patron, numberOfPendingLoans positive int

**Postconditions:** returns true if the number of loans a patron currently holds plus the numberOfPendingLoans adds up to the maximum permissible number of loans

ILoan issueLoan(IBook book, IPatron patron);

**Preconditions:** valid book, valid patron, book is AVAILABLE, and patron CAN\_BORROW

**Postconditions:**

Throws a RuntimeError if preconditions not met

Otherwise returns a valid PENDING loan

**void** commitLoan(ILoan loan);

**Preconditions:** valid PENDING loan

Calls loan.commit

**Postconditions:**

Throws a RuntimeError if preconditions not met

Otherwise

Loan added to full loan list

Loan added to current loan list

\*Loan added to member borrowing record

\*Book state is ON\_LOAN

\*Loan state is current

**void** dischargeLoan(ILoan currentLoan, **boolean** isDamaged);

**Preconditions:** valid CURRENT or OVERDUE loan

Calculates any overdue fines and if any are due. incurs a fine to associated patron

If book is damaged incurs a fine to associated patron, adds book to damaged book list

Calls loan.discharge

Removes loan from current loan list

Resets patron borrowing privileges as appropriate

**Postconditions:**

Throws a RuntimeError if preconditions not met

Otherwise

Patron is charged any fines incurred

Loan removed from current loan list

If damaged, book is on damaged list

\*Patrons borrowing privileges updated as appropriate

\*Loan state is DISCHARGED

\*If not damaged Book state is AVAILABLE

\*If damaged Book state is DAMAGED

**void** checkCurrentLoansOverDue();

**Preconditions:** none

Gets the current date from the Calendar singleton

For each loan in currentLoanList

Checks if loan is now overdue

If overdue

Resets the associated patron’s borrowing privileges

**Postconditions:**

\*All loans state is updated as appropriate

\*Any members with overdue loans borrowing privileges have been updated.

**double** calculateOverDueFine(ILoan loan);

**Preconditions:** valid loan

Get the current date from the calendar singleton

Checks if loan is overdue

If overdue

calculates the number of days overdue

calculates the fine payable

returns the fine payable

**Postconditions:**

Throws RuntimeError if preconditions not met

Returns the fine payable (0.0 if no fine payable)

**double** payFine(IPatron patron, **double** amount);

**Preconditions:** valid patron, amount is 0 or positive double

Calculates change due by calling patron.payFine with amount as parameter

Resets members borrowing privileges as appropriate

Returns any change due

**Postconditions:**

Throws RuntimeError if preconditions not met

\*patrons finesOwing and borrowing privileges updated as appropriate

**void** repairBook(IBook currentBook);

**Preconditions:** valid book, book in DAMAGED state

Calls book.repair

Removes book from damaged book list

**Postconditions:**

Throws RuntimeError if preconditions not met

Book is not on damaged book list

\*Book in AVAILABLE state

Book

**int** getId();

**Preconditions:** none

**Postconditions:** returns book’s id

String getTitle();

**Preconditions:** none

**Postconditions:** returns book’s title

**boolean** isAvailable();

**Preconditions:** none

**Postconditions:** returns true if book state is AVAILABLE otherwise false

**boolean** isOnLoan();

**Preconditions:** none

**Postconditions:** returns true if book state is ON\_LOAN otherwise false

**boolean** isDamaged();

**Preconditions:** none

**Postconditions:** returns true if book state is DAMAGED otherwise false

**void** borrowFromLibrary();

**Preconditions:** book in AVAILABLE state

**Postconditions:**

Throws RuntimeError if preconditions not met

Otherwise updates book state to ON\_LOAN

**void** returnToLibrary(**boolean** isDamaged);

**Preconditions:** book in ON\_LOAN state

**Postconditions:**

Throws RuntimeError if preconditions not met

If isDamaged is true

Updates book state to DAMAGED

Otherwise

Updates book state to AVAILABLE

**void** repair();

**Preconditions:** book in DAMAGED state

**Postconditions:**

Throws RuntimeError if preconditions not met

Otherwise

Updates book state to AVAILABLE

Patron

**int** getId();

**Preconditions:** none

**Postconditions:** returns patron’s id

String getLastName();

**Preconditions:** none

**Postconditions:** returns patron’s last name

String getFirstName();

**Preconditions:** none

**Postconditions:** returns patron’s first name

List<ILoan> getLoans();

String getFirstName();

**Preconditions:** none

**Postconditions:** returns list of patron’s current loans (empty list if no loans)

**double** getFinesPayable();

**Preconditions:** none

**Postconditions:** returns amount of fines payable (0.0 if none)

**boolean** hasOverDueLoans();

**Preconditions:** none

For each current loan

Calls loan.isOverdue

**Postconditions:** returns true if any loan is overdue, otherwise false

**void** restrictBorrowing();

**Preconditions:** none

**Postconditions:** patron state set to RESTRICTED

**void** allowBorrowing();

**Preconditions:** none

**Postconditions:** patron state set to CAN\_BORROW

**int** getNumberOfCurrentLoans();

**Preconditions:** none

**Postconditions:** returns number of current loans (0 if no loans)

**void** takeOutLoan(ILoan loan);

**Preconditions:** patron state is CAN\_BORROW, valid loan, loan does not exist in current borrowing record, loan state is CURRENT

Adds loan to current borrowing record

**Postconditions:**

Throws RuntimeError if preconditions not met

Loan is added to current borrowing record

**void** dischargeLoan(ILoan loan);

**Preconditions:** valid loan, loan exists in borrowing record, loan state is DISCHARGED

Removes loan from current borrowing record

**Postconditions:**

Throws RuntimeError if preconditions not met

Loan is removed from current borrowing record

**void** incurFine(**double** fine);

**Preconditions:** fine must be 0 or positive double

**Postconditions:** fine added to patron’s fines payable

**double** payFine(**double** paymentAmount);

**Preconditions:** paymentAmount must be 0 or positive double

If payment amount greater that fines payable

Calculates any change due

Sets fines payable to zero

Otherwise

Subtracts paymentAmount from fines payable

Set change to zero

Returns change

**Postconditions:**

fine subtracted from patron’s fines payable

any change due returned

Loan

**int** getId();

**Preconditions:** none

**Postconditions:** returns loan’s id

**boolean** checkOverDue(Date currentDate);

**Preconditions:** none

If loan state is CURRENT and currentDate is after loan’s dueDate

Updates loan’s status to OVER\_DUE

**Postconditions:**

Loan state updated as appropriate

**boolean** isOverDue();

**Preconditions:** none

**Postconditions:** returns true if loan state is OVER\_DUE, otherwise false

Date getDueDate();

**Preconditions:** none

**Postconditions:**

Returns dueDate (null if loan state is PENDING)

IPatron getPatron();

**Preconditions:** none

**Postconditions:** returns patron

IBook getBook();

**Preconditions:** none

**Postconditions:** returns book

**void** commit(**int** loanId, Date dueDate);

**Preconditions:** loan state is PENDING

Updates loan state to CURRENT

Calls patron.takeOutLoan

Calls book.borrowFromLibrary

**Postconditions:**

Throws RuntimeError if preconditions not met

Otherwise

Loan state is current

\*Loan added to patrons current borrowing record

\*Book state updated to ON\_LOAN

**void** discharge(**boolean** isDamaged);

**Preconditions:** loan state is CURRENT or OVER\_DUE

Updates loan state to DISCHARGED

Calls patron.dischargeLoan

Calls book.returnToLibrary

**Postconditions:**

Throws RuntimeError if preconditions not met

Otherwise

Loan state is DISCHARGED

\*Loan removed rfrom patrons current borrowing record

\*Book state updated to AVAILABLE or DAMAGED as appropriate