*[The Test Case ID should be unique. In addition, the name of each Test Case should reflect the intent of the test case, ideally expressed as a Boolean condition.]*

**1 - testHelperIsNull()**

Description:

This test throws an exception if the helper is null.

Pre-conditions:

helper is invalid

Post-conditions:

throw IllegalArgumentException

Data required:

ILoanHelper helper

**2 - testLoanMapDAOValues()**

Description:

Checks an instance of the helper holds the same value as helper.

Pre-conditions:

Set ILoanHelper actualHelper to helper;

Post-conditions:

actualHelper equals helper

Data required:

ILoanHelper actualHelper

helper

**3 - testGetLoanByID()**

Description:

Confirms if a loan can be located using a particular loan ID instead of showing null.

Pre-conditions:

id is valid

Post-conditions:

Value of loanMap returned

Data required:

int id

**4 - testGetLoanByBookIfBookIsNull()**

Description:

Check if a loan can be found using a book. Set book to invalid. Throws an exception if book is null.

Pre-conditions:

book is invalid

Post-conditions:

throw IllegalArgumentException

Data required:

IBook book

**5 - testGetLoanByBook()**

Description:

Check if a loan can be found using a book. Set book to valid and return the current loan associated to that book.

Pre-conditions:

book is valid

Post-conditions:

return loan

Data required:

IBook book

**6 - testFindLoansByBorrowerIfBorrowerIsNull()**

Description:

If a borrower is null then this test should throw an exception.

Pre-conditions:

borrower is invalid

Post-conditions:

throw IllegalArgumentException

Data required:

IMember borrower

**7 - testFindLoansByBorrower()**

Description:

Search for a loan using a borrower and see if this equals a borrower attached to a current loan.

Pre-conditions:

borrower is valid

Post-conditions:

borrower exists

Data required:

IMember borrower

**8 - testFindLoansByBookTitleIfTitleEmpty()**

Description:

Search for a loan using the title of a book and throw exception if no title exits.

Pre-conditions:

title is invalid

Post-conditions:

throw IllegalArgumentException

Data required:

String title

**9 - testFindLoansByBookTitle()**

Description:

Search for a loan using the title of a book.

Pre-conditions:

title is valid

Post-conditions:

title exits

Data required:

String title

**10 - testUpdateOverDueStatus()**

Description:

Update the overdue status of a loan using the current date.

Pre-conditions:

loan is valid

Post-conditions:

set overdue state using currentDate

Data required:

Date currentDate

**11 - testGetNextID()**

Description:

Check the getNextID method to confirm the starting ID is equal to 0.

Pre-conditions:

set actualID to id

Post-conditions:

actualID equals 0

Data required:

Int actualID

**12 - testCreateLoanNullBorrower()**

Description:

Tests if borrower details are invalid and book details valid. If the borrower is invalid and the book valid then the program throws an exception.

Pre-conditions:

borrower is invalid

book1 is valid

Post-conditions:

throw IllegalArgumentException

Data required:

borrower

IBook book 1

**13 - testCreateLoanNullBook()**

Description:

Tests if borrower is valid and book is invalid. If the book is invalid and the borrower valid then the program throws an exception.

Pre-conditions:

book is invalid

borrower1 is valid

Post-conditions:

throw IllegalArgumentException

Data required:

book

IMember borrower1

**14 - testCommitLoan()**

Description:

Checks if the ID of a loan is equal to a specific loan ID. This ensures that the loan ID is being committed correctly.

Pre-conditions:

id is valid

Post-conditions:

id equals loan id

Data required:

int id