*[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 = null;

new LoanMapDAO(helper);

Post-conditions:

assertNull(helper = null);

Data required:

Helper set to null

**2 - testLoanMapDAOValues()**

Description:

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

Pre-conditions:

ILoanHelper actualHelper = helper;

Post-conditions:

assertEquals(helper, actualHelper);

Data required:

IBook book

IMember borrower

Date borrowDate

Date dueDate

**3 - testGetLoanByID()**

Description:

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

Pre-conditions:

int id = 10;

loanMapDAO.getLoanByID(id);

Post-conditions:

assertNotNull(loanMap.containsKey(Integer.valueOf(id)));

Data required:

An integer value should be given to the id parameter.

**4 - testGetLoanByBookIfBookIsNull()**

Description:

Set book to null. Throws an exception if book is null.

Pre-conditions:

book = null;

loanMapDAO.getLoanByBook(book);

Post-conditions:

assertNull(book);

Data required:

Book set to null

**5 - testGetLoanByBook()**

Description:

Checks if an instance of a book is equal to book.

Pre-conditions:

loanMapDAO.getLoanByBook(book);

IBook tempBook = loan.getBook();

Post-conditions:

assertEquals(book, tempBook);

Data required:

Create an instance variable of type IBook and compare this to book

**6 - testFindLoansByBorrowerIfBorrowerIsNull()**

Description:

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

Pre-conditions:

borrower = null;

loanMapDAO.findLoansByBorrower(borrower);

Post-conditions:

assertNull(borrower);

Data required:

Borrower set to null

**7 - testFindLoansByBorrower()**

Description:

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

Pre-conditions:

loanMapDAO.findLoansByBorrower(borrower);

Post-conditions:

assertEquals(loan.getBorrower(), borrower);

Data required:

Borrower

* First Name
* Last Name
* Contact Phone
* Email Address

**8 - testFindLoansByBookTitleIfTitleEmpty()**

Description:

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

Pre-conditions:

String title = null;

loanMapDAO.findLoansByBookTitle(title);

Post-conditions:

assertNull(book);

Data required:

Book

* Title

**9 - testFindLoansByBookTitle()**

Description:

Search for a loan using the title of a book.

Pre-conditions:

String title = book.getTitle();

String tempTitle = loan.getBook().getTitle();

Post-conditions:

assertEquals(title, tempTitle);

Data required:

Book

* Title

**10 - testUpdateOverDueStatus()**

Description:

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

Pre-conditions:

loan.commit(1);

loan.isOverDue();

boolean overDue = loan.checkOverDue(currentDate);

Post-conditions:

assertTrue(overDue);

Data required:

This case requires the current date to instantiate the overdue method.

**11 - testGetNextID()**

Description:

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

Pre-conditions:

int actualID = loan.getID();

Post-conditions:

assertEquals(0, actualID);

Data required:

An instance of ILoan should be created then reference given to the getID method.

**12 - testCreateLoanNullBorrower()**

Description:

Tests if borrower details are empty and book details contain detail. If the borrower is empty and the book contains detail then the program throws an exception.

Pre-conditions:

borrower = null;

IBook book1 = book;

loanMapDAO.createLoan(borrower, book);

Post-conditions:

assertNull(borrower);

assertNotNull(book1);

Data required:

* Borrower should contain null values
* Create an instance variable of IBook

**13 - testCreateLoanNullBook()**

Description:

Tests if borrower contains details of borrower and the book is empty or null. If the book is empty and the borrower contains detail then the program throws an exception.

Pre-conditions:

book = null;

loanMapDAO.getLoanByBook(book);

Post-conditions:

assertNull(book);

Data required:

Book should contain null values

**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:

loanMapDAO.commitLoan(loan);

int id = 1;

Post-conditions:

assertEquals(loan.getID(), id);

Data required:

An instance of ILoan

* Loan

Variable to replicate loan number

* Id = 1