*[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 - testLoanParametersNotNull()**

Description:

Checks the detail of a new loan to ensure that the parameters are not empty. If the parameters are not empty then the program will set the book, borrower, borrowDate and dueDate to the loan. The state of the book will be set to ‘PENDING’.

Pre-conditions:

book is valid

borrower is valid

borrowDate is valid to date book borrowed

dueDate is valid to 14 days after borrowDate

ELoanState is valid with PENDING state

Post-conditions:

ELoanState set to PENDING

Value of book set

Value of borrower set

Value of borrowDate set

Value of returnDate set

Data required:

IBook book;

IMember borrower;

Date borrowDate;

Date dueDate;

**2 - testLoanParametersAreNull()**

Description:

Checks the detail of a new loan to ensure that the parameters are empty. This will throw an exception to the user as the parameters are empty for a new loan to be created.

Pre-conditions:

book is invalid

borrower is invalid

borrowDate is invalid

dueDate is invalid

Post-conditions:

throw IllegalArgumentException

Data required:

IBook book;

IMember borrower;

Date borrowDate;

Date dueDate;

**3 - testLoanCurrentStateOfLoan()**

Description:

Checks the current state of a loan is PENDING before taking out a new loan. This test confirms the current loan state is PENDING.

Pre-conditions:

currentState is valid to PENDING state

Post-conditions:

Set ELoanState to CURRENT

Data required:

int loanId

**4 - testLoanParameterValues()**

Description:

This test checks the parameters of a loan. The parameters actualBook, actualBorrower, actualBorrowDate and actualDueDate should all contain values.

Pre-conditions:

book is valid

borrower is valid

borrowDate is valid to date book borrowed

dueDate is valid to 14 days after borrowDate

Post-conditions:

ELoanState set to PENDING

Value of book set

Value of borrower set

Value of borrowDate set

Value of returnDate set

Data required:

IBook book;

IMember borrower;

Date borrowDate;

Date dueDate;

**5 - testCommitWithPositiveID()**

Description:

Check the loans state, then commit the loan with a positive ID and compare the loans end state. This test should show the loan start as PENDING and end as CURRENT without throwing an exception.

Pre-conditions:

loanId is valid

actualId equals loanId

ELoanState equals PENDING

Post-conditions:

a new loanId exists

ELoanState set to CURRENT

book added to new loan

borrower added to new loan

Data required:

int loanId

**6 - testCommitWithNegativeID()**

Description:

Try to commit a loan with a negative loan ID. This will throw an exception as the loan ID must be positive.

Pre-conditions:

loanId is invalid

actualId equals loanId

ELoanState equals CURRENT

Post-conditions:

throw IllegalArgumentException

Data required:

int loanId

**7 - testCompleteState()**

Description:

Checks the current state of the loan is in COMPLETE state.

Pre-conditions:

ELoanState is valid

Post-conditions:

ELoanState set to COMPLETE

Data required:

ELoanState

**8 - testIsOverDue()**

Description:

The method returns OVERDUE state.

Pre-conditions:

ELoanState is valid

Post-conditions:

ELoanState is set to OVERDUE

Data required:

ELoanState

**9 - testCheckOverDueError()**

Description:

Throws an exception if the current state of the loan is invalid. Then checks the current state is COMPLETE.

Pre-conditions:

ELoanState is invalid

Post-conditions:

ELoanState equals CURRENT

Data required:

ELoanState

**10 - testCheckOverDue()**

Description:

Check if the loan is overdue by confirming the current state of the loan and current date is valid then check if the loan is OVERDUE.

Pre-conditions:

currentDate is a valid date

ELoanState is valid

Post-conditions:

ELoanState equals CURRENT

Data required:

ELoanState

currentDate

**11 - testGetBorrower()**

Description:

Confirms the borrower of a book so that it can be returned to the user.

Pre-conditions:

actualBorrower is valid

Post-conditions:

actualBorrower equals borrower

Data required:

IMember actualBorrower

**12 - testGetBook()**

Description:

Confirms the details of a book so that it can be returned to the user.

Pre-conditions:

actualBook is valid

Post-conditions:

actualBook equals book

Data required:

IBook actualBook

**13 - testGetID()**

Description:

Checks the current value of ID to confirm the method can return a value.

Pre-conditions:

actualID is valid

Post-conditions:

actualID equals 0

Data required:

Int actualID

**14 - testGetCurrentState()**

Description:

Goes through each state of a typical loan and checks to see if the state changes.

Pre-conditions:

ELoanState is valid

Post-conditions:

ELoanState equals PENDING

ELoanState equals CURRENT

ELoanState equals OVERDUE

ELoanState equals COMPLETE

Data required:

ELoanState

currentDate

int loanId

**15 - testToStringNotNull()**

Description:

Confirms that if details of a loan (author, title, borrowerFirstName, borrowerLastName, borrowDate and dueDate) can be returned via toString method.

Pre-conditions:

id is valid

author is valid

title is valid

borrowerFirstName is valid

borrowerLastName is valid

borrowDate is valid

dueDate is valid

Post-conditions:

assertNotNull(string);

Data required:

int id

String author

String title

String bd

String dd

String toString