Feature / Class	Test Case ID	Unit Test Case Name	Test Scenario	Input	Expected Output	Remarks
Book Class	TC-BOOK-001	testSetDueDate	Set due date on checked-out book	Book checked out for 7 days	Due date should be set correctly	Verifies correct setting of due date
Book Class	TC-BOOK-002	testSetDueDateWhenNotCheckedOut	Set due date on book not checked out	Book not checked out, attempt to set due date	Throws IllegalStateException	Ensures due date can't be set unless book is checked out
Book Class	TC-BOOK-003	testSetCheckedOut	Check out a book	Book checked out for 7 days	Book is marked as checked out, due date is set	Confirms proper checkout behavior
Book Class	TC-BOOK-004	testReturnBook	Return a checked-out book	Book checked out and returned	Book marked as not checked out, due date is cleared	Validates return process
Book Class	TC-BOOK-005	testSetCheckOutTrue	Mark book as checked out manually	setCheckout(true)	Book is marked as checked out	Test explicit state change
Book Class	TC-BOOK-006	testSetCheckOutFalse	Mark book as no checked out manually	setCheckOut(false)	Book is marked as not checked out	Test manual change to not checked out state
Patron Class	TC-PATRON-001	testAddBook	Add book to patron’s checked-out books	Patron adds Book “1984” by George Orwell	Book is added to patron´s checked-out list	Verifies book addition to patron

Patron Class	TC-PATRON-002	testRemoveBook	Remove book from patron´s checked-out boos	Book “1984” is removed after being checked out	Book is removed from patron´s checked-out list	Confirms proper removal functionality
Patron Class	TC-PATRON-003	testFineCalculation	Calculate fine for overdue books	Fine set to 5.00 for overdue book	Fine amount is set and retrieved correctly	Validates fine calculation
Library Class	TC-LIBRARY-001	testAddBook	Add a book to the library	Book: “1984” by George Orwell	Book is added to the list of available books	Verifies that the book is stored correctly
Library Class	TC-LIBRARY-002	testAddDupilicateBooks	Prevent duplicate book addition	Add “1984” twice	Only one instance should exist	Validates book uniqueness in library
Library Class	TC-LIBRARY-003	testNonExistentBook	Prevent checkout of nonexistent book	Try to check out a book not in library	Operation should fail (return false)	Ensures integrity of library content
Library Class	TC-LIBRARY-004	testCalFine	Calculate fine for overdue book	Book checked out and overdue by 3 days	Fine = $1.50	Validates fine as $0.50 per day
Library Class	TC-LIBRARY-005	testCalFineAfterReturn	Ensure fine is zero after return	Overdue book is returned	Fine before = $1.00, after = $0.00	Ensures fine resets post return
Library Class	TC-LIBRARY-006	List current books and patrons	List Current books and patrons	Add 2 books and 2 patrons	Both books and patrons are listed correctly	Confirms list reflects current state

Chapter 5: Mocks and test Fragility

1. **Differentiating Mocks from Stubs:**
    - **Test Doubles:** Khorikov categorizes test doubles into various types, including dummies, fakes, stubs, and mocks. Each serves a distinct purpose in testing.
    - **Mocks vs. Stubs:** While both are types of test doubles, mocks are used to verify interactions by asserting that certain methods are called with expected arguments, whereas stubs provide predetermined responses to method calls without enforcing behavior verification.
2. **Observable Behavior vs. Implementation Details:**
    - The chapter emphasizes the importance of focusing tests on the observable behavior of the system rather than its internal implementation. Overemphasis on implementation details can lead to brittle tests that fail with minor code changes, even if the overall behavior remains correct.
3. **Mocks and Test Fragility:**
    - Excessive use of mocks can tightly couple tests to the specific implementations of the code under test. This coupling can result in fragile tests that break upon refactoring, hindering code maintainability and evolution. citeturn0search0
4. **Classical vs. London Schools of Unit Testing:**
    - Khorikov revisits the debate between the Classical (Detroit) and London schools of unit testing. The Classical school advocates for testing with real collaborators when possible, while the London school prefers isolating units with mocks. The chapter discusses the strengths and weaknesses of each approach, guiding readers toward balanced testing strategies.