**New College Lanarkshire**

**Technical Manual**

**Simple Library**

**DANAYAL IFTIKHAR**

# Technical Manual

## Project Brief

The Simple Library wanted a system that could allow users to be able to store information about members and stock. The system had to also store details of when loans were taken out where the stock would store loan information and the date that the item should be returned.

Additional features had to include coding which allowed the user to create journals and books to store details about book titles, authors and number of volumes.

## Specification Document

### Simple Library Scenario (Specification)

The **Simple Library** requires a system to store member and stock information. The system should also store loan details where the stock stores the member loaned to and the date that the item should be returned.

The initial analysis has identified the following required Use Cases. All user input should be validated within the LibrayGUI class as indicated within each use case.

## Use Cases

### Add Member

This use case should prompt for the new members name, street, town and postcode. The postcode should be stored as text and should be validated to ensure that it is between 6 and 8 characters long. The new member should be automatically assigned an ID number, which is 1 more than the number of members in the library.

### Add Journal

This use case should prompt for the new journals title and volume number. The volume number must be greater than 0. The number of copies should be between 1 and 3 and each copy is stored as a Journal object with the library number that is 1 more than the number of stock objects (books and journals) currently in the library. The member attribute of the stock object created should be set to null.

### Add Book

This use case should prompt for the new books title, author and number of copies. The number of copies should be between 5 and 10. Each copy is stored as a Book object with a library number that is 1 more than the number of stock objects (books and journals) currently in the library. For example if 5 copies of a new book are added to the library currently holding 100 stock objects. The five new Book objects should have the ids 101 to 105. The member attribute of each stock object created should be set to null.

### List members

This use case should simply list all the members in the library on standard output (Console Window).

### Print Stock

This use case should “*print”* all the stock objects currently in the library. If a stock object is currently on loan it should also show the member name and ID to whom they are on loan to. The print should be sent to a dummy printer text file (overwrite previous text file) **and** to the Console Window to make testing easier.

### Borrow Copy

This use case should prompt for the library number and members id. The library number should used to find the Stock object. If the stock object does not exist or the stock object is already on loan (member attribute not null) an appropriate error message should be shown.

The member should be found using the member ID. If there is no such member, an appropriate error message should be shown.

If everything is OK, the stock objects member attribute should be set to the member and the return date attribute should be set to 30 days after the current date for a Book and 2 days after the current date for Journals.

There is no limit on the number of stock copies that a member can borrow and there is no need for a Member to store what stock copies they currently have on loan.

### Return Copy

This use case should prompt for the library number of the stock object being returned. If the library does not have a stock object with that ID or the ID entered is not numeric, appropriate error messages should be displayed. Otherwise the stock objects member attribute should be set to null.

The following two Use Cases are optional, but could be useful practice for your graded project

### Start System

The librayGUI constructor should read in all stock and all members from a binary file.

### Exit System

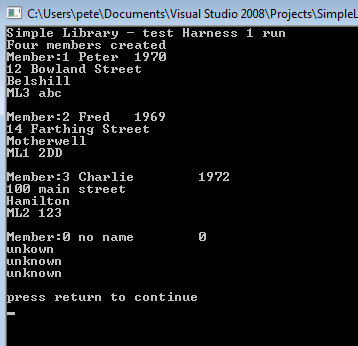
This should also write all the stock and all the members to a binary file.

Testing Documentation

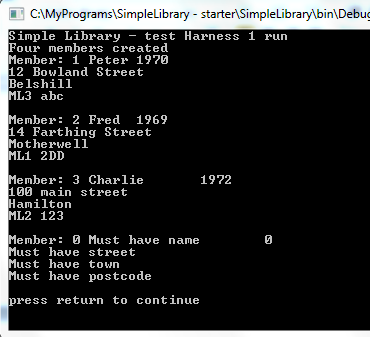
The classes should be tested as they are developed and the appropriate test harness run to ensure the classes support the expected behaviour at the end of each task development. The completed application should be tested using the given acceptance task after all of the development tasks have been completed.

## Test Harness 1.

Expected Output



Actual Output



Date Test Run 27/02/14

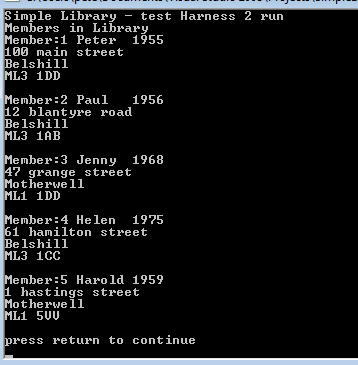
Test Passed / Failed Pass

Name of tester Danayal Iftikhar

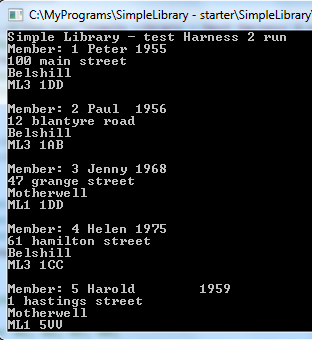
Attach a copy of the output from your program that should be similar to the one above.

## Test Harness 2.

Expected Output



Actual Output



Date Test Run 27/02/14

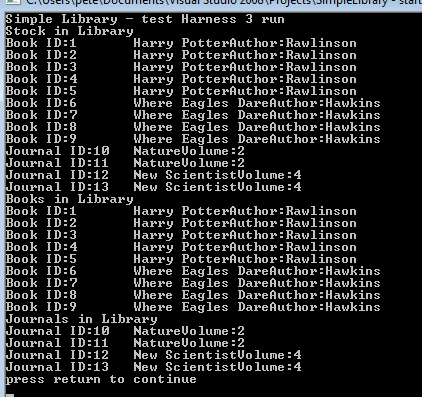
Test Passed / Failed Pass

Name of tester Danayal Iftikhar

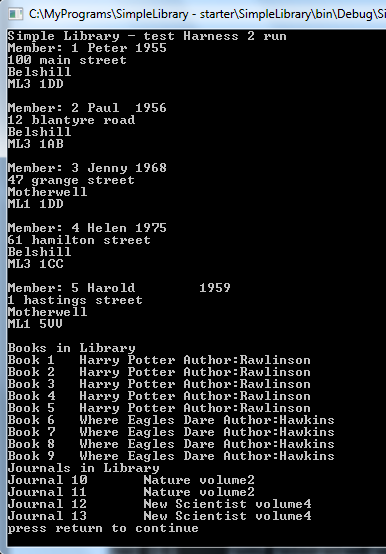
Attach a copy of the output from your program that should be similar to the one above.

## Test Harness 3.

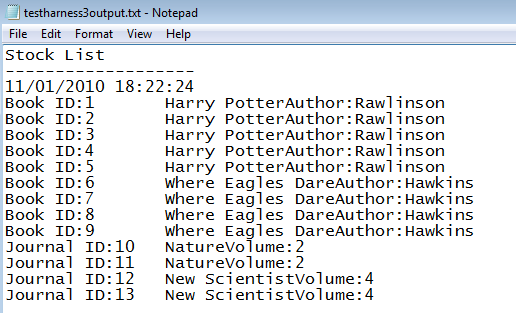
Expected Output on Console



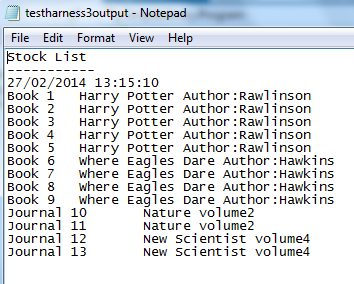
Actual output



Expected output in text file testharness3output.txt



Actual Output



Date Test Run 27/02/14

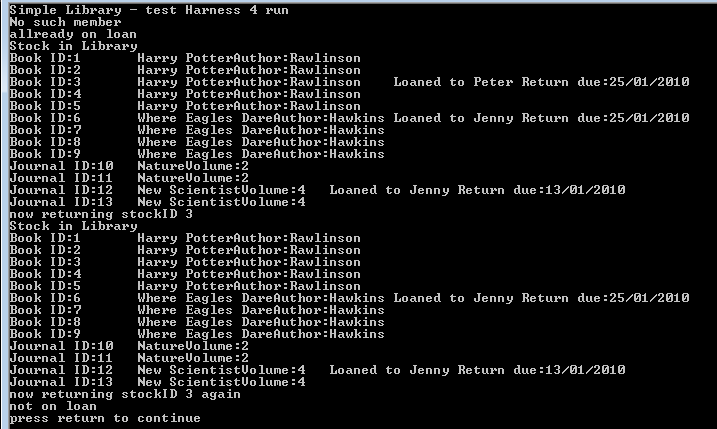
Test Passed / Failed Pass

Name of tester Danayal Iftikhar

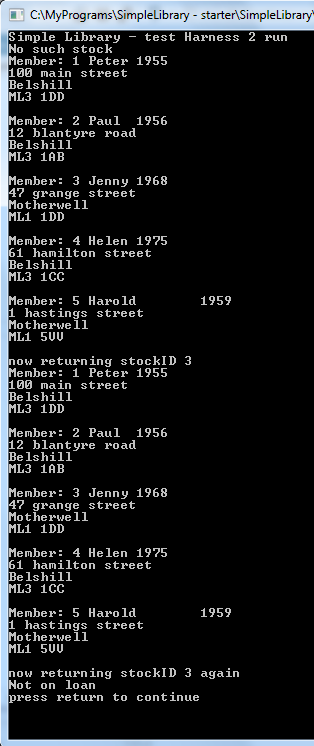
Attach a copy of the outputs from your program that should be similar to the ones above.

## Test Harness 4.

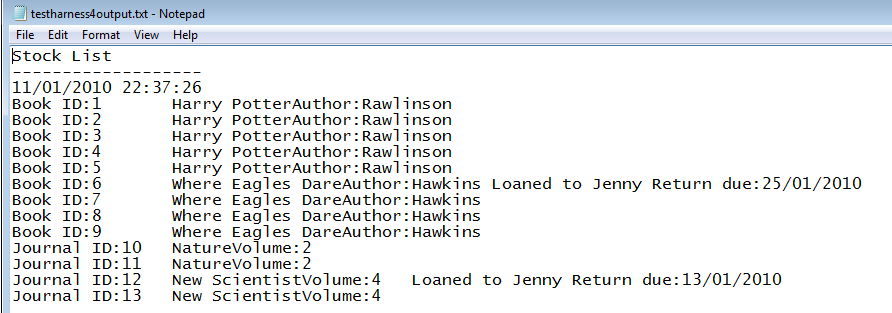
Expected Output on Console



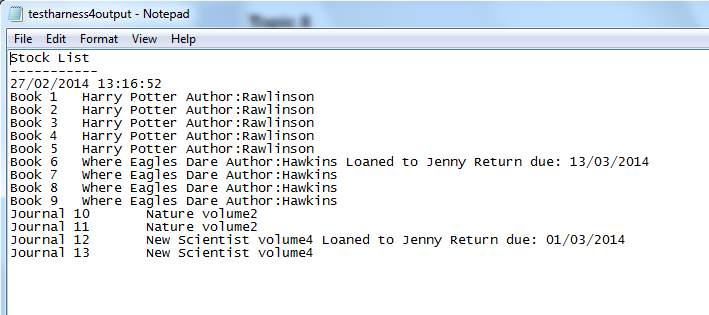
Actual Output



Expected output in text file testharness3output.txt



Actual Output



Date Test Run 27/02/14

Test Passed / Failed Pass

Name of tester Danayal Iftikhar

Attach a copy of the outputs from your program that should be similar to the ones above.

## Partial Acceptance Test (should be run as one continuous test)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Description | Expected Result | Comments | Test Pass/Fail |
| 1 | AddMember  Add a member with name “Peter” , street “Watling Street”, town “Motherwell” , and postcode “ABC12345” | Error message must have year of birth | “Must have year of birth” | Pass |
| 2 | AddMember  Add a member with street “Watling Street”, town “Motherwell” , year 1956 and postcode “ABC12345” | Error message must have name | “Must have name” | Pass |
| 3 | AddMember  Add a member with name “Peter”, town “Motherwell” , year 1956 and postcode “ABC12345” | Error message must have street | “Must have street” | Pass |
| 4 | AddMember  Add a member with name “John” , street “Watling Street”, year 1964 and postcode “ML1234” | Error message must have town | “Must have town” | Pass |
| 5 | AddMember  Add a member with name “John” , street “Watling Street”,town “wishaw” year 1964 and postcode “ML234” | Error message postcode must be between 6 and 8 characters | “Postcode must be between 6 and 8 characters” | Pass |
| 6 | AddMember  Add a member with name “John” , street “Watling Street”,town “wishaw” year 1964 and postcode “ML1234578” | Error message postcode must be between 6 and 8 characters | “Postcode must be between 6 and 8 characters” | Pass |

Acceptance test page 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Description | Expected Result | Comments | Test Pass/Fail |
| 7 | AddMember  Add a member with name “John” , street “Watling Street”,town “wishaw” year 1964 and postcode “ML1234” | Member created with next available ID | Member created with next available ID | Pass |
| 8 | AddMember  Add a member with name “Fred” , street “Watling Street”,town “wishaw” year 1964 and postcode “ML123456” | Member created with next available ID | Member created with next available ID | Pass |

Date Test Run: 27/02/14 Test Passed/Failed: Pass

Name of Tester: Danayal Iftikhar