COMPUTER SCIENCE 4: PROGRAMMING ASSIGNMENT

Hamza Bhatti (21223241)

Contents

Introduction

In this document, the development of a Library style program will be explored. The program had been developed using the Java Programming language. The program was created with the techniques that had been taught throughout the 12 week semester in mind.

This document includes Analysis of the problem the program was to solve. This document also contains Testing of the program to ensure specifications where met. Also included is a User guide, containing instructions of how to use the system.

Analysis of the Problem

This section of the document covers the Analysis of the program that was created. The objective of this section is to identify the purpose of the programme and the way I could solve the problem.

Understanding the Problem

The program created is to be used as a Library system for a collection of books. The program is aimed at Libraries that are in need of a means to enter the books that they gather into a database.

The program will allow the users to keep track of a large volume of books. This can also be very useful for individuals that are collectors.

Features of the program

As stated above, the Library system is used to enter data into a system to keep track of the books they have. For this Library system to be affective, certain features must be included. These features that had been implemented are:

Menu System: The options that should be provided to them will include adding a book, viewing their collection and quitting the program when they are finished. This is more ideal than having a command line type system. The menu avoids confusion on the user's part.

Adding Books: The user should be able to easily enter data into the system and have it stored. In turn the same book's reference number should not be saved more than once. This avoids over lapping of data entered into the Library system.

Printing Books: This will allow the user to quickly view what books the Library currently holds. A number of books as well as the details should be shown to the user.

Input Process Output

To create an efficient solution for the library, with the features that were stated above, the inputs, processes and outputs need to be identified.

Feature	Input	Process	Output
Menu system.	Choose an option from the menu.	The program will use a specific method.	The program will allow user to carry out an option.
Adding books.	Gather the book's details.	Enter details into the system.	The details are saved into an array list.
Viewing all the books.	Select option to view the books.	Books details will be read from the array list.	All books from the array list are displayed.

Developing the program

This section includes a plan of how the programme was created. The class diagrams and pseudo-code for developing the system where used to make the development of the program easier.

Class diagram

Below are the specific components of the classes that were created for the Library program.

The class diagrams show the variables included, as well as the methods that were used that carry out specific functions. The methods and variables shown here kept the final version of the program in mind.

Book Class

The Book class was the main method. This would contain all the methods that would then enable objects to be created from it.

Class: Book

Variables:
String refNum;
String bookName;
String genre;
String publisher;
int publishYear;
int stock;

Methods: public void set(String aRefNum, String aBookName, String aGenre, String aPublisher, int aPublishYear, int aStock) public Book() public Book(String aRefNum, String aBookName, String aGenre, String aPublisher, int aPublishYear, int aStock) – with set method; public void setRefNum(String aRefNum) public void setBookName(String aBookName) public void setGenre(String aGenre) public void setPublisher(String aPublisher) public void setPublishYear(int aPublishYear) public void setStock(int aStock) public String getRefNum()

public String getBookName() public String getGenre() public String getPublisher() public int getPublishYear() public int getStock() public boolean hasRefNum(String aRefNum) public void ask(String prompt) public void print() public String toString()

Library Class

This class was used to create smaller objects or other Books. The Books created here would then be stored into an array list.

Class: Library

Array Items:
ArrayList<Book> library;
ArrayList<String> keys;

Methods: public Library(); public int getSize() public int getSize() public void add(Book aLibrary) public Book find(String aRefNum) public void print(String header)

Pseudo-Code

The program created was developed using different versions. There are three classes that had been created. These included Book, Library and Stock. The pseudo code presented here include what had to be included in each version of each class.

Book class

BookV1:

```
1) Define the variables.
2) Create default constructor.
3) Identify the values that are assigned to each variable.
4) Create a print method.
```

BookV2:

```
1) Create a constructor with parameters.
2) Identify the parameters that are included in the constructor.
3) Identify which variables correspond to each parameters in the method.
```

BookV3:

```
1) Create a set method.
2) Identify the parameters that are part of the set method.
3) Identify which variables will be mutated.
4) Set the default values with the set method.
```

BookV4:

```
1) Create data members for every variable.
2) Create get methods for each variable.
```

BookV5:

```
1) Create a toString method.
2) Order the string appropriately.
```

Library Class

Array List:

```
1) Import Java array list utility.
2) User ArrayLIst to identify what data will be stored in the array.
3) Create another array that will take a string and store it in a separate index.
```

Constructor:

1) Create a constructor.

2) Use it as a container which uses methods from Book class.

Get method:

1) Create for loop.

2) Loop compares the counter with the amount of objects in Library.

3) Method saves the number of books the library contains.

Add Method:

1) Links to the ask method.

2) Save the entered data into a space in the array.

Find Method:

1) Use second array list created earlier.

2) Create variable index.

3) Entered data saved into the index.

4) If statement checks if that reference number is present.

5) Boolean will be null if reference number is not found.

Print method:

1) Use print method from the Book class.

Stock

Stock:

1) Create new object called stock.

2) Methods will come from the Books class.

3) Boolean for the menu will be written.

4) Boolean will be have a default value of false.

5) Write 3 options. Add, Print and Quit.

6) Options will take the methods from the Book class.

Testing

In this section of the document, a series of tests will be explored, identifying the efficiency of the program. The testing helps evaluate what worked well in the program and what could be added in future programs in this nature.

The tests carried out relate to the features that had been identified earlier in this document.

Test Plan

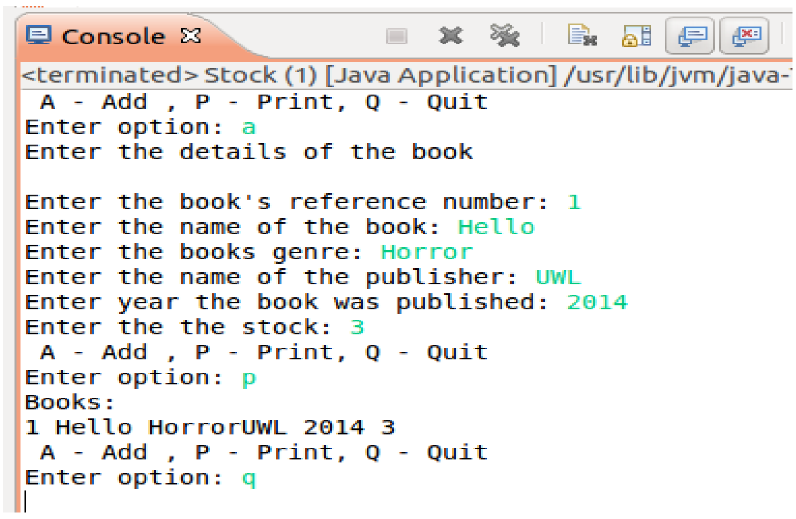
Test No.	Test Description	Testing Method	Expected Outcome	Actual Outcome	Comments
1	Checking whether all options in the menu can be selected without errors occurring.	Each option will be selected.	All options should start accordingly without errors.	No errors occurred when each option was selected.	
2	Check to see if any errors occur after the add option has been selected.	The add option will be selected. The Book details will then be entered.	No errors messages should be presented from the terminal after this action is completed.	The details where added without any errors.	
3	Check to see if wrong inputs can be entered into the system when using Add.	The add option will be used. A Wrong input will be purposefully entered.	An error will occur when an integer value is meant to be added.	The program halted when a string value was entered in place of an integer value.	In the next version of the system, error handling must be used for the ask method.
4	Check if the print option shows each Book appropriately.	The print option will be selected.	The details of the books that were added will be displayed.	The details of each book was printed, but was not clearly labelled.	
5	Check whether the correct number of books is displayed with the print option.	Three will be entered into the system. The print option will be used to see if the correct number of books is displayed.	The correct number of books should be displayed.	The correct number was displayed.	
6	Check to see if the menu rejects entries that are not assigned to current options.	Values other than a, p and q will be entered into the main menu.	The menu should reject any other values entered.	Values where rejected due to error handling.	

7	Check to see if the same reference number can be entered more than once.	The add option will be used twice. The same reference number will be used.	The system should respond with an error message and re-turn to the menu.	The second input was rejected as it had the same refer-ence number as the previous book.	

Testing Evidence

In this section, tests that where identified, along with their outcomes, are documented. Print screens are provided to show the process of the tests being carried out.

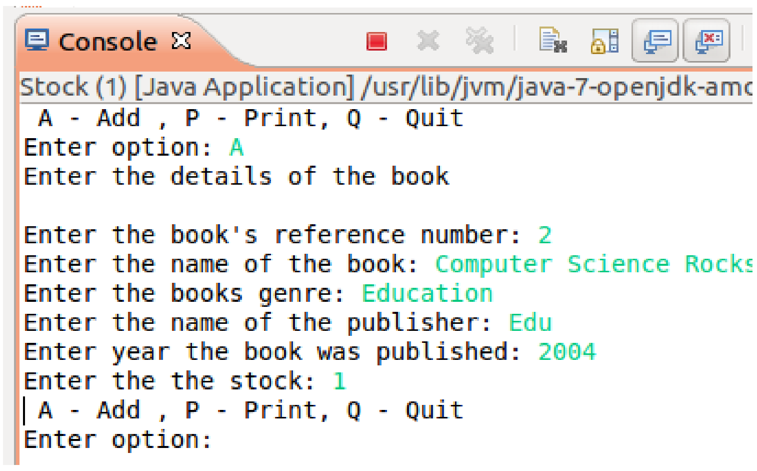
Test1: Checking whether all options in the menu can be selected without errors occurring.



```
<terminated> Stock (1) [Java Application] /usr/lib/jvm/java-
 A - Add , P - Print, Q - Quit
Enter option: a
Enter the details of the book

Enter the book's reference number: 1
Enter the name of the book: Hello
Enter the books genre: Horror
Enter the name of the publisher: UWL
Enter year the book was published: 2014
Enter the the stock: 3
 A - Add , P - Print, Q - Quit
Enter option: p
Books:
1 Hello HorrorUWL 2014 3
 A - Add , P - Print, Q - Quit
Enter option: q
```

Each option was selected one after another. No errors where encountered when the options where selected.

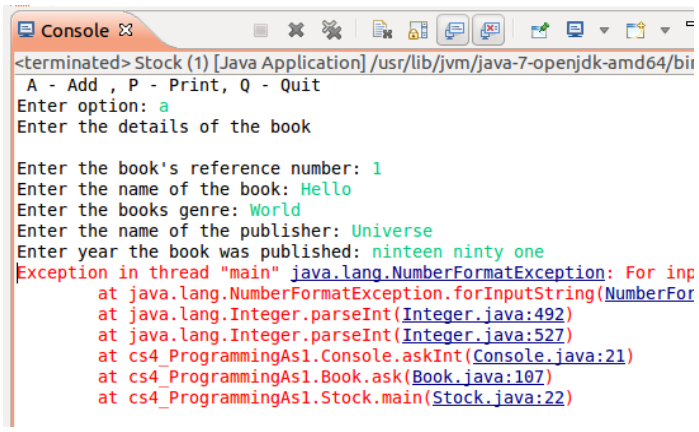
Test2: Check to see if any errors occur after the add option has been selected.



```
Stock (1) [Java Application] /usr/lib/jvm/java-7-openjdk-amc
 A - Add , P - Print, Q - Quit
Enter option: A
Enter the details of the book

Enter the book's reference number: 2
Enter the name of the book: Computer Science Rocks
Enter the books genre: Education
Enter the name of the publisher: Edu
Enter year the book was published: 2004
Enter the the stock: 1
 A - Add , P - Print, Q - Quit
Enter option:
```

After the add option was selected, the details for the books where entered. No errors occurred after the data entry occurred. The program looped back to the main menu correctly.

Test3: Check to see if wrong inputs can be entered into the system when using Add.

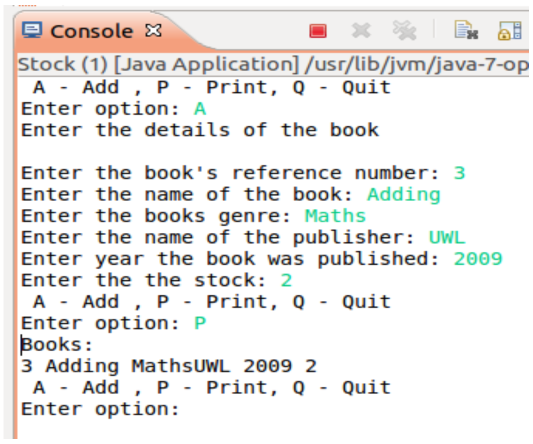


```
Console ☒
<terminated> Stock (1) [Java Application] /usr/lib/jvm/java-7-openjdk-amd64/bi
 A - Add , P - Print, Q - Quit
Enter option: a
Enter the details of the book

Enter the book's reference number: 1
Enter the name of the book: Hello
Enter the books genre: World
Enter the name of the publisher: Universe
Enter year the book was published: ninteen ninty one
Exception in thread "main" java.lang.NumberFormatException: For inp
	at java.lang.NumberFormatException.forInputString(NumberFor
	at java.lang.Integer.parseInt(Integer.java:492)
	at java.lang.Integer.parseInt(Integer.java:527)
	at cs4_ProgrammingAs1.Console.askInt(Console.java:21)
	at cs4_ProgrammingAs1.Book.ask(Book.java:107)
	at cs4_ProgrammingAs1.Stock.main(Stock.java:22)
```

The data entry fields for reference number, name of book, genre and publisher are all strings. When a string was entered into the publish year field, an error occurred. This would also occur if a string is entered in the field for stock. This was expected as those fields required integers to be entered.

One way that this can be amended is to have error handling for the ask method in the Books class. A validation check can be used to enable a wrong input to be entered but still ask the user the same question.

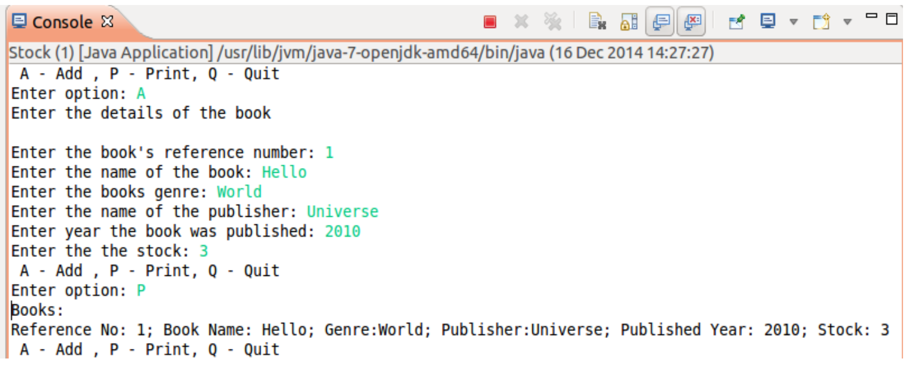
Test4: Check if the print option shows each Book appropriately.



```
Console ☒
Stock (1) [Java Application] /usr/lib/jvm/java-7-op
 A - Add , P - Print, Q - Quit
Enter option: A
Enter the details of the book

Enter the book's reference number: 3
Enter the name of the book: Adding
Enter the books genre: Maths
Enter the name of the publisher: UWL
Enter year the book was published: 2009
Enter the the stock: 2
 A - Add , P - Print, Q - Quit
Enter option: P
Books:
3 Adding MathsUWL 2009 2
 A - Add , P - Print, Q - Quit
Enter option:
```

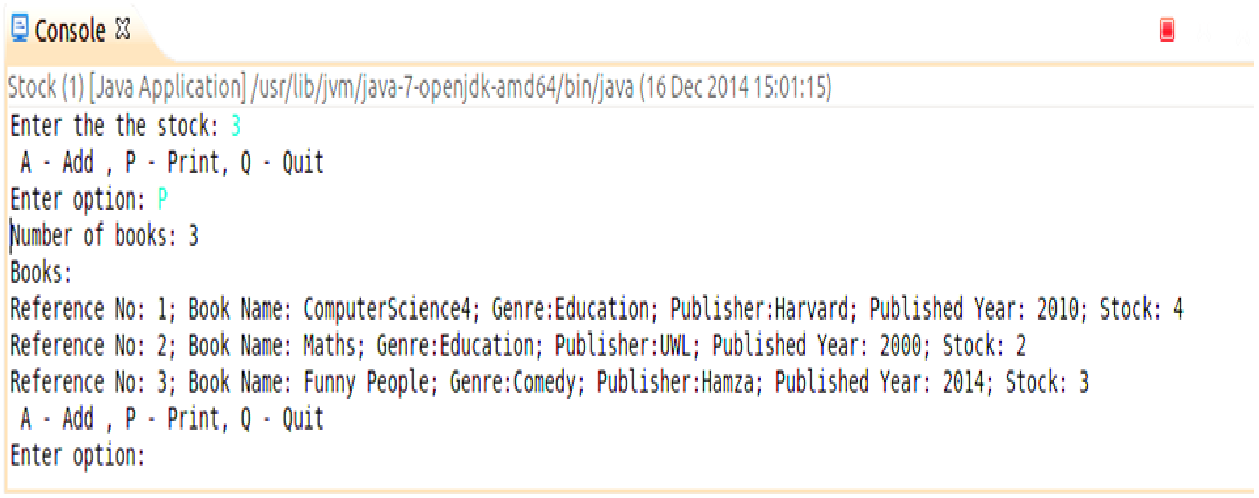
The book details were added to the system. The details stored in the array where then printed out. An issue with the print option was that the details of the book where not clearly labelled and where confusing.

The toString method was edited. More strings where added to the statement, identifying what was being shown. This resulted in a better output.



```
Console ☒
Stock (1) [Java Application] /usr/lib/jvm/java-7-openjdk-amd64/bin/java (16 Dec 2014 14:27:27)
 A - Add , P - Print, Q - Quit
Enter option: A
Enter the details of the book

Enter the book's reference number: 1
Enter the name of the book: Hello
Enter the books genre: World
Enter the name of the publisher: Universe
Enter year the book was published: 2010
Enter the the stock: 3
 A - Add , P - Print, Q - Quit
Enter option: P
Books:
Reference No: 1; Book Name: Hello; Genre:World; Publisher:Universe; Published Year: 2010; Stock: 3
 A - Add , P - Print, Q - Quit
```

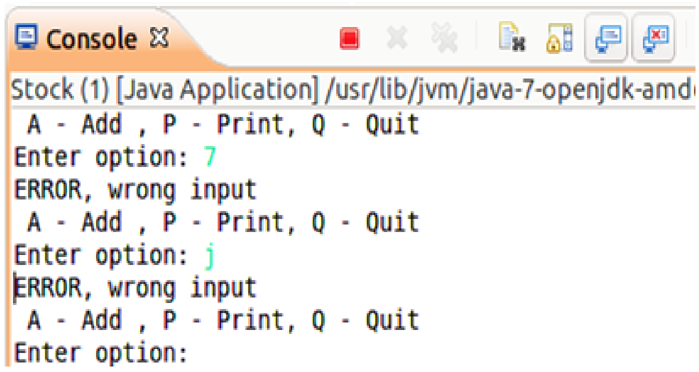
Test5: Check whether the correct number of books is displayed with the print option.



```
Stock (1) [Java Application] /usr/lib/jvm/java-7-openjdk-amd64/bin/java (16 Dec 2014 15:01:15)
Enter the the stock: 3
 A - Add , P - Print, Q - Quit
Enter option: P
Number of books: 3
Books:
Reference No: 1; Book Name: ComputerScience4; Genre:Education; Publisher:Harvard; Published Year: 2010; Stock: 4
Reference No: 2; Book Name: Maths; Genre:Education; Publisher:UWL; Published Year: 2000; Stock: 2
Reference No: 3; Book Name: Funny People; Genre:Comedy; Publisher:Hamza; Published Year: 2014; Stock: 3
 A - Add , P - Print, Q - Quit
Enter option:
```

Three different books where added to the system using the add option. The print option was then selected. getSize() method is used to show the amount of books that are stored within the array list. As three books where entered, the number 3 is displayed. All the details for the books are also displayed using the toString method.

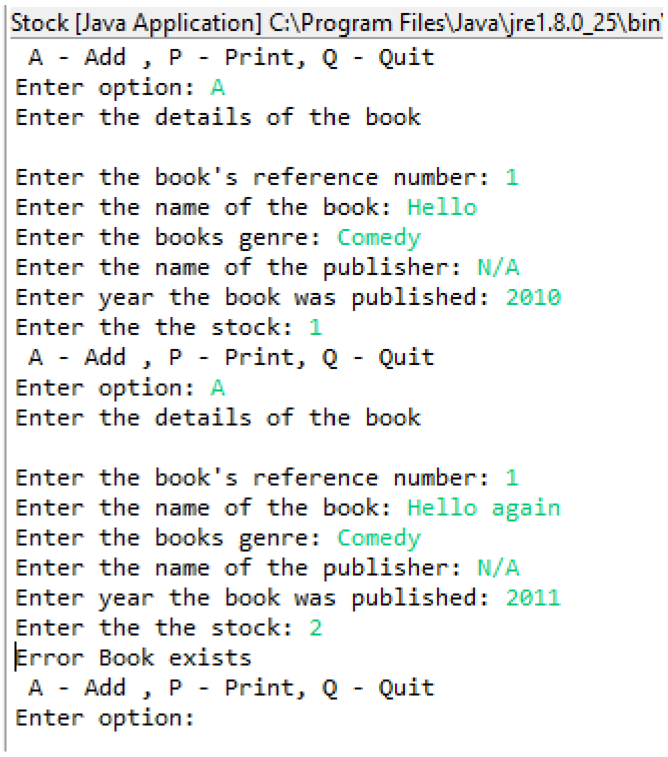
Test6: Check to see if the menu rejects entries that are not assigned to current options.



```
Stock (1) [Java Application] /usr/lib/jvm/java-7-openjdk-amd
 A - Add , P - Print, Q - Quit
Enter option: 7
ERROR, wrong input
 A - Add , P - Print, Q - Quit
Enter option: j
ERROR, wrong input
 A - Add , P - Print, Q - Quit
Enter option:
```

Data that was not part of the selection of options where entered into the main menu. A string error message displayed every time the entry was wrong. This was implemented in the Stock class.

Test7: Check to see if the same reference number can be entered more than once.



```
Stock [Java Application] C:\Program Files\Java\jre1.8.0_25\bin'
 A - Add , P - Print, Q - Quit
Enter option: A
Enter the details of the book

Enter the book's reference number: 1
Enter the name of the book: Hello
Enter the books genre: Comedy
Enter the name of the publisher: N/A
Enter year the book was published: 2010
Enter the the stock: 1
 A - Add , P - Print, Q - Quit
Enter option: A
Enter the details of the book

Enter the book's reference number: 1
Enter the name of the book: Hello again
Enter the books genre: Comedy
Enter the name of the publisher: N/A
Enter year the book was published: 2011
Enter the the stock: 2
Error Book exists
 A - Add , P - Print, Q - Quit
Enter option:
```

The first book that was added to the system had a reference number of 1. The second book was added to the system with the same reference number.

Because of the find method that was implemented, the book's reference number was already present in the array list. This caused the whole entry for the second book to be rejected. The program looped back to the main menu accordingly.

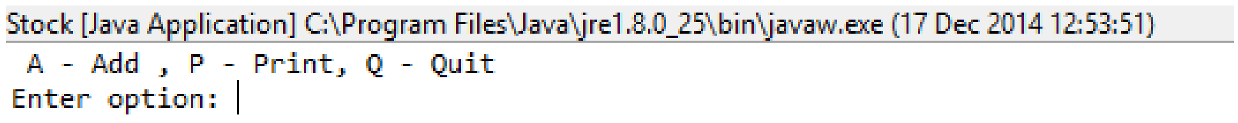
User Guide

Presented here, are instructions on how to use the Library system. The User Guide is based on the three main features that where stated earlier in this document.

The features that will be discussed in this User Guide are:

1) The inclusion of a Menu System.
2) The ability to Add Books.
3) Printing Books that have been added to the system.

The Menu

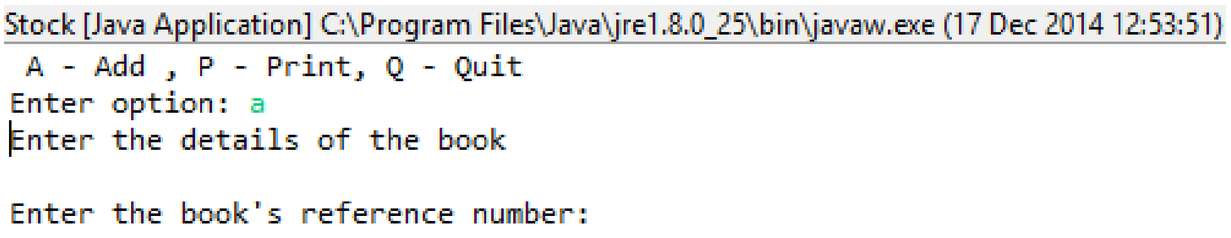


```
Stock [Java Application] C:\Program Files\Java\jre1.8.0_25\bin\javaw.exe (17 Dec 2014 12:53:51)
 A - Add , P - Print, Q - Quit
Enter option: 
```

The menu (shown above) includes three different options including: Add, Print and Quit.

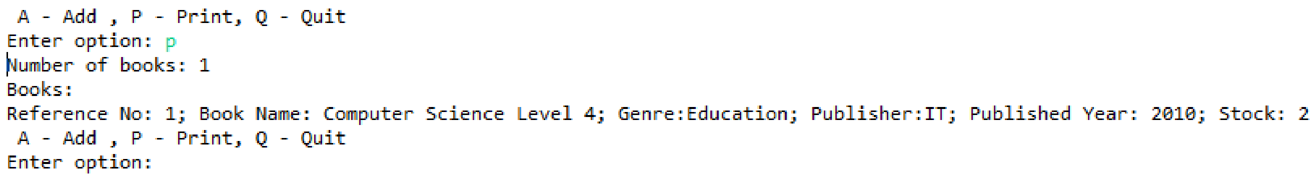
The user must enter one of the three options for a process to occur.

When the user enters a, the add option will be initiated which is shown below.



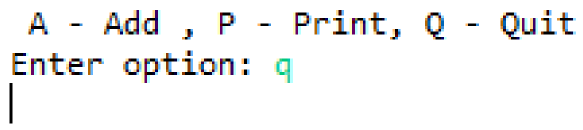
```
Stock [Java Application] C:\Program Files\Java\jre1.8.0_25\bin\javaw.exe (17 Dec 2014 12:53:51)
 A - Add , P - Print, Q - Quit
Enter option: a
Enter the details of the book

Enter the book's reference number:
```

When the user enters p, the books details that were entered will be presented to the user.



```
 A - Add , P - Print, Q - Quit
Enter option: p
Number of books: 1
Books:
Reference No: 1; Book Name: Computer Science Level 4; Genre:Education; Publisher:IT; Published Year: 2010; Stock: 2
 A - Add , P - Print, Q - Quit
Enter option:
```

When the user enters q, the program will close.



```
 A - Add , P - Print, Q - Quit
Enter option: q
```

Adding Books

The program allows the user to add books to the system.

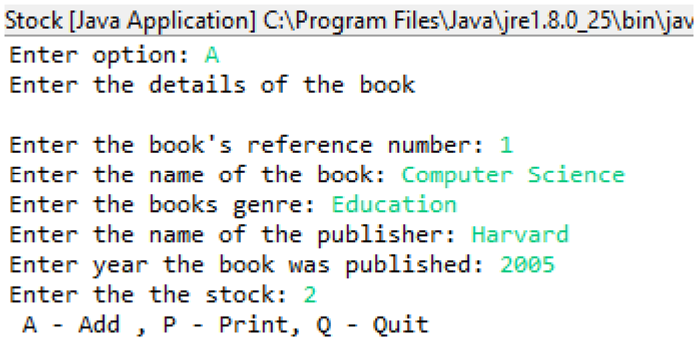
The system will ask the user for certain book details. These details include:

1) Reference number
2) Book name
3) Genre
4) Publisher
5) Year the book was published
6) Stock

Note that all the details must be filled in.

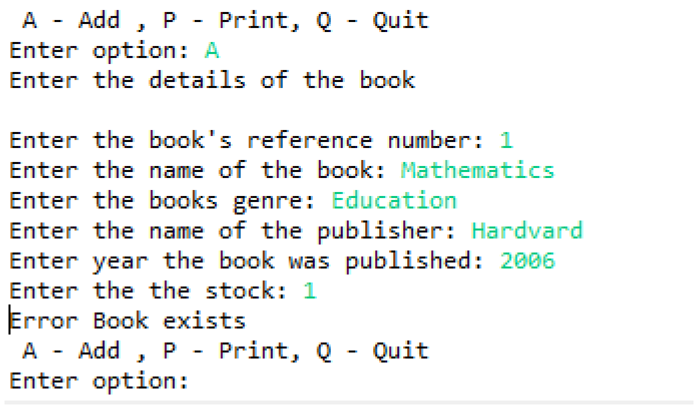
Below is an image of the data entry for a book.

Notice that the reference number for the book is 1. When the user adds another book, but adds the same reference number, the book will be rejected.



```
Stock [Java Application] C:\Program Files\Java\jre1.8.0_25\bin\jav
Enter option: A
Enter the details of the book

Enter the book's reference number: 1
Enter the name of the book: Computer Science
Enter the books genre: Education
Enter the name of the publisher: Harvard
Enter year the book was published: 2005
Enter the the stock: 2
 A - Add , P - Print, Q - Quit
```

Rejected Book

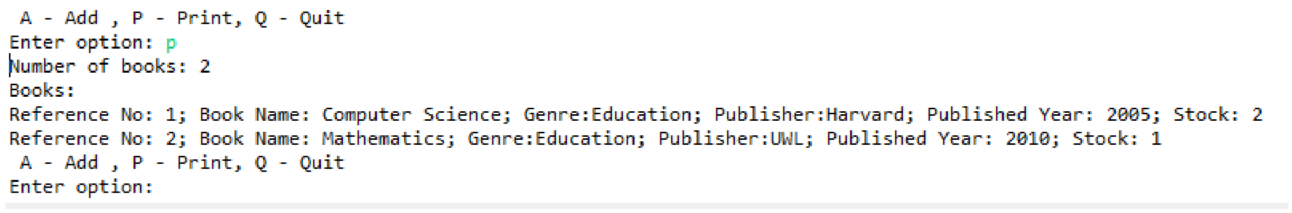


```
 A - Add , P - Print, Q - Quit
Enter option: A
Enter the details of the book

Enter the book's reference number: 1
Enter the name of the book: Mathematics
Enter the books genre: Education
Enter the name of the publisher: Hardvard
Enter year the book was published: 2006
Enter the the stock: 1
Error Book exists
 A - Add , P - Print, Q - Quit
Enter option:
```

When the book entered is added in this case, it is rejected as the previous book had the same reference number. An error message is displayed to the user, stating the book already exists.

Printing Books

After a book has been entered into the system, the book can be viewed using the print option.



```
 A - Add , P - Print, Q - Quit
Enter option: p
Number of books: 2
Books:
Reference No: 1; Book Name: Computer Science; Genre:Education; Publisher:Harvard; Published Year: 2005; Stock: 2
Reference No: 2; Book Name: Mathematics; Genre:Education; Publisher:UWL; Published Year: 2010; Stock: 1
 A - Add , P - Print, Q - Quit
Enter option:
```

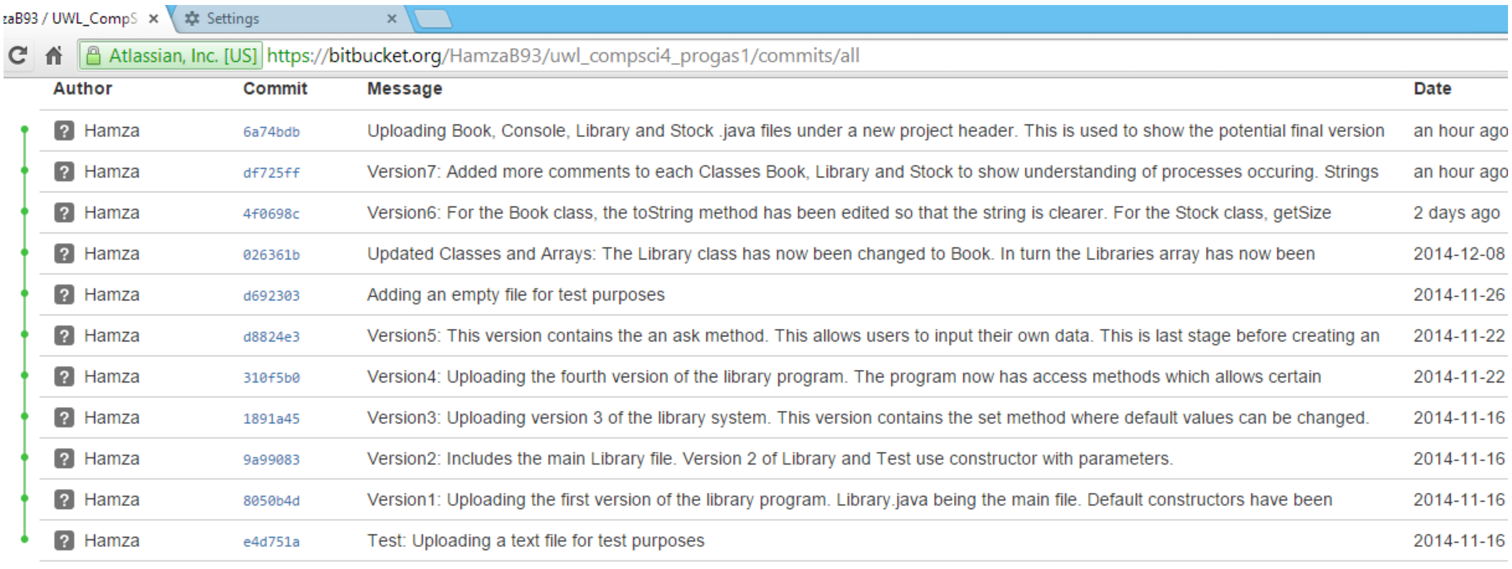
All the details that had been entered using the add option are viewable.

The amount of books that are included in the collection are also provided.

Git Log

The program that was created was developed with the use of version control.

As seen from the pseudo code above, the code was developed in small stages. Each version introduced a new method. This was very beneficial as it would ensure that old versions could be reused if errors occurred in future versions of the Library program.


zaB93 / UWL_CompS
Settings
Atlassian, Inc. [US]
https://bitbucket.org/HamzaB93/uwl_compsci4_progas1/commits/all

Author	Commit	Message	Date
Hamza	6a74bdb	Uploading Book, Console, Library and Stock .java files under a new project header. This is used to show the potential final version	an hour ago
Hamza	df725ff	Version7: Added more comments to each Classes Book, Library and Stock to show understanding of processes occuring. Strings	an hour ago
Hamza	4f0698c	Version6: For the Book class, the toString method has been edited so that the string is clearer. For the Stock class, getSize	2 days ago
Hamza	026361b	Updated Classes and Arrays: The Library class has now been changed to Book. In turn the Libraries array has now been	2014-12-08
Hamza	d692303	Adding an empty file for test purposes	2014-11-26
Hamza	d8824e3	Version5: This version contains the an ask method. This allows users to input their own data. This is last stage before creating an	2014-11-22
Hamza	310f5b0	Version4: Uploading the fourth version of the library program. The program now has access methods which allows certain	2014-11-22
Hamza	1891a45	Version3: Uploading version 3 of the library system. This version contains the set method where default values can be changed.	2014-11-16
Hamza	9a99083	Version2: Includes the main Library file. Version 2 of Library and Test use constructor with parameters.	2014-11-16
Hamza	8050b4d	Version1: Uploading the first version of the library program. Library.java being the main file. Default constructors have been	2014-11-16
Hamza	e4d751a	Test: Uploading a text file for test purposes	2014-11-16

The image above shows the commit messages that where given to each version of the programme. Each message detailed what was added to each uploaded file.

Program Listing

Book Class

Here is the listing of the Book class. The Book was the main method for the whole project.

```
package library;

public class Book
{
    //Variables
    String refNum; // Reference number
    String bookName; // Book name
    String genre; // Genre
    String publisher; // Publisher name
    int publishYear; // Published year
    int stock; // Stock


    //Set method - Used later to change the default value of Variables
    public void set(String aRefNum, String aBookName, String aGenre,
            String aPublisher, int aPublishYear, int aStock)
    {
        //The parameters are associated with the variables defined above
        refNum = aRefNum;
        bookName = aBookName;
        genre = aGenre;
        publisher = aPublisher;
        publishYear = aPublishYear;
        stock = aStock;
    }


    //Set method - creating default values with the
    public Book()
    {   //refNum    , bookName  , genre     , publisher, year,  stock
        set("0000" , "Unknown" , "Unknown" , "Unknown" , 0000, 2);
        /*The values are default. If using the method on its own,
         * these values will be used.
         */

    }


    //Set method - creating a constructor with parameters
    public Book(String aRefNum, String aBookName, String aGenre,
            String aPublisher, int aPublishYear, int aStock)
    {
        set(aRefNum , aBookName , aGenre , aPublisher , aPublishYear, aStock);
    }


    //Set individual data members -  details can be altered individually
    public void setRefNum (String aRefNum)
    {
        refNum = aRefNum;
    }
    public void setBookName(String aBookName)
    {
        bookName= aBookName;
    }
    public void setGenre (String aGenre)
    {
        genre = aGenre;
    }
    public void setPublisher(String aPublisher)
    {
        publisher = aPublisher;
    }
    public void setPublishYear(int aPublishYear)
    {
        publishYear = aPublishYear;
    }
    public void setStock(int aStock)
    {
        stock = aStock;
    }
    //Accessor methods - allows data to be provided to other classes
    public String getRefNum()
    {
        return refNum;
    }
    public String getBookName()
    {
        return bookName;
    }
    public String getGenre()
    {
        return genre;
    }
    public String getPublisher()
    {
        return publisher;
    }
    public int getPublishYear()
    {
        return publishYear;
    }
    public int getStock()
    {
        return stock;
    }

    /*The boolean checks if a reference number entered is already present.
     * Used later to compare if the reference number entered is already present
     * array list
     */

    public boolean hasRefNum(String aRefNum)
    {
        return refNum.equals(aRefNum);
    }

    /*Ask method - using the console class. Will allow users to enter their own
values
    *These values will be using the set method also
    */
    public void ask(String prompt)
    {
        System.out.println("");
        System.out.println("Enter The Details Of The Book\n");
        setRefNum (Console.askString("Enter The Book's Reference Number: "));
        setBookName (Console.askString("Enter The Name Of The Book: "));
        setGenre (Console.askString("Enter The Book's Genre: "));
        setPublisher (Console.askString("Enter The Name Of The Publisher: "));
        setPublishYear (Console.askInt("Enter Year The Book Was Published: "));
        setStock (Console.askInt("Enter The Stock: \n"));
    }

    //Print method - prints out values that have been either mutated or are de-
fault
    public void print()
    {
        System.out.println("Reference Number: " + refNum);
        System.out.println("Book Name: " + bookName);
        System.out.println("Genre: " + genre);
        System.out.println("Publisher: " + publisher);
        System.out.println("Published Year: " + publishYear);
        System.out.println("Stock: " + stock);
    }

    /*toString method- prints out strings and variables in a cleaner way.
     * Is used later when printing the details in the Library array class.
     */
    public String toString()
    {
        return "Reference No: " + refNum +"; Book Name: "+ bookName +"; Genre:"+
genre +"; Publisher:"+publisher+
                "; Published Year: "+ publishYear +"; Stock: "+ stock + "\n";
    }
}
```

Library Class

The library class was used to create array lists for the books that where to be added to the system along with a separate array list for the reference numbers.

```
package library;

//Imported ArrayList utility from java library
import java.util.ArrayList;

public class Library
{
    //ArrayList - The array list will store contents of the Library class
    ArrayList<Book> library;
    //ArrayList - Adding another property to the array called keys for use as an
index
    ArrayList<String> keys;

    /*Created constructors - Has a container which holds items from the Library
class
     * Another container for keys which involve storing reference numbers
     */
    public Library()
    {
        library = new ArrayList<Book>();
        keys = new ArrayList<String>();
    }

    //get.size method - will return a number of books stored in Library
    public int getSize()
    {
        return library.size();
    }

    /* Add method - after a the data is entered using the ask method, the data
will be added
     * library array. The reference number will be added to the keys array.
     */

    public void add(Book aLibrary)
    {
        library.add(aLibrary);
        keys.add(aLibrary.getRefNum());
    }

    /*Find method - finds a reference number.
     * If there is no reference number, return null
     * Else get it from the library class
     */
    public Book find(String aRefNum)
    {
        int index = keys.indexOf(aRefNum);
        if(index == -1)
        {
            return null;
        }
        else
        {
            return library.get(index);
        }
    }
    /* Print method - to show how many items/books there are in the
     * Libraries class
     */
    public void print (String header)
    {
        System.out.println(header);
        /* When i is 0, until it is less than Libraries, increment
         * by 1, get the object and print it out
         */
        for (int i=0; i < library.size(); i++)
            System.out.println(library.get(i));
    }
}
```

Stock Class

The Stock class contains the menu for the whole system. It accessed the Book class methods and stored the books into the Library array list.

```
package library;

public class Stock
{
    public static void main(String[] args)
    {
        //Creating an object - called stock using library class
        Library stock = new Library();

        //Using methods from the Book class
        Book library;

        //Define a default boolean called finish and assigned it as false
        boolean finished = false;

        //Defined a variable option which accepts a character only
        char option;

        System.out.println("Welcome To The Book Entry Sytsem For Your Library.
Options Are Shown Below\n ");
        //Menu loop
        //When the menu is not closed/finished/exited
        while( ! finished )
        {
            //Using console to ask for a character value stored in option
            option = Console.askOption("Enter A to Add , P to Print or Q to
Quit\n");

            //Switch - determines what happens when a certain value is stored in
option
            switch (option)
            {
            //When option = a/A
            case 'A':
                //Create a new object
                library = new Book();

                //Ask method - ask for the book details
                library.ask("Enter book details: ");

                /*Using getRefNum and find method,
                 * If book refNumber is present, give an error
                 */
                if (stock.find(library.getRefNum() ) != null)
                {
                    System.out.println("ERROR! Another book has this reference
number. \n");
                }

                //Else add that to the library
                else
                {
                    stock.add(library);
                }
                break;

            //When option = p/P
            case 'P':
                //Print the size of the array/ number of books in the stock
class with getSize
                System.out.println("Number of books: " + stock.getSize());

                //print all the books in the stock using print method
                stock.print("Books:");
                break;

            //When option = q/Q
            case 'Q':
                //Change finished to true and close the program
                finished = true;
                break;

            //When option is none of the above, do nothing
            case '\0':
                break;

            //Otherwise the input is wrong and an error message appears.
            default:
                System.out.println("ERROR! Enter either A , P or Q\n");
                break;
            }
        }

    }
}
```

Evaluation

In this section, I will discuss what I felt I did correctly and the many different additions that I would add to future projects of this nature.

What I did well

I feel that each of the features that I included in the programme worked in a simplistic way. This meant that the features worked in a way that they did the job. This shows that there is room for improvement.

The add option, using the ask method, did work. The details where saved accordingly into the array list. This linked well to the print method which showed the book details.

The print method also worked accordingly. The details of the books were displayed correctly. This, however, needed amendments which were identified in the testing phase of the development of the programme.

The menu also worked well. When the user entered an option, the option/ method loaded correctly without errors. The menu would reject any entries that were not part of the if statement and would loop back to the menu.

What I would change

A major problem encountered during testing was the fact the error handling was not part of the ask method in the program. If a wrong data value was entered into certain fields, an error would cause the whole programme to halt. This would then require the user to re execute the programme.

A way that this problem can be solved is by adding error handling to the ask method in the Book class. A loop can be used to prevent certain values to be entered and the question can be asked again. For example, if the stock of the book was not an integer value, a programmed error message would be displayed but still allow the user to carry on entering their data till it is correct.

I would also add an option to edit certain data members. For example, I would add a method that could change the stock of the books that had been previously entered. This could use the get and setting of individual data members, in this case the stock. Another way the book data members could be edited is by having a rented status. The book's details could be edited to change the status of the book to rented or available.