IN PARTNERSHIP WITH PLYMOUTH UNIVERSITY

Name: Hewawasam P Madhavee
Student Reference Number:10516225

Module Code: SOFT225SL	Module Name: Software Engineering for the internet using Java
Coursework Title: Java Based Application Development	
Deadline Date:	Member of staff responsible for coursework:
Programme:	

Please note that University Academic Regulations are available under Rules and Regulations on the University website www.plymouth.ac.uk/studenthandbook.

Group work: please list all names of all participants formally associated with this work and state whether the work was undertaken alone or as part of a team. Please note you may be required to identify individual responsibility for component parts.***We confirm that we have read and understood the Plymouth University regulations relating to Assessment Offences and that we are aware of the possible penalties for any breach of these regulations. We confirm that this is the independent work of the group.***

Signed on behalf of the group:

Individual assignment: ***I confirm that I have read and understood the Plymouth University regulations relating to Assessment Offences and that I am aware of the possible penalties for any breach of these regulations. I confirm that this is my own independent work.***
Signed:

Use of translation software: failure to declare that translation software or a similar writing aid has been used will be treated as an assessment offence.
I *have used/not used translation software.
If used, please state name of software……

Overall mark _____% **Assessors Initials _____** **Date_____**

SOFT225SL

PROJECT REPORT FOR

LIBRO,

A STAND-ALONE APPLICATION

FOR

LIBRARY MANAGEMENT SYSTEM

BIMLA MADHAVEE
10516225
BSC-PLY-COM-13.1-038

Acknowledgement

I take this opportunity to express my gratitude toward everybody who supported me throughout this project. I sincerely thank our lecturer Ms.Manoja Weeresekara for her support toward this project and I'm very thankful for her aspiring guidance until the end of the project. I would also like to express my gratitude toward my family members and friends who helped me directly and indirectly in many ways to make this project a success.

TABLE OF CONTENT

INTRODUCTION

What is LIBRO?

Libro is a standalone application which is created for Library Management system. This is built using Java with Object Oriented Programming where code is reused. Libro builds a friendly platform between the library system and its users. This mainly consist registration of books and registration of library members. When considering about the books I have categorized books into two groups. Namely reference books and general books. Books of these types are registered commonly. But according to the preference of the librarian he/she can select general books or reference books where only special facts are been considered. In registration of members creating of profiles are done. And they can view and edit profile data later. These functions are done only by the librarian.

PROPOSED SYSTEM

This is a standalone application for a library management system. Libro emphasize two main features, which are namely registration of books and registration of members. These two functions are carried out by librarian only. As mentioned earlier to cover all the book types I categorized books into reference books and general books. To carry out any of the features librarian should log into the system by entering the username and password correctly. If the user is a novice user that person have to sign up and then by log in he/she can select books or members section.

In registering books librarian can choose either reference books or general books. Or else librarian can register any book without considering their type. But a book which is going to be registered under these two categories should be first registered in books section. In book registration adding a new book, editing, deleting and viewing books are implemented. Each time when these actions take part message boxes emerge to inform the user regarding the actions. Other than those features librarian is allowed to search books by its title. All the records are displayed for the librarian. Whenever a change is occurred to the records it's frequently refreshed and displayed to the user automatically.

Librarian can select general books after log into Libro. Afterward librarian can add new books, edit, delete and view as in book registration.

Here also records are displayed in a table. If librarian select reference book registration at the beginning adding of new books are allowed. List of reference book names are shown in a jList. In general and reference book section only essential facts are been considered in registration. To register a book in general or reference section first the user should register that book in book section

In the member registration section librarian can add new members through the data provided by the members and then a profile for each member is created. Here librarian have to add the supplied photograph of the particular person. Librarian can view member's profile and if the entered data are not accurate librarian can edit data. If the particular person is no longer a member deleting profiles are also allowed for the librarian.

Another feature is that user can logout when he feel to do so. For example user can logout after registering books or members.

ASSOCIATION OF OBJECT ORIENTED PROGRAMMING WITH LIBRO

Encapsulation:

All private, protected, public access modifiers are used throughout the application. Specially protected variables are used within the database (dbCon.java) class to control the accessibility. As password which is required for the login should be accessed only within Librarian class, private modifier is assigned to the password variable.

```
public class dbCon {



        public  Connection connection = null;


        public static String driver = "com.mysql.jdbc.Driver";
        public static  String url = "jdbc:mysql://localhost:3306/library";

        //db details
        public static String username = "root";

         protected static  String password = "";
```

Polymorphism:

Polymorphism include method overloading and method overriding. Libro consist Book.java class which contain addBook(), deleteBook(), uploadRecord() which are used to manipulate data in the database. These methods are overloaded in the GeneralBook.java class. viewBookDetails() method which is implemented in Book.java class is overridden again in the GeneralBook.java class. Through these polymorphism is used in Libro

```
//inserting record to database
//overloading method from book class
public void addBook(int genrlBkId,int bkId,int shelf_id,String gnrlBkName)
{

    String gnBkQuery = "insert into generalbook( general_bk_id,book_id,genaral_bk_name,shelf_id)"+"values"+"('"+genrlBkId+"','"+bkId+"','"+gnrlBkName+"','"+shelf_id+"')
     try {
        pst = connection.createStatement();
        int rws = pst.executeUpdate(gnBkQuery);

        if(rws > 0)
        {
            JOptionPane.showMessageDialog(null, "data regarding general books are added");

        }

        } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
         }
}

//load data to the table
//overriding the viewBookDetails() in book class
public void viewBookDetails()
 {

    String sql = "select * from generalbook";
    try
    {
        //Statement st = connection.createStatement();
        PreparedStatement pst = connection.prepareStatement(sql);
```

Inheritance:

In Libro there is super class sub class relationship.book.java is the super class and GeneralBook.java and RefernceBook.java are the subclasses. So that members in the superclass are inherited to subclasses. Other than that database class (dbCon.java) is inherited by Book.java, log.java, Librarian.java and other classes indirectly where each class need to access the database class.

```
import java.sql.*;

public class GeneralBook extends Book{


    public int  gnrlBookId,shelfNo;
    public String gnrlBookName;
```

```
public class Book extends dbCon {

    //declaring variables

        public int b_id,t_id,cost;
        public String b_title,auth_fname,auth_lname,publisher;
        //public Date date;
        Statement pst = null;
        ResultSet rs = null;
```

Abstraction:

Abstract methods are declared inside interface Person and those methods are implemented inside Librarian.java class separately. There are three abstract methods namely log () method which implement the method required for log into the library system and searchBook () method which is required in searching books by the librarian and signUp () method which is for novice users

Following screenshot denotes the declaration of the abstract methods and the next screenshot denotes how one of those abstract method is implemented in a separate class

```
    public abstract void  log(String name,String pass);
   public abstract void searchBook(String bkName);
    public abstract int signUp(String fnme,String lnme,String addOne,String addTwo,String addThree,String telNu
```

```
public void searchBook(String search)
{

    String searchQuery = " select * from book where b_title = ? ";
    //String searchQuery = " select b_id,b_title,auth_fname,auth_lname,cost,publisher from book whe
    try {
        PreparedStatement pst = connection.prepareStatement(searchQuery);

        pst.setString(1, search);

         rs = (ResultSet) pst.executeQuery();


    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

Interfaces and static, final variables:

Libro consist one interface which consist static final variable and abstract methods. Abstract methods for login, signup and searching books are declared inside the interface. For the interface person, librarian, library members are included. Only these persons can log into the system and only they can search for books. Therefore those two methods are abstract and included in the interface.

```
public interface Person {
    public static final String fname = null;
    public static final String lname = null;
    public static final String address = null;
    public static final int tel = 0;
    public static final int NIC = 0;
    public abstract void  log(String name,String pass);
   public abstract void searchBook(String bkName);
    public abstract int signUp(String fnme,String lnme,String addOne,String addTwo,String addThree,String telNum,String nic,String usernm,S

}
```

```
public class Librarian extends dbCon implements Person{

    public String fname,lname,address,username,dob;
    public int SSN;
    private String password;
    ResultSet rs ;
```

Packages, Classes and objects:

Libro contain classes and these classes contain various methods. These methods are evoked by creating instances of those classes. All the classes of Libro are inside one package called library.

The following screenshot shows the package, class and the object created inside the class.

```
package library;

import java.awt.EventQueue;


public class log extends dbCon{

    private JFrame frmLogin;
    private JTextField txtUser;
    private JTextField txtPassword;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    log window = new log();
                    window.frmLogin.setVisible(true);
```

Methods:

All the processes that take place in Libro are implemented inside different methods. Inside each class required methods are created with meaningful names. For example methods for inserting a new book, deleting a book, viewing the book list are implemented separately.

```
    //deleting records from the database

    protected void deleteBook(int bNumber)
    {

        String delSql = "delete from book where b_id = '"+bNumber+"'";
        try
        {
             pst = connection.createStatement();
            int n3 = pst.executeUpdate(delSql);

            if( n3 > 0)
            {
            JOptionPane.showMessageDialog(null, "we delete the book from the database");
            }

        }
        catch(Exception ex)
        {
        JOptionPane.showMessageDialog(null, "we coudnt delete the record");

        }
    }
```

Getters and setters:

Since password variable inside Librarian.java class is private I used getters and setters so that password variable will be accessible.

```
public class Librarian extends dbCon implements Person{

    public String fname,lname,address,username,dob;
    public int SSN;
    private String password;
    ResultSet rs ;


    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
```

Constructors:

Within Libro inside classes there are constructors created for initialization. Inside the database class I have created a constructor as follows

```
public class dbCon {



        public  Connection connection = null;
         public dbCon()
         {
             connection = dbCon.dbConnector();
         }
```

Exception handling:

To prevent the application being stopped suddenly due to errors that occur during execution I have used exception handling in each class. Whenever a SQL query is running I used exception handling to ensure that the connection is successful and that the query is correct.

In some classes I have used exceptions to inform the user if they enter duplicate data to the database .The following demonstrate the exception handling correctly

```
            try {

          pst = connection.createStatement();
          int st = pst.executeUpdate(addSql);

          if(st > 0)
          {
                    JOptionPane.showMessageDialog(null, "records inserted");
          }
                else
                    JOptionPane.showMessageDialog(null, "coudnt");


            } catch (SQLException e) {

                e.getMessage();

            }
```

OTHER FEATURES OF LIBRO

File Upload:

In registering library members, librarian is allowed to upload an image file to the database and when viewing the profiles those image files of .jpg and .png are displayed along with the details of the member. This is a special feature within Libro. For this I have used jFileChooser

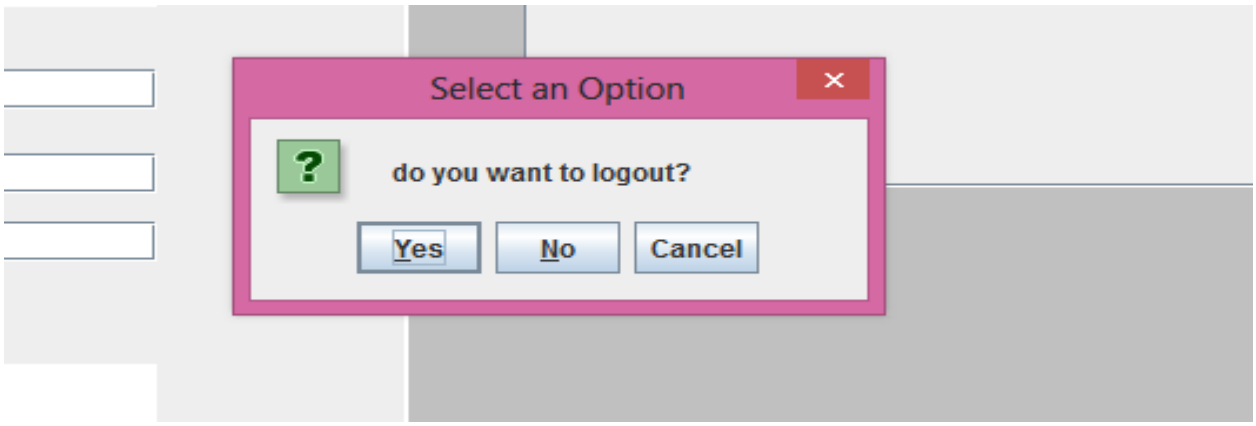
```
    public void actionPerformed(ActionEvent e) {

        fileChooser = new JFileChooser("C:\\",FileSystemView.getFileSystemView());
         fileChooser.setFileFilter(new FileNameExtensionFilter("image files", "jpg","png"));
         int returnVal = fileChooser.showOpenDialog(contentPane);
         if(returnVal == JFileChooser.APPROVE_OPTION){
             String filename = fileChooser.getSelectedFile().getName();
             String extension = filename.substring(filename.lastIndexOf("."));
             if(extension.equalsIgnoreCase(".jpg")||extension.equalsIgnoreCase(".png"))
             {
                 txtPath.setText(fileChooser.getSelectedFile().getPath());
             }
             else
             {
                 JOptionPane.showMessageDialog(null, "kindly select the required file only");
             }
         }
         else
         {
             txtPath.setText("no files selected");
         }

    }
});
```

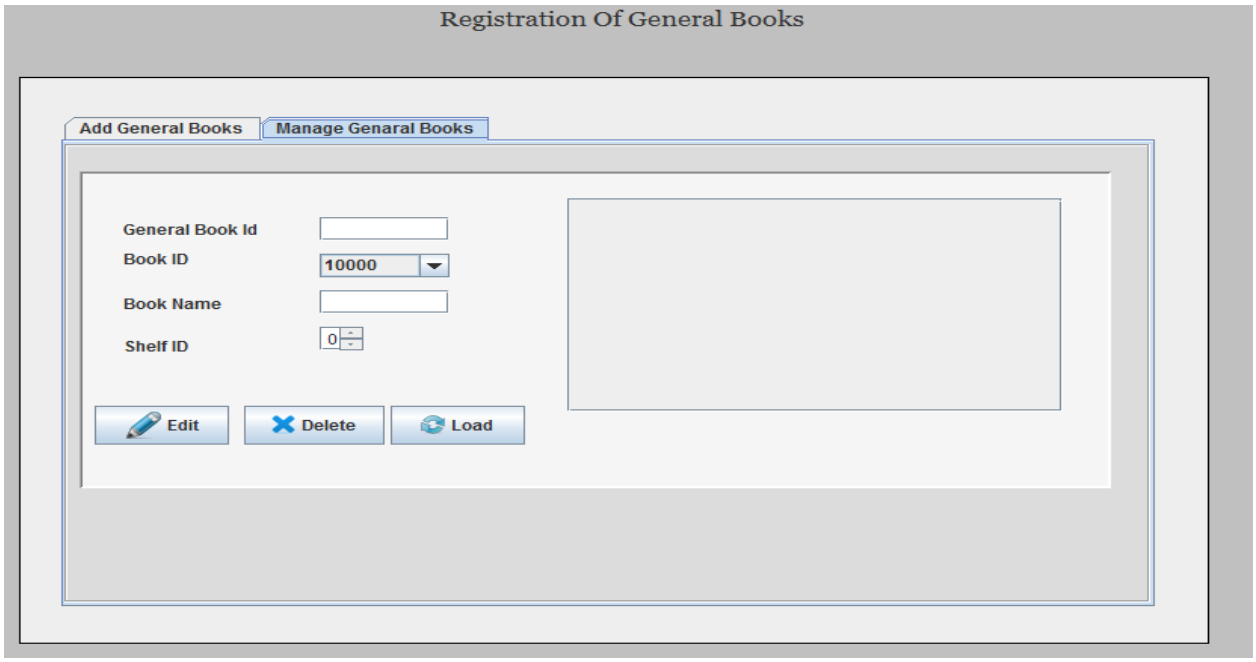
External Libraries

Mysql-connector-java3.1.14-bin.jar which used for database connection and rs2xml**.jar** which was required for jTables are used as external libraries.

Graphical user interfaces:

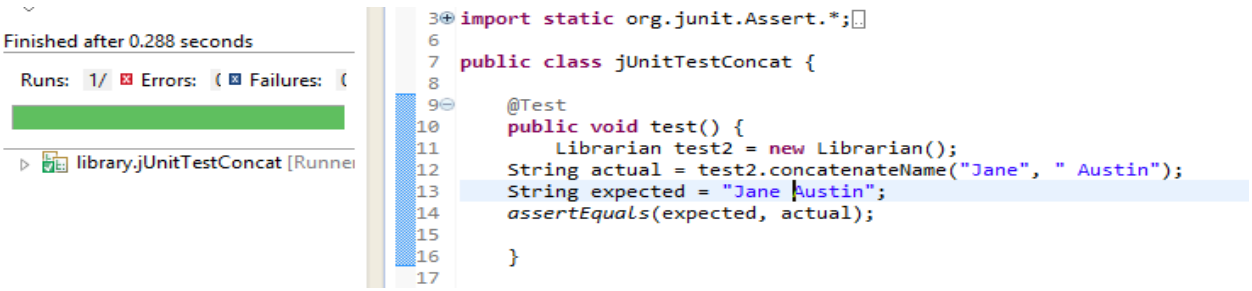
In creating Libro I used java Graphical User Interfaces. In creating the user interfaces I used Abstract Windowing Toolkit (AWT) and Swing in java. Libro consist simple user friendly interfaces which are easily understandable. To make it more user friendly I used *nimbus Look and feel*. For each button an icon is also inserted so that user will be able understand the process carried out by each button. Another thing is that to make Libro user friendly always message boxes are displayed for the user.



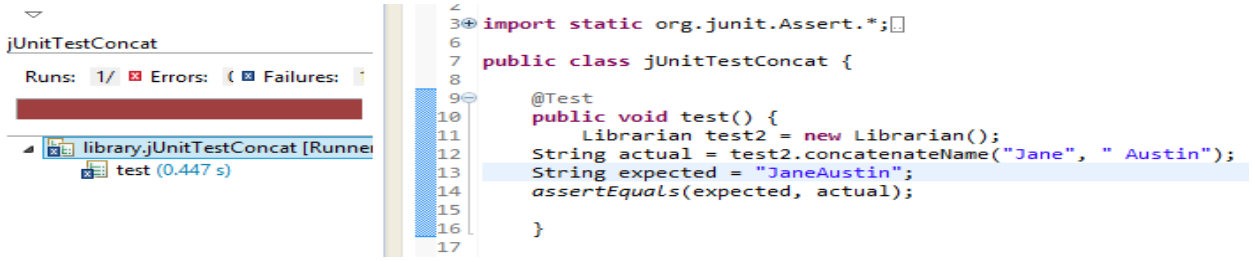


Junit Testing:

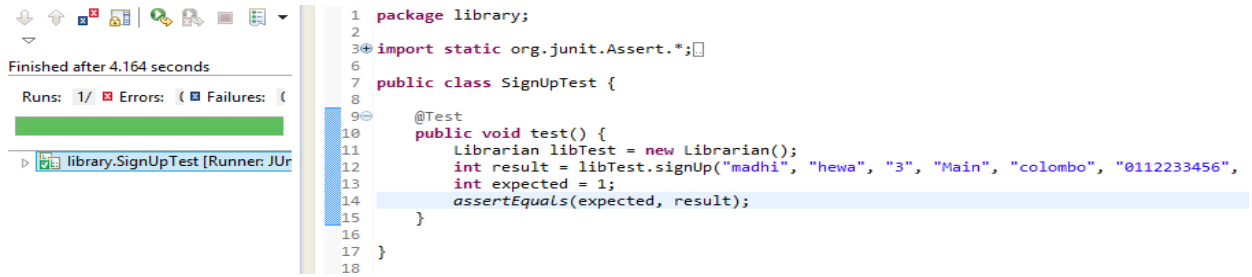
To make sure that Libro application is working correctly I used jUnit test where necessary. Annotations are also inserted. The following screenshots will demonstrate three different jUnit tests where two tests provide a positive result while the other provide a negative result. Here I have used two jUnit tests just to show the difference between the two results.



Test with a positive result



Test with a negative result



JUnit test for sign up method

Comments: To increase the readability and understandability commenting is done where necessary.

String class: Inside the application String class is frequently used where necessary.

SQL queries:

To manipulate data in the database I have used SQL queries. Mainly insert records, update records, delete records are often found in the application.

```
ected void editMembrs(String fname, String lname, int age,String address1,String address2)

String editMmbrSql = " update member set member_fname ='"+fname+"',member_lname ='"+lname+"',age ='"+age+
        "',add1 = '"+address1+"',add2 = '"+address2+"' where member_fname = '"+fname+"'";
```

Methodologies

To implement Libro I used **java Eclipse IDE** which is most commonly used in the current industry. To create the database of the application I used *phpMyAdmin* and to store data in the database ***MySQL* database management system** is used. Insert, update, delete, select queries are often used within the application. One special feature in the database is **image files** are inserted as *BLOB* type.

Object oriented programming in java language is mainly used in implementing the application. Classes, objects, methods, polymorphism, encapsulation, inheritance, multiple inheritance, abstraction are key features. Interfaces are created using java GUI. To make Libro more user friendly default GUI are changed to ***Nimbus Look and Feel*** .For each button click, event handlers are used. Other event handlers like mouse click events are used where necessary.

Using of **java Swing class and Abstract Window Toolkit (AWT)** user interfaces are designed. String class is also used with in the application. Validations for user inputs are done using message dialog boxes. **JTable**, **jFileChooser, jDesktopPane, jList** should be specially mentioned.

Few **external libraries** such as **mysql-connector-java3.1.14-bin.jar** which used for database connection and **rs2xml.jar** which was required for jTables.

Exposure

This is a valuable opportunity to practice the object oriented programming in java individually. Before starting the application I went through the concepts and got thorough with them.

The other important thing was I learned how to use java swing class and java AWT class in designing the user interfaces. I should specially mention about file upload as it was a key feature within Libro.

I learned how to deal with jFileChooser in java which was something new to myself.
Another fact is that I obtained a good experience in using string class, packages, getters and setters etc. which are marking points in java language. I got the opportunity in designing required diagrams for the requirement analysis of the application. Gained the basic experience in handling the MySQL queries in manipulating data in the database. Another main experience that I gained was using of jUnit testing for testing phase of the project.

Conclusion

Libro, a library management system was implemented using J2SE which is a standalone application using OOP concepts mainly. It was a success. I got a vast knowledge regarding J2SE and basic knowledge on MySQL language. This was a great opportunity to gain experience in developing a standalone application in java.

References

- Stack overflow. (2014) Stack overflow. [ONLINE]
  Available at: http://stackoverflow.com/
  [Accessed at 28th December 2014]
- Oracle. (2014). the Java™ Tutorials. [ONLINE]
  Available at: http://docs.oracle.com/javase/tutorial/.
  [Accessed at 20th December 2014]
- 4shared. (2014). 4shared. [ONLINE]
  Available at: http://www.4shared.com/get/69Se66yD/rs2xml.html
  [Accessed at 30th November 2014]
- MySQL. (2014) Downloads. [ONLINE]
  Available at: http://dev.mysql.com/downloads/connector/j/3.1.html
  [Accessed at 30th November 2014]