Criterion C: Development

Techniques:

1. GUI Programming
    a. Navigation
    b. JTables
    c. Dialogue Boxes

2. Object Oriented Programming (OOP)
    a. Objects & Encapsulation
    b. JFrames

3. MySQL Database
    a. JDBC
    b. MySQL Queries

4. Data Validation

5. Exception Handling

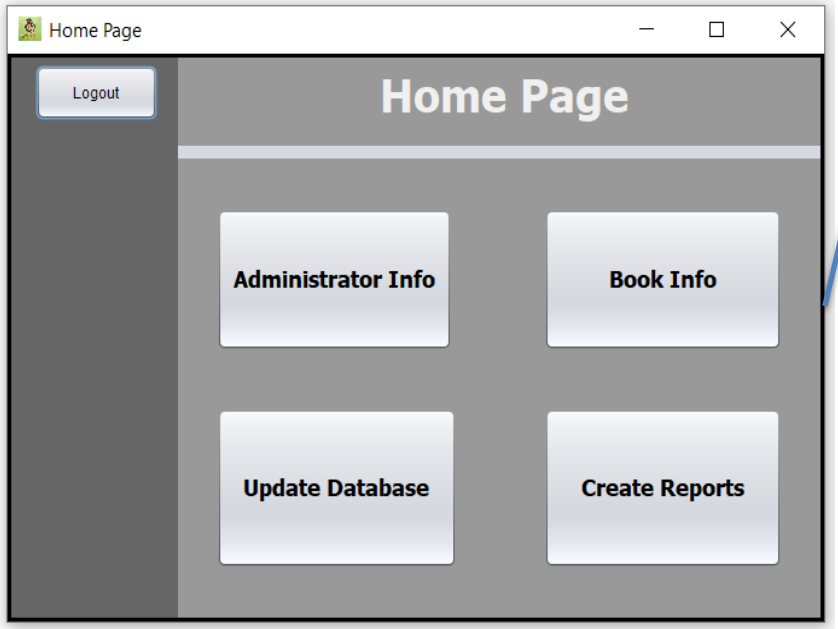
6. Parameter Passing and Global Variables

7. Generating Reports

8. Library Functions

1. GUI Programming

a) Navigation



When this JButton is clicked, it will make the Book Info Page visible using the function setVisible( )

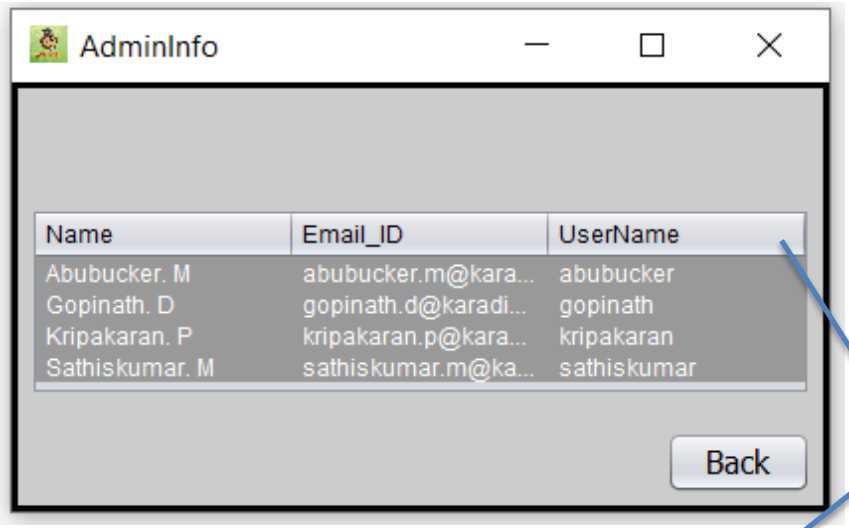
JButtons are used throughout the program to navigate from one page to another. Since NetBeans can detect when a button is pressed and trigger a particular "Action", it allows for smooth transition from one page to another.

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    // Takes administrator to Create Reports page:
    CreateReports cr = new CreateReports();
    cr.setVisible(true);
    setVisible(false);
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // Takes administrator to Update Database page:
    UpdateDatabase UD= new UpdateDatabase();
    UD.setVisible(true);
    setVisible(false);
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Takes administrator to Admin Information page:
    AdminInfo adl= new AdminInfo();
    adl.setVisible(true);
    setVisible(false);
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // Takes administrator to Book Information page:
    BookInfo bkl= new BookInfo();
    bkl.setVisible(true);
    setVisible(false);
}
```

Creates a new object to reference the "CreateReports" JFrame and then makes "CreateReports" JFrame visible by setting its Boolean value to "true" and makes the current page (in this context main menu) invisible by setting its Boolean value to "false"

b) JTables

This software revolves around displaying and filtering large amounts of data. To display the raw or filtered data JTables are used. This works well with the result set model of MySQL data as the field names and records can be directly inserted into the JTable by using a single function.

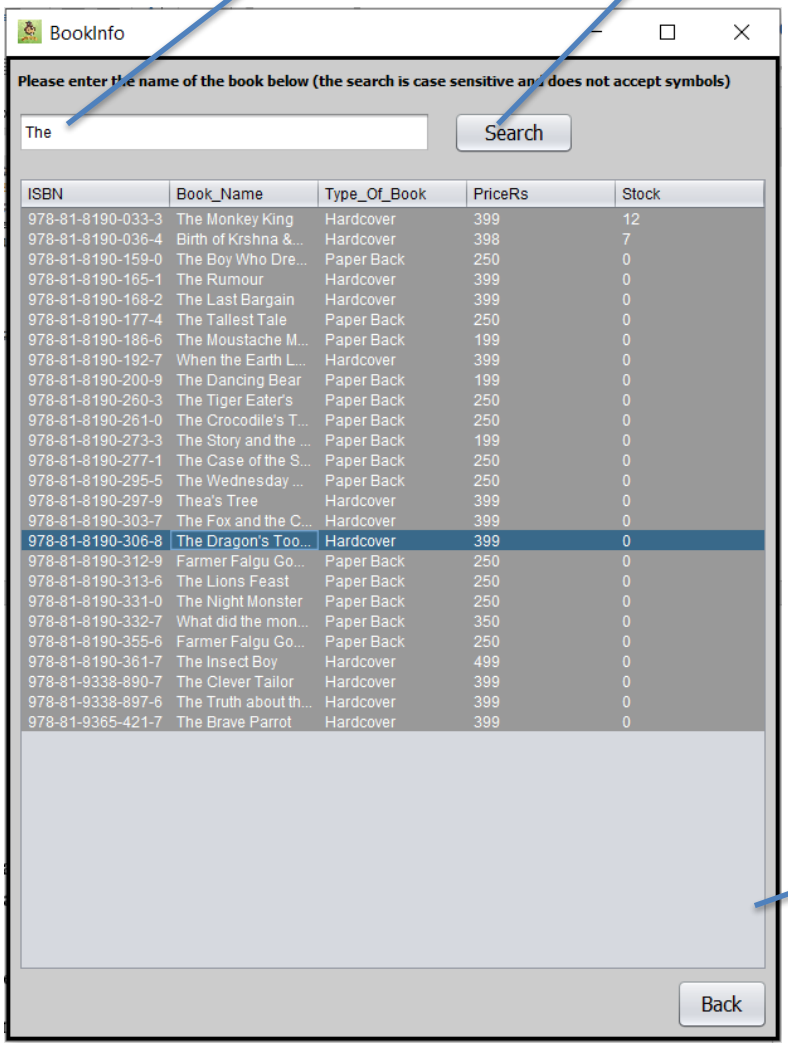


JTable containing Name, Email ID, and User name of each administrator retrieved from the MySQL database [from the table user_details]

```
private void updateTable()
{
    try
    {
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection Conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/cs_ia_data","root","xeno28972");
        String sql= "Select `Name`,`Email_ID`,`UserName` from user_details";
        PreparedStatement pst= Conn.prepareStatement(sql);
        ResultSet rs = pst.executeQuery();
        Display_Admin.setModel(DbUtils.resultSetToTableModel(rs));
        Conn.close();
    }
    catch(Exception e)
    {
        JOptionPane.showMessageDialog(null,e);
    }
}
```

Search Parameter for function to sort the JTable below by "Book_Name"

Button that is pressed to search the table, calling the bookSearch( ) function



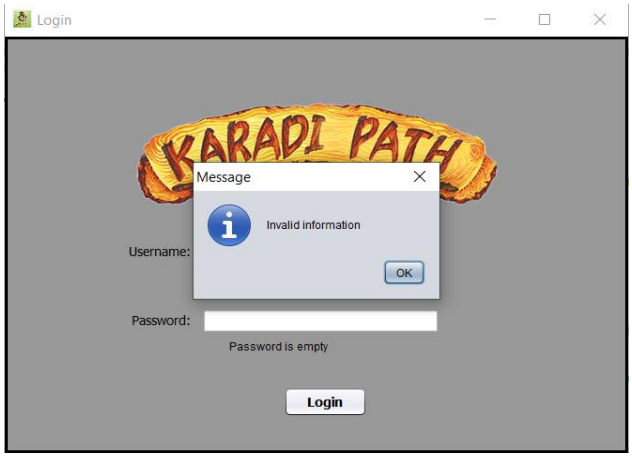
For example, the search parameter "The" returns the ISBN, Book_Name, Type_ Of_Book, PriceRs, and Stock of any title that has the characters "The" in it. This is shown in the image to the left

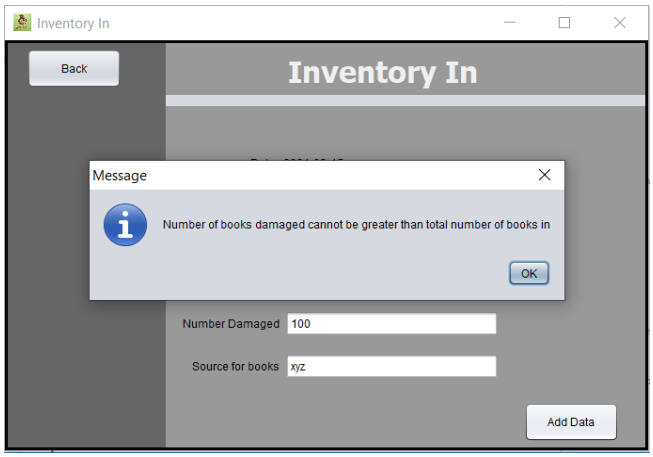
Returns every row from book_identification table where the book name is like the entered search parameter. This filtered result set can be directly entered into the JTable

```
private void bookSearch()
{
    try
    {
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection Conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/cs_ia_data","root","xeno28972");
        //If the text field is empty, the data forall the books is returned.
        if (Search.getText().trim().isEmpty())
        {
            String sql= "Select * from book_identification";
            PreparedStatement pst= Conn.prepareStatement(sql);
            ResultSet rs = pst.executeQuery();
            Display_Books.setModel(DbUtils.resultSetToTableModel(rs));
        }
        //If the search parameter is found, every book which has the search string is rerutrned.
        else
        {
            String sql="Select * from book_identification where Book_Name like '%"+Search.getText()+"%'";
            PreparedStatement pst= Conn.prepareStatement(sql);
            ResultSet rs= pst.executeQuery();
            Display_Books.setModel(DbUtils.resultSetToTableModel(rs));
        }
        Conn.close();
    }
    //If both these are not satisfied, the user is notified that there is no such book in the inventory and is asked to renter the data
    catch(Exception e)
    {
        JOptionPane.showMessageDialog(null,"The book name was not found- make sure to not enter special characters and the wrong case");
        Search.setText("");
    }
}
```

c) Dialogue Boxes

Dialogue boxes are an important GUI component which allow the software to notify the user if any changes need to made to data entry and detect logical error and point it out to the user. This is made possible using the JOptionPane, a JavaSWING feature.






```
else if(Integer.parseInt(InvenIn.getText())<Integer.parseInt(NumDmg.getText()))
{
    JOptionPane.showMessageDialog(null,"Number of books damaged cannot be greater than total number of books in");
    InvenIn.setText("");
    NumDmg.setText("");
}
```

In this context, if the value entered for “InvenIn” is lesser than the value entered for “NumDmg”, a logical error is detected. Since this can only be fixed by data reentry, the user is prompted to re-enter the correct data through a customizable JOptionPane

```
if(rs.next())
{
    JOptionPane.showMessageDialog(null,"Login Successfull");
    setVisible(false);
    MainMenu mm = new MainMenu();
    mm.setVisible(true);
}
else
{
    JOptionPane.showMessageDialog(null,"Invalid information");
    UserName.setText("");
    Password.setText("");
}
Conn.close();
```

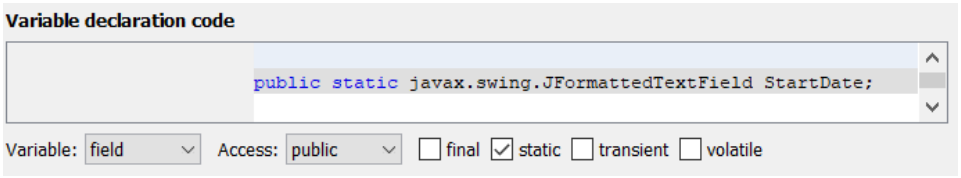
If the credentials that the user entered is valid, the user is directed to the MainMenu/ Homepage

However, if the credentials are left empty or if the credentials entered are wrong, the user is not allowed into the software. Instead, the user is shown a JOptionPane which prompts the user to reenter the correct data to access the software.

2. Object Oriented Programming (OOP)

a) Objects and Encapsulation

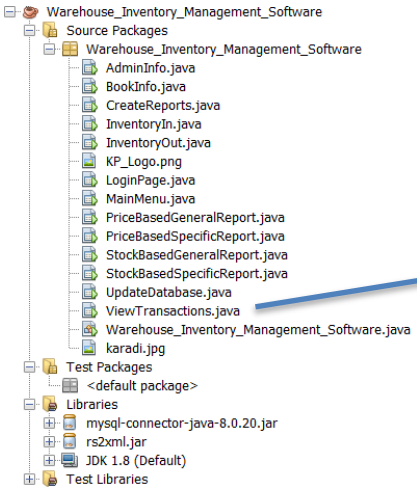
Objects allow for passing of both data and methods from one class to another, increasing flexibility and reusability of the code and encapsulation allows for thorough segregation of code

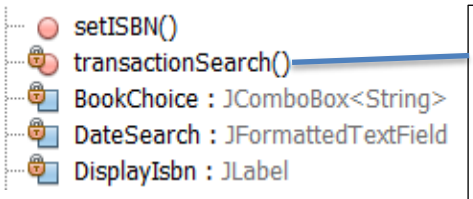


Encapsulation allows me to assign access modifier to variables within a class

b) JFrames

JFrames are the basic components of a NetBeans project and can be used to separate and organize different pages in the software is a hierarchical and logical manner. They each contain attributes such as JButtons and contain methods such as jButtonActionPerformed, acting as a class. Each of them is used to create a layout of a particular page.





Each JFrame can be treated as a class. For example the JFrame "View Transactions" has its own attributes such as JButtons and JTables and also has functions such as transactionSearch( )

The image to the right shows the different JFrames in the project

3. MySQL Database

a) JDBC

JDBC stands for Java Database Connectivity. JDBC is a Java Application Programming Interface to connect and execute the query with the database[1]. It can be used to connect to any MySQL schema, the code below[2] connects to the schema "cs_ia_data" using the username "root" and the password "xeno28972"

Connecting with the username and password

```
Class.forName("com.mysql.cj.jdbc.Driver");
Connection Conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/cs_ia_data","root","xeno28972");
```

JDBC can also be used to pass parameters in an ordered manner in SQL queries and get a result set containing the data. The code below passes the ISBN of the book and the date entered to search the database and return the sought data (transactionSearch( )).

[1] https://www.javatpoint.com/java-jdbc

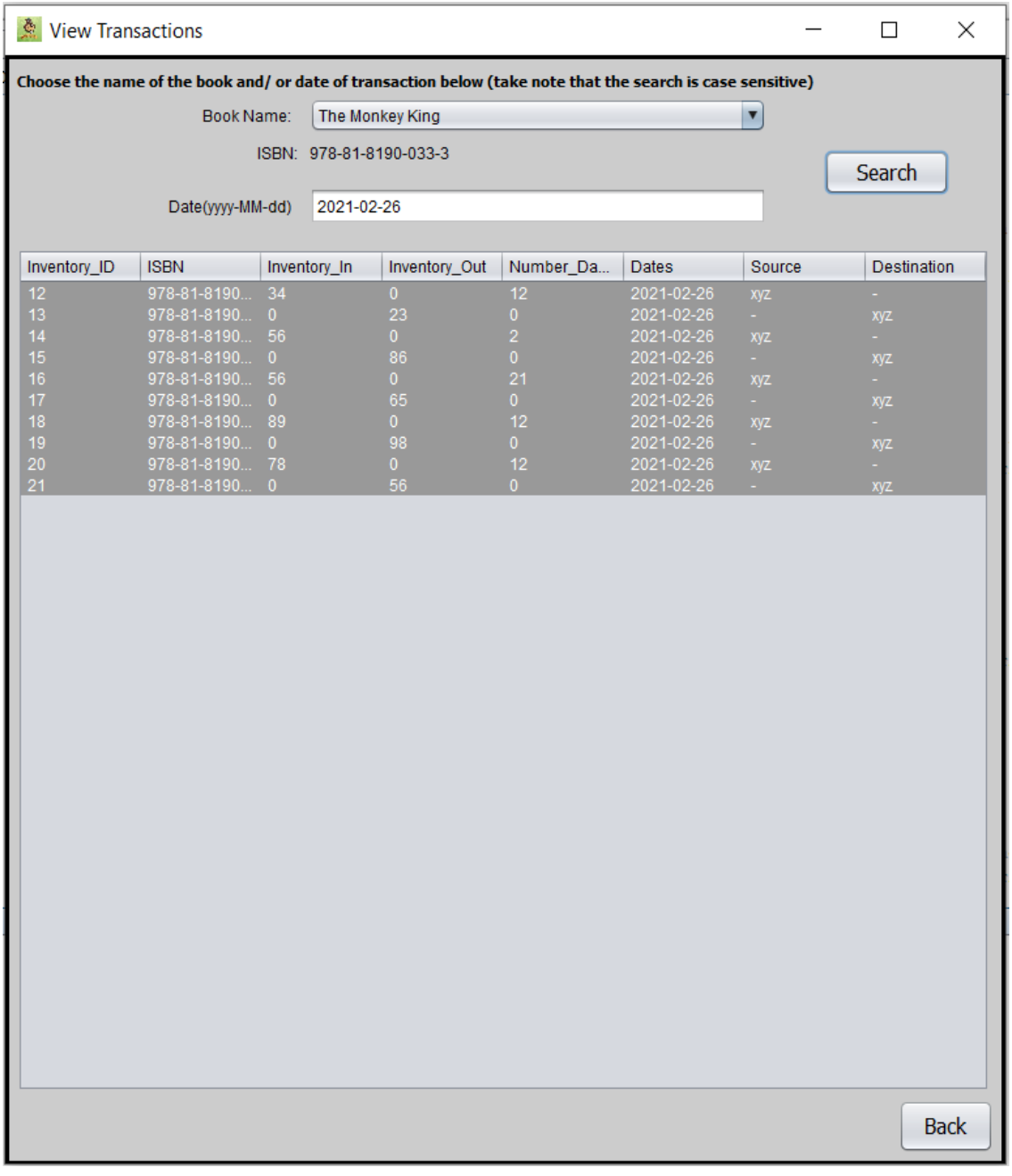
[2] https://www.javatpoint.com/example-to-connect-to-the-mysql-database

```
String sql= "Select * from transaction_repository where ISBN=? and Dates=?";
PreparedStatement pst= Conn.prepareStatement(sql);
pst.setString(1,DisplayIsbn.getText());
pst.setString(2,DateSearch.getText());
ResultSet rs = pst.executeQuery();
Display_Books.setModel(DbUtils.resultSetToTableModel(rs));
```

"Select * from warehouse details where ISBN=DisplayIsbn.getText( ) and Dates=DateSearch.getText( )"

The number within the brackets determines the position in which the data is passed and the variable after the number is the data that will be passed. This facilitates querying a database based on user input.

Example of the query above in action- filtering a result set according to date and book name



b) MySQL Queries

As this project uses a relational database component (from MySQL), MySQL queries are used to select certain groups, starting with the SELECT statement, of data in a structured manner: namely a table or a result set.[3] However, the queries can also start with an INSERT/UPDATE statement, which allows the values in the database to be changed. In this software, these two are used along with other mid line statements such as WHERE[4], LIKE[5],SUM[6], and JOIN[7] statements to narrow down and specify the queries by joining multiple tables and calculating values for each row.

```
int midStock= Integer.parseInt(InvenIn.getText())-Integer.parseInt(NumDmg.getText());
String sql2="update book_identification set Stock=Stock+? where ISBN=?";
PreparedStatement pst2= Conn.prepareStatement(sql2);
pst2.setInt(1,midStock);
pst2.setString(2,DisplayIsbn.getText());
pst2.execute();
JOptionPane.showMessageDialog(null,"The data has been sucessfully added");
InvenIn.setText("");
NumDmg.setText("");
Source.setText("");
```

Wherever ISBN is equal to DisplayISBN, the stock is changed to midStock

```
//Inserting the number of books in and the number of books damaged as entered by the user
Class.forName("com.mysql.cj.jdbc.Driver");
Connection Conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/cs_ia_data","root","xeno28972");
String sql= "insert into transaction_repository(Dates,ISBN,Inventory_In,Inventory_Out,Number_Damaged,Source,Destination) values(?,?,?,?,?,?,?)";
PreparedStatement pst= Conn.prepareStatement(sql);
pst.setString(1,DateDisplay.getText());
pst.setString(2,DisplayIsbn.getText());
pst.setInt(3,Integer.parseInt(InvenIn.getText()));
pst.setInt(4,0);
pst.setInt(5,Integer.parseInt(NumDmg.getText()));
pst.setString(6,Source.getText());
pst.setString(7,"-");
pst.executeUpdate();
```

Inserts the text from each of the JLabels, and zeroes into the respective values, to add a row in the transaction_repository

On the other hand, the below query returns a result set containing the fields total number damaged, total inventory in, total inventory out, stock flow, and percentage damaged for a single book within a set date range- the query for a stock based specific report. The values from “IsbnDisplay”, “IsbnDisplay”, “DisplayStartDate”, and “DisplayEndDate” are passed as parameters in their respective positions to complete the query.

```
Class.forName("com.mysql.cj.jdbc.Driver");
Connection Conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/cs_ia_data","root","xeno28972");
String sql="select transaction_repository.ISBN,SUM(Number_Damaged) as \"No.Damaged\", SUM(Inventory_In) as \"Inventory In\","
+ "SUM(Inventory_Out) as \"Inventory Out\",(SUM(Inventory_In)-SUM(Number_Damaged)-SUM(Inventory_Out)) as \"Stock Flow\","
+ "((SUM(Number_Damaged)/SUM(Inventory_In))*100) as \"% Damaged\" from book_identification JOIN transaction_repository "
+ "ON book_identification.ISBN = transaction_repository.ISBN where book_identification.ISBN=? and transaction_repository.ISBN=? "
+ "and Dates between ? and ? group by transaction_repository.ISBN";
PreparedStatement pst= Conn.prepareStatement(sql);
pst.setString(1,IsbnDisplay.getText());
pst.setString(2,IsbnDisplay.getText());
pst.setString(3,DisplayStartDate.getText());
pst.setString(4,DisplayEndDate.getText());
```

[3] https://www.digitalocean.com/community/tutorials/introduction-to-queries-mysql
[4] https://www.w3schools.com/sql/sql_where.asp
[5] https://www.w3schools.com/sql/sql_like.asp
[6] https://www.w3schools.com/sql/sql_count_avg_sum.asp
[7] https://www.w3schools.com/sql/sql_join.asp

```
1  -- Example query for a stock based specific report--
2  Select transaction_repository.ISBN,SUM(Number_Damaged) as
   "No.Damaged", SUM(Inventory_In) as "Inventory In",SUM(Inventory_Out
   ) as "Inventory Out",(SUM(Inventory_In)-SUM(Number_Damaged)-SUM(
   Inventory_Out)) as "Stock Flow",((SUM(Number_Damaged)/SUM(
   Inventory_In))*100) as "% Damaged" FROM book_identification JOIN
   transaction_repository ON book_identification.ISBN =
   transaction_repository.ISBN where book_identification.ISBN=
   "978-81-8190-033-3" and  transaction_repository.ISBN=
   "978-81-8190-033-3" and Dates>="17-12-2020" and Dates<="18-12-2020";
```

The JOIN...ON keywords together specify on which column, and on which row if necessary, the two tables, in this context transaction_repository and book_identification should interlink. In this case, the 2 tables join on the ISBN of the book to create one merged table which share data with each other. In the price based specific query below, the PriceRS is shared from book_identification to transaction_repository.

```
4  --Example query for price based specific report--
5  Select transaction_repository.ISBN,(SUM(Number_Damaged)*PriceRs) as
    "Losses",(SUM(Inventory_Out)*PriceRs) as "Revenue",((SUM(
   Inventory_Out)*PriceRs)-(SUM(Number_Damaged)*PriceRs)) as "Profits"
    FROM book_identification JOIN transaction_repository ON
   book_identification.ISBN = transaction_repository.ISBN where
   book_identification.ISBN= "978-81-8190-033-3" and
   transaction_repository.ISBN="978-81-8190-033-3"and Dates>=
   "17-12-2020" and Dates<="18-12-2020"
```

The SUM keyword essentially totals the values of one particular column. This totaling can be further specified by adding a WHERE keyword to make sure only the values from certain rows in that field are added up. In this context, a calculated field “Revenue” is created by adding the Inventory In values where the ISBN is equal to 978-91-8190-033-3 and multiply it by its respective price from book_identification table

```
10 --Example query for price based general report--
11 Select transaction_repository.ISBN,(SUM(Number_Damaged)*PriceRs) as
    "Losses",(SUM(Inventory_Out)*PriceRs) as "Revenue",((SUM(
   Inventory_Out)*PriceRs)-(SUM(Number_Damaged)*PriceRs)) as "Profits"
    from book_identification JOIN transaction_repository ON
   book_identification.ISBN = transaction_repository.ISBN where
   book_identification.ISBN=transaction_repository.ISBN and Dates>=
   "17-12-2020" and Dates<="18-12-2020" group by
   transaction_repository.ISBN ;
```

The GROUP BY keyword is used when the reports contain several ISBN, namely for any general report. When multiple ISBN’s are present, it is important to specify what field acts as the unique return value. GROUP BY enables a unique for each ISBN and hence the data is grouped according to the ISBN of the title

4. Data Validation

Since the software records data regarding real life objects (titles in this case), data validation needs to be applied so that the returned values in reports or searches are logical (to ensure number of books shipped out isn't greater than total number of books available). The code below, from transactionSearch( ), checks for a logical error of Inventory Out being greater than the Stock.

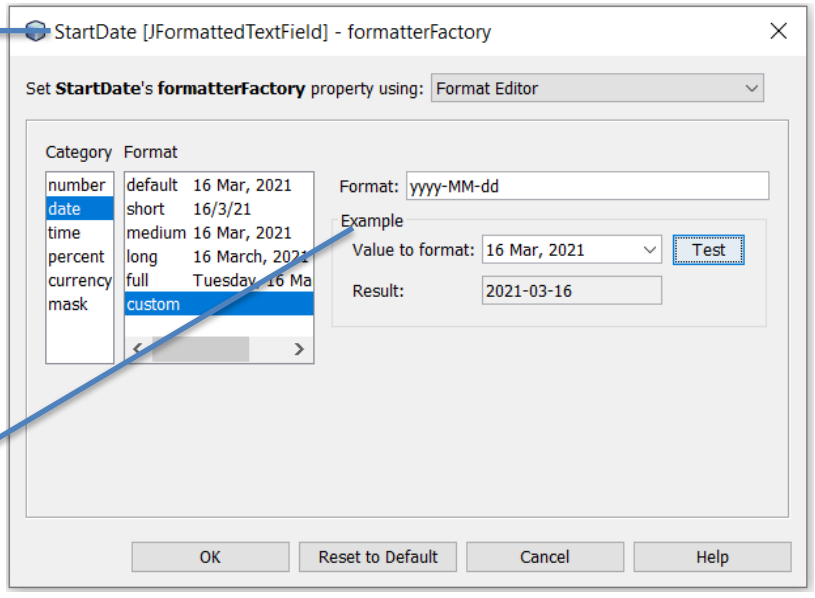
```
//The below code is used to check whether any of the text field or combo boxes are empty
if(InvenOut.getText().trim().isEmpty()||BookChoice.getSelectedItem().toString()=="Select")
{
    JOptionPane.showMessageDialog(null,"Please ensure all fields are filled");
    InvenOut.setText("");
}
//The if condition below is to validate the entered data by making sure that
//the number of books shipped out is not more than the stock in the warehouse
else if (Integer.parseInt(InvenOut.getText())>Integer.parseInt(DisplayStock.getText()))
{
    JOptionPane.showMessageDialog(null,"Number of books out cannot be greater than stock of book");
    InvenOut.setText("");
}
```

The validation for data types was not needed as JFormattedTextFields (a Java SWING feature) was used to do so. They were used to make sure the data entered matched the date format (yyyy-MM-dd) or was an integer

```
StartDate = new javax.swing.JFormattedTextField();
EndDate = new javax.swing.JFormattedTextField();
InvenIn = new javax.swing.JFormattedTextField();
NumDmg = new javax.swing.JFormattedTextField();
InvenOut = new javax.swing.JFormattedTextField();
DateSearch = new javax.swing.JFormattedTextField();
```

First on the left, there are a number of textfields which are formatted automatically by NetBeans. InvenIn, NumDmg, and InvenOut are formatted to always be an integer, otherwise the entered value is removed from the textbox and it is made empty. Start Date, EndDate, and DateSearch are formatted so that any date entered is converted into the format yyyy-MM-dd.

The properties, or namely the formatterFactory, on the right is for the variable StartDate. You can see the custom setting for yyyy-MM-dd.



A test value is also given there. 16 Mar, 2021 gets automatically converted into 2021-03-16- reducing the amount of code needed for validation

5. Exception Handling

While running long lengths of code, especially those interacting with an external database, exceptions could occur. To catch any errors, "try-catch" blocks were used anywhere parameter passing or MySQL queries were used to display the relevant error. The try-catch block from the function transactionSearch( ) is used below to identify SQL errors

```
try
{
    Class.forName("com.mysql.cj.jdbc.Driver");
    Connection Conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/cs_ia_data","root","xeno28972");
    //When both search paramaters are empty the entire list of transactions is returned
    if (BookChoice.getSelectedItem().toString()=="Select" && DateSearch.getText().trim().isEmpty() )
    {
        String sql= "Select * from transaction_repository";
        PreparedStatement pst= Conn.prepareStatement(sql);
        ResultSet rs = pst.executeQuery();
        Display_Books.setModel(DbUtils.resultSetToTableModel(rs));
    }
    //When one of the search paramaters are entered, the list of transactions that match the date or book name is returned
    else if (BookChoice.getSelectedItem().toString()=="Select" && DateSearch.getText().trim().isEmpty()==false )
    {
        String sql= "Select * from transaction_repository where Dates=?";
        PreparedStatement pst= Conn.prepareStatement(sql);
        pst.setString(1,DateSearch.getText());
        ResultSet rs = pst.executeQuery();
        Display_Books.setModel(DbUtils.resultSetToTableModel(rs));
    }
    else if (BookChoice.getSelectedItem().toString()!="Select" && DateSearch.getText().trim().isEmpty()==true )
    {
        String sql= "Select * from transaction_repository where ISBN=?";
        PreparedStatement pst= Conn.prepareStatement(sql);
        pst.setString(1,DisplayIsbn.getText());
        ResultSet rs = pst.executeQuery();
        Display_Books.setModel(DbUtils.resultSetToTableModel(rs));
    }
    //When both the search paramaters are entered, the list of transactions that match the date and book name is returned
    else if (BookChoice.getSelectedItem().toString()!="Select" && DateSearch.getText().trim().isEmpty()==false )
    {
        String sql= "Select * from transaction_repository where ISBN=? and Dates=?";
        PreparedStatement pst= Conn.prepareStatement(sql);
        pst.setString(1,DisplayIsbn.getText());
        pst.setString(2,DateSearch.getText());
        ResultSet rs = pst.executeQuery();
        Display_Books.setModel(DbUtils.resultSetToTableModel(rs));
    }
    Conn.close();
    }
catch(Exception e)
{
    JOptionPane.showMessageDialog(null,e);
}
```

Exception e will catch the error and the JOptionPane shows this error to the user / code if none of the if conditions are satisfied or if the MySQL syntax leads to an error. This prompts the user to renter data

6. Parameter Passing and Global Variables

While creating a report, the user enters data into one of the input forms, namely "CreateReport". The values entered, after validation, are passed on to the report layout themselves. To do so, parameter passing is applied, creating dynamic reports. Parameter passing involves passing input parameters into a module[8]. The entered start date, end date, ISBN, report author, and comments are transferred to and displayed in the report layout by using global variables. The code below shows this functionality.[9]

[8] https://www.sciencedirect.com/science/article/pii/B9780340700143500517

[9] Referred video: https://www.youtube.com/watch?v=QNTLJc4MaD0

```
//Checking whether the user wants a price based specific report
else if (ReportChoice.getSelectedItem().toString()=="Price-Based" && BookChoice.getSelectedItem().toString()!="General")
{
    //Creating object to help pass variable values to the next JFrame
    PriceBasedSpecificReport pbsr= new PriceBasedSpecificReport();
    //Passing the name, start date, comments, and ISBN to the report layout
    PriceBasedSpecificReport.Name.setText(CreateReports.NameChoice.getSelectedItem().toString());
    PriceBasedSpecificReport.DisplayStartDate.setText(CreateReports.StartDate.getText());
    PriceBasedSpecificReport.DisplayEndDate.setText(CreateReports.EndDate.getText());
    PriceBasedSpecificReport.DisplayComments.setText(CreateReports.Comments.getText());
    PriceBasedSpecificReport.IsbnDisplay.setText(CreateReports.DisplayIsbn.getText());
    //Making the report visible
    pbsr.setVisible(true);
    setVisible(false);
}
```

Details are passed from "Create Reports" to "PriceBasedSpecificReport" by using parameter passing

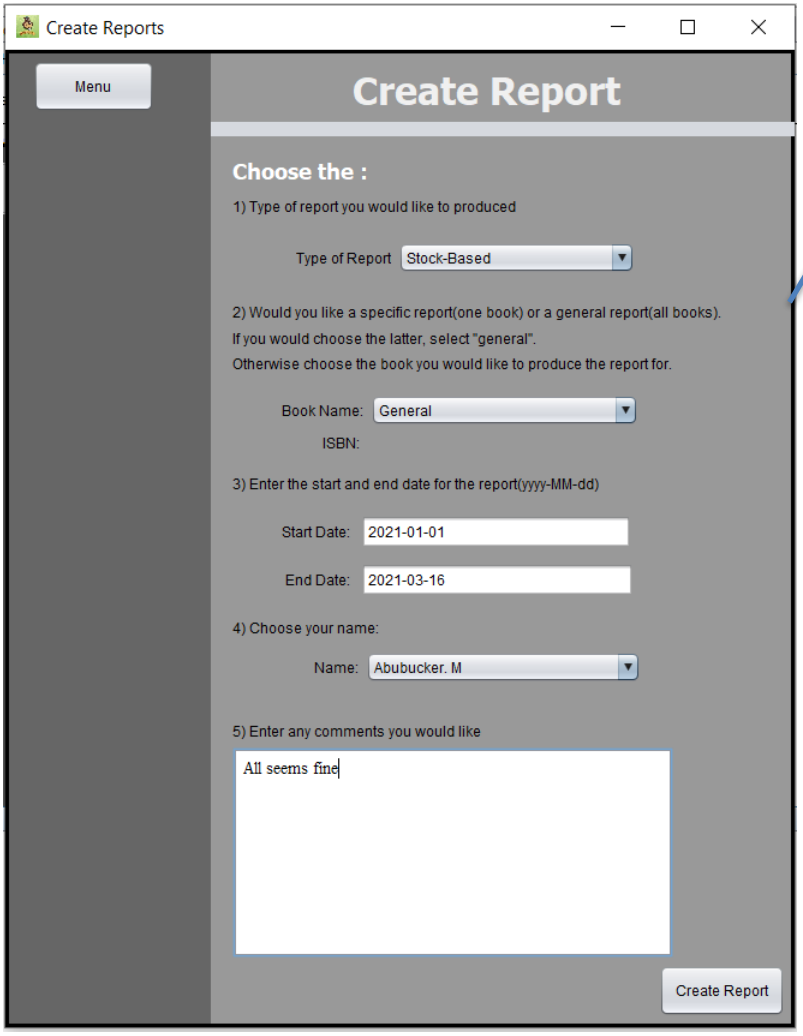
To emulate a global variable in Java, the access modifiers "public" and "static" are used. Applying the modifier "public" means that the variable is visible and can be called from other objects of other types. This means that the method is associated with the class, not a specific instance (object) of that class.[10]

Variable declaration code

```
public static javax.swing.JFormattedTextField StartDate;
```

Variable: field Access: public ☐ final ☑ static ☐ transient ☐ volatile

Both public and static checked in variable declaration

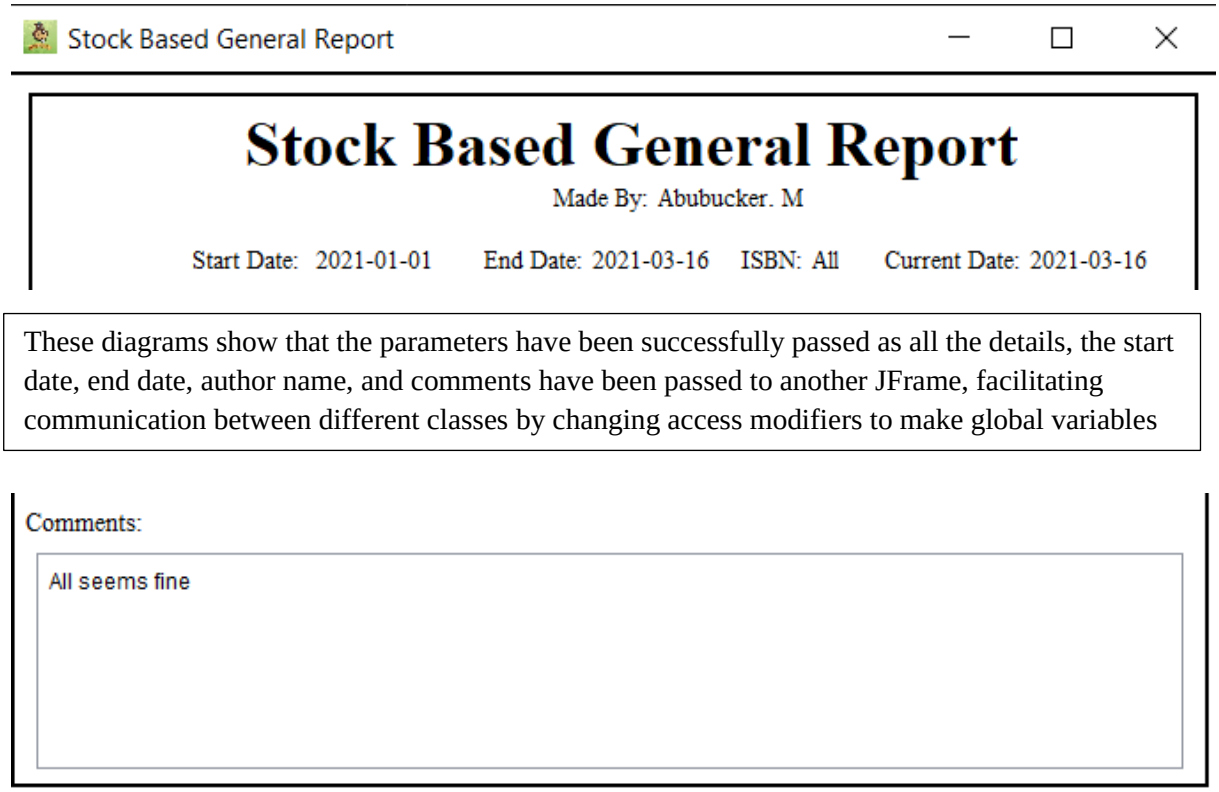


For example

- The type of report chosen is stock based
- The scope of the report chosen is General
- The start date entered is 2021-01-01
- The end date entered is 2021-03-16
- The author of the report chosen is Mr.XX
- Finally, the report author can enter relevant comments for future reference

Since the data entered across all the fields is valid, when the create report button is clicked, the user will be led to the stock based general report page

[10] https://stackoverflow.com/questions/2390063/what-does-public-static-void-mean-in-java



These diagrams show that the parameters have been successfully passed as all the details, the start date, end date, author name, and comments have been passed to another JFrame, facilitating communication between different classes by changing access modifiers to make global variables

7. Generating Reports

As per success criteria number 9 and 11, four different types of report need to be made: stock based and specific, stock based and general, price based and specific, price based and general. A report is a printable and saveable PDF layout. Specific information and evidence are presented, analysed and applied to a particular problem or issue[11]. A general report should display the data for all books and a specific report should do so for any one book within the date range entered. Stock based will return the number damaged, inventory in, inventory out, stock flow, and percentage damaged. Price based will return losses, revenue and profit.

The code above used this package[12]

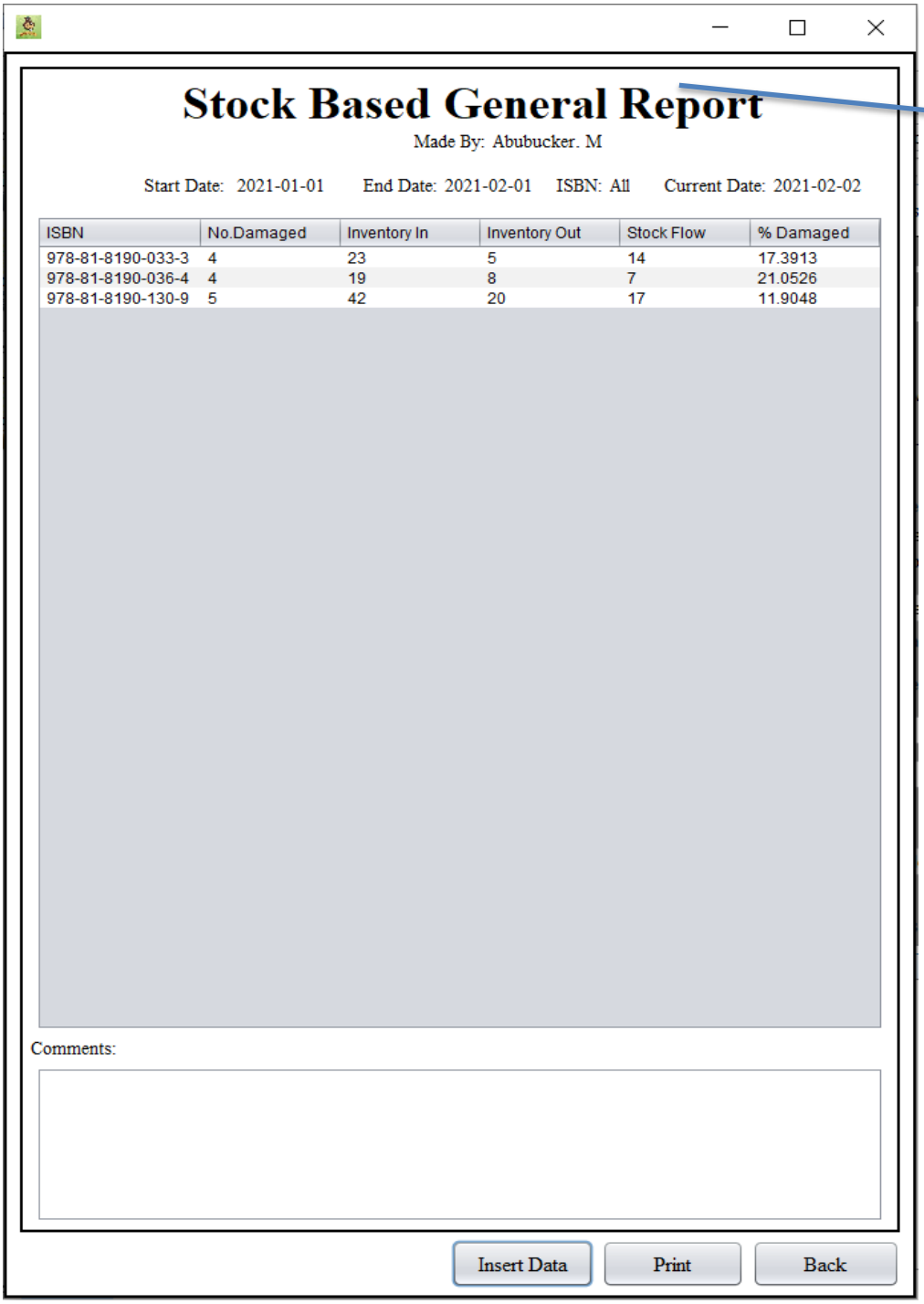
```
//Printer Instance is created when the "Print" button is clicked
PrinterJob job = PrinterJob.getPrinterJob();
job.setJobName(" Print Data ");
//The characteristics of the printable sheet are determined
job.setPrintable (new Printable()
{
    public int print(Graphics pg, PageFormat pf, int pageNum)
    {
        pf.setOrientation(PageFormat.LANDSCAPE);
        if (pageNum > 0)
        {
            return Printable.NO_SUCH_PAGE;
        }
        Graphics2D g2 = (Graphics2D) pg;
        g2.translate(pf.getImageableX(), pf.getImageableY());
        //Scaling of Report_Print Jframe to the A4 sheet size
        g2.scale(0.92641,0.85628);
        Report_Print.paint(g2);
        return Printable.PAGE_EXISTS;
    }
});
```

```
//Checking whether the button "OK" is pressed in the print dialogue
boolean ok= job.printDialog();
if(ok)
{
    try
    {
        //Document is printed
        job.print();
    }
    catch (PrinterException ex)
    {

    }
}
```

[11] https://www.dictionary.com/browse/report

[12]https://github.com/Parveshdhull/Right2Trick/blob/master/YouTube/NetBeans/JFrame%20printing%20code.txt

Figures below shows examples of report layout for all types of reports



The JPanel inside the margins is what will be printed. The panel's variable name is Report_Print.

Before calling the JPanel, the scale to which it will be expanded to print is first determined

Since the ratio of an A4 size sheet is around $1:\sqrt{2}$, the pixel size of the Report_Print Jpanel followed it. To be specific it had the pixel size of 660, 924 (x, y respectively).

Once this was done, the report could be scaled up to an A4 sheet PDF/ Print by calling the function scale ( ) from the class Jframes, Graphics 2D.

This allowed the software to create a printable pdf. The same was done for the following

Price Based General Report

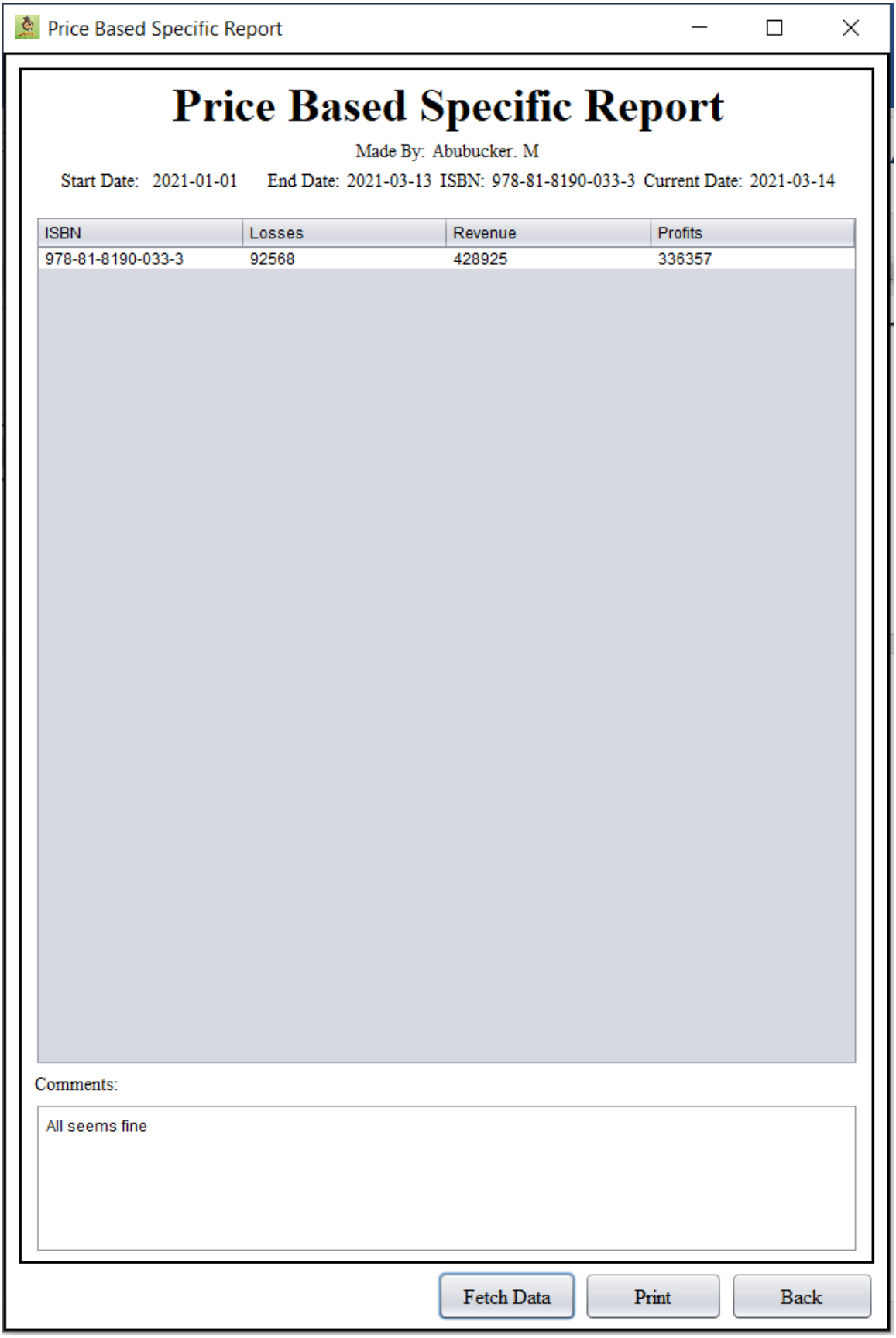
Made By: Abubucker. M

Start Date: 2021-01-01 End Date: 2021-02-01 ISBN: All Current Date: 2021-02-02

ISBN	Losses	Revenue	Profits
978-81-8190-033-3	1596	1995	399
978-81-8190-036-4	1592	3184	1592
978-81-8190-130-9	1750	7000	5250

Comments:

Insert Data Print Back


Price Based Specific Report
Price Based Specific Report
Made By: Abubucker. M
Start Date: 2021-01-01 End Date: 2021-03-13 ISBN: 978-81-8190-033-3 Current Date: 2021-03-14
ISBN
Losses
Revenue
Profits
978-81-8190-033-3
92568
428925
336357
Comments:
All seems fine
Fetch Data
Print
Back

8. Library Functions

A library provides a set of helper functions/objects/modules which your application code calls for specific functionality[13]. Other than the standard Java libraries in the Java Development Kit (JDK), rx2sml[14], mysql-connector[15], and Java SWING[16] libraries were used. The rx2sml and mysql-connector libraries were used to perform MySQL operations. The Java SWING library was used to create the UI

```
//Examples of JDK libraries used
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.print.PageFormat;
import java.awt.print.Printable;
import java.awt.print.PrinterException;
import java.awt.print.PrinterJob;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
```

Used for facilities such as data validation, formatting data entry, and printing of reports

```
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;
```

```
// Java SWING components used to create UI
    // Variables declaration - do not modify
    private javax.swing.JLabel DateDisplay;
    public static javax.swing.JTextArea DisplayComments;
    public static javax.swing.JLabel DisplayEndDate;
    public static javax.swing.JLabel DisplayStartDate;
    private javax.swing.JTable Display_Report;
    public static javax.swing.JLabel IsbnDisplay;
    public static javax.swing.JLabel Name;
    private javax.swing.JButton PrintButton;
    private javax.swing.JPanel Report_Print;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton4;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel10;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JScrollPane jScrollPanel;
    private javax.swing.JScrollPane jScrollPane2;
```

Used across the entire software to mainly create the GUI components of each JFrame.

It is integrated in all the functionalities as JTextFields and JComboboxes are the methods of data entry in this software which are both facilitated by JavaSWING methods

JavaSWING methods are also used to display data to the user in the form of JTables, JOptionPanes, and JLabels, allowing for a more understandable interface.

```
//Examples of Java-sql libraries use
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
```

Since MySQL is an external software, the only way to connect to it and interact with it was using the MySQL libraries and classes such as PreparedStatement and Resultset.

Word Count: 997

[13] https://www.geeksforgeeks.org/software-framework-vs-library/
[14] https://stackoverflow.com/questions/27679867/jtable-how-to-use-rs2xml
[15] https://dev.mysql.com/downloads/connector/j/8.0.html
[16] https://www.geeksforgeeks.org/java-swing-simple-calculator/