Digital Library management

Criteria C – Development

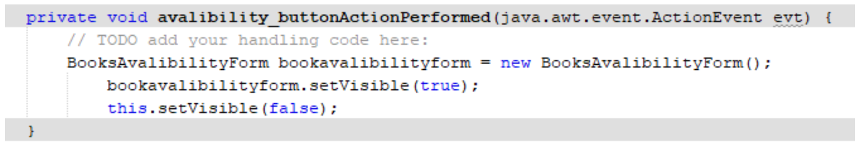
List of techniques

1. GUI Programming
   1.1. Navigation
   1.2. Jtables
   1.3. Forms
2. Object-Oriented Programming
   2.1. Polymorphism
   2.2. Encapsulation
   2.3. Inheritance
3. Class and Objects
4. Sorting and Searching
5. Exception handling with validation and prompts
6. External Libraries
   6.1. Jasper report
   6.2. Jcalender
   6.3. JfreeChart
   6.4. PostgresSQL
7. SQL Database
   7.1. Database Connectivity
   7.2. Database Structure
   7.3. SQL Commands and Queries
8. Data Structure
9. Nested Loops and Complex Conditional Statements
10. Dynamic Chart Generation
11. Report Generation

1) GUI Programming

1.1) Navigation

To help client navigate between different screens, JButton is used in the dashboard form to connect the client to different panels. Also, a BACK button is provided to help the client navigate back to the main screen.



```
private void avalibility_buttonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    BooksAvalibilityForm bookavalibilityform = new BooksAvalibilityForm();
        bookavalibilityform.setVisible(true);
        this.setVisible(false);
}
```

Figure 1 – The code used for using JButtons to navigate between the screen and example here is button to go to the book availability form

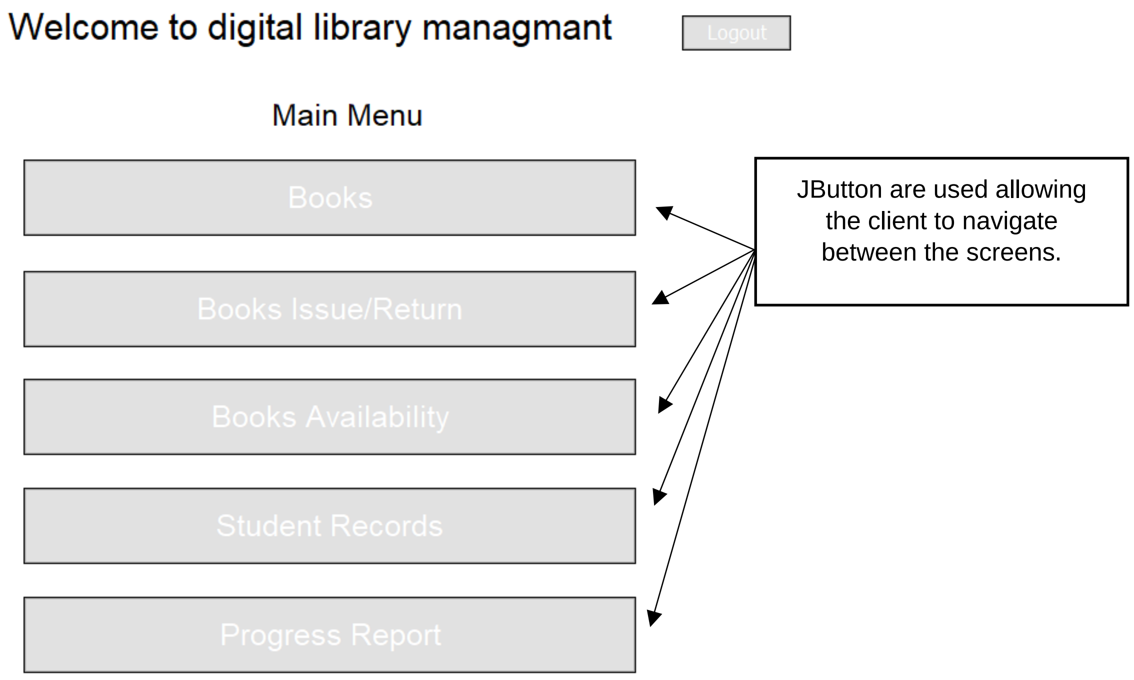
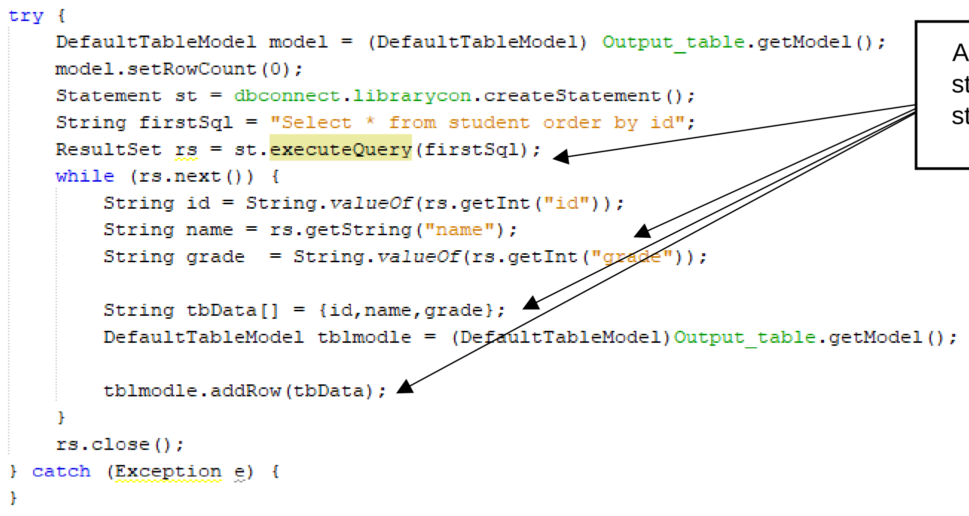

Welcome to digital library managmant
Logout
Main Menu
Books
Books Issue/Return
Books Availability
Student Records
Progress Report
JButton are used allowing the client to navigate between the screens.

Figure 2 – The dashboard form with Jbutton to navigate to other windows

1.2) **JTable**[1]

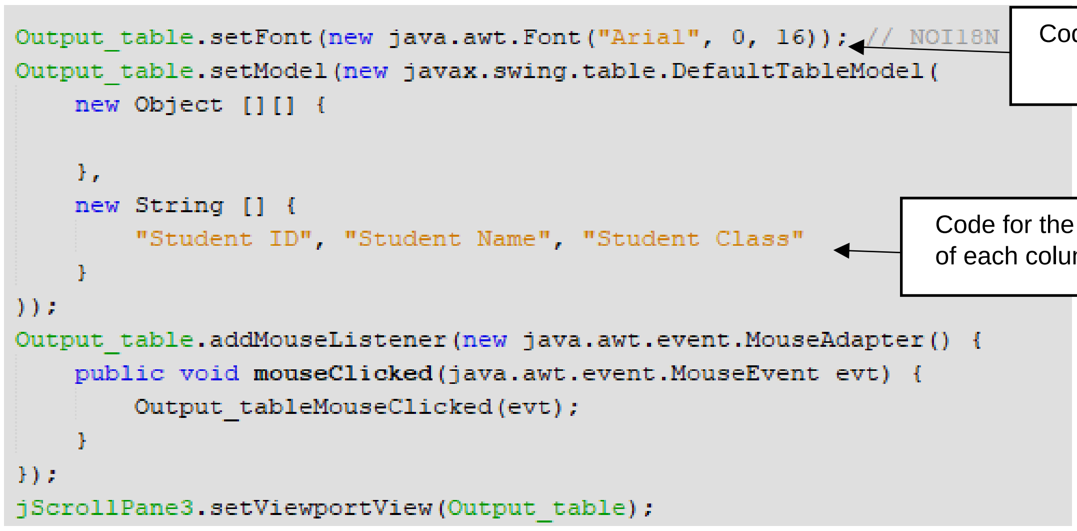
JTable is used to store and display information to the client about all the information from the database after certain events are executed. The JTables in the code is used at various places but here the example taken is of student form where the JTable is being used.



```
try {
    DefaultTableModel model = (DefaultTableModel) Output_table.getModel();
    model.setRowCount(0);
    Statement st = dbconnect.librarycon.createStatement();
    String firstSql = "Select * from student order by id";
    ResultSet rs = st.executeQuery(firstSql);
    while (rs.next()) {
        String id = String.valueOf(rs.getInt("id"));
        String name = rs.getString("name");
        String grade  = String.valueOf(rs.getInt("grade"));

        String tbData[] = {id,name,grade};
        DefaultTableModel tblmodle = (DefaultTableModel)Output_table.getModel();

        tblmodle.addRow(tbData);
    }
    rs.close();
} catch (Exception e) {
}
```

All fields are input from the student form and stored as string values into the table.

Figure 3 – Code for loading the values into the JTable from the database



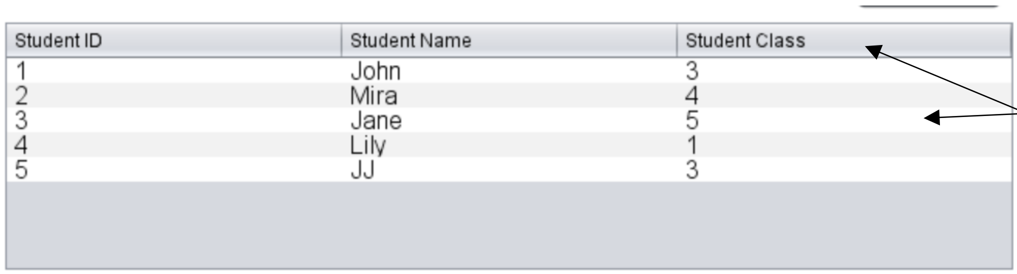
```
Output_table.setFont(new java.awt.Font("Arial", 0, 16)); // NOI18N
Output_table.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {

    },
    new String [] {
        "Student ID", "Student Name", "Student Class"
    }
));
Output_table.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        Output_tableMouseClicked(evt);
    }
});
jScrollPane3.setViewportView(Output_table);
```

Code for the propertities of the table

Code for the title is on top of each column in JTable.

Figure 4 – Code for the properties, title and the setting of the JTable

[1] https://www.youtube.com/watch?v=CQMpXGwHeYQ

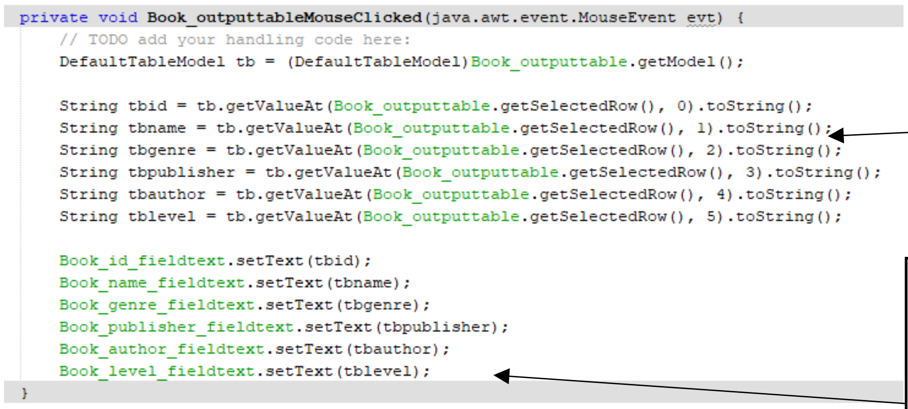


Student ID	Student Name	Student Class
1	John	3
2	Mira	4
3	Jane	5
4	Lily	1
5	JJ	3

We can see the title and the properties set in figure 4 and the output is like this.

Figure 5 – Output of JTable in the student form for depicting the students who are part of the library program

JTable is used to auto-populate book details on the screen on click of the particular book entry as stored in the database

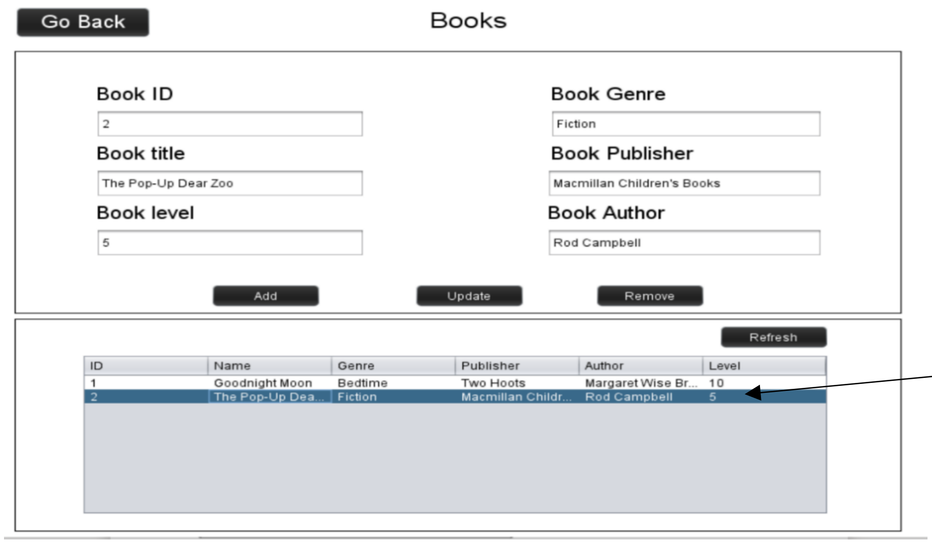


```
private void Book_outputtableMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    DefaultTableModel tb = (DefaultTableModel)Book_outputtable.getModel();

    String tbid = tb.getValueAt(Book_outputtable.getSelectedRow(), 0).toString();
    String tbname = tb.getValueAt(Book_outputtable.getSelectedRow(), 1).toString();
    String tbgenre = tb.getValueAt(Book_outputtable.getSelectedRow(), 2).toString();
    String tbpublisher = tb.getValueAt(Book_outputtable.getSelectedRow(), 3).toString();
    String tbauthor = tb.getValueAt(Book_outputtable.getSelectedRow(), 4).toString();
    String tblevel = tb.getValueAt(Book_outputtable.getSelectedRow(), 5).toString();

    Book_id_fieldtext.setText(tbid);
    Book_name_fieldtext.setText(tbname);
    Book_genre_fieldtext.setText(tbgenre);
    Book_publisher_fieldtext.setText(tbpublisher);
    Book_author_fieldtext.setText(tbauthor);
    Book_level_fieldtext.setText(tblevel);
}
```

These statements are used to retrieve values when a specific row is clicked.

Now the values which are received before is now added into the specific textbox

Figure 6 – OnClick Mouse event used when the user clicks on a row of the JTable then the values will automatically be loaded into the text fields in the student form section for updating/deleting the



Book ID	2
Book title	The Pop-Up Dear Zoo
Book level	5
Book Genre	Fiction
Book Publisher	Macmillan Children's Books
Book Author	Rod Campbell

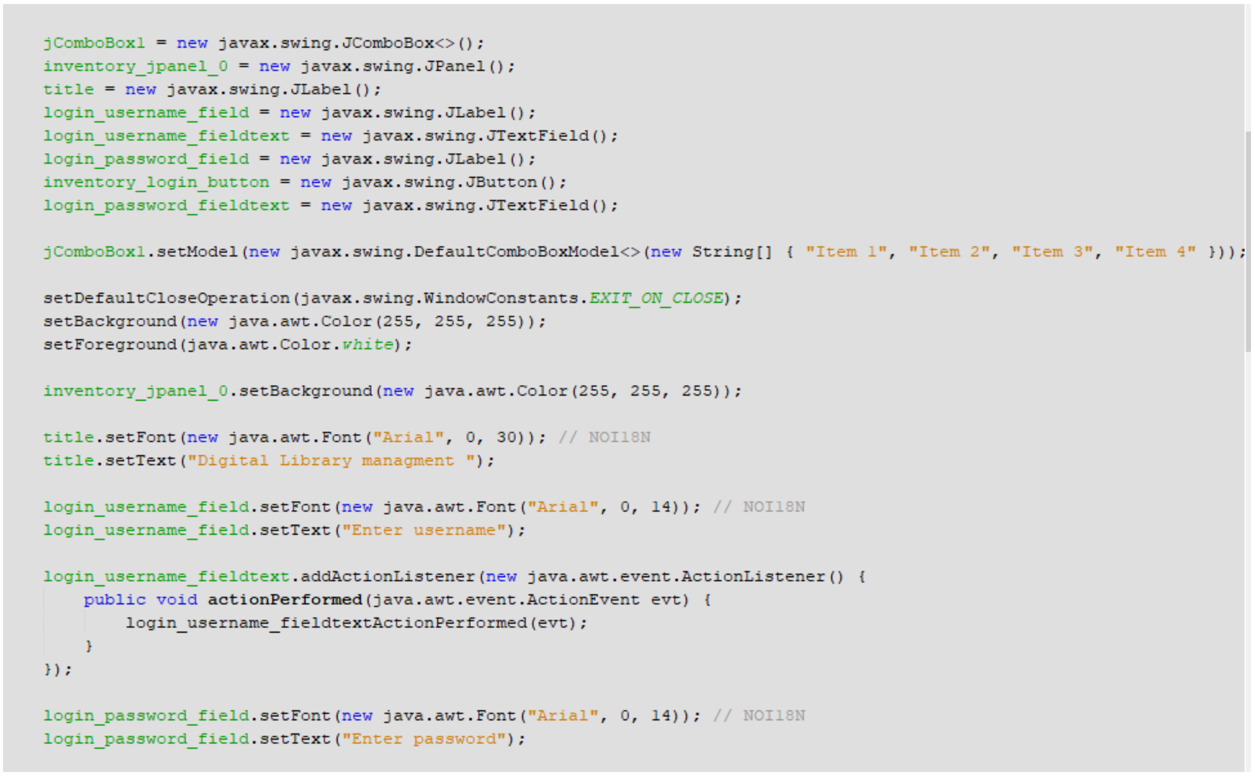
ID	Name	Genre	Publisher	Author	Level
1	Goodnight Moon	Bedtime	Two Hoots	Margaret Wise Br...	10
2	The Pop-Up Dea...	Fiction	Macmillan Childr...	Rod Campbell	5

The blue color depicts that this row of the JTable has been selected.

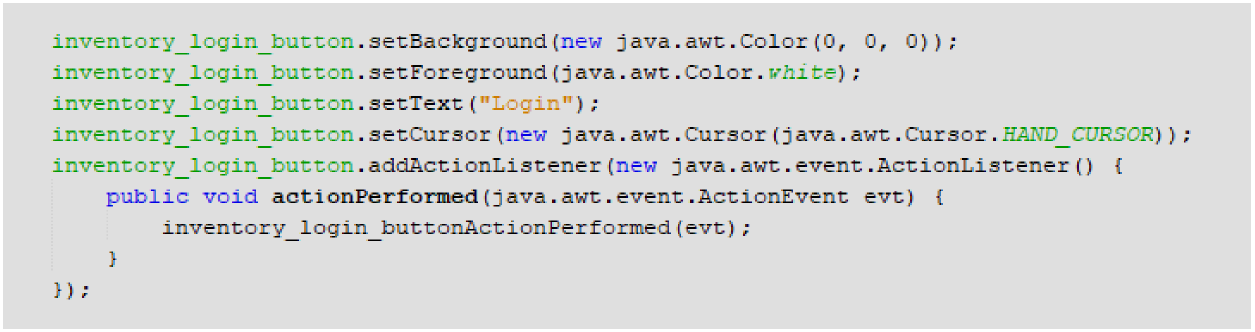
Figure 7 – Output when a book is chosen then its value is automatically loaded in the textbox.

1.3) JForms

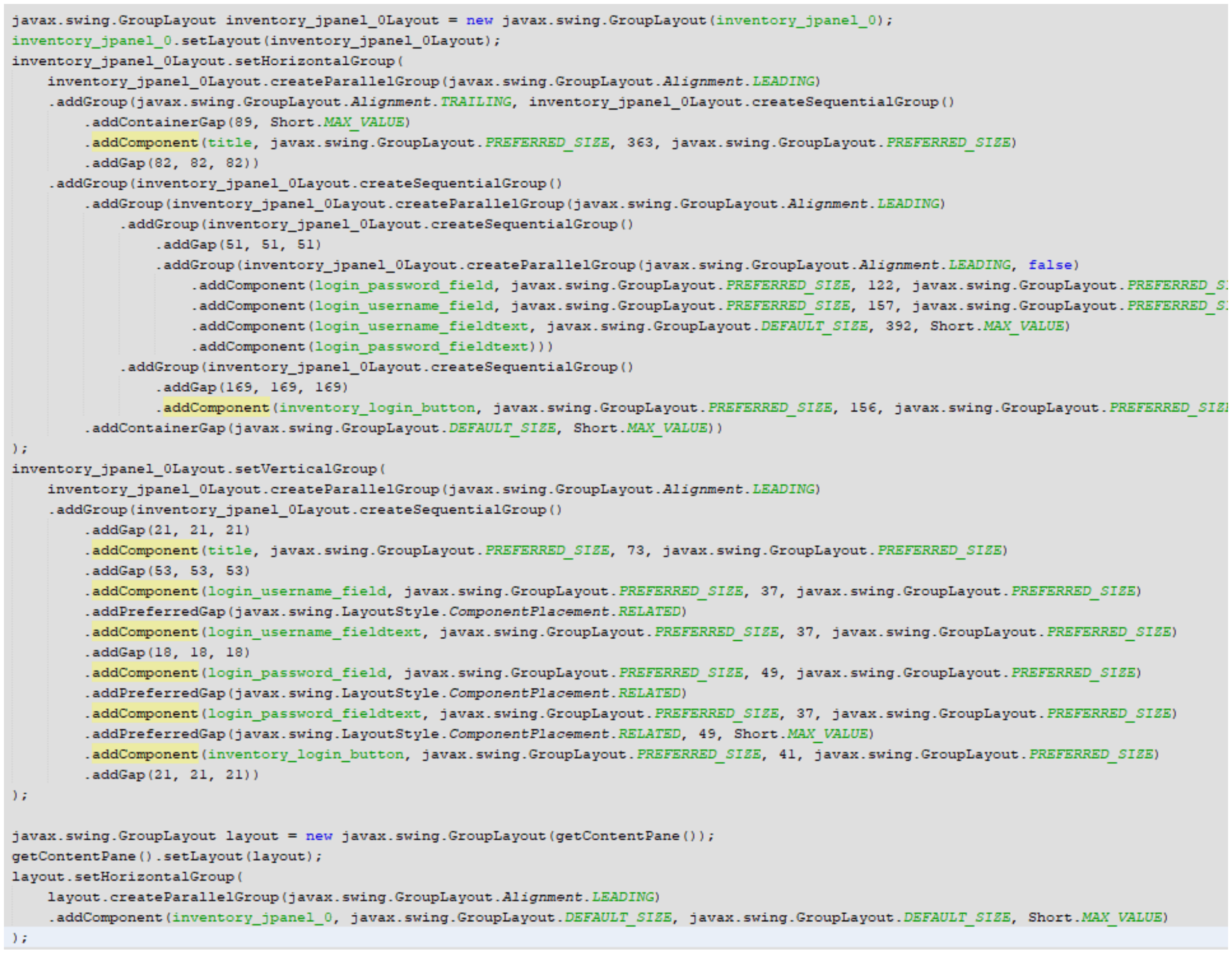
JForms are used to make the GUI of this desktop app. The JForms and their functionalities are used everywhere. To depict the functionality of the JForm, I will take login form as an example.



```
jComboBox1 = new javax.swing.JComboBox<>();
inventory_jpanel_0 = new javax.swing.JPanel();
title = new javax.swing.JLabel();
login_username_field = new javax.swing.JLabel();
login_username_fieldtext = new javax.swing.JTextField();
login_password_field = new javax.swing.JLabel();
inventory_login_button = new javax.swing.JButton();
login_password_fieldtext = new javax.swing.JTextField();

jComboBox1.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "Item 1", "Item 2", "Item 3", "Item 4" }));

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setBackground(new java.awt.Color(255, 255, 255));
setForeground(java.awt.Color.white);

inventory_jpanel_0.setBackground(new java.awt.Color(255, 255, 255));

title.setFont(new java.awt.Font("Arial", 0, 30)); // NOI18N
title.setText("Digital Library managment ");

login_username_field.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
login_username_field.setText("Enter username");

login_username_fieldtext.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        login_username_fieldtextActionPerformed(evt);
    }
});

login_password_field.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N
login_password_field.setText("Enter password");
```



```
inventory_login_button.setBackground(new java.awt.Color(0, 0, 0));
inventory_login_button.setForeground(java.awt.Color.white);
inventory_login_button.setText("Login");
inventory_login_button.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
inventory_login_button.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        inventory_login_buttonActionPerformed(evt);
    }
});
```



```
javax.swing.GroupLayout inventory_jpanel_0Layout = new javax.swing.GroupLayout(inventory_jpanel_0);
inventory_jpanel_0.setLayout(inventory_jpanel_0Layout);
inventory_jpanel_0Layout.setHorizontalGroup(
    inventory_jpanel_0Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, inventory_jpanel_0Layout.createSequentialGroup()
        .addContainerGap(89, Short.MAX_VALUE)
        .addComponent(title, javax.swing.GroupLayout.PREFERRED_SIZE, 363, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(82, 82, 82))
    .addGroup(inventory_jpanel_0Layout.createSequentialGroup()
        .addGroup(inventory_jpanel_0Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(inventory_jpanel_0Layout.createSequentialGroup()
                .addGap(51, 51, 51)
                .addGroup(inventory_jpanel_0Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                    .addComponent(login_password_field, javax.swing.GroupLayout.PREFERRED_SIZE, 122, javax.swing.GroupLayout.PREFERRED_S
                    .addComponent(login_username_field, javax.swing.GroupLayout.PREFERRED_SIZE, 157, javax.swing.GroupLayout.PREFERRED_S
                    .addComponent(login_username_fieldtext, javax.swing.GroupLayout.DEFAULT_SIZE, 392, Short.MAX_VALUE)
                    .addComponent(login_password_fieldtext)))
            .addGroup(inventory_jpanel_0Layout.createSequentialGroup()
                .addGap(169, 169, 169)
                .addComponent(inventory_login_button, javax.swing.GroupLayout.PREFERRED_SIZE, 156, javax.swing.GroupLayout.PREFERRED_SIZ
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
);
inventory_jpanel_0Layout.setVerticalGroup(
    inventory_jpanel_0Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(inventory_jpanel_0Layout.createSequentialGroup()
        .addGap(21, 21, 21)
        .addComponent(title, javax.swing.GroupLayout.PREFERRED_SIZE, 73, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(53, 53, 53)
        .addComponent(login_username_field, javax.swing.GroupLayout.PREFERRED_SIZE, 37, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(login_username_fieldtext, javax.swing.GroupLayout.PREFERRED_SIZE, 37, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(18, 18, 18)
        .addComponent(login_password_field, javax.swing.GroupLayout.PREFERRED_SIZE, 49, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(login_password_fieldtext, javax.swing.GroupLayout.PREFERRED_SIZE, 37, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 49, Short.MAX_VALUE)
        .addComponent(inventory_login_button, javax.swing.GroupLayout.PREFERRED_SIZE, 41, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(21, 21, 21))
);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addComponent(inventory_jpanel_0, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
);
```

Figure 8 – Code of designing the JForm

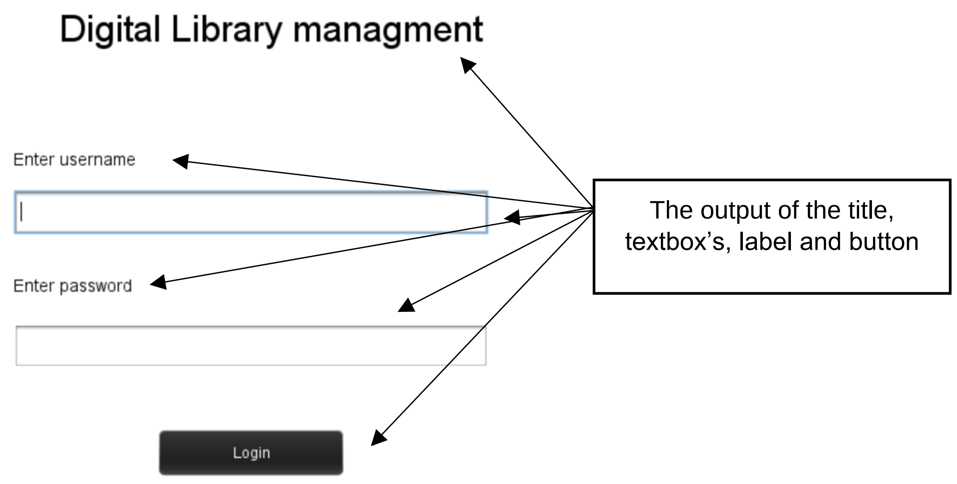
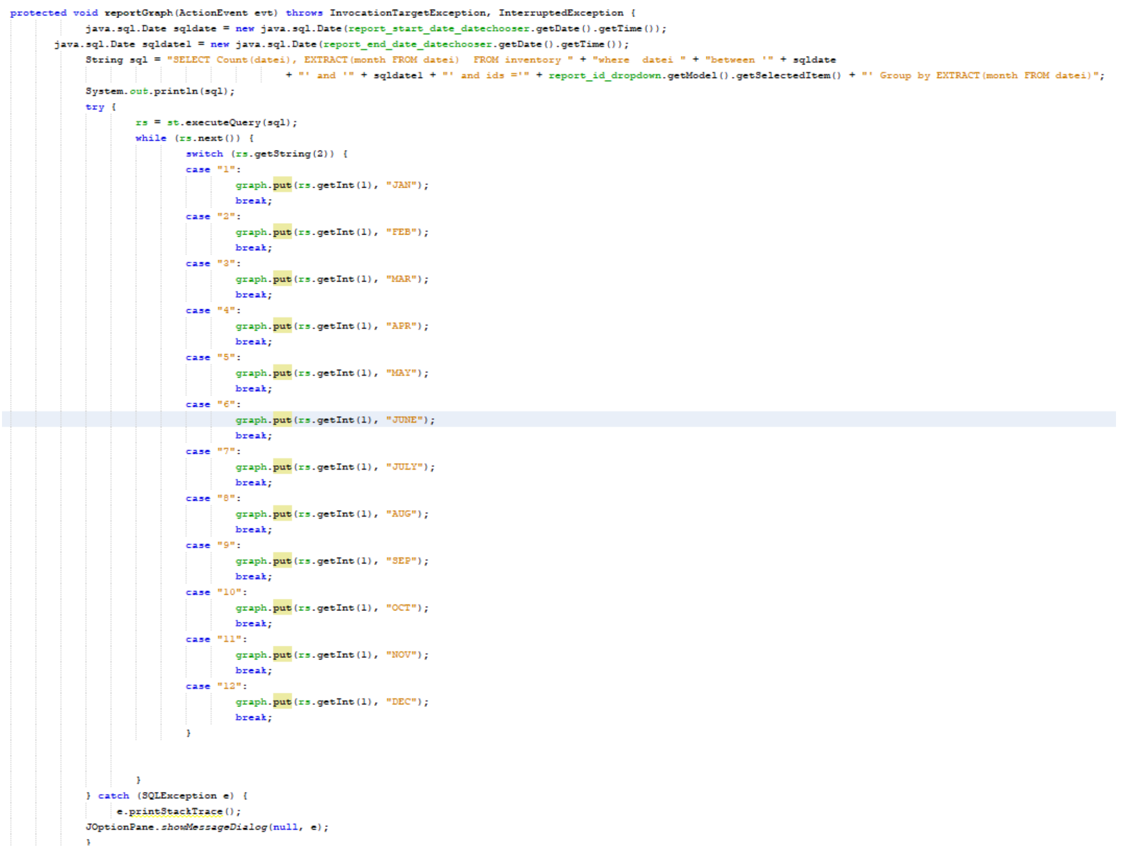

Digital Library managment
Enter username
Enter password
Login
The output of the title, textbox's, label and button

Figure 9 – The code for designing the Jforms output looks like this

2) Object-oriented programming

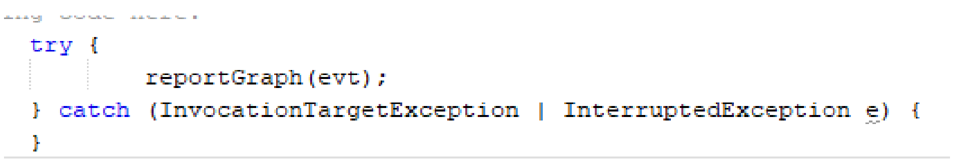
2.1) Abstraction[2]

Abstraction refers to minimum information being visible to the end-user and the complexity is hidden from the end-user. In this program, abstraction is used when a method is called to get the values/information for generating a graph but the code for this is not directly visible to the end-user. Therefore, it allows the repetitive use of code and information hiding.



```java
protected void reportGraph(ActionEvent evt) throws InvocationTargetException, InterruptedException {
            java.sql.Date sqldate = new java.sql.Date(report_start_date_datechooser.getDate().getTime());
        java.sql.Date sqldatel = new java.sql.Date(report_end_date_datechooser.getDate().getTime());
            String sql = "SELECT Count(datei), EXTRACT(month FROM datei)  FROM inventory " + "where  datei " + "between '" + sqldate
                                                + "' and '" + sqldatel + "' and ids ='" + report_id_dropdown.getModel().getSelectedItem() + "' Group by EXTRACT(month FROM datei)";
            System.out.println(sql);
            try {
                    rs = st.executeQuery(sql);
                    while (rs.next()) {
                            switch (rs.getString(2)) {
                            case "1":
                                    graph.put(rs.getInt(1), "JAN");
                                    break;
                            case "2":
                                    graph.put(rs.getInt(1), "FEB");
                                    break;
                            case "3":
                                    graph.put(rs.getInt(1), "MAR");
                                    break;
                            case "4":
                                    graph.put(rs.getInt(1), "APR");
                                    break;
                            case "5":
                                    graph.put(rs.getInt(1), "MAY");
                                    break;
                            case "6":
                                    graph.put(rs.getInt(1), "JUNE");
                                    break;
                            case "7":
                                    graph.put(rs.getInt(1), "JULY");
                                    break;
                            case "8":
                                    graph.put(rs.getInt(1), "AUG");
                                    break;
                            case "9":
                                    graph.put(rs.getInt(1), "SEP");
                                    break;
                            case "10":
                                    graph.put(rs.getInt(1), "OCT");
                                    break;
                            case "11":
                                    graph.put(rs.getInt(1), "NOV");
                                    break;
                            case "12":
                                    graph.put(rs.getInt(1), "DEC");
                                    break;
                            }


                    }
            } catch (SQLException e) {
                e.printStackTrace();
            JOptionPane.showMessageDialog(null, e);
            }
```

Figure 10 – The method is defined to get all books issued records for a particular student from the database



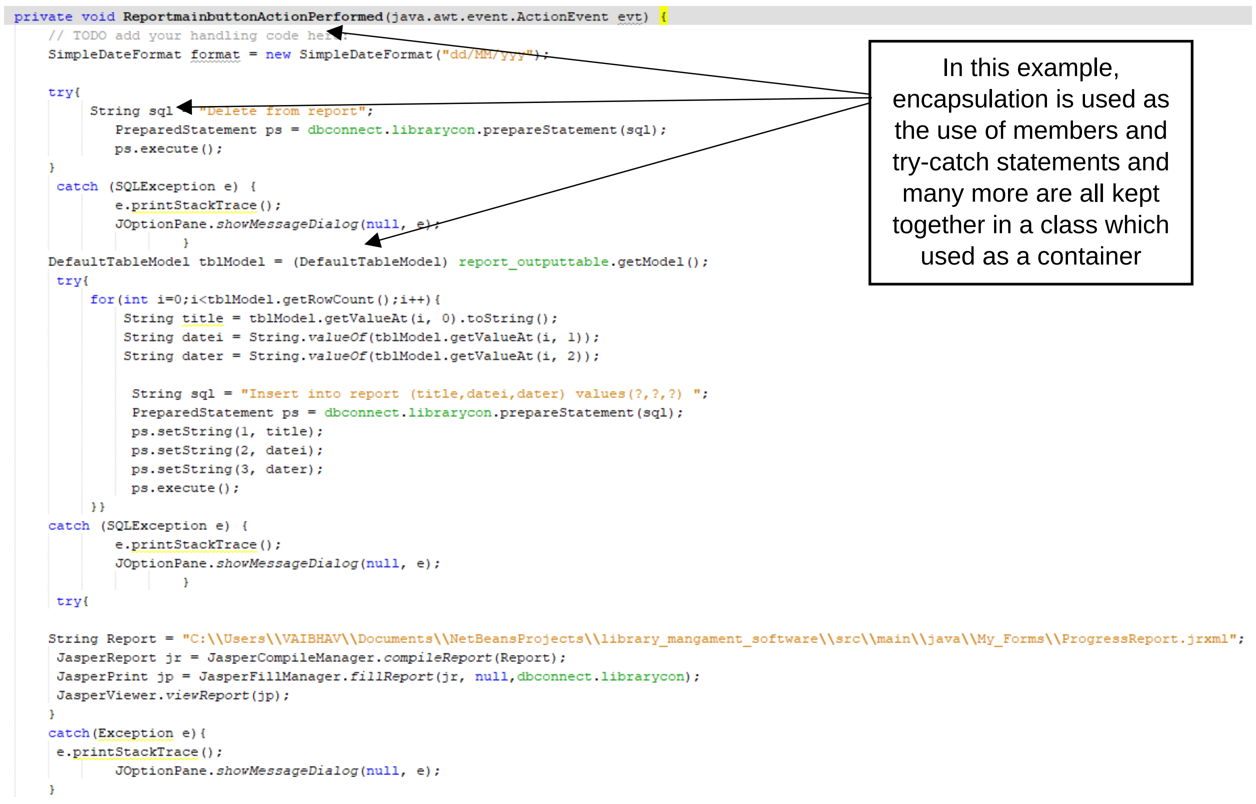
```java
try {
        reportGraph(evt);
} catch (InvocationTargetException | InterruptedException e) {
}
```

Figure 11 – Calling the method to get the information and hidden complexity

[2] https://www.tutorialspoint.com/java/java_abstraction.htm

2.2) **Encapsulation**[3]

Encapsulation is used to bind all data together to be used as a single unit. As used at various instances throughout the program. One such example of Encapsulation is below.



```java
private void ReportmainbuttonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyy");

    try{
        String sql = "Delete from report";
           PreparedStatement ps = dbconnect.librarycon.prepareStatement(sql);
           ps.execute();
    }
     catch (SQLException e) {
           e.printStackTrace();
           JOptionPane.showMessageDialog(null, e);
                   }
    DefaultTableModel tblModel = (DefaultTableModel) report_outputtable.getModel();
     try{
        for(int i=0;i<tblModel.getRowCount();i++){
            String title = tblModel.getValueAt(i, 0).toString();
            String datei = String.valueOf(tblModel.getValueAt(i, 1));
            String dater = String.valueOf(tblModel.getValueAt(i, 2));

             String sql = "Insert into report (title,datei,dater) values(?,?,?) ";
             PreparedStatement ps = dbconnect.librarycon.prepareStatement(sql);
             ps.setString(1, title);
             ps.setString(2, datei);
             ps.setString(3, dater);
             ps.execute();
        }}
    catch (SQLException e) {
           e.printStackTrace();
           JOptionPane.showMessageDialog(null, e);
                   }
     try{

    String Report = "C:\\Users\\VAIBHAV\\Documents\\NetBeansProjects\\library_mangament_software\\src\\main\\java\\My_Forms\\ProgressReport.jrxml";
     JasperReport jr = JasperCompileManager.compileReport(Report);
     JasperPrint jp = JasperFillManager.fillReport(jr, null,dbconnect.librarycon);
     JasperViewer.viewReport(jp);
    }
    catch(Exception e){
     e.printStackTrace();
           JOptionPane.showMessageDialog(null, e);
    }
```

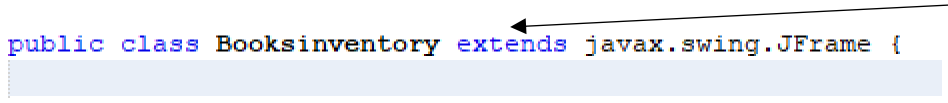
In this example, encapsulation is used as the use of members and try-catch statements and many more are all kept together in a class which used as a container

Figure 12 – Code is an example of encapsulation

[3] https://www.geeksforgeeks.org/encapsulation-in-java/

2.3) Inheritance[4]

The classes used in this program are inherited from pre-defined Java methods such as Javax.swing.JFrame.



```java
public class Booksinventory extends javax.swing.JFrame {
```

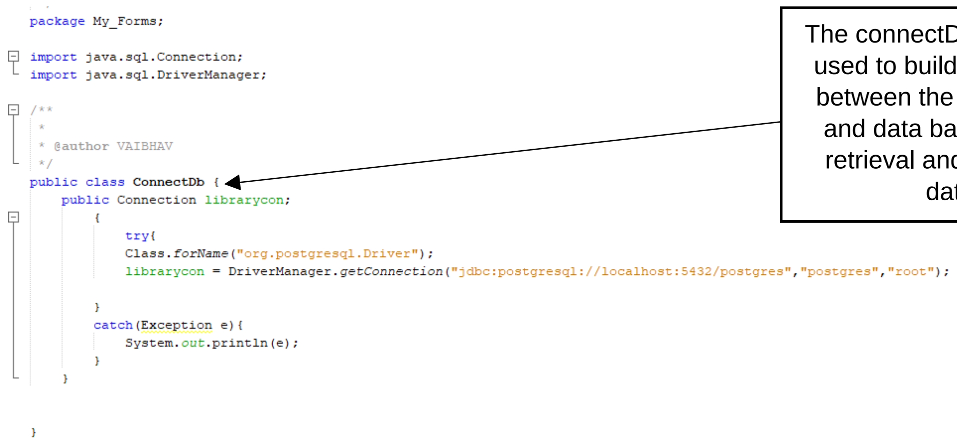
The word extend depicts that it a pre-defined method is inherited into a Java class

Figure 13 – Code is an example of inheritance

3) Class and objects

A Java program is made using a Java Class and all the information related to the class is accessed by the objects created and implemented. In this software, classes and objects are used in various places throughout the program.

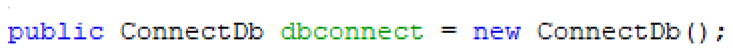
For instance, we will be taking ConnectDB as a class. The purpose of this class is to build the connection between the database and the frontend program. The object dbconnect is used to access the ConnectDB class.



```java
package My_Forms;

import java.sql.Connection;
import java.sql.DriverManager;

/**
 *
 * @author VAIBHAV
 */
public class ConnectDb {
    public Connection librarycon;
        {
            try{
            Class.forName("org.postgresql.Driver");
            librarycon = DriverManager.getConnection("jdbc:postgresql://localhost:5432/postgres","postgres","root");

        }
        catch(Exception e){
            System.out.println(e);
        }
    }


}
```

The connectDB method is used to build connection between the Java forms and data base to allow retrieval and storing of data

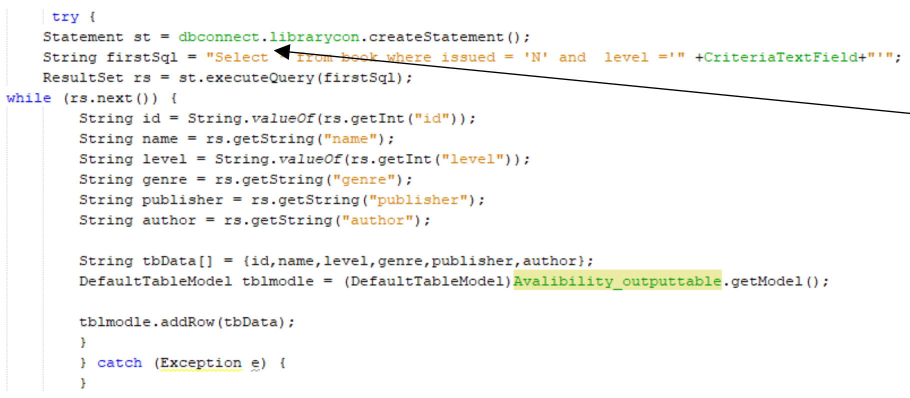
Figure 14 – Depicts the ConnectDB Class being implemented



```java
public ConnectDb dbconnect = new ConnectDb();
```

Figure 15 – Depicts the ConnectDB class being converted into dbConnect

[4] https://www.simplilearn.com/tutorials/java-tutorial/inheritence-in-java



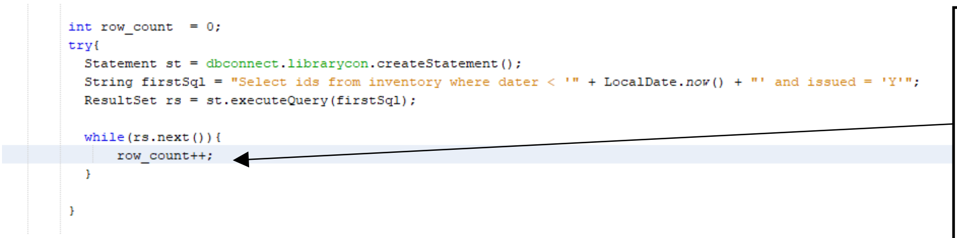
```
 try {
Statement st = dbconnect.librarycon.createStatement();
String firstSql = "Select * from book where issued = 'N' and  level ='" +CriteriaTextField+"'";
ResultSet rs = st.executeQuery(firstSql);
while (rs.next()) {
    String id = String.valueOf(rs.getInt("id"));
    String name = rs.getString("name");
    String level = String.valueOf(rs.getInt("level"));
    String genre = rs.getString("genre");
    String publisher = rs.getString("publisher");
    String author = rs.getString("author");

    String tbData[] = {id,name,level,genre,publisher,author};
    DefaultTableModel tblmodle = (DefaultTableModel)Avalibility_outputtable.getModel();

    tblmodle.addRow(tbData);
    }
    } catch (Exception e) {
    }
```

After converting the connectDB method to dbconnect. Then we can see it is used here to retrieve that method and used in this piece of code to retrieve data from the database.

Figure 16 – Depicts the implication of the dbConnect method

4) **Sorting and searching**

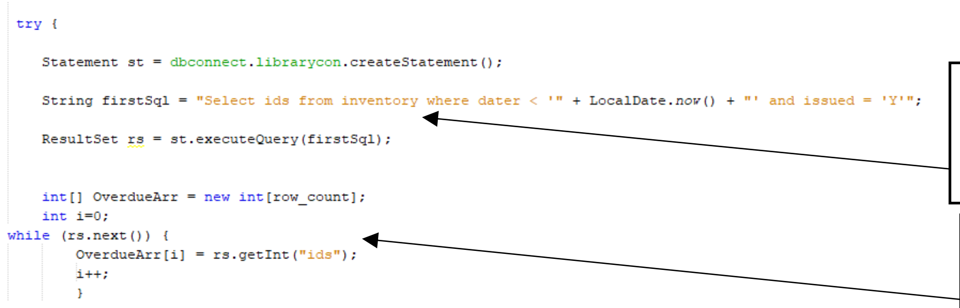
In the program, sorting and searching algorithms are used together to provide warnings and resistance to the client from issuing books to students who have overdue books already. Bubble sort[5] is used to arrange the list of student IDs of students with overdue books in ascending order. Then making use of binary search[6] to cross-check if a particular Student ID is there in the array.



```
int row_count  = 0;
try{
  Statement st = dbconnect.librarycon.createStatement();
  String firstSql = "Select ids from inventory where dater < '" + LocalDate.now() + "' and issued = 'Y'";
  ResultSet rs = st.executeQuery(firstSql);

  while(rs.next()){
      row_count++;
  }

}
```

The values from the database are being retrieved to see how many overdue books there are, and each value present means the total values count will be stored in row count variable

Figure 17 – Code for setting the size of the array using the row count.



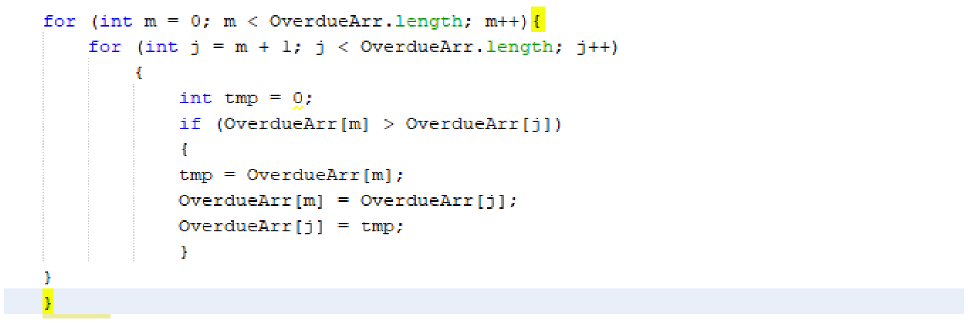
```
try {

  Statement st = dbconnect.librarycon.createStatement();

  String firstSql = "Select ids from inventory where dater < '" + LocalDate.now() + "' and issued = 'Y'";

  ResultSet rs = st.executeQuery(firstSql);


  int[] OverdueArr = new int[row_count];
  int i=0;
while (rs.next()) {
      OverdueArr[i] = rs.getInt("ids");
      i++;
      }
```

Using this the books which are overdue are being selected from the inventory database

Here all the student ID is being stored in an Array "OverdueArr". Also, the size of the array is defined by the row count set before.

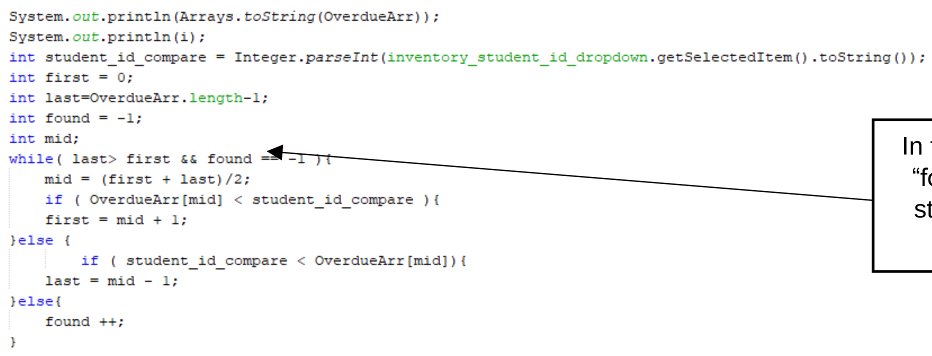
Figure 18 – Code for storing the student ID who have overdue books in the array

5 https://www.geeksforgeeks.org/bubble-sort/
6 https://www.javatpoint.com/binary-search-in-java



```java
for (int m = 0; m < OverdueArr.length; m++){
    for (int j = m + 1; j < OverdueArr.length; j++)
        {
            int tmp = 0;
            if (OverdueArr[m] > OverdueArr[j])
            {
            tmp = OverdueArr[m];
            OverdueArr[m] = OverdueArr[j];
            OverdueArr[j] = tmp;
            }
}
}
```

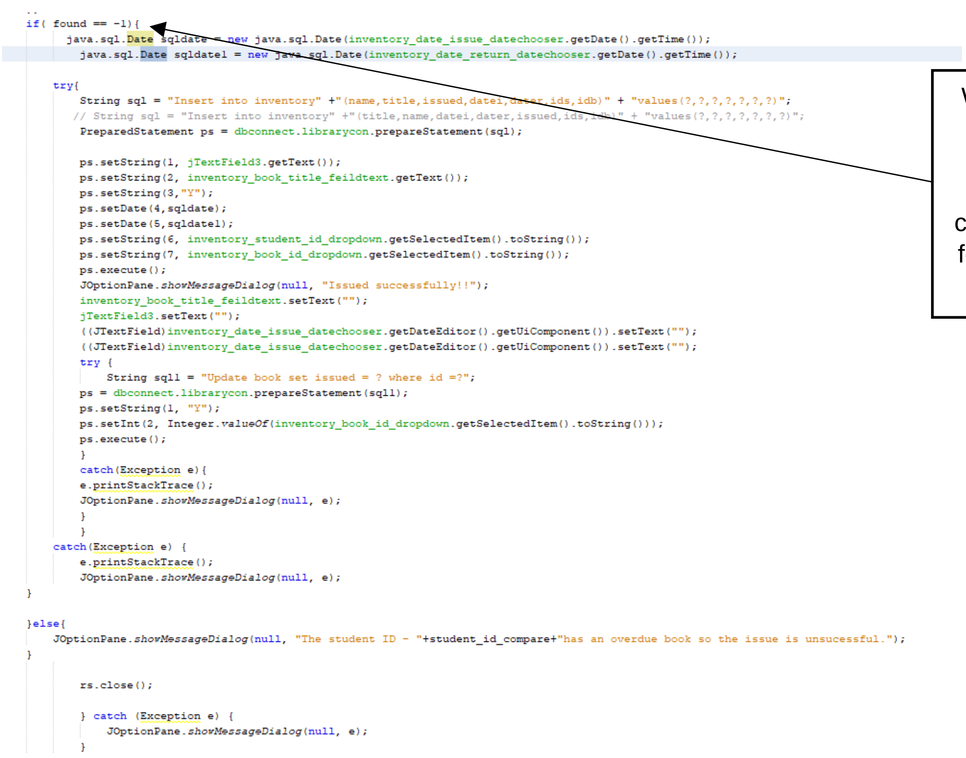
Figure 19 – Code for Bubble sort to arrange the student ID in the array in an ascending order. This is done since in binary search requires a sorted array.



```java
System.out.println(Arrays.toString(OverdueArr));
System.out.println(i);
int student_id_compare = Integer.parseInt(inventory_student_id_dropdown.getSelectedItem().toString());
int first = 0;
int last=OverdueArr.length-1;
int found = -1;
int mid;
while( last> first && found == -1 ){
    mid = (first + last)/2;
    if ( OverdueArr[mid] < student_id_compare ){
    first = mid + 1;
}else {
        if ( student_id_compare < OverdueArr[mid]){
    last = mid - 1;
}else{
    found ++;
}
```

In the binary search, variable "found" is used to see if the student ID is present in the overdueArr or not

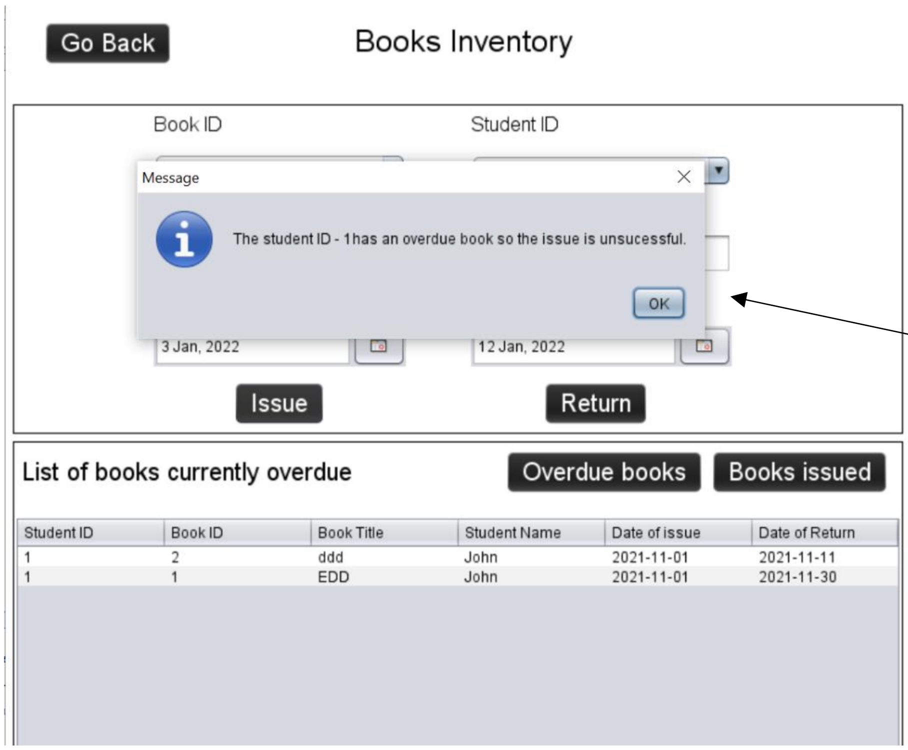
Figure 20 – Code for Binary search to check if the Student ID to whom the book is being issued has any overdue book or not.



```
if( found == -1){
      java.sql.Date sqldate = new java.sql.Date(inventory_date_issue_datechooser.getDate().getTime());
        java.sql.Date sqldate1 = new java.sql.Date(inventory_date_return_datechooser.getDate().getTime());

    try{
        String sql = "Insert into inventory" +"(name,title,issued,datei,dater,ids,idb)" + "values(?,?,?,?,?,?,?)";
       // String sql = "Insert into inventory" +"(title,name,datei,dater,issued,ids,idb)" + "values(?,?,?,?,?,?,?)";
        PreparedStatement ps = dbconnect.librarycon.prepareStatement(sql);

        ps.setString(1, jTextField3.getText());
        ps.setString(2, inventory_book_title_feildtext.getText());
        ps.setString(3,"Y");
        ps.setDate(4,sqldate);
        ps.setDate(5,sqldate1);
        ps.setString(6, inventory_student_id_dropdown.getSelectedItem().toString());
        ps.setString(7, inventory_book_id_dropdown.getSelectedItem().toString());
        ps.execute();
        JOptionPane.showMessageDialog(null, "Issued successfully!!");
        inventory_book_title_feildtext.setText("");
        jTextField3.setText("");
        ((JTextField)inventory_date_issue_datechooser.getDateEditor().getUiComponent()).setText("");
        ((JTextField)inventory_date_issue_datechooser.getDateEditor().getUiComponent()).setText("");
        try {
            String sql1 = "Update book set issued = ? where id =?";
        ps = dbconnect.librarycon.prepareStatement(sql1);
        ps.setString(1, "Y");
        ps.setInt(2, Integer.valueOf(inventory_book_id_dropdown.getSelectedItem().toString()));
        ps.execute();
        }
        catch(Exception e){
        e.printStackTrace();
        JOptionPane.showMessageDialog(null, e);
        }
        }
    catch(Exception e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(null, e);
}

}else{
    JOptionPane.showMessageDialog(null, "The student ID - "+student_id_compare+"has an overdue book so the issue is unsucessful.");
}

        rs.close();

        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, e);
        }
```

When the found variable == -1 it means that the student has no overdue book. Thereafter the book issue process continues. On the other hand, If found has any other value then a warning is given.

Figure 21 – Based on the student ID found or not in the overdue array, action will be taken using the above code.



The student ID - 1has an overdue book so the issue is unsucessful.

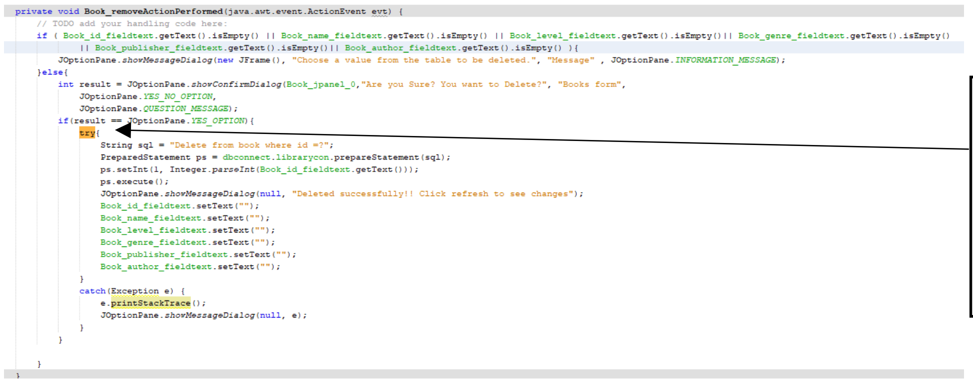
Student ID	Book ID	Book Title	Student Name	Date of issue	Date of Return
1	2	ddd	John	2021-11-01	2021-11-11
1	1	EDD	John	2021-11-01	2021-11-30

This warning message will be displayed and the book will not be issued until the student return back the overdue book.

Figure 22 – Output when a student has an overdue book then this kind of a warning comes.

5) Exception handling[7] with validation and prompts

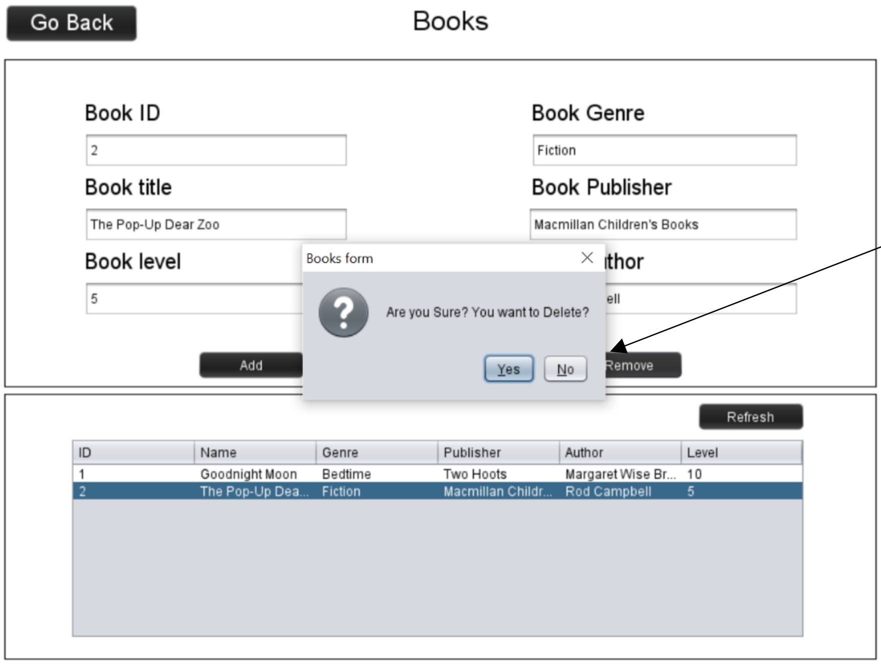
This system makes use of structured programming hence the client needs to receive error messages or information when invalid input is entered. This can act as a validation to inform the user about what mistakes are they making. Also, to avoid any runtime error or database error exception handling is used such as Try-Catch statement.



```java
private void Book_removeActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if ( Book_id_fieldtext.getText().isEmpty() || Book_name_fieldtext.getText().isEmpty() || Book_level_fieldtext.getText().isEmpty()|| Book_genre_fieldtext.getText().isEmpty()
            || Book_publisher_fieldtext.getText().isEmpty()|| Book_author_fieldtext.getText().isEmpty() ){
        JOptionPane.showMessageDialog(new JFrame(), "Choose a value from the table to be deleted.", "Message" , JOptionPane.INFORMATION_MESSAGE);
    }else{
        int result = JOptionPane.showConfirmDialog(Book_jpanel_0,"Are you Sure? You want to Delete?", "Books form",
            JOptionPane.YES_NO_OPTION,
            JOptionPane.QUESTION_MESSAGE);
        if(result == JOptionPane.YES_OPTION){
            try{
                String sql = "Delete from book where id =?";
                PreparedStatement ps = dbconnect.librarycon.prepareStatement(sql);
                ps.setInt(1, Integer.parseInt(Book_id_fieldtext.getText()));
                ps.execute();
                JOptionPane.showMessageDialog(null, "Deleted successfully!! Click refresh to see changes");
                Book_id_fieldtext.setText("");
                Book_name_fieldtext.setText("");
                Book_level_fieldtext.setText("");
                Book_genre_fieldtext.setText("");
                Book_publisher_fieldtext.setText("");
                Book_author_fieldtext.setText("");
            }
            catch(Exception e) {
                e.printStackTrace();
                JOptionPane.showMessageDialog(null, e);
            }
        }

    }
}
```

We can see the code of removing a book and this is covered with Try- Catch statement which helps avoid any run-time error.

Figure 23 – Code for removing a book from the database which has been removed from the library



Book ID: 2
Book Genre: Fiction
Book title: The Pop-Up Dear Zoo
Book Publisher: Macmillan Children's Books
Book level: 5

Books form

Are you Sure? You want to Delete?

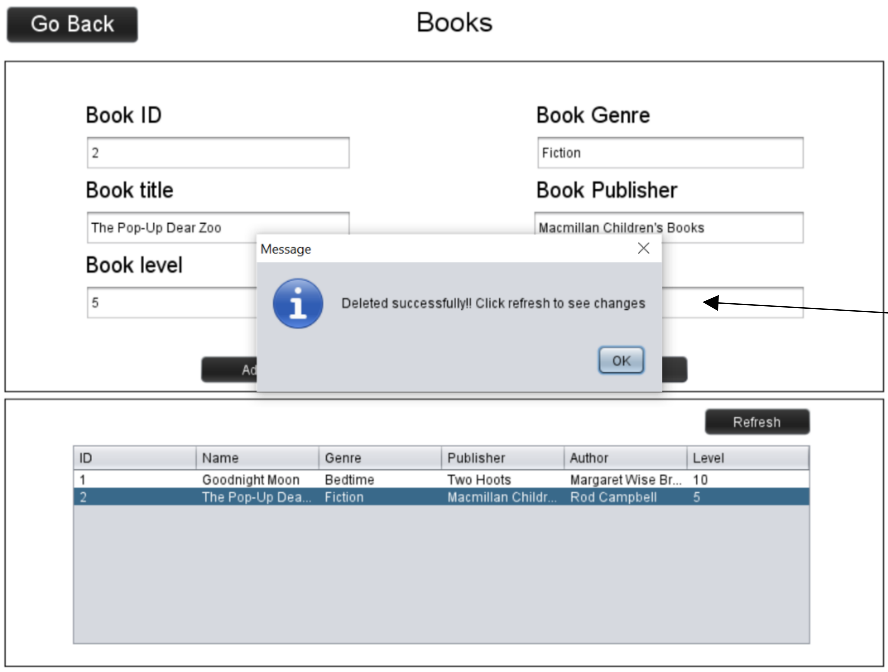
Yes No

ID	Name	Genre	Publisher	Author	Level
1	Goodnight Moon	Bedtime	Two Hoots	Margaret Wise Br...	10
2	The Pop-Up Dea...	Fiction	Macmillan Childr...	Rod Campbell	5

This is a confirmation note. This is validation prompt re-validating if the client wants to delete the book or was its a by mistake click.

Figure 24 – Output when a book is chosen to be delete.

[7] https://www.programiz.com/java-programming/exception-handling



Book ID: 2
Book Genre: Fiction
Book title: The Pop-Up Dear Zoo
Book Publisher: Macmillan Children's Books
Book level: 5

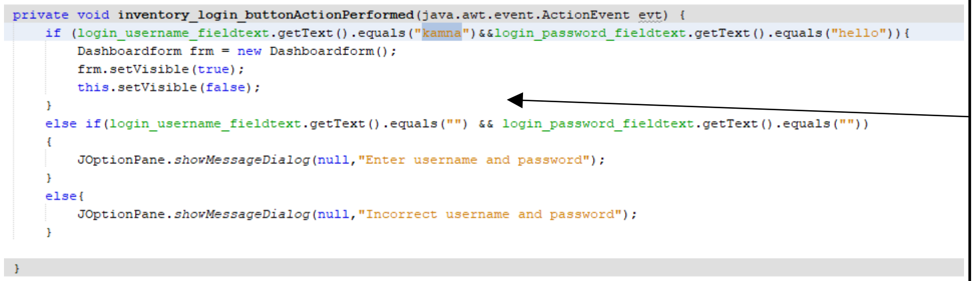
Message

Deleted successfully!! Click refresh to see changes

ID	Name	Genre	Publisher	Author	Level
1	Goodnight Moon	Bedtime	Two Hoots	Margaret Wise Br...	10
2	The Pop-Up Dea...	Fiction	Macmillan Childr...	Rod Campbell	5

Deleted error box depicting the book has been deleted successfully.

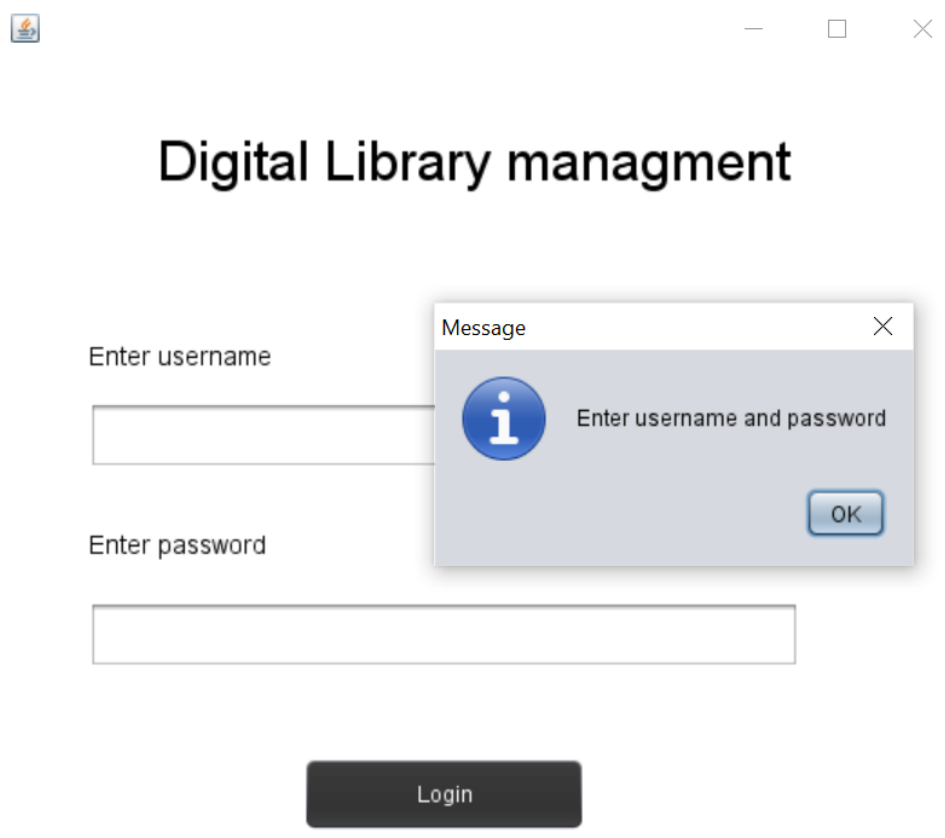
Figure 25 – Output when a book is successfully deleted.



```
private void inventory_login_buttonActionPerformed(java.awt.event.ActionEvent evt) {
    if (login_username_fieldtext.getText().equals("kamna")&&login_password_fieldtext.getText().equals("hello")){
        Dashboardform frm = new Dashboardform();
        frm.setVisible(true);
        this.setVisible(false);
    }
    else if(login_username_fieldtext.getText().equals("") && login_password_fieldtext.getText().equals(""))
    {
        JOptionPane.showMessageDialog(null,"Enter username and password");
    }
    else{
        JOptionPane.showMessageDialog(null,"Incorrect username and password");
    }

}
```

Use of If statement to check if nothing is entered then appropriate dialogue box is visible and similarly is wrong username or password is entered then there is warning message as well.

Figure 26 – Code for validating if the password and the username entered matches the set password and username


Digital Library managment

Enter username

Enter password

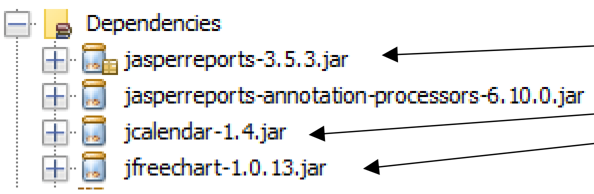
Message

Enter username and password

This error message appears when no information is entered and the login button is clicked.

Figure 27 – Output of a case where there is no login details are entered but the login button is pressed.

6) External libraries

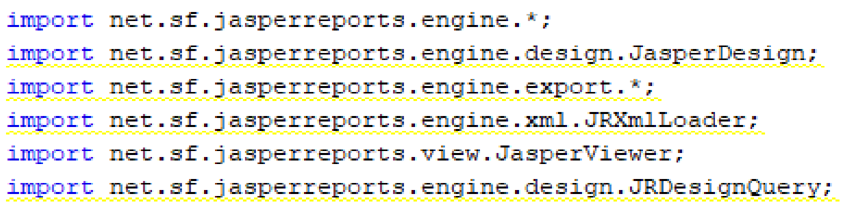

Dependencies
jasperreports-3.5.3.jar
jasperreports-annotation-processors-6.10.0.jar
jcalendar-1.4.jar
jfreechart-1.0.13.jar

These are the external libraries used.

Figure 28 – List of all external libraries used in the solution

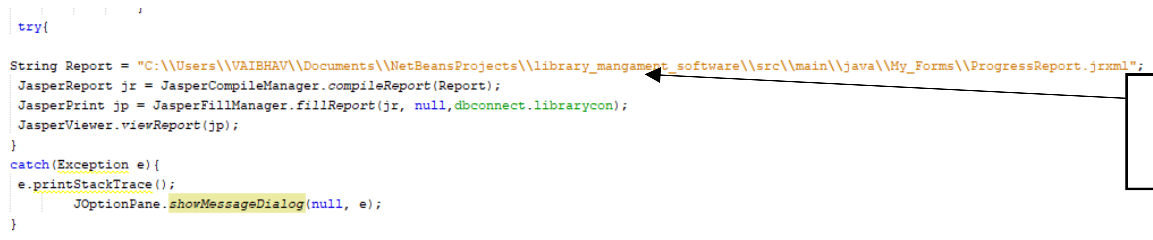
6.1) Jasper Report[8]

Jasper Report is used to generate a dynamic report to depict the progress of the student by listing all the books the student has read in a specific period.



```
import net.sf.jasperreports.engine.*;
import net.sf.jasperreports.engine.design.JasperDesign;
import net.sf.jasperreports.engine.export.*;
import net.sf.jasperreports.engine.xml.JRXmlLoader;
import net.sf.jasperreports.view.JasperViewer;
import net.sf.jasperreports.engine.design.JRDesignQuery;
```

Figure 29 – Importing Jasper report in the progress report form



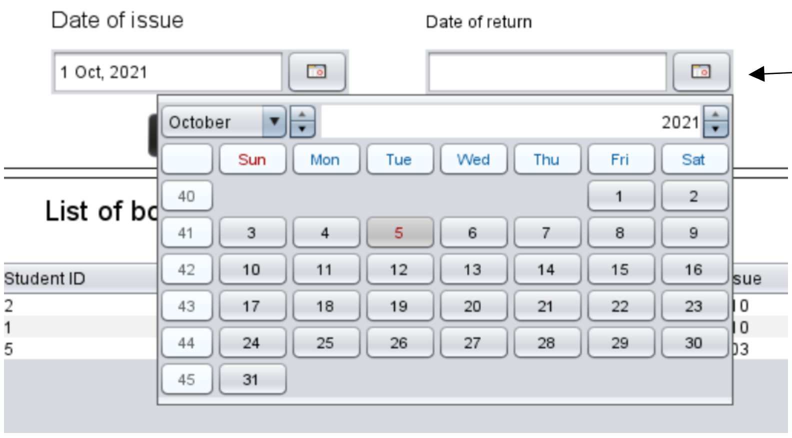
```
 try{

String Report = "C:\\Users\\VAIBHAV\\Documents\\NetBeansProjects\\library_mangament_software\\src\\main\\java\\My_Forms\\ProgressReport.jrxml";
 JasperReport jr = JasperCompileManager.compileReport(Report);
 JasperPrint jp = JasperFillManager.fillReport(jr, null,dbconnect.librarycon);
 JasperViewer.viewReport(jp);
}
catch(Exception e){
 e.printStackTrace();
        JOptionPane.showMessageDialog(null, e);
}
```

Storing the report on a local file space for now

Figure 30 – Implementing the Jasper Report library to create an automatic report

6.2) Jcalender[9]

Jcalender library is used to provide a calendar like a framework where the user can easily choose the date from instead of manually typing it.



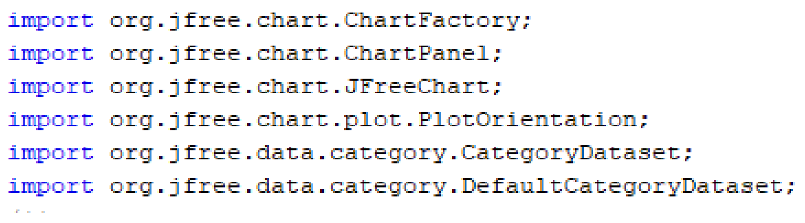
On click of this button the calendar view will appear.

Figure 31 – JCalendar is used in Book Inventory form to let the user choose the date

[8] https://community.jaspersoft.com/wiki/build-jasper-report-using-jasper-library-using-net-beans-ide
[9] https://toedter.com/jcalendar/

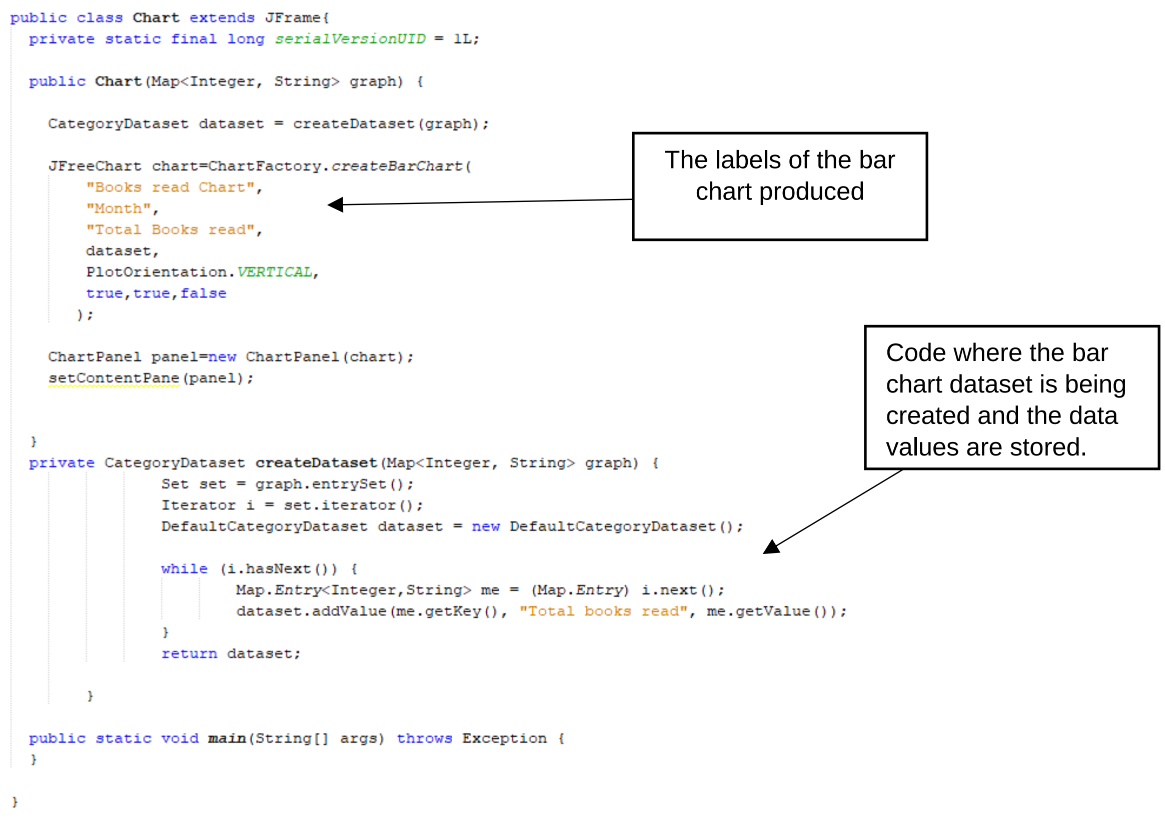
6.3) JFreeChart[10]

JFreeChart is used to generate a dynamic chart for the progress of the student on the monthly basis from the chosen period by the client.



```java
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.CategoryDataset;
import org.jfree.data.category.DefaultCategoryDataset;
```

Figure 32 – Importing JFreeChart in the progress report form



```java
public class Chart extends JFrame{
  private static final long serialVersionUID = 1L;

  public Chart(Map<Integer, String> graph) {

    CategoryDataset dataset = createDataset(graph);

    JFreeChart chart=ChartFactory.createBarChart(
        "Books read Chart",
        "Month",
        "Total Books read",
        dataset,
        PlotOrientation.VERTICAL,
        true,true,false
       );

    ChartPanel panel=new ChartPanel(chart);
    setContentPane(panel);


  }
  private CategoryDataset createDataset(Map<Integer, String> graph) {
              Set set = graph.entrySet();
              Iterator i = set.iterator();
              DefaultCategoryDataset dataset = new DefaultCategoryDataset();

              while (i.hasNext()) {
                      Map.Entry<Integer,String> me = (Map.Entry) i.next();
                      dataset.addValue(me.getKey(), "Total books read", me.getValue());
              }
              return dataset;

        }

  public static void main(String[] args) throws Exception {
  }

}
```

The labels of the bar chart produced
Code where the bar chart dataset is being created and the data values are stored.

Figure 33 – Implementing the JFreeChart library with the code to generate a chart with hashmap collection where the data is stored from the database

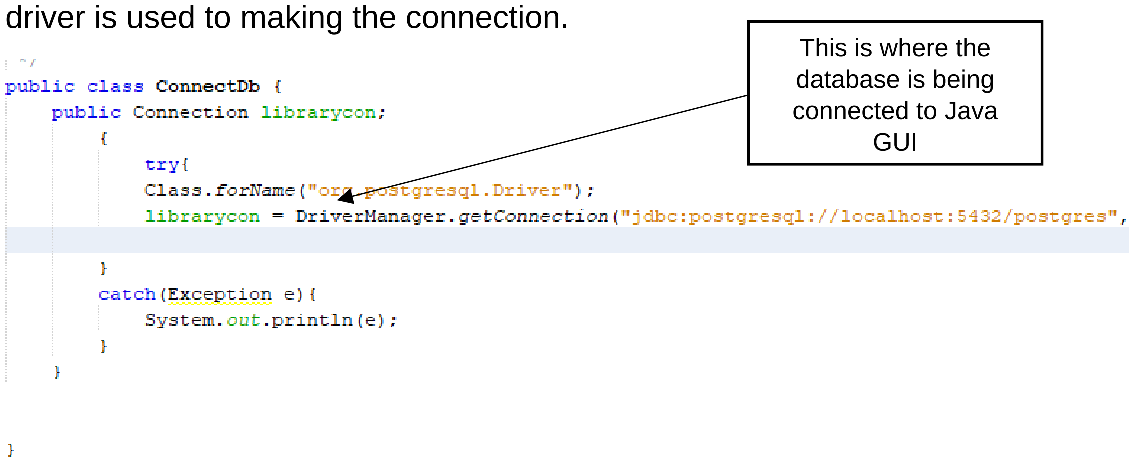
[10] https://www.jfree.org/jfreechart/

7) SQL Database

In this product, PostgreSQL is used as a database to store all the information related to the students, books, track of books issued and returned and progress reports.

7.1) Database connectivity[11]

The database is connected with the GUI program using the connection method. JDBC driver is used to making the connection.



```
*/
public class ConnectDb {
    public Connection librarycon;
        {
            try{
            Class.forName("org.postgresql.Driver");
            librarycon = DriverManager.getConnection("jdbc:postgresql://localhost:5432/postgres",

        }
        catch(Exception e){
            System.out.println(e);
        }
    }


}
```

Figure 34 – Code for connecting the database with the GUI application

[11] https://www.youtube.com/watch?v=Nzxqg8I0tcQ

7.2) Database structure

My database connection is made and tested with a local server for proper functioning.

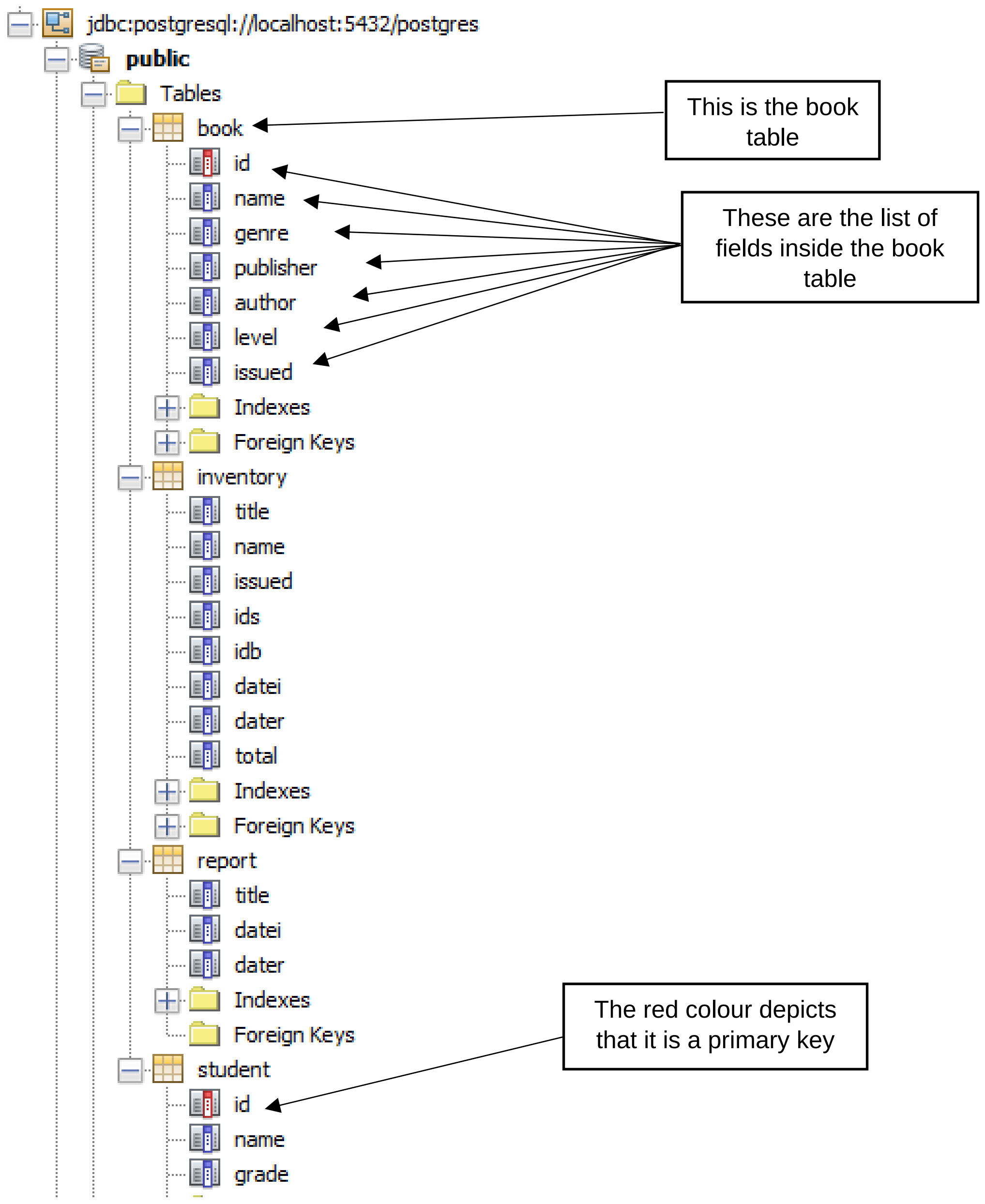
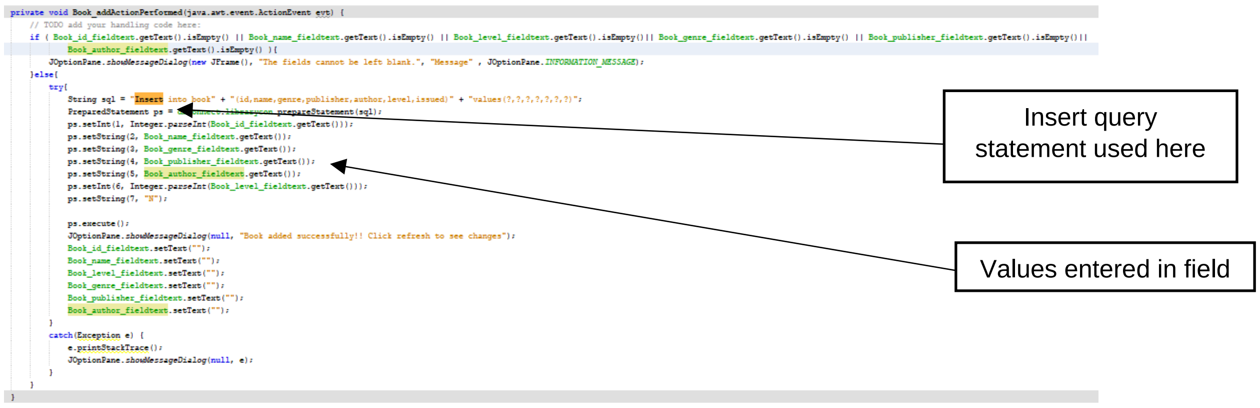


Figure 35 – List of the tables and fields used in the application

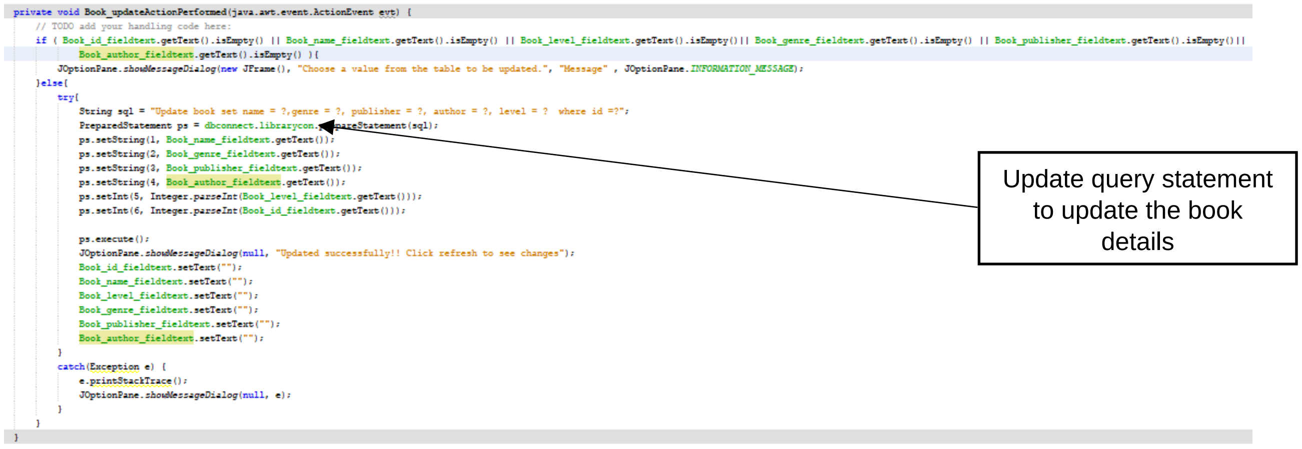
7.3) <u>SQL Commands and Queries</u>

The SQL queries commands techniques are used to insert, update, filter, order by, select and search information with specific criteria.



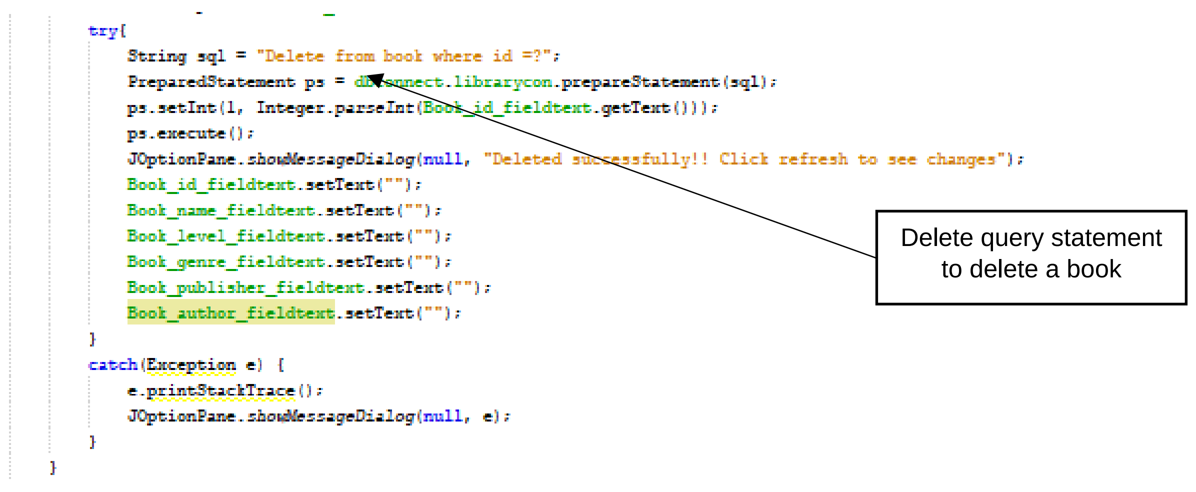
```
private void Book_addActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if ( Book_id_fieldtext.getText().isEmpty() || Book_name_fieldtext.getText().isEmpty() || Book_level_fieldtext.getText().isEmpty()|| Book_genre_fieldtext.getText().isEmpty() || Book_publisher_fieldtext.getText().isEmpty()||
            Book_author_fieldtext.getText().isEmpty() ){
        JOptionPane.showMessageDialog(new JFrame(), "The fields cannot be left blank.", "Message" , JOptionPane.INFORMATION_MESSAGE);
    }else{
        try{
            String sql = "Insert into book" + "(id,name,genre,publisher,author,level,issued)" + "values(?,?,?,?,?,?,?)";
            PreparedStatement ps = dbconnect.librarycon.prepareStatement(sql);
            ps.setInt(1, Integer.parseInt(Book_id_fieldtext.getText()));
            ps.setString(2, Book_name_fieldtext.getText());
            ps.setString(3, Book_genre_fieldtext.getText());
            ps.setString(4, Book_publisher_fieldtext.getText());
            ps.setString(5, Book_author_fieldtext.getText());
            ps.setInt(6, Integer.parseInt(Book_level_fieldtext.getText()));
            ps.setString(7, "N");

            ps.execute();
            JOptionPane.showMessageDialog(null, "Book added successfully!! Click refresh to see changes");
            Book_id_fieldtext.setText("");
            Book_name_fieldtext.setText("");
            Book_level_fieldtext.setText("");
            Book_genre_fieldtext.setText("");
            Book_publisher_fieldtext.setText("");
            Book_author_fieldtext.setText("");
        }
        catch(Exception e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(null, e);
        }
    }
}
```

Figure 36 – Insert query to add new books in the database



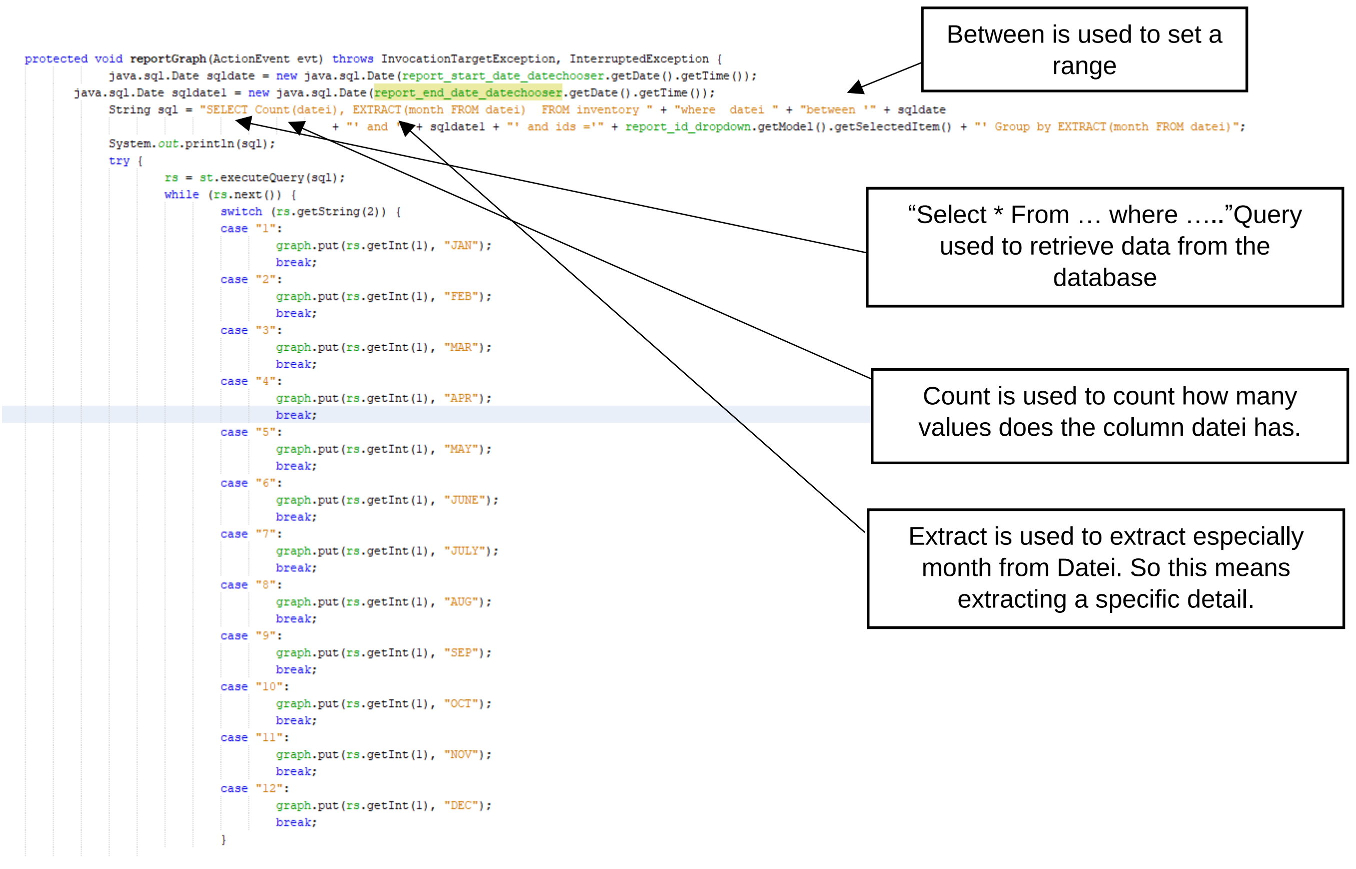
```
private void Book_updateActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if ( Book_id_fieldtext.getText().isEmpty() || Book_name_fieldtext.getText().isEmpty() || Book_level_fieldtext.getText().isEmpty()|| Book_genre_fieldtext.getText().isEmpty() || Book_publisher_fieldtext.getText().isEmpty()||
            Book_author_fieldtext.getText().isEmpty() ){
        JOptionPane.showMessageDialog(new JFrame(), "Choose a value from the table to be updated.", "Message" , JOptionPane.INFORMATION_MESSAGE);
    }else{
        try{
            String sql = "Update book set name = ?,genre = ?, publisher = ?, author = ?, level = ?  where id =?";
            PreparedStatement ps = dbconnect.librarycon.prepareStatement(sql);
            ps.setString(1, Book_name_fieldtext.getText());
            ps.setString(2, Book_genre_fieldtext.getText());
            ps.setString(3, Book_publisher_fieldtext.getText());
            ps.setString(4, Book_author_fieldtext.getText());
            ps.setInt(5, Integer.parseInt(Book_level_fieldtext.getText()));
            ps.setInt(6, Integer.parseInt(Book_id_fieldtext.getText()));

            ps.execute();
            JOptionPane.showMessageDialog(null, "Updated successfully!! Click refresh to see changes");
            Book_id_fieldtext.setText("");
            Book_name_fieldtext.setText("");
            Book_level_fieldtext.setText("");
            Book_genre_fieldtext.setText("");
            Book_publisher_fieldtext.setText("");
            Book_author_fieldtext.setText("");
        }
        catch(Exception e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(null, e);
        }
    }
}
```

Figure 37 – Update query to update an existing book detail in the database



```
try{
    String sql = "Delete from book where id =?";
    PreparedStatement ps = dbconnect.librarycon.prepareStatement(sql);
    ps.setInt(1, Integer.parseInt(Book_id_fieldtext.getText()));
    ps.execute();
    JOptionPane.showMessageDialog(null, "Deleted successfully!! Click refresh to see changes");
    Book_id_fieldtext.setText("");
    Book_name_fieldtext.setText("");
    Book_level_fieldtext.setText("");
    Book_genre_fieldtext.setText("");
    Book_publisher_fieldtext.setText("");
    Book_author_fieldtext.setText("");
}
catch(Exception e) {
    e.printStackTrace();
    JOptionPane.showMessageDialog(null, e);
}
}
```

Figure 38 – Delete query to delete an existing book detail in the database



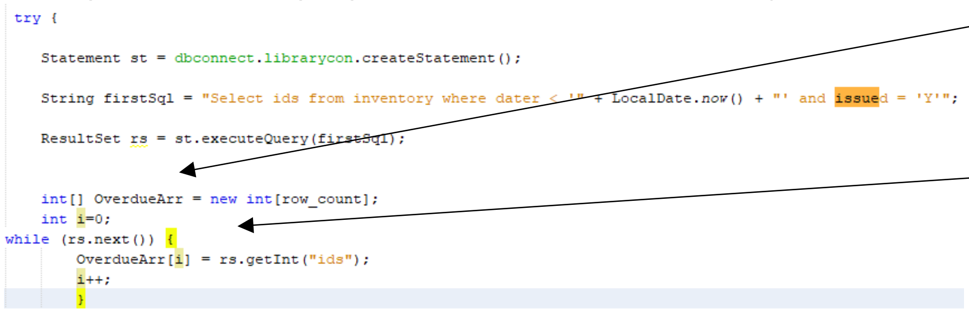
```
protected void reportGraph(ActionEvent evt) throws InvocationTargetException, InterruptedException {
            java.sql.Date sqldate = new java.sql.Date(report_start_date_datechooser.getDate().getTime());
        java.sql.Date sqldate1 = new java.sql.Date(report_end_date_datechooser.getDate().getTime());
            String sql = "SELECT Count(datei), EXTRACT(month FROM datei)  FROM inventory " + "where  datei " + "between '" + sqldate
                                    + "' and '" + sqldate1 + "' and ids ='" + report_id_dropdown.getModel().getSelectedItem() + "' Group by EXTRACT(month FROM datei)";
            System.out.println(sql);
            try {
                    rs = st.executeQuery(sql);
                    while (rs.next()) {
                            switch (rs.getString(2)) {
                            case "1":
                                    graph.put(rs.getInt(1), "JAN");
                                    break;
                            case "2":
                                    graph.put(rs.getInt(1), "FEB");
                                    break;
                            case "3":
                                    graph.put(rs.getInt(1), "MAR");
                                    break;
                            case "4":
                                    graph.put(rs.getInt(1), "APR");
                                    break;
                            case "5":
                                    graph.put(rs.getInt(1), "MAY");
                                    break;
                            case "6":
                                    graph.put(rs.getInt(1), "JUNE");
                                    break;
                            case "7":
                                    graph.put(rs.getInt(1), "JULY");
                                    break;
                            case "8":
                                    graph.put(rs.getInt(1), "AUG");
                                    break;
                            case "9":
                                    graph.put(rs.getInt(1), "SEP");
                                    break;
                            case "10":
                                    graph.put(rs.getInt(1), "OCT");
                                    break;
                            case "11":
                                    graph.put(rs.getInt(1), "NOV");
                                    break;
                            case "12":
                                    graph.put(rs.getInt(1), "DEC");
                                    break;
                            }
```

Between is used to set a range
"Select * From … where …..."Query used to retrieve data from the database
Count is used to count how many values does the column datei has.
Extract is used to extract especially month from Datei. So this means extracting a specific detail.

Figure 39 – Various SQL commands used to create bar Chart

8) <u>Data Structure</u>

One dimensional array has been used to store the student ID of students who have an overdue book. This is for warning and restricting the clients from issuing new books in the book to students who have overdue books. This array can store integers and this array size is set based on the row count of the students who have overdue books. Also, sorting and searching algorithms are used on this array later on.



```
 try {

    Statement st = dbconnect.librarycon.createStatement();

    String firstSql = "Select ids from inventory where dater < '" + LocalDate.now() + "' and issued = 'Y'";

    ResultSet rs = st.executeQuery(firstSql);


    int[] OverdueArr = new int[row_count];
    int i=0;
while (rs.next()) {
        OverdueArr[i] = rs.getInt("ids");
        i++;
        }
```

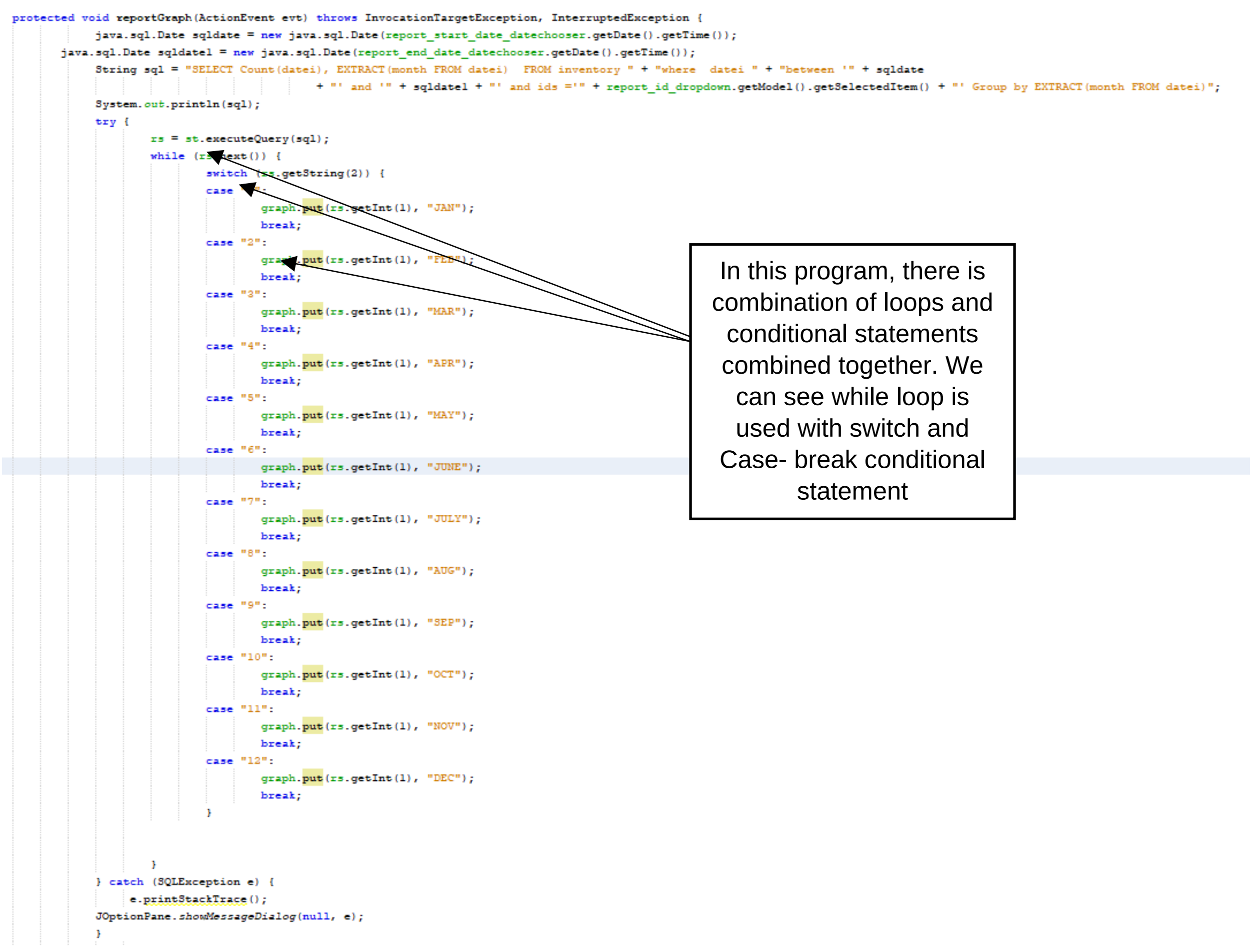
OverdueArr is used to store the list of student ID which have overdue books

We can see the student Id ("ids") being stored in an Array. The row count is the size of available student ID which is calculated before.

Figure 40 – Code for storing the student Id of students who have overdue books

9) **Nested Loops and Complex Conditional Statements**

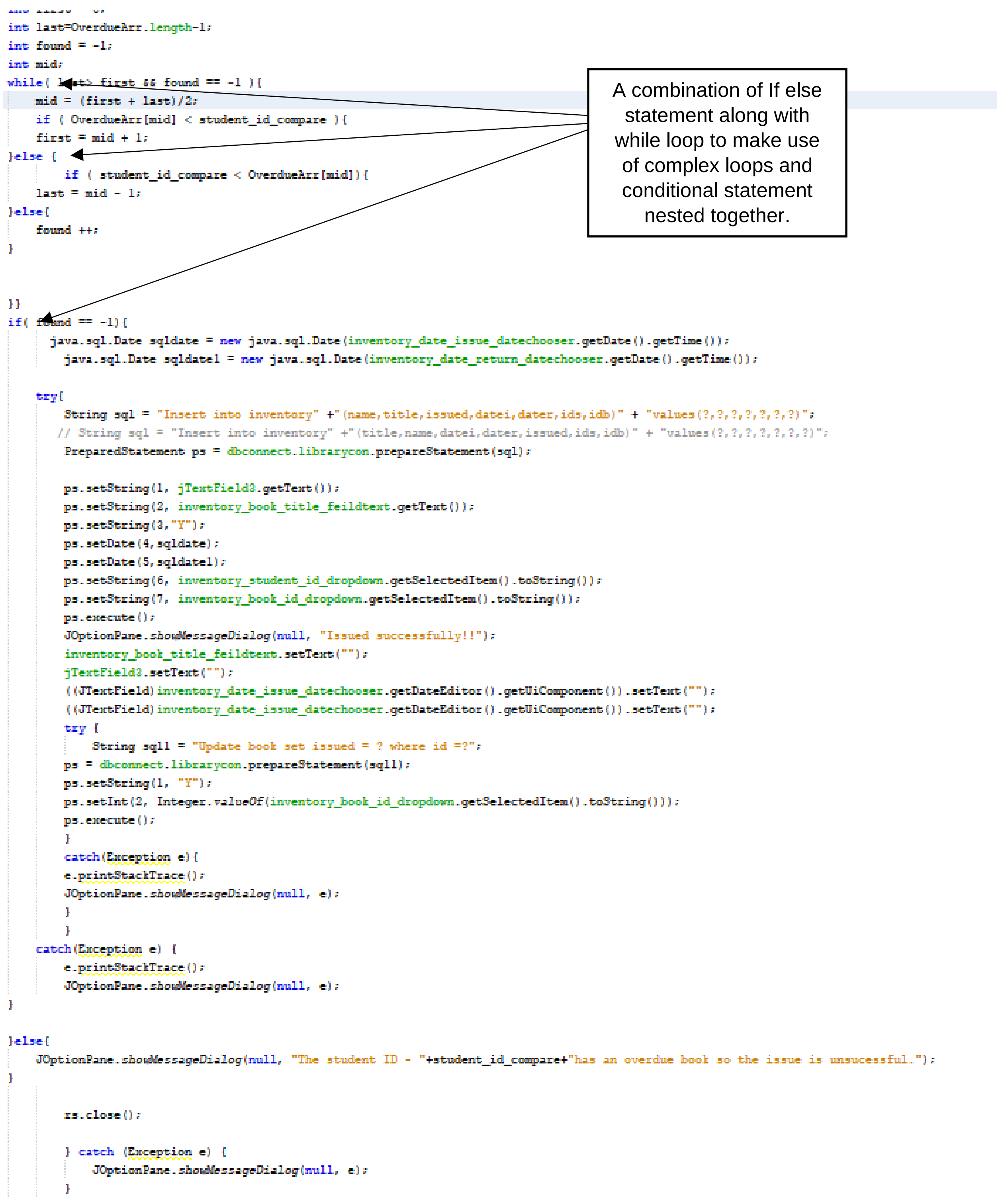
In the program, there is the use of complex and nested loops with conditional statements. It is used to make the program more user-friendly such as creating automatic graphs, reports and many more.



```
protected void reportGraph(ActionEvent evt) throws InvocationTargetException, InterruptedException {
            java.sql.Date sqldate = new java.sql.Date(report_start_date_datechooser.getDate().getTime());
        java.sql.Date sqldate1 = new java.sql.Date(report_end_date_datechooser.getDate().getTime());
            String sql = "SELECT Count(datei), EXTRACT(month FROM datei)  FROM inventory " + "where  datei " + "between '" + sqldate
                                                + "' and '" + sqldate1 + "' and ids ='" + report_id_dropdown.getModel().getSelectedItem() + "' Group by EXTRACT(month FROM datei)";
            System.out.println(sql);
            try {
                    rs = st.executeQuery(sql);
                    while (rs.next()) {
                            switch (rs.getString(2)) {
                            case "1":
                                    graph.put(rs.getInt(1), "JAN");
                                    break;
                            case "2":
                                    graph.put(rs.getInt(1), "FEB");
                                    break;
                            case "3":
                                    graph.put(rs.getInt(1), "MAR");
                                    break;
                            case "4":
                                    graph.put(rs.getInt(1), "APR");
                                    break;
                            case "5":
                                    graph.put(rs.getInt(1), "MAY");
                                    break;
                            case "6":
                                    graph.put(rs.getInt(1), "JUNE");
                                    break;
                            case "7":
                                    graph.put(rs.getInt(1), "JULY");
                                    break;
                            case "8":
                                    graph.put(rs.getInt(1), "AUG");
                                    break;
                            case "9":
                                    graph.put(rs.getInt(1), "SEP");
                                    break;
                            case "10":
                                    graph.put(rs.getInt(1), "OCT");
                                    break;
                            case "11":
                                    graph.put(rs.getInt(1), "NOV");
                                    break;
                            case "12":
                                    graph.put(rs.getInt(1), "DEC");
                                    break;
                            }


                    }
            } catch (SQLException e) {
                 e.printStackTrace();
            JOptionPane.showMessageDialog(null, e);
            }
```

In this program, there is combination of loops and conditional statements combined together. We can see while loop is used with switch and Case- break conditional statement

Figure 41 – Code for getting all the data records about the books issued to a particular student. Using these records to create a graph in the progress report form.



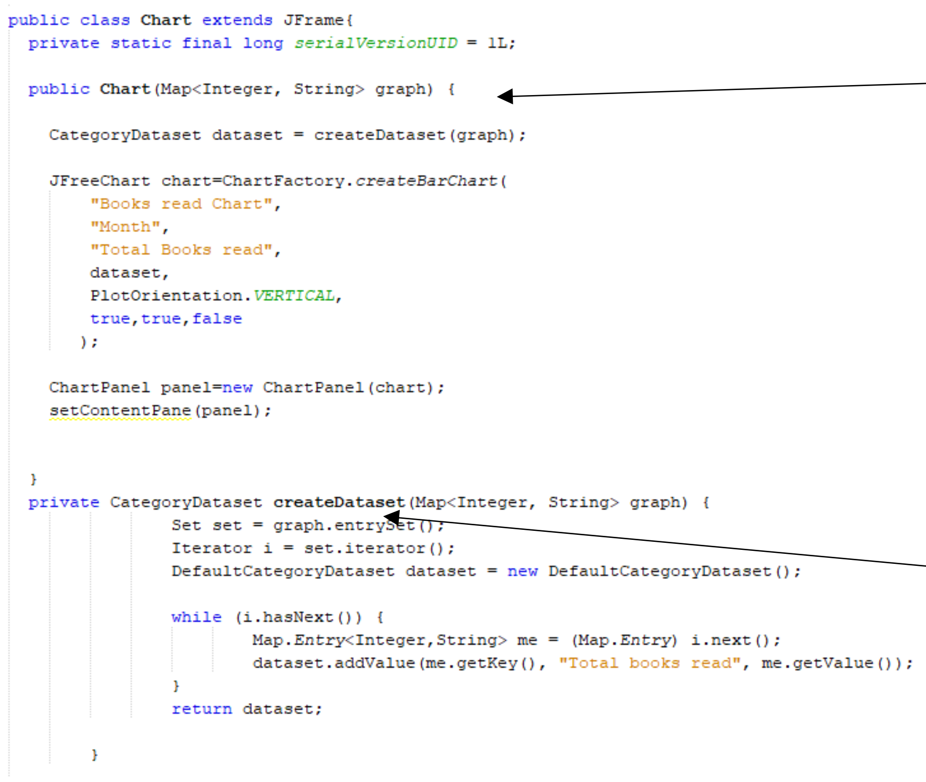
```
int last=OverdueArr.length-1;
int found = -1;
int mid;
while( last> first && found == -1 ){
    mid = (first + last)/2;
    if ( OverdueArr[mid] < student_id_compare ){
    first = mid + 1;
}else {
        if ( student_id_compare < OverdueArr[mid]){
    last = mid - 1;
}else{
    found ++;
}


}}
if( found == -1){
     java.sql.Date sqldate = new java.sql.Date(inventory_date_issue_datechooser.getDate().getTime());
       java.sql.Date sqldate1 = new java.sql.Date(inventory_date_return_datechooser.getDate().getTime());

    try{
       String sql = "Insert into inventory" +"(name,title,issued,datei,dater,ids,idb)" + "values(?,?,?,?,?,?,?)";
      // String sql = "Insert into inventory" +"(title,name,datei,dater,issued,ids,idb)" + "values(?,?,?,?,?,?,?)";
       PreparedStatement ps = dbconnect.librarycon.prepareStatement(sql);

       ps.setString(1, jTextField3.getText());
       ps.setString(2, inventory_book_title_feildtext.getText());
       ps.setString(3,"Y");
       ps.setDate(4,sqldate);
       ps.setDate(5,sqldate1);
       ps.setString(6, inventory_student_id_dropdown.getSelectedItem().toString());
       ps.setString(7, inventory_book_id_dropdown.getSelectedItem().toString());
       ps.execute();
       JOptionPane.showMessageDialog(null, "Issued successfully!!");
       inventory_book_title_feildtext.setText("");
       jTextField3.setText("");
       ((JTextField)inventory_date_issue_datechooser.getDateEditor().getUiComponent()).setText("");
       ((JTextField)inventory_date_issue_datechooser.getDateEditor().getUiComponent()).setText("");
       try {
           String sql1 = "Update book set issued = ? where id =?";
       ps = dbconnect.librarycon.prepareStatement(sql1);
       ps.setString(1, "Y");
       ps.setInt(2, Integer.valueOf(inventory_book_id_dropdown.getSelectedItem().toString()));
       ps.execute();
       }
       catch(Exception e){
       e.printStackTrace();
       JOptionPane.showMessageDialog(null, e);
       }
       }
    catch(Exception e) {
       e.printStackTrace();
       JOptionPane.showMessageDialog(null, e);
}

}else{
    JOptionPane.showMessageDialog(null, "The student ID - "+student_id_compare+"has an overdue book so the issue is unsucessful.");
}


       rs.close();

       } catch (Exception e) {
           JOptionPane.showMessageDialog(null, e);
       }
```

A combination of If else statement along with while loop to make use of complex loops and conditional statement nested together.

Figure 42 – Part of the code used to check if a student has an overdue book or not before issuing a book to the student

10) Dynamic chart generation

To make it convent for the client to analyze the regular progress of the student over some time, a monthly Bar chart is generated using JFreeChart external library.

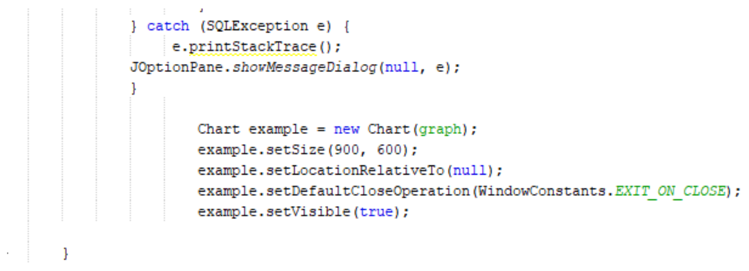


```
public class Chart extends JFrame{
  private static final long serialVersionUID = 1L;

  public Chart(Map<Integer, String> graph) {

    CategoryDataset dataset = createDataset(graph);

    JFreeChart chart=ChartFactory.createBarChart(
        "Books read Chart",
        "Month",
        "Total Books read",
        dataset,
        PlotOrientation.VERTICAL,
        true,true,false
       );

    ChartPanel panel=new ChartPanel(chart);
    setContentPane(panel);


  }
  private CategoryDataset createDataset(Map<Integer, String> graph) {
              Set set = graph.entrySet();
              Iterator i = set.iterator();
              DefaultCategoryDataset dataset = new DefaultCategoryDataset();

              while (i.hasNext()) {
                      Map.Entry<Integer,String> me = (Map.Entry) i.next();
                      dataset.addValue(me.getKey(), "Total books read", me.getValue());
              }
              return dataset;

        }
```

This is the method chart where the code of creating the bar chart is placed

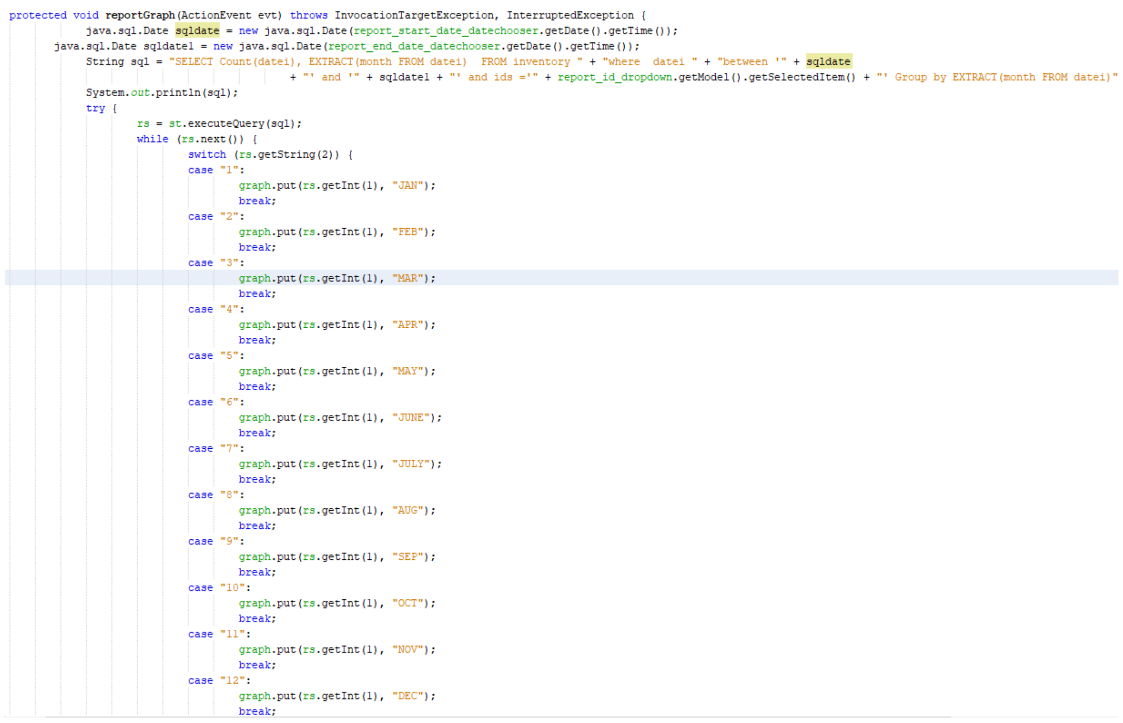
CreateDataset Method is used to create a dataset of all the values need to create the barchart.

Figure 43 – Code for creating the Bar chart



```
            } catch (SQLException e) {
                e.printStackTrace();
            JOptionPane.showMessageDialog(null, e);
            }

                Chart example = new Chart(graph);
                example.setSize(900, 600);
                example.setLocationRelativeTo(null);
                example.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
                example.setVisible(true);

      }
```

Figure 44 – Code for setting the charts window



```
protected void reportGraph(ActionEvent evt) throws InvocationTargetException, InterruptedException {
            java.sql.Date sqldate = new java.sql.Date(report_start_date_datechooser.getDate().getTime());
        java.sql.Date sqldate1 = new java.sql.Date(report_end_date_datechooser.getDate().getTime());
            String sql = "SELECT Count(datei), EXTRACT(month FROM datei)  FROM inventory " + "where  datei " + "between '" + sqldate
                                                  + "' and '" + sqldate1 + "' and ids ='" + report_id_dropdown.getModel().getSelectedItem() + "' Group by EXTRACT(month FROM datei)"
            System.out.println(sql);
            try {
                    rs = st.executeQuery(sql);
                    while (rs.next()) {
                            switch (rs.getString(2)) {
                            case "1":
                                    graph.put(rs.getInt(1), "JAN");
                                    break;
                            case "2":
                                    graph.put(rs.getInt(1), "FEB");
                                    break;
                            case "3":
                                    graph.put(rs.getInt(1), "MAR");
                                    break;
                            case "4":
                                    graph.put(rs.getInt(1), "APR");
                                    break;
                            case "5":
                                    graph.put(rs.getInt(1), "MAY");
                                    break;
                            case "6":
                                    graph.put(rs.getInt(1), "JUNE");
                                    break;
                            case "7":
                                    graph.put(rs.getInt(1), "JULY");
                                    break;
                            case "8":
                                    graph.put(rs.getInt(1), "AUG");
                                    break;
                            case "9":
                                    graph.put(rs.getInt(1), "SEP");
                                    break;
                            case "10":
                                    graph.put(rs.getInt(1), "OCT");
                                    break;
                            case "11":
                                    graph.put(rs.getInt(1), "NOV");
                                    break;
                            case "12":
                                    graph.put(rs.getInt(1), "DEC");
                                    break;
```

Figure 45 – Code for retrieving data from the database for creating the dataset to create the barchart

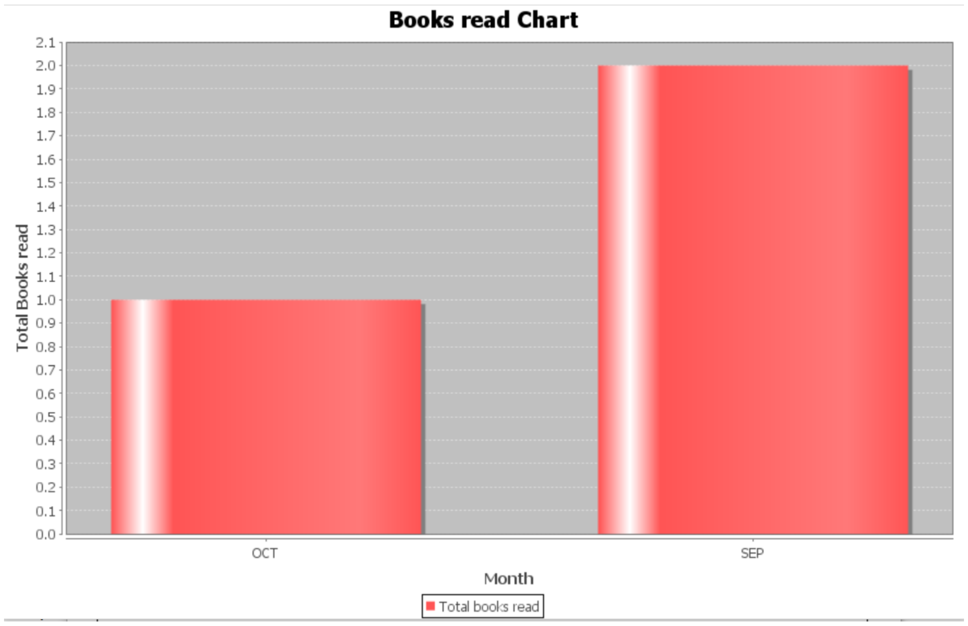
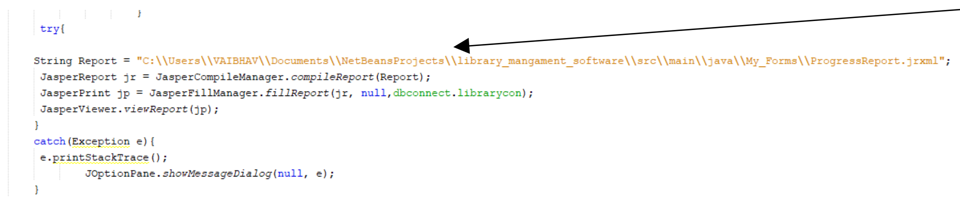


Figure 46 – This is the output chart which is generated

11) Report Generation

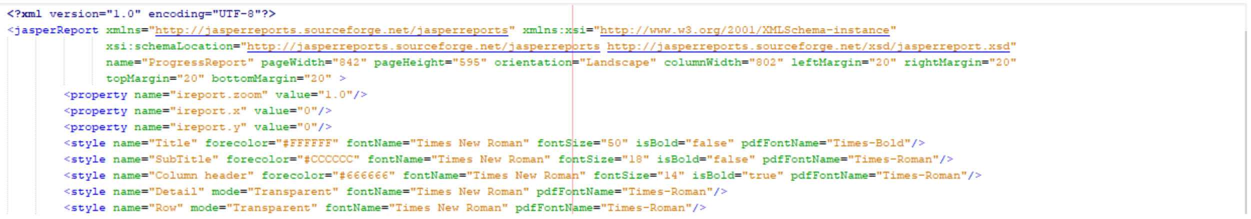
A downloadable and printable report is automatically generated for the client to share the list of the books the student has read over a period of time with the parents of the student. The report is coded using XML with the Jasper Report.



```
            }
 try{

String Report = "C:\\Users\\VAIBHAV\\Documents\\NetBeansProjects\\library_mangament_software\\src\\main\\java\\My_Forms\\ProgressReport.jrxml";
 JasperReport jr = JasperCompileManager.compileReport(Report);
 JasperPrint jp = JasperFillManager.fillReport(jr, null,dbconnect.librarycon);
 JasperViewer.viewReport(jp);
}
catch(Exception e){
 e.printStackTrace();
        JOptionPane.showMessageDialog(null, e);
}
```

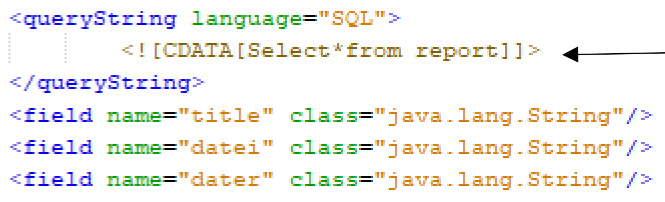
This is the address to the local file

Figure 47 – This is the code for initiating the progress report generation to a local file



```
<?xml version="1.0" encoding="UTF-8"?>
<jasperReport xmlns="http://jasperreports.sourceforge.net/jasperreports" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="http://jasperreports.sourceforge.net/jasperreports http://jasperreports.sourceforge.net/xsd/jasperreport.xsd"
              name="ProgressReport" pageWidth="842" pageHeight="595" orientation="Landscape" columnWidth="802" leftMargin="20" rightMargin="20"
              topMargin="20" bottomMargin="20" >
      <property name="ireport.zoom" value="1.0"/>
      <property name="ireport.x" value="0"/>
      <property name="ireport.y" value="0"/>
      <style name="Title" forecolor="#FFFFFF" fontName="Times New Roman" fontSize="50" isBold="false" pdfFontName="Times-Bold"/>
      <style name="SubTitle" forecolor="#CCCCCC" fontName="Times New Roman" fontSize="18" isBold="false" pdfFontName="Times-Roman"/>
      <style name="Column header" forecolor="#666666" fontName="Times New Roman" fontSize="14" isBold="true" pdfFontName="Times-Roman"/>
      <style name="Detail" mode="Transparent" fontName="Times New Roman" pdfFontName="Times-Roman"/>
      <style name="Row" mode="Transparent" fontName="Times New Roman" pdfFontName="Times-Roman"/>
```

Figure 48 – This is the code for setting up the layout of the report

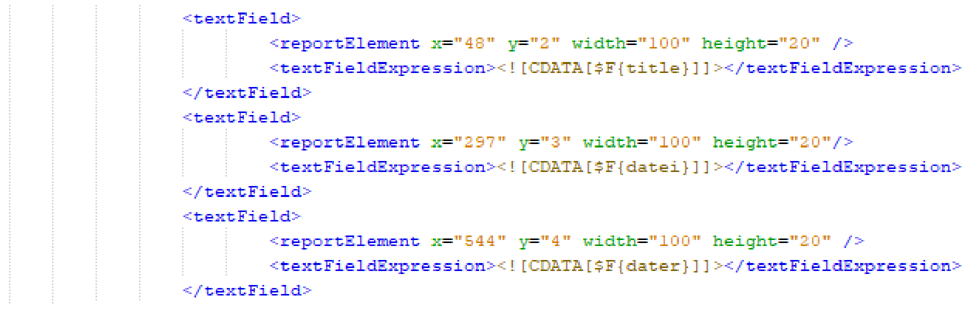


```
<queryString language="SQL">
        <![CDATA[Select*from report]]>
</queryString>
<field name="title" class="java.lang.String"/>
<field name="datei" class="java.lang.String"/>
<field name="dater" class="java.lang.String"/>
```

This statement is used to select data from the database

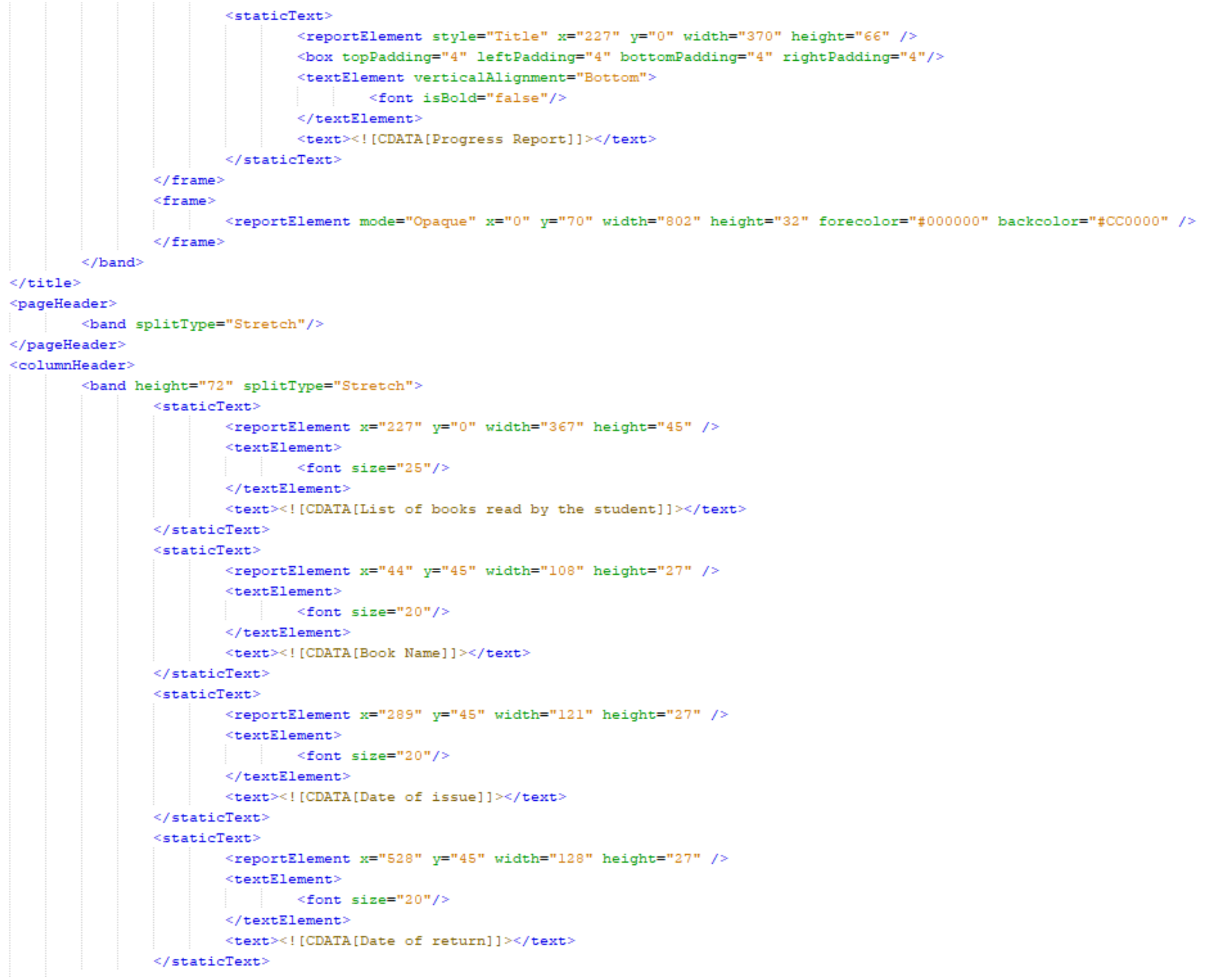
These statements are the fields of the report table in database

Figure 49 – This is the code for retrieving the information from the report field to the Progress Report



```
                <textField>
                        <reportElement x="48" y="2" width="100" height="20" />
                        <textFieldExpression><![CDATA[$F{title}]]></textFieldExpression>
                </textField>
                <textField>
                        <reportElement x="297" y="3" width="100" height="20"/>
                        <textFieldExpression><![CDATA[$F{datei}]]></textFieldExpression>
                </textField>
                <textField>
                        <reportElement x="544" y="4" width="100" height="20" />
                        <textFieldExpression><![CDATA[$F{dater}]]></textFieldExpression>
                </textField>
```

Figure 50 – This is the code for the text field for retrieving data from the database



```
                    <staticText>
                        <reportElement style="Title" x="227" y="0" width="370" height="66" />
                        <box topPadding="4" leftPadding="4" bottomPadding="4" rightPadding="4"/>
                        <textElement verticalAlignment="Bottom">
                            <font isBold="false"/>
                        </textElement>
                        <text><![CDATA[Progress Report]]></text>
                    </staticText>
                </frame>
                <frame>
                    <reportElement mode="Opaque" x="0" y="70" width="802" height="32" forecolor="#000000" backcolor="#CC0000" />
                </frame>
        </band>
</title>
<pageHeader>
        <band splitType="Stretch"/>
</pageHeader>
<columnHeader>
        <band height="72" splitType="Stretch">
                <staticText>
                    <reportElement x="227" y="0" width="367" height="45" />
                    <textElement>
                        <font size="25"/>
                    </textElement>
                    <text><![CDATA[List of books read by the student]]></text>
                </staticText>
                <staticText>
                    <reportElement x="44" y="45" width="108" height="27" />
                    <textElement>
                        <font size="20"/>
                    </textElement>
                    <text><![CDATA[Book Name]]></text>
                </staticText>
                <staticText>
                    <reportElement x="289" y="45" width="121" height="27" />
                    <textElement>
                        <font size="20"/>
                    </textElement>
                    <text><![CDATA[Date of issue]]></text>
                </staticText>
                <staticText>
                    <reportElement x="528" y="45" width="128" height="27" />
                    <textElement>
                        <font size="20"/>
                    </textElement>
                    <text><![CDATA[Date of return]]></text>
                </staticText>
```

Figure 51 – This is the code for the text labels under which the database values are printed with the heading and sub heading of the report.

Word count - 870