

Programing in Java

ITJA211 - 2017

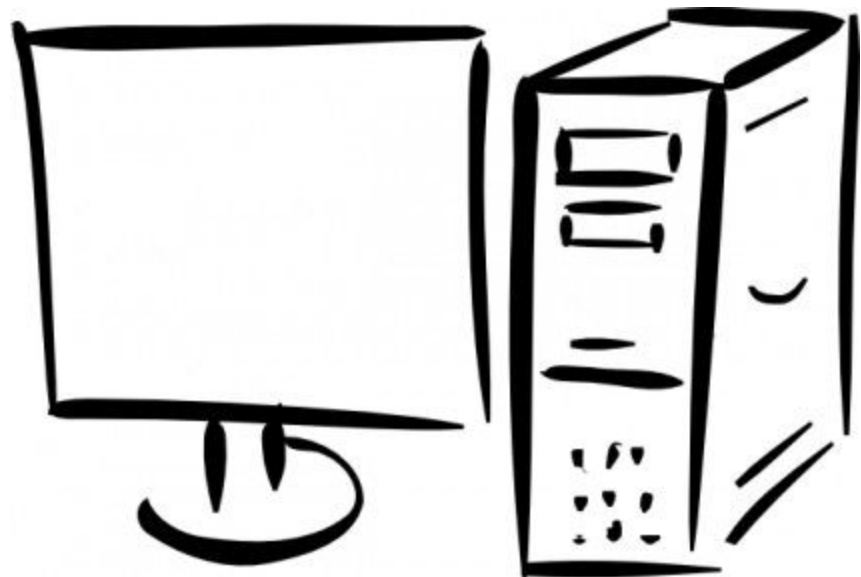


Table of Contents

Question 1	2
1.1	2
Question 2	8
2.1	8
Question 3	13
3.1	13
Appendices	14
Appendix A	14
Bibliography	15
Websites	16



Question 1

1.1

Please read through the README.txt that has been handed in along side this document. The images below are prototypes of possible user interfaces designs that make use of Nielsen's usability heuristics (see Appendix A). All Graphical User Interfaces (GUIs) are neat and professional. All JFrame were created by following examples set by Oracle (2015) and CodeJava (2015).

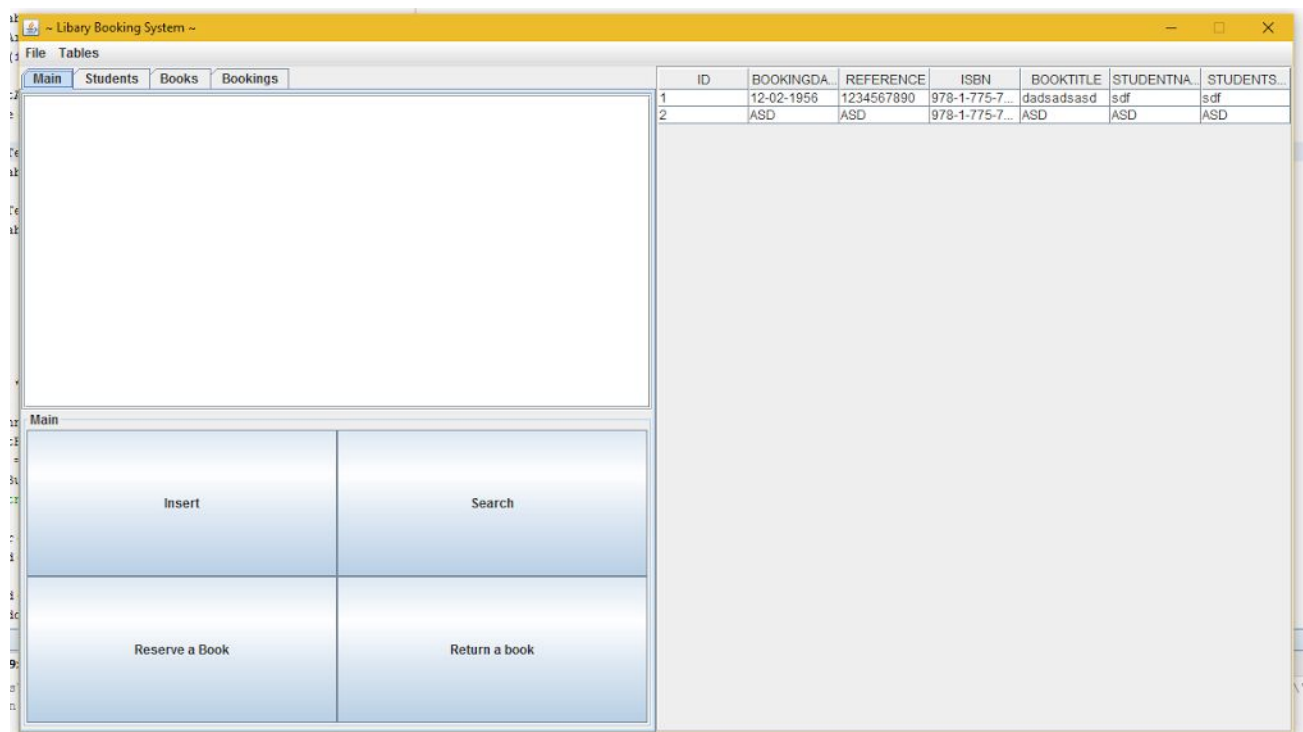
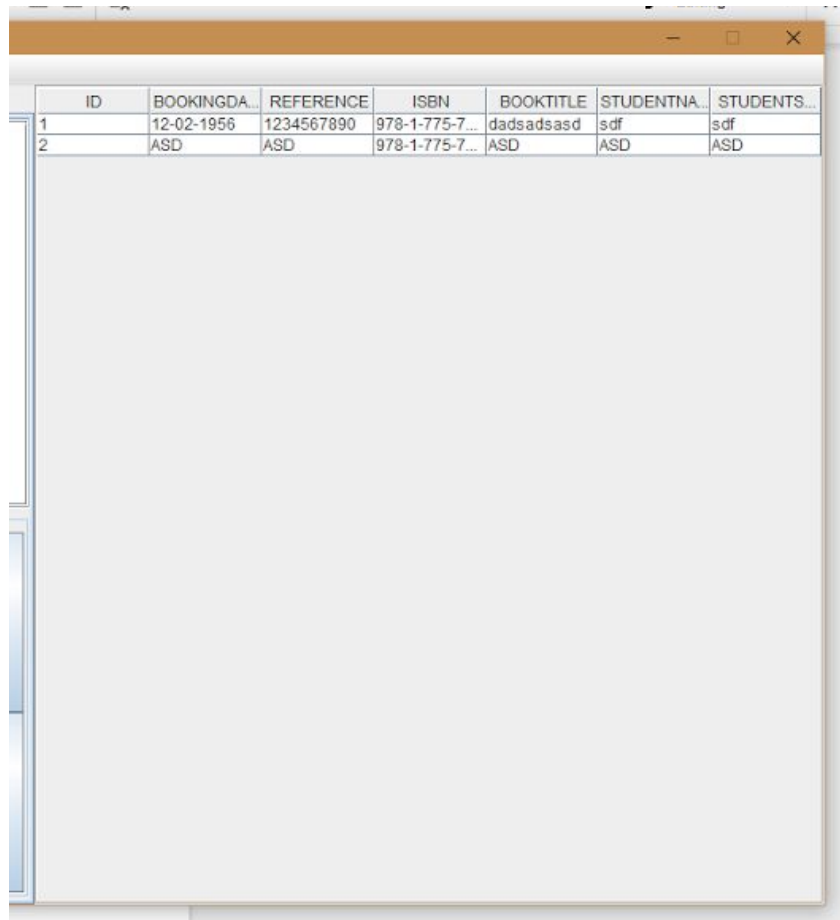


Figure 1 The opening screen of the program

This JFrame will act as the main/parent frame for the assignment. The main method will be held in the class named Main.java with an additional Question 3 specific main method will be in its own class.

The current bookings stored in the database will always be visible to the user through the bookings table as per the scenario.

A screenshot of a Java Swing window with a title bar containing standard OS controls (minimize, maximize, close). The window contains a JTable with 7 columns and 2 data rows. The columns are labeled: ID, BOOKINGDA..., REFERENCE, ISBN, BOOKTITLE, STUDENTNA..., and STUDENTS... The first row contains the values: 1, 12-02-1956, 1234567890, 978-1-775-7..., dadasadsasd, sdf, and sdf. The second row contains the values: 2, ASD, ASD, 978-1-775-7..., ASD, ASD, and ASD. The table is set against a light gray background.

ID	BOOKINGDA...	REFERENCE	ISBN	BOOKTITLE	STUDENTNA...	STUDENTS...
1	12-02-1956	1234567890	978-1-775-7...	dadasadsasd	sdf	sdf
2	ASD	ASD	978-1-775-7...	ASD	ASD	ASD

Figure 2 Bookings displayed in a JTable

The three screenshot that follow show three primary tabs of the program, namely: the students tab, the books tab and the bookings tab respectively.

From each tab the users is able to interact with the database is various ways. In each tab the user is able to insert to and update, search and delete from one of the three tables stored in the database.

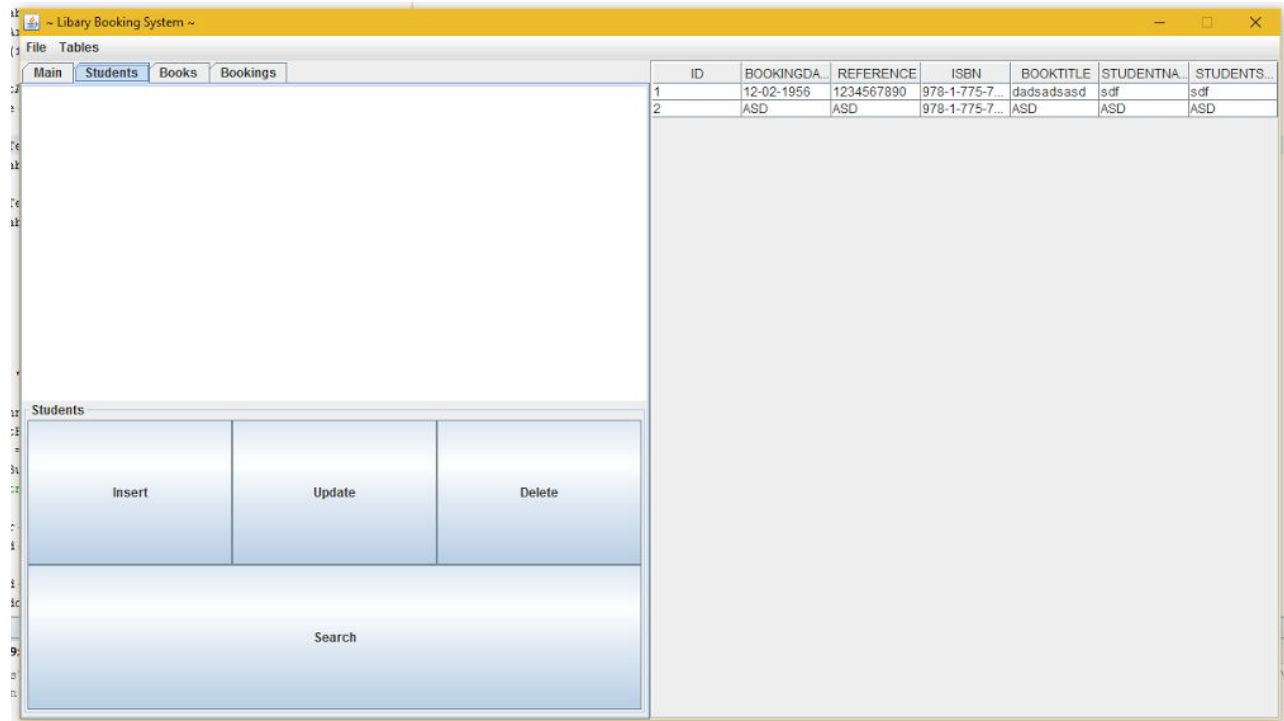


Figure 2 Student panel of the program

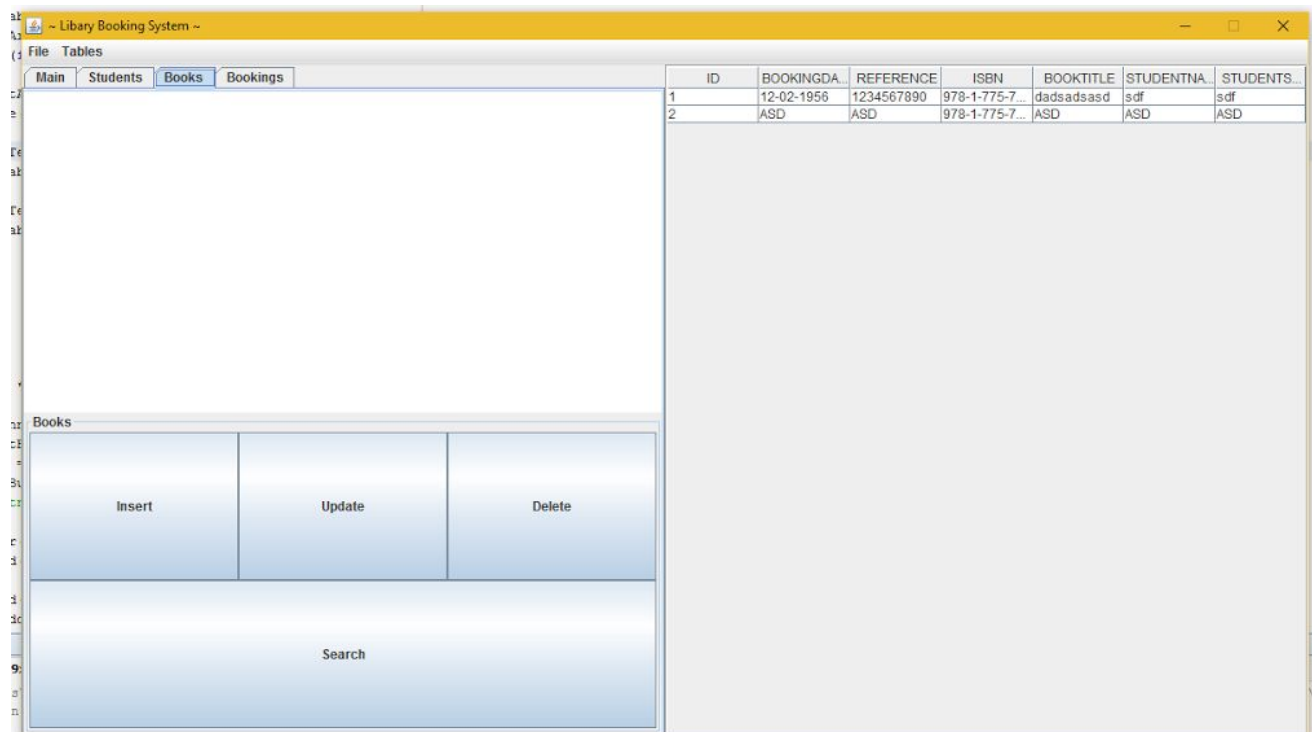


Figure 3 Books panel of the program

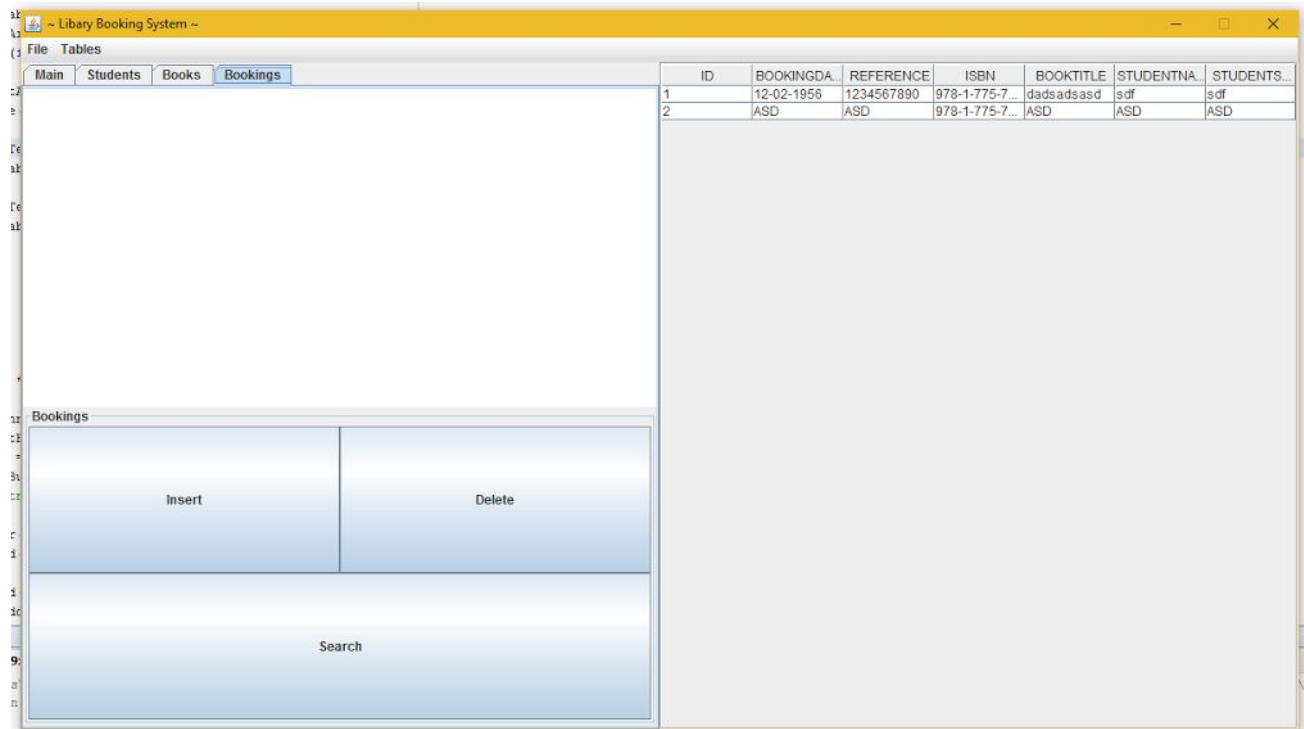


Figure 4 Bookings panel of the program

When users click on the 'Insert' button on the main page the following frame will be presented to them and prompt them to select what entity they wish to insert into the database.

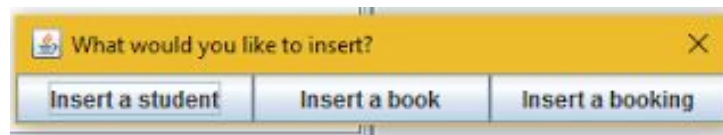
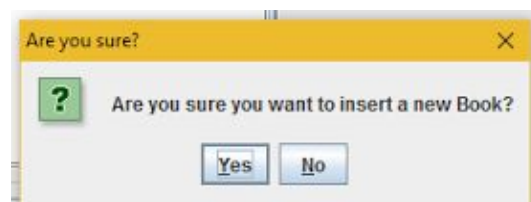
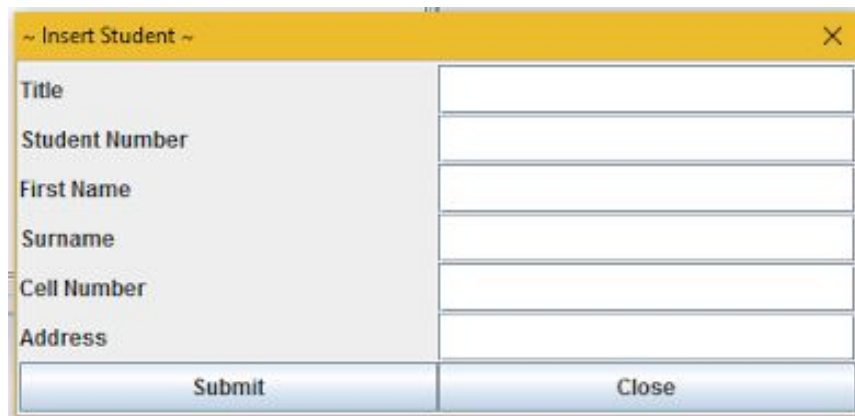


Figure 5 Search selection dialog

As indicated below, the program should always confirm the user's intent when interacting with the database.

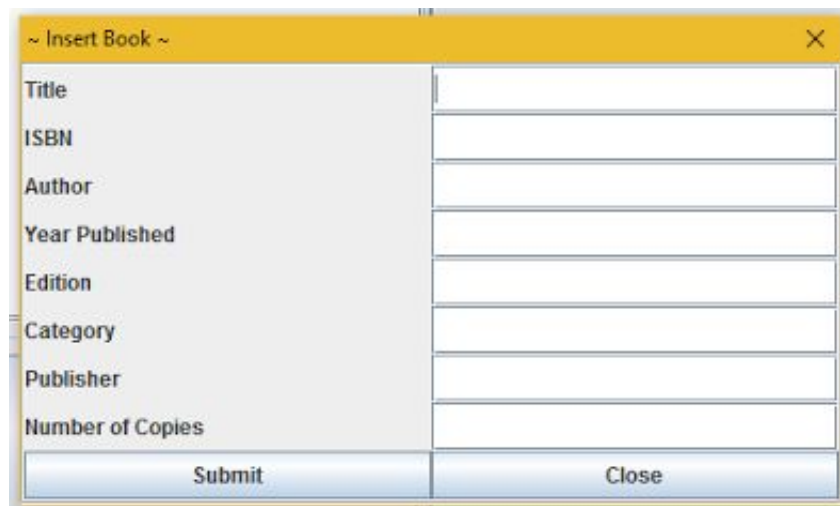


The following three frame are used by the program to capture data from the user in order to insert a specific record into the database.



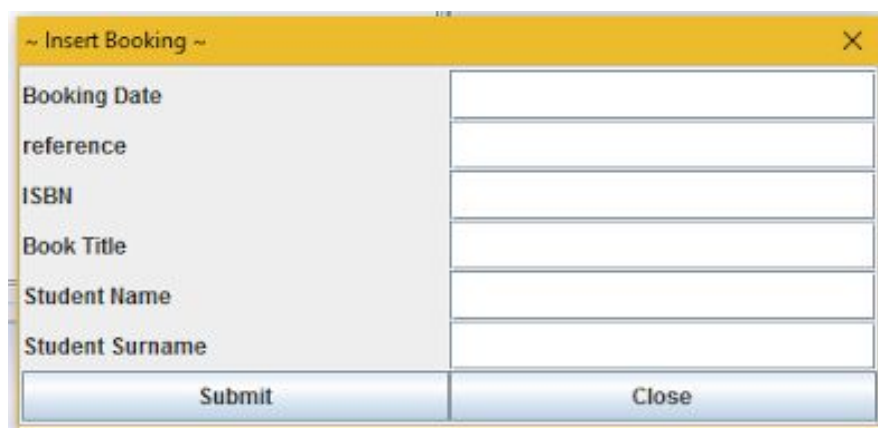
A screenshot of a Java Swing window titled "~ Insert Student ~" with a yellow title bar and a close button (X) in the top right corner. The window contains a list of labels on the left and corresponding text input fields on the right. The labels are: Title, Student Number, First Name, Surname, Cell Number, and Address. At the bottom of the window, there are two buttons: "Submit" and "Close".

Figure 6 Frame to capture the new student's details



A screenshot of a Java Swing window titled "~ Insert Book ~" with a yellow title bar and a close button (X) in the top right corner. The window contains a list of labels on the left and corresponding text input fields on the right. The labels are: Title, ISBN, Author, Year Published, Edition, Category, Publisher, and Number of Copies. At the bottom of the window, there are two buttons: "Submit" and "Close".

Figure 7 Frame to capture the new book's details



A screenshot of a Java Swing window titled "~ Insert Booking ~" with a yellow title bar and a close button (X) in the top right corner. The window contains a list of labels on the left and corresponding text input fields on the right. The labels are: Booking Date, reference, ISBN, Book Title, Student Name, and Student Surname. At the bottom of the window, there are two buttons: "Submit" and "Close".

Figure 8 Frame to capture the details of a new booking

Please notes that the following custom Button class was created for use throughout the project.

```
1 package booking.system.cmp;
2
3 import java.awt.event.ActionListener;
4 import java.util.Objects;
5 import javax.swing.JButton;
6
7 public class Button extends JButton {
8
9     private final String name;
10
11     public Button(String text, String name, String toolTipText, ActionListener parent) {
12         super(text);
13         this.name = name;
14
15         super.addActionListener(parent);
16         super.setToolTipText(toolTipText);
17     }
18
19     @Override
20     public String getName() {
21         return this.name;
22     }
23
24     @Override
25     public String toString() {
26         return "Button[" + getName() + "]";
27     }
28
29     @Override
30     public int hashCode() {
31         int hash = 5;
32         hash = 19 * hash + Objects.hashCode(this.name);
33         return hash;
34     }
35
36     @Override
37     public boolean equals(Object obj) {
38         if (obj instanceof Button) {
39             return ((Button) obj).getName().equals(getName());
40         }
41         return false;
42     }
43 }
```

Figure 9 Custom JButton code

Question 2

2.1

The SQL statements below are use to create the tables in the database and follow standards presented by W3Schools (2017).

```
1  /**
2   * Author: Matthew Van der Bijl - xq9x3wv31
3   * Created: 20 Feb 2017
4   */
5  CREATE TABLE Books (
6      id          INTEGER NOT NULL PRIMARY KEY GENERATED ALWAYS AS IDENTITY
7                  (START WITH 1, INCREMENT BY 1), /* PK */
8      title       VARCHAR(64),
9      ISBN        VARCHAR(32),
10     author       VARCHAR(64),
11     yearPublished INTEGER,
12     edition      INTEGER,
13     category     VARCHAR(32),
14     publisher    VARCHAR(32),
15     numCopies    INTEGER
16 );
17
18 CREATE TABLE Students (
19     id          INTEGER NOT NULL PRIMARY KEY GENERATED ALWAYS AS IDENTITY
20                 (START WITH 1, INCREMENT BY 1), /* PK */
21     title       VARCHAR(32),
22     studentNumber VARCHAR(64),
23     firstName   VARCHAR(64),
24     surname     VARCHAR(64),
25     cellNumber  VARCHAR(10),
26     address     VARCHAR(64)
27 );
28
29 CREATE TABLE Bookings (
30     id          INTEGER NOT NULL PRIMARY KEY GENERATED ALWAYS AS IDENTITY
31                 (START WITH 1, INCREMENT BY 1), /* PK */
32     bookingDate  VARCHAR(64),
33     reference    VARCHAR(64),
34     ISBN        VARCHAR(64),
35     bookTitle    VARCHAR(64),
36     studentName  VARCHAR(64),
37     studentSurname VARCHAR(32)
38 );
39
```

Figure 10 Statements used to create the three tables of the database

The insert statement below demonstrate how data is inserted into the bookings table, books table and students table respectively.

```
1 insert into Bookings (bookingDate, reference, ISBN, bookTitle, studentName, studentSurname)
2 values ('12-02-1956', '1234567890', '978-1-775-78603-0', 'dadsadsad', 'adf', 'adf');
3
4 insert into Books (title, ISBN, author, yearPublished, edition, category, publisher, numCopies)
5 values ('Basic Programming Principles', '978-1-775-78603-0', 'CM Pretorius and HG Erasmus', 2012, 2, 'textbook', 'Pearson Education', 5);
6
7 insert into Students (title, studentNumber, firstName, surname, cellNumber, address)
8 values ('Mrs.', 'J15T269DKR', 'Eugenie', 'Spore', '0836191089', '76 Elmwood Avenue');
```

Figure 11 Example insert statements

The 'dbConnect' method is used to connect to the database.

```
/**
 * Used to connect to the database.
 */
private void dbConnection() {
    System.out.println("dbConnection");
    try {
        this.conn = DriverManager.getConnection(
            "jdbc:derby://127.0.0.1:1527/LibraryBookingSystem", "nbuser", "nbuser"
        );
    } catch (SQLException sqle) {
        sqle.printStackTrace(System.err);
        JOptionPane.showMessageDialog(this, sqle.getMessage(), sqle.getClass()
            .getSimpleName(), JOptionPane.ERROR_MESSAGE);
        System.exit(2);
    }
}
```

Figure 12 The 'dbConnect' method

The 'dbSearch' method is used to search for a specific record in the database. The method of searching used was derived by examples given by ApexSQL (2013) and 1KeyData (2017).

```
/**
 * Used to call the different search messages.
 */
private void dbSearch() {
    System.out.println("dbSearch");

    String selection = JOptionPane.showInputDialog(this,
        "Please enter: 'student', 'book' or 'booking'",
        "What would you like to search for?", JOptionPane.QUESTION_MESSAGE);
    if (selection == null || selection.equals("")) {
        return; // clicked on the button by mistake perhaps
    }
    switch (selection.toLowerCase()) {
        case "student":
            this.searchStudent();
            break;
        case "book":
            this.searchBooks();
            break;
        case "booking":
            this.searchBooking();
            break;
        default:
            JOptionPane.showMessageDialog(this, "Please enter: 'student', "
                + "'book' or 'booking'", "Invalid Input", JOptionPane.ERROR_MESSAGE);
    }
}
```

Figure 13 The 'dbSearch' method

The 'booking' method is used to create a new booking record in the Bookings table of the database.

```
/**
 * Used for reserving books.
 */
@SuppressWarnings("empty-statement")
private void booking() {
    InsertBooking booking;
    try {
        booking = new InsertBooking(this);
        booking.setVisible(true);
        while (!booking.hasSelected());
    } catch (RuntimeException ex) {
        System.err.println(ex);
        return;
    }

    String bookingDate = booking.getBookingDate();
    String reference = booking.getReference();
    String ISBN = booking.getISBN();
    String bookTitle = booking.getBookTitle();
    String studentName = booking.getStudentName();
    String studentSurname = booking.getStudentSurname();

    String q1 = String.format("INSERT INTO Bookings (bookingDate, reference, ISBN, bookTitle, studentName, studentSurname) "
        + "VALUES ('%s', '%s', '%s', '%s', '%s', '%s')", bookingDate, reference, ISBN, bookTitle, studentName, studentSurname);
    System.out.println(q1);

    String q2 = String.format("UPDATE Books SET numCopies = numCopies - 1 WHERE ISBN='%s'", ISBN);
    System.out.println(q2);

    try {
        Statement stmt = conn.createStatement();

        if (stmt.execute(q1) && stmt.execute(q2)) {
            JOptionPane.showMessageDialog(this, "An error occurred",
                "Error.", JOptionPane.ERROR_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(this, "The new booking has been inserted.",
                "Success!", JOptionPane.PLAIN_MESSAGE);
        }
        stmt.close();
    } catch (SQLException sqle) {
        sqle.printStackTrace(System.err);
        JOptionPane.showMessageDialog(this, sqle.getMessage(), sqle.getClass()
            .getSimpleName(), JOptionPane.ERROR_MESSAGE);
    }

    this.refreshTable();
}
```

Figure 14 The 'booking' method

The 'dbInsert' method is called when the user wish to enter general data into the database by calling the different insert methods, namely: 'insertBooking', 'insertBook' and 'insertStudent'.

```
/**
 * Used to call the different insert methods.
 */
@SuppressWarnings("empty-statement")
private void dbInsert() {
    System.out.println("dbInsert");

    InsertDialog dailog = new InsertDialog(this);
    dailog.setVisible(true);
    while (dailog.isSelecting());

    Tables selection;
    System.out.println(selection = dailog.getSelection());

    switch (selection) {
        case TBL_BOOKINGS: {
            this.insertBooking();
        }
        break;
        case TBL_BOOKS: {
            this.insertBook();
        }
        break;
        case TBL_STUDENTS: {
            this.insertStudent();
        }
        break;
    }
}
```

Figure 15 The 'dbInsert' method

Question 3

3.1

The code below is a possible method of using a parameterized method to reverse an integer array. It was formulated by referring to the explanations and examples given by Parlante (2012), Sedgewick & Wayne (2016) and Adamchik (2009).

```
1 package question3;
2
3 /**
4  * Write a method called 'reverse' that returns an array which is the reverse of
5  * the one it receives and then write the main method that uses your 'reverse'
6  * method to print the reverse of the number array below.
7  *
8  * @author Matthew Van der Bijl - XQ9X3WV31
9  */
10 public class Question3 {
11
12     public Question3() {
13     }
14
15     /**
16      * Method that returns an array which is the reverse of the one it receives.
17      *
18      * @param arr the array to be reversed
19      * @return the reversed array
20      */
21     public int[] reverse(int[] arr) {
22         int[] tmp = new int[arr.length];
23
24         for (int i = 0; i < arr.length; i++) {
25             tmp[arr.length - i - 1] = arr[i];
26         }
27
28         return tmp;
29     }
30
31     /**
32      * @param args arguments from the command line
33      */
34     public static void main(String[] args) {
35         int[] numbers = {34, 78, 2, 4, 5};
36         Question3 obj = new Question3();
37
38         int[] reverse = obj.reverse(numbers);
39
40         for (int i = 0; i < reverse.length; i++) {
41             System.out.println(reverse[i]);
42         }
43     }
44 }
45
```

Figure 16 Possible solution to Question 3

Appendices

Appendix A

Nielsen's usability heuristics are a set of high-level design principles that are used to evaluate a user-interface to determine whether the interfaces conforms to well established and tested design principles (Preece et al. 2015). A revised version of Nielsen's heuristics are listed below:

- 1) The system should always provide the user with feedback;
- 2) The systems should match the user perception of the real world;
- 3) The system should provide the user with a clear exit to an unwanted state;
- 4) The systems should maintain the same standard and a high a level of consistency throughout;
- 5) The system should seek to eliminate error-prone conditions;
- 6) The system should minimise the load on the user's memory by making options and potential actions more visible;
- 7) The system must be efficient to use;
- 8) The system should make use of a minimalist aesthetic;
- 9) The system should aid users to recognise potential errors and assist the user in accounting for them; and
- 10) The system should provide the user with help documentation (Preece et al. 2015).

These heuristics are intended to be used by designers to compare evaluate their designs and change their design accordingly. Each iteration of the design should seek to use these guidelines to solve usability problems (Stair & Reynolds 2015).

Bibliography

- 1KeyData, 2017. SQL - WHERE Clause Available at:
<http://www.1keydata.com/sql/sqlwhere.html>
[Accessed March 28, 2017].
- Adamchik, V.S., 2009. Arrays. *Carnegie Mellon School of Computer Science*. Available at:
<https://www.cs.cmu.edu/~adamchik/15-121/lectures/Arrays/arrays.html>
[Accessed March 18, 2017].
- ApexSQL, 2013. How to quickly search for SQL database data and objects. *Solution center*. Available at:
<https://solutioncenter.apexsql.com/quickly-search-for-sql-database-data-and-objects/>
[Accessed March 16, 2017].
- CodeJava, 2015. JFrame basic tutorial and examples. Available at:
<http://www.codejava.net/java-se/swing/jframe-basic-tutorial-and-examples>
[Accessed March 28, 2017].
- Oracle, 2015. How to Make Frames (Main Windows) Available at:
<https://docs.oracle.com/javase/tutorial/uiswing/components/frame.html> [Accessed February 28, 2017].
- Parlante, N., 2012. Java Arrays and Loops. *CodingBat*. Available at:
<http://codingbat.com/doc/java-array-loops.html>
[Accessed March 1, 2017].
- Preece, J., Rogers, Y. & Sharp, H., 2015. *Interaction Design: Beyond Human-Computer Interaction*, John Wiley & Sons.
- Sedgewick, R. & Wayne, K., 2016. Arrays. *Introduction to Programming in Java*. Available at:
<http://introcs.cs.princeton.edu/java/14array/>
[Accessed March 28, 2017].
- Stair, R. & Reynolds, G., 2015. *Principles of Information Systems*, Cengage Learning.
- W3Schools, 2017. SQL CREATE TABLE Statement. *W3Schools*. Available at:
https://www.w3schools.com/sql/sql_create_table.asp
[Accessed March 3, 2017].

Websites

The list below is of all website consulted throughout the physical coding of this project.

1. <http://docs.oracle.com/javase/tutorial/uiswing/components/menu.html>
2. <http://stackoverflow.com/questions/11627279/running-java-helloworld>
3. <http://stackoverflow.com/questions/19271493/subtract-1-from-the-number-in-a-row-sql-query>
4. <http://stackoverflow.com/questions/25163805/jpanel-doesnt-update-when-adding-component-in-another-class>
5. <http://stackoverflow.com/questions/32941499/how-do-i-add-jbutton-to-jframe>
6. <http://stackoverflow.com/questions/3732422/select-from-nothing>
7. <https://coderanch.com/t/333247/java/JPanel-TitledBorder>
8. <https://docs.oracle.com/javase/tutorial/uiswing/components/border.html>
9. <https://docs.oracle.com/javase/tutorial/uiswing/components/button.html>
10. [https://technet.microsoft.com/en-us/library/ms189245\(v=sql.105\).aspx](https://technet.microsoft.com/en-us/library/ms189245(v=sql.105).aspx)
11. http://www.w3schools.com/sql/sql_update.asp
12. <https://www.youtube.com/watch?v=706Ye4ubtEY>