LONDON METROPOLITAN UNIVERSITY

Islington college
(इस्लिङ्टन कलेज)

CS5003NI Data Structure and Specialist Programming

30% Individual Coursework

2023-24 Autumn

Student Name: Aakarshan Khadka

London Met ID: 22085855

College ID: NP01AI4S230002

Assignment Due Date: Friday, January 12, 2024

Assignment Submission Date: Friday, January 12, 2024

Word Count: 0

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

Table of Contents

Table Of Figures

Table Of Table

1. Introduction

Step into the realm of LeBalls, an exclusive database tailored for Los Angeles Lakers enthusiasts. Your gateway to seamless Lakers roster management and insightful stats exploration, LeBalls transcends conventional databases. It's your hands-on toolkit, offering the precision of a seasoned coach. This project simplifies player management, focusing entirely on the Lakers roster.

Explore the intricacies with LeBalls' intuitive sorting features. Whether it's 'LeName,' 'LePoints,' or other 'LeInformation' (pun intended), navigate Lakers stats effortlessly. Picture it as a practical tool for tracking LeBron James and his teammates.

So, immerse yourself in LeBalls, the no-nonsense platform for Lakers aficionados. Each click is a direct engagement with Lakers' data, a straightforward avenue for basketball insights. Keep the spirit of basketball alive with LeBalls – more than just a database, it's a personalized journey into Lakers' realm.

1.1 Introduction to JAVA

Java, an object-oriented programming language founded by James Gosling, made its debut at Sun Microsystems in 1995 before Sun's merger with Oracle Corporation. Renowned for its platform independence, Java operates on the "write once, run anytime" principle, allowing code to execute on diverse platforms without recompilation, provided a Java interpreter is present. The language, evolving rapidly, reached its SE 20 version in March 2023.

Java's key features include object-oriented structure, platform independence through the Java Virtual Machine (JVM), automatic memory management, and a compiled-interpreted hybrid nature. Its syntax, derived from C++, is case-sensitive, akin to its precursor. Java GUI, or Graphical User Interface, transforms intricate technical processes into user-friendly visuals using buttons, icons, and menus. This visual interaction method is prevalent in mobile apps, software, and web interfaces, simplifying user engagement with digital systems.

1.2 Aim

The primary goal of this project is to establish a robust database to effectively monitor LA Lakers players. By enabling users to seamlessly add, update, or delete player records, the aim is to ensure the database remains consistent in real-time. The implementation of input constraints will be pivotal in upholding data integrity, providing a reliable foundation to prevent inconsistencies.

Furthermore, the focus is on enhancing user experience through planned functionalities. Users should be able to intuitively sort the player list and effortlessly find specific details. These enhancements are envisioned to make the database not only more user-friendly but also to improve overall efficiency. The project is geared towards creating a user-centric database that adeptly manages the intricate details of LA Lakers' player information.

1.3 Objectives

1.3.1 JFrame GUI Design

Design an aesthetically pleasing and user-friendly JFrame interface, incorporating components like panels, buttons, text fields, tables, and labels for effective data presentation and interaction.

1.3.2 Data Validation and Exception Handling

Incorporate thorough data validation checks in the Java code to ensure the integrity and correctness of user inputs. Implement error handling mechanisms to gracefully manage and communicate errors, providing clear feedback to users when data issues arise.

1.3.3 Manage Team Data

Enable operations such as add, update or delete a player through the JFrame interface to allow the users to Manage Team Data

1.4 Tools Used

1.4.1 Draw.io

Draw.io is a user-friendly online tool for creating a variety of diagrams. This tool was used to make the class diagrams for this coursework.

Figure 1: Draw.io Logo

1.4.2 Apache NetBeans

NetBeans is a free and open-source platform and integrated development environment (IDE). This tool was used for the development and running the code.

Figure 2: NetBeans logo

2. Algorithms Utilized

2.1 Merge Sort

Merge sort is a sorting algorithm that is highly efficient. It uses the divide-and-conquer approach by systematically dividing an array into smaller halves, independently sorting them and then merging the sorted halves back to a single sorted array.

2.1.1Why Merge Sort?

- Stability: Merge sort preserves the order of the original elements which is essential in this case for users to want to sort with preference such as Name, or Statistics.
- Consistency: Merge Sort exhibits a predictable behaviour that leads to consistent and deterministic results.
- Scalability: Merge Sort is known to perform well across databases with various sizes which is vital to this work, with the option to add or delete players.

2.1.2 Pseudo Code

CREATE CLASS MergeSort

CREATE METHOD mergeSort **RETURNS** ArrayList<ModelClass> WITH PARAMETERS arr, sortBy

DO

IF length(arr) is less than or equal to 1

RETURN arr

END IF

mid = length(arr) divided by 2

left = mergeSort(sublist(arr, 0, mid), sortBy)

right = mergeSort(sublist(arr, mid, length(arr)), sortBy)

RETURN merge(left, right, sortBy)

END DO

```
CREATE METHOD merge RETURNS ArrayList<ModelClass> WITH PARAMETERS left, right, sortBy
DO
    CREATE ArrayList<ModelClass> result
    DECLARE leftIndex, rightIndex

    WHILE leftIndex is less than length(left) AND rightIndex is less than length(right)
        leftModel = left[leftIndex]
        rightModel = right[rightIndex]
        comparison = getComparisonResult(leftModel, rightModel, sortBy)

        IF comparison is less than or equal to 0
            ADD leftModel TO result
            leftIndex increment by 1
        END IF
        ELSE
            ADD rightModel TO result
            rightIndex increment by 1
        END ELSE
    END WHILE

    ADD remaining elements of left TO result
    ADD remaining elements of right TO result

    RETURN result
END DO
```

```
CREATE METHOD getComparisonResult RETURNS int
WITH PARAMETERS model1, model2, sortBy

DO

    IF sortBy is "Points Per Game"

        pointsPerGame1 =
parseFloat(model1.getPointsPerGame())

        pointsPerGame2 =
parseFloat(model2.getPointsPerGame())

        RETURN compareFloat(pointsPerGame1,
pointsPerGame2)

    END IF


    ELSE IF sortBy is "Assists Per Game"

        assistsPerGame1 =
parseFloat(model1.getAssistsPerGame())

        assistsPerGame2 =
parseFloat(model2.getAssistsPerGame())

        RETURN compareFloat(assistsPerGame1,
assistsPerGame2)

    END ELSE IF


    ELSE IF sortBy is "Name"

        RETURN compareString(model1.getName(),
model2.getName())

    END ELSE IF


    ELSE IF sortBy is "Jersey Number"

        jerseyNumber1 = parseInt(model1.getJerseyNumber())

        jerseyNumber2 = parseInt(model2.getJerseyNumber())

        RETURN compareInt(jerseyNumber1, jerseyNumber2)

    END ELSE IF
```

ELSE

THROW NEW IllegalArgumentException("Invalid sorting criteria: " + sortBy)

END ELSE

END DO

2.2 Binary Search

Binary Search is a search algorithm designed for efficiently locating a specific element within a sorted array. It follows a systematic approach of repeatedly dividing the array in half and comparing the target value with the middle element. If a match is found, the search concludes successfully; otherwise, it continues in the lower or upper half of the array based on the comparison result.

2.2.1 Why Binary Search?

- Integration: As Merge Sort was used in the project already, binary search aligns well with a sorted array resulting from Merge Sort.
- Space Complexity: Binary Search operated directly on the existing sorted array, minimizing the need for additional data structures.
- Consistency: Binary Search exhibits a predictable behaviour that leads to consistent and deterministic results.

2.2.2 Pseudo Code

CREATE class BinarySearch

DO

CREATE method searchString **WITH PARAMETERS** objectHolder, low, high, searchKey, positionHolder, field

DO

DECLARE SearchField

SearchField = ModelClass.getSearchField()

IF high is greater than or equal to low

mid = (low + high) / 2

midValue = ""

```
        SWITCH SearchField
            CASE "Search by Name"
                midValue = objectHolder.get(mid).getName()
            END CASE

            CASE "Search by Jersey Number"
                midValue =
objectHolder.get(mid).getJerseyNumber()
            END CASE
        END SWITCH

        IF midValue is equal to searchKey
            positionHolder.add(mid)
            CALL searchString WITH PARAMETERS
objectHolder, low, mid - 1, searchKey, positionHolder, field
            CALL searchString WITH PARAMETERS
objectHolder, mid + 1, high, searchKey, positionHolder, field
        ELSE IF midValue is greater than searchKey
            CALL searchString WITH PARAMETERS
objectHolder, low, mid - 1, searchKey, positionHolder, field
        ELSE
            CALL searchString WITH PARAMETERS
objectHolder, mid + 1, high, searchKey, positionHolder, field
        END ELSE
    END IF
  END DO
END DO
```

3. Class Diagram

3.1 LeBalls

Figure 3 : Class Diagram (LeBalls)

3.2 ModelClass

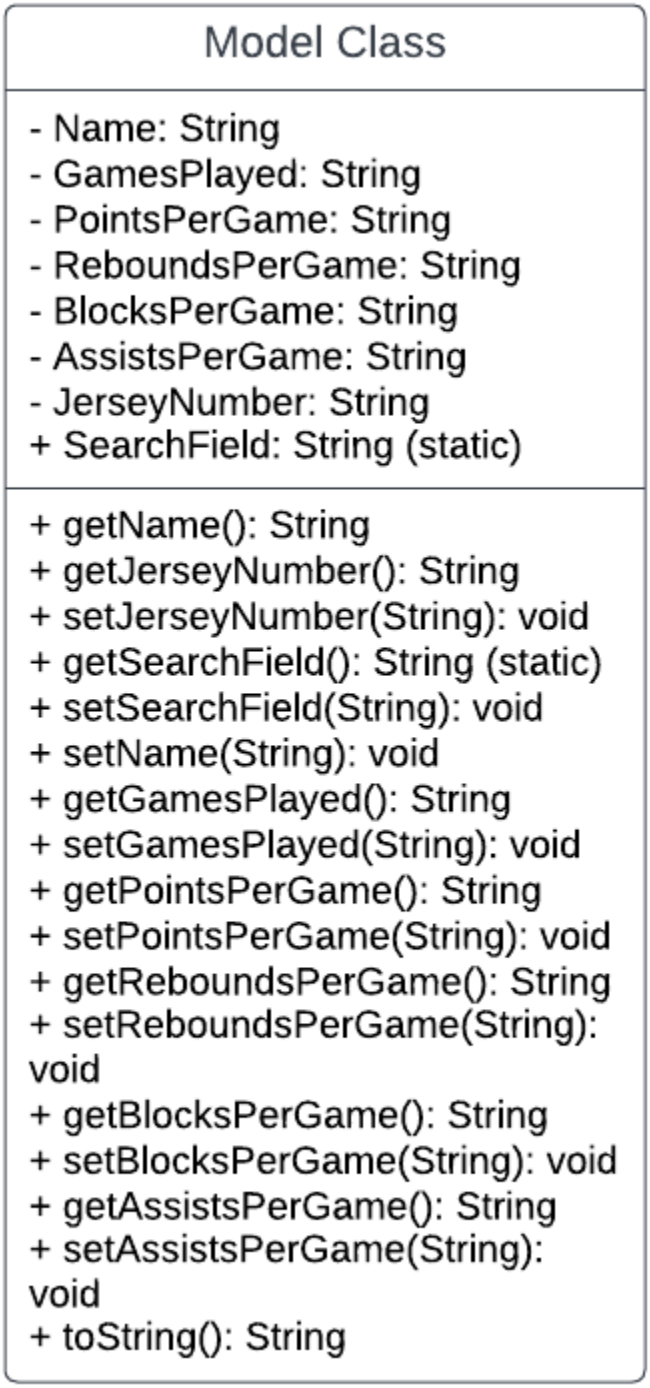


Figure 4: Class Diagram (ModelClass)

3.3 BinarySearch

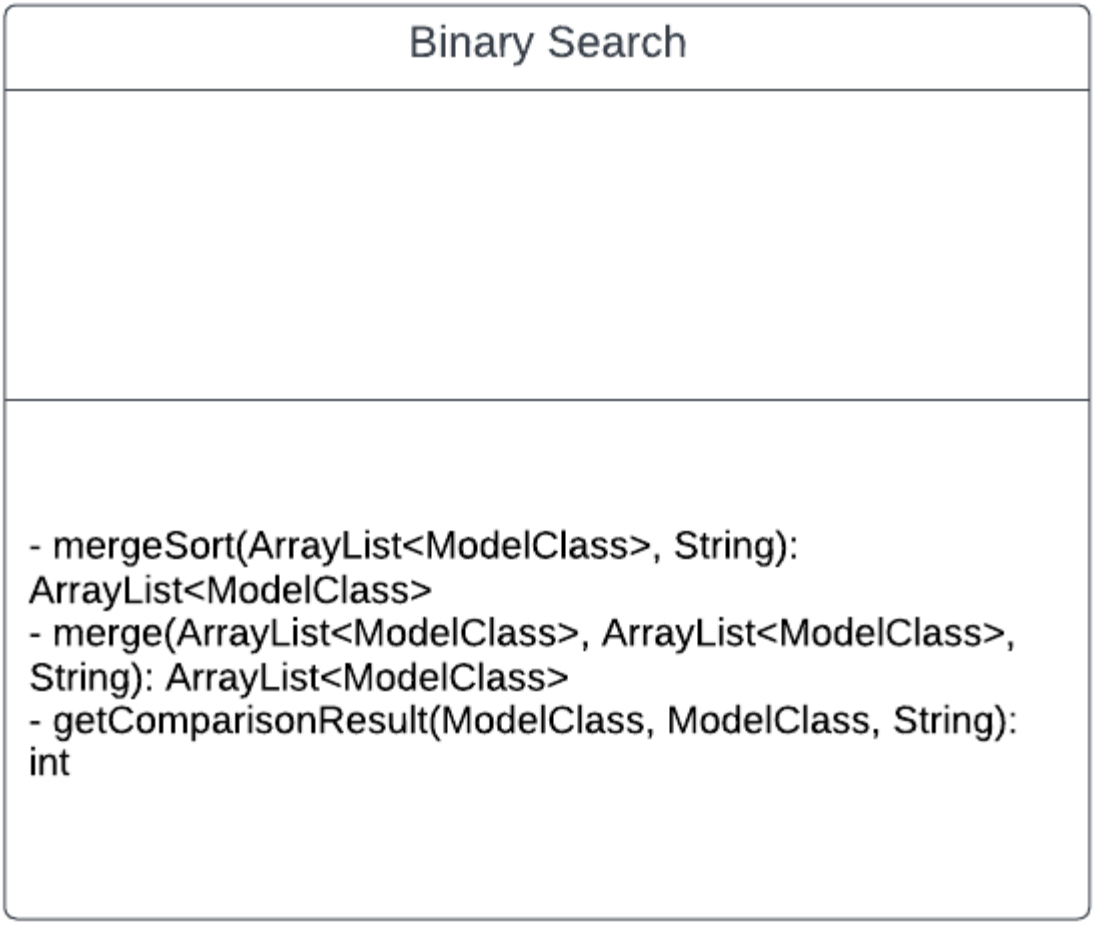


Figure 5:Class Diagram (BinarySearch)

3.4 MergeSort

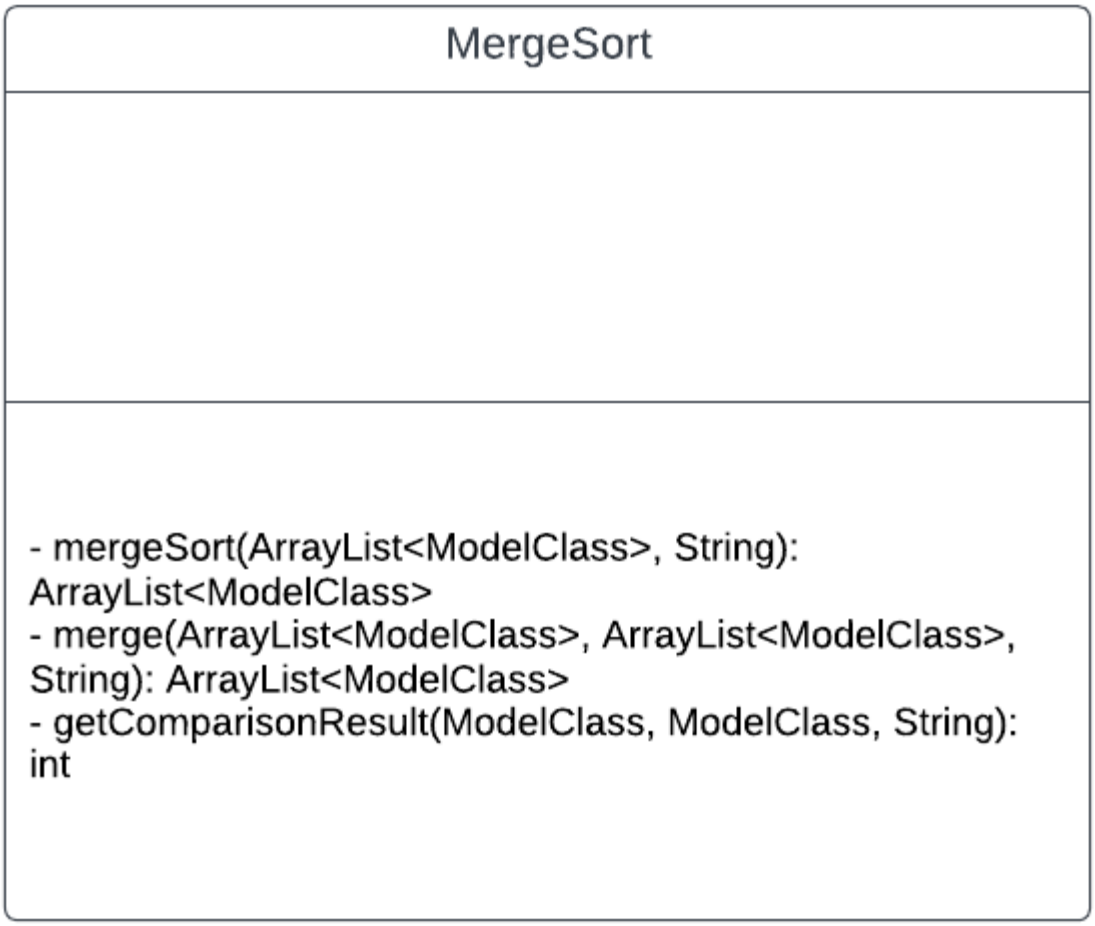


Figure 6:Class Diagram (MergeSort)

3.5 StringUtil

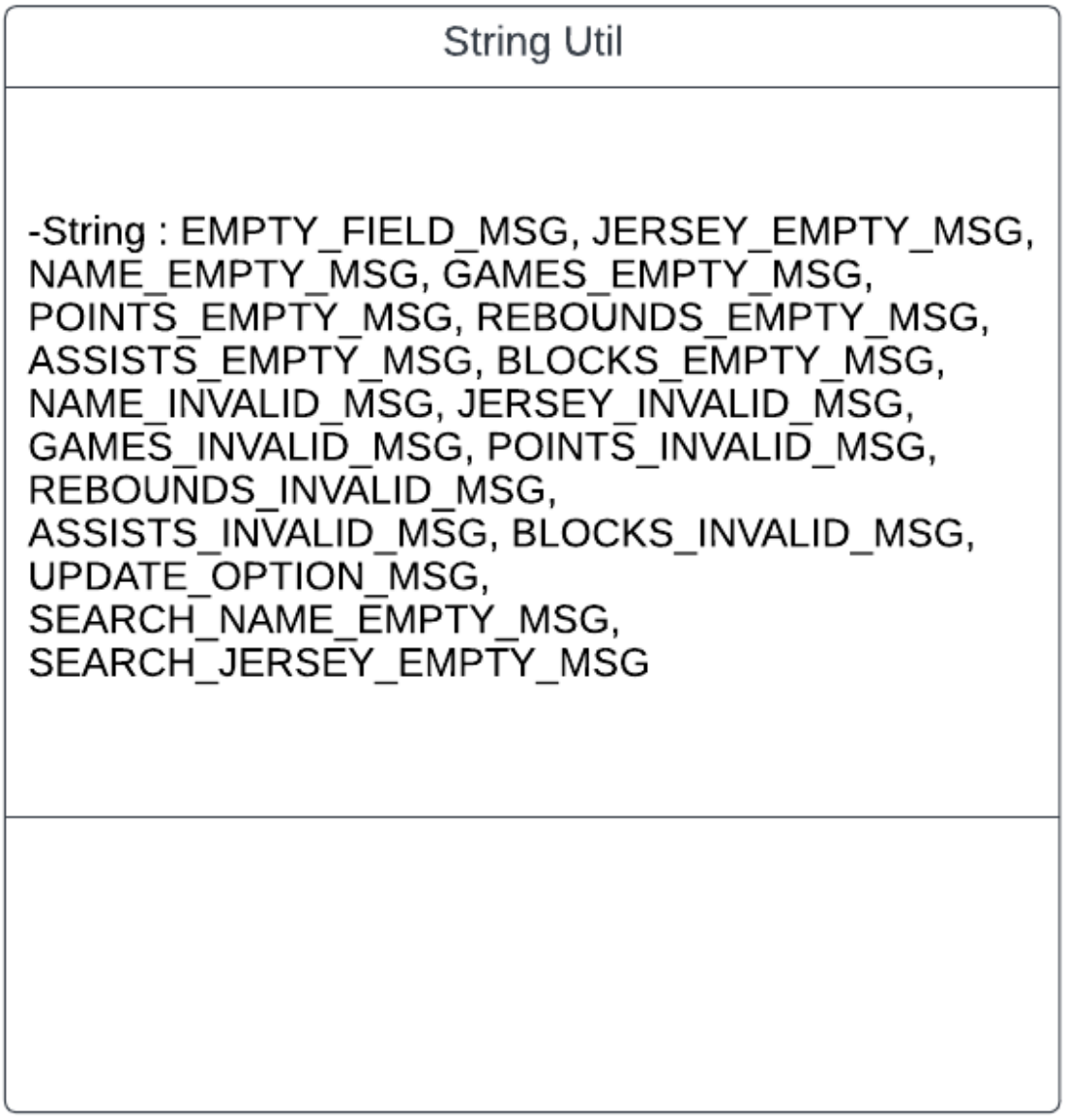


Figure 7:Class Diagram (StringUtil)

4. Method Description

4.1 Class LeBalls

4.1.1 arrayList()

This method populates the array with the values in the table.

```
public void arrayList() {
    arLeballs.clear();
    for (int i = 0; i < tbMainTable.getRowCount(); i++) {
        int JerseyNumber = Integer.parseInt((String) tbMainTable.getValueAt(row:i, column:0));
        String Name = (String) tbMainTable.getValueAt(row:i, column:1);
        String GamesPlayed = (String) tbMainTable.getValueAt(row:i, column:2);
        float PointsPerGame = Float.parseFloat((String) tbMainTable.getValueAt(row:i, column:3));
        float ReboundsPerGame = Float.parseFloat((String) tbMainTable.getValueAt(row:i, column:4));
        float AssistsPerGame = Float.parseFloat((String) tbMainTable.getValueAt(row:i, column:5));
        float BlocksPerGame = Float.parseFloat((String) tbMainTable.getValueAt(row:i, column:6));

        // Using model class constructor
        ModelClass model = new ModelClass();
        model.setJerseyNumber(JerseyNumber: String.valueOf(i: JerseyNumber));
        model.setName(Name);
        model.setGamesPlayed(GamesPlayed);
        model.setPointsPerGame(PointsPerGame:String.valueOf(f: PointsPerGame));
        model.setReboundsPerGame(ReboundsPerGame: String.valueOf(f: ReboundsPerGame)); // Convert float to String
        model.setAssistsPerGame(AssistsPerGame: String.valueOf(f: AssistsPerGame)); // Convert float to String
        model.setBlocksPerGame(BlocksPerGame:String.valueOf(f: BlocksPerGame)); // Convert float to String

        // Add the ModelClass instance to the cine_list_model ArrayList
        arLeballs.add(e: model);

    }
}
```

Figure 8: arrayList();

4.1.2 bnBackButtonA1ActionPerformed(evt)

This method sets the visibility of update panel to false and the visibility of main panel to true.

```
private void bnBackButtonA1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    pnMainPanel.setVisible(aFlag: true);
    pnUpdatePanel.setVisible(aFlag: false);
}
```

Figure 9: bnBackButtonA1ActionPerformed

4.1.3 bnBackButtonAddActionPerformed(evt)

This method sets the visibility of add panel to false and the visibility of main panel to true.

```
private void bnBackButtonAddActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    pnMainPanel.setVisible(aFlag: true);
    pnAddPanel.setVisible(aFlag: false);

}
```

Figure 10:bnBackButtonAddActionPerformed

4.1.4 buAddButtonActionPerformed(evt)

This method sets the visibility of add panel to true and the visibility of main panel to false while also setting the default value for all the text fields in the add panel.

```
private void buAddButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    pnAddPanel.setVisible(aFlag: true);
    pnAddPanel.setBounds(x: 0, y: 0, width: 950, height:480);
    pnMainPanel.setVisible(aFlag: false);

    tfJN.setText(t: "");
    tfName.setText(t: "");
    tfGP.setText(t: "");
    tfRPG.setText(t: "");
    tfBPG.setText(t: "");
    tfAPG.setText(t: "");
    tfPPG.setText(t: "");


}
```

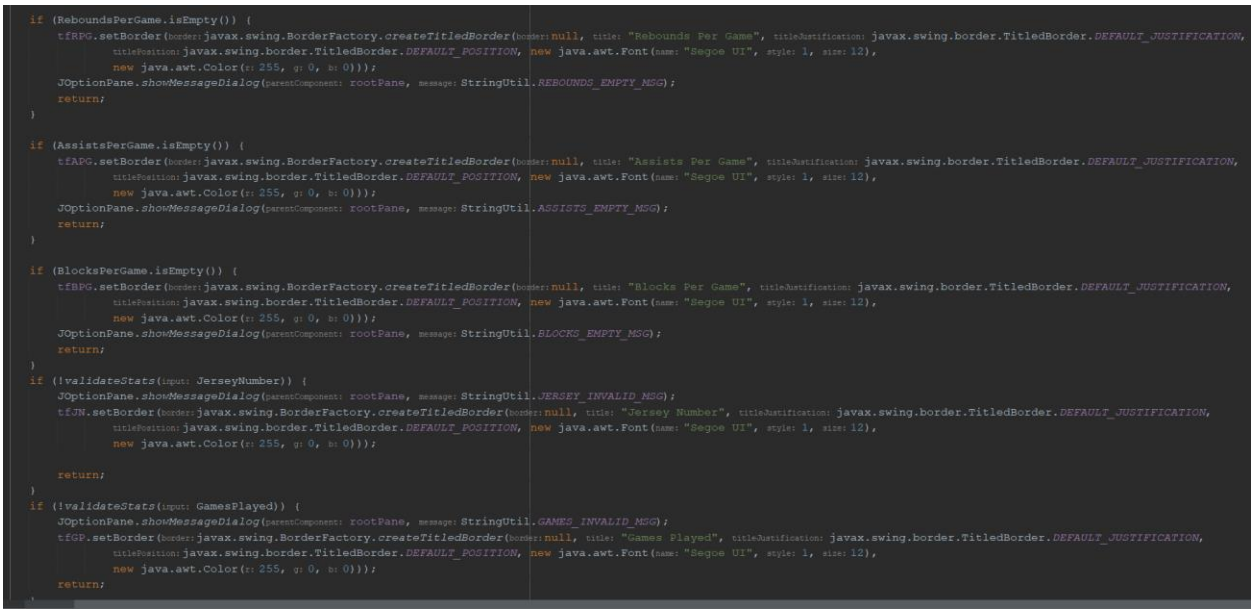
Figure 11:buAddButtonActionPerformed

4.1.5 buAddInTableActionPerformed(evt)

This method adds all the user input values into the table with proper validation.

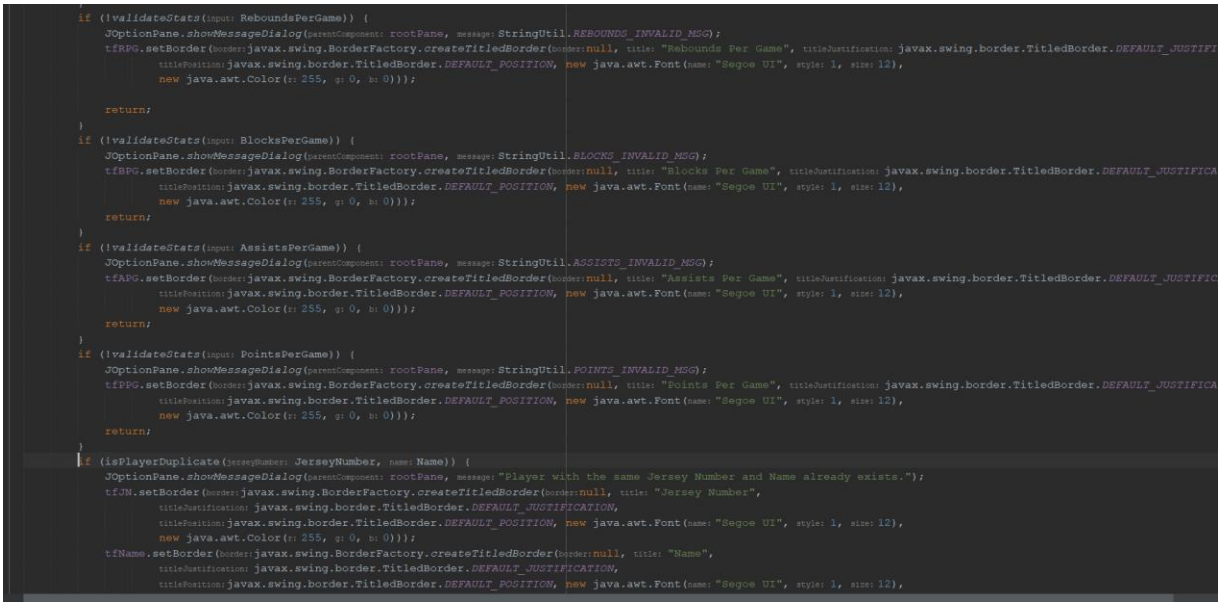
```
private void buAddInTableActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    buResetButtonActionPerformed(evt);

    // fetching data
    String JerseyNumber = tfJN.getText();
    String Name = tfName.getText();
    String GamesPlayed = tfGP.getText();
    String PointsPerGame = tfPPG.getText();
    String ReboundsPerGame = tfRPG.getText();
    String AssistsPerGame = tfAPG.getText();
    String BlocksPerGame = tfBPG.getText();

    //validation
    if (Name.isEmpty()) {
        tfName.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Name", titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
                titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(name: "Segoe UI", style: 1, size: 12),
                new java.awt.Color(r: 255, g: 0, b: 0)));
        JOptionPane.showMessageDialog(parentComponent: rootPane, message: StringUtil.NAME_EMPTY_MSG);
        return;
    }

    if (!Name.matches(regex: "[a-zA-Z,\\s.]+")) {
        JOptionPane.showMessageDialog(parentComponent: rootPane, message: StringUtil.NAME_INVALID_MSG);
        tfName.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Name", titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
                titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(name: "Segoe UI", style: 1, size: 12),
                new java.awt.Color(r: 255, g: 0, b: 0)));

        return;
    }

    if (JerseyNumber.isEmpty()) {
        tfJN.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Jersey Number", titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION
                titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(name: "Segoe UI", style: 1, size: 12),
                new java.awt.Color(r: 255, g: 0, b: 0)));
        JOptionPane.showMessageDialog(parentComponent: rootPane, message: StringUtil.JERSEY_EMPTY_MSG);
        return;
    }
```

Figure 12:buAddInTableActionPerformed 1



```
if (ReboundsPerGame.isEmpty()) {
    tfRPG.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Rebounds Per Game", titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
            titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(name: "Segoe UI", style: 1, size:12),
            new java.awt.Color(r: 255, g: 0, b: 0)));
    JOptionPane.showMessageDialog(parentComponent: rootPane, message: StringUtil.REBOUNDS_EMPTY_MSG);
    return;
}

if (AssistsPerGame.isEmpty()) {
    tfAPG.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Assists Per Game", titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
            titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(name: "Segoe UI", style: 1, size:12),
            new java.awt.Color(r: 255, g: 0, b: 0)));
    JOptionPane.showMessageDialog(parentComponent: rootPane, message: StringUtil.ASSISTS_EMPTY_MSG);
    return;
}

if (BlocksPerGame.isEmpty()) {
    tfBPG.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Blocks Per Game", titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
            titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(name: "Segoe UI", style: 1, size:12),
            new java.awt.Color(r: 255, g: 0, b: 0)));
    JOptionPane.showMessageDialog(parentComponent: rootPane, message: StringUtil.BLOCKS_EMPTY_MSG);
    return;
}
if (!validateStats(input: JerseyNumber)) {
    JOptionPane.showMessageDialog(parentComponent: rootPane, message: StringUtil.JERSEY_INVALID_MSG);
    tfJN.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Jersey Number", titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
            titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(name: "Segoe UI", style: 1, size:12),
            new java.awt.Color(r: 255, g: 0, b: 0)));

    return;
}
if (!validateStats(input: GamesPlayed)) {
    JOptionPane.showMessageDialog(parentComponent: rootPane, message: StringUtil.GAMES_INVALID_MSG);
    tfGP.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Games Played", titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
            titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(name: "Segoe UI", style: 1, size:12),
            new java.awt.Color(r: 255, g: 0, b: 0)));
    return;
}
```

Figure 13:buAddInTableActionPerformed 2



```
        if (!validateStats(input: ReboundsPerGame)) {
            JOptionPane.showMessageDialog(parentComponent: rootPane, message: StringUtil.REBOUNDS_INVALID_MSG);
            tfRPG.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Rebounds Per Game", titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFI
                    titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(name: "Segoe UI", style: 1, size:12),
                    new java.awt.Color(r: 255, g: 0, b: 0)));

            return;
        }
        if (!validateStats(input: BlocksPerGame)) {
            JOptionPane.showMessageDialog(parentComponent: rootPane, message: StringUtil.BLOCKS_INVALID_MSG);
            tfBPG.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Blocks Per Game", titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICA
                    titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(name: "Segoe UI", style: 1, size:12),
                    new java.awt.Color(r: 255, g: 0, b: 0)));
            return;
        }
        if (!validateStats(input: AssistsPerGame)) {
            JOptionPane.showMessageDialog(parentComponent: rootPane, message: StringUtil.ASSISTS_INVALID_MSG);
            tfAPG.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Assists Per Game", titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFIC
                    titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(name: "Segoe UI", style: 1, size:12),
                    new java.awt.Color(r: 255, g: 0, b: 0)));
            return;
        }
        if (!validateStats(input: PointsPerGame)) {
            JOptionPane.showMessageDialog(parentComponent: rootPane, message: StringUtil.POINTS_INVALID_MSG);
            tfPPG.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Points Per Game", titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICA
                    titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(name: "Segoe UI", style: 1, size:12),
                    new java.awt.Color(r: 255, g: 0, b: 0)));
            return;
        }
        if (isPlayerDuplicate(jerseyNumber: JerseyNumber, name:Name)) {
            JOptionPane.showMessageDialog(parentComponent: rootPane, message: "Player with the same Jersey Number and Name already exists.");
            tfJN.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Jersey Number",
                    titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
                    titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(name: "Segoe UI", style: 1, size:12),
                    new java.awt.Color(r: 255, g: 0, b: 0)));
            tfName.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Name",
                    titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
                    titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(name: "Segoe UI", style: 1, size:12),
```

Figure 14: buAddInTableActionPerformed 3

```
            int updateOption = JOptionPane.showConfirmDialog(parentComponent: rootPane, message: StringUtil.UPDATE_OPTION_MSG);
            if (updateOption == JOptionPane.YES_OPTION) {
                tfName.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Name",
                        titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
                        titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION,
                        new java.awt.Font(name: "Segoe UI", style: 1, size: 12), new java.awt.Color(r: 228, g: 253, b: 225)));
                tfJN.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Jersey Number",
                        titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
                        titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION,
                        new java.awt.Font(name: "Segoe UI", style: 1, size: 12), new java.awt.Color(r: 228, g: 253, b: 225)));

                updateJerseyNumber = tfJN.getText();

                for (int row = 0; row < tbMainTable.getRowCount(); row++) {
                    int column = 0;
                    Object rowValue = tbMainTable.getValueAt(row, column);
                    if (rowValue.toString().equals(anObject: updateJerseyNumber)) {
                        pnMainPanel.setVisible(aFlag: false);
                        pnAddPanel.setVisible(aFlag: false);
                        pnUpdatePanel.setVisible(aFlag: true);
                        tfName1.setText(t: (tbMainTable.getValueAt(row, column:1)).toString());
                        tfGP1.setText(t: (tbMainTable.getValueAt(row, column:2)).toString());
                        tfPPG1.setText(t: (tbMainTable.getValueAt(row, column:3)).toString());
                        tfRPG1.setText(t: (tbMainTable.getValueAt(row, column:4)).toString());
                        tfAPG1.setText(t: (tbMainTable.getValueAt(row, column:5)).toString());
                        tfBPG1.setText(t: (tbMainTable.getValueAt(row, column:6)).toString());

                        return;
                    }
                }
            }
```

Figure 15:buAddInTableActionPerformed 4

```
        // Check for duplicate jersey number
        if (isJerseyNumberDuplicate(jerseyNumber: JerseyNumber)) {
            JOptionPane.showMessageDialog(parentComponent: rootPane, message: "Cannot add player. Jersey Number already exists.");
            tfJN.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Jersey Number",
                    titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
                    titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(name: "Segoe UI", style: 1, size: 12),
                    new java.awt.Color(r: 255, g: 0, b: 0)));
            return;
        } else {
            //adding to the table

            DefaultTableModel LeTable
                    = (DefaultTableModel) tbMainTable.getModel();

            LeTable.addRow(new Object[]{JerseyNumber, Name, GamesPlayed, PointsPerGame, ReboundsPerGame, AssistsPerGame, BlocksPerGame});

            // confirmation
            int userResponse = JOptionPane.showConfirmDialog(parentComponent: rootPane, message: "Are you sure?");
            if (userResponse == JOptionPane.YES_OPTION) {
                JOptionPane.showMessageDialog(parentComponent: rootPane, message: "Player added successfully!");
                pnSuccessPanel.setVisible(aFlag: true);
                pnAddPanel.setVisible(aFlag: false);
                setSize(width: 490, height:500);
                arrayList();
            }
        }
    }
```

Figure 16 : buAddInTableActionPerformed 5

4.1.6 buDeleteButtonActionPerformed(evt)

This method deletes the row of the selected Jersey Number from the table.

```
private void buDeleteButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    String JerseyNumber = JOptionPane.showInputDialog(message: "Enter Jersey Number");

    if (JerseyNumber == null) {
        return;
    }

    if (JerseyNumber.isEmpty()) {
        JOptionPane.showMessageDialog(parentComponent: null, message: "Please Enter Jersey Number");
    } //return;
    else {
        DefaultTableModel tableModel = (DefaultTableModel) tbMainTable.getModel();
        for (int row = 0; row < tbMainTable.getRowCount(); row++) {
            int column = 0;
            Object rowValue = tbMainTable.getValueAt(row, column);
            if (rowValue.toString().equals(anObject: JerseyNumber)) {
                tableModel.removeRow(row);
                JOptionPane.showMessageDialog(parentComponent: this, message: "Player  has been sucessfully Removed");
                arrayList();
                return;
            }
        }
        // if jersey number name is incorrect
        JOptionPane.showMessageDialog(parentComponent: this, message: "Jersy Number not found ");

        //return;
    }


}
```

Figure 17: buDeleteButtonActionPerformed

4.1.7 buRadioButtonActionPerformed(evt)

This method sets what the search button searches from i.e jersey number or player name.

```
private void buRadioButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if (buRadioButton.isSelected()) {
        lbSearchLabel.setText(text: "JerseyNo.");
        ModelClass.setSearchField(SearchField: "Search by Jersey Number");
    } else {
        lbSearchLabel.setText(text: "Name");
        ModelClass.setSearchField(SearchField: "Search by Name");
    }

}
```

Figure 18: buRadioButtonActionPerformed

4.1.8 buRadioButtonMouseEntered(evt)

This method changes the label title from Name to Jersey number.

```
private void buRadioButtonMouseEntered(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    if (!buRadioButton.isSelected()) {
        lbSearchLabel.setText(text: "JerseyNo.");
    }
}
```

Figure 19:buRadioButtonMouseEntered

4.1.9 buRadioButtonMouseExited(evt)

This method changes the label title back to Name from Jersey number.

```
private void buRadioButtonMouseExited(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    if (!buRadioButton.isSelected()) {
        lbSearchLabel.setText(text: "Name");
    }
}
```

Figure 20: buRadioButtonMouseExited

4.1.10 buResetButtonActionPerformed(evt)

This method resets the table to the order it was in before sorting.

```
private void buResetButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    ArrayList<ModelClass> unsortedTable = new ArrayList<>();
    unsortedTable.addAll(c: arLeballs);
    DefaultTableModel model = (DefaultTableModel) tbMainTable.getModel();
    model.setRowCount(rowCount: 0);

    for (ModelClass sortedModel : unsortedTable) {
        Object[] rowData = {
            sortedModel.getJerseyNumber(),
            sortedModel.getName(),
            sortedModel.getGamesPlayed(),
            sortedModel.getPointsPerGame(),
            sortedModel.getReboundsPerGame(),
            sortedModel.getAssistsPerGame(),
            sortedModel.getBlocksPerGame(), // Add other columns as needed
        };
        model.addRow(rowData);
    }

    tbMainTable.repaint();
}
```

Figure 21: buResetButtonActionPerformed

4.1.11 buSearchButtonActionPerformed(evt)

This method searches the player from the user input value and displays it on the table.

```
private void buSearchButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if (!buRadioButton.isSelected()) {
        searchName();
    } else {
        searchJerseyNumber();
    }
}
```

Figure 22: buSearchButtonActionPerformed

4.1.12 buSuccessButtonActionPerformed(evt)

This method sets the visibility of main panel to true while setting the visibility of the success panel to false.

```
private void buSuccessButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    // pnAddPanel.setVisible(false);
    pnMainPanel.setVisible(aFlag: true);
    pnSuccessPanel.setVisible(aFlag: false);
    setSize(width: 960, height:490);
}
```

Figure 23: buSuccessButtonActionPerformed

4.1.13 buUpdateButtonActionPerformed(evt)

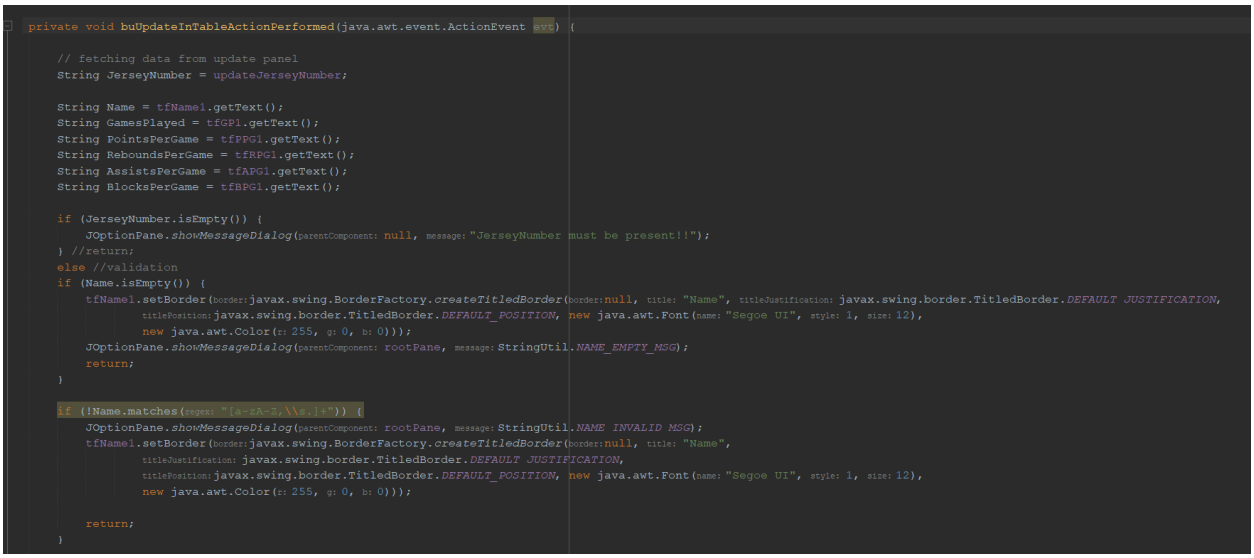
This method sets the visibility of main panel to false while setting the visibility of the update panel to true.

```
private void buUpdateButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    updateJerseyNumber = JOptionPane.showInputDialog(message: "Enter Jersey Number");

    if (this.updateJerseyNumber == null) {
        return;
    }
    if (updateJerseyNumber.isEmpty()) {
        JOptionPane.showMessageDialog(parentComponent: null, message: "JerseyNumber must be present!!");
        return;
    }

    pnMainPanel.setVisible(aFlag: false);
    pnUpdatePanel.setVisible(aFlag: true);
    pnUpdatePanel.setBounds(x: 0, y: 0, width: 950, height:480);
    pnAddPanel.setVisible(aFlag: false);

    for (int row = 0; row < tbMainTable.getRowCount(); row++) {
        int column = 0;
        Object rowValue = tbMainTable.getValueAt(row, column);
        if (rowValue.toString().equals(anObject: updateJerseyNumber)) {
            pnMainPanel.setVisible(aFlag: false);
            pnUpdatePanel.setVisible(aFlag: true);
            tfName1.setText(t: (tbMainTable.getValueAt(row, column:1)).toString());
            tfGP1.setText(t: (tbMainTable.getValueAt(row, column:2)).toString());
            tfPPG1.setText(t: (tbMainTable.getValueAt(row, column:3)).toString());
            tfRPG1.setText(t: (tbMainTable.getValueAt(row, column:4)).toString());
            tfAPG1.setText(t: (tbMainTable.getValueAt(row, column:5)).toString());
            tfBPG1.setText(t: (tbMainTable.getValueAt(row, column:6)).toString());

            return;
        }
    }
}
```

Figure 24:buUpdateButtonActionPerformed

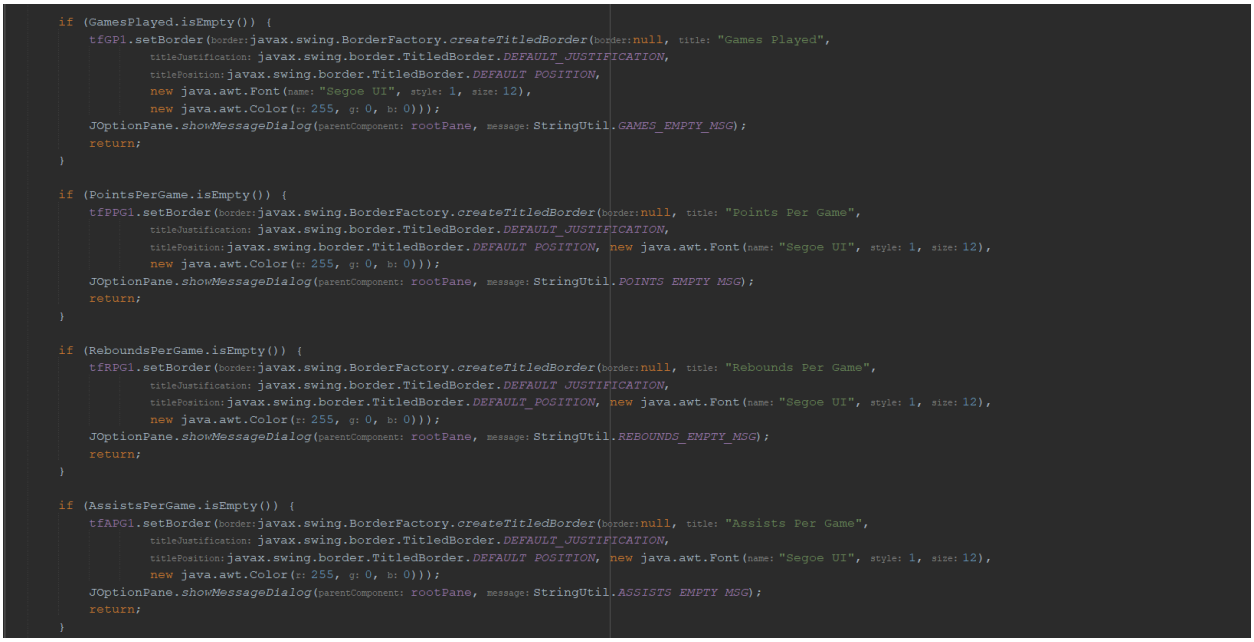
4.1.14 buUpdateInTableActionPerformed(evt)

This method updates the user input value in the row of the selected desired Jersey Number with proper validation.



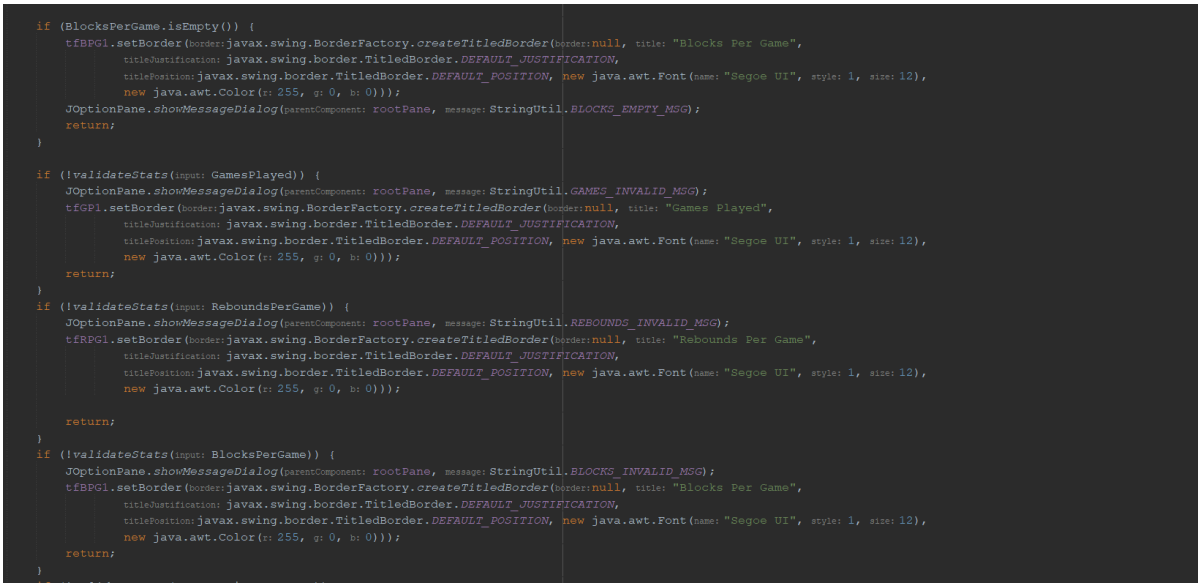
```
private void buUpdateInTableActionPerformed(java.awt.event.ActionEvent evt) {

    // fetching data from update panel
    String JerseyNumber = updateJerseyNumber;

    String Name = tfName1.getText();
    String GamesPlayed = tfGP1.getText();
    String PointsPerGame = tfPPG1.getText();
    String ReboundsPerGame = tfRPG1.getText();
    String AssistsPerGame = tfAPG1.getText();
    String BlocksPerGame = tfBPG1.getText();

    if (JerseyNumber.isEmpty()) {
        JOptionPane.showMessageDialog(parentComponent: null, message: "JerseyNumber must be present!!");
    } //return;
    else //validation
    if (Name.isEmpty()) {
        tfName1.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Name", titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
                titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(name: "Segoe UI", style: 1, size: 12),
                new java.awt.Color(r: 255, g: 0, b: 0)));
        JOptionPane.showMessageDialog(parentComponent: rootPane, message: StringUtil.NAME_EMPTY_MSG);
        return;
    }

    if (!Name.matches(regex: "[a-zA-Z.\\s.]+")) {
        JOptionPane.showMessageDialog(parentComponent: rootPane, message: StringUtil.NAME_INVALID_MSG);
        tfName1.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Name",
                titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
                titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(name: "Segoe UI", style: 1, size: 12),
                new java.awt.Color(r: 255, g: 0, b: 0)));

        return;
    }
```

Figure 25:buUpdateInTableActionPerformed 1



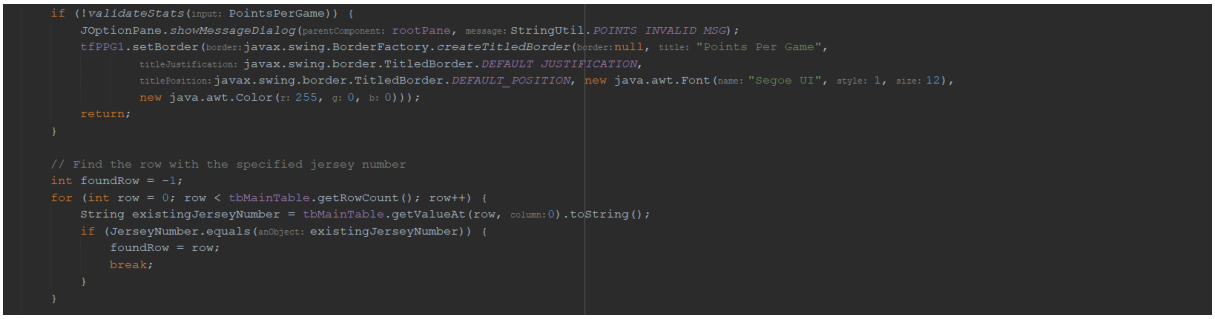
```
    if (GamesPlayed.isEmpty()) {
        tfGP1.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Games Played",
                titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
                titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION,
                new java.awt.Font(name: "Segoe UI", style: 1, size: 12),
                new java.awt.Color(r: 255, g: 0, b: 0)));
        JOptionPane.showMessageDialog(parentComponent: rootPane, message: StringUtil.GAMES_EMPTY_MSG);
        return;
    }

    if (PointsPerGame.isEmpty()) {
        tfPPG1.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Points Per Game",
                titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
                titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(name: "Segoe UI", style: 1, size: 12),
                new java.awt.Color(r: 255, g: 0, b: 0)));
        JOptionPane.showMessageDialog(parentComponent: rootPane, message: StringUtil.POINTS_EMPTY_MSG);
        return;
    }

    if (ReboundsPerGame.isEmpty()) {
        tfRPG1.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Rebounds Per Game",
                titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
                titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(name: "Segoe UI", style: 1, size: 12),
                new java.awt.Color(r: 255, g: 0, b: 0)));
        JOptionPane.showMessageDialog(parentComponent: rootPane, message: StringUtil.REBOUNDS_EMPTY_MSG);
        return;
    }

    if (AssistsPerGame.isEmpty()) {
        tfAPG1.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Assists Per Game",
                titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
                titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(name: "Segoe UI", style: 1, size: 12),
                new java.awt.Color(r: 255, g: 0, b: 0)));
        JOptionPane.showMessageDialog(parentComponent: rootPane, message: StringUtil.ASSISTS_EMPTY_MSG);
        return;
    }
```

Figure 26: buUpdateInTableActionPerformed 2



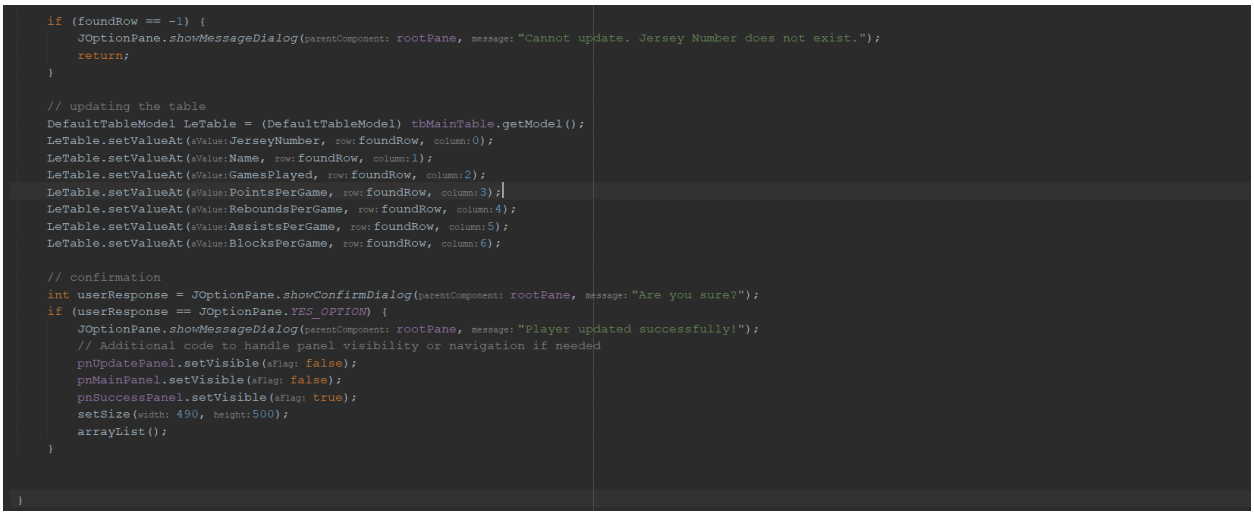
```
if (BlocksPerGame.isEmpty()) {
    tfBPG1.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Blocks Per Game",
            titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
            titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(name: "Segoe UI", style: 1, size: 12),
            new java.awt.Color(r: 255, g: 0, b: 0)));
    JOptionPane.showMessageDialog(parentComponent: rootPane, message: StringUtil.BLOCKS_EMPTY_MSG);
    return;
}

if (!validateStats(input: GamesPlayed)) {
    JOptionPane.showMessageDialog(parentComponent: rootPane, message: StringUtil.GAMES_INVALID_MSG);
    tfGP1.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Games Played",
            titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
            titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(name: "Segoe UI", style: 1, size: 12),
            new java.awt.Color(r: 255, g: 0, b: 0)));
    return;
}
if (!validateStats(input: ReboundsPerGame)) {
    JOptionPane.showMessageDialog(parentComponent: rootPane, message: StringUtil.REBOUNDS_INVALID_MSG);
    tfRPG1.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Rebounds Per Game",
            titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
            titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(name: "Segoe UI", style: 1, size: 12),
            new java.awt.Color(r: 255, g: 0, b: 0)));

    return;
}
if (!validateStats(input: BlocksPerGame)) {
    JOptionPane.showMessageDialog(parentComponent: rootPane, message: StringUtil.BLOCKS_INVALID_MSG);
    tfBPG1.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Blocks Per Game",
            titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
            titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(name: "Segoe UI", style: 1, size: 12),
            new java.awt.Color(r: 255, g: 0, b: 0)));
    return;
}
```

Figure 27: buUpdateInTableActionPerformed 3



```
if (!validateStats(input: PointsPerGame)) {
    JOptionPane.showMessageDialog(parentComponent: rootPane, message: StringUtil.POINTS_INVALID_MSG);
    tfPPG1.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Points Per Game",
            titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
            titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font(name: "Segoe UI", style: 1, size: 12),
            new java.awt.Color(r: 255, g: 0, b: 0)));
    return;
}

// Find the row with the specified jersey number
int foundRow = -1;
for (int row = 0; row < tbMainTable.getRowCount(); row++) {
    String existingJerseyNumber = tbMainTable.getValueAt(row, column:0).toString();
    if (JerseyNumber.equals(anObject: existingJerseyNumber)) {
        foundRow = row;
        break;
    }
}
```

Figure 28:buUpdateInTableActionPerformed 4

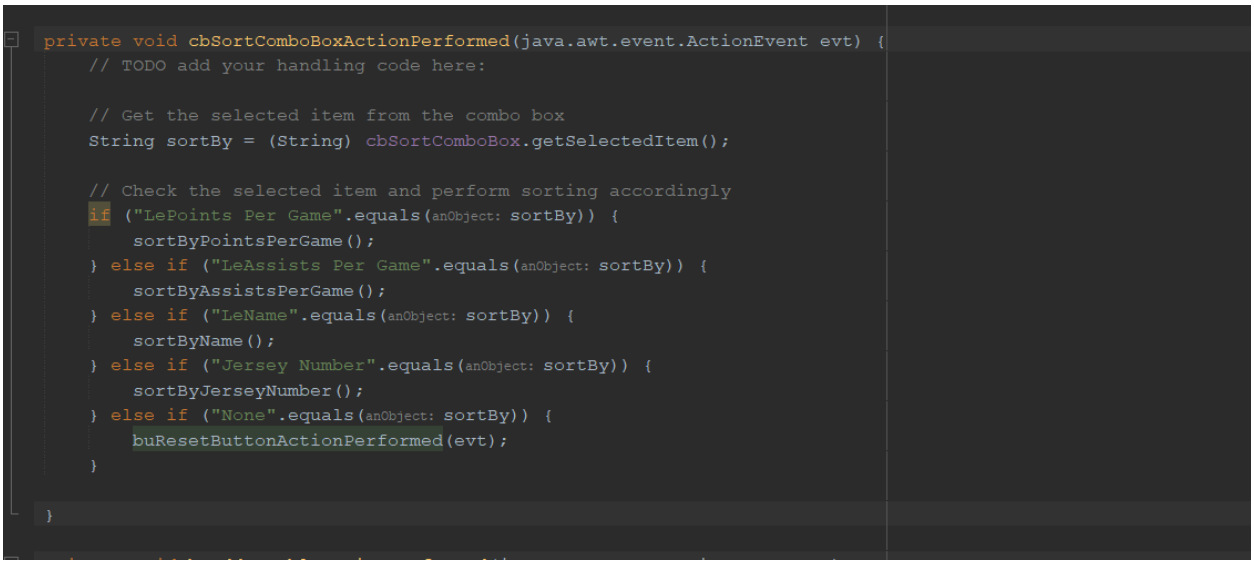


```
if (foundRow == -1) {
    JOptionPane.showMessageDialog(parentComponent: rootPane, message: "Cannot update. Jersey Number does not exist.");
    return;
}

// updating the table
DefaultTableModel LeTable = (DefaultTableModel) tbMainTable.getModel();
LeTable.setValueAt(aValue:JerseyNumber, row:foundRow, column:0);
LeTable.setValueAt(aValue:Name, row:foundRow, column:1);
LeTable.setValueAt(aValue:GamesPlayed, row:foundRow, column:2);
LeTable.setValueAt(aValue:PointsPerGame, row:foundRow, column:3);
LeTable.setValueAt(aValue:ReboundsPerGame, row:foundRow, column:4);
LeTable.setValueAt(aValue:AssistsPerGame, row:foundRow, column:5);
LeTable.setValueAt(aValue:BlocksPerGame, row:foundRow, column:6);

// confirmation
int userResponse = JOptionPane.showConfirmDialog(parentComponent: rootPane, message: "Are you sure?");
if (userResponse == JOptionPane.YES_OPTION) {
    JOptionPane.showMessageDialog(parentComponent: rootPane, message: "Player updated successfully!");
    // Additional code to handle panel visibility or navigation if needed
    pnUpdatePanel.setVisible(aFlag: false);
    pnMainPanel.setVisible(aFlag: false);
    pnSuccessPanel.setVisible(aFlag: true);
    setSize(width: 490, height:500);
    arrayList();
}


}
```

Figure 29:buUpdateInTableActionPerformed 5

4.1.15 cbSortComboBoxActionPerformed(evt)

This method sorts the table according to the option selection in the combo box.



```
private void cbSortComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    // Get the selected item from the combo box
    String sortBy = (String) cbSortComboBox.getSelectedItem();

    // Check the selected item and perform sorting accordingly
    if ("LePoints Per Game".equals(anObject: sortBy)) {
        sortByPointsPerGame();
    } else if ("LeAssists Per Game".equals(anObject: sortBy)) {
        sortByAssistsPerGame();
    } else if ("LeName".equals(anObject: sortBy)) {
        sortByName();
    } else if ("Jersey Number".equals(anObject: sortBy)) {
        sortByJerseyNumber();
    } else if ("None".equals(anObject: sortBy)) {
        buResetButtonActionPerformed(evt);
    }

}
```

Figure 30:cbSortComboBoxActionPerformed

4.1.16 initComponents()

This method initializes and arranges all GUI components according to the design, ensuring proper layout and configuration.

```
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    pnMainPanel = new javax.swing.JPanel();
    pnTitlePanel = new javax.swing.JPanel();
    lbTitle = new javax.swing.JLabel();
    lbLogo = new javax.swing.JLabel();
    lbLogo1 = new javax.swing.JLabel();
    pnScrollPane = new javax.swing.JScrollPane();
    tbMainTable = new javax.swing.JTable();
    lbSortLabel = new javax.swing.JLabel();
    cbSortComboBox = new javax.swing.JComboBox<>();
    buDeleteButton = new javax.swing.JButton();
    buUpdateButton = new javax.swing.JButton();
    buAddButton = new javax.swing.JButton();
    buRadioButton = new javax.swing.JRadioButton();
    lbSearchLabel = new javax.swing.JLabel();
    tfSearchField = new javax.swing.JTextField();
    buSearchButton = new javax.swing.JButton();
    buResetButton = new javax.swing.JButton();
    pnAddPanel = new javax.swing.JPanel();
    tfRPG = new javax.swing.JTextField();
    buAddInTable = new javax.swing.JButton();
    tfGP = new javax.swing.JTextField();
    tfAPG = new javax.swing.JTextField();
    tfName = new javax.swing.JTextField();
    tfBPG = new javax.swing.JTextField();
    tfPPG = new javax.swing.JTextField();
    bnBackButtonAdd = new javax.swing.JButton();
    tfJN = new javax.swing.JTextField();
    lbAddGIF = new javax.swing.JLabel();
    pnUpdatePanel = new javax.swing.JPanel();
    buUpdateInTable = new javax.swing.JButton();
    tfRPG1 = new javax.swing.JTextField();
    tfGP1 = new javax.swing.JTextField();
    tfAPG1 = new javax.swing.JTextField();
    tfName1 = new javax.swing.JTextField();
    tfBPG1 = new javax.swing.JTextField();
```

Figure 31: initComponents 1

```
            .addComponent(component:lbLogo1, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:86, max:javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(type:javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(component:lbTitle, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:284, max:javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(type:javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(component:lbLogo, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:86, max:javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(min:292, pref:292, max:292))
    );
    pnTitlePanelLayout.setVerticalGroup(
        group:pnTitlePanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(group:pnTitlePanelLayout.createSequentialGroup()
            .addGroup(group:pnTitlePanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(component:lbLogo, min:javax.swing.GroupLayout.DEFAULT_SIZE, pref:114, max:Short.MAX_VALUE)
                .addComponent(component:lbLogo1, alignment:javax.swing.GroupLayout.Alignment.TRAILING, min:javax.swing.GroupLayout.DEFAULT_SIZE, pref:javax.swing.GroupLayout.DEFAU
                .addComponent(component:lbTitle, alignment:javax.swing.GroupLayout.Alignment.TRAILING, min:javax.swing.GroupLayout.DEFAULT_SIZE, pref:javax.swing.GroupLayout.DEFAU
            .addContainerGap())
    );

    tbMainTable.setBackground(new java.awt.Color(r:0, g:0, b:0));
    tbMainTable.setBorder(border:javax.swing.BorderFactory.createMatteBorder(top:1, left:1, bottom:1, right:1, new java.awt.Color(r:239, g:211, b:215)));
    tbMainTable.setFont(new java.awt.Font(name:"Dialog", style:1, size:12)); // NOI18N
    tbMainTable.setForeground(new java.awt.Color(r:228, g:253, b:225));
    tbMainTable.setModel(new javax.swing.table.DefaultTableModel(
        new Object [][] {
            {"27", "LeBron James", "22", "25.0", "7.5", "6.6", "0.7"},
            {"22", "A. Davis", "22", "22.7", "12.7", "3.2", "2.7"},
            {"21", "D. Russel", "23", "16.6", "3.3", "6.3", "0.5"},
            {"47", "A. Reaves", "23", "14.1", "4.7", "4.7", "0.1"},
            {"59", "R. Hachimura", "14", "11.4", "3.5", "1.1", "0.2"},
            {"48", "T. Prince", "21", "8.8", "2.5", "1.4", "0.4"},
            {"29", "C. Reddish", "19", "7.1", "2.8", "1.2", "0.3"},
            {"81", "C. Wood", "22", "6.9", "5.9", "0.9", "0.5"},
            {"999", "G. Vincent", "04", "6.0", "1.0", "3.0", "0.0"},
            {"101", "M. Christie", "18", "5.0", "2.7", "1.0", "0.2"},
            {"83", "C. Castleton", "04", "3.0", "1.8", "0.8", "0.0"},
            {"1", "J. Hayes", "31", "2.9", "1.8", "0.3", "0.3"},
            {"63", "J. Vanderbilt", "17", "2.5", "4.1", "0.9", "0.2"},
```

Figure 32: initComponents 2

```
    lbSortLabel.setFont(new java.awt.Font(name:"Dialog", style:1, size:12)); // NOI18N
    lbSortLabel.setForeground(new java.awt.Color(r:228, g:253, b:225));
    lbSortLabel.setText(text:"Sort By");

    cbSortComboBox.setBackground(new java.awt.Color(r:0, g:0, b:0));
    cbSortComboBox.setFont(new java.awt.Font(name:"Dubai", style:1, size:12)); // NOI18N
    cbSortComboBox.setForeground(new java.awt.Color(r:228, g:253, b:225));
    cbSortComboBox.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "None", "LeName", "Jersey Number", "LePoints Per Game", "LeAssists Per Game" }));
    cbSortComboBox.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            cbSortComboBoxActionPerformed(evt);
        }
    });

    buDeleteButton.setBackground(new java.awt.Color(r:0, g:0, b:0));
    buDeleteButton.setForeground(new java.awt.Color(r:228, g:253, b:225));
    buDeleteButton.setText(text:"Delete");
    buDeleteButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            buDeleteButtonActionPerformed(evt);
        }
    });

    buUpdateButton.setBackground(new java.awt.Color(r:0, g:0, b:0));
    buUpdateButton.setForeground(new java.awt.Color(r:228, g:253, b:225));
    buUpdateButton.setText(text:"Update");
    buUpdateButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            buUpdateButtonActionPerformed(evt);
        }
    });
```

Figure 33:initComponents 3

```
buResetButton.setBackground(new java.awt.Color(r: 0, g: 0, b: 0));
buResetButton.setForeground(new java.awt.Color(r: 228, g: 253, b: 225));
buResetButton.setText(text: "Reset");
buResetButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        buResetButtonActionPerformed(evt);
    }
});

javax.swing.GroupLayout pnMainPanelLayout = new javax.swing.GroupLayout(host: pnMainPanel);
pnMainPanel.setLayout(mgr:pnMainPanelLayout);
pnMainPanelLayout.setHorizontalGroup(
    group: pnMainPanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(group: pnMainPanelLayout.createSequentialGroup()
        .addContainerGap()
        .addGroup(group: pnMainPanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(component:pnTitlePanel, alignment:javax.swing.GroupLayout.Alignment.TRAILING, min:javax.swing.GroupLayout.DEFAULT_SIZE, pref: javax.swing.GroupLayout.
            .addGroup(group: pnMainPanelLayout.createSequentialGroup()
                .addComponent(component:lbSortLabel, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref: 43, max:javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(type: javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(component:cbSortComboBox, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref: javax.swing.GroupLayout.DEFAULT_SIZE, max:javax.swing.GroupLayout.PR
                .addGap(min:0, pref: 0, max:Short.MAX_VALUE))
            .addGroup(alignment:javax.swing.GroupLayout.Alignment.TRAILING, group: pnMainPanelLayout.createSequentialGroup()
                .addGap(min:0, pref: 0, max:Short.MAX_VALUE)
                .addGroup(group: pnMainPanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(alignment:javax.swing.GroupLayout.Alignment.TRAILING, group: pnMainPanelLayout.createSequentialGroup()
                        .addComponent(component:lbSearchLabel)
                        .addPreferredGap(type: javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(component:tfSearchField, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref: 159, max:javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(type: javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(component:buSearchButton, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref: 26, max:javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addComponent(component:buRadioButton, alignment:javax.swing.GroupLayout.Alignment.TRAILING, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref: 96, max:javax.
            .addGroup(alignment:javax.swing.GroupLayout.Alignment.TRAILING, group: pnMainPanelLayout.createSequentialGroup()
                .addComponent(component:pnScrollPane)
                .addPreferredGap(type: javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

Figure 34: initComponents 4

```
                    .addComponent(component:buRadioButton, alignment:javax.swing.GroupLayout.Alignment.TRAILING, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref: 96, max:javax.
            .addGroup(alignment:javax.swing.GroupLayout.Alignment.TRAILING, group: pnMainPanelLayout.createSequentialGroup()
                .addComponent(component:pnScrollPane)
                .addPreferredGap(type: javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(group: pnMainPanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.LEADING, resizable:false)
                    .addComponent(component:buDeleteButton, min:javax.swing.GroupLayout.DEFAULT_SIZE, pref: javax.swing.GroupLayout.DEFAULT_SIZE, max:Short.MAX_VALUE)
                    .addComponent(component:buUpdateButton, min:javax.swing.GroupLayout.DEFAULT_SIZE, pref: javax.swing.GroupLayout.DEFAULT_SIZE, max:Short.MAX_VALUE)
                    .addComponent(component:buAddButton, min:javax.swing.GroupLayout.DEFAULT_SIZE, pref: javax.swing.GroupLayout.DEFAULT_SIZE, max:Short.MAX_VALUE)
                    .addComponent(component:buResetButton, min:javax.swing.GroupLayout.DEFAULT_SIZE, pref: javax.swing.GroupLayout.DEFAULT_SIZE, max:Short.MAX_VALUE))
                .addGap(min:8, pref: 8, max:8)))
        .addContainerGap())
);
pnMainPanelLayout.setVerticalGroup(
    group: pnMainPanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(group: pnMainPanelLayout.createSequentialGroup()
        .addComponent(component:pnTitlePanel, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref: javax.swing.GroupLayout.DEFAULT_SIZE, max:javax.swing.GroupLayout.PREFERRED_SI
        .addPreferredGap(type: javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(group: pnMainPanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.TRAILING)
            .addComponent(component:buSearchButton)
            .addGroup(group: pnMainPanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(component:lbSearchLabel)
                .addComponent(component:tfSearchField, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref: javax.swing.GroupLayout.DEFAULT_SIZE, max:javax.swing.GroupLayout.PRE
        .addPreferredGap(type: javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(component:buRadioButton)
        .addPreferredGap(type: javax.swing.LayoutStyle.ComponentPlacement.RELATED, pref: javax.swing.GroupLayout.DEFAULT_SIZE, max:Short.MAX_VALUE)
        .addGroup(group: pnMainPanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(component:lbSortLabel)
            .addComponent(component:cbSortComboBox, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref: 22, max:javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(min:18, pref: 18, max:18)
        .addGroup(group: pnMainPanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(alignment:javax.swing.GroupLayout.Alignment.TRAILING, group: pnMainPanelLayout.createSequentialGroup()
                .addComponent(component:buAddButton, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref: 23, max:javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(type: javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(component:buUpdateButton)
                .addPreferredGap(type: javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(component:buDeleteButton)
                .addPreferredGap(type: javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(component:buResetButton)
```

Figure 35: initComponents 5

```
        .addContainerGap())
);
pnMainPanelLayout.setVerticalGroup(
    group: pnMainPanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(group: pnMainPanelLayout.createSequentialGroup()
        .addComponent(component:pnTitlePanel, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref: javax.swing.GroupLayout.DEFAULT_SIZE, max:javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(type: javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(group: pnMainPanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.TRAILING)
            .addComponent(component:buSearchButton)
            .addGroup(group: pnMainPanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(component:lbSearchLabel)
                .addComponent(component:tfSearchField, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref: javax.swing.GroupLayout.DEFAULT_SIZE, max:javax.swing.GroupLayout.PREFERRED
        .addPreferredGap(type: javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(component:buRadioButton)
        .addPreferredGap(type: javax.swing.LayoutStyle.ComponentPlacement.RELATED, pref: javax.swing.GroupLayout.DEFAULT_SIZE, max:Short.MAX_VALUE)
        .addGroup(group: pnMainPanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(component:lbSortLabel)
            .addComponent(component:cbSortComboBox, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref: 22, max:javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(min:18, pref: 18, max:18)
        .addGroup(group: pnMainPanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(alignment:javax.swing.GroupLayout.Alignment.TRAILING, group: pnMainPanelLayout.createSequentialGroup()
                .addComponent(component:buAddButton, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref: 23, max:javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(type: javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(component:buUpdateButton)
                .addPreferredGap(type: javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(component:buDeleteButton)
                .addPreferredGap(type: javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(component:buResetButton)
                .addGap(min: 77, pref: 77, max: 77))
            .addGroup(alignment:javax.swing.GroupLayout.Alignment.TRAILING, group: pnMainPanelLayout.createSequentialGroup()
                .addComponent(component:pnScrollPane, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref: 244, max:javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(min:14, pref: 14, max:14))))
);
```

Figure 36: initComponents 6

```
tfGP.setBackground(new java.awt.Color(r: 0, g: 0, b: 0));
tfGP.setForeground(new java.awt.Color(r: 228, g: 253, b: 225));
tfGP.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Games Played", titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION, titlePos
tfGP.setPreferredSize(new java.awt.Dimension(width: 150, height:55));
tfGP.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyPressed(java.awt.event.KeyEvent evt) {
        tfGPKeyPressed(evt);
    }
});

tfAPG.setBackground(new java.awt.Color(r: 0, g: 0, b: 0));
tfAPG.setForeground(new java.awt.Color(r: 228, g: 253, b: 225));
tfAPG.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Assists Per Game", titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION, t
tfAPG.setPreferredSize(new java.awt.Dimension(width: 150, height:55));
tfAPG.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyPressed(java.awt.event.KeyEvent evt) {
        tfAPGKeyPressed(evt);
    }
});
```

Figure 37: initComponents 7

```
bnBackButtonAdd.setBackground(new java.awt.Color(r: 0, g: 0, b: 0));
bnBackButtonAdd.setFont(new java.awt.Font(name: "Segoe UI", style: 1, size: 14)); // NOI18N
bnBackButtonAdd.setForeground(new java.awt.Color(r: 228, g: 253, b: 225));
bnBackButtonAdd.setText(text: "Back");
bnBackButtonAdd.setPreferredSize(new java.awt.Dimension(width: 85, height:27));
bnBackButtonAdd.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        bnBackButtonAddActionPerformed(evt);
    }
});

tfJN.setBackground(new java.awt.Color(r: 0, g: 0, b: 0));
tfJN.setForeground(new java.awt.Color(r: 228, g: 253, b: 225));
tfJN.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Jersey Number", titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION, titlePosit
tfJN.setPreferredSize(new java.awt.Dimension(width: 150, height:55));
tfJN.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyPressed(java.awt.event.KeyEvent evt) {
        tfJNKeyPressed(evt);
    }
});

lbAddGIF.setIcon(new javax.swing.ImageIcon(filename: "C:\\Users\\user\\Desktop\\LoRalls\\AddGIF.gif")); // NOI18N
```

Figure 38: initComponents 8

```
lbAddGIF.setIcon(new javax.swing.ImageIcon(filename: "C:\\Users\\user\\Desktop\\LeBalls\\AddGIF.gif")); // NOI18N

javax.swing.GroupLayout pnAddPanelLayout = new javax.swing.GroupLayout(host: pnAddPanel);
pnAddPanel.setLayout(mgr: pnAddPanelLayout);
pnAddPanelLayout.setHorizontalGroup(
    group: pnAddPanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(group: pnAddPanelLayout.createSequentialGroup()
        .addGroup(group: pnAddPanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(group: pnAddPanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(component:bnBackButtonAdd, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:javax.swing.GroupLayout.DEFAULT_SIZE, max:javax.swing.GroupLayout.PREFERRE
                .addPreferredGap(type: javax.swing.LayoutStyle.ComponentPlacement.RELATED, pref: javax.swing.GroupLayout.DEFAULT_SIZE, max:Short.MAX_VALUE)
                .addComponent(component:buAddInTable, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:85, max:javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(min:29, pref:29, max:29))
            .addGroup(group: pnAddPanelLayout.createSequentialGroup()
                .addGap(min:26, pref:26, max:26)
                .addGroup(group: pnAddPanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(group: pnAddPanelLayout.createSequentialGroup()
                        .addGap(min:95, pref:95, max:95)
                        .addComponent(component:tfJN, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:150, max:javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(group: pnAddPanelLayout.createSequentialGroup()
                        .addComponent(component:tfName, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:150, max:javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(min:18, pref:18, max:18)
                        .addComponent(component:tfGP, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:150, max:javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(group: pnAddPanelLayout.createSequentialGroup()
                        .addComponent(component:tfPPG, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:150, max:javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(min:18, pref:18, max:18)
                        .addComponent(component:tfRPG, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:javax.swing.GroupLayout.DEFAULT_SIZE, max:javax.swing.GroupLayout.PREFERRED_
                    .addGroup(group: pnAddPanelLayout.createSequentialGroup()
                        .addComponent(component:tfAPG, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:150, max:javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(min:18, pref:18, max:18)
                        .addComponent(component:tfBPG, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:150, max:javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addPreferredGap(type: javax.swing.LayoutStyle.ComponentPlacement.RELATED, pref: 160, max:Short.MAX_VALUE)))
        .addComponent(component:lbAddGIF, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:430, max:javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap())
);
```

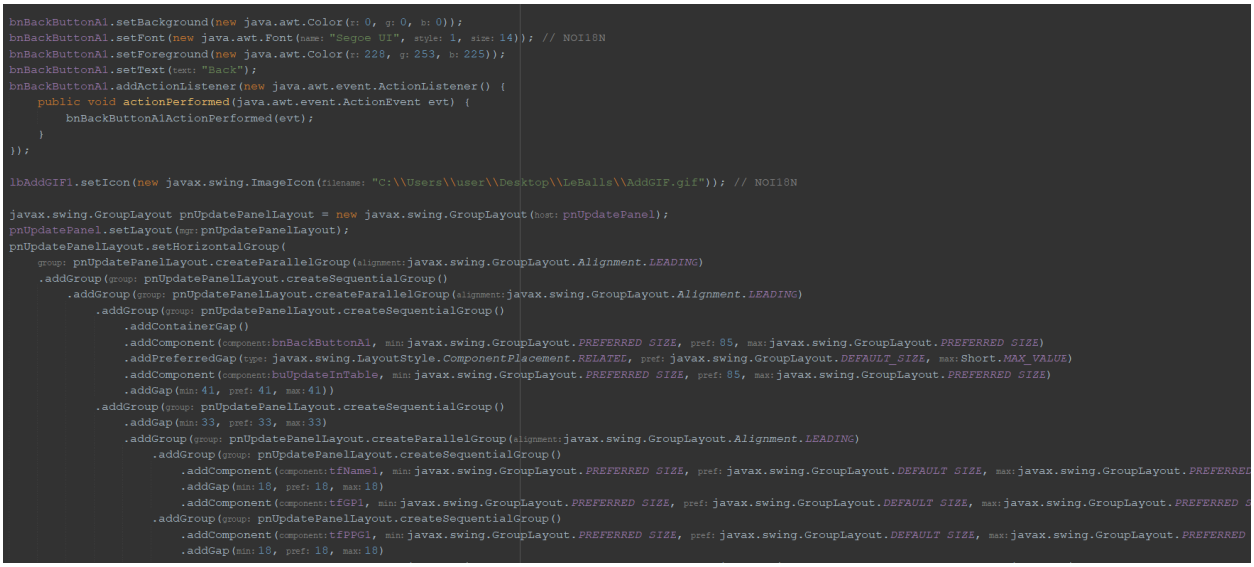
Figure 39 : initComponents 9

```
pnAddPanelLayout.setVerticalGroup(
    group: pnAddPanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(alignment:javax.swing.GroupLayout.Alignment.TRAILING, group: pnAddPanelLayout.createSequentialGroup()
        .addContainerGap(pref: 34, max:Short.MAX_VALUE)
        .addGroup(group: pnAddPanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(component:tfGP, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:javax.swing.GroupLayout.DEFAULT_SIZE, max:javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(component:tfName, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:javax.swing.GroupLayout.DEFAULT_SIZE, max:javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(min:18, pref:18, max:18)
        .addGroup(group: pnAddPanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(component:tfPPG, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:javax.swing.GroupLayout.DEFAULT_SIZE, max:javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(component:tfRPG, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:javax.swing.GroupLayout.DEFAULT_SIZE, max:javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(min:18, pref:18, max:18)
        .addGroup(group: pnAddPanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(component:tfAPG, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:javax.swing.GroupLayout.DEFAULT_SIZE, max:javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(component:tfBPG, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:javax.swing.GroupLayout.DEFAULT_SIZE, max:javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(min:18, pref:18, max:18)
        .addComponent(component:tfJN, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:javax.swing.GroupLayout.DEFAULT_SIZE, max:javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(min:36, pref:36, max:36)
        .addGroup(group: pnAddPanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(component:bnBackButtonAdd, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:javax.swing.GroupLayout.DEFAULT_SIZE, max:javax.swing.GroupLayout.PREFERRED_
            .addComponent(component:buAddInTable, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:27, max:javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(min:55, pref:55, max:55))
    .addGroup(group: pnAddPanelLayout.createSequentialGroup()
        .addContainerGap()
        .addComponent(component:lbAddGIF, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:382, max:javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(pref: javax.swing.GroupLayout.DEFAULT_SIZE, max:Short.MAX_VALUE))
);
```

Figure 40 : initComponents 10

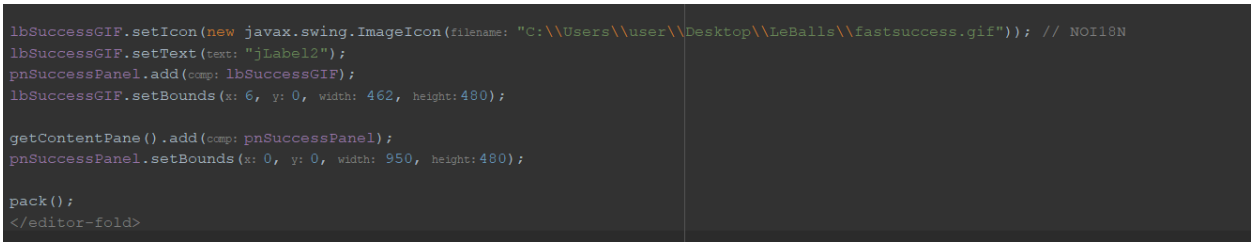
```
tfName1.setBackground(new java.awt.Color(r: 0, g: 0, b: 0));
tfName1.setForeground(new java.awt.Color(r: 228, g: 253, b: 225));
tfName1.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Name", titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION, titlePosition:jav
tfName1.setPreferredSize(new java.awt.Dimension(width: 150, height:55));
tfName1.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyPressed(java.awt.event.KeyEvent evt) {
        tfName1KeyPressed(evt);
    }
});

tfBPG1.setBackground(new java.awt.Color(r: 0, g: 0, b: 0));
tfBPG1.setForeground(new java.awt.Color(r: 228, g: 253, b: 225));
tfBPG1.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Blocks Per Game", titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION, title
tfBPG1.setPreferredSize(new java.awt.Dimension(width: 150, height:55));
tfBPG1.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyPressed(java.awt.event.KeyEvent evt) {
        tfBPG1KeyPressed(evt);
    }
});

tfPPG1.setBackground(new java.awt.Color(r: 0, g: 0, b: 0));
tfPPG1.setForeground(new java.awt.Color(r: 228, g: 253, b: 225));
tfPPG1.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Points Per Game\n", titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION, ti
tfPPG1.setPreferredSize(new java.awt.Dimension(width: 150, height:55));
tfPPG1.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyPressed(java.awt.event.KeyEvent evt) {
        tfPPG1KeyPressed(evt);
    }
});
```

Figure 41 : initComponents 11



```
bnBackButtonA1.setBackground(new java.awt.Color(r: 0, g: 0, b: 0));
bnBackButtonA1.setFont(new java.awt.Font(name: "Segoe UI", style: 1, size: 14)); // NOI18N
bnBackButtonA1.setForeground(new java.awt.Color(r: 228, g: 253, b: 225));
bnBackButtonA1.setText(text: "Back");
bnBackButtonA1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        bnBackButtonA1ActionPerformed(evt);
    }
});

lbAddGIF1.setIcon(new javax.swing.ImageIcon(filename: "C:\\Users\\user\\Desktop\\LeBalls\\AddGIF.gif")); // NOI18N

javax.swing.GroupLayout pnUpdatePanelLayout = new javax.swing.GroupLayout(host: pnUpdatePanel);
pnUpdatePanel.setLayout(mgr:pnUpdatePanelLayout);
pnUpdatePanelLayout.setHorizontalGroup(
    group: pnUpdatePanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(group: pnUpdatePanelLayout.createSequentialGroup()
        .addGroup(group: pnUpdatePanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(group: pnUpdatePanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(component:bnBackButtonA1, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:85, max:javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(type: javax.swing.LayoutStyle.ComponentPlacement.RELATED, pref:javax.swing.GroupLayout.DEFAULT_SIZE, max:Short.MAX_VALUE)
                .addComponent(component:buUpdateInTable, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:85, max:javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(min:41, pref:41, max:41))
            .addGroup(group: pnUpdatePanelLayout.createSequentialGroup()
                .addGap(min:33, pref:33, max:33)
                .addGroup(group: pnUpdatePanelLayout.createParallelGroup(alignment:javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(group: pnUpdatePanelLayout.createSequentialGroup()
                        .addComponent(component:tfName1, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:javax.swing.GroupLayout.DEFAULT_SIZE, max:javax.swing.GroupLayout.PREFERRED
                        .addGap(min:18, pref:18, max:18)
                        .addComponent(component:tfGP1, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:javax.swing.GroupLayout.DEFAULT_SIZE, max:javax.swing.GroupLayout.PREFERRED_S
                    .addGroup(group: pnUpdatePanelLayout.createSequentialGroup()
                        .addComponent(component:tfPPG1, min:javax.swing.GroupLayout.PREFERRED_SIZE, pref:javax.swing.GroupLayout.DEFAULT_SIZE, max:javax.swing.GroupLayout.PREFERRED
                        .addGap(min:18, pref:18, max:18)
```

Figure 42 : initComponents 12



```
lbSuccessGIF.setIcon(new javax.swing.ImageIcon(filename: "C:\\Users\\user\\Desktop\\LeBalls\\fastsuccess.gif")); // NOI18N
lbSuccessGIF.setText(text: "jLabel2");
pnSuccessPanel.add(comp: lbSuccessGIF);
lbSuccessGIF.setBounds(x: 6, y: 0, width: 462, height:480);

getContentPane().add(comp: pnSuccessPanel);
pnSuccessPanel.setBounds(x: 0, y: 0, width: 950, height:480);

pack();
</editor-fold>
```

Figure 43 : initComponents 13

4.1.17 isJerseyNumberDuplicate(String jerseyNumber)

This number check if the JerseyNumber added is duplicate or not.

```
// Check if Jersey Number already exists in the table
private boolean isJerseyNumberDuplicate(String jerseyNumber) {
    DefaultTableModel tableModel = (DefaultTableModel) tbMainTable.getModel();
    int rowCount = tableModel.getRowCount();
    for (int i = 0; i < rowCount; i++) {
        String existingJerseyNumber = (String) tableModel.getValueAt(row: i, column: 0);
        if (jerseyNumber.equals(anObject: existingJerseyNumber)) {
            return true;
        }
    }
    return false;
}
```

Figure 44: isJerseyNumberDuplicate

4.1.18 isPlayerDuplicate(String jerseynumber, String name)

This method checks the Jersey Number and Player Name to check if the added player is duplicate or not.

```
// Check if combination of Jersey Number and Name already exists in the table
private boolean isPlayerDuplicate(String jerseyNumber, String name) {
    DefaultTableModel tableModel = (DefaultTableModel) tbMainTable.getModel();
    int rowCount = tableModel.getRowCount();
    for (int i = 0; i < rowCount; i++) {
        String existingJerseyNumber = ((String) tableModel.getValueAt(row: i, column: 0)).trim().toLowerCase();
        String existingName = ((String) tableModel.getValueAt(row: i, column: 1)).trim().toLowerCase();
        if (jerseyNumber.trim().toLowerCase().equals(anObject: existingJerseyNumber) &&
                name.trim().toLowerCase().equals(anObject: existingName)) {
            return true;
        }
    }
    return false;
}
```

Figure 45: isPlayerDuplicate

4.1.19 main(String[] args)

This method serves as the entry point for the Java application, initiating the program's execution

```
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
     Look and feel setting code (optional)

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(() -> {
        new LeBalls().setVisible(b: true);
    });
}
```

Figure 46: main method

4.1.20 searchJerseyNumber()

This method searches the table using Jersey Number.

```
private void searchJerseyNumber() {

    ModelClass.setSearchField(SearchField: "Search by Jersey Number");
    ArrayList<Integer> positionHolder = new ArrayList<>();
    String binaryText = tfSearchField.getText();

    BinarySearch binaryModel = new BinarySearch();
    MergeSort mergesorter = new MergeSort();
    ArrayList<ModelClass> searchedList = new ArrayList<>();

    ArrayList<ModelClass> sortedList = mergesorter.mergeSort(arr:arLeballs, (String) "Jersey Number");

    if (binaryText.isEmpty()) {
        JOptionPane.showMessageDialog(parentComponent: null, message: StringUtil.SEARCH_JERSEY_EMPTY_MSG);
        return;
    } else {

        binaryModel.searchString(objectHolder: sortedList, low:0, arLeballs.size() - 1,
                searchKey:binaryText, positionHolder, (String) ModelClass.getSearchField());
        if (!validateStats(input: binaryText)) {
            JOptionPane.showMessageDialog(parentComponent: rootPane, message: StringUtil.JERSEY_INVALID_MSG);
            return;
        }
        if (positionHolder.isEmpty()) {
            JOptionPane.showMessageDialog(parentComponent: rootPane, message: "No Player Found");
            return;
        }
        for (int i : positionHolder) {
            searchedList.add(e: sortedList.get(index: i));
        }

        DefaultTableModel model = (DefaultTableModel) tbMainTable.getModel();

        model.setRowCount(rowCount: 0);
```

Figure 47: searchJerseyNumber 1

```
        }
        for (int i : positionHolder) {
            searchedList.add(e: sortedList.get(index: i));
        }

        DefaultTableModel model = (DefaultTableModel) tbMainTable.getModel();

        model.setRowCount(rowCount: 0);

        for (ModelClass sortedModel : searchedList) {
            Object[] rowData = {
                sortedModel.getJerseyNumber(),
                sortedModel.getName(),
                sortedModel.getGamesPlayed(),
                sortedModel.getPointsPerGame(),
                sortedModel.getReboundsPerGame(),
                sortedModel.getAssistsPerGame(),
                sortedModel.getBlocksPerGame(),};
            model.addRow(rowData);
        }

        tbMainTable.repaint();
    }
}
```

Figure 48: searchJerseyNumber 2

4.1.21 searchName()

This method searches the table using Player Name.

```
    private void searchName() {

        ArrayList<Integer> positionHolder = new ArrayList<>();
        String binaryText = tfSearchField.getText();
        BinarySearch binaryModel = new BinarySearch();
        MergeSort mergesorter = new MergeSort();
        ArrayList<ModelClass> searchedList = new ArrayList<>();

        ArrayList<ModelClass> sortedList = mergesorter.mergeSort(arr:arLeballs, (String) "Name");

        if (binaryText.isEmpty()) {
            JOptionPane.showMessageDialog(parentComponent: null, message: StringUtil.SEARCH_NAME_EMPTY_MSG);
            return;
        } else {

            binaryModel.searchString(objectHolder: sortedList, low:0, arLeballs.size() - 1, searchKey:binaryText,
                    positionHolder, (String) ModelClass.getSearchField());
            if (!binaryText.matches(regex: "[a-zA-Z,\\s.]+")) {
                JOptionPane.showMessageDialog(parentComponent: rootPane, message: StringUtil.NAME_INVALID_MSG);
                return;
            }
            if (positionHolder.isEmpty()) {
                JOptionPane.showMessageDialog(parentComponent: rootPane, message: "No Player Found");
                return;
            }
            for (int i : positionHolder) {
                searchedList.add(e: sortedList.get(index: i));
            }

            DefaultTableModel model = (DefaultTableModel) tbMainTable.getModel();

            model.setRowCount(rowCount: 0);
```

Figure 49: searchName 1

```
            for (ModelClass sortedModel : searchedList) {
                Object[] rowData = {
                    sortedModel.getJerseyNumber(),
                    sortedModel.getName(),
                    sortedModel.getGamesPlayed(),
                    sortedModel.getPointsPerGame(),
                    sortedModel.getReboundsPerGame(),
                    sortedModel.getAssistsPerGame(),
                    sortedModel.getBlocksPerGame(),};
                model.addRow(rowData);
            }

            tbMainTable.repaint();

        }
    }
```

Figure 50: searchName 2

4.1.22 sortByAssistsPerGame()

This method sorts the table by Assists Per Game (ascending)

```
private void sortByAssistsPerGame() {

    // Create an instance of MergeSort
    MergeSort mergeSorter = new MergeSort();

    // Perform the sorting based on assists per game
    ArrayList<ModelClass> sortedList = mergeSorter.mergeSort(arr:arLeballs, sortBy:"Assists Per Game");

    // Update the table with sorted data
    updateTableWithSortedData(sortedList);
}
```

Figure 51: sortByAssistsPerGame

4.1.23 sortByJerseyNumber()

This method sorts the table by Jersey Number (ascending)

```
private void sortByJerseyNumber() {

    MergeSort mergeSorter = new MergeSort();
    ArrayList<ModelClass> sortedList = mergeSorter.mergeSort(arr:arLeballs, sortBy:"Jersey Number");

    // Update the table with sorted data
    updateTableWithSortedData(sortedList);

}
```

Figure 52: sortByJerseyNumber

4.1.24 sortByName()

This method sorts the table by Player Name.

```
private void sortByName() {

    MergeSort mergeSorter = new MergeSort();

    ArrayList<ModelClass> sortedList = mergeSorter.mergeSort(arr:arLeballs, sortBy:"Name");

    updateTableWithSortedData(sortedList);

}
```

Figure 53: sortByName

4.1.25 sortByPointsPerGame()

This method sorts the table by Points Per Game (ascending)

```
private void sortByPointsPerGame() {

    // Create an instance of MergeSort
    MergeSort mergeSorter = new MergeSort();

    // Perform the sorting based on points per game
    ArrayList<ModelClass> sortedList = mergeSorter.mergeSort(arr:arLeballs, sortBy:"Points Per Game");

    // Update the table with sorted data
    updateTableWithSortedData(sortedList);
}
```

Figure 54: sortByPointsPerGame

4.1.26 tfAPG1KeyPressed(evt)

This method resets the title color of Assist Per Game in update panel.

```
private void tfAPG1KeyPressed(java.awt.event.KeyEvent evt) {
    // TODO add your handling code here:
    tfAPG1.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Assists Per Game",
            titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
            titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION,
            new java.awt.Font(name: "Segoe UI", style: 1, size: 12), new java.awt.Color(r: 228, g: 253, b: 225)));
}
```

Figure 55: tfAPG1KeyPressed

4.1.27 tfAPGKeyPressed(evt)

This method resets the title color of Assist Per Game in add panel.

```
private void tfAPGKeyPressed(java.awt.event.KeyEvent evt) {
    // TODO add your handling code here:
    tfAPG.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Assists Per Game",
            titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
            titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION,
            new java.awt.Font(name: "Segoe UI", style: 1, size: 12), new java.awt.Color(r: 228, g: 253, b: 225)));
}
```

Figure 56: tfAPGKeyPressed

4.1.28 tfBPG1KeyPressed(evt)

This method resets the title color of Blocks Per Game in update panel.

```
    private void tfBPG1KeyPressed(java.awt.event.KeyEvent evt) {
        // TODO add your handling code here:
        tfBPG1.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Blocks Per Game",
                titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
                titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION,
                new java.awt.Font(name: "Segoe UI", style: 1, size: 12), new java.awt.Color(r: 228, g: 253, b: 225)));

    }
```

Figure 57: tfBPG1KeyPressed

4.1.29 tfBPGKeyPressed(evt)

This method resets the title color of Block Per Game in add panel.

```
private void tfBPGKeyPressed(java.awt.event.KeyEvent evt) {
    // TODO add your handling code here:
    tfBPG.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Blocks Per Game",
            titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
            titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION,
            new java.awt.Font(name: "Segoe UI", style: 1, size: 12), new java.awt.Color(r: 228, g: 253, b: 225)));
}
```

Figure 58:tfBPGKeyPressed

4.1.30 tfGP1KeyPressed(evt)

This method resets the title color of Games Played in update panel.

```
private void tfGP1KeyPressed(java.awt.event.KeyEvent evt) {
    // TODO add your handling code here:
    tfGP1.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Games Played",
            titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
            titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION,
            new java.awt.Font(name: "Segoe UI", style: 1, size: 12), new java.awt.Color(r: 228, g: 253, b: 225)));
}
```

Figure 59: tfGP1KeyPressed

4.1.31 tfGPGKeyPressed(evt)

This method resets the title color of Games Played in add panel.

```
    private void tfGPKeyPressed(java.awt.event.KeyEvent evt) {
        // TODO add your handling code here:
        tfGP.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Games Played",
                titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
                titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION,
                new java.awt.Font(name: "Segoe UI", style: 1, size: 12), new java.awt.Color(r: 228, g: 253, b: 225)));
    }
```

Figure 60:tfGPKeyPressed

4.1.32 tfJNKeyPressed(evt)

This method resets the title color of Jersey Number in add panel.

```
private void tfJNKeyPressed(java.awt.event.KeyEvent evt) {
    // TODO add your handling code here:
    tfJN.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Jersey Number",
            titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
            titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION,
            new java.awt.Font(name: "Segoe UI", style: 1, size: 12), new java.awt.Color(r: 228, g: 253, b: 225)));
}
```

Figure 61: tfJNKeyPressed

4.1.33 tfName1KeyPressed(evt)

This method resets the color of Name in update panel.

```
private void tfName1KeyPressed(java.awt.event.KeyEvent evt) {
    // TODO add your handling code here:
    tfName1.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Name",
            titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
            titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION,
            new java.awt.Font(name: "Segoe UI", style: 1, size: 12), new java.awt.Color(r: 228, g: 253, b: 225)));
}
```

Figure 62: tfName1KeyPressed

4.1.34 tfNameKeyPressed(evt)

This method resets the color of Name in add panel.

```
private void tfNameKeyPressed(java.awt.event.KeyEvent evt) {
    // TODO add your handling code here:
    tfName.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Name",
            titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
            titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION,
            new java.awt.Font(name: "Segoe UI", style: 1, size: 12), new java.awt.Color(r: 228, g: 253, b: 225)));
}
```

Figure 63: tfNameKeyPressed

4.1.35 tfPPG1KeyPressed(evt)

This method resets the color of Points Per Game in update panel.

```
private void tfPPG1KeyPressed(java.awt.event.KeyEvent evt) {
    // TODO add your handling code here:
    tfPPG1.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Points Per Game",
            titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
            titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION,
            new java.awt.Font(name: "Segoe UI", style: 1, size: 12), new java.awt.Color(r: 228, g: 253, b: 225)));
}
```

Figure 64: tfPPG1KeyPressed

4.1.36 tfPPGKeyPressed(evt)

This method resets the color of Points Per Game in add panel.

```
private void tfPPGKeyPressed(java.awt.event.KeyEvent evt) {
    // TODO add your handling code here:
    tfPPG.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Points Per Game",
            titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
            titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION,
            new java.awt.Font(name: "Segoe UI", style: 1, size: 12), new java.awt.Color(r: 228, g: 253, b: 225)));
}
```

Figure 65: tfPPGKeyPressed

4.1.37 tfRPG1KeyPressed(evt)

This method resets the color of Rebounds Per Game in the update panel.

```
private void tfRPG1KeyPressed(java.awt.event.KeyEvent evt) {
    // TODO add your handling code here:
    tfRPG1.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Rebounds Per Game",
            titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
            titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION,
            new java.awt.Font(name: "Segoe UI", style: 1, size: 12), new java.awt.Color(r: 228, g: 253, b: 225)));
}
```

Figure 66: tfRPG1KeyPressed

4.1.38 tfRPGKeyPressed(evt)

This method resets the color of Rebounds Per Game in the add panel.

```
private void tfRPGKeyPressed(java.awt.event.KeyEvent evt) {
    // TODO add your handling code here:
    tfRPG.setBorder(border:javax.swing.BorderFactory.createTitledBorder(border:null, title: "Rebounds Per Game",
            titleJustification: javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
            titlePosition:javax.swing.border.TitledBorder.DEFAULT_POSITION,
            new java.awt.Font(name: "Segoe UI", style: 1, size: 12), new java.awt.Color(r: 228, g: 253, b: 225)));
}
```

Figure 67: tfRPGKeyPressed

4.1.39 updateTableWithSortedData(ArrayList<ModelClass> sortedtList)

This method updates the table with the value in the sortedList

```
private void updateTableWithSortedData(ArrayList<ModelClass> sortedList) {

    DefaultTableModel model = (DefaultTableModel) tbMainTable.getModel();
    model.setRowCount(rowCount: 0);

    for (ModelClass sortedModel : sortedList) {
        Object[] rowData = {
            sortedModel.getJerseyNumber(),
            sortedModel.getName(),
            sortedModel.getGamesPlayed(),
            sortedModel.getPointsPerGame(),
            sortedModel.getReboundsPerGame(),
            sortedModel.getAssistsPerGame(),
            sortedModel.getBlocksPerGame(), // Add other columns as needed
        };
        model.addRow(rowData);
    }

    tbMainTable.repaint();

}
```

Figure 68: updateTableWithSortedData

4.1.40 validateStats(String input)

This method validates the input integer values.

```
 */
private static boolean validateStats(String input) {
    try {

        float floatValue = Float.parseFloat(s: input);

        return floatValue >= 0 && floatValue < 1000;
    } catch (NumberFormatException e) {
        return false;
    }
}
```

Figure 69: validateStats

4.2 Model Class

4.2.1 getAssistsPerGame()

This method gets the value of Assists Per Game.

```
public String getAssistsPerGame() {
    return AssistsPerGame;
}
```

Figure 70: getAssistsPerGame

4.2.2 getBlocksPerGame()

This method gets the value of Blocks Per Game.

```
public String getBlocksPerGame() {
    return BlocksPerGame;
}
```

Figure 71: getBlocksPerGame

4.2.3 getGamesPlayed()

This method gets the value of Games played.

```
public String getGamesPlayed() {
    return GamesPlayed;
}
```

Figure 72: getGamesPlayed

4.2.4 getJerseyNumber()

This method gets the value of Jersey Number.

```
public String getJerseyNumber() {
    return JerseyNumber;
}
```

Figure 73 : getJerseyNumber

4.2.5 getName()
This method gets the value of Player Name.

```
public String getName() {
    return Name;
}
```

Figure 74: getName

4.2.6 getPointsPerGame()
This method gets the value of Points Per Game.

```
public String getPointsPerGame() {
    return PointsPerGame;
}
```

Figure 75 : getPointsPerGame

4.2.7 getReboundsPerGame()
This method gets the value of Rebounds Per Game.

```
public String getReboundsPerGame() {
    return ReboundsPerGame;
}
```

Figure 76 : getReboundsPerGame

4.2.8 getSearchField()
This method gets the value of Search Field.

```
public static String getSearchField() {
    return SearchField;
}
```

Figure 77: getSearchField

4.2.9 setAssistsPerGame(String AssistsPerGame)
This method sets the value of Assists Per Game.

```
public void setAssistsPerGame(String AssistsPerGame) {
    this.AssistsPerGame = AssistsPerGame;
}
```

Figure 78: setAssistsPerGame

4.2.10 setBlocksPerGame(String BlocksPerGame)
This method sets the value of Blocks Per Game.

```
public void setBlocksPerGame(String BlocksPerGame) {
    this.BlocksPerGame = BlocksPerGame;
}
```

Figure 79: setBlocksPerGame

4.2.11 setGamesPlayed (String GamesPlayed)
This method sets the value of Games Played.

```
public void setGamesPlayed(String GamesPlayed) {
    this.GamesPlayed = GamesPlayed;
}
```

Figure 80: setGamesPlayed

4.2.12 setJerseyNumber(String JerseyNumber)
This method sets the value of Jersey Number.

```
public void setJerseyNumber(String JerseyNumber) {
    this.JerseyNumber = JerseyNumber;
}
```

Figure 81: setJerseyNumber

4.2.13 setName(String Name)
This method sets the value of Player Name

```
public void setName(String Name) {
    this.Name = Name;
}
```

Figure 82: setName

4.2.14 setPointsPerGame(String PointsPerGame)
This method sets the value of Points Per Game.

```
public void setPointsPerGame(String PointsPerGame) {
    this.PointsPerGame = PointsPerGame;
}
```

Figure 83: setPointsPerGame

4.2.15 setReboundsPerGame(String ReboundsPerGame)

This method sets the value of Rebounds Per Game.

```
public void setReboundsPerGame(String ReboundsPerGame) {
    this.ReboundsPerGame = ReboundsPerGame;
}
```

Figure 84: setReboundsPerGame

4.2.16 setSearchField(String SearchField)

This method sets the value of Search Field.

```
public static void setSearchField(String SearchField) {
    ModelClass.SearchField = SearchField;
}
```

Figure 85: setSearchField

4.3 Binary Search

4.3.1 SearchString (ArrayList<ModelClass> objectHolder, int low, int high, String searchKey, ArrayList<Integer> positionHolder, String field)

This method applies the binary search algorithm into the array list.

```
15
16  public void searchString(ArrayList<ModelClass> objectHolder, int low, int high,
17          String searchKey, ArrayList<Integer> positionHolder, String field) {
18      String SearchField = ModelClass.getSearchField();
19
20          if (high >= low) {
21              int mid = (low + high) / 2;
22              String midValue = "";
23
24              // Get the value of the specified field
26              switch (SearchField) {
27                  case "Search by Name":
28                      midValue = objectHolder.get(index: mid).getName();
29                      break;
30
31                  case "Search by Jersey Number":
32                      midValue = objectHolder.get(index: mid).getJerseyNumber();
33                      break;
34              }
35
36              // If the element is present at the middle itself
37              if (midValue.equals(anObject: searchKey)) {
38                  // Add the current index to the holder
39                  positionHolder.add(e: mid);
40
41                  // Recursion: Continue searching in both subarrays
42                  searchString(objectHolder, low, mid - 1, searchKey, positionHolder, field);
43                  searchString(objectHolder, mid + 1, high, searchKey, positionHolder, field);
44              } else if (midValue.compareTo(anotherString:searchKey) > 0) {
45                  // If the element is smaller than mid, search in the left subarray
46                  searchString(objectHolder, low, mid - 1, searchKey, positionHolder, field);
47              } else {
48                  // If the element is larger than mid, search in the right subarray
49                  searchString(objectHolder, mid + 1, high, searchKey, positionHolder, field);
50              }
51          }
52          // Base case: Stop recursion when low exceeds high
53      }
```

Figure 86: SearchString

4.4 MergeSort

4.4.1 ArrayList<ModelClass> mergeSort(ArrayList<ModelClass> arr, String sortBy)

This method applies the Merge Sort algorithm to the array list.

```
public ArrayList<ModelClass> mergeSort(ArrayList<ModelClass> arr, String sortBy) {
    if (arr.size() <= 1) {
        return arr;
    }

    int mid = arr.size() / 2;
    ArrayList<ModelClass> left = new ArrayList<>(c: arr.subList(fromIndex:0, toIndex: mid));
    ArrayList<ModelClass> right = new ArrayList<>(c: arr.subList(fromIndex:mid, toIndex: arr.size()));

    return merge(left: mergeSort(arr:left, sortBy), right: mergeSort(arr:right, sortBy), sortBy);
}

private ArrayList<ModelClass> merge(ArrayList<ModelClass> left, ArrayList<ModelClass> right, String sortBy) {
    ArrayList<ModelClass> result = new ArrayList<>();
    int leftIndex = 0, rightIndex = 0;

    while (leftIndex < left.size() && rightIndex < right.size()) {
        ModelClass leftModel = left.get(index: leftIndex);
        ModelClass rightModel = right.get(index: rightIndex);

        int comparison = getComparisonResult(model1:leftModel, model2:rightModel, sortBy);

        if (comparison <= 0) {
            result.add(e: leftModel);
            leftIndex++;
        } else {
            result.add(e: rightModel);
            rightIndex++;
        }
    }

    result.addAll(c: left.subList(fromIndex:leftIndex, toIndex: left.size()));
    result.addAll(c: right.subList(fromIndex:rightIndex, toIndex: right.size()));

    return result;
}
```

Figure 87: Merge Sort

4.4.2 getComparisionResult (ModelClass model1, ModelClass model2, String sortBy

This method compares the instances of model class based on sorting criteria.

```
    }
private int getComparisonResult(ModelClass model1, ModelClass model2, String sortBy) {
    switch (sortBy) {
        case "Points Per Game":
            // Compare based on points per game
            float pointsPerGame1 = Float.parseFloat(s: model1.getPointsPerGame());
            float pointsPerGame2 = Float.parseFloat(s: model2.getPointsPerGame());
            return Float.compare(f1: pointsPerGame1, f2: pointsPerGame2);

        case "Assists Per Game":
            // Compare based on assists per game)
            float assistsPerGame1 = Float.parseFloat(s: model1.getAssistsPerGame());
            float assistsPerGame2 = Float.parseFloat(s: model2.getAssistsPerGame());
            return Float.compare(f1: assistsPerGame1, f2: assistsPerGame2);

        case "Name":
            //compare name
            String name1 = model1.getName();
            String name2 = model2.getName();
            return name1.compareTo(anotherString:name2);
        case "Jersey Number":
            //compare based on Jersey Number
            int jerseyNumber1 = Integer.parseInt(s: model1.getJerseyNumber());
            int jerseyNumber2 = Integer.parseInt(s: model2.getJerseyNumber());
            return Integer.compare(x: jerseyNumber1, y: jerseyNumber2);

        default:
            throw new IllegalArgumentException("Invalid sorting criteria: " + sortBy);
    }
}}
```

Figure 88: getComparisionResult

5. Test Cases

5.1 Test 1

Test Number	1
Objective	To provide evidence for: Add a Player into the table.
Action	The add button has been pressed and the following values are entered into the text fields. Name = Aakarshan Khadka Games Played = 10 Points Per Game = 13 Assists Per Game = 3 Rebounds Per Game = 0 Blocks Per Game = 0.5 Jersey Number = 69
Expected Result	A confirmation box should appear followed by a message that says player has been added to the table. The player should then be inserted in the table.
Actual Result	Confirmation box was shown, and the player was inserted into the table with the values listed above.
Conclusion	Test was successful.

Table 1 : Test 1 Add Player

The Player's values have been input into the text fields and the add button has been pressed.

Figure 89: Test 1: Adding player information

After confirmation, the following message is displayed.

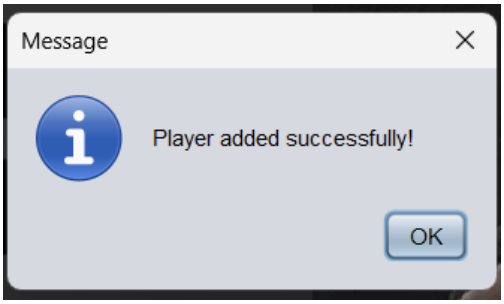


Figure 90: Test 1: Success Message

The table after the player has been added.

Jersey Number	LePlayer	LeGames Played	LePoints Per Game	LeRebounds Per Ga...	LeAssists Per Game	LeBlocks Per Game
1	LeBron James	22	25.0	7.5	6.6	0.7
22	A. Davis	22	22.7	12.7	3.2	2.7
21	D. Russel	23	16.6	3.3	6.3	0.5
4	A. Reaves	23	14.1	4.7	4.7	0.1
5	R. Hachimura	14	11.4	3.5	1.1	0.2
6	T. Prince	21	8.8	2.5	1.4	0.4
7	C. Reddish	19	7.1	2.8	1.2	0.3
8	C. Wood	22	6.9	5.9	0.9	0.5
9	G. Vincent	04	6.0	1.0	3.0	0.0
10	M. Christie	18	5.0	2.7	1.0	0.2
69	Aakarshan Khadka	10	13	0	3	0.5

Figure 91:Test 1: Table After Adding

5.2 Test 2

Table 2: Test 2:Update Player

Test Number	2
Objective	To provide evidence for: Updating a player from the table.
Action	The update button has been pressed and the jersey number ‘69’ for the desired player has been entered in the dialog box. The following values are collected from the table. Name = Aakarshan Khadka Games Played = 10 Points Per Game = 13 Assists Per Game = 3 Rebounds Per Game = 0 Blocks Per Game = 0.5 The value for games played and points per game have been updates to: Games Played = 12 Points Per Game = 15
Expected Result	A confirmation box should appear followed by a message that says player has been updated. The player’s value should then be updated as entered.
Actual Result	Confirmation box was shown, and the player’s value was updated as per entered.
Conclusion	Test was successful.

The Update Button and the desired Jersey Number has been entered the box.

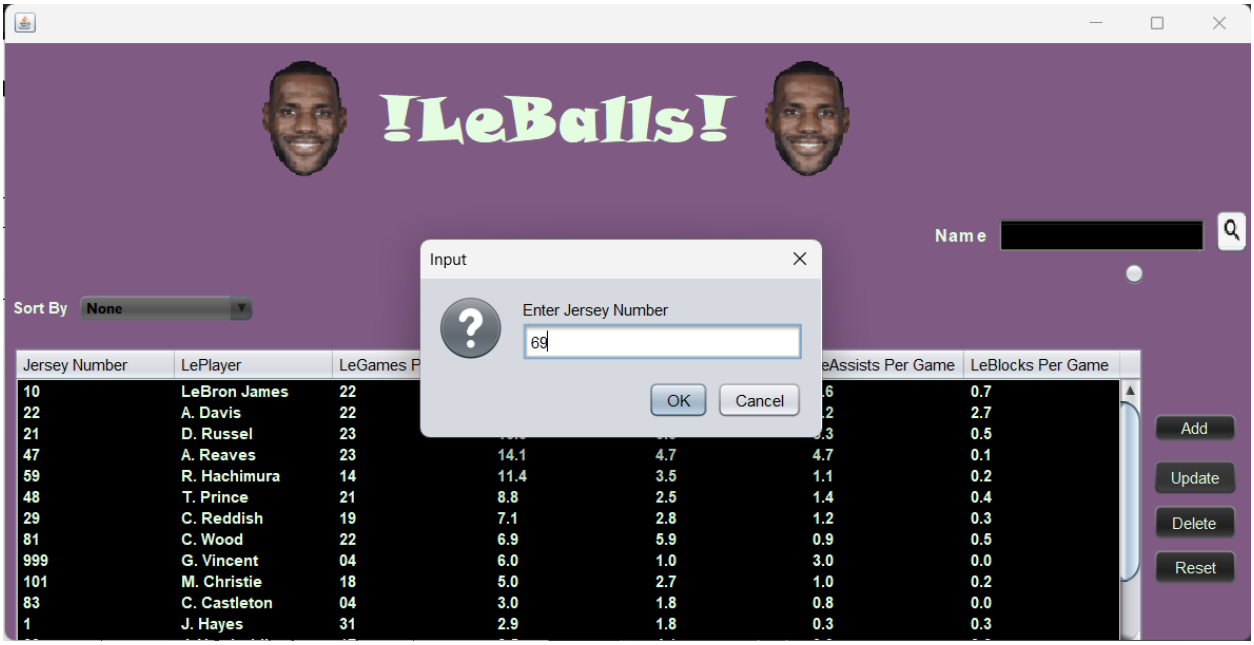


Figure 92: Test 2: Entering desired Jersey Number

The new values has been entered in the text fields.

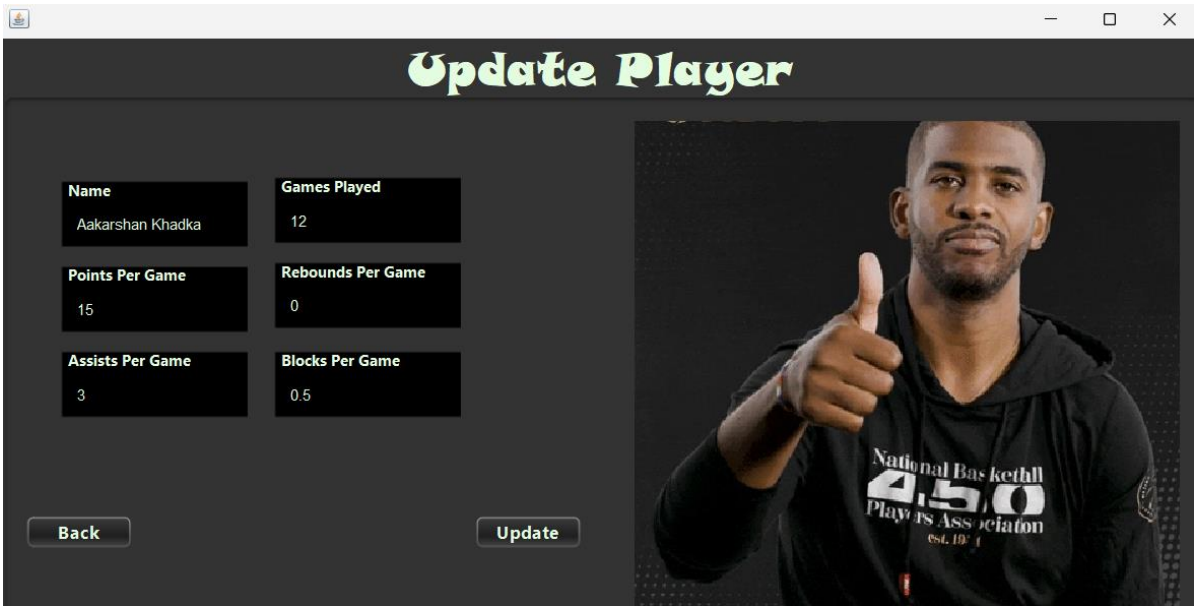
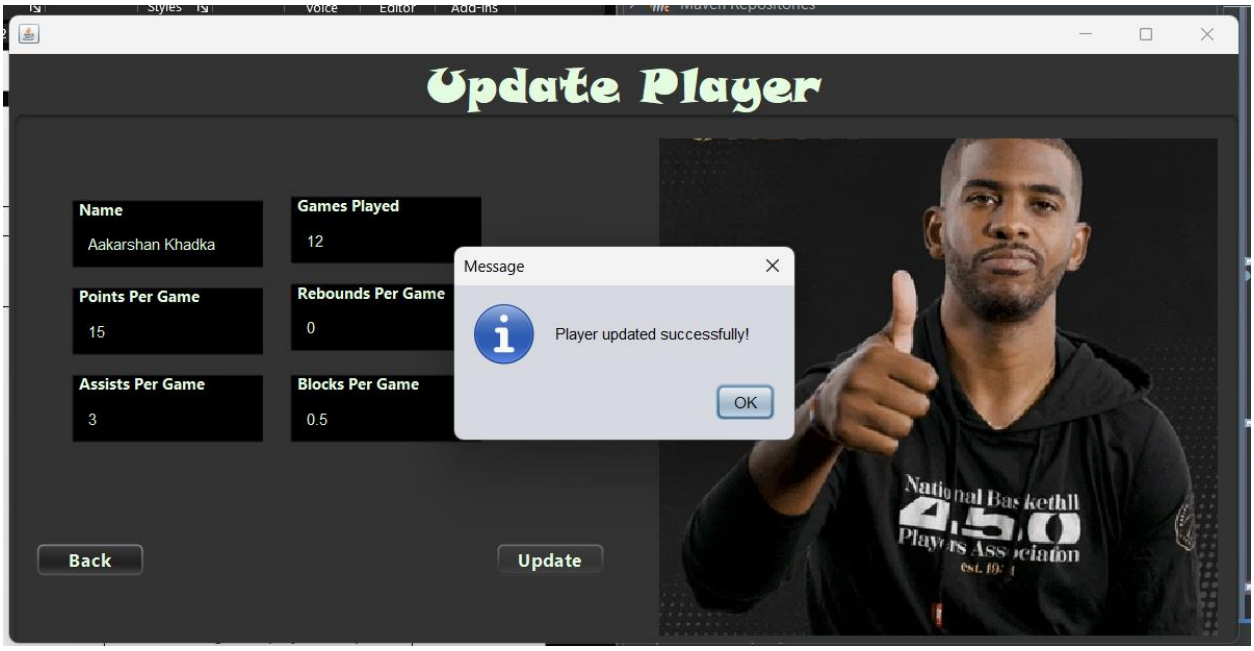


Figure 93:Test 2: Updating Player Information

The update button has been pressed and after confirmation the message has been displayed.



The Value in table after updating

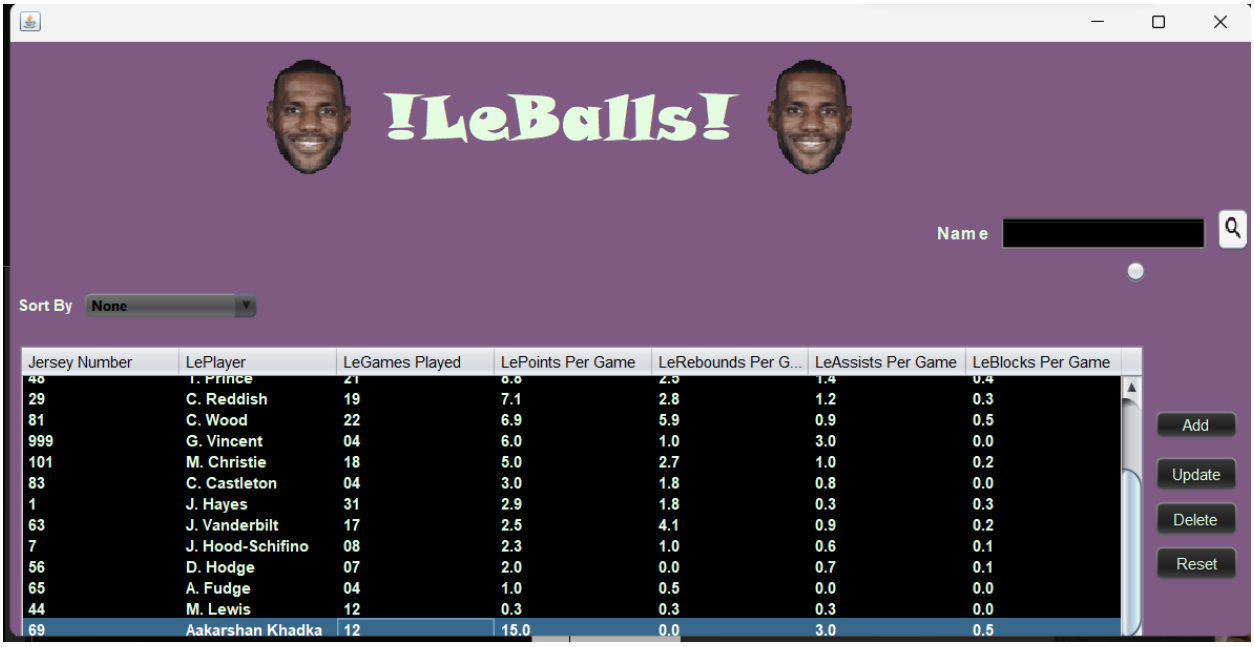


Figure 94:Test 3: Value in Table After Updating

5.3 Test 3

Table 3: Test 3:Delete Player

Test Number	3
Objective	To provide evidence for: Deleting a player from the table.
Action	The Delete button has been pressed, and the desired jersey number ‘69’ has been entered in the dialog box.
Expected Result	A message should be displayed and the row with the entered jersey number should be removed from the table.
Actual Result	Message was displayed and the jersey number ‘69’ row has been removed from the table
Conclusion	Test was successful.

The Delete button has been pressed and the desired jersey number has been entered in the box.

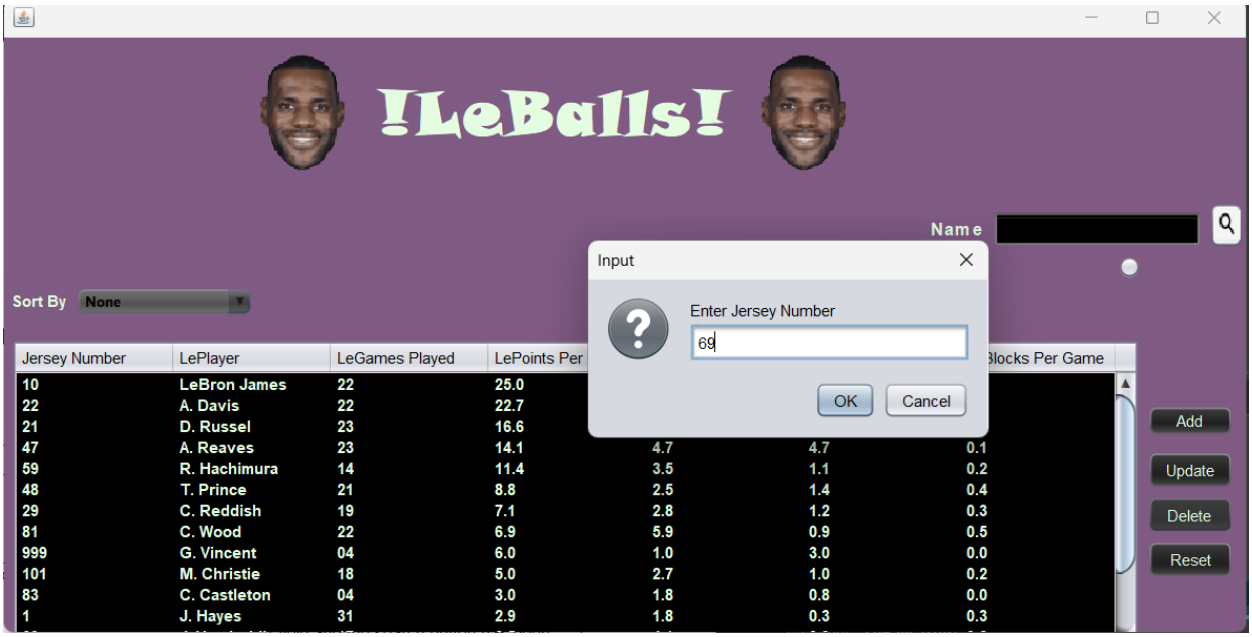


Figure 95: Test 3:Entering Desired Jersey Number

The message after hitting the ok button. The player has been removed from the table.

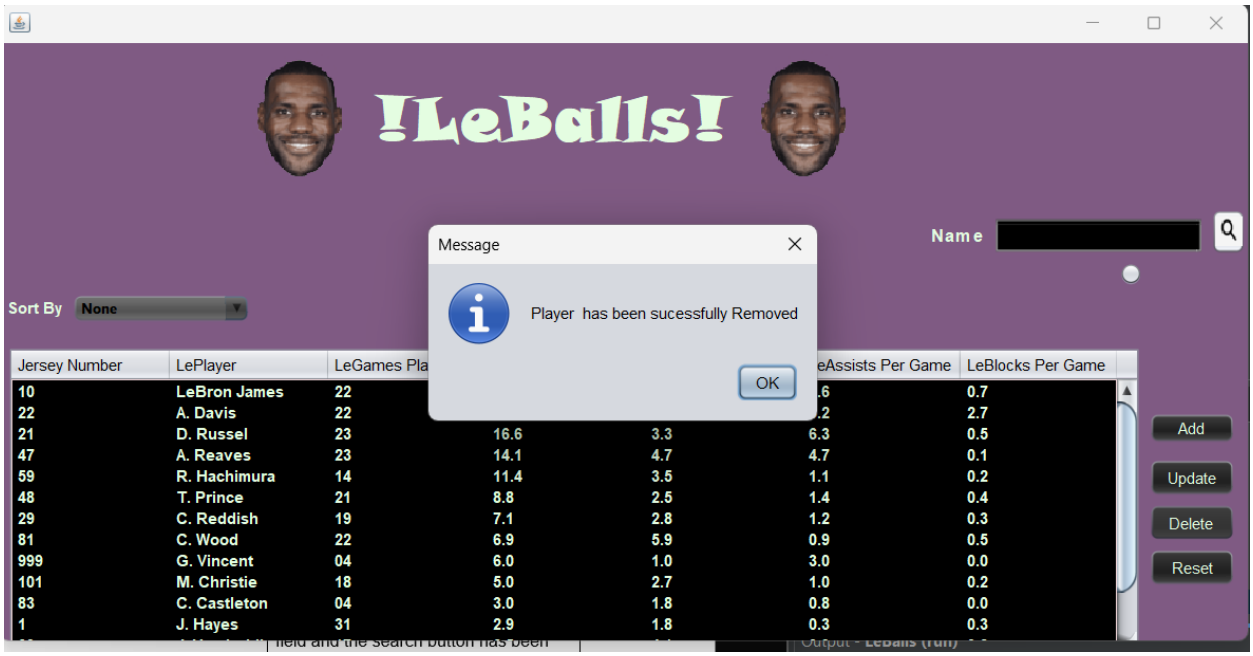


Figure 96:Test 3: Deletion Message

5.4 Test 4

Table 4: Test 4: Merge Sort

Test Number	4
Objective	To provide evidence for: Merge Sort Algorithm
Action	The combo box has been set from ‘None’ to ‘Jersey Number’
Expected Result	The table should be rearranged in ascending order taking Jersey Number as the key.
Actual Result	The table was rearranged in ascending order with Jersey Number as the key.
Conclusion	Test was successful.

The table before sorting.

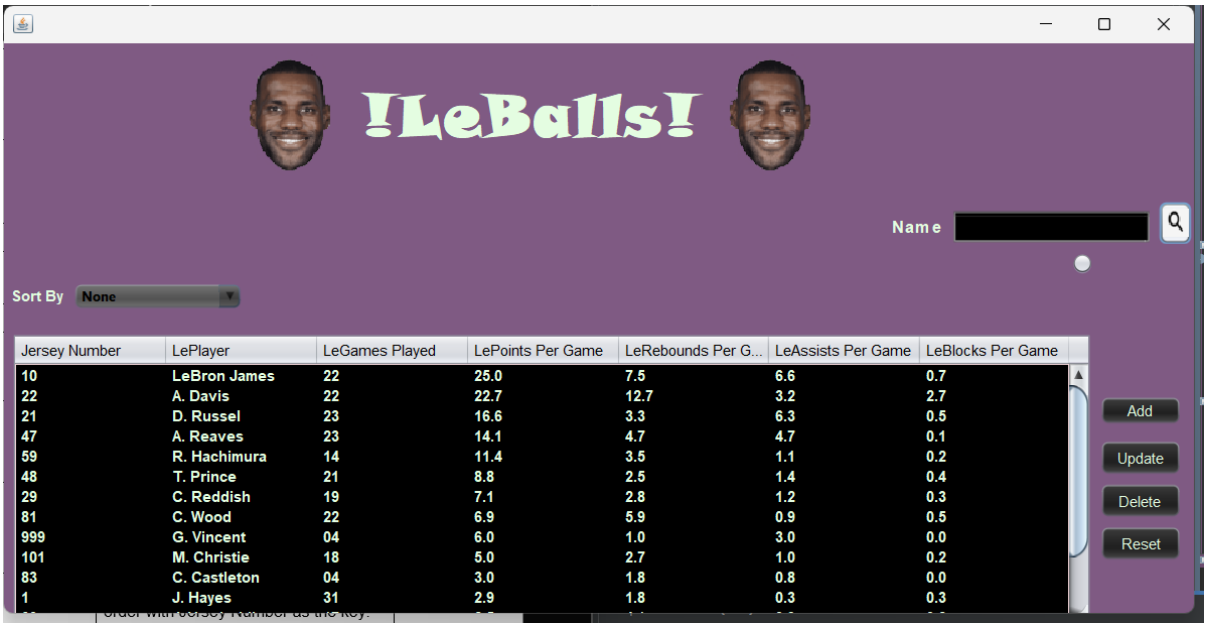


Figure 97:Test 4:Table before Sorting

The Table After sorting with Jersey Number.

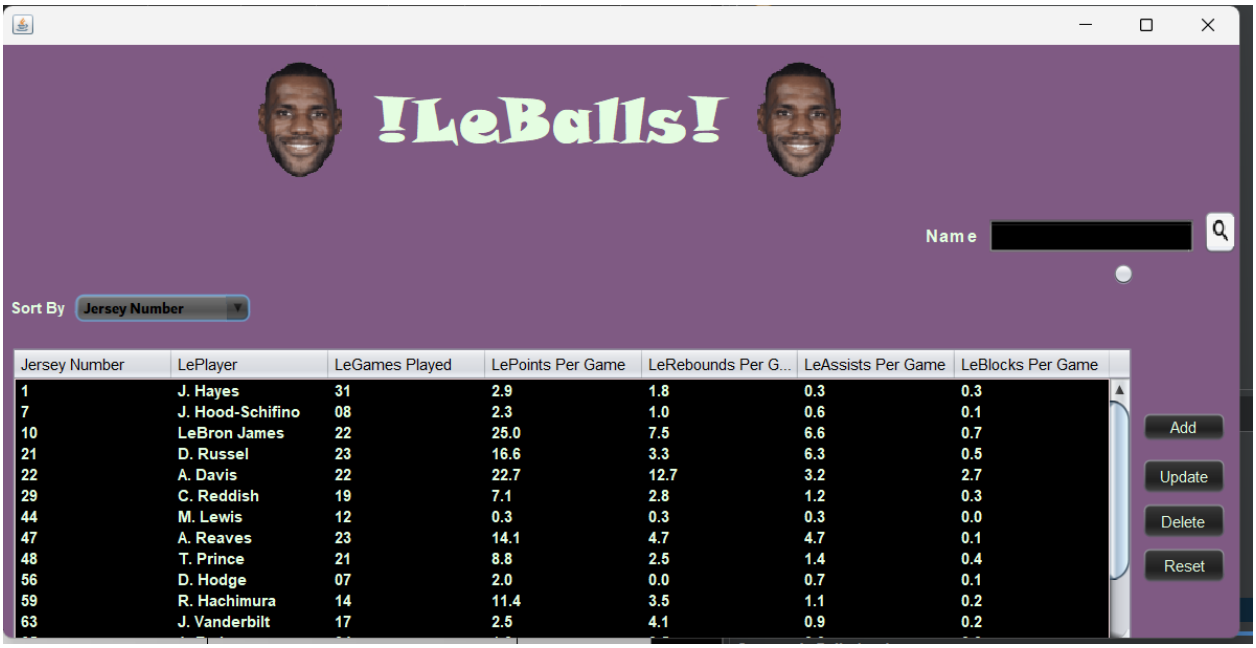


Figure 98:Test 4:Table After Sorting

5.5 Test 5

Table 5: Test 5:Serch using Name

Test Number	5
Objective	To provide evidence for: Binary Search Algorithm using player name
Action	The radio button has been turned off and ‘LeBron James’ has been entered in the search field and the search button has been pressed.
Expected Result	Only the row with Player Name, ‘LeBron James’ should appear in the table.
Actual Result	Only the row with Player Name ‘Lebron James’ appeared in the table.
Conclusion	Test was successful.

‘LeBron James’ has been entered in the text field and the search button has been pressed.

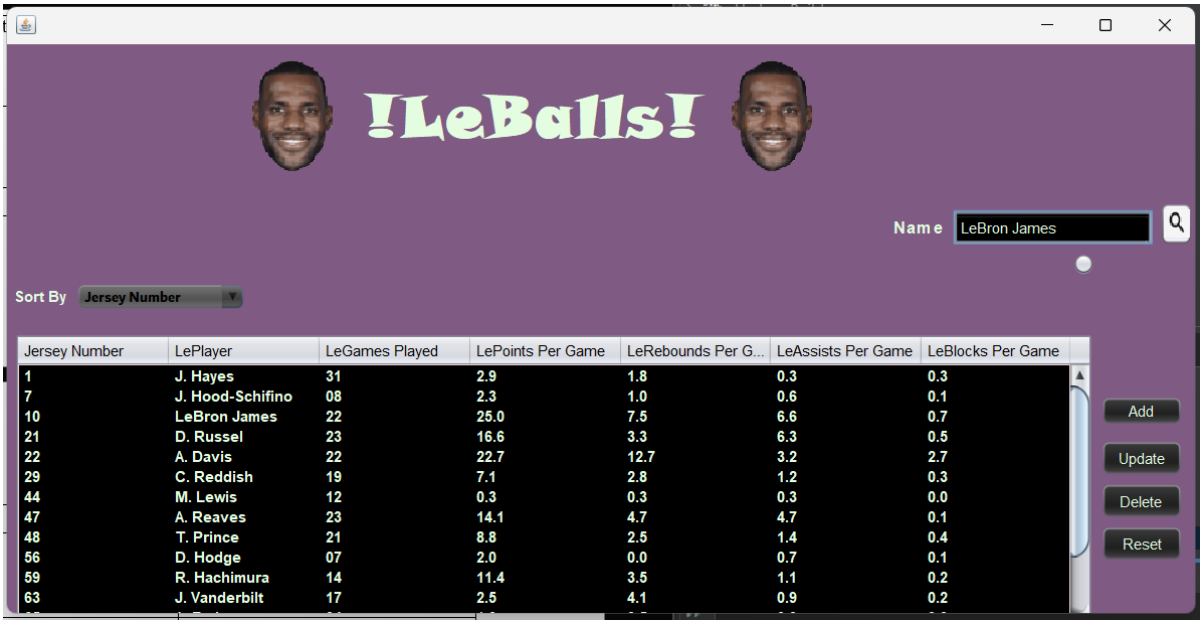


Figure 99:Test 5: Search field

The table after the search button has been pressed.

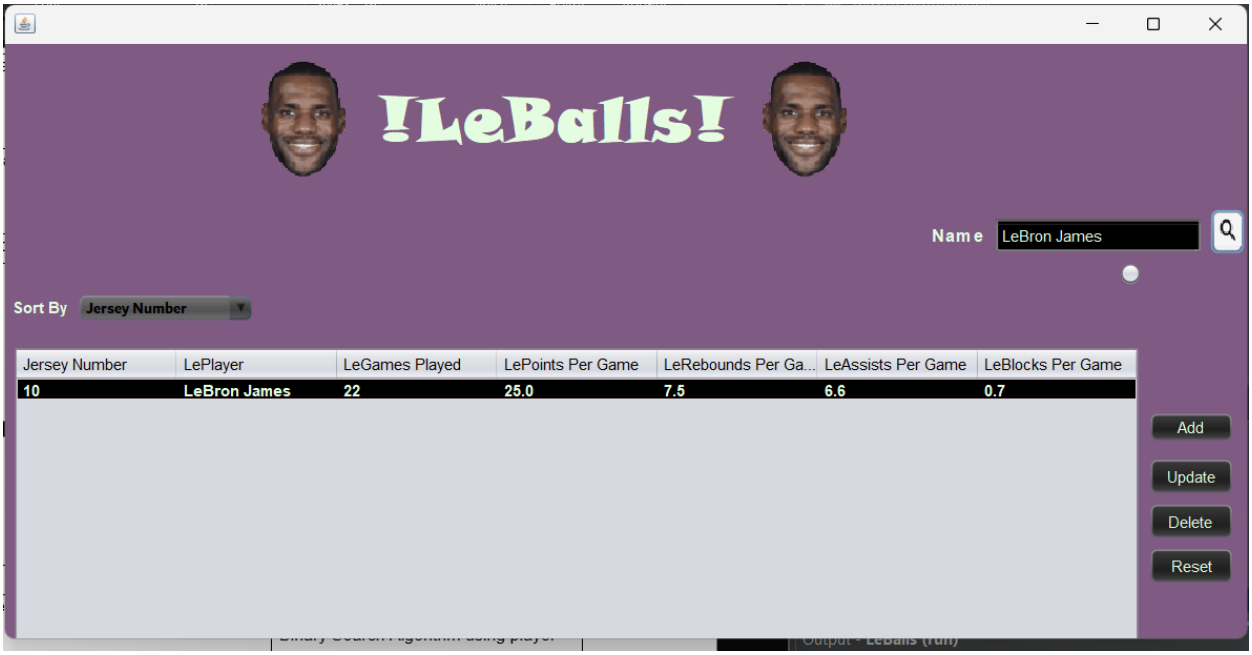


Figure 100:Test 5:Table after searching

5.5 Test 6

Table 6: Test 6:Serach Using Jersey Number

Test Number	6
Objective	To provide evidence for: Binary Search Algorithm using jersey number
Action	The radio button has been turned on and ‘69’ has been entered in the search field and the search button has been pressed.
Expected Result	Only the row with jersey number ‘69’ should appear in the table.
Actual Result	Only the row with jersey number ‘69’ appeared in the table.
Conclusion	Test was successful.

‘69’ has been entered in the search field and the search button has been pressed.

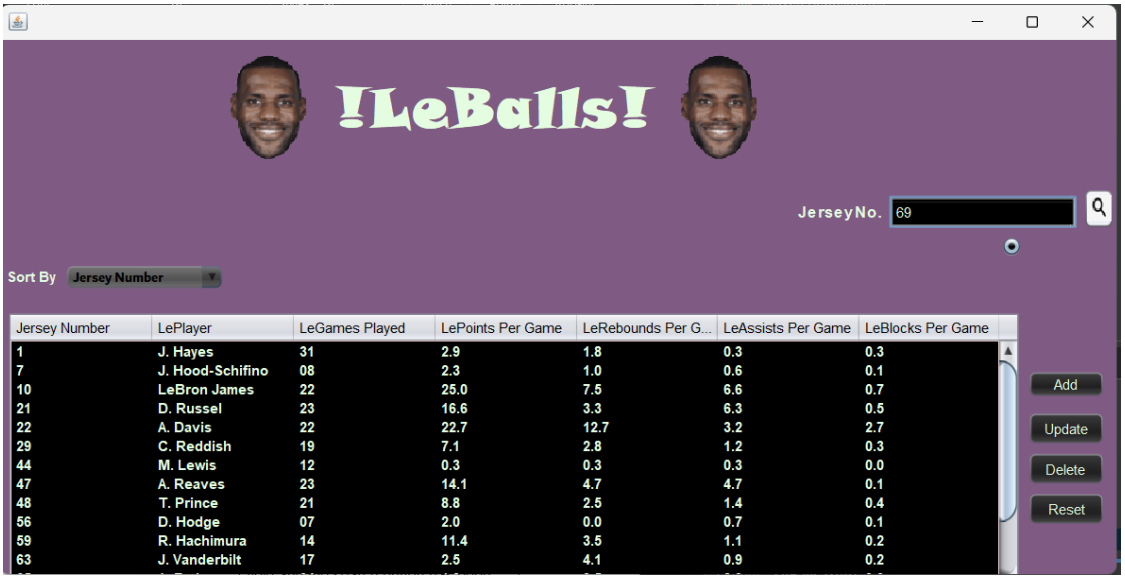


Figure 101:Test 6:Table Before Searching

The table after the search button has been pressed. Only the jersey number 69 is shown.

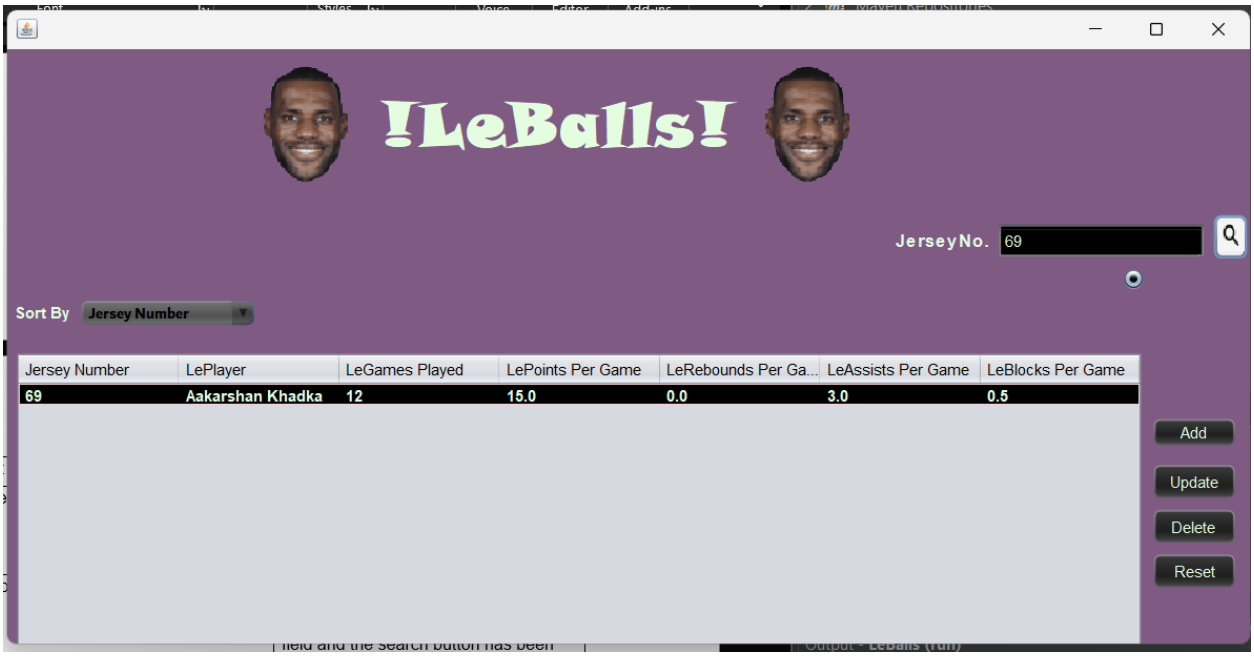


Figure 102:Test 6:Table after searching

6. Challenges and problems faced

During this coursework, I encountered a lot of problems.

During the implementation of the binary search, I encountered difficulties in ensuring correct array indexing. Mistakes in indexing led to issues in the comparison process and accessing array elements. Achieving consistency in data types between the array elements and the search key was also challenging, as it required careful validation to prevent data type mismatches. Additionally, the requirement of a sorted array for binary search necessitated implementing a sorting mechanism, which added complexity to the overall process.

In the case of merge sort, understanding the divide-and-conquer strategy and recursion posed a notable challenge. Correctly splitting the array during recursive calls and accurately implementing the merging logic required a deep understanding of the algorithm's intricacies. The complexity of the merge sort algorithm itself presented hurdles, especially in the early stages of implementation.

User interface design proved to be a challenging aspect of the project. Creating a seamless and intuitive interface that provides a positive user experience (UX) demanded meticulous attention to detail. Achieving consistency in design elements, layout, and color schemes required extra effort. Furthermore, adapting the user interface to different screen sizes and resolutions for a responsive design added an additional layer of complexity.

Implementing an ArrayList came with its own set of challenges. Managing dynamic resizing mechanisms and ensuring memory efficiency during array operations proved to be intricate tasks. The correct addition and removal of elements while maintaining the integrity of the list demanded careful consideration. Additionally, addressing issues related to indexing and element access within the ArrayList presented ongoing challenges.

In the broader context of general implementation, robust error handling became a key consideration. The task of creating comprehensive test cases to ensure the correctness and reliability of implemented functionalities required constant attention. Documenting the code proved to be crucial, especially in addressing challenges related to array initialization, adding table values into the array, and updating the array without introducing duplicate values. Regular testing and iterative refinement were essential to overcome these challenges and achieve a successful implementation.

7. Conclusion

In conclusion, the development of LeBalls has provided me with a hands-on learning experience, offering insights into database management and user interface design. This project has equipped me with valuable skills that extend beyond the classroom, and I look forward to applying these capabilities in future endeavors.

I appreciate the opportunity to work on LeBalls and the chance to explore the intricacies of tailoring a database to specific needs. This practical experience has deepened my understanding of the subject matter, reinforcing theoretical knowledge with real-world application.

I acknowledge Mr. Prithivi Maharjan for overseeing this project, providing guidance when needed. His insights have been beneficial, contributing to the overall learning process. As I move forward, I'm excited about the prospect of using the skills acquired from LeBalls in upcoming projects and challenges. The journey doesn't end here, and I'm eager to apply this newfound knowledge in diverse settings.