Criteria C

Details

Created using:

- Mac OS X El Capitan 10.11.1
- NetBeans IDE 8.0.2
- MySql
- JDBC 4.2 to connect MySql and Java

Techniques Used

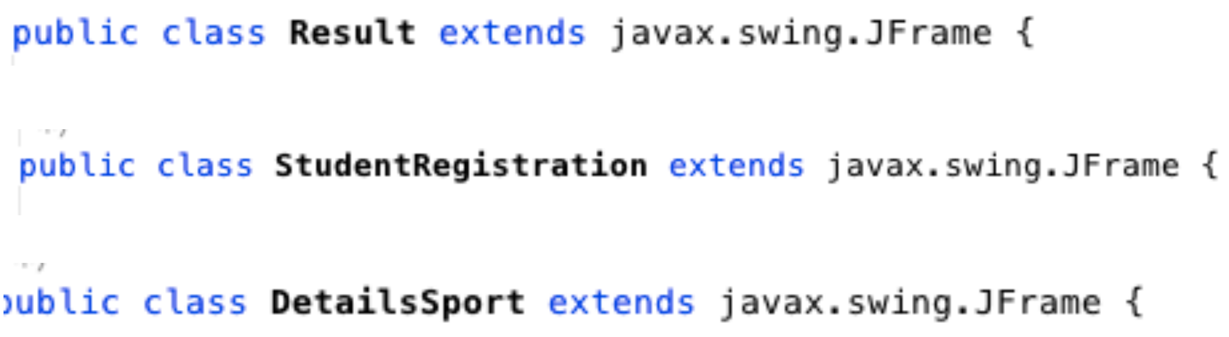
1. **GUI Features**
2. **Inheritance**
3. **Validation & Verification**
4. **Encapsulation**
5. **Adding data into the database**
6. **Polymorphism**
7. **Updating/editing data in the database**
8. **Deleting data in the database**
9. **Constructor**
10. **Array List, Iterator, Array**
11. **Searching data in the database**
12. **Error Checking**
    - I. **Popup Menus**
    - II. **Combo Boxes**
13. **Sorting**

1. GUI Features:

This has been done using JFrames and JPanels.

1.1 Technique-Inheritance

Inheritance is defined as the process where one class acquires the properties (methods and fields) of another.



```
public class Result extends javax.swing.JFrame {

public class StudentRegistration extends javax.swing.JFrame {

ublic class DetailsSport extends javax.swing.JFrame {
```

FIGURE 1 CLASS NAMES

All public classes with a GUI inherit from the javax,swing.JFrame class. Objects of JPanels were used for all the panels. Inheritance was used:

- To minimise the amount of duplicate code
- To easily override the functions of the JFrame class when required

2. Validation and Authentication

To ensure security a login form has been used. It consists of User ID and Password, which verifies data with the Login table.

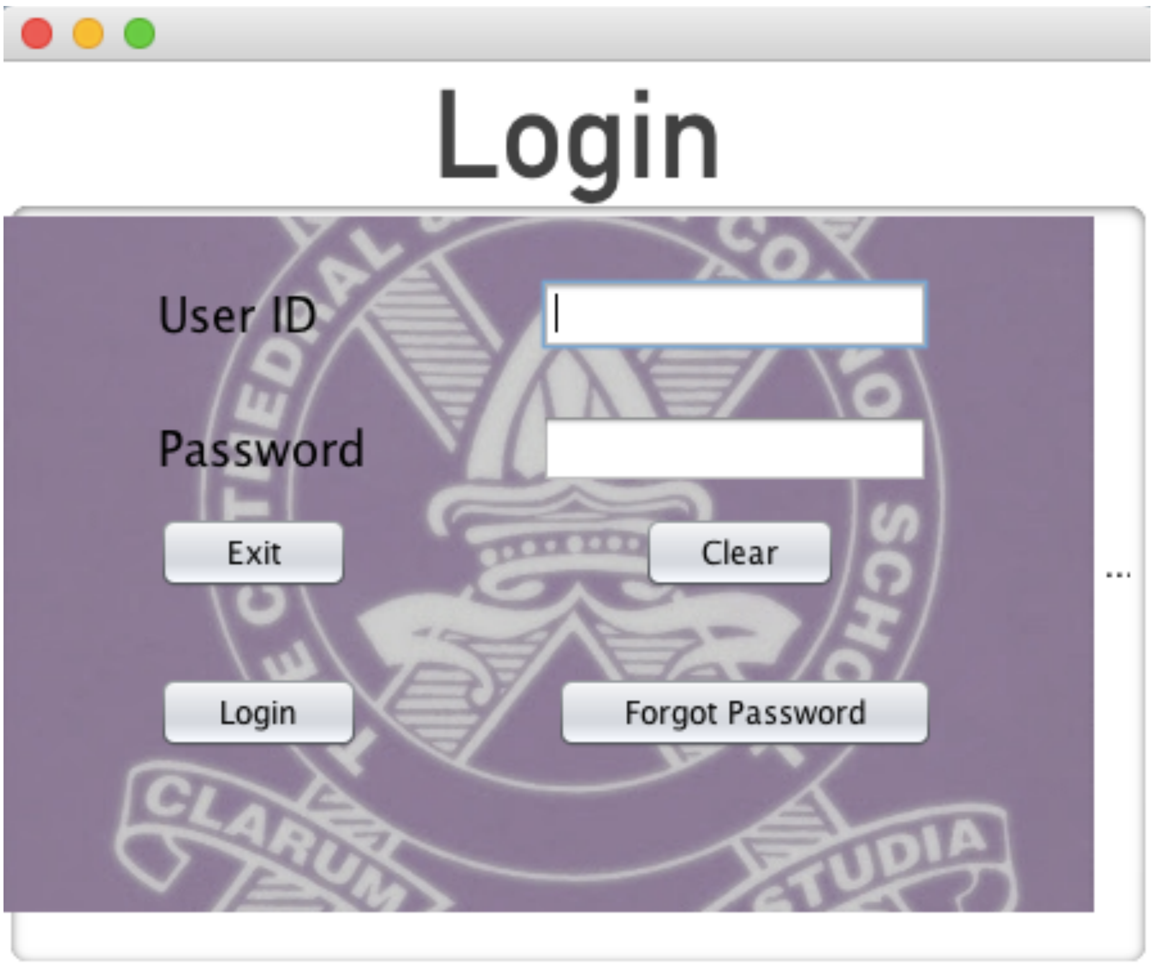
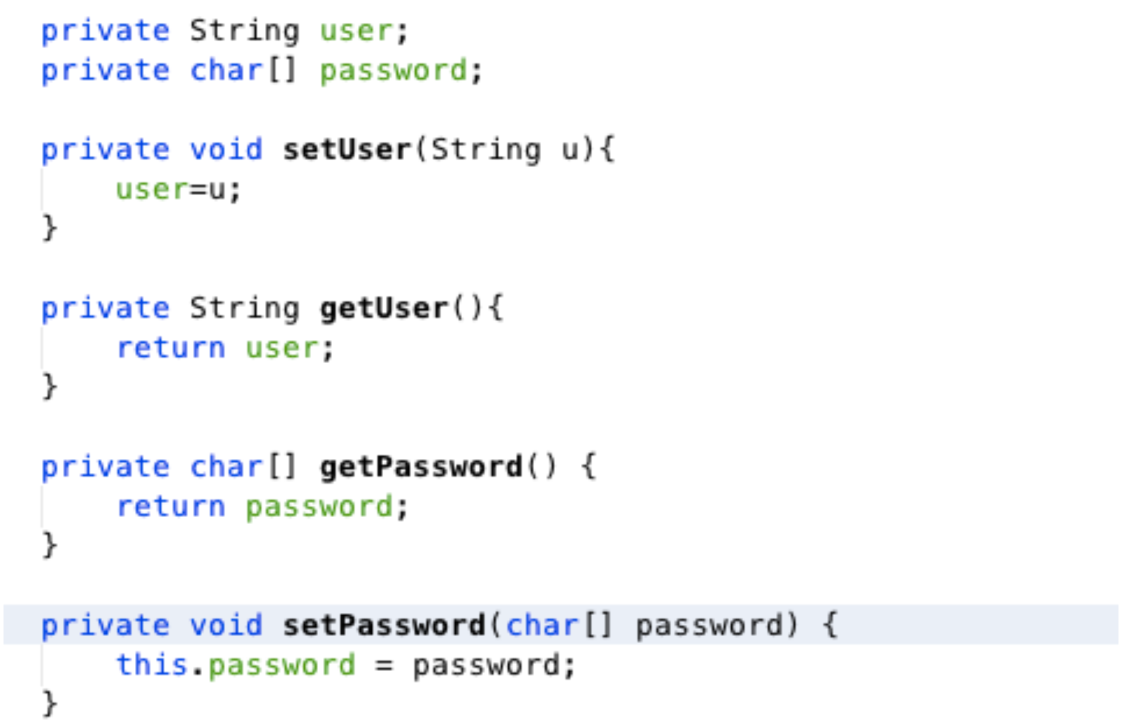


FIGURE 2 LOGIN FORM

2.1 Encapsulation Technique

In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class(Tutorialspoint.com).

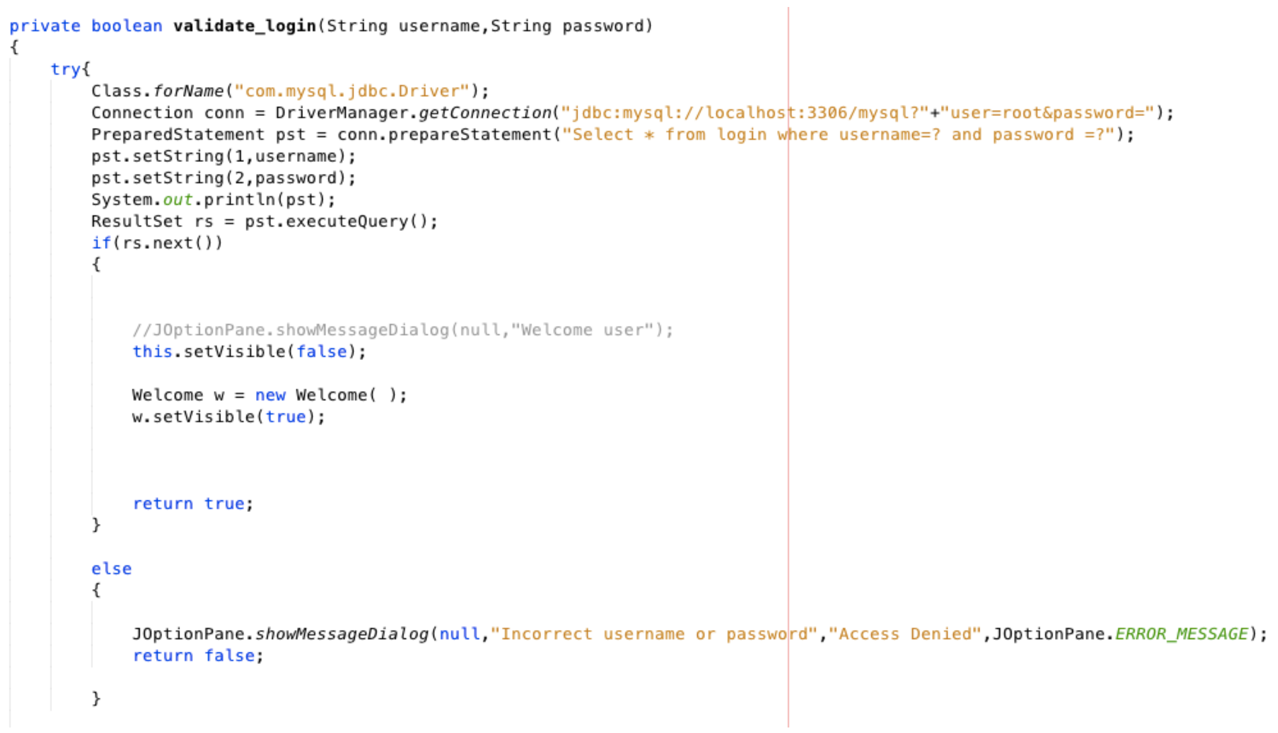


```
private String user;
private char[] password;

private void setUser(String u){
    user=u;
}

private String getUser(){
    return user;
}

private char[] getPassword() {
    return password;
}

private void setPassword(char[] password) {
    this.password = password;
}
```

FIGURE 3

The user input variables : User ID and Password are encapsulated. These fields of the Login2 class are read-only or write-only. The Login2 class can have total control over what is stored in its fields. This type of encapsulation leads to high security in the application as it prevents the code and data from being randomly accessed by other code defined outside the class. Access to the data and code is limited. It also enables the implemented code to be modified easily. This leads to flexibility and extensibility of the code.

2.2 Verification Function

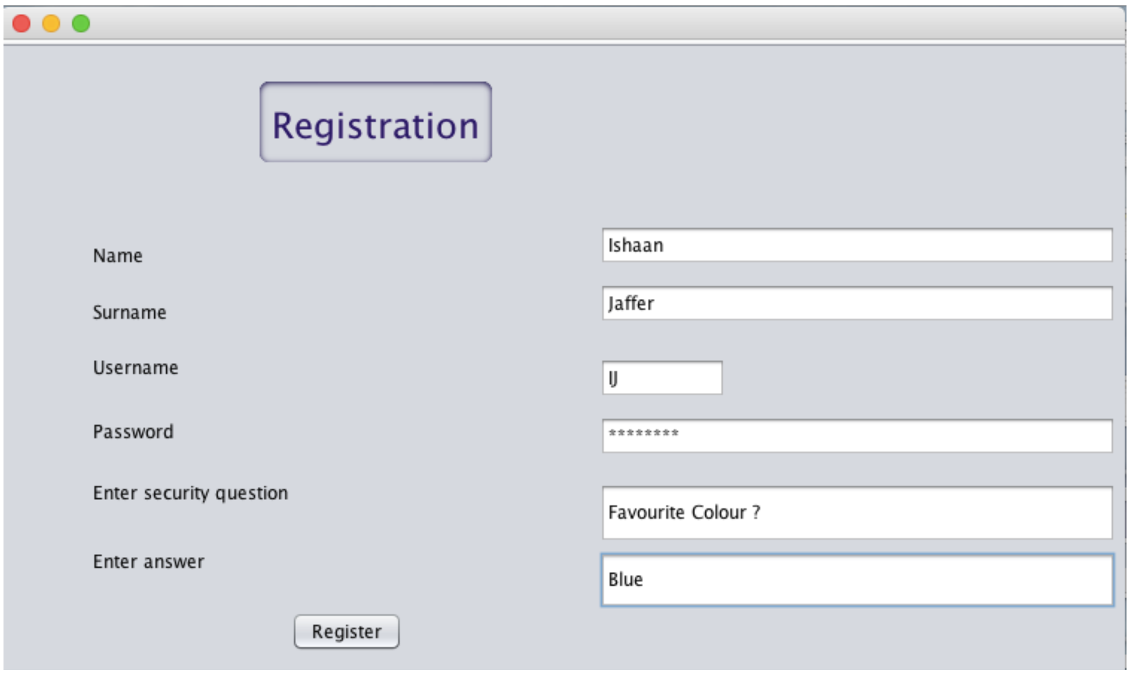


```
private boolean validate_login(String username,String password)
{
    try{
        Class.forName("com.mysql.jdbc.Driver");
        Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/mysql?"+"user=root&password=");
        PreparedStatement pst = conn.prepareStatement("Select * from login where username=? and password =?");
        pst.setString(1,username);
        pst.setString(2,password);
        System.out.println(pst);
        ResultSet rs = pst.executeQuery();
        if(rs.next())
        {


            //JOptionPane.showMessageDialog(null,"Welcome user");
            this.setVisible(false);

            Welcome w = new Welcome( );
            w.setVisible(true);



            return true;
        }

        else
        {

            JOptionPane.showMessageDialog(null,"Incorrect username or password","Access Denied",JOptionPane.ERROR_MESSAGE);
            return false;

        }
```

FIGURE 4

validate_login( ) function checks whether user input data matches with the credentials of the client present in the database. By using queries the database values are retrieved and compared with username, password. This ensure security of the software.

3. Registration of a new user



Name	Ishaan
Surname	Jaffer
Username	IJ
Password	********
Enter security question	Favourite Colour ?
Enter answer	Blue

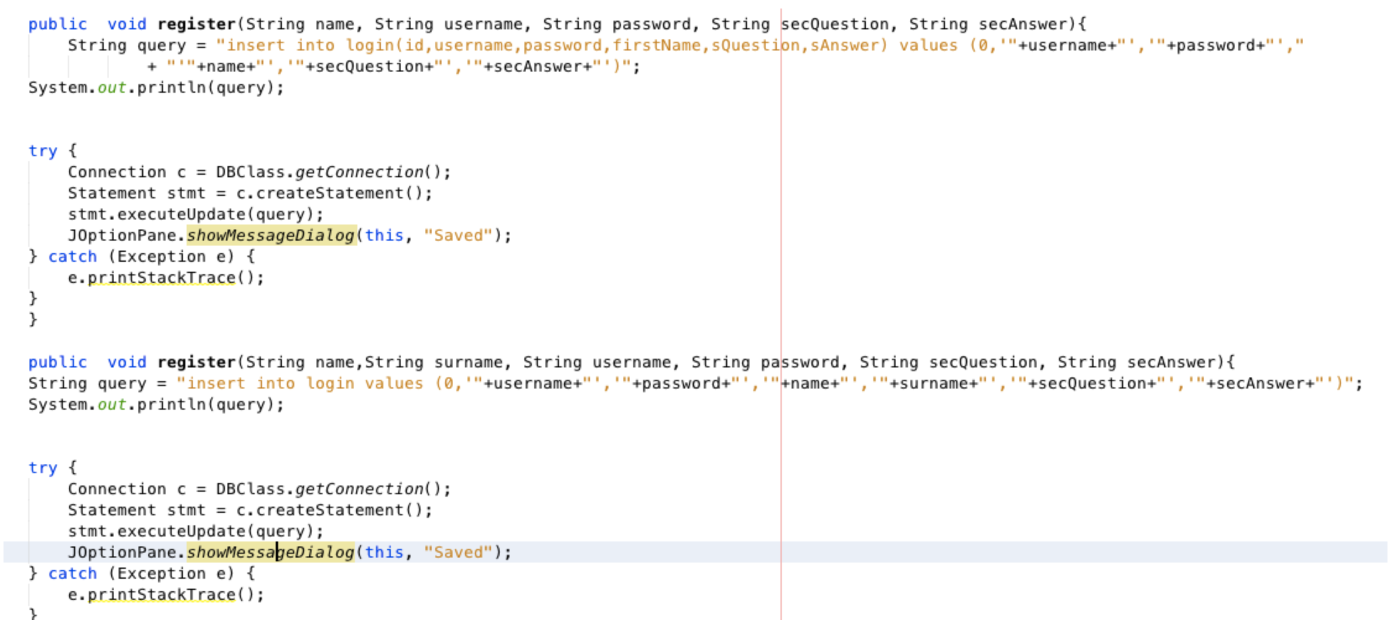
FIGURE 5 : THE FOLLOWING FORM IS USED TO REGISTER A NEW USER TO THE SYSTEM.

Query Used :
This query inserts data into the login table of the database

```
String query = "insert into login values (0,'"+username+"','"+password+"','"+name+"','"+surname+"','"+secQuestion+"','"+secAnswer+"')
```

3.1 Technique Used : Polymorphism

The use of multiple methods having the same name but different functionality, provided the argument or parameters are different (Singh).

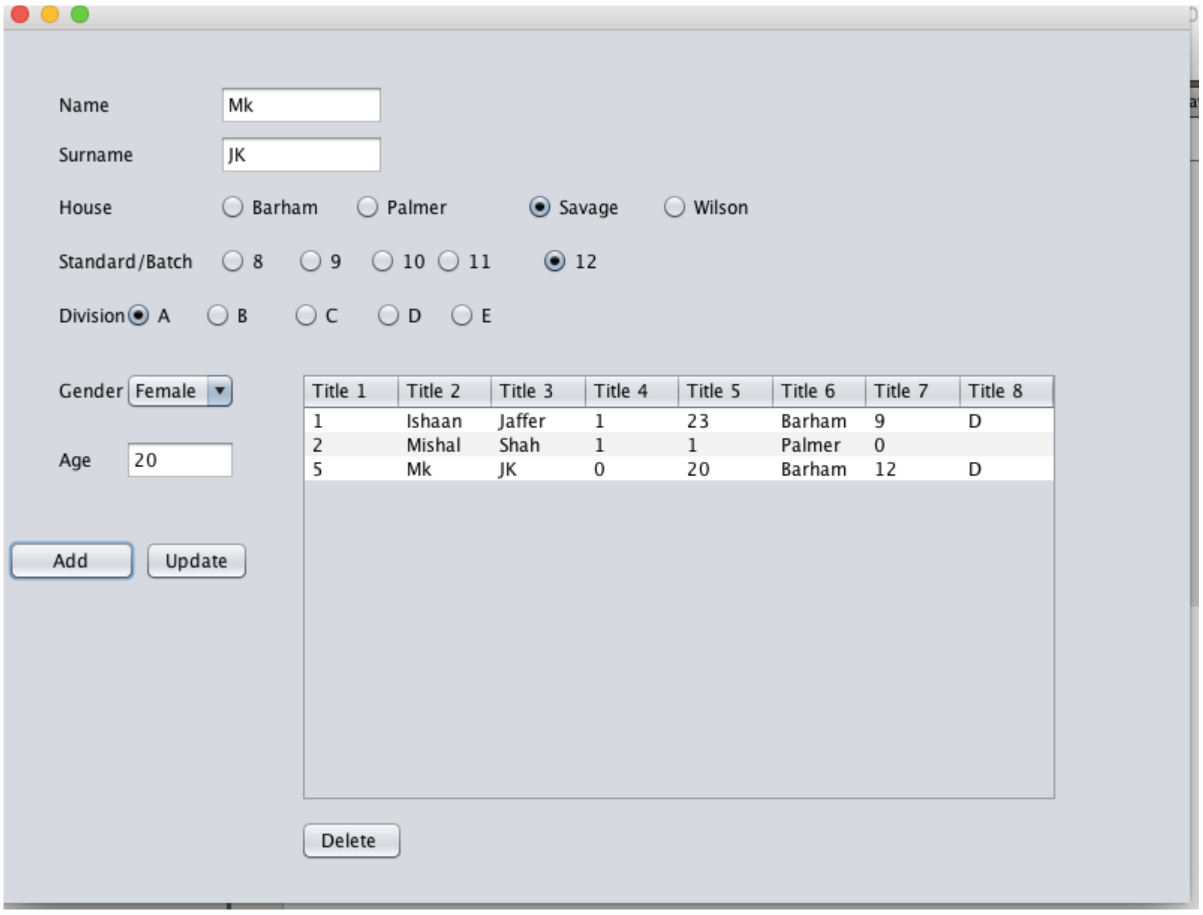


```java
public  void register(String name, String username, String password, String secQuestion, String secAnswer){
    String query = "insert into login(id,username,password,firstName,sQuestion,sAnswer) values (0,'"+username+"','"+password+"',"
            + "'"+name+"','"+secQuestion+"','"+secAnswer+"')";
System.out.println(query);


try {
    Connection c = DBClass.getConnection();
    Statement stmt = c.createStatement();
    stmt.executeUpdate(query);
    JOptionPane.showMessageDialog(this, "Saved");
} catch (Exception e) {
    e.printStackTrace();
}
}

public  void register(String name,String surname, String username, String password, String secQuestion, String secAnswer){
String query = "insert into login values (0,'"+username+"','"+password+"','"+name+"','"+surname+"','"+secQuestion+"','"+secAnswer+"')";
System.out.println(query);


try {
    Connection c = DBClass.getConnection();
    Statement stmt = c.createStatement();
    stmt.executeUpdate(query);
    JOptionPane.showMessageDialog(this, "Saved");
} catch (Exception e) {
    e.printStackTrace();
}
```

FIGURE 6

public void register ( ) has been used twice; Both methods have different parameters. (String surname is present in the second register( ). My Client had conveyed to me that in some cases some users may not enter their surname and she would like the user to still be registered. Polymorphism allows both methods that perform the similar function of registering a new user to be accessed through a common name register( ). If “String Surname” has not been entered the program will still be able to run and data will be stored in the database. By using polymorphism there is code re-use.

4. Registering a new student



Name	Mk
Surname	JK
House	Barham Palmer [✓ Savage] Wilson
Standard/Batch	8 9 10 11 [✓ 12]
Division	[✓ A] B C D E
Gender	Female
Age	20

Title 1	Title 2	Title 3	Title 4	Title 5	Title 6	Title 7	Title 8
1	Ishaan	Jaffer	1	23	Barham	9	D
2	Mishal	Shah	1	1	Palmer	0	
5	Mk	JK	0	20	Barham	12	D

FIGURE 7 DATA ENTRY FORM TO REGISTER A STUDENT

The subform shows the contents of the Students table in the database.
It is based on a complex query and relationship between the tables and ensures that the form and subform are linked on Student_ID to ensure that the data relates to the student.

Query Used:

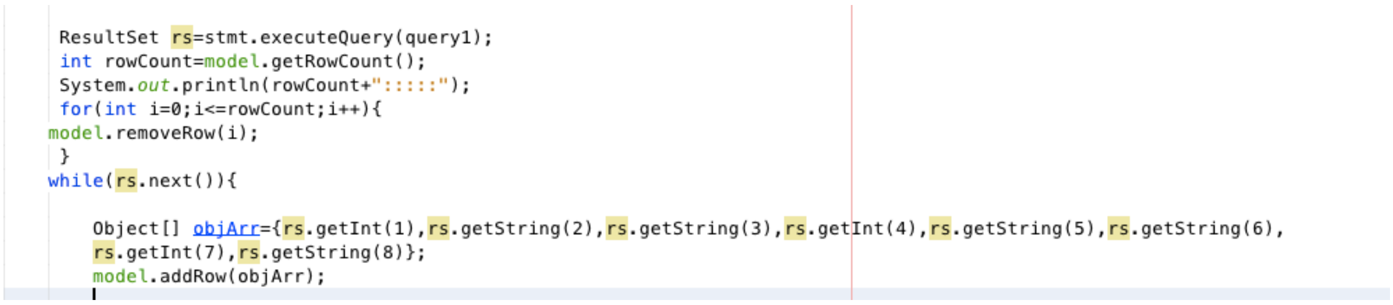
```
String query = "insert into student values(0,'"+name+"','"+surname+"',"+gen+",'"+age+"','"+house+"',"
        + ""+standard+",'"+division+"')";
```

To add a student into the student table

```
String sql2="update student set name='"+name+"', surname='"+surname+"',gender="+gen+",age='"+age+"',house='"+house+"',"
        + "standard="+standard+",division='"+division+"' where studentId="+id+"";
```

To update/edit pre existing values

4.1 Technique Used : Array Of Objects

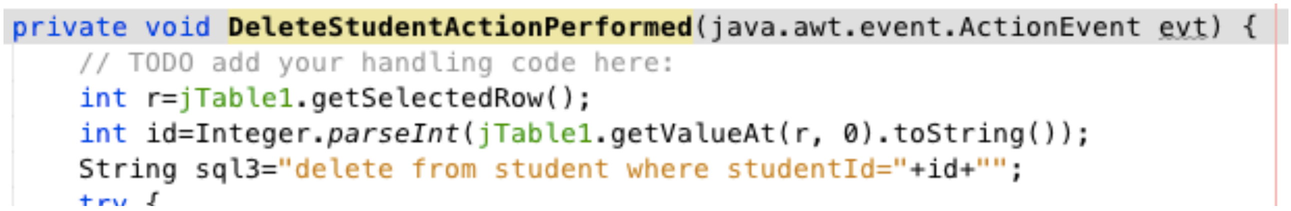


```
ResultSet rs=stmt.executeQuery(query1);
int rowCount=model.getRowCount();
System.out.println(rowCount+":::::");
for(int i=0;i<=rowCount;i++){
model.removeRow(i);
}
while(rs.next()){

    Object[] objArr={rs.getInt(1),rs.getString(2),rs.getString(3),rs.getInt(4),rs.getString(5),rs.getString(6),
    rs.getInt(7),rs.getString(8)};
    model.addRow(objArr);
```

FIGURE 8

An array of objects of the different input data(StudentId(1), Name(2), Surname(3), Gender(4), Age(5), House(6), Standard(7), Division(8)) has been created. The array stores the input data and then displays the data in the table: model; shown in figure 6.

4.2 Delete Selected Student

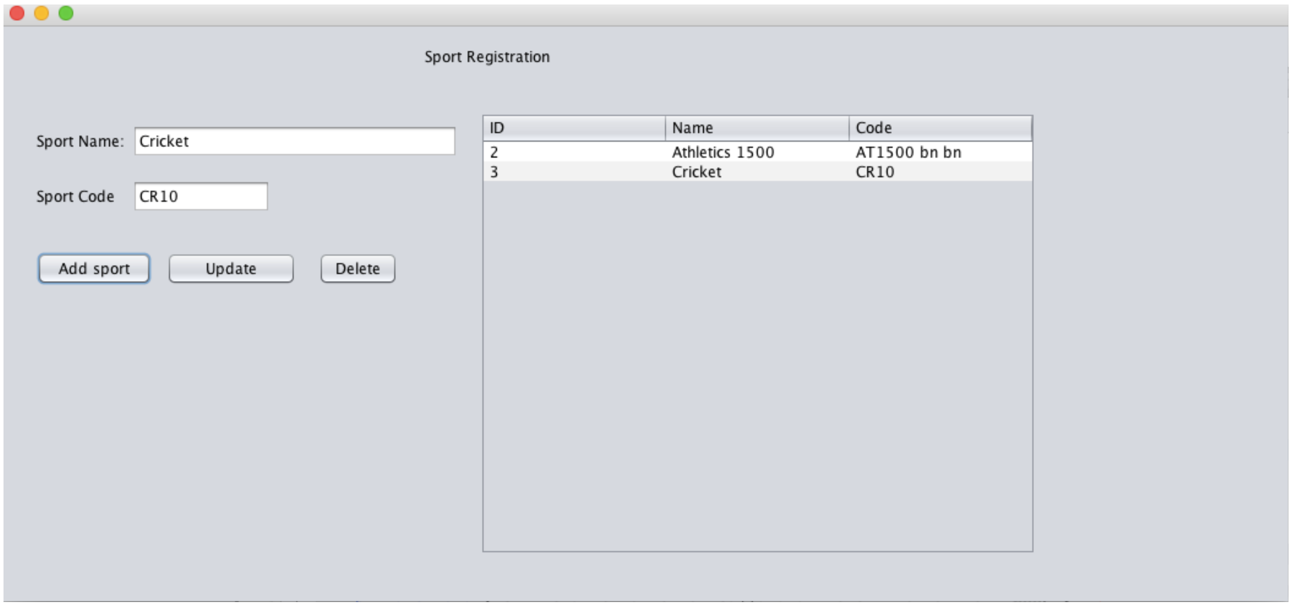


```
private void DeleteStudentActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int r=jTable1.getSelectedRow();
    int id=Integer.parseInt(jTable1.getValueAt(r, 0).toString());
    String sql3="delete from student where studentId="+id+"";
    try {
```

FIGURE 9
A student is selected in the table in figure6 after which the id = value at (r,0) is obtained and the student with the id is deleted from the table.

5. Adding a new sport to the database

Each sport is assigned a specific code. For example: Swimming 200 metre freestyle gets the code as “SW200F”. The values can be updated or deleted.



Sport Name: Cricket

Sport Code: CR10

ID	Name	Code
2	Athletics 1500	AT1500 bn bn
3	Cricket	CR10

FIGURE 10

5.1 Query Used:

```
String query = "insert into sport values(0,'"+sportName+"','"+sportCode+"')";
```

5.2 Technique Used: Constructor

A constructor is a method having the same name as the class. It has been used in all classes in order to initialise the variables.

```
public class SportRegistration extends javax.swing.JFrame {

    /**
     * Creates new form SportRegistration
     */
    public SportRegistration() {
        initComponents();
        model=(DefaultTableModel) jTable1.getModel();
        String sportName = jTextField1.getText();
        String sportCode = jTextField2.getText();

    }
```

FIGURE 11

6. Entering Results into the database

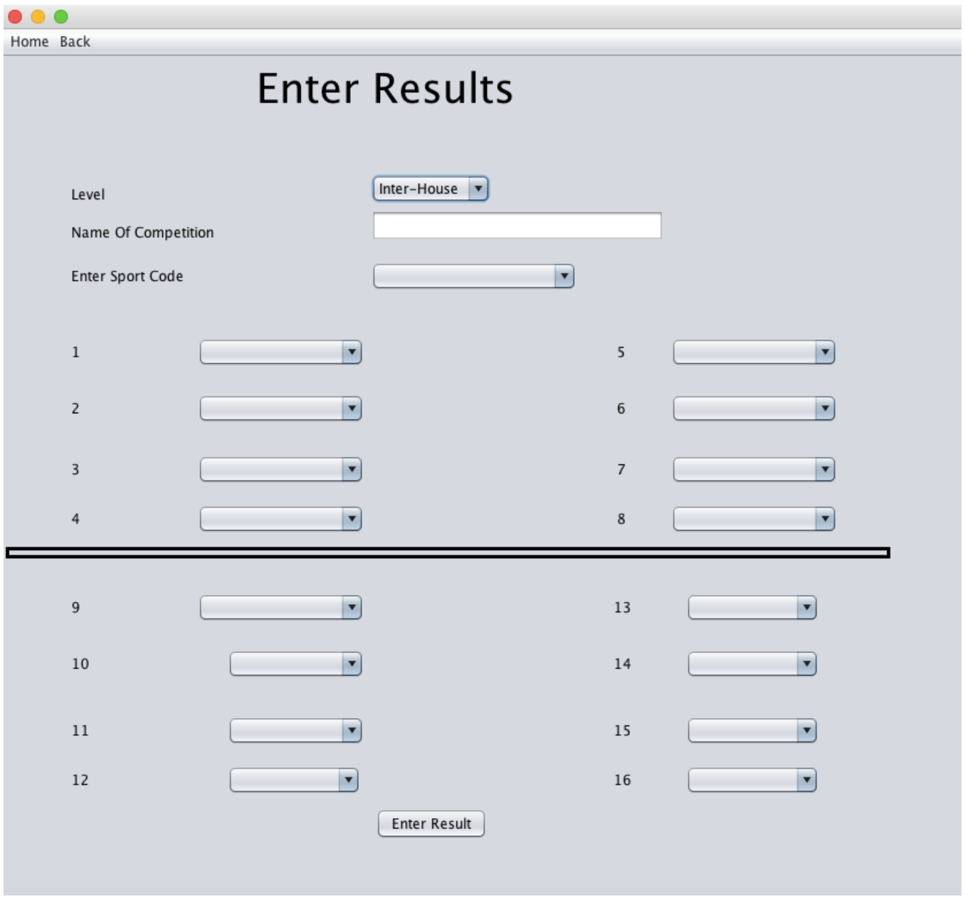


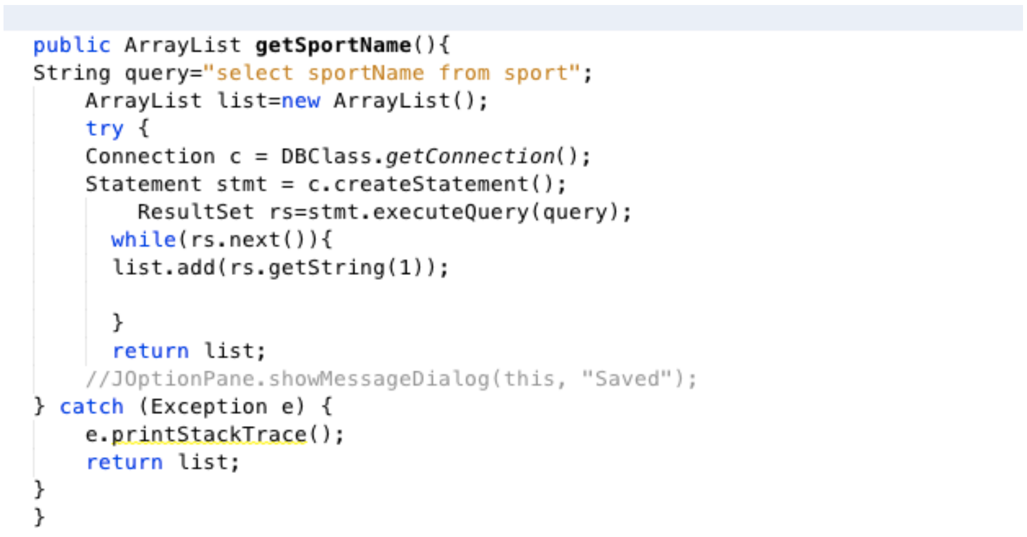
FIGURE 12: ENTER RESULTS FORM

6.1 Query Used:

```
String query = "insert into result values(0,'"+name+"','"+sportCode+"','"+level+"',"+rank[1]+","+rank[2]+","+rank[3]+","
        + ""+rank[4]+","+rank[5]+","+ ""+rank[6]+","+rank[7]+","+rank[8]+","+rank[9]+","+rank[10]+","+rank[11]+","
        + ""+rank[12]+","+rank[13]+","+rank[14]+","+rank[15]+","+rank[16]+")";
```

6.2 Techniques used : Array List, Iterator and Array

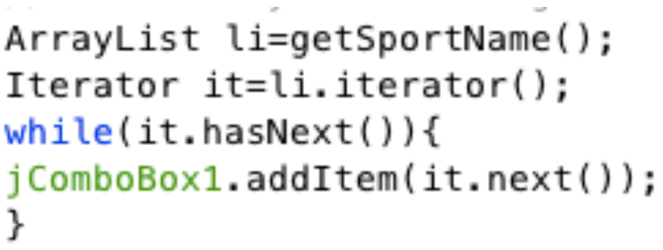
Java ArrayList class uses a dynamic array for storing the elements(“Java ArrayList”).



```
public ArrayList getSportName(){
String query="select sportName from sport";
    ArrayList list=new ArrayList();
    try {
    Connection c = DBClass.getConnection();
    Statement stmt = c.createStatement();
        ResultSet rs=stmt.executeQuery(query);
      while(rs.next()){
      list.add(rs.getString(1));

      }
      return list;
    //JOptionPane.showMessageDialog(this, "Saved");
} catch (Exception e) {
    e.printStackTrace();
    return list;
}
}
```

FIGURE 13

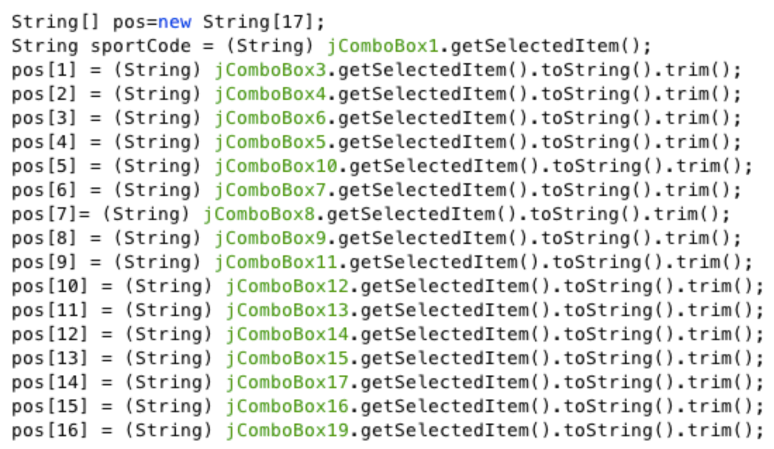
An array list has been used to store the different sports that can be taken part in (eg- Athletics, Rugby,Table-Tennis,Cricket etc) and displays it in the combobox in the entry form. An array list is used as when a sport is added to the list the collection is automatically enlarged and when a sport is removed the list can be automatically shrunk.



```
ArrayList li=getSportName();
Iterator it=li.iterator();
while(it.hasNext()){
jComboBox1.addItem(it.next());
}
```

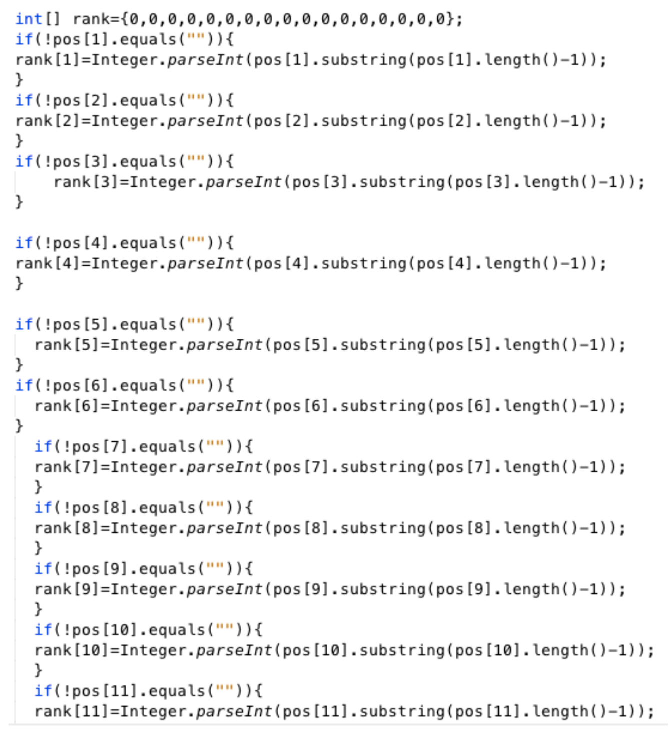
FIGURE 14

The Iterator cycles through the array list and displays each sport in the combo box.



```java
String[] pos=new String[17];
String sportCode = (String) jComboBox1.getSelectedItem();
pos[1] = (String) jComboBox3.getSelectedItem().toString().trim();
pos[2] = (String) jComboBox4.getSelectedItem().toString().trim();
pos[3] = (String) jComboBox6.getSelectedItem().toString().trim();
pos[4] = (String) jComboBox5.getSelectedItem().toString().trim();
pos[5] = (String) jComboBox10.getSelectedItem().toString().trim();
pos[6] = (String) jComboBox7.getSelectedItem().toString().trim();
pos[7]= (String) jComboBox8.getSelectedItem().toString().trim();
pos[8] = (String) jComboBox9.getSelectedItem().toString().trim();
pos[9] = (String) jComboBox11.getSelectedItem().toString().trim();
pos[10] = (String) jComboBox12.getSelectedItem().toString().trim();
pos[11] = (String) jComboBox13.getSelectedItem().toString().trim();
pos[12] = (String) jComboBox14.getSelectedItem().toString().trim();
pos[13] = (String) jComboBox15.getSelectedItem().toString().trim();
pos[14] = (String) jComboBox17.getSelectedItem().toString().trim();
pos[15] = (String) jComboBox16.getSelectedItem().toString().trim();
pos[16] = (String) jComboBox19.getSelectedItem().toString().trim();
```

FIGURE 15
The Array pos[ ] represents multiple data items of the same type(String) by using a single name(pos[ ]).



```java
int[] rank={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
if(!pos[1].equals("")){
rank[1]=Integer.parseInt(pos[1].substring(pos[1].length()-1));
}
if(!pos[2].equals("")){
rank[2]=Integer.parseInt(pos[2].substring(pos[2].length()-1));
}
if(!pos[3].equals("")){
    rank[3]=Integer.parseInt(pos[3].substring(pos[3].length()-1));
}

if(!pos[4].equals("")){
rank[4]=Integer.parseInt(pos[4].substring(pos[4].length()-1));
}

if(!pos[5].equals("")){
  rank[5]=Integer.parseInt(pos[5].substring(pos[5].length()-1));
}
if(!pos[6].equals("")){
  rank[6]=Integer.parseInt(pos[6].substring(pos[6].length()-1));
}
  if(!pos[7].equals("")){
  rank[7]=Integer.parseInt(pos[7].substring(pos[7].length()-1));
  }
  if(!pos[8].equals("")){
  rank[8]=Integer.parseInt(pos[8].substring(pos[8].length()-1));
  }
  if(!pos[9].equals("")){
  rank[9]=Integer.parseInt(pos[9].substring(pos[9].length()-1));
  }
  if(!pos[10].equals("")){
  rank[10]=Integer.parseInt(pos[10].substring(pos[10].length()-1));
  }
  if(!pos[11].equals("")){
  rank[11]=Integer.parseInt(pos[11].substring(pos[11].length()-1));
```

FIGURE 16
All values in the array rank[ ] are stored as 0 initially. If a student wins or participates in the sport then the rank[ ] at that position = studentId

7. Searching for a Student

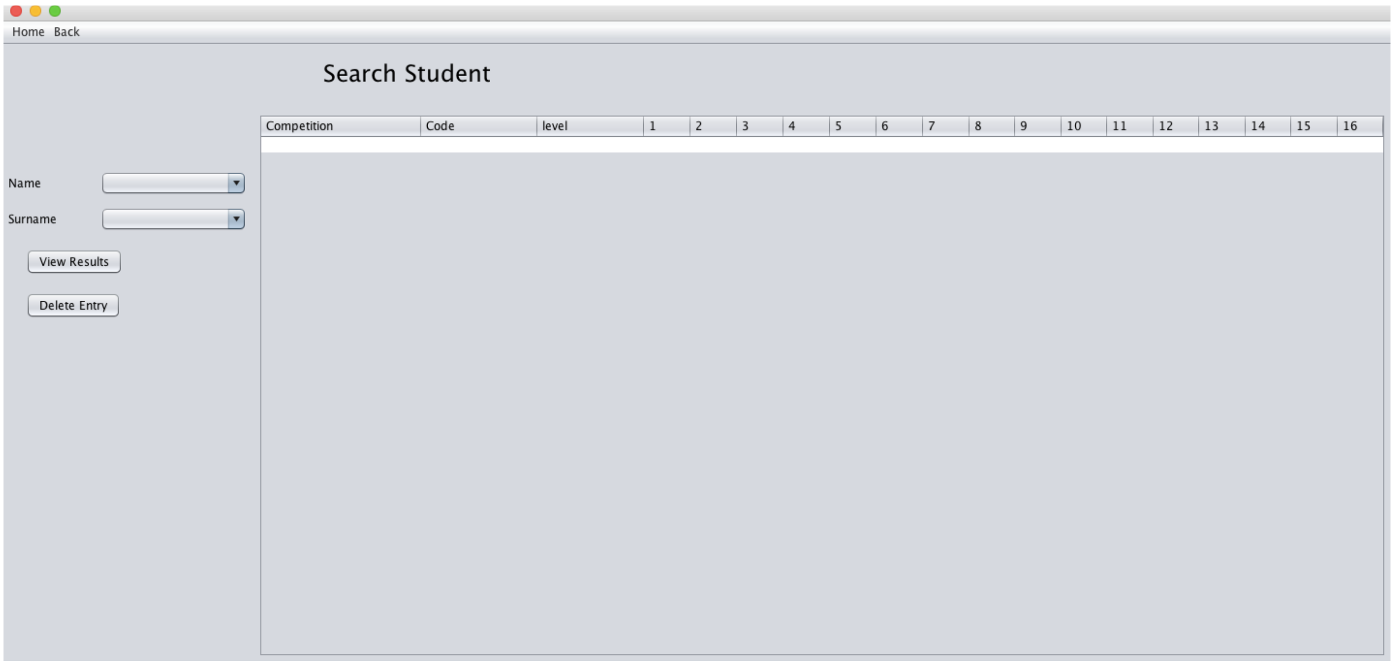
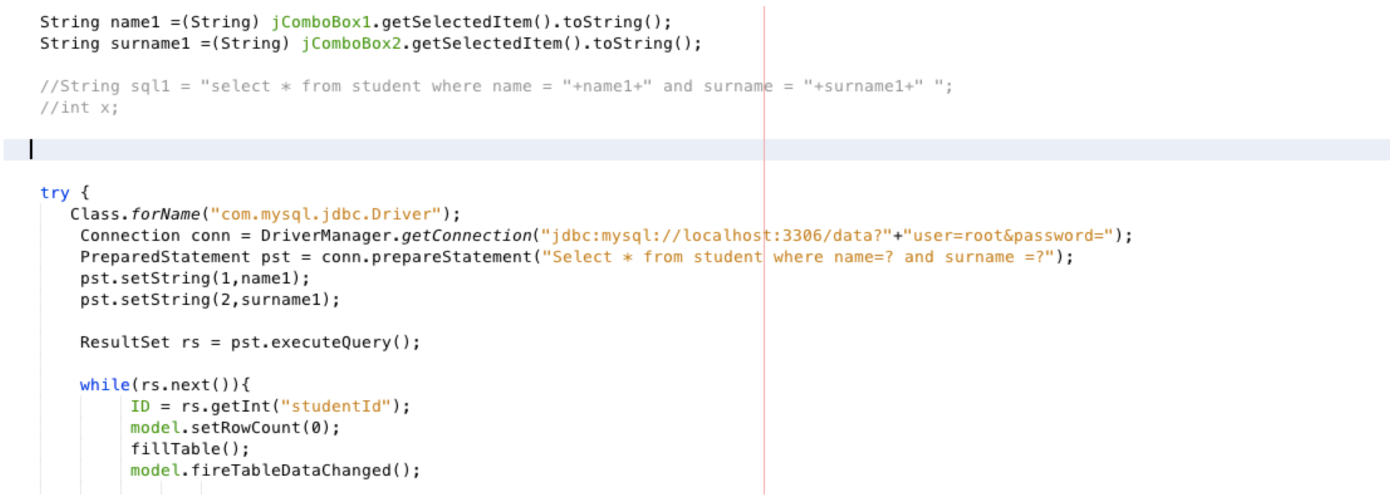


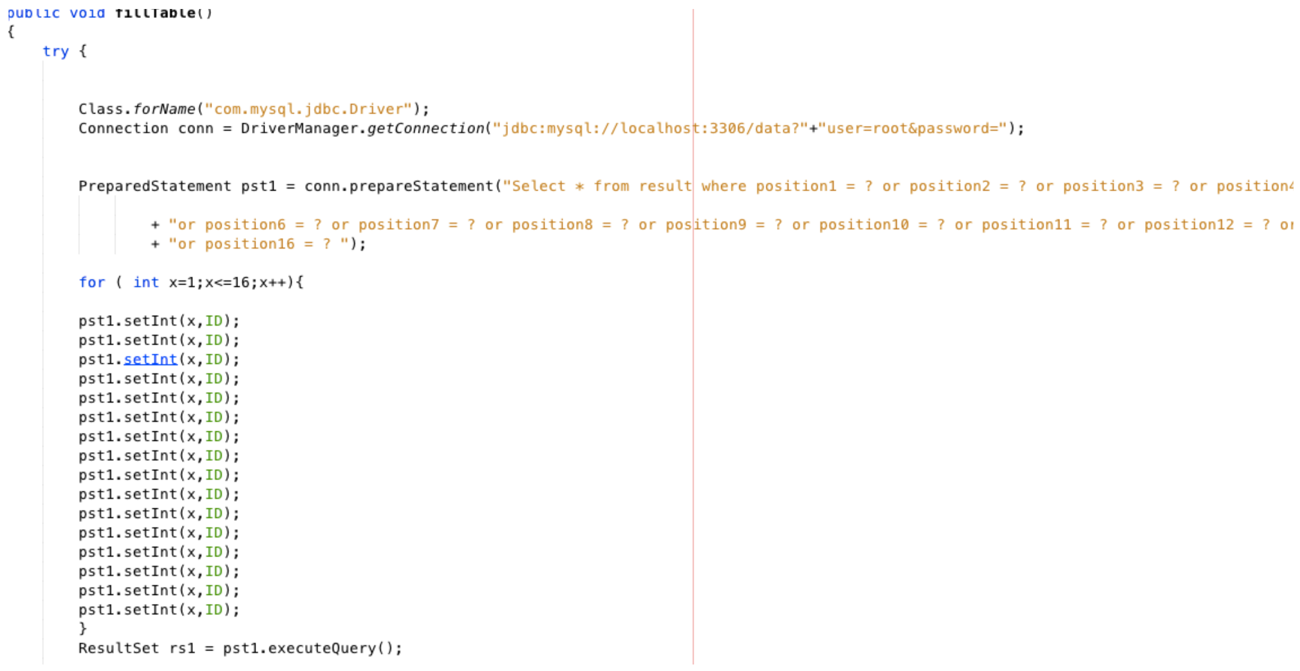
FIGURE 17: DATA ENTRY FORM FOR SEARCHING A STUDENT



```
    String name1 =(String) jComboBox1.getSelectedItem().toString();
    String surname1 =(String) jComboBox2.getSelectedItem().toString();

    //String sql1 = "select * from student where name = "+name1+" and surname = "+surname1+" ";
    //int x;


    try {
       Class.forName("com.mysql.jdbc.Driver");
        Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/data?"+"user=root&password=");
        PreparedStatement pst = conn.prepareStatement("Select * from student where name=? and surname =?");
        pst.setString(1,name1);
        pst.setString(2,surname1);

        ResultSet rs = pst.executeQuery();

        while(rs.next()){
             ID = rs.getInt("studentId");
             model.setRowCount(0);
             fillTable();
             model.fireTableDataChanged();
```

FIGURE 18

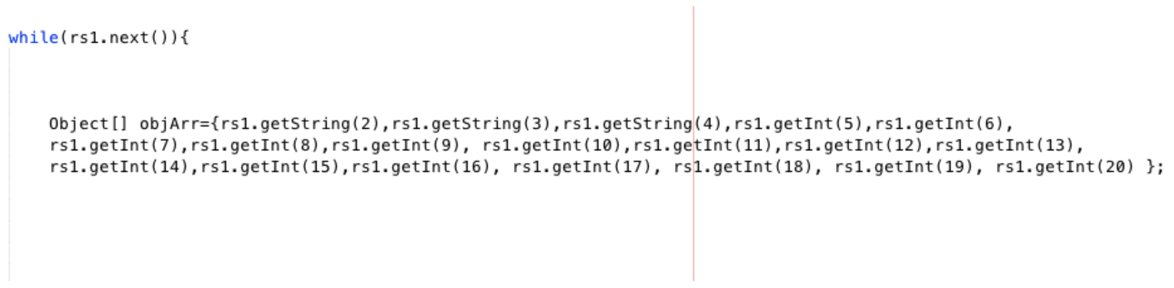
Name and surname entered are used to search student table. studentId of the student is obtained and set as ID.



```java
public void fillTable()
{
    try {


        Class.forName("com.mysql.jdbc.Driver");
        Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/data?"+"user=root&password=");


        PreparedStatement pst1 = conn.prepareStatement("Select * from result where position1 = ? or position2 = ? or position3 = ? or position4

                + "or position6 = ? or position7 = ? or position8 = ? or position9 = ? or position10 = ? or position11 = ? or position12 = ? o
                + "or position16 = ? ");

        for ( int x=1;x<=16;x++){

        pst1.setInt(x,ID);
        pst1.setInt(x,ID);
        pst1.setInt(x,ID);
        pst1.setInt(x,ID);
        pst1.setInt(x,ID);
        pst1.setInt(x,ID);
        pst1.setInt(x,ID);
        pst1.setInt(x,ID);
        pst1.setInt(x,ID);
        pst1.setInt(x,ID);
        pst1.setInt(x,ID);
        pst1.setInt(x,ID);
        pst1.setInt(x,ID);
        pst1.setInt(x,ID);
        pst1.setInt(x,ID);
        pst1.setInt(x,ID);
        }
        ResultSet rs1 = pst1.executeQuery();
```

FIGURE 19

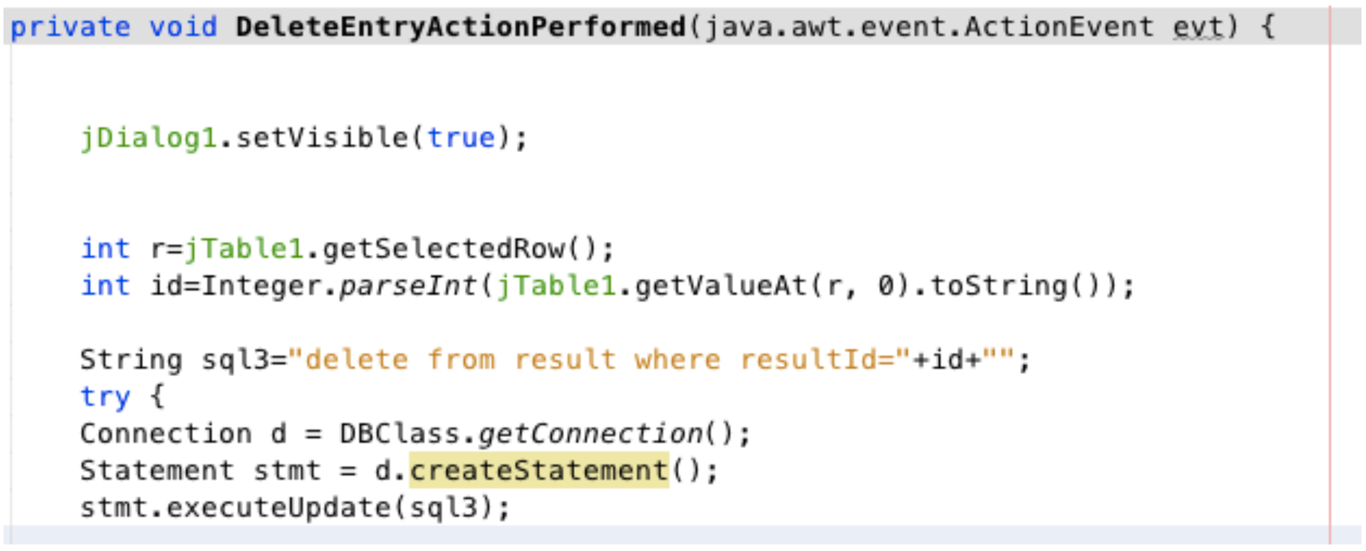
Using the prepared statement it was checked if the ID matches with any of the ID's stored in positions 1 to 16 in all sports. If there is a match it means the student participated or won in the sport.



```java
  while(rs1.next()){



      Object[] objArr={rs1.getString(2),rs1.getString(3),rs1.getString(4),rs1.getInt(5),rs1.getInt(6),
      rs1.getInt(7),rs1.getInt(8),rs1.getInt(9), rs1.getInt(10),rs1.getInt(11),rs1.getInt(12),rs1.getInt(13),
      rs1.getInt(14),rs1.getInt(15),rs1.getInt(16), rs1.getInt(17), rs1.getInt(18), rs1.getInt(19), rs1.getInt(20) };
```

FIGURE 20

An Object Array is then used to store the obtained data. This data is then represented in a table. An object Array has been used as the data obtained is in the from of Strings and Integers.



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


    jDialog1.setVisible(true);


    int r=jTable1.getSelectedRow();
    int id=Integer.parseInt(jTable1.getValueAt(r, 0).toString());

    String sql3="delete from result where resultId="+id+"";
    try {
    Connection d = DBClass.getConnection();
    Statement stmt = d.createStatement();
    stmt.executeUpdate(sql3);
```

FIGURE 21

Process of deleting an entry, ValueAt(r,0) is the resultId. The query checks where the resultId selected matches with the resultId in the result table, and then deletes the selected entry.

8. Error Checking Methods

8.1 Combo Boxes

Errors are prevented as the user is not required to type any data but rather select the data that has been retrieved from the tables of the database.

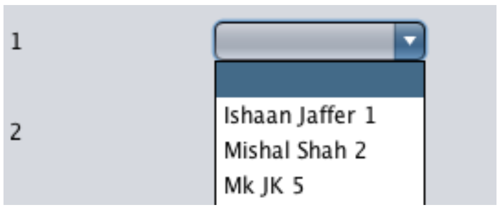


FIGURE 22:NAMES ARE RETRIEVED FROM THE TABLE STUDENT

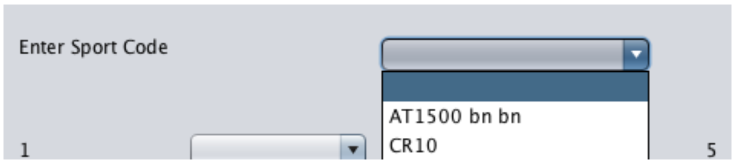
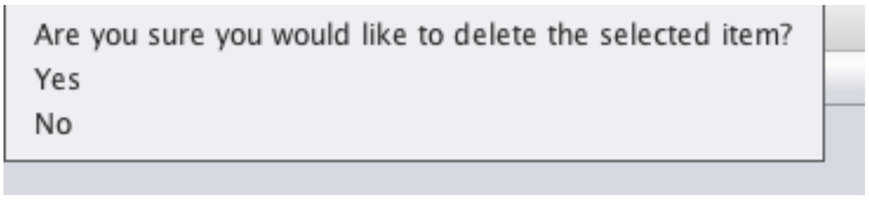


FIGURE 23: SPORT CODES ARE RETRIEVED FROM THE TABLE SPORT

8.2 Popup Menus

The Popup menu displays whenever the user decides to delete an entry from the database. Since my client informed me that she would like a user friendly software, I have added this feature, incase the user clicks delete by mistake crucial data should not be lost.



Are you sure you would like to delete the selected item?
Yes
No

FIGURE 24: POPUP MENU

9. Sorting :

The process of arranging data with similar properties systematically is known as Sorting. The student names, surnames and Sport Codes are sorted alphabetically such that it is easier for the client to select the desired data.

Sorting has been done using the following queries:

```
String sql1 = "select SportCode from sport order by SportCode asc";
```

1 Alphabetical sorting of Sport Code

```
String sql1 = "select name,surname,studentId from student order by name asc";
```

2 Sorting of name, surname, studentId on the basis of alphabetical order of names.

```
String sql1 = "select surname from student order by surname asc";
```

3 Alphabetical sorting of surname

Word Count approx : 800 words

Work Cited - MLA

Singh, Chaitanya. "Polymorphism in Java." Beginnersbook.com. N.p., 29 Apr. 2015. Web. 05 Feb. 2017.

Tutorialspoint.com. "Java Encapsulation." Www.tutorialspoint.com. N.p., n.d. Web. 05 Feb. 2017.