Blood Donors Management System

Submitted by

Bin Yeamin Chowdhury (161-115-150)

&

Ali Ahmed (161-115-128)

Submitted to

Rahatara Ferdousi

Lecturer, Metropolitan University, Sylhet.

Department of Computer Science Engineering

Metropolitan University, Sylhet.

July, 2018

Date of Submission: 12/07/2018

Project Summary

Emergency situations such as accidents, create an immediate and critical need for specific blood types. In addition, advances in medicine have also increased the need of blood for various treatments and surgeries. Since blood can not be produced in the lab, we can collect blood from human donors.

Keeping track of blood donors in hand written document is analog system. In analog system finding information of a specific donor is tiresome. But in digital storage system one can find information of a specific donor just in few seconds. This application store information of blood donors including donors photo, contact info and many more information. User can search donor with specific blood group or from a specific region.

Contents

Chapter
1

Introduction

1.1 Background:

Storing information in hard copy is tiresome, and it is difficult to modify existing data. Sometimes they might be exposed to hazards such as water or pests. Saving data in digital format is a good alternative. And for this we tried to build an application that store information of blood donor's in computer database.

1.2 Objective:

This project is aimed to develop an application that store blood donors information. User can add new donor or modify existing donor very easily. All information of blood donors can be easily accessed or modified. From a huge list of blood donors user can search blood donors of specific blood groups or specific areas.

1.3 Hardware and Software Requirements:

Hardware Requirements:

OS	:	Windows/Linux
Ram	:	2GB
Hard Disk	:	20GB
Processor	:	Celeron/ Pentium/ Core i3/ Core i5/ Core i7

Software Requirements:

Eclipse / Netbeans / any other IDE

jdk 1.8

mysql 5.7

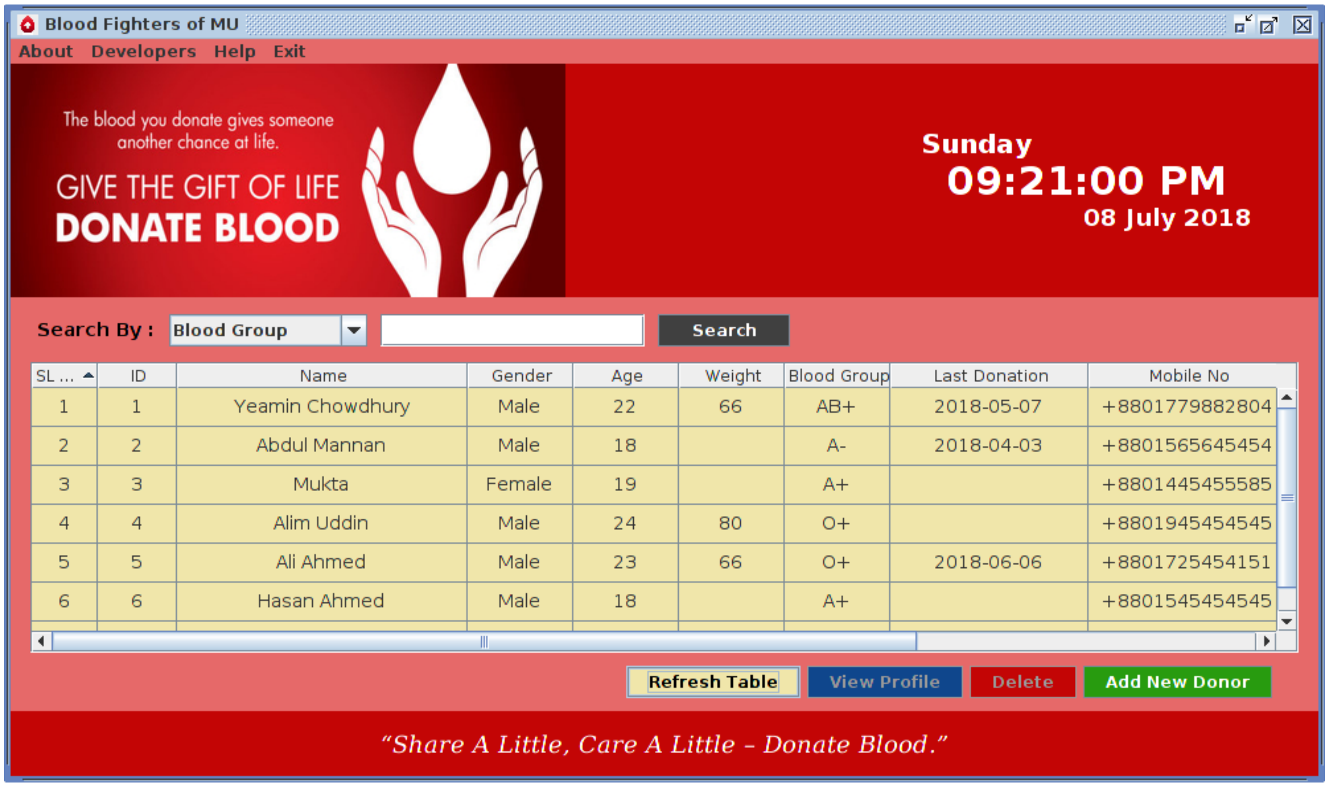
1.4 Technologies used

Technologies	:	Java
Library Used	:	jcalendar-1.4, mysql-connector-java8.0.11
Database	:	Mysql

Chapter 2

Project Description

At start up of application a window will open like Figure 2.1. Here user can see the list blood donors. From this window user can add new donor, view donor's full profile, delete information of donor and search specific donor.



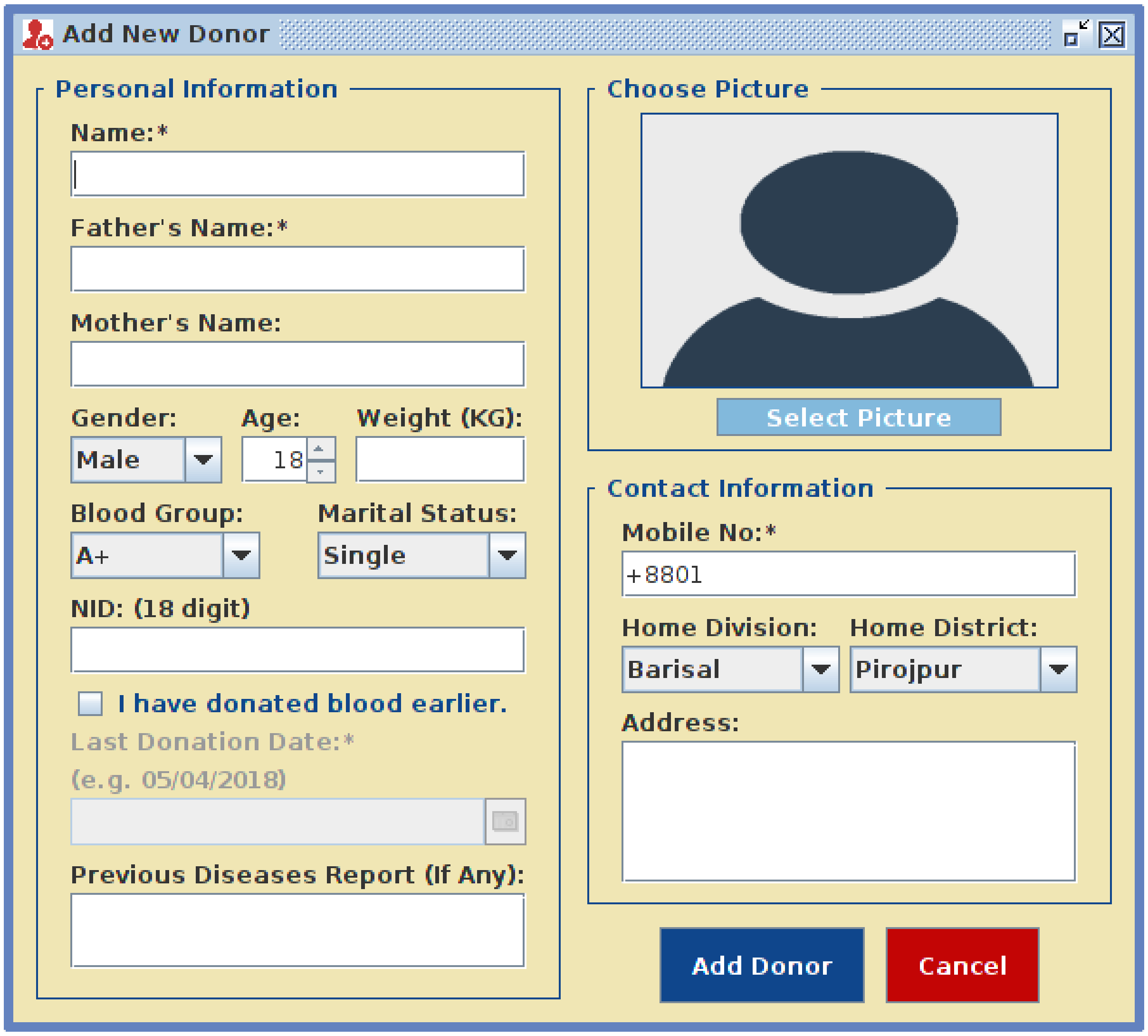
Sunday
09:21:00 PM
08 July 2018

SL ...	ID	Name	Gender	Age	Weight	Blood Group	Last Donation	Mobile No
1	1	Yeamin Chowdhury	Male	22	66	AB+	2018-05-07	+8801779882804
2	2	Abdul Mannan	Male	18		A-	2018-04-03	+8801565645454
3	3	Mukta	Female	19		A+		+8801445455585
4	4	Alim Uddin	Male	24	80	O+		+8801945454545
5	5	Ali Ahmed	Male	23	66	O+	2018-06-06	+8801725454151
6	6	Hasan Ahmed	Male	18		A+		+8801545454545

"Share A Little, Care A Little - Donate Blood."

Figure 2.1: Home Window

To add new donor user have to click 'Add New Donor' button. After clicking this button a new window will up with 'Add New Blood Donor Form' like Figure 2.2. After filling up all necessary field, click on 'Add Donor' button. If donor added successfully a pop-up message will show that data has been inserted successfully on database.



Add New Donor

Personal Information

Name:*

Father's Name:*

Mother's Name:

Gender: Male

Age: 18

Weight (KG):

Blood Group: A+

Marital Status: Single

NID: (18 digit)

I have donated blood earlier.

Last Donation Date:*
(e.g. 05/04/2018)

Previous Diseases Report (If Any):

Choose Picture

Contact Information

Mobile No:* +8801

Home Division: Barisal

Home District: Pirojpur

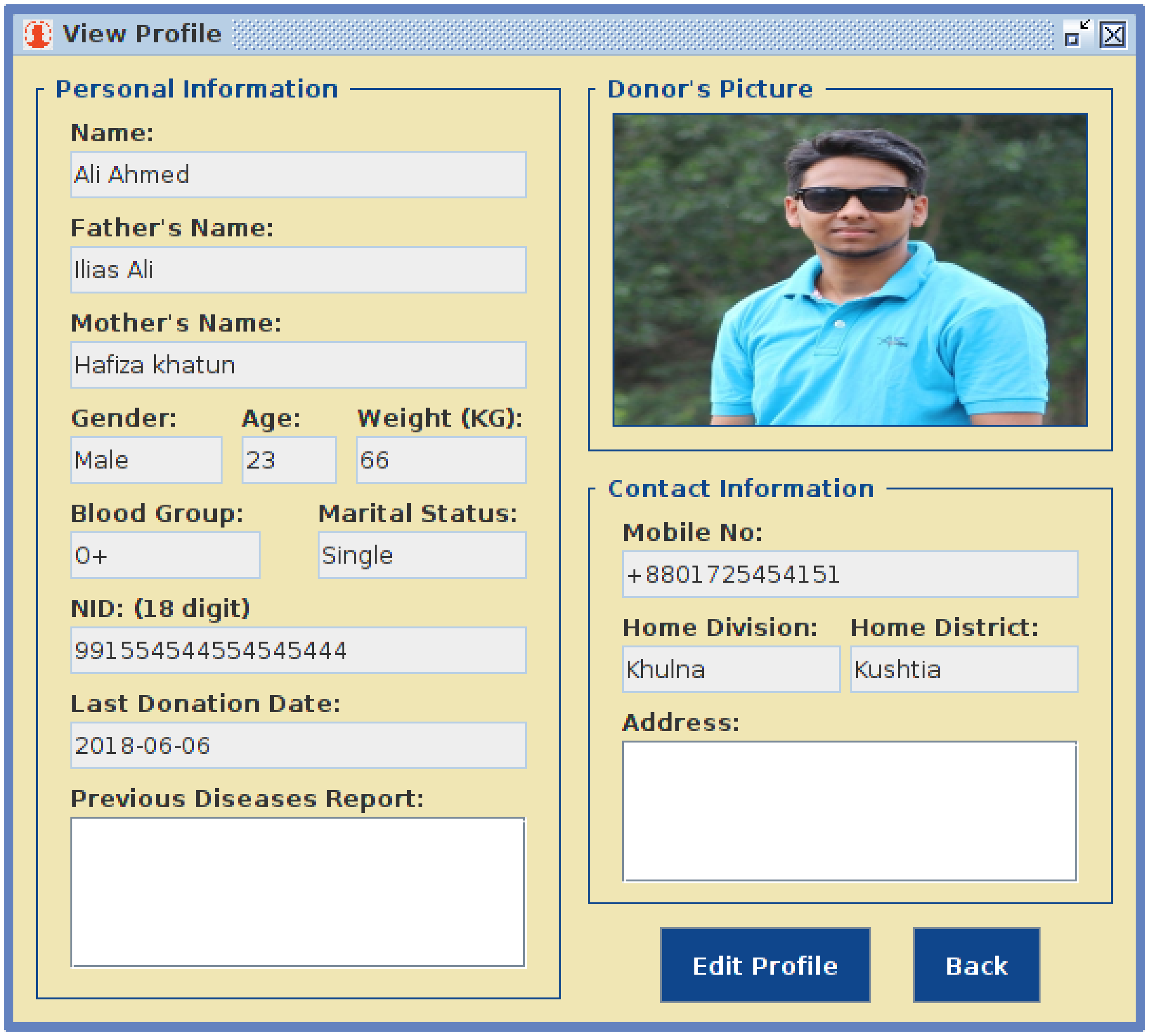
Address:

Figure 2.2: Add New Donor

From donors list user can delete any donor’s info by selecting the donor and then clicking ‘Delete’ button.

After selecting any donor from donors list user can view donors full profile including donor’s photo, father’s name, mother’s name, previous disease history etc by clicking ‘View Profile’ button.

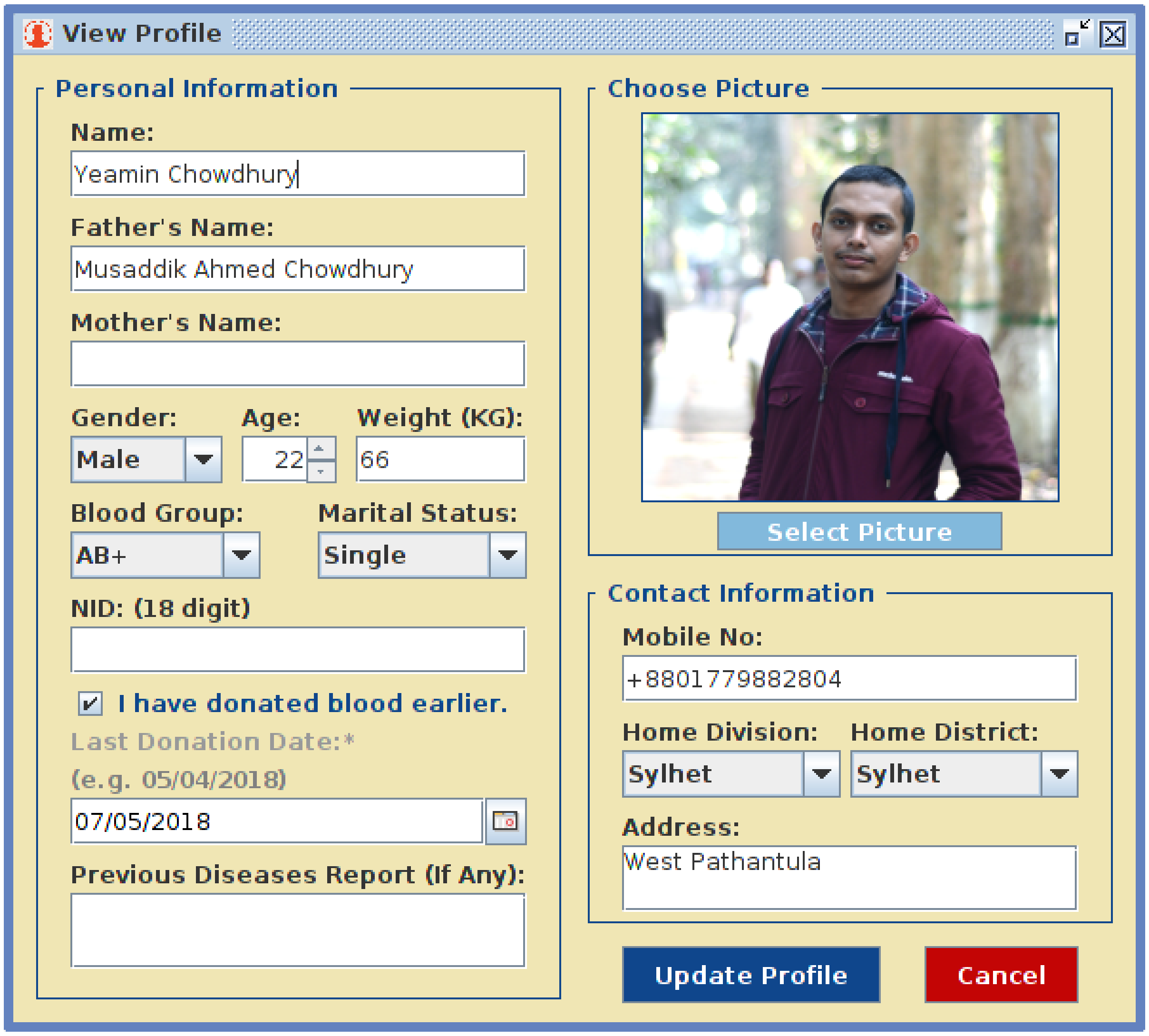
After clicking ‘View Profile’ button a new window will pop with donors profile like Figure 2.3.



Name: Ali Ahmed
Father's Name: Ilias Ali
Mother's Name: Hafiza khatun
Gender: Male
Age: 23
Weight (KG): 66
Blood Group: O+
Marital Status: Single
NID: (18 digit) 991554544554545444
Last Donation Date: 2018-06-06
Previous Diseases Report:
Mobile No: +8801725454151
Home Division: Khulna
Home District: Kushtia
Address:

Figure 2.3: Donor’s Profile

To modify information of existing blood donor, from donor's profile click 'Edit Profile' button. After modifying desired data click on 'Update Profile'.


View Profile

Personal Information

Name: Yeamin Chowdhury

Father's Name: Musaddik Ahmed Chowdhury

Mother's Name:

Gender: Male

Age: 22

Weight (KG): 66

Blood Group: AB+

Marital Status: Single

NID: (18 digit)

☑ I have donated blood earlier.

Last Donation Date:*
(e.g. 05/04/2018)
07/05/2018

Previous Diseases Report (If Any):

Choose Picture

Contact Information

Mobile No: +8801779882804

Home Division: Sylhet

Home District: Sylhet

Address: West Pathantula

Figure 2.4: Edit Donor's Profile

To search specific donor select category from option list above donor's list. Then type search key in the text field beside option's list. After clicking 'Search' button matched result will be displayed in donor's list.

There is a menu bar at the top side of ‘Home’ window like Figure 2.5. Clicking ‘About’ menu will open a new window containing information about this project. ‘Developers’ menu is used for showing developers of this project. ‘Help’ menu will show using instruction in a new window. And by clicking ‘Exit’ menu user can close the application.

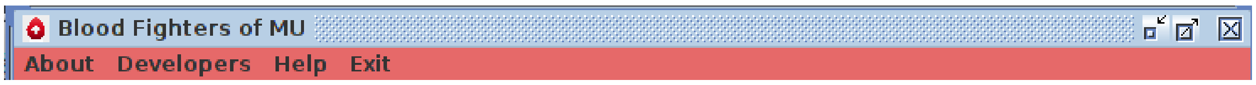


Figure 2.5: Menu Bar

Chapter
3

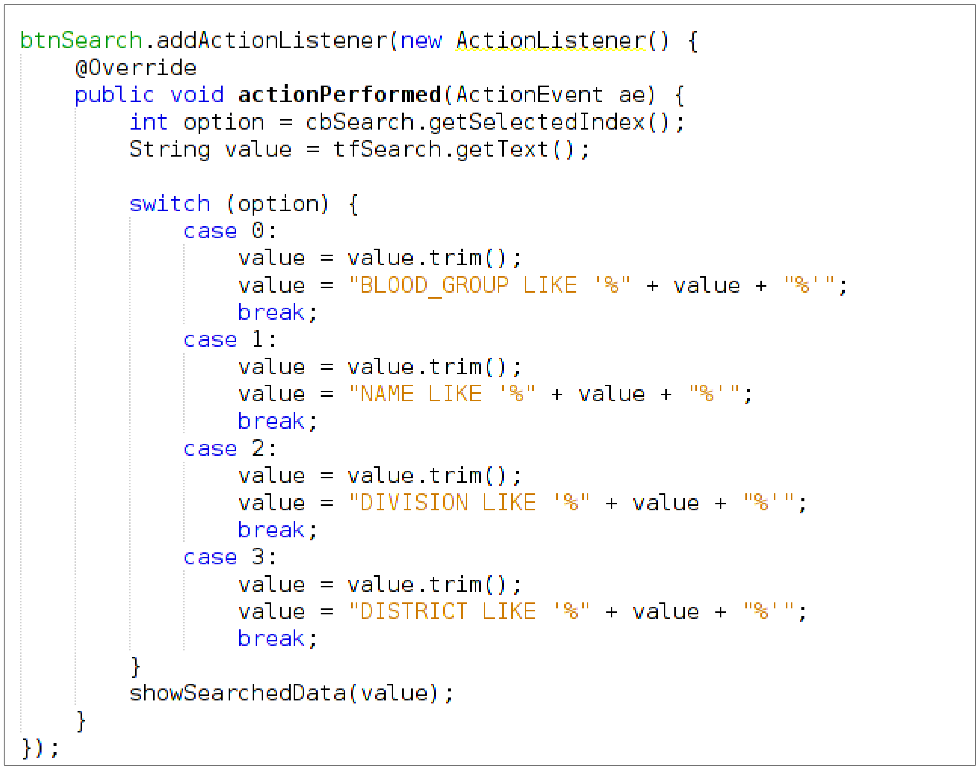
Implementation

Code 3.1 is used for showing digital clock on home window of Figure Fig 2.1.

```
JLabel lblDay = new JLabel();
lblDay.setFont(new Font("Sans", Font.BOLD, 20));                //font: Sans, Style: Bold, Size: 20
lblDay.setForeground(Color.WHITE);                              //font color: White
lblDay.setHorizontalAlignment(JLabel.LEFT);                     //left alignment

JLabel lblTime = new JLabel();
lblTime.setFont(new Font("Sans", Font.BOLD, 30));               //font: Sans, Style: Bold, Size: 30
lblTime.setForeground(Color.WHITE);                             //font color: White
lblTime.setHorizontalAlignment(JLabel.CENTER);                  //center alignment

JLabel lblDate = new JLabel();
lblDate.setFont(new Font("Sans", Font.BOLD, 18));               //font: Sans, Style: Bold, Size: 18
lblDate.setForeground(Color.WHITE);                             //font color: White
lblDate.setHorizontalAlignment(JLabel.RIGHT);                   //right alignment

SimpleDateFormat ftDay = new SimpleDateFormat("EEEE");          //setting format for day
SimpleDateFormat ftTime = new SimpleDateFormat("hh:mm:ss a");   //setting format for time
SimpleDateFormat ftDate = new SimpleDateFormat("dd MMMM yyyy"); //setting format for date

ActionListener updateClockAction = new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        lblDay.setText(ftDay.format(new Date()));
        lblTime.setText(ftTime.format(new Date()));
        lblDate.setText(ftDate.format(new Date()));
    }
};

Timer time = new Timer(100, updateClockAction);                 //set timer to update time of clock
time.start();                                                   //start the timer
```

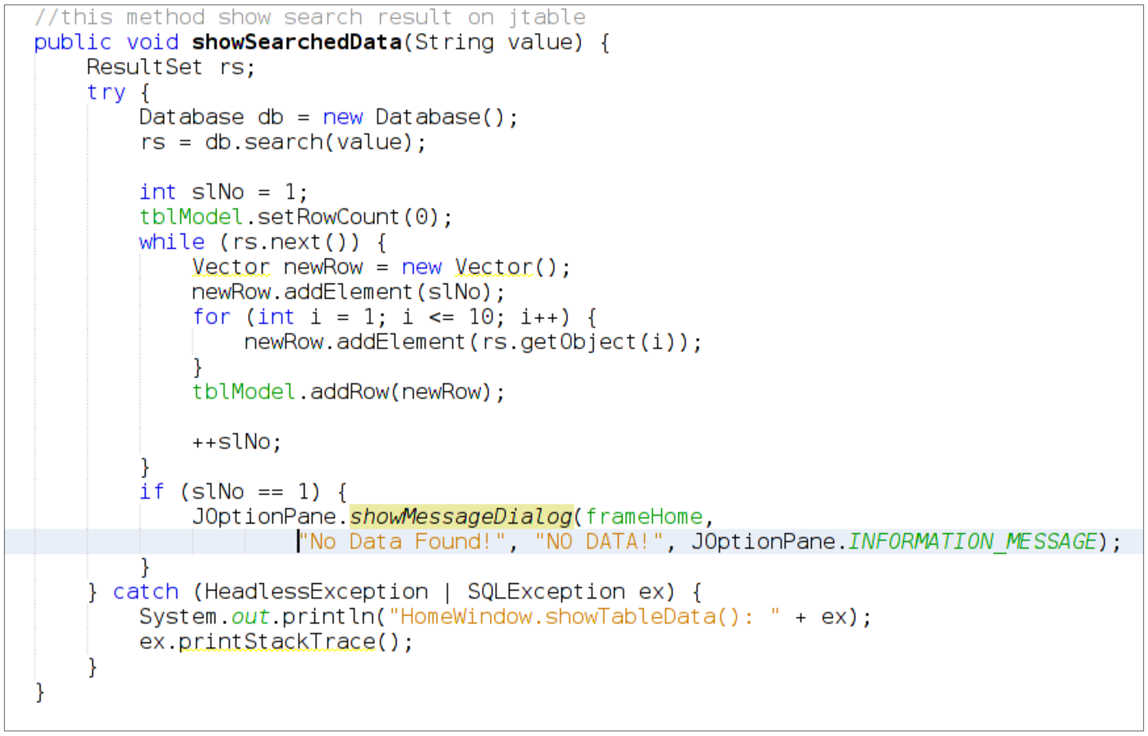
Code 3.1

Code 3.2 is used for action listener of ‘Search’ button and code 3.3 display searched data on table. And code 3.4 method collect data from database for specific query.



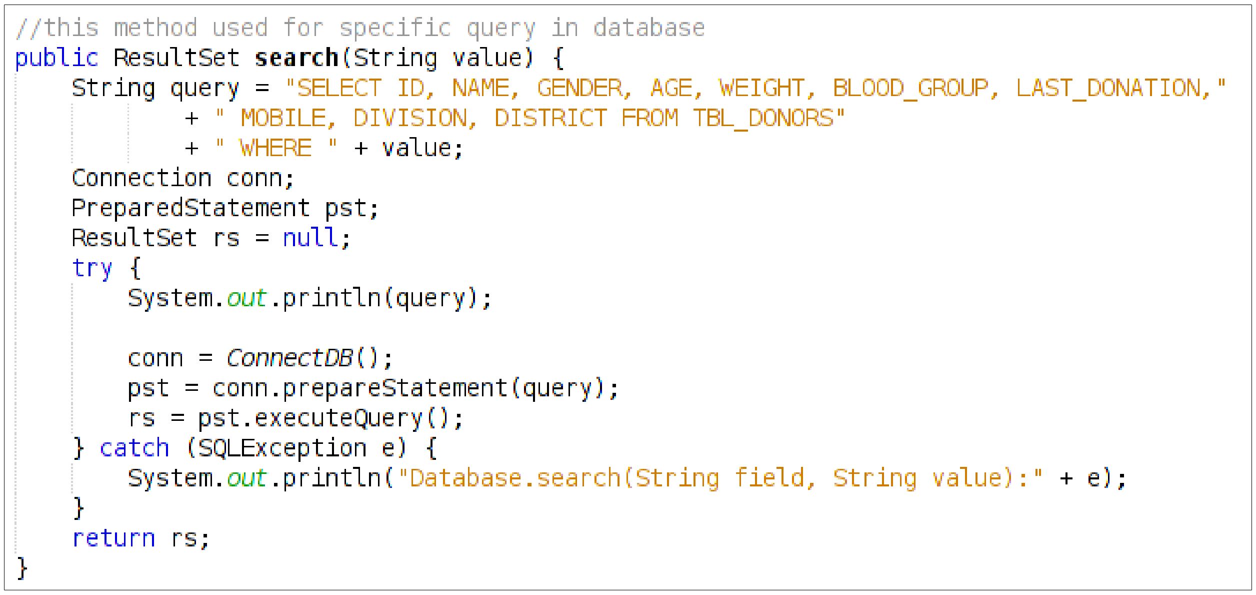
```java
btnSearch.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent ae) {
        int option = cbSearch.getSelectedIndex();
        String value = tfSearch.getText();

        switch (option) {
            case 0:
                value = value.trim();
                value = "BLOOD_GROUP LIKE '%" + value + "%'";
                break;
            case 1:
                value = value.trim();
                value = "NAME LIKE '%" + value + "%'";
                break;
            case 2:
                value = value.trim();
                value = "DIVISION LIKE '%" + value + "%'";
                break;
            case 3:
                value = value.trim();
                value = "DISTRICT LIKE '%" + value + "%'";
                break;
        }
        showSearchedData(value);
    }
});
```

Code 3.2



```java
//this method show search result on jtable
public void showSearchedData(String value) {
    ResultSet rs;
    try {
        Database db = new Database();
        rs = db.search(value);

        int slNo = 1;
        tblModel.setRowCount(0);
        while (rs.next()) {
            Vector newRow = new Vector();
            newRow.addElement(slNo);
            for (int i = 1; i <= 10; i++) {
                newRow.addElement(rs.getObject(i));
            }
            tblModel.addRow(newRow);

            ++slNo;
        }
        if (slNo == 1) {
            JOptionPane.showMessageDialog(frameHome,
                    "No Data Found!", "NO DATA!", JOptionPane.INFORMATION_MESSAGE);
        }
    } catch (HeadlessException | SQLException ex) {
        System.out.println("HomeWindow.showTableData(): " + ex);
        ex.printStackTrace();
    }
}
```

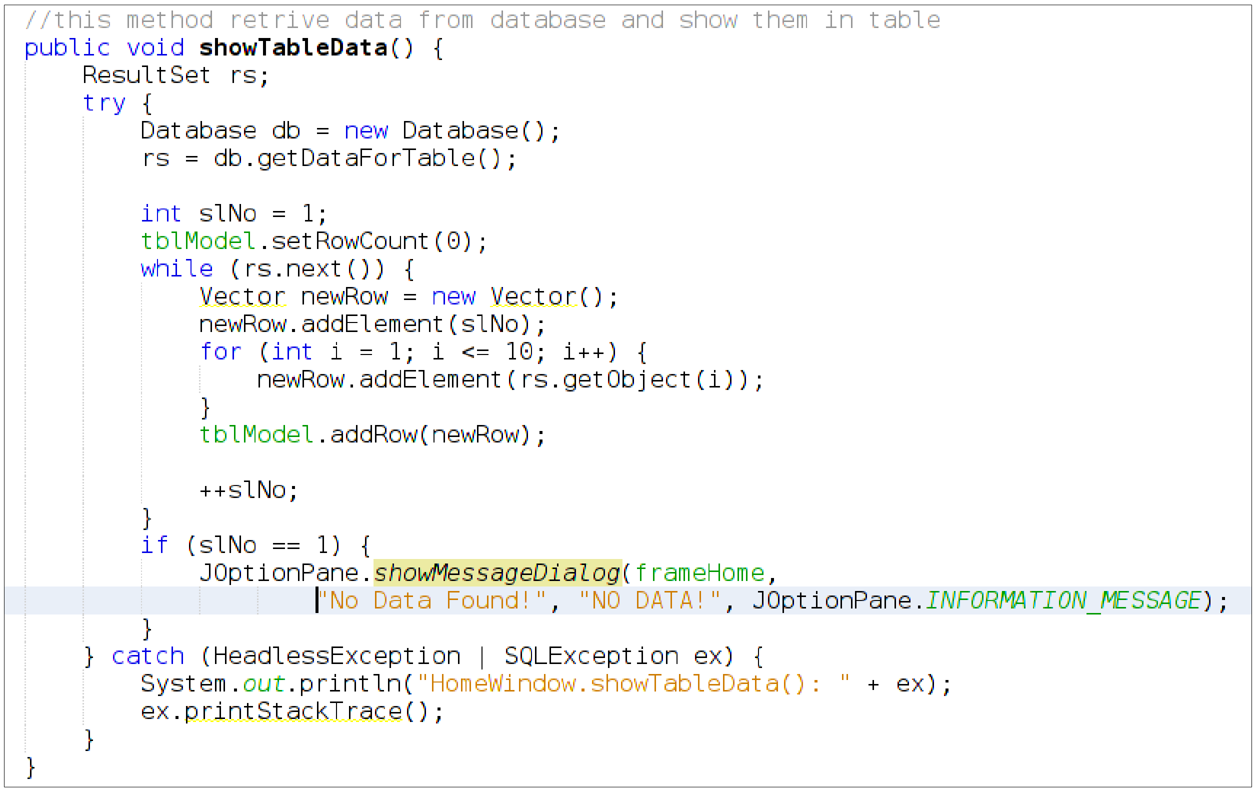
Code 3.3



```java
//this method used for specific query in database
public ResultSet search(String value) {
    String query = "SELECT ID, NAME, GENDER, AGE, WEIGHT, BLOOD_GROUP, LAST_DONATION,"
            + " MOBILE, DIVISION, DISTRICT FROM TBL_DONORS"
            + " WHERE " + value;
    Connection conn;
    PreparedStatement pst;
    ResultSet rs = null;
    try {
        System.out.println(query);

        conn = ConnectDB();
        pst = conn.prepareStatement(query);
        rs = pst.executeQuery();
    } catch (SQLException e) {
        System.out.println("Database.search(String field, String value):" + e);
    }
    return rs;
}
```

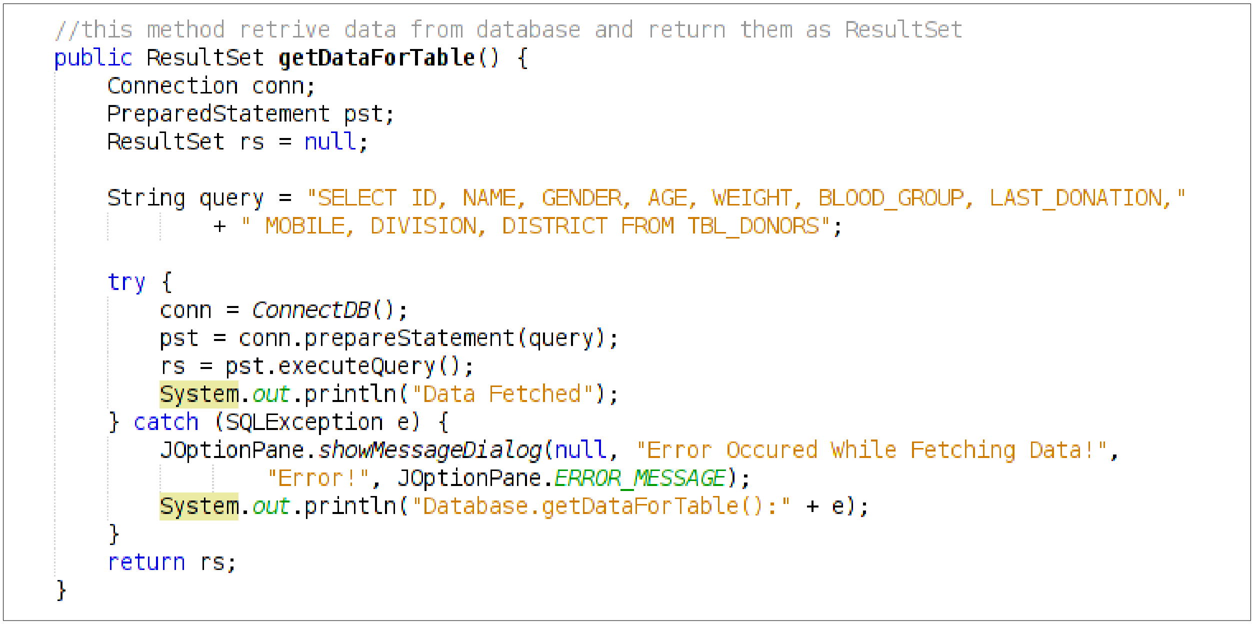
Code 3.4

To display donors list initially on the table code 3.5 is used and code 3.6 is used for fetch data from database.



```java
//this method retrive data from database and show them in table
public void showTableData() {
    ResultSet rs;
    try {
        Database db = new Database();
        rs = db.getDataForTable();

        int slNo = 1;
        tblModel.setRowCount(0);
        while (rs.next()) {
            Vector newRow = new Vector();
            newRow.addElement(slNo);
            for (int i = 1; i <= 10; i++) {
                newRow.addElement(rs.getObject(i));
            }
            tblModel.addRow(newRow);

            ++slNo;
        }
        if (slNo == 1) {
            JOptionPane.showMessageDialog(frameHome,
                    "No Data Found!", "NO DATA!", JOptionPane.INFORMATION_MESSAGE);
        }
    } catch (HeadlessException | SQLException ex) {
        System.out.println("HomeWindow.showTableData(): " + ex);
        ex.printStackTrace();
    }
}
```

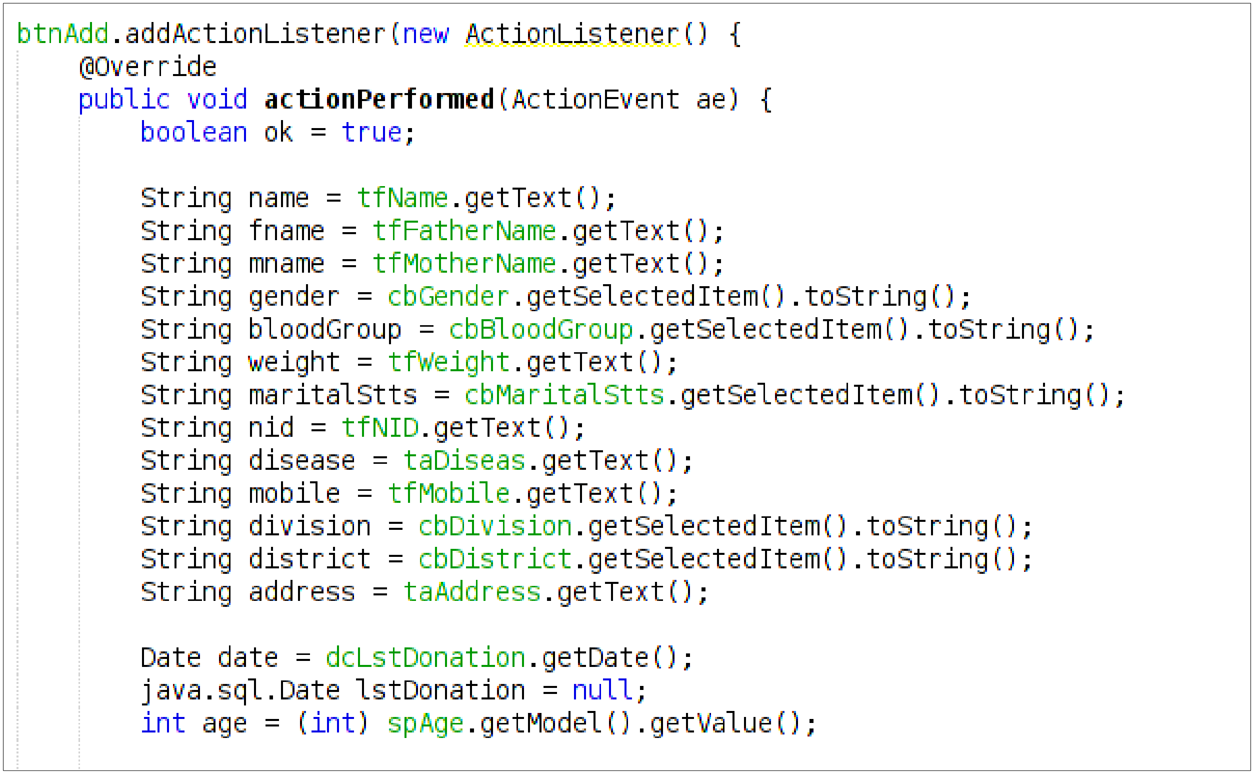
Code 3.5



```java
//this method retrive data from database and return them as ResultSet
public ResultSet getDataForTable() {
    Connection conn;
    PreparedStatement pst;
    ResultSet rs = null;

    String query = "SELECT ID, NAME, GENDER, AGE, WEIGHT, BLOOD_GROUP, LAST_DONATION,"
            + " MOBILE, DIVISION, DISTRICT FROM TBL_DONORS";

    try {
        conn = ConnectDB();
        pst = conn.prepareStatement(query);
        rs = pst.executeQuery();
        System.out.println("Data Fetched");
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, "Error Occured While Fetching Data!",
                "Error!", JOptionPane.ERROR_MESSAGE);
        System.out.println("Database.getDataForTable():" + e);
    }
    return rs;
}
```

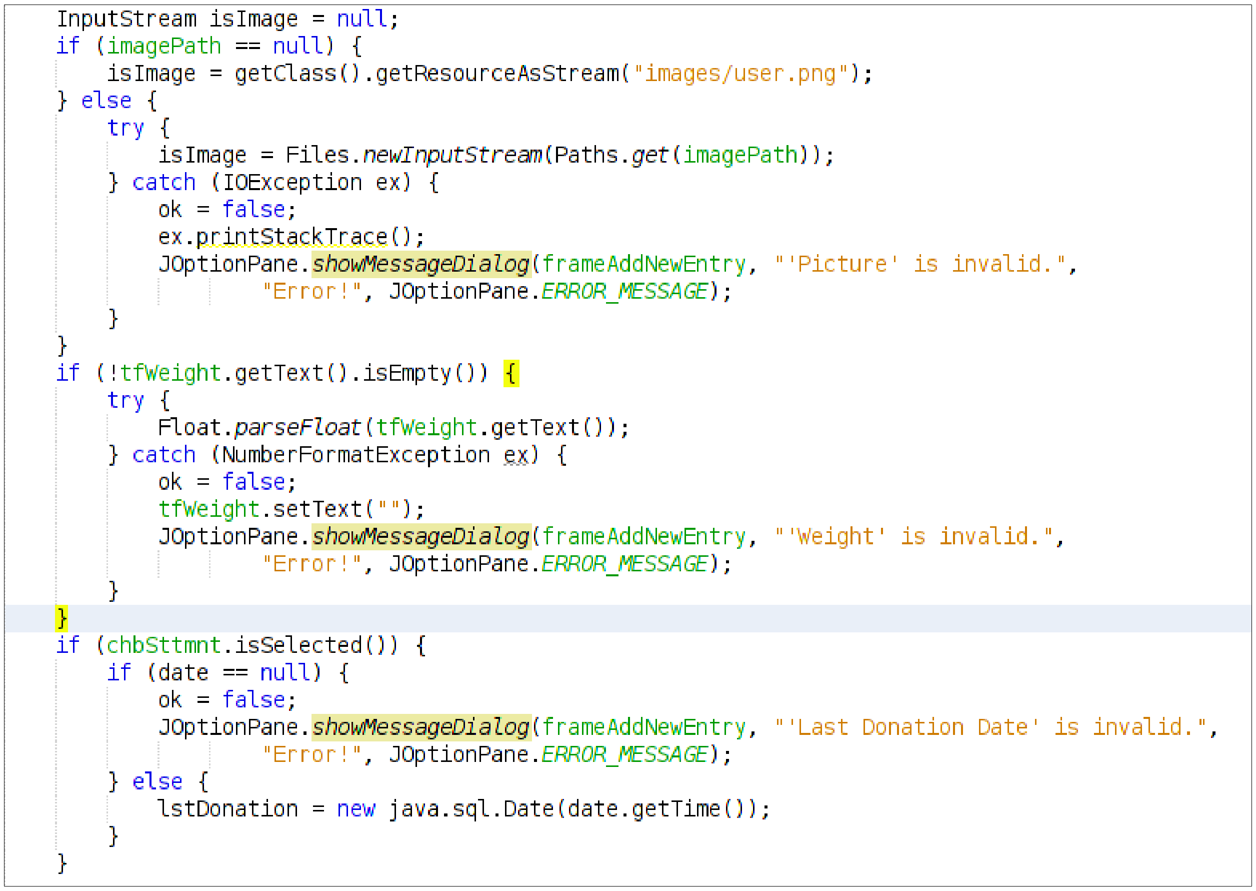
Code 3.6

To add new donor information in database code 3.8.1 and 3.8.2 is used and for validating data code 3.7.1 - 3.7.4 is used.



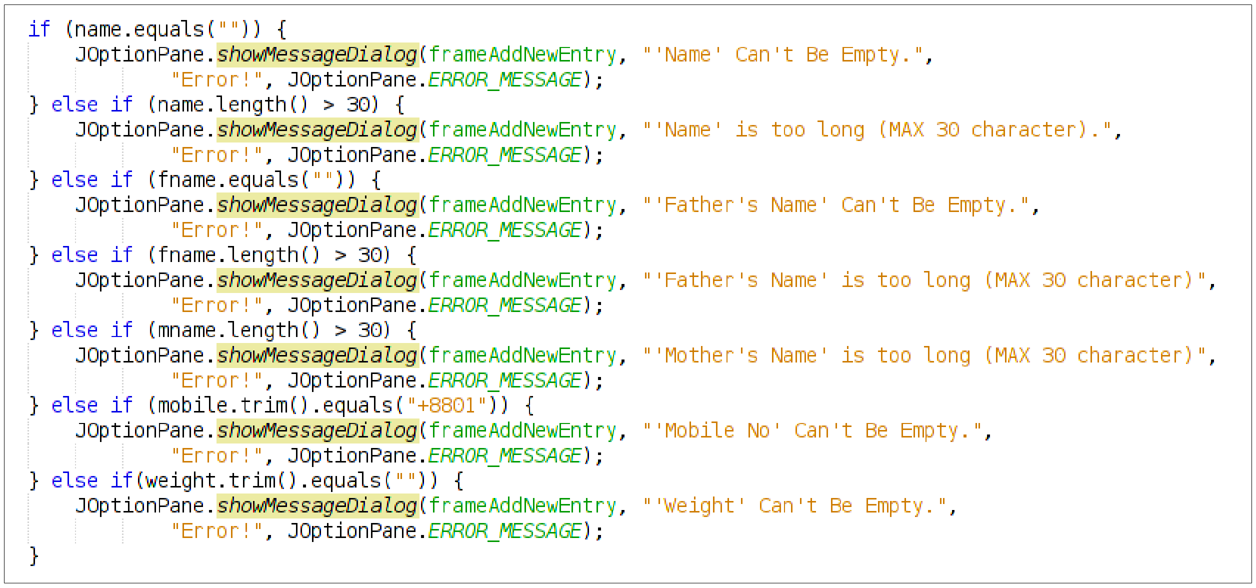
```java
btnAdd.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent ae) {
        boolean ok = true;

        String name = tfName.getText();
        String fname = tfFatherName.getText();
        String mname = tfMotherName.getText();
        String gender = cbGender.getSelectedItem().toString();
        String bloodGroup = cbBloodGroup.getSelectedItem().toString();
        String weight = tfWeight.getText();
        String maritalStts = cbMaritalStts.getSelectedItem().toString();
        String nid = tfNID.getText();
        String disease = taDiseas.getText();
        String mobile = tfMobile.getText();
        String division = cbDivision.getSelectedItem().toString();
        String district = cbDistrict.getSelectedItem().toString();
        String address = taAddress.getText();

        Date date = dcLstDonation.getDate();
        java.sql.Date lstDonation = null;
        int age = (int) spAge.getModel().getValue();
```

Code 3.7.1



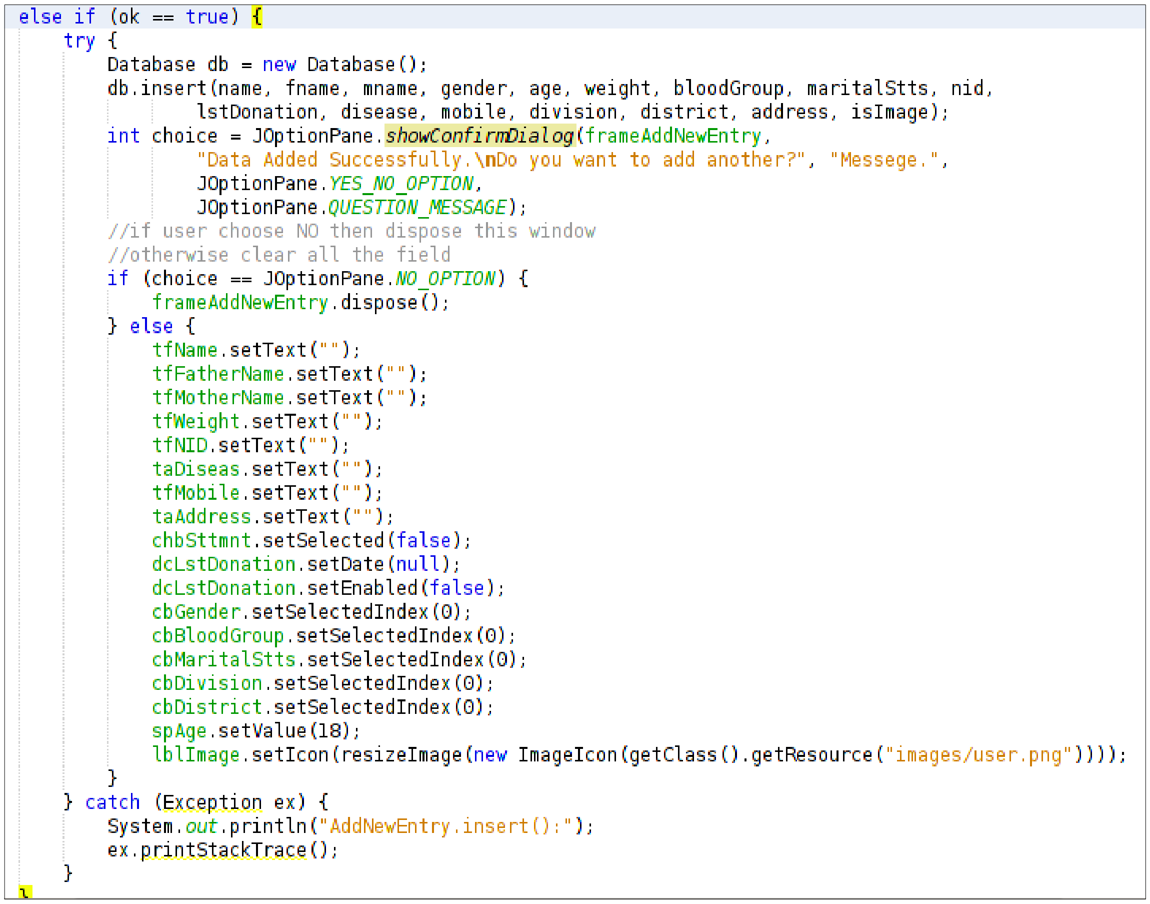
```java
InputStream isImage = null;
if (imagePath == null) {
    isImage = getClass().getResourceAsStream("images/user.png");
} else {
    try {
        isImage = Files.newInputStream(Paths.get(imagePath));
    } catch (IOException ex) {
        ok = false;
        ex.printStackTrace();
        JOptionPane.showMessageDialog(frameAddNewEntry, "'Picture' is invalid.",
                "Error!", JOptionPane.ERROR_MESSAGE);
    }
}
if (!tfWeight.getText().isEmpty()) {
    try {
        Float.parseFloat(tfWeight.getText());
    } catch (NumberFormatException ex) {
        ok = false;
        tfWeight.setText("");
        JOptionPane.showMessageDialog(frameAddNewEntry, "'Weight' is invalid.",
                "Error!", JOptionPane.ERROR_MESSAGE);
    }
}
if (chbSttmnt.isSelected()) {
    if (date == null) {
        ok = false;
        JOptionPane.showMessageDialog(frameAddNewEntry, "'Last Donation Date' is invalid.",
                "Error!", JOptionPane.ERROR_MESSAGE);
    } else {
        lstDonation = new java.sql.Date(date.getTime());
    }
}
```

Code 3.7.2



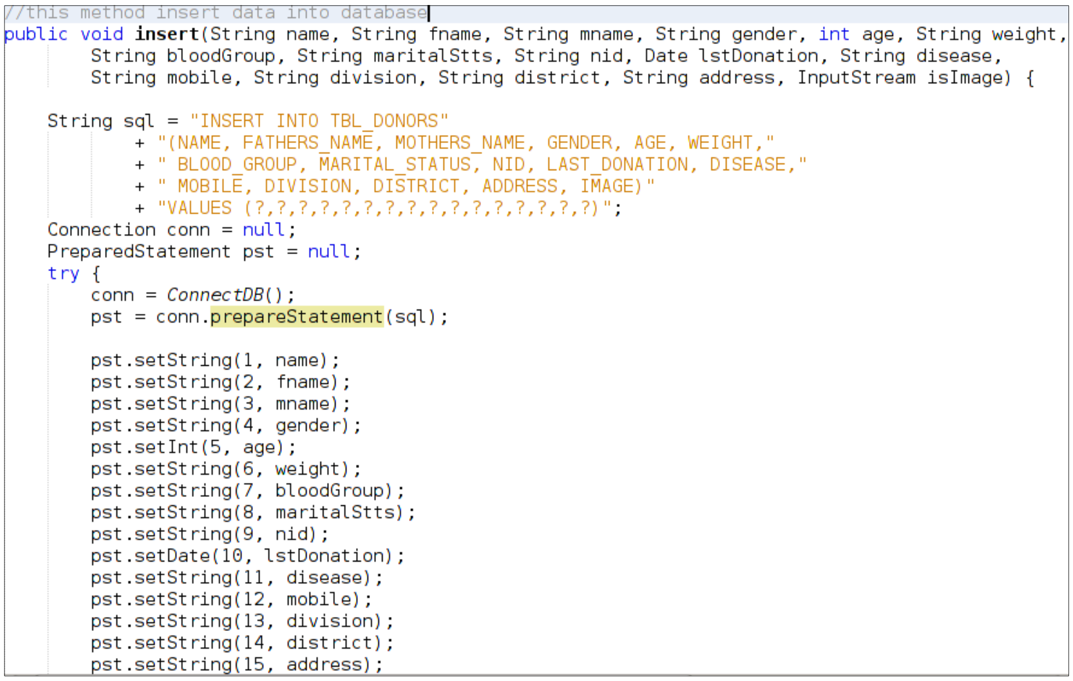
```java
if (name.equals("")) {
    JOptionPane.showMessageDialog(frameAddNewEntry, "'Name' Can't Be Empty.",
            "Error!", JOptionPane.ERROR_MESSAGE);
} else if (name.length() > 30) {
    JOptionPane.showMessageDialog(frameAddNewEntry, "'Name' is too long (MAX 30 character).",
            "Error!", JOptionPane.ERROR_MESSAGE);
} else if (fname.equals("")) {
    JOptionPane.showMessageDialog(frameAddNewEntry, "'Father's Name' Can't Be Empty.",
            "Error!", JOptionPane.ERROR_MESSAGE);
} else if (fname.length() > 30) {
    JOptionPane.showMessageDialog(frameAddNewEntry, "'Father's Name' is too long (MAX 30 character)",
            "Error!", JOptionPane.ERROR_MESSAGE);
} else if (mname.length() > 30) {
    JOptionPane.showMessageDialog(frameAddNewEntry, "'Mother's Name' is too long (MAX 30 character)",
            "Error!", JOptionPane.ERROR_MESSAGE);
} else if (mobile.trim().equals("+8801")) {
    JOptionPane.showMessageDialog(frameAddNewEntry, "'Mobile No' Can't Be Empty.",
            "Error!", JOptionPane.ERROR_MESSAGE);
} else if(weight.trim().equals("")) {
    JOptionPane.showMessageDialog(frameAddNewEntry, "'Weight' Can't Be Empty.",
            "Error!", JOptionPane.ERROR_MESSAGE);
}
```

Code 3.7.3



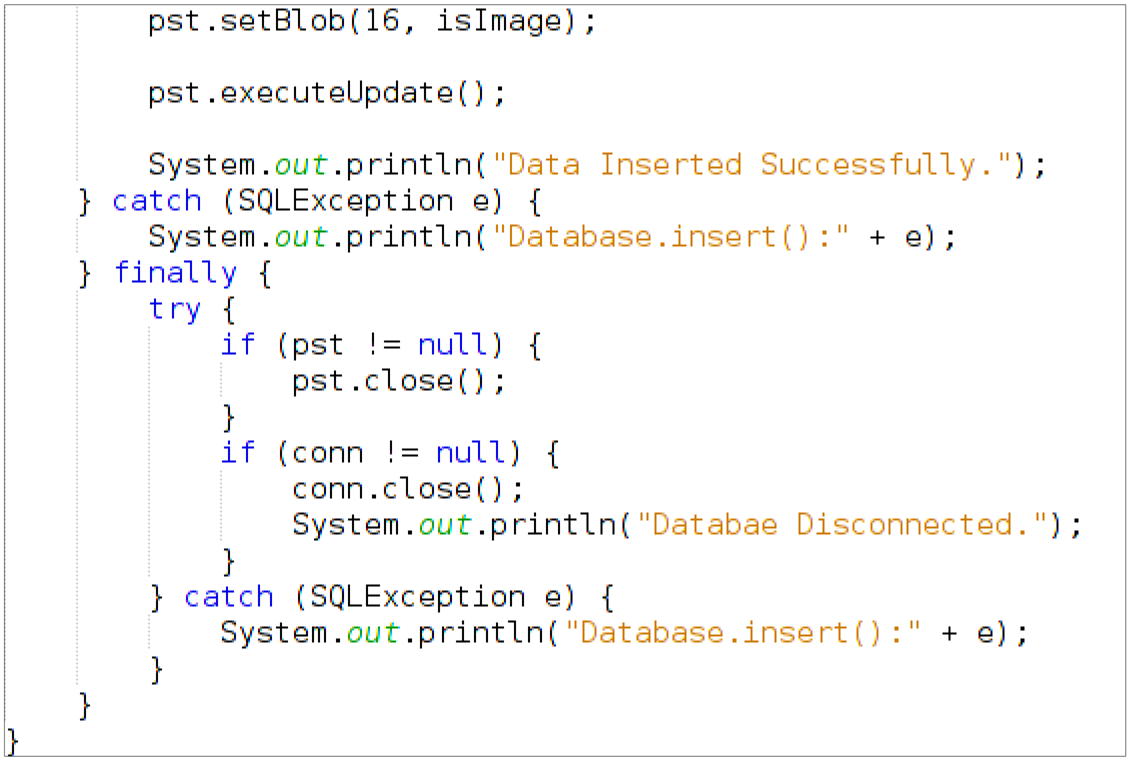
```java
else if (ok == true) {
    try {
        Database db = new Database();
        db.insert(name, fname, mname, gender, age, weight, bloodGroup, maritalStts, nid,
                lstDonation, disease, mobile, division, district, address, isImage);
        int choice = JOptionPane.showConfirmDialog(frameAddNewEntry,
                "Data Added Successfully.\nDo you want to add another?", "Messege.",
                JOptionPane.YES_NO_OPTION,
                JOptionPane.QUESTION_MESSAGE);
        //if user choose NO then dispose this window
        //otherwise clear all the field
        if (choice == JOptionPane.NO_OPTION) {
            frameAddNewEntry.dispose();
        } else {
            tfName.setText("");
            tfFatherName.setText("");
            tfMotherName.setText("");
            tfWeight.setText("");
            tfNID.setText("");
            taDiseas.setText("");
            tfMobile.setText("");
            taAddress.setText("");
            chbSttmnt.setSelected(false);
            dcLstDonation.setDate(null);
            dcLstDonation.setEnabled(false);
            cbGender.setSelectedIndex(0);
            cbBloodGroup.setSelectedIndex(0);
            cbMaritalStts.setSelectedIndex(0);
            cbDivision.setSelectedIndex(0);
            cbDistrict.setSelectedIndex(0);
            spAge.setValue(18);
            lblImage.setIcon(resizeImage(new ImageIcon(getClass().getResource("images/user.png"))));
        }
    } catch (Exception ex) {
        System.out.println("AddNewEntry.insert():");
        ex.printStackTrace();
    }
}
```

Code 3.7.4



```java
//this method insert data into database
public void insert(String name, String fname, String mname, String gender, int age, String weight,
        String bloodGroup, String maritalStts, String nid, Date lstDonation, String disease,
        String mobile, String division, String district, String address, InputStream isImage) {

    String sql = "INSERT INTO TBL_DONORS"
            + "(NAME, FATHERS_NAME, MOTHERS_NAME, GENDER, AGE, WEIGHT,"
            + " BLOOD_GROUP, MARITAL_STATUS, NID, LAST_DONATION, DISEASE,"
            + " MOBILE, DIVISION, DISTRICT, ADDRESS, IMAGE)"
            + "VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)";
    Connection conn = null;
    PreparedStatement pst = null;
    try {
        conn = ConnectDB();
        pst = conn.prepareStatement(sql);

        pst.setString(1, name);
        pst.setString(2, fname);
        pst.setString(3, mname);
        pst.setString(4, gender);
        pst.setInt(5, age);
        pst.setString(6, weight);
        pst.setString(7, bloodGroup);
        pst.setString(8, maritalStts);
        pst.setString(9, nid);
        pst.setDate(10, lstDonation);
        pst.setString(11, disease);
        pst.setString(12, mobile);
        pst.setString(13, division);
        pst.setString(14, district);
        pst.setString(15, address);
```

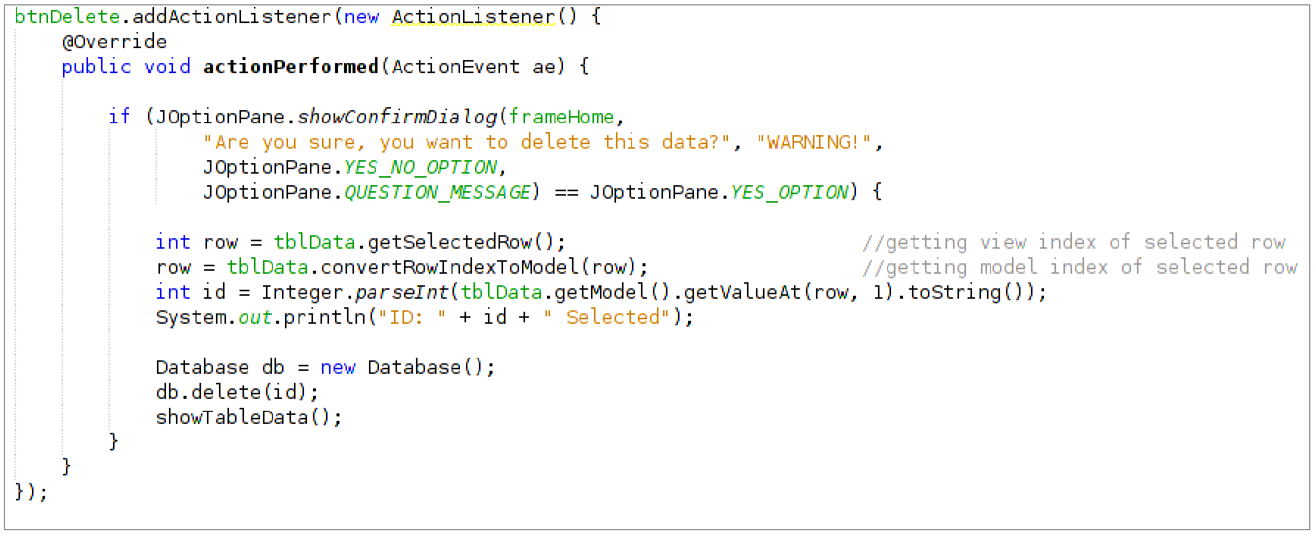
Code 3.8.1



```java
            pst.setBlob(16, isImage);

            pst.executeUpdate();

            System.out.println("Data Inserted Successfully.");
        } catch (SQLException e) {
            System.out.println("Database.insert():" + e);
        } finally {
            try {
                if (pst != null) {
                    pst.close();
                }
                if (conn != null) {
                    conn.close();
                    System.out.println("Databae Disconnected.");
                }
            } catch (SQLException e) {
                System.out.println("Database.insert():" + e);
            }
        }
}
```

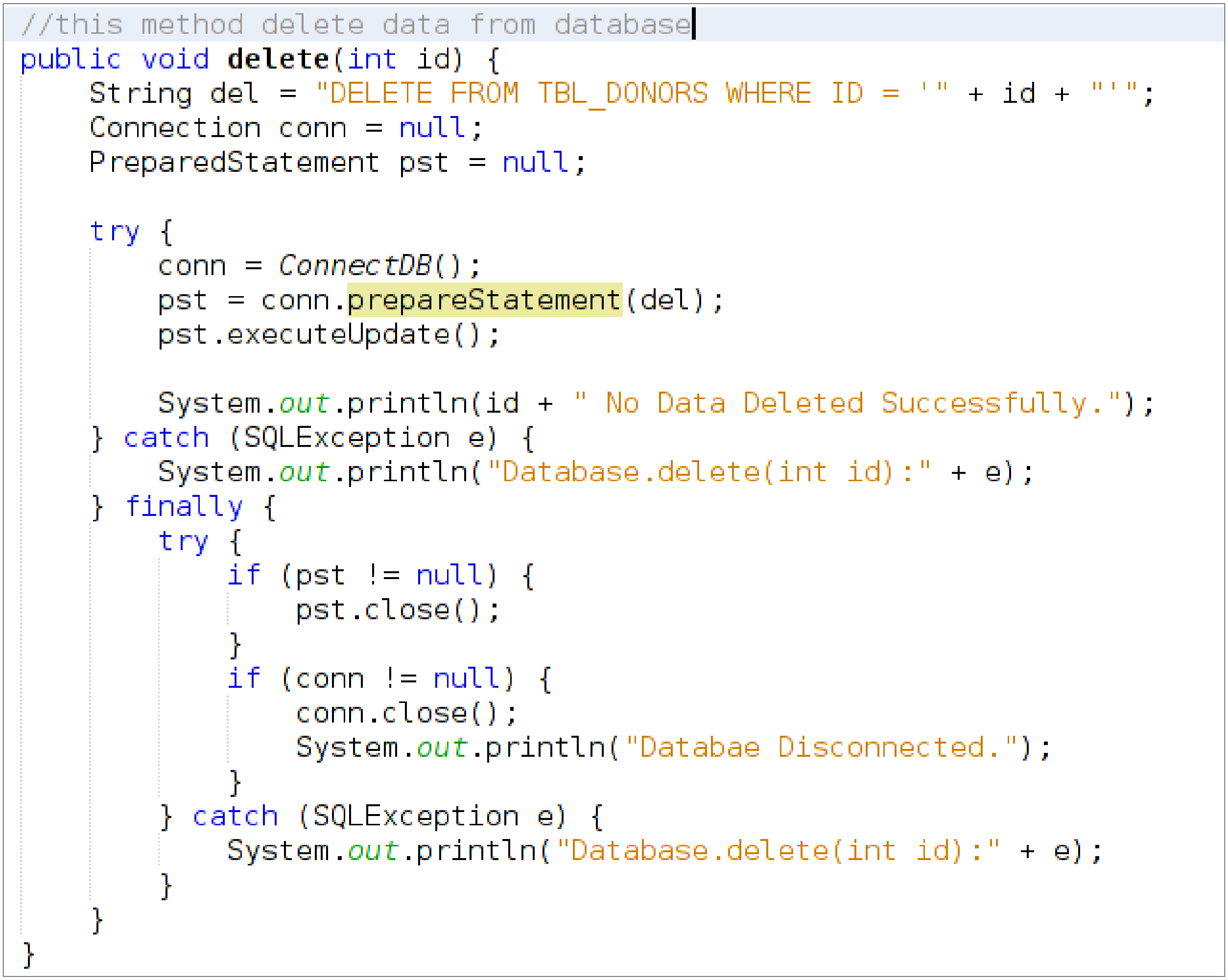
Code 3.8.2

To delete a donor from database code 3.9 is used for action listener of ‘Delete’ button and code 3.10 is used for delete data from database.



```java
btnDelete.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent ae) {

        if (JOptionPane.showConfirmDialog(frameHome,
                "Are you sure, you want to delete this data?", "WARNING!",
                JOptionPane.YES_NO_OPTION,
                JOptionPane.QUESTION_MESSAGE) == JOptionPane.YES_OPTION) {

            int row = tblData.getSelectedRow();                      //getting view index of selected row
            row = tblData.convertRowIndexToModel(row);               //getting model index of selected row
            int id = Integer.parseInt(tblData.getModel().getValueAt(row, 1).toString());
            System.out.println("ID: " + id + " Selected");

            Database db = new Database();
            db.delete(id);
            showTableData();
        }
    }
});
```

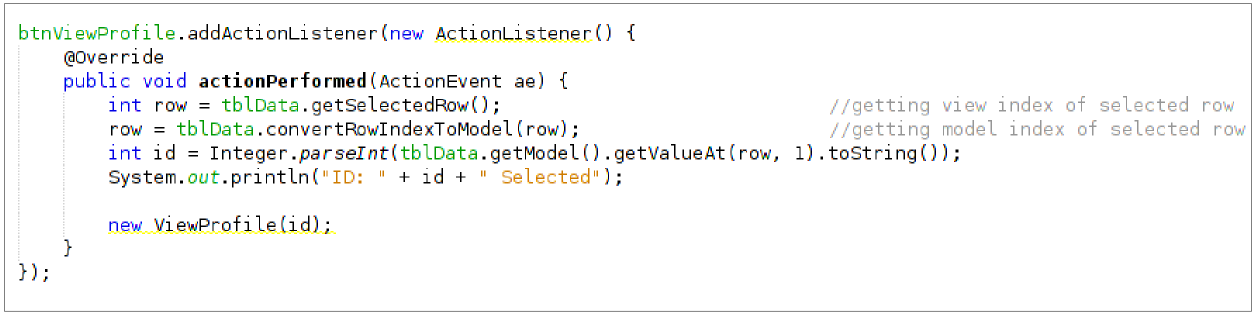
Code 3.9



```
//this method delete data from database
public void delete(int id) {
    String del = "DELETE FROM TBL_DONORS WHERE ID = '" + id + "'";
    Connection conn = null;
    PreparedStatement pst = null;

    try {
        conn = ConnectDB();
        pst = conn.prepareStatement(del);
        pst.executeUpdate();

        System.out.println(id + " No Data Deleted Successfully.");
    } catch (SQLException e) {
        System.out.println("Database.delete(int id):" + e);
    } finally {
        try {
            if (pst != null) {
                pst.close();
            }
            if (conn != null) {
                conn.close();
                System.out.println("Databae Disconnected.");
            }
        } catch (SQLException e) {
            System.out.println("Database.delete(int id):" + e);
        }
    }
}
```

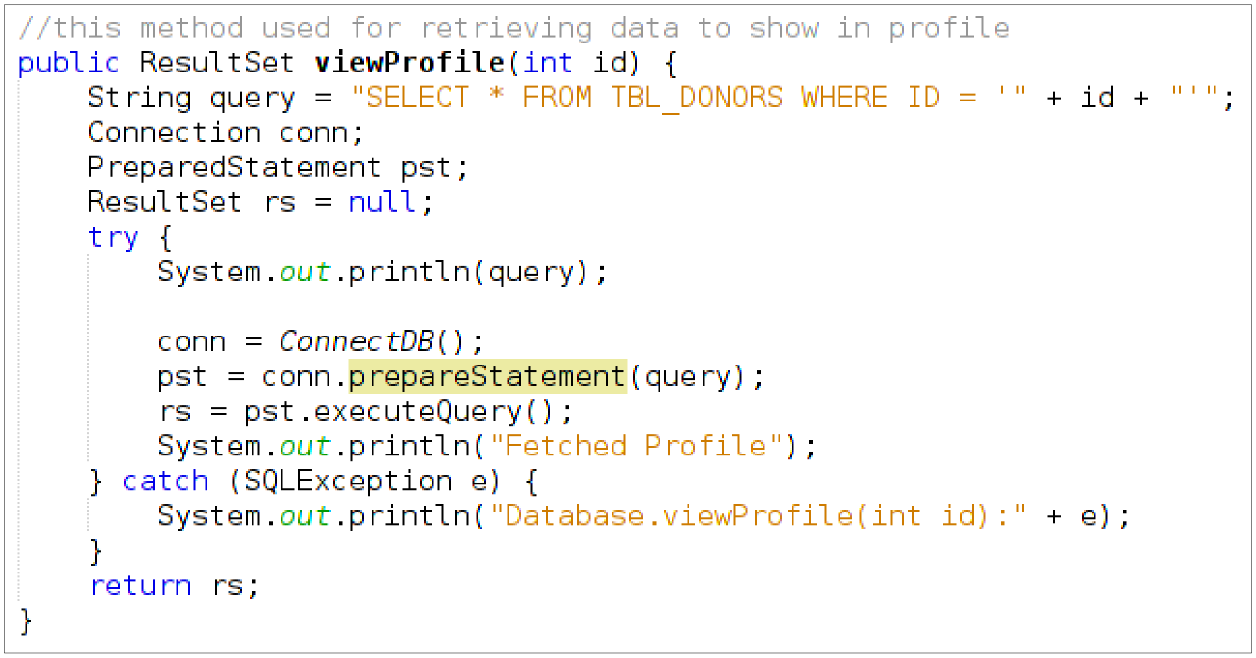
Code 3.10

To view profile code 3.11 is used as button listener of 'View Profile' button and code 3.12 is used for fetch data from database of a specific donor.



```
btnViewProfile.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent ae) {
        int row = tblData.getSelectedRow();                       //getting view index of selected row
        row = tblData.convertRowIndexToModel(row);                //getting model index of selected row
        int id = Integer.parseInt(tblData.getModel().getValueAt(row, 1).toString());
        System.out.println("ID: " + id + " Selected");

        new ViewProfile(id);
    }
});
```

Code 3.11



```java
//this method used for retrieving data to show in profile
public ResultSet viewProfile(int id) {
    String query = "SELECT * FROM TBL_DONORS WHERE ID = '" + id + "'";
    Connection conn;
    PreparedStatement pst;
    ResultSet rs = null;
    try {
        System.out.println(query);

        conn = ConnectDB();
        pst = conn.prepareStatement(query);
        rs = pst.executeQuery();
        System.out.println("Fetched Profile");
    } catch (SQLException e) {
        System.out.println("Database.viewProfile(int id):" + e);
    }
    return rs;
}
```

Code 3.12

Chapter 4

Conclusion

The main reason of this ‘Blood Donors Management’ system is to provide a quick and easy way to enhance the efficiency of database supervision and keep up donors details. Reducing search delay.

4.1 Challenges

- Storing and retrieving image to/from database
- Keep the image while editing donor’s profile if it is unchanged
- Showing digital clock on the home window(Sync of labels)
- Refresh table after adding new donor

4.2 Future work

We will bring android version of this application very soon, where user can directly communicate with donor. And anyone can send request to admin to be a blood donor.

References

1. https://docs.oracle.com
2. www.javatpoint.com
3. www.java2s.com
4. www.javacodex.com
5. https://dev.mysql.com/doc/