Chapter 6:

Database Connectivity

Informatics Practices

Class XII (CBSE Board)

Revised as per
CBSE
Curriculum
2015

"Open Teaching-Learning Material"

Visit www.ip4you.blogspot.com for more….

Authored By:- Rajesh Kumar Mishra, PGT (Comp.Sc.)
Kendriya Vidyalaya Upper Camp, Dehradun (Uttarakhand)
e-mail : rkmalld@gmail.com

Introduction

A real life application needs to manipulate data stored in a Database.

A database is a collection of related data in the form of Tables. Most of the database uses SQL (Structured Query Language) to Insert, Delete, Update or retrieve stored records.

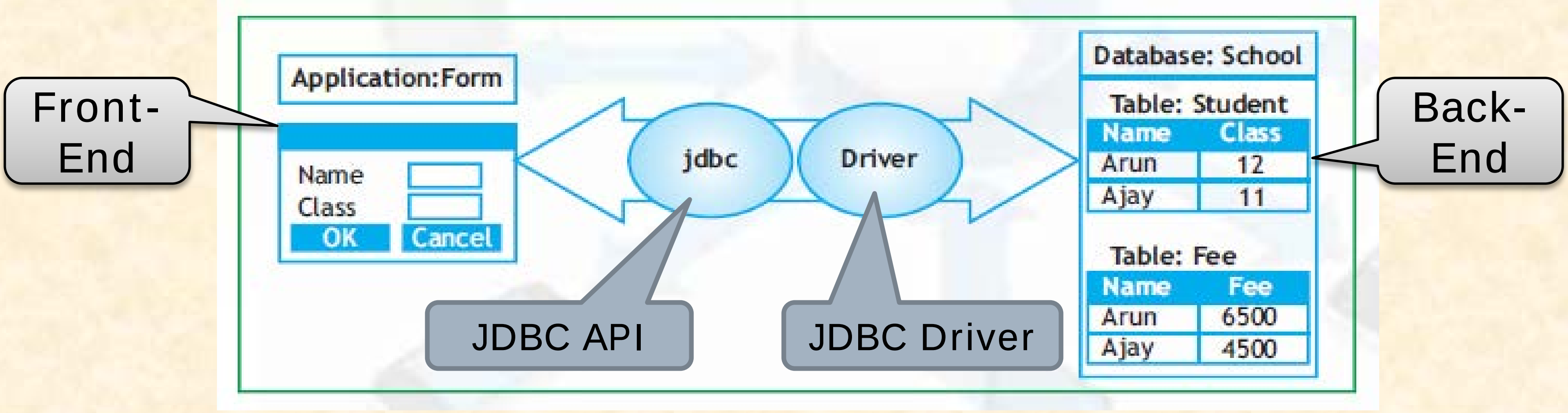
In order to connect a Java application (Front-End) to a Database (Back-End) designed in MySQL, Oracle, Sybase, MS SQL Server etc, you need a Interface Driver Program.

Java Provides JDBC API (Java Database Connection - Application Program Interface) and JDBC Driver for MySQL to connect a MySQL database.

What is JDBC ?

JDBC is JAVA’s Database connection driver interface which performs the following task for the application.

- ❑ Establish a connection with a Database.
- ❑ Send SQL request (Query) to a Database Server.
- ❑ Returns Result obtained against Query.



Database: School

Table: Student

Name	Class
Arun	12
Ajay	11

Table: Fee

Name	Fee
Arun	6500
Ajay	4500

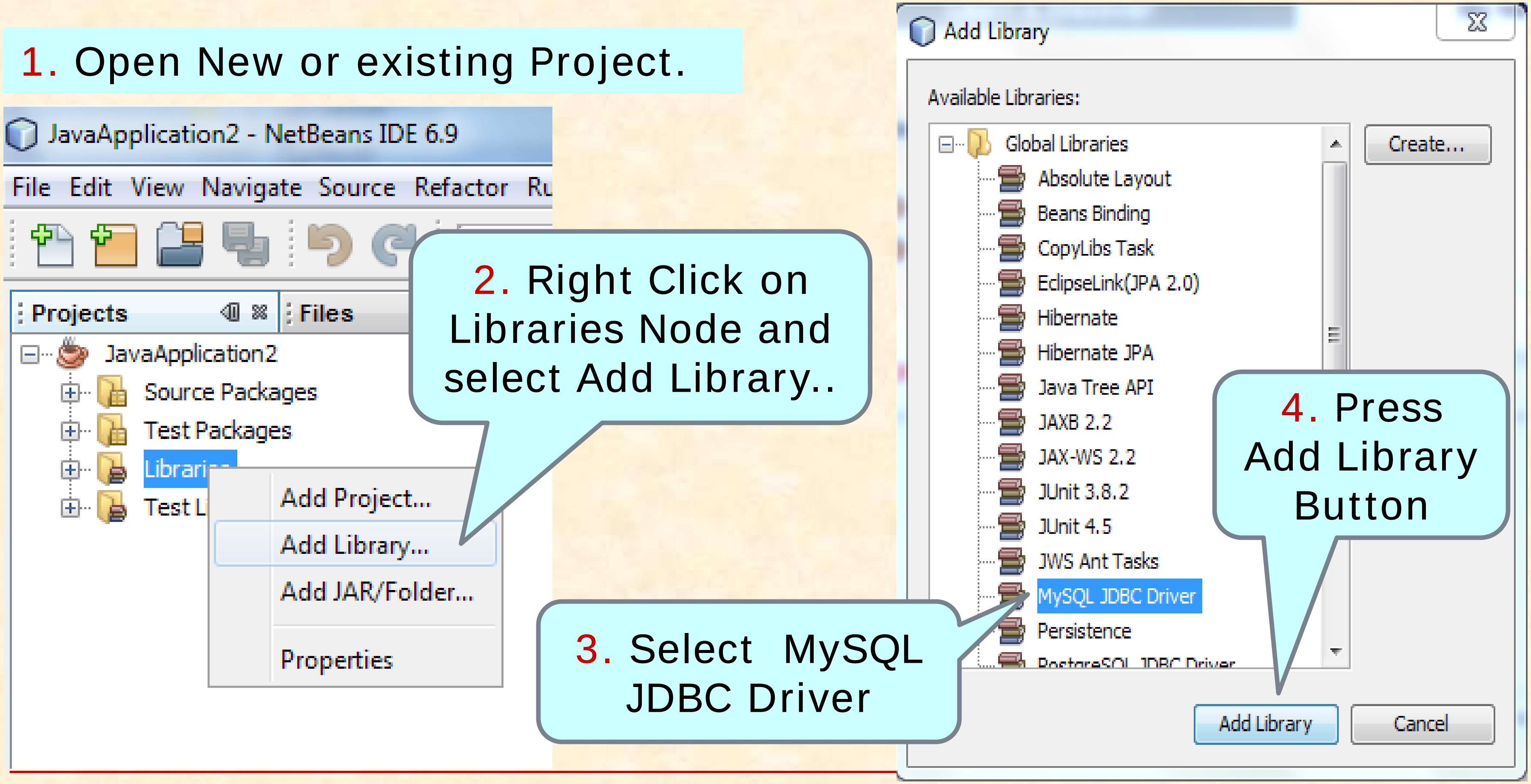
Communication with a Database using JDBC API & Driver

Adding MySQL JDBC Driver in NetBeans IDE

The Prerequisite for connecting a Java application to MySQL is adding MySQL JDBC driver in the Project/Program.

The NetBeans IDE comes with pre-bundled MySQL JDBC Driver. You may add JDBC Driver in the Database Connectivity Project as follows-

1. Open New or existing Project.



2. Right Click on Libraries Node and select Add Library..

3. Select MySQL JDBC Driver

4. Press Add Library Button

Classes used for Database Connectivity

The Core element of JDBC is JDBC API, which consists of a set of Java classes equipped with predefined methods to handle various data access functions such as Selecting appropriate database driver, establishing connection, submitting SQL query and processing results.

JDBC API offers four main classes, which are-

- ❑ **Driver Manager Class:** It loads the JDBC driver to locate, logs and access a database.
- ❑ **Connection Class:** It manages communication between Java Client Application and Database, through SQL statements.
- ❑ **Statement Class:** It contains SQL commands which is submitted to the Database Server and returns ResultSet object containing the result of SQL statement.
- ❑ **Result Set Class:** It provides predefined mehods to access and convert data values returned by the executed SQL statement.

A JDBC driver must be registered with JDBC Driver Manage using Class.forName() method before establishing a connection.

Connecting MySQL from JAVA Application

After installing JDBC Driver, you may access MySQL database through JAVA Application.

The Following Six steps may be followed to establish a connection with MySQL database.

- ❑ **Step 1:** Import Required package/classes in the application.
- ❑ **Step 2:** Register the JDBC Driver to JDBC Driver Manager.
- ❑ **Step 3:** Open a Connection.
- ❑ **Step 4:** Execute a Query.
- ❑ **Step 5:** Extract data from Result set
- ❑ **Step 6:** Close Connection.

Working with Data Connectivity Project

❑ Step 1: Importing Required package/classes

To Import Java.sql Library package in the Application you need to give following import statements.

```
import  java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

import java.sql.ResultSet;
```

Or

```
import java.sql.*;
```

❑ Step 2: Registering the JDBC Driver

To open a Communication channel, you require to initialize driver by registering the JDBC driver with JDBC Driver Manager using **Class.forName()** method of *java.lang* package.

```
Class.forName(“java.sql.DriverManager”);
```

Working with Data Connectivity Project

Step 3: Opening a Connection

DriverManager.getConnection() method is used to create a connection object that represents a physical connection with database. It requires the complete address/path of the database (**Database URL**), **user name** and **password** as a parameter. A database URL can be formed as- **jdbc:mysql :// localhost/ <database name>**

Suppose school is a database designed in MySQL, then Database URL will be as follows-

"jdbc:mysql://localhost/school"

You can assign this string on a variable, which can be used later with DriverManager.getConnection() method.

```
String DB_URL = "jdbc:mysql://localhost/school";
Connection con = DriverManager.getConnection(DB_URL,"root", "abc")
```

Working with Data Connectivity Project

Step 4: Executing a Query

You must create a Statement object for building and submitting a SQL query, using CreateStatement() method of Connection object created in *Step 3*.

```
Statement stmt = con.createStatement();
```

To execute a query executeQuery() method along with a valid SQL statement is used, which returns the records from the database (Result Set) on ResultSet type object.

```
ResultSet rs = stmt.executeQuery("<SQL Query>");
```

```
Statement stmt = con.createStatement();
ResultSet  rs = stmt.executeQuery("select roll,name,class  from  student");
```

➢**Result Set** refers to a logical set of records from the database.

➢An executeUpdate() method is used in place of executeQuery() for Insert, Delete or Update SQL command.

Working with Data Connectivity Project

Step 5: Extracting Data from ResultSet object

To retrieve the data from the ResultSet object, which contains records, You may use the following method.

<ResultSet object>.get<type>(<column name/number>);

Where <type> may be **Int, Long, String, Float** etc. depending on the type of column the table.

Generally, the data values are assigned on the variables and later used in the TextField controls of the Form using setText() method.

```
int r= rs.getInt(“roll”);
String n= rs.getString(“name”);
int c= rs.getInt(“class”);
```

```
int r= rs.getInt(1);
String n= rs.getString(2);
int c= rs.getInt(3);
```

You can use Column number instead of column name of the table

The variable can be used to display the values in the Text boxes like this-

jTextField1.setText(“”+r);

Working with Data Connectivity Project

Since a ResultSet object may contain more than one records (when SQL query may return multiple records) , so a loop is required to process all the records. A while… loop is generally used to read all records.

To break a loop **<ResultSet object>.next()** method is used, which returns false when all the records have been read from the Result set.

```
int r,c ;
String n;
while (rs.next())
{ r= rs.getInt(“roll”);
  n= rs.getString(“name”);
  c= rs.getInt(“class”);
  // statements to display variables on Multi-line display controls //
   ……………………………………………..
}
```

You can use jTextArea or jTable swing controls to display multiple records instead of jTextField.

Working with Data Connectivity Project

Step 6: Closing connection

After all the processing , the final step is to close the environment by closing ResultSet, Statement and Connection objects using close() method.

```
rs.close();
stmt.close();
con.close();
```

To handle errors during establishing connection all the required statements are kept in a try{…} catch (){…} block like this–

```
          try{.......................
                <Data connectivity statements……..>
             }
          catch (  Exception <variable>)
             {
                   <error statement>;
             }
```

A Sample Code for Database Connectivity

```
import  java.sql.*;                                           // 1. import package at the top//
/* The following code may be placed in ActionPerformed event of a button*/
    String db="jdbc:mysql://loacalhost/school");            // Database URL
    String qr= "select  roll, name, class from student"; // Query
try{
    Class.forName("java.sql.DriverManager");                     //2. Register Driver
    Connection con=Driver.getConnection(db, "root", "xyz");  //3.Open Connection
    Statement stmt=con.createStatement();                       // 4. Execute Query
    ResultSet rs= stmt.executeQuery( qr);
    int r, c;
    String n;
    while (rs.next())                                           // 5. Extract Data//
          { r= rs.getInt("roll");
            n= rs.getString("name");
            c= rs.getInt("class");
          ......................................; // Code to manipulate data//
          }
     rs.close();                                               //6.Close Environment//
     stmt.close();
     con.close();
   }
catch (Exception e)
             { JOptionPane.showMessageDialog(null, e.getMessage());  }
```

Commonly used ResultSet Methods

A Result set object maintains a **cursor**, which points to its current row of data. When it is created, cursor is positioned before the first row. You can move the cursor using the following methods.

Method	Purpose
next ()	Moves the cursor forward one row. It returns false when cursor is positioned after the last record.
previous()	Moves cursor to previous record from current position. It returns false when cursor is positioned before the first record.
first()	Moves cursor to first record. It returns true if it positioned at first record otherwise returns false.
last()	Moves cursor to last record. It returns true if it positioned at last record otherwise returns false.
relative(n)	Moves cursor relative to its current position i.e if it is on 2nd row, then relative(3) places cursor at 5th record.
absolute(n)	Moves cursor at nth record of result set irrespective to its current position.
getRow()	Returns the current row number where cursor is positioned.

Example 1:
Search & Display Record using Text Fields

Objective :

Consider the following design of a database application to Search and display a record as per given Mobile number from the Teacher table containing Name, Subject and Mobile Number column.

Assumption

Database : School

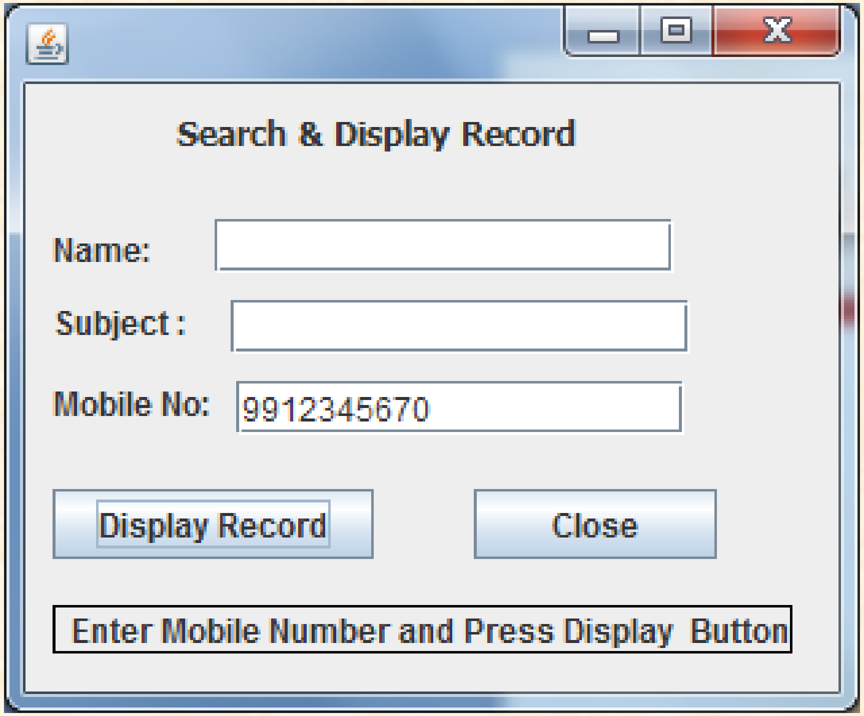
Table : Teacher

Column/Field & Type

- Name Character (40)
- Subject Varchar(30)
- Mobile Char(12)

With some records.

MySQL
User Name: root
Password: kvuc



Search & Display Record

Name:

Subject :

Mobile No: 9912345670

Enter Mobile Number and Press Display Button

Example 1:
Design of the Table

It is assumed that a database and table is designed in MySQL and some records are present. However, if database and tables are not available then follow the following steps for creating database & tables in MySQL.

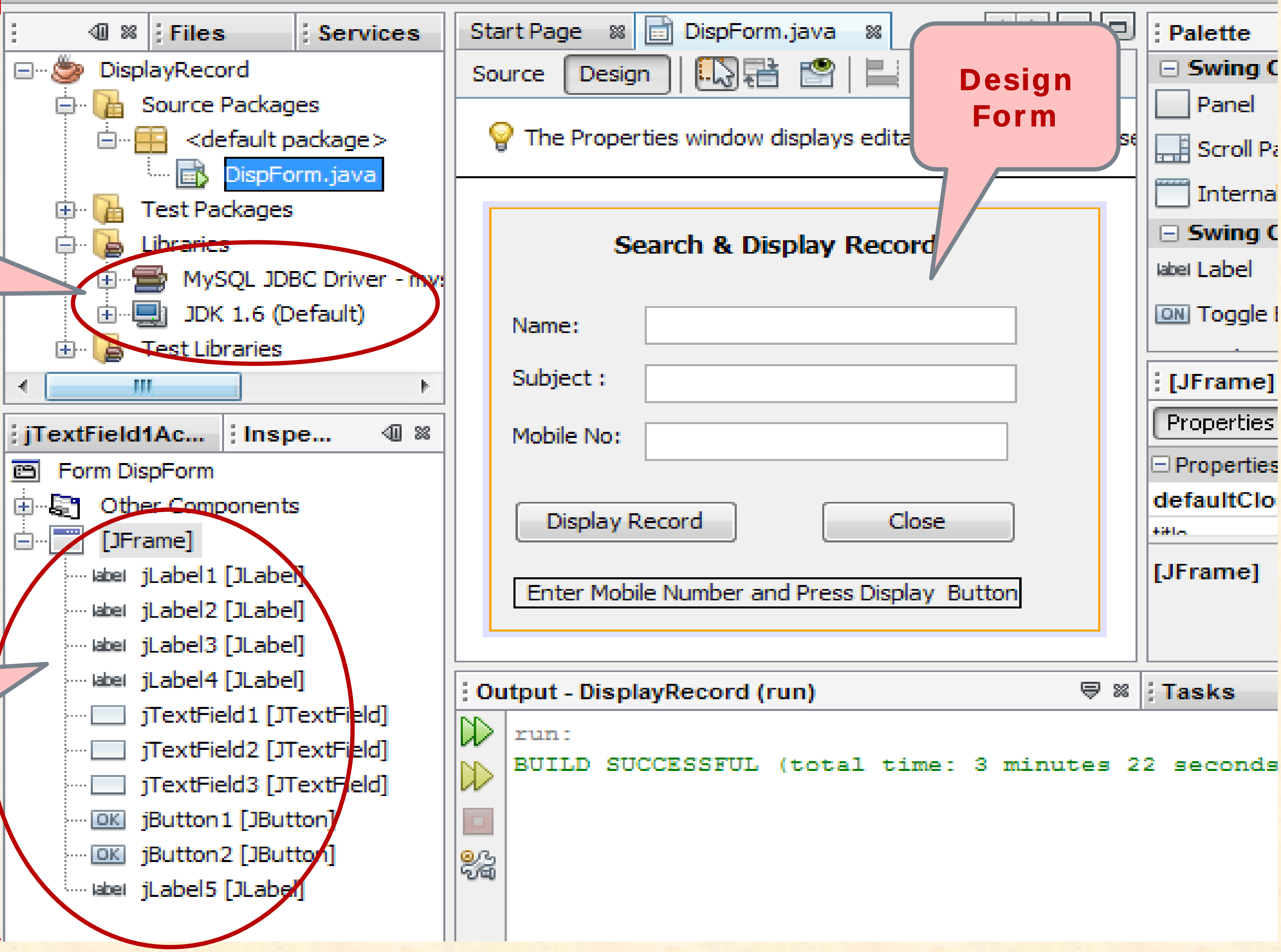
Step 1: Open MySQL and give password to login.

Step 2: Type the following MySQL commands.

```
mysql>  create database school;
mysql>  use school;
mysql>  create table teacher
        ->(name char(40), subject varchar(30), mobile char(12));
mysql>  insert into teacher values (‘Ramesh’, ’Biology’, ’9998123444’);
mysql>  insert into teacher values (‘Ajay’, ’Physics’, ’9899123322’);
mysql>  insert into teacher values (‘Naveen’, ’Maths’, ’9412335454’);
```

Kindly note that Mobile Number of teachers should be different to facilitate unique search/match of the record.

Example 1:
Design of Application in NetBeans



The Properties window displays edita

Search & Display Record

Name:

Subject :

Mobile No:

Display Record | Close

Enter Mobile Number and Press Display Button

Output - DisplayRecord (run)

```
run:
BUILD SUCCESSFUL (total time: 3 minutes 22 seconds
```

Design Form

Add MySQL JDBC Driver

Add Swing Controls

Example 1:
Coding of Event in NetBeans

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
 // TODO code for Display Record Button:
      String DB="jdbc:mysql://localhost/school";      //Database URL
      String name, sub, mob;
      mob=jTextField3.getText();
      String qr= "select name, subject, mobile from teacher where mobile='"+mob+"';";
try{
      Class.forName("java.sql.DriverManager");
      Connection con= DriverManager.getConnection(DB,"root","kvuc");
      Statement stmt= con.createStatement();
      ResultSet rs= stmt.executeQuery(qr);
      if(rs.next())                                   // if record found extract & display
       {  name = rs.getString("Name");
          sub = rs.getString("subject");
          jTextField1.setText(name);
          jTextField2.setText(sub);
          con.close(); stmt.close(); rs.close();    // close connection
       }
      else                                 // if record not found, Display Error in a dialog
        JOptionPane.showMessageDialog(null, "Mobile Number Not Found");
   }
catch(Exception e)
      { JOptionPane.showMessageDialog(null,e.getMessage());}   }
```

Example 2:
Entry of records in a table using a Form

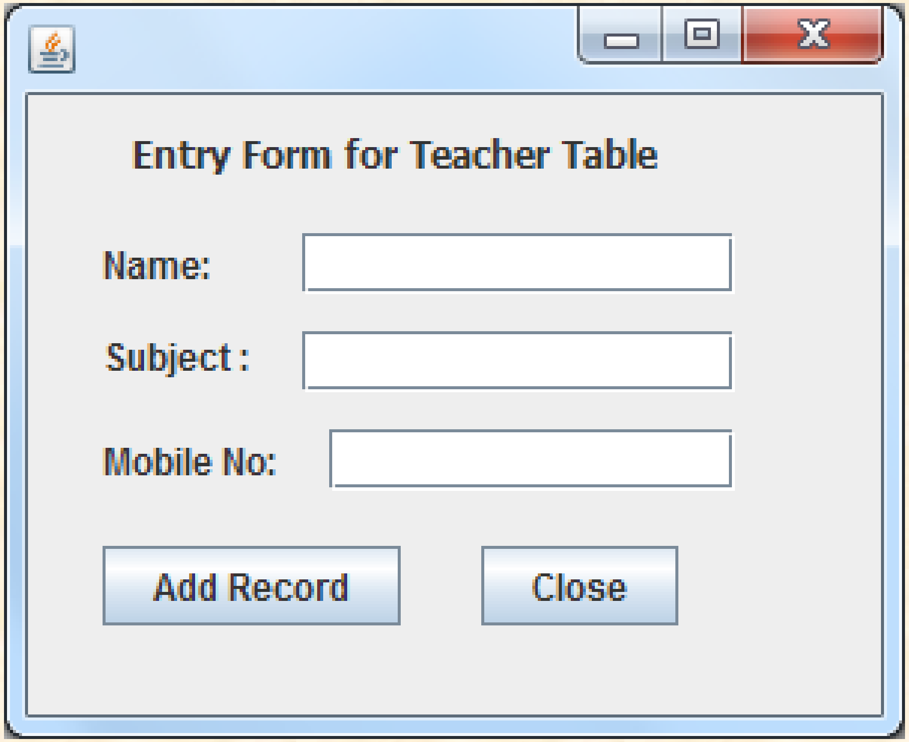
Objective :

Consider the following design of a database application to Enter records in the Teacher table containing Name, Subject and Mobile Number column.

Assumption
Database : School
Table : Teacher
Column / Field & Type
- Name Character (40)
- Subject Varchar(30)
- Mobile Char(12)

With some records.

MySQL
User Name: root
Password: kvuc



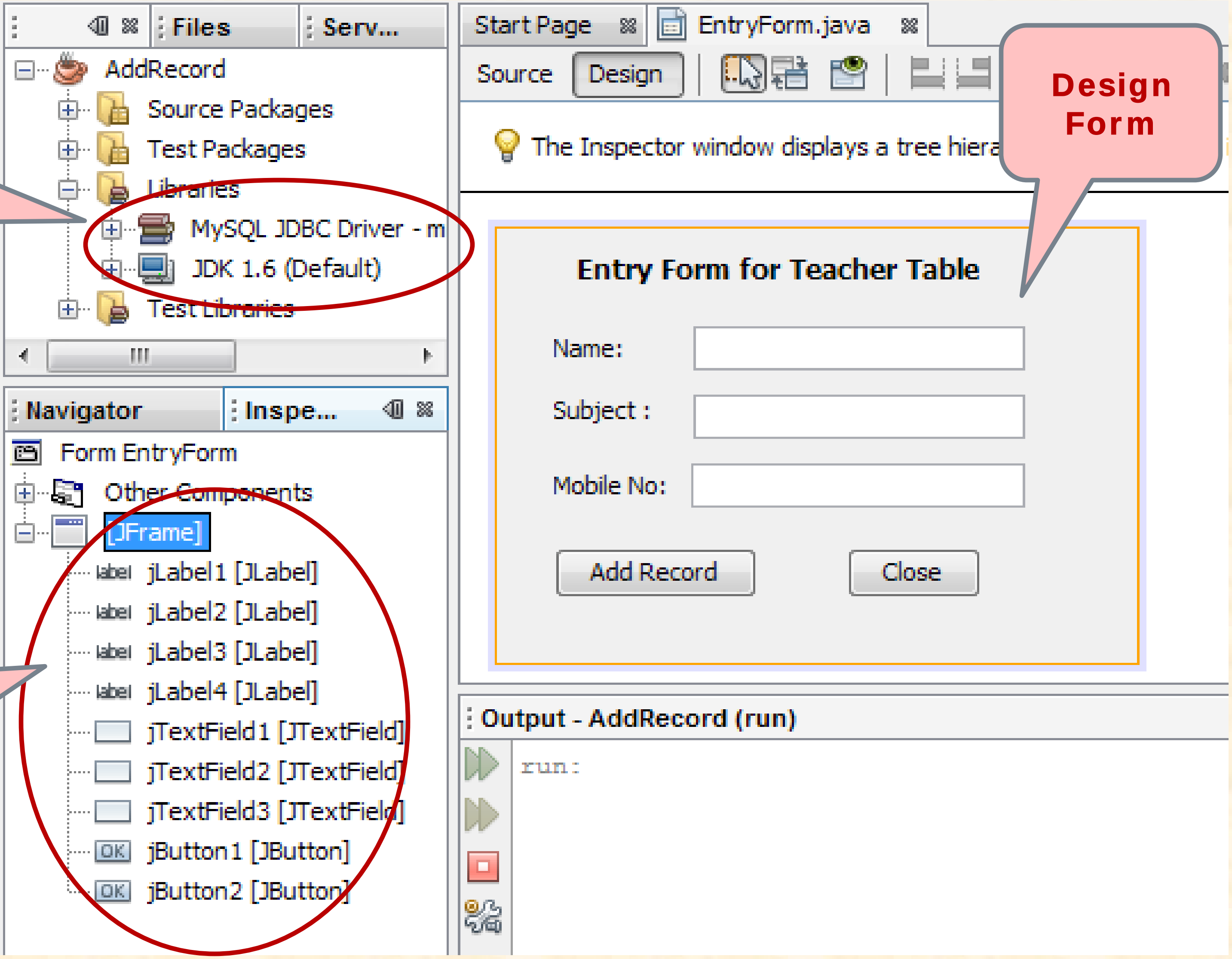
Entry Form for Teacher Table

Name:

Subject :

Mobile No:

Example 2:
Design of Entry Form in NetBeans



AddRecord
- Source Packages
- Test Packages
- Libraries
  - MySQL JDBC Driver - m
  - JDK 1.6 (Default)
- Test Libraries

Form EntryForm
- Other Components
- [JFrame]
  - jLabel1 [JLabel]
  - jLabel2 [JLabel]
  - jLabel3 [JLabel]
  - jLabel4 [JLabel]
  - jTextField1 [JTextField]
  - jTextField2 [JTextField]
  - jTextField3 [JTextField]
  - jButton1 [JButton]
  - jButton2 [JButton]

The Inspector window displays a tree hiera

Entry Form for Teacher Table

Name:

Subject :

Mobile No:

Add Record   Close

Output - AddRecord (run)

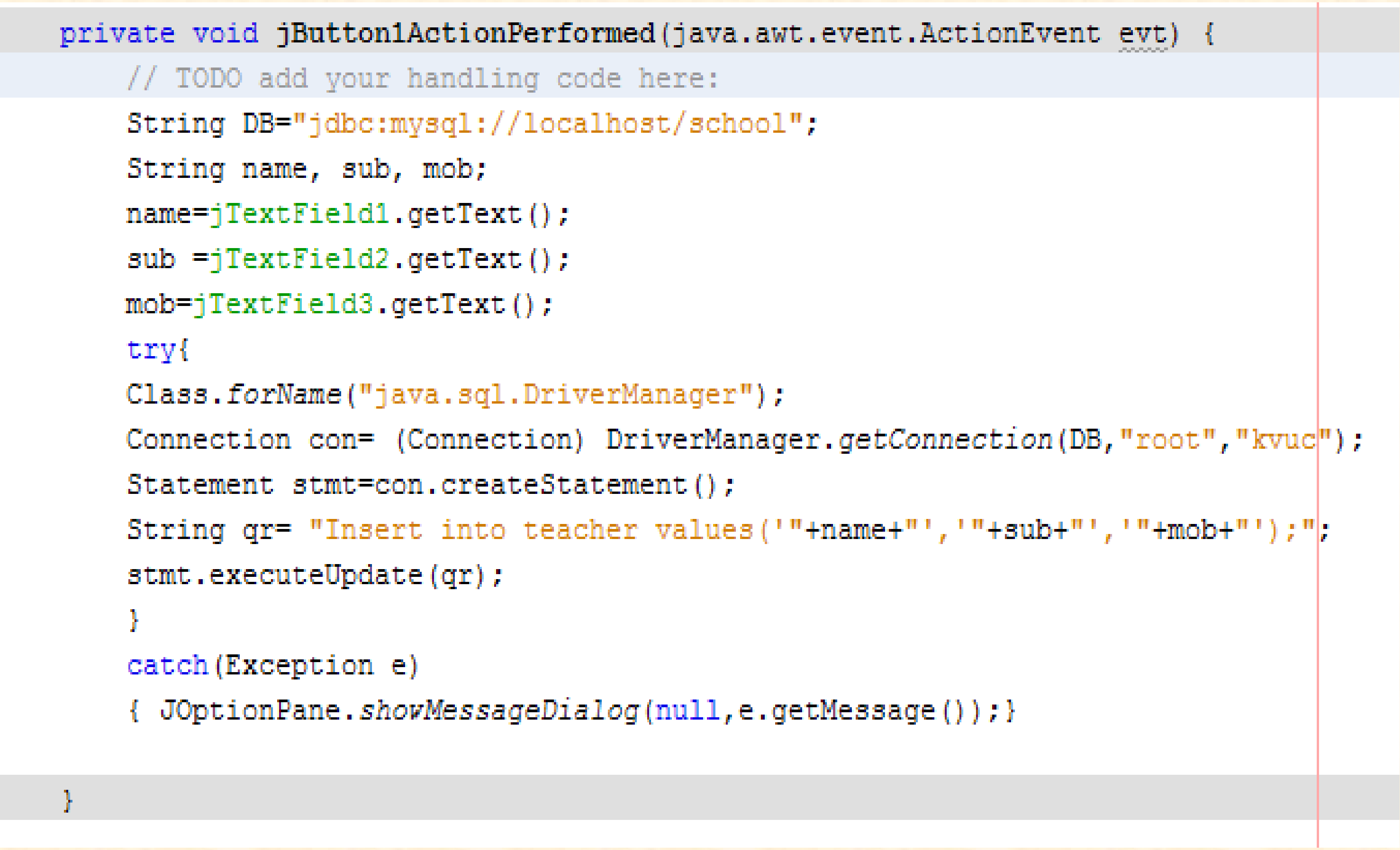
```
run:
```

Add MySQL JDBC Driver

Add Swing Controls

Design Form

Example 2:
Coding of Event in NetBeans

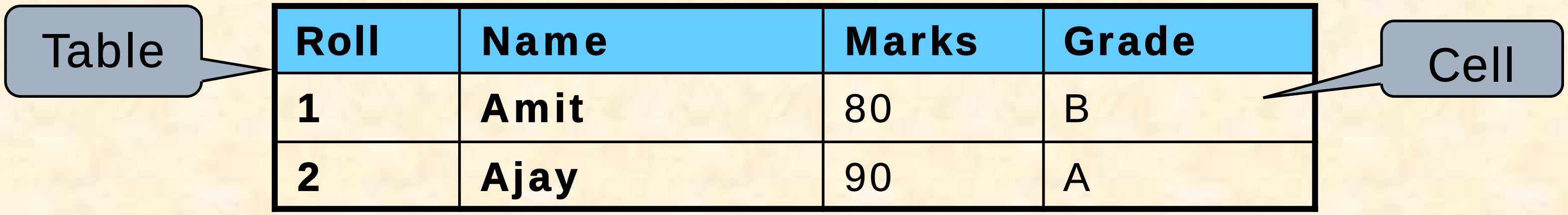


```
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        String DB="jdbc:mysql://localhost/school";
        String name, sub, mob;
        name=jTextField1.getText();
        sub =jTextField2.getText();
        mob=jTextField3.getText();
        try{
        Class.forName("java.sql.DriverManager");
        Connection con= (Connection) DriverManager.getConnection(DB,"root","kvuc");
        Statement stmt=con.createStatement();
        String qr= "Insert into teacher values('"+name+"','"+sub+"','"+mob+"');";
        stmt.executeUpdate(qr);
        }
        catch(Exception e)
        { JOptionPane.showMessageDialog(null,e.getMessage());}

    }
```

jTable Swing Control

Sometimes it is required to represent information in tabular form. Java provides JTable swing control to handle multiple records retrieved from the database. A table consists of certain rows and columns.



Roll	Name	Marks	Grade
1	Amit	80	B
2	Ajay	90	A

A table model works behind JTable control which contains source data for JTable. Java provides multiple Table model, but **DefaultTableModel** is commonly used.

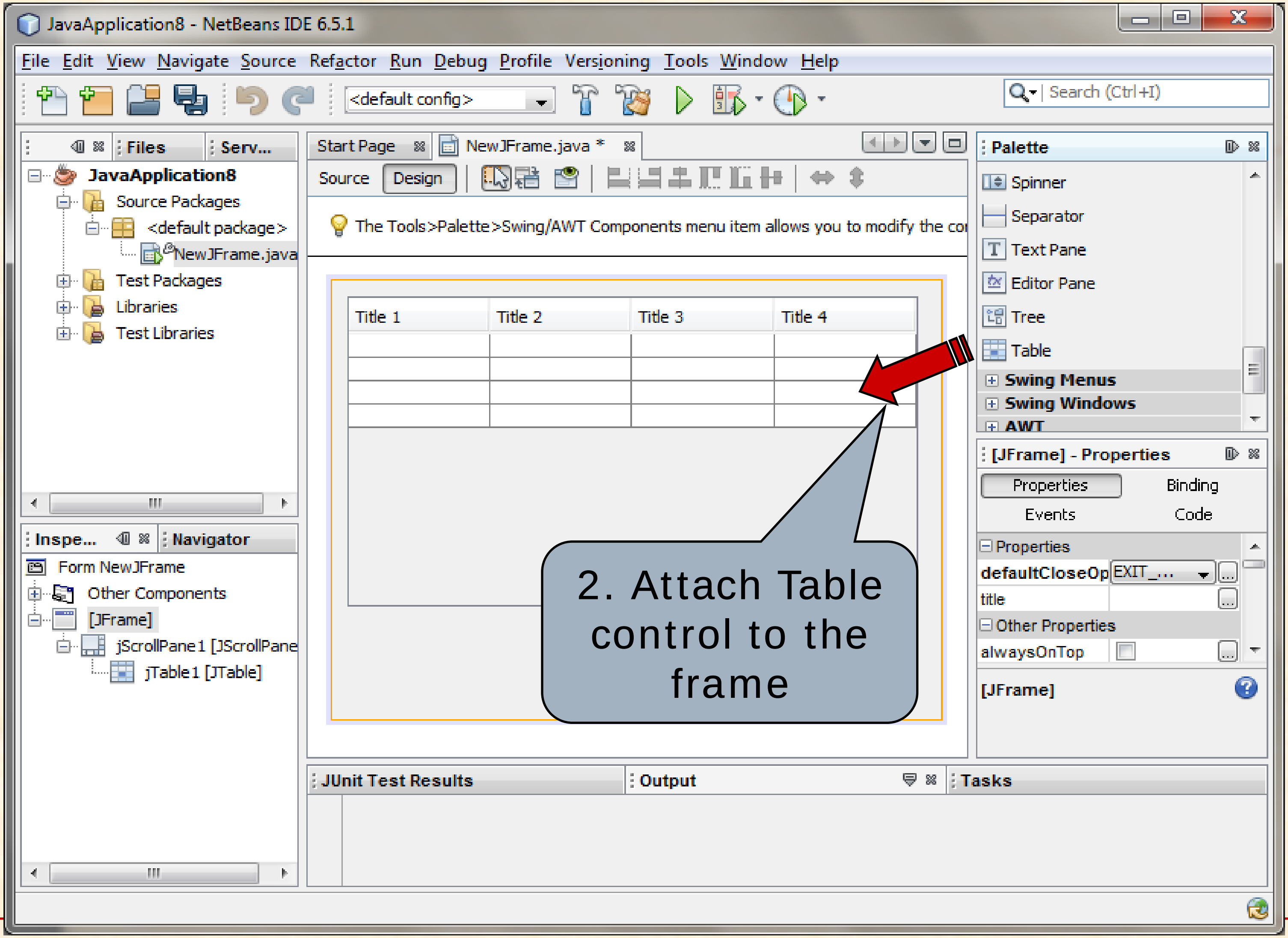
❑ Properties & Methods of JTable control:

Method	Description
int getColumnCount()	Returns the number of column in the table
int getRowCount()	Returns the number of rows in the table
Object getValueAt(row,col)	Returns value of given row & column of the table

The most commonly used properties are Font, Foreground and Enabled etc.

Designing a Simple Table

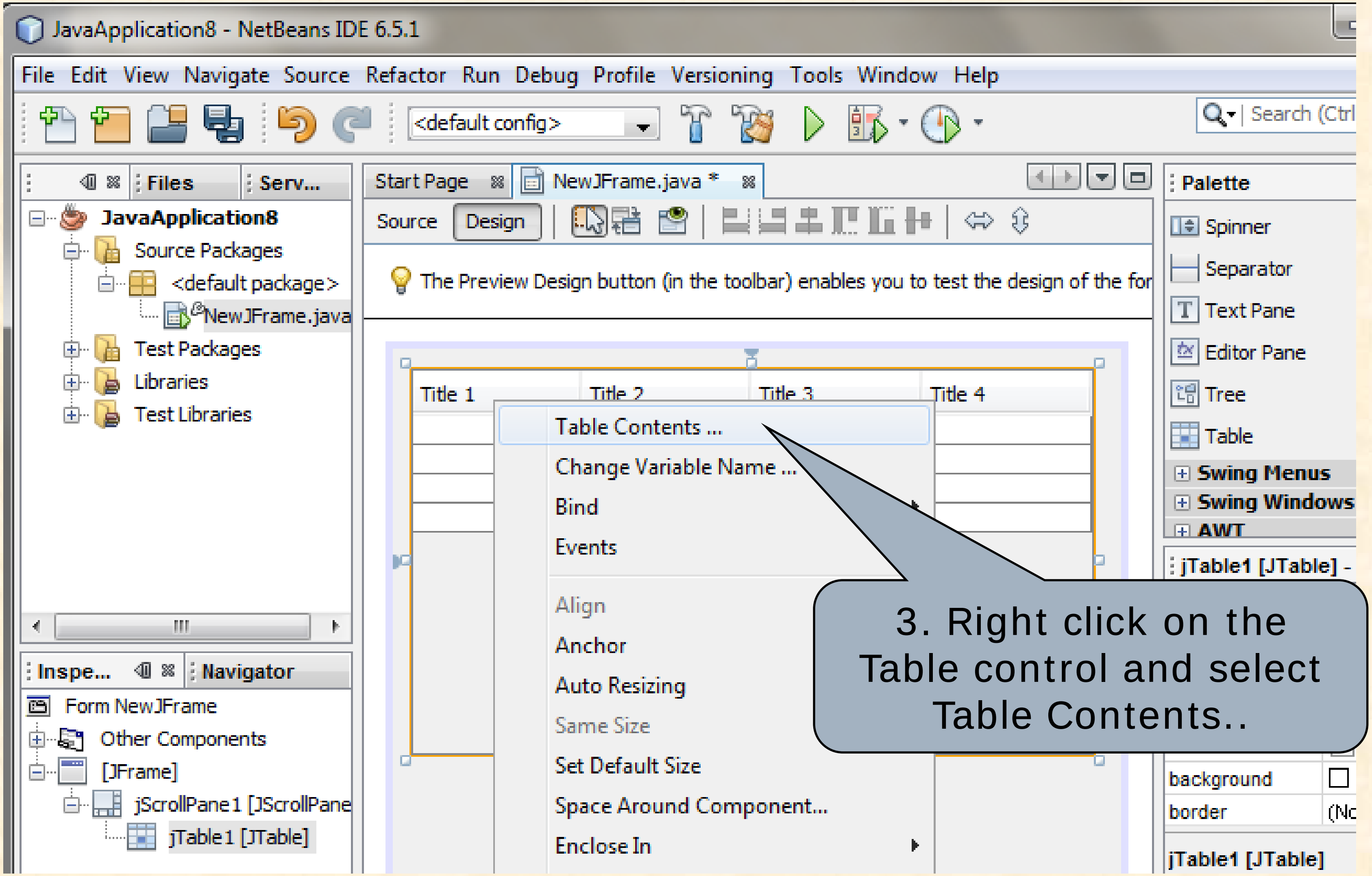
1. Create an application and attach a JFrame (Form).



The Tools>Palette>Swing/AWT Components menu item allows you to modify the co

2. Attach Table control to the frame

Designing a Simple Table

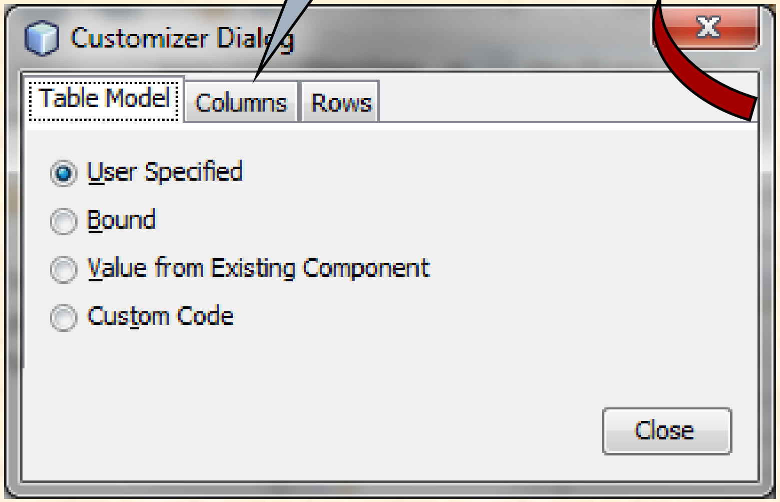


The Preview Design button (in the toolbar) enables you to test the design of the for

3. Right click on the Table control and select Table Contents..

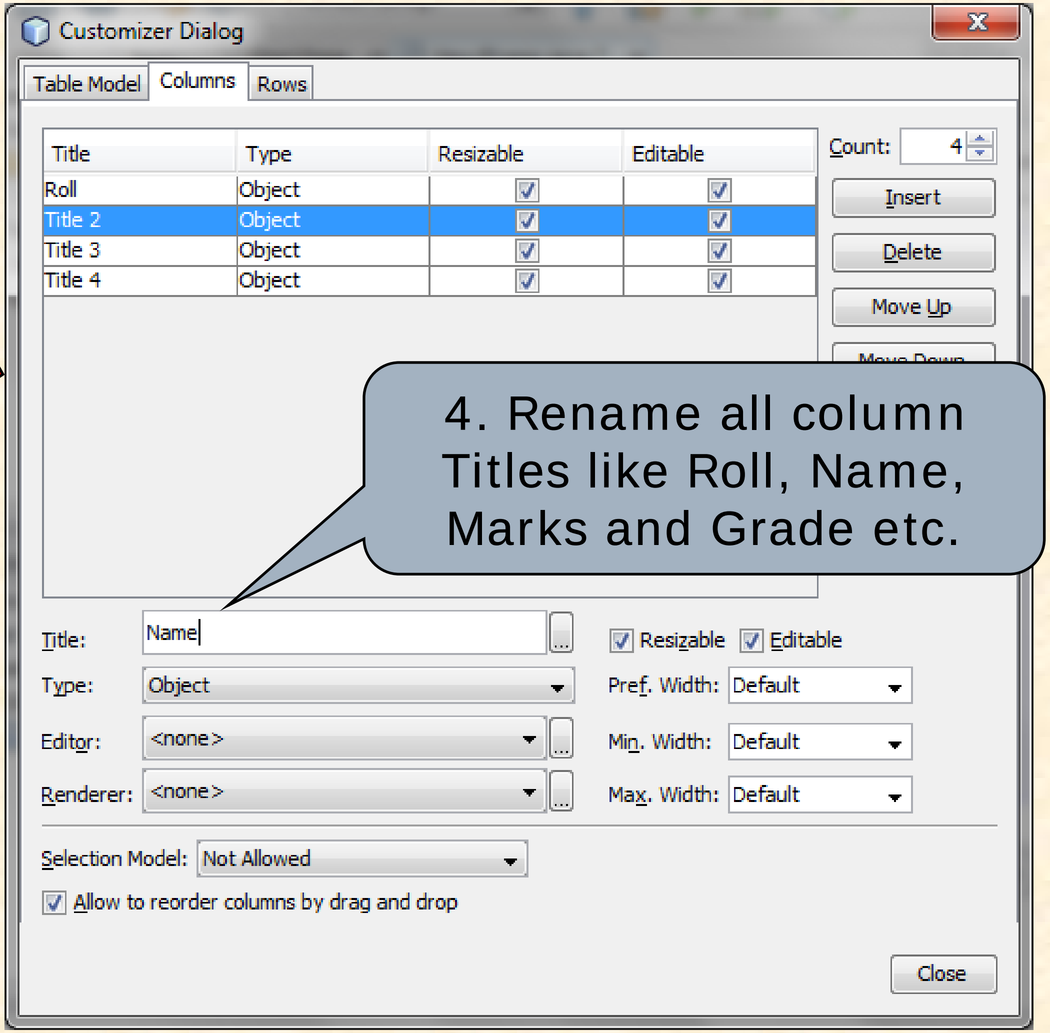
Designing a Simple Table

3. Select Column tab



Customizer Dialog

● User Specified
○ Bound
○ Value from Existing Component
○ Custom Code



Customizer Dialog

Title	Type	Resizable	Editable
Roll	Object	☑	☑
Title 2	Object	☑	☑
Title 3	Object	☑	☑
Title 4	Object	☑	☑

Count: 4

Title: Name

☑ Resizable ☑ Editable

Type: Object

Pref. Width: Default

Editor: <none>

Min. Width: Default

Renderer: <none>

Max. Width: Default

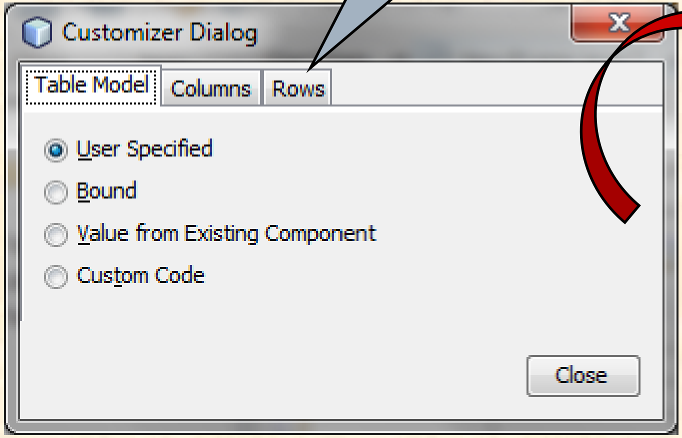
Selection Model: Not Allowed

☑ Allow to reorder columns by drag and drop

4. Rename all column Titles like Roll, Name, Marks and Grade etc.

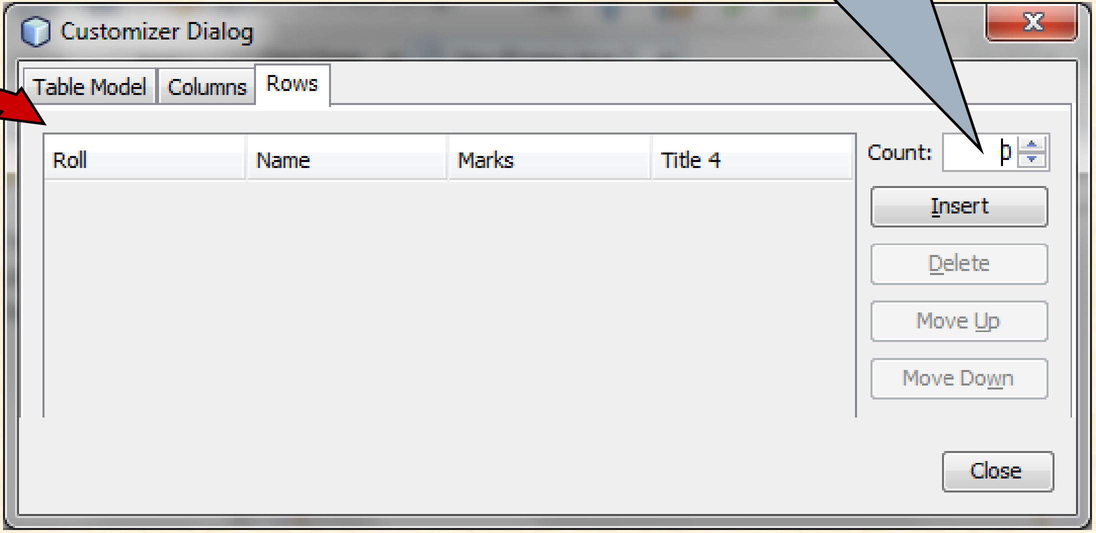
Designing a Simple Table

5. Select Row tab



● User Specified
○ Bound
○ Value from Existing Component
○ Custom Code

6. Set Count as 0



Roll	Name	Marks	Title 4

Count: 0

Now attach Button controls on the Form and write TODO code in ActionPerformed event for the specific functionality.

Working with jTable

- Insert the following import statement at the beginning.

```
import javax.swing.table.*;
```

- Obtain table's model in a DefaultTableModel object as per the following (Suppose *tm* is an identifier and *jTable1* is table)-

```
DefaultTableModel tm=(DefaultTableModel) jTable1.getModel();
```

- **Adding Rows**

1. Create an object array and put values (directly or using TextFields) in the order in which jTable is designed.

```
object myrow[ ]= {5, "Mukesh Kumar",60,"B"};
object myrow[ ]= {jTextField1.getText(), jTextField2.getText(),
                  jTextField3.getText(), jTextField4.getText()};
```

2. Add object array in TableModel by using addrow() method.

```
tm.addRow(myrow);
```

- **Deleting Rows**

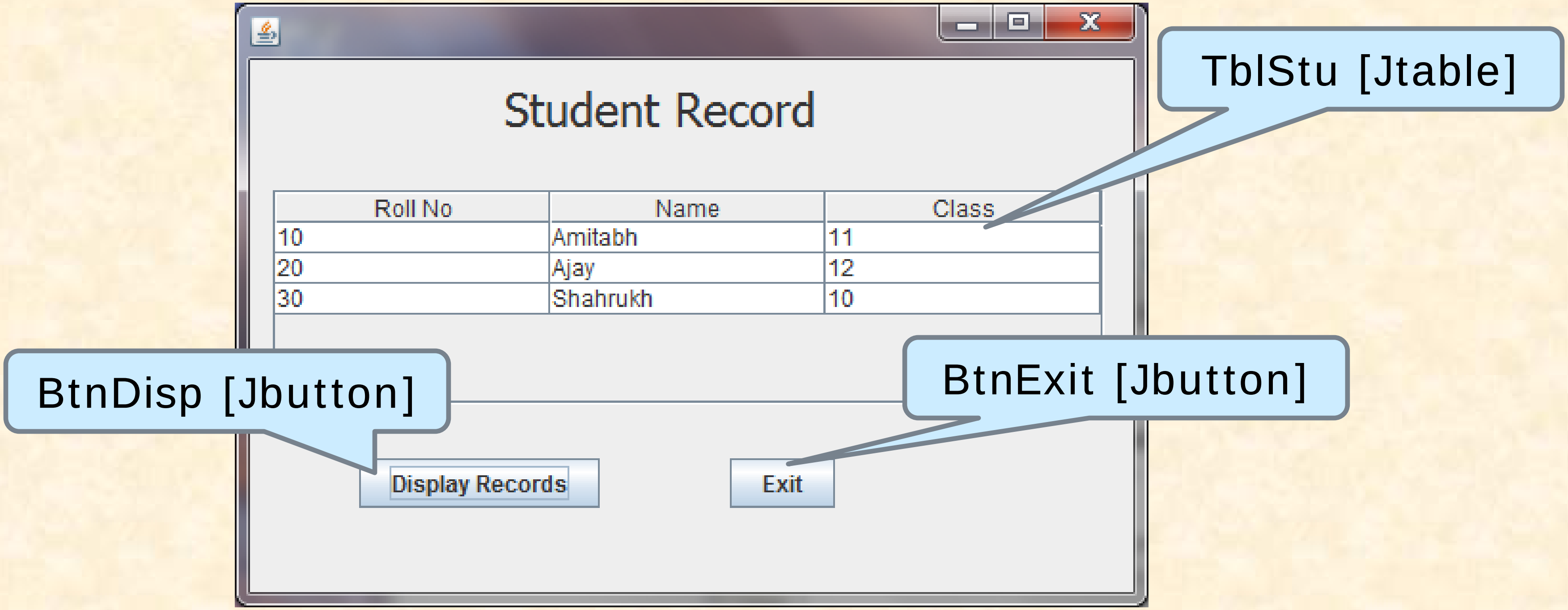
To delete a row, you may call removeRow() method with row number to be deleted.

```
tm.removerow(2);
```

Example 3:
Displaying Records in jTable Control

Let us design an Application as per the following screen shot. We assume that a Database named School containing a Student (Roll, Name, Class) table with some test records has been created already in MySQL.

A Simple Database Application using Table



Student Record

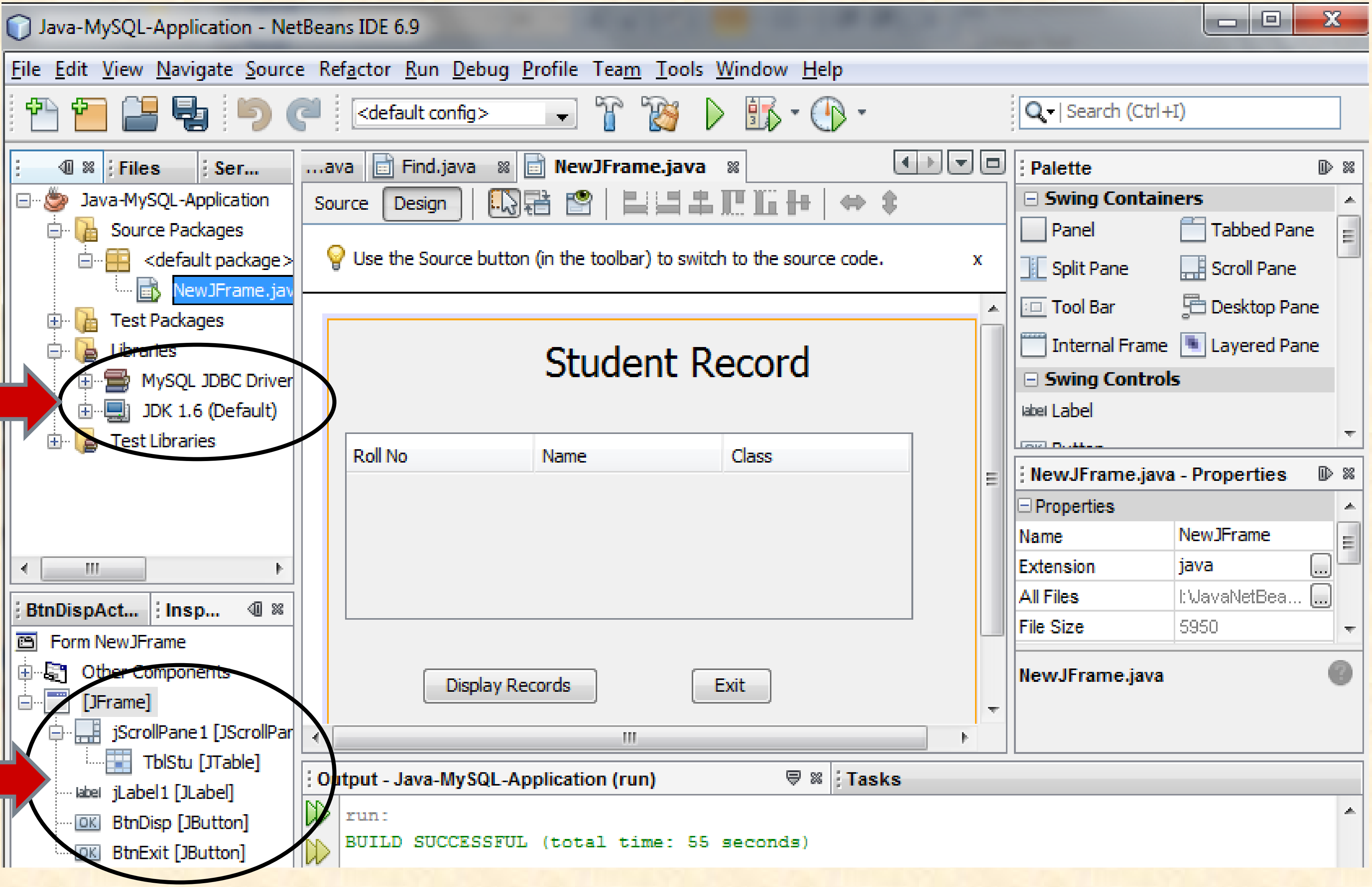
Roll No	Name	Class
10	Amitabh	11
20	Ajay	12
30	Shahrukh	10

TblStu [Jtable]

BtnDisp [Jbutton]

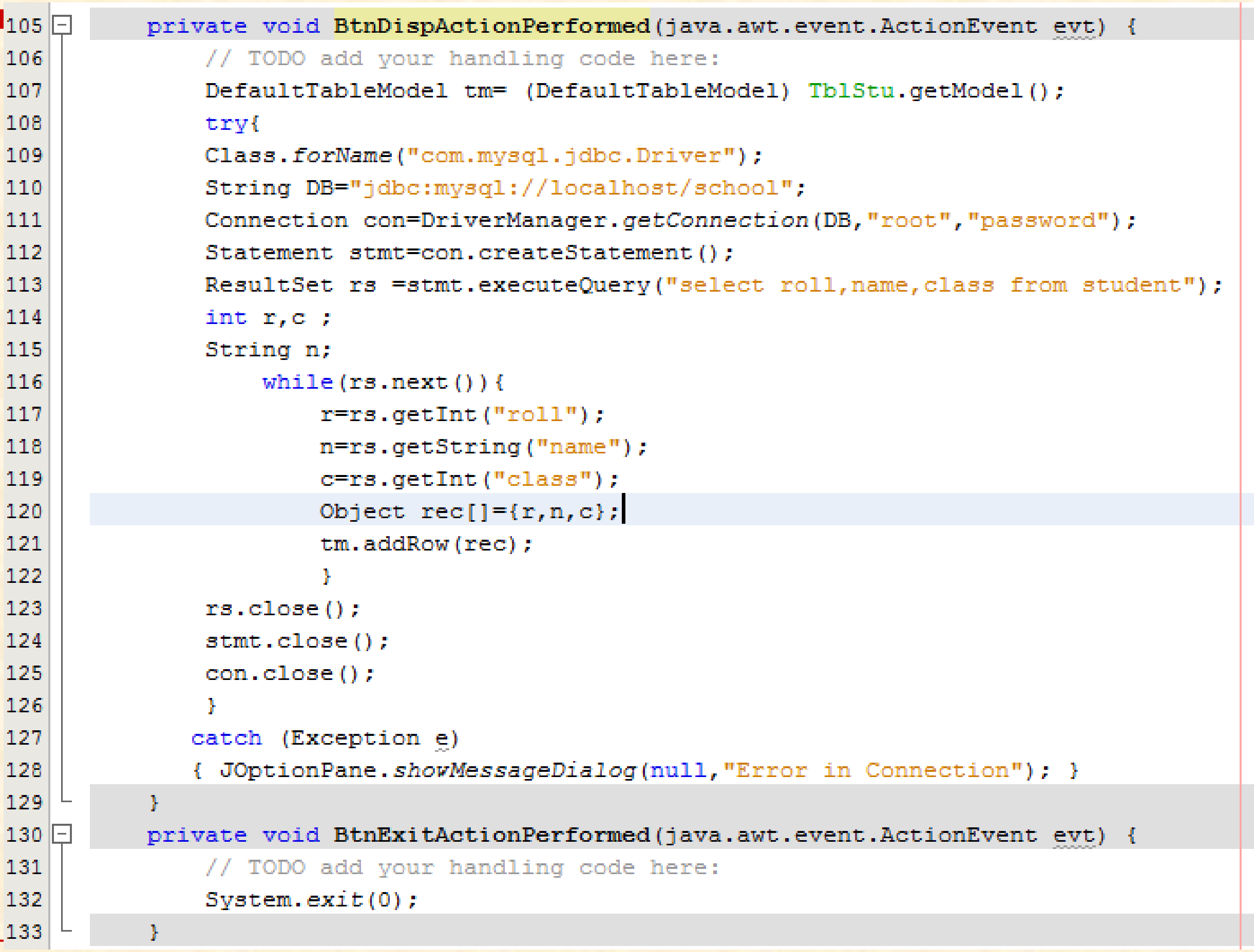
BtnExit [Jbutton]

Example 3:
Designing Frame with jTable



```
run:
BUILD SUCCESSFUL (total time: 55 seconds)
```

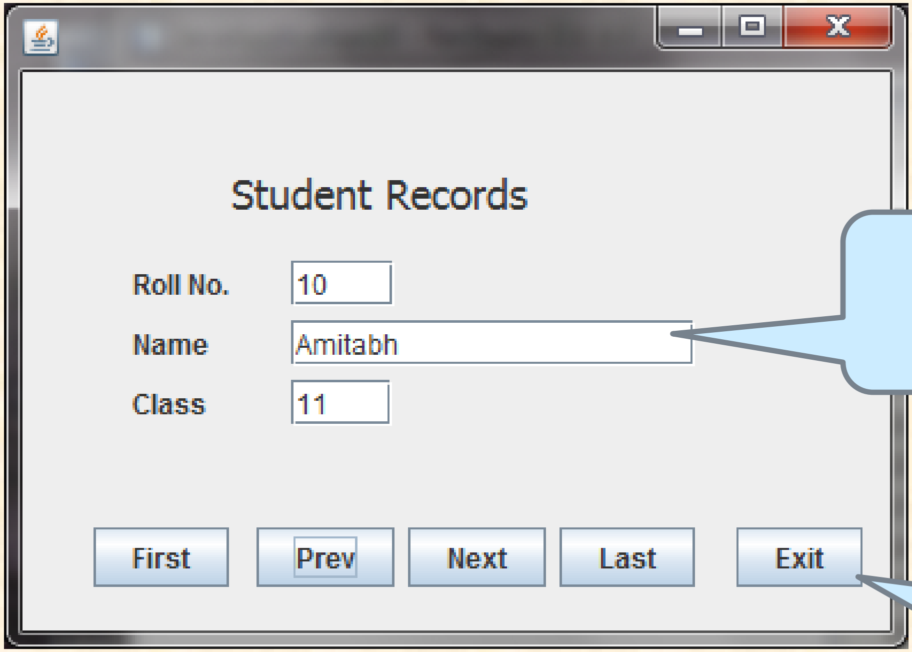
Example 3:
Coding Event for the jTable & Database Connectivity



```
    private void BtnDispActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        DefaultTableModel tm= (DefaultTableModel) TblStu.getModel();
        try{
        Class.forName("com.mysql.jdbc.Driver");
        String DB="jdbc:mysql://localhost/school";
        Connection con=DriverManager.getConnection(DB,"root","password");
        Statement stmt=con.createStatement();
        ResultSet rs =stmt.executeQuery("select roll,name,class from student");
        int r,c ;
        String n;
            while(rs.next()){
                r=rs.getInt("roll");
                n=rs.getString("name");
                c=rs.getInt("class");
                Object rec[]={r,n,c};
                tm.addRow(rec);
                }
        rs.close();
        stmt.close();
        con.close();
        }
       catch (Exception e)
       { JOptionPane.showMessageDialog(null,"Error in Connection"); }
    }
    private void BtnExitActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        System.exit(0);
    }
```

Example 4:
Navigating Records in Text Fields

Let us Redesign design the Previous Application as per the following screen shot using Text Fields and Navigation Buttons.
We assume the same Database named School containing a Student (Roll, Name, Class) table with some test records has been created already in MySQL.



Student Records

Roll No. 10

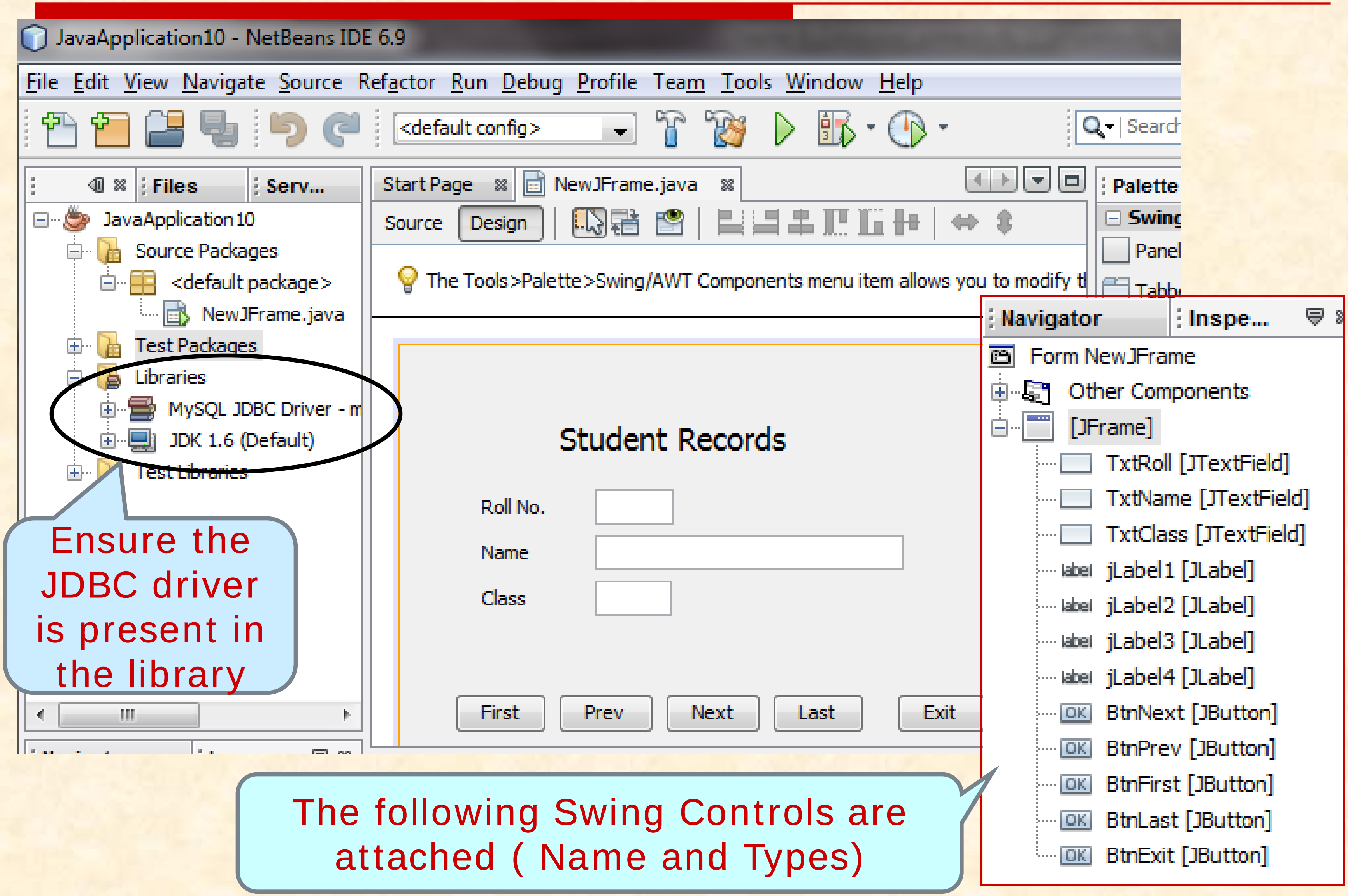
Name Amitabh

Class 11

JTextField (s) as TxtRoll, TxtName & TxtClass

JButtons as BtnFirst, BtnPrev, BtnNext, BtnLast & BtnExit

Example 4:
Design of Application in NetBeans



The Tools>Palette>Swing/AWT Components menu item allows you to modify t

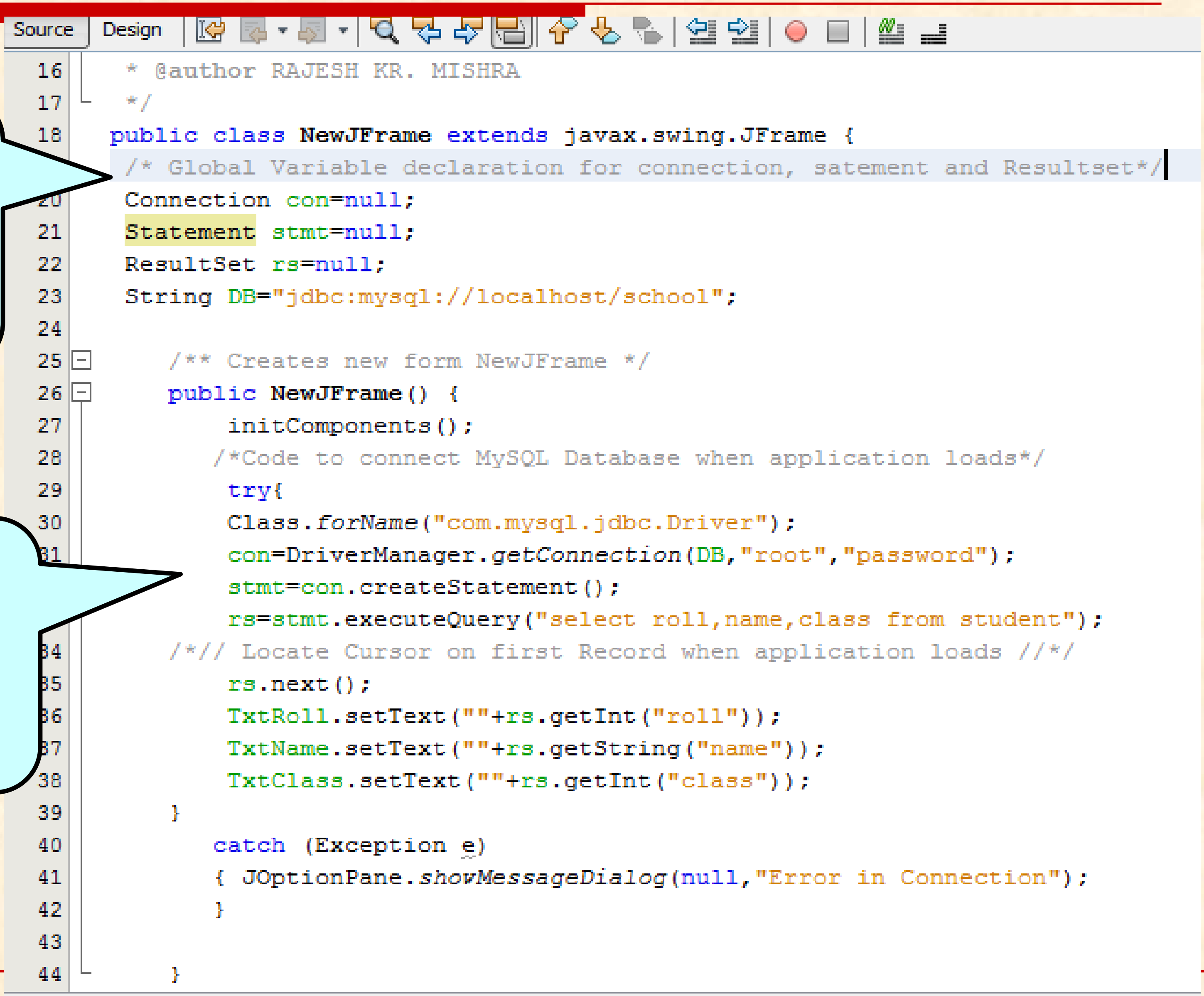
- Form NewJFrame
  - Other Components
  - [JFrame]
    - TxtRoll [JTextField]
    - TxtName [JTextField]
    - TxtClass [JTextField]
    - jLabel1 [JLabel]
    - jLabel2 [JLabel]
    - jLabel3 [JLabel]
    - jLabel4 [JLabel]
    - BtnNext [JButton]
    - BtnPrev [JButton]
    - BtnFirst [JButton]
    - BtnLast [JButton]
    - BtnExit [JButton]

Ensure the JDBC driver is present in the library

The following Swing Controls are attached (Name and Types)

Example 4:
Coding of events in NetBeans

Object are globally declared, so that they can be access in all methods.



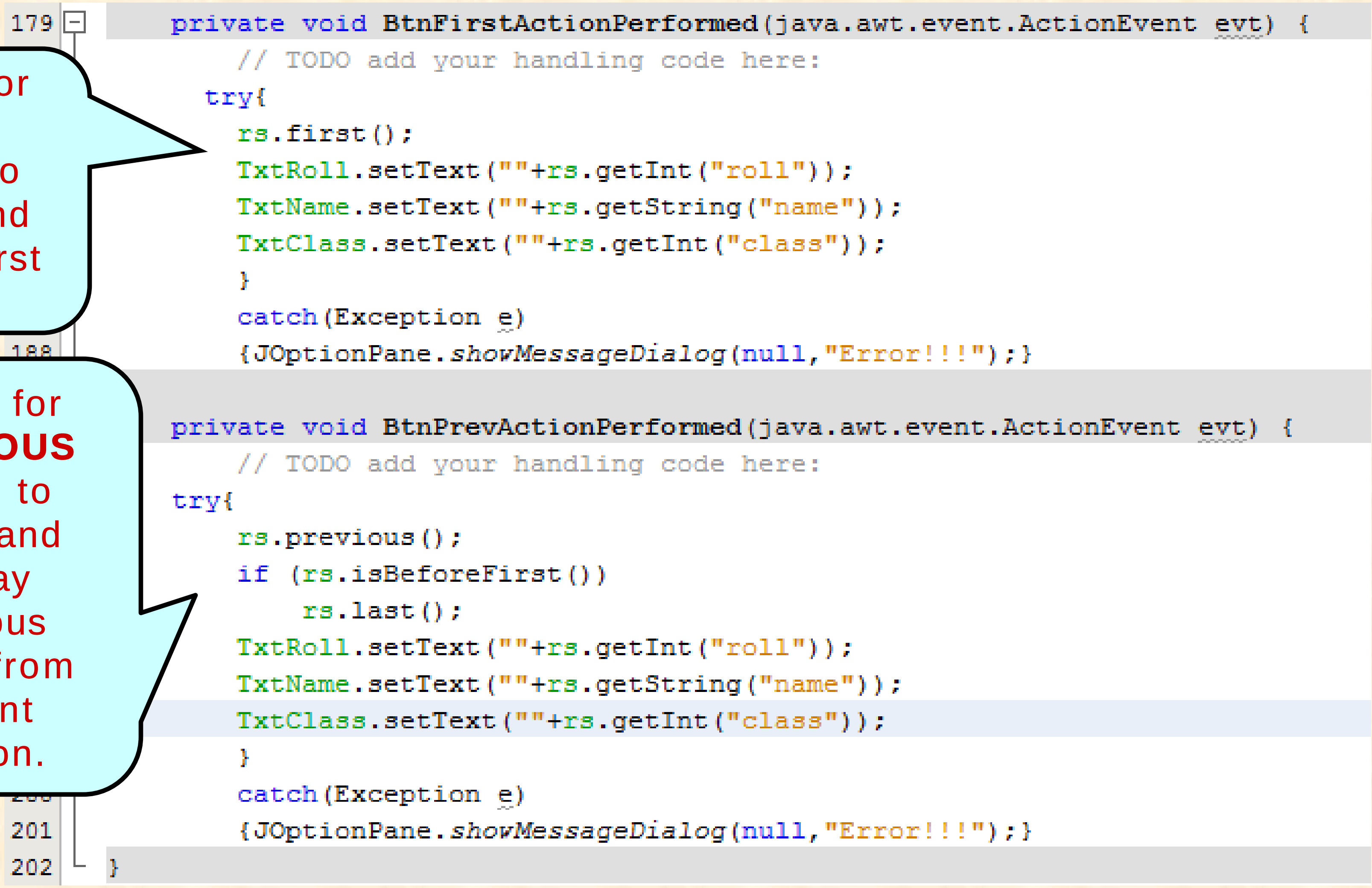
```
 * @author RAJESH KR. MISHRA
 */
public class NewJFrame extends javax.swing.JFrame {
 /* Global Variable declaration for connection, satement and Resultset*/
 Connection con=null;
 Statement stmt=null;
 ResultSet rs=null;
 String DB="jdbc:mysql://localhost/school";

    /** Creates new form NewJFrame */
    public NewJFrame() {
        initComponents();
       /*Code to connect MySQL Database when application loads*/
        try{
        Class.forName("com.mysql.jdbc.Driver");
        con=DriverManager.getConnection(DB,"root","password");
        stmt=con.createStatement();
        rs=stmt.executeQuery("select roll,name,class from student");
    /*// Locate Cursor on first Record when application loads //*/
        rs.next();
        TxtRoll.setText(""+rs.getInt("roll"));
        TxtName.setText(""+rs.getString("name"));
        TxtClass.setText(""+rs.getInt("class"));
    }
       catch (Exception e)
       { JOptionPane.showMessageDialog(null,"Error in Connection");
       }

    }
```

Connection is established and cursor is placed on first record when Frame loads.

Example 4:
Coding of events in NetBeans



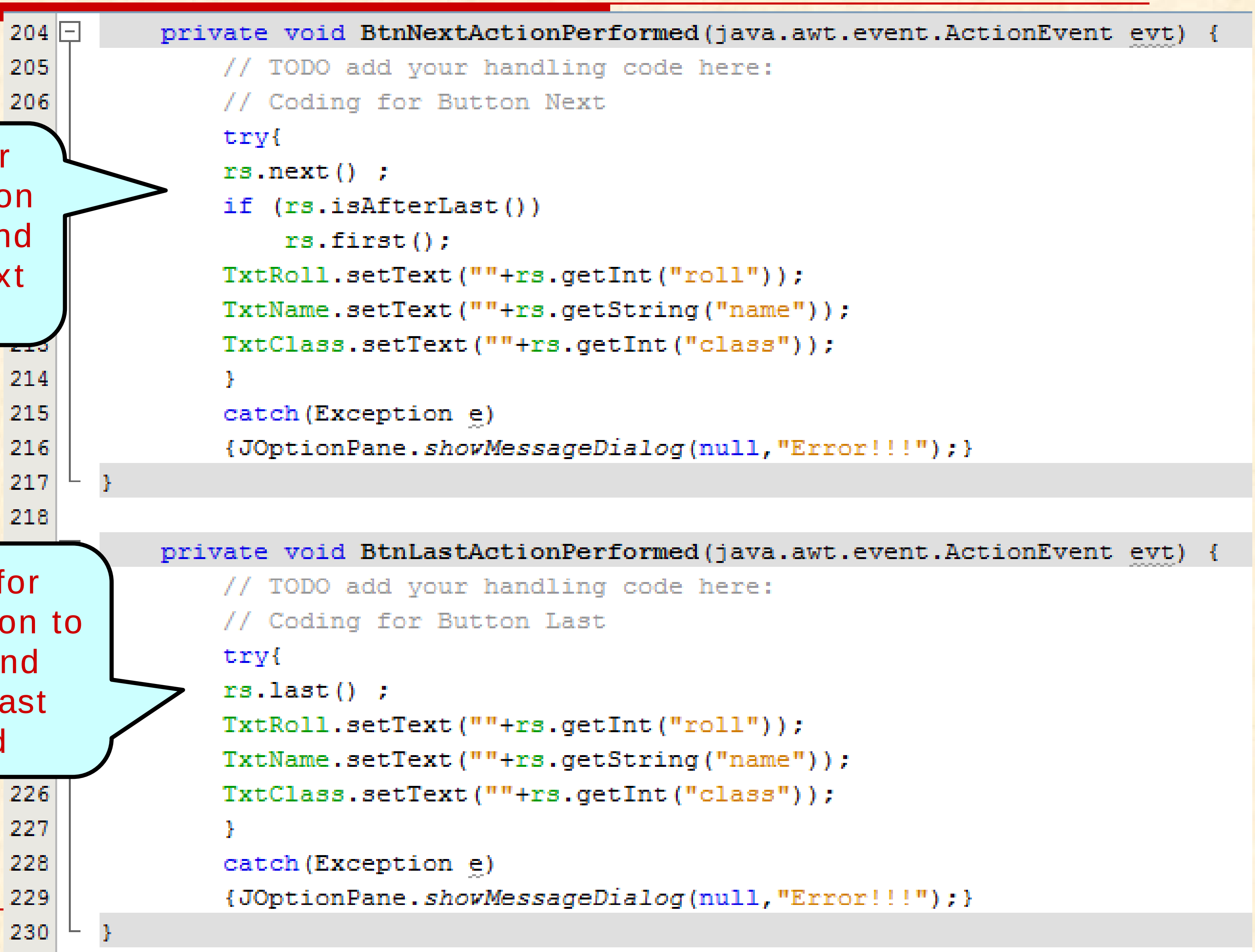
```java
    private void BtnFirstActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
      try{
        rs.first();
        TxtRoll.setText(""+rs.getInt("roll"));
        TxtName.setText(""+rs.getString("name"));
        TxtClass.setText(""+rs.getInt("class"));
        }
        catch(Exception e)
        {JOptionPane.showMessageDialog(null,"Error!!!");}

    private void BtnPrevActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    try{
        rs.previous();
        if (rs.isBeforeFirst())
            rs.last();
        TxtRoll.setText(""+rs.getInt("roll"));
        TxtName.setText(""+rs.getString("name"));
        TxtClass.setText(""+rs.getInt("class"));
        }
        catch(Exception e)
        {JOptionPane.showMessageDialog(null,"Error!!!");}
}
```

Coding for **FIRST** button to locate and display first record.

Coding for **PREVIOUS** button to locate and display previous record from current position.

Example 4:
Coding of events in NetBeans



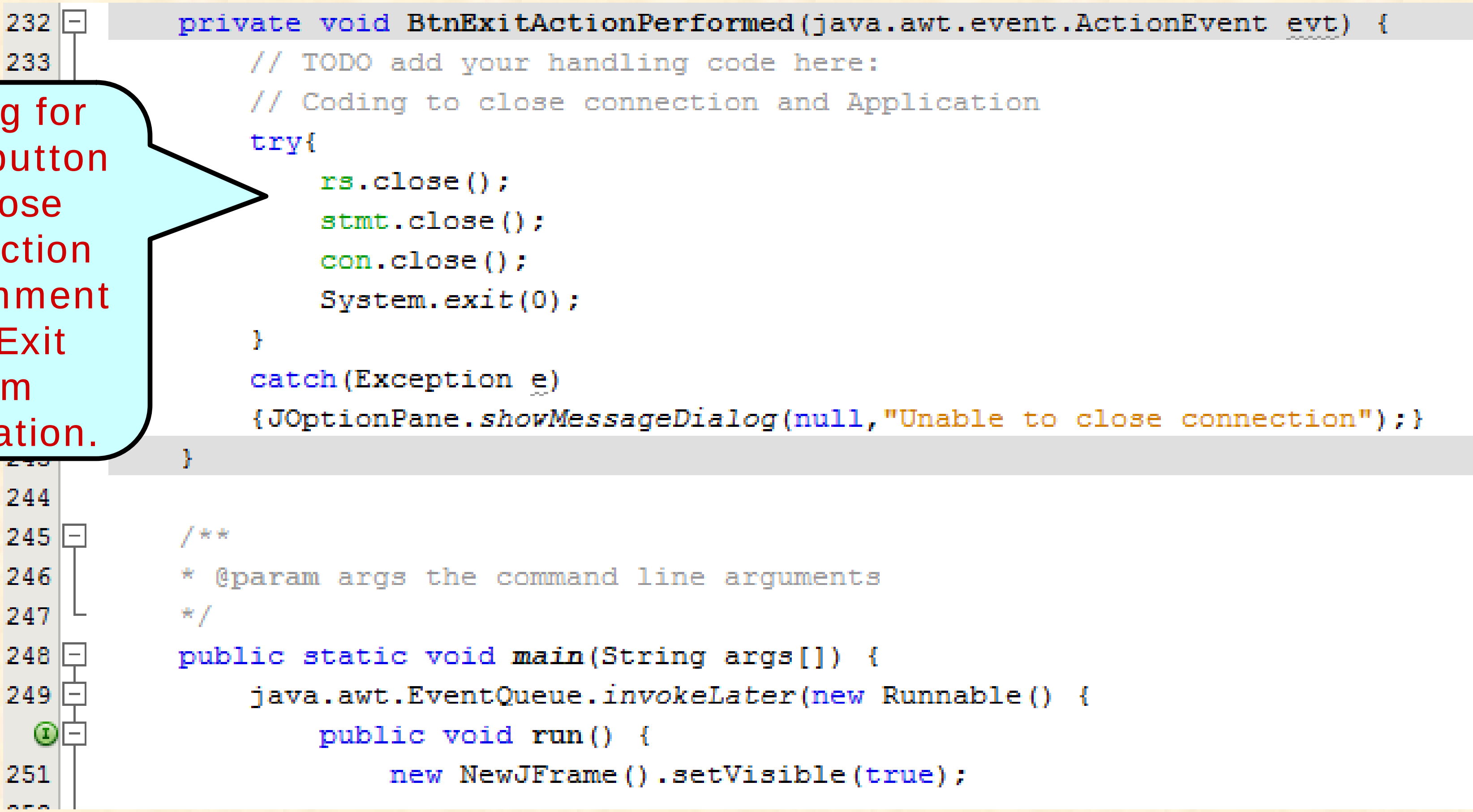
```java
    private void BtnNextActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        // Coding for Button Next
        try{
        rs.next() ;
        if (rs.isAfterLast())
            rs.first();
        TxtRoll.setText(""+rs.getInt("roll"));
        TxtName.setText(""+rs.getString("name"));
        TxtClass.setText(""+rs.getInt("class"));
        }
        catch(Exception e)
        {JOptionPane.showMessageDialog(null,"Error!!!");}
}

    private void BtnLastActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        // Coding for Button Last
        try{
        rs.last() ;
        TxtRoll.setText(""+rs.getInt("roll"));
        TxtName.setText(""+rs.getString("name"));
        TxtClass.setText(""+rs.getInt("class"));
        }
        catch(Exception e)
        {JOptionPane.showMessageDialog(null,"Error!!!");}
}
```

Coding for **NEXT** button to locate and display next record.

Coding for **LAST** button to locate and display last record

Example 4:
Coding of events in NetBeans



```
    private void BtnExitActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        // Coding to close connection and Application
        try{
            rs.close();
            stmt.close();
            con.close();
            System.exit(0);
        }
        catch(Exception e)
        {JOptionPane.showMessageDialog(null,"Unable to close connection");}
    }

    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new NewJFrame().setVisible(true);
```

Coding for **EXIT** button to close connection environment and Exit from application.