Java Project 2 – Video Store

Donovan van Heerden

EL2014-0043

**Contents:**

* Source Code
* User Documentation

Source Code:

/\* Filename: VideoStoreGUI.java

\* Author: Donovan van Heerden (EL2014-0043)

\* Created: 19-01-2014

\* Operating System: Windows 7

\* Version: 1.0

\* Description: This program handles interation to a SQL Server Database,

\* called 'VideoStoreDb'.

\*/

package javaproject2\_videostore;

import java.sql.\*;

import javax.swing.\*;

import java.util.\*;

import java.util.regex.Pattern;

/\*\*

\*

\* VideoStoreGUI Class, handles interaction to a SQL Server Database called,

\* VideoStoreDb.

\*

\* @author Donovan van Heerden

\*/

public class VideoStoreGUI

{

//Globals

private static final String[] tableCols = {"Movie Name", "Movie Description", "Genre"};

//------------------------------------------------------------------------------

//GUI Globals - VideoStore Window

private static final JMenuBar MenuBar = new JMenuBar();

private static final JMenu InfoMenu = new JMenu();

private static final JMenu AdminMenu = new JMenu();

private static final JMenu SearchMenu = new JMenu();

private static final JMenuItem btnHelp = new JMenuItem();

private static final JMenuItem btnAbout = new JMenuItem();

private static final JMenuItem btnSearchMov = new JMenuItem();

private static final JMenuItem btnSearchGen = new JMenuItem();

private static final JMenuItem btnAdmin = new JMenuItem();

private static final JFrame ClientMainWindow = new JFrame();

private static final JLabel lblSearch = new JLabel();

private static final JTextField txtSearch = new JTextField();

private static JTable dtResults = new JTable();

private static final JScrollPane spResults = new JScrollPane();

private static final JPanel pnlSearchControls = new JPanel();

//------------------------------------------------------------------------------

//GUI Globals - Administrator Window

private static final JFrame AdminMainWindow = new JFrame();

private static final JMenuBar AdminMenuBar = new JMenuBar();

private static final JMenu AdminInfoMenu = new JMenu();

private static final JMenu AdminMainMenu = new JMenu();

private static final JMenuItem btnAdminHelp = new JMenuItem();

private static final JMenuItem btnAdminAbout = new JMenuItem();

private static final JMenuItem btnAdminLogout = new JMenuItem();

private static final JSeparator sepMovie = new JSeparator();

private static final JSeparator sepGenre = new JSeparator();

private static final JComboBox cbxMovieList = new JComboBox();

private static final JComboBox cbxMovieGenre = new JComboBox();

private static final JComboBox cbxGenreList = new JComboBox();

private static final JLabel lblMovieTitle = new JLabel();

private static final JLabel lblMovieDesc = new JLabel();

private static final JLabel lblMovieGenre = new JLabel();

private static final JTextField txtMovieTitle = new JTextField();

//public static JTextField txtMovieGenre = new JTextField();

private static final JTextArea taMovieDesc = new JTextArea();

private static final JScrollPane spMovieDesc = new JScrollPane();

private static final JTextField txtGenre = new JTextField();

private static final JLabel lblGenre = new JLabel();

private static final JLabel lblAllGenre = new JLabel();

private static final JLabel lblAllMovie = new JLabel();

private static final JButton btnAddMov = new JButton();

private static final JButton btnDeleteMov = new JButton();

private static final JButton btnAddGen = new JButton();

//private static JButton btnDeleteGen = new JButton();

private static final JPanel pnlMovieInfo = new JPanel();

private static final JPanel pnlGenreInfo = new JPanel();

private static final JTabbedPane tpAdminPnl = new JTabbedPane();

//------------------------------------------------------------------------------

//GUI Globals - Login Window

private static final JFrame LoginMainWindow = new JFrame();

private static final JTextField txtUsername = new JTextField(20);

private static final JLabel lblUsername = new JLabel();

private static final JPasswordField txtPassword = new JPasswordField(20);

private static final JLabel lblPassword = new JLabel();

private static final JButton btnLogin = new JButton();

private static final JButton btnCancel = new JButton();

//------------------------------------------------------------------------------

/\*\*

\* This is the constructor for the VideoStoreGUI class, which sets the ClientMainWindow's

\* Title, setDefaultCloseOperation, and sets the JFrame's resizable property to false.

\* Followed by calling the ConfigureMainWindow() method and MainWindow\_Action() method.

\* And lastly by making the JFrame visible.

\*/

public VideoStoreGUI()

{

ClientMainWindow.setTitle("VideoStore: Browse");

ClientMainWindow.setDefaultCloseOperation(JFrame.DO\_NOTHING\_ON\_CLOSE);

ClientMainWindow.setResizable(false);

ConfigureMainWindow();

MainWindow\_Action();

ClientMainWindow.setVisible(true);

}

//------------------------------------------------------------------------------

/\*\*

\* The main method, just builds the three various GUIs, but leaves the LoginWindow

\* and AdminWindow invisible until called for. Thus leaving the MainWindow, the

\* only accessable window initially.

\* @param args

\*/

public static void main(String[] args)

{

VideoStoreGUI VideoStore = new VideoStoreGUI();

BuildLoginWindow();

BuildAdminMainWindow();

}

//------------------------------------------------------------------------------

/\*\*

\* This method sets up the MainWindow GUI, setting the position of all the

\* controls.

\*/

public static void ConfigureMainWindow()

{

ClientMainWindow.setBackground(new java.awt.Color(255, 255, 255));

ClientMainWindow.setSize(620, 370);

ClientMainWindow.setLocation(220,180);

ClientMainWindow.getContentPane().setLayout(null);

ClientMainWindow.setJMenuBar(MenuBar);

MenuBar.setVisible(true);

btnAdmin.setText("Login");

AdminMenu.add(btnAdmin);

AdminMenu.setText("Admin");

MenuBar.add(AdminMenu);

btnSearchMov.setText("Movie Search");

btnSearchGen.setText("Genre Search");

SearchMenu.setText("Search Options");

SearchMenu.add(btnSearchMov);

SearchMenu.add(btnSearchGen);

MenuBar.add(SearchMenu);

btnHelp.setText("Help");

btnAbout.setText("About");

InfoMenu.setText("Info");

InfoMenu.add(btnAbout);

InfoMenu.add(btnHelp);

MenuBar.add(InfoMenu);

pnlSearchControls.setBounds(10, 10, 595, 300);

pnlSearchControls.setBorder(BorderFactory.createEtchedBorder());

pnlSearchControls.setLayout(null);

pnlSearchControls.setVisible(true);

lblSearch.setBounds(10, 10, 80, 25);

lblSearch.setText("Search:");

txtSearch.setBounds(60, 10, 525, 25);

pnlSearchControls.add(lblSearch);

pnlSearchControls.add(txtSearch);

spResults.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL\_SCROLLBAR\_NEVER);

spResults.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL\_SCROLLBAR\_ALWAYS);

spResults.getViewport().add(dtResults, null);

pnlSearchControls.add(spResults);

spResults.setBounds(10, 50, 575, 240);

dtResults.setFont(new java.awt.Font("Tahoma", 0, 12));

dtResults.setForeground(new java.awt.Color(0, 0, 0));

ClientMainWindow.getContentPane().add(pnlSearchControls);

}

//------------------------------------------------------------------------------

/\*\*

\* This method adds all the ActionListeners to the various controls within the

\* MainWindow.

\*/

public static void MainWindow\_Action()

{

try

{

btnAdmin.addActionListener(new java.awt.event.ActionListener() {

@Override

public void actionPerformed(java.awt.event.ActionEvent evt)

{

ClientMainWindow.setVisible(false);

txtSearch.setText(null);

dtResults = new JTable();

spResults.getViewport().add(dtResults);

spResults.setViewportView(dtResults);

LoginMainWindow.setVisible(true);

}

});

btnSearchMov.addActionListener(new java.awt.event.ActionListener() {

@Override

public void actionPerformed(java.awt.event.ActionEvent evt)

{

ACTION\_SEARCH("Movie");

}

});

btnAbout.addActionListener(new java.awt.event.ActionListener() {

@Override

public void actionPerformed(java.awt.event.ActionEvent evt)

{

ACTION\_ABOUT();

}

});

btnHelp.addActionListener(new java.awt.event.ActionListener() {

@Override

public void actionPerformed(java.awt.event.ActionEvent evt)

{

BROWSE\_HELP();

}

});

btnSearchGen.addActionListener(new java.awt.event.ActionListener() {

@Override

public void actionPerformed(java.awt.event.ActionEvent evt)

{

ACTION\_SEARCH("Genre");

}

});

ClientMainWindow.addWindowListener(new java.awt.event.WindowAdapter(){

@Override

public void windowClosing(java.awt.event.WindowEvent windowEvent)

{

ACTION\_CLOSE(ClientMainWindow);

}

});

}

catch(Exception X)

{

System.out.println(X);

}

}

//------------------------------------------------------------------------------

/\*\*

\* This method sets the LoginWindow GUI's title, sets the defaultCloseOperation,

\* and sets the JFrame's resizable property to false. Followed by calling the ConfigureLoginWindow()

\* and LoginWindow\_Action() methods. Thus building the AdminWindow.

\*/

public static void BuildLoginWindow()

{

LoginMainWindow.setTitle("VideoStore: Login");

LoginMainWindow.setDefaultCloseOperation(JFrame.DO\_NOTHING\_ON\_CLOSE);

LoginMainWindow.setResizable(false);

ConfigureLoginWindow();

LoginWindow\_Action();

}

//------------------------------------------------------------------------------

/\*\*

\* This method sets the LoginWindow GUI, setting the position of all the controls.

\*/

public static void ConfigureLoginWindow()

{

LoginMainWindow.setBackground(new java.awt.Color(255, 255, 255));

LoginMainWindow.setSize(250, 190);

LoginMainWindow.setLocation(520,180);

LoginMainWindow.getContentPane().setLayout(null);

lblUsername.setText("Username:");

lblUsername.setBounds(10, 10, 70, 25);

txtUsername.setBounds(80, 10, 155, 25);

LoginMainWindow.add(lblUsername);

LoginMainWindow.add(txtUsername);

lblPassword.setText("Password:");

lblPassword.setBounds(10, 45, 70, 25);

txtPassword.setBounds(80, 45, 155, 25);

LoginMainWindow.add(lblPassword);

LoginMainWindow.add(txtPassword);

btnLogin.setText("Login");

btnLogin.setBounds(55, 90, 140, 25);

btnCancel.setText("Cancel");

btnCancel.setBounds(55, 125, 140, 25);

LoginMainWindow.add(btnLogin);

LoginMainWindow.add(btnCancel);

}

//------------------------------------------------------------------------------

/\*\*

\* This method adds all the ActionListeners to the various controls within the

\* LoginWindow.

\*/

public static void LoginWindow\_Action()

{

btnLogin.addActionListener(new java.awt.event.ActionListener() {

@Override

public void actionPerformed(java.awt.event.ActionEvent evt)

{

if (ACTION\_LOGINCHECK(txtUsername.getText().trim(), txtPassword.getText().trim()))

{

LoginMainWindow.setVisible(false);

txtPassword.setText(null);

txtUsername.setText(null);

txtUsername.requestFocus();

AdminMainWindow.setVisible(true);

}

else

{

JOptionPane.showMessageDialog(null, "Username and Password\ncombination are incorrect!", "Alert!", 2);

txtUsername.setText("");

txtPassword.setText("");

txtUsername.requestFocus();

}

}

});

btnCancel.addActionListener(new java.awt.event.ActionListener() {

@Override

public void actionPerformed(java.awt.event.ActionEvent evt)

{

ClientMainWindow.setVisible(true);

LoginMainWindow.setVisible(false);

}

});

LoginMainWindow.addWindowListener(new java.awt.event.WindowAdapter(){

@Override

public void windowClosing(java.awt.event.WindowEvent windowEvent)

{

ACTION\_CLOSE(LoginMainWindow);

}

});

}

//------------------------------------------------------------------------------

/\*\*

\* This method sets the AdminWindow GUI's title, sets the defaultCloseOperation,

\* and sets the window's resizable property to false. Followed by calling the AdminWindow\_Action() method,

\* ConfigureAdminWindow() method and ACTION\_SET\_MOVIECBX() and ACTION\_SET\_GENRECBX() methods.

\* Thus building the AdminWindow.

\*/

public static void BuildAdminMainWindow()

{

AdminMainWindow.setTitle("VideoStore: Administrator");

AdminMainWindow.setDefaultCloseOperation(JFrame.DO\_NOTHING\_ON\_CLOSE);

AdminMainWindow.setResizable(false);

ConfigureAdminWindow();

AdminWindow\_Action();

ACTION\_SET\_CBX("Movie");

ACTION\_SET\_CBX("Genre");

}

//------------------------------------------------------------------------------

/\*\*

\*This method configures the AdminWindow GUI, setting the position of

\* all controls.

\*/

public static void ConfigureAdminWindow()

{

AdminMainWindow.setBackground(new java.awt.Color(255, 255, 255));

AdminMainWindow.setSize(620, 370);

AdminMainWindow.setLocation(220,180);

AdminMainWindow.getContentPane().setLayout(null);

AdminMainWindow.setJMenuBar(AdminMenuBar);

AdminMenuBar.setVisible(true);

btnAdminLogout.setText("Logout");

AdminMainMenu.setText("Admin");

AdminMainMenu.add(btnAdminLogout);

AdminMenuBar.add(AdminMainMenu);

btnAdminHelp.setText("Help");

btnAdminAbout.setText("About");

AdminInfoMenu.setText("Info");

AdminInfoMenu.add(btnAdminAbout);

AdminInfoMenu.add(btnAdminHelp);

AdminMenuBar.add(AdminInfoMenu);

pnlMovieInfo.setBounds(10, 50, 595, 255);

pnlMovieInfo.setBorder(BorderFactory.createEtchedBorder());

pnlMovieInfo.setLayout(null);

pnlMovieInfo.setVisible(true);

lblAllMovie.setText("All Movies:");

lblAllMovie.setBounds(10, 10, 80, 25);

cbxMovieList.setBounds(80, 10, 395, 25);

btnDeleteMov.setBounds(490, 10, 90, 25);

btnDeleteMov.setText("Delete");

pnlMovieInfo.add(lblAllMovie);

pnlMovieInfo.add(cbxMovieList);

pnlMovieInfo.add(btnDeleteMov);

sepMovie.setBounds(0, 45, 593, 1);

sepMovie.setForeground(new java.awt.Color(166, 166, 166));

pnlMovieInfo.add(sepMovie);

lblMovieTitle.setBounds(10, 55, 100, 25);

lblMovieTitle.setText("Movie Title:");

lblMovieGenre.setBounds(10, 90, 100, 25);

lblMovieGenre.setText("Genre:");

lblMovieDesc.setBounds(10, 120, 150, 25);

lblMovieDesc.setText("Movie Description:");

pnlMovieInfo.add(lblMovieTitle);

pnlMovieInfo.add(lblMovieDesc);

pnlMovieInfo.add(lblMovieGenre);

txtMovieTitle.setBounds(80, 55, 200, 25);

cbxMovieGenre.setBounds(80, 90, 200, 25);

spMovieDesc.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL\_SCROLLBAR\_NEVER);

spMovieDesc.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL\_SCROLLBAR\_ALWAYS);

spMovieDesc.setViewportView(taMovieDesc);

pnlSearchControls.add(spMovieDesc);

spMovieDesc.setBounds(10, 145, 570, 80);

btnAddMov.setBounds(480, 235, 100, 25);

btnAddMov.setText("Add");

pnlMovieInfo.add(btnAddMov);

pnlMovieInfo.add(txtMovieTitle);

pnlMovieInfo.add(cbxMovieGenre);

pnlMovieInfo.add(spMovieDesc);

pnlGenreInfo.setBounds(10, 50, 595, 255);

pnlGenreInfo.setBorder(BorderFactory.createEtchedBorder());

pnlGenreInfo.setLayout(null);

pnlGenreInfo.setVisible(true);

lblAllGenre.setText("All Genres:");

lblAllGenre.setBounds(10, 10, 80, 25);

cbxGenreList.setBounds(80, 10, 502, 25);

// btnDeleteGen.setBounds(490, 10, 90, 25);

// btnDeleteGen.setText("Delete");

pnlGenreInfo.add(cbxGenreList);

// pnlGenreInfo.add(btnDeleteGen);

//

sepGenre.setBounds(0, 45, 593, 1);

sepGenre.setForeground(new java.awt.Color(166, 166, 166));

pnlGenreInfo.add(lblAllGenre);

pnlGenreInfo.add(sepGenre);

lblGenre.setBounds(10, 55, 100, 25);

lblGenre.setText("Genre:");

txtGenre.setBounds(80, 55, 150, 25);

pnlGenreInfo.add(lblGenre);

pnlGenreInfo.add(txtGenre);

btnAddGen.setBounds(480, 235, 100, 25);

btnAddGen.setText("Add");

pnlGenreInfo.add(btnAddGen);

tpAdminPnl.addTab("Movie", pnlMovieInfo);

tpAdminPnl.addTab("Genre", pnlGenreInfo);

tpAdminPnl.setVisible(true);

tpAdminPnl.setBounds(7, 5, 600, 305);

AdminMainWindow.getContentPane().add(tpAdminPnl);

}

//------------------------------------------------------------------------------

/\*\*

\* This method adds all the ActionListeners to the various controls within

\* the AdminWindow.

\*/

public static void AdminWindow\_Action()

{

btnAdminLogout.addActionListener(new java.awt.event.ActionListener() {

@Override

public void actionPerformed(java.awt.event.ActionEvent evt)

{

AdminMainWindow.setVisible(false);

txtMovieTitle.setText(null);

taMovieDesc.setText(null);

txtGenre.setText(null);

ClientMainWindow.setVisible(true);

}

});

btnAdminAbout.addActionListener(new java.awt.event.ActionListener() {

@Override

public void actionPerformed(java.awt.event.ActionEvent evt)

{

ACTION\_ABOUT();

}

});

btnAdminHelp.addActionListener(new java.awt.event.ActionListener() {

@Override

public void actionPerformed(java.awt.event.ActionEvent evt)

{

ADMIN\_HELP();

}

});

btnAddMov.addActionListener(new java.awt.event.ActionListener() {

@Override

public void actionPerformed(java.awt.event.ActionEvent evt)

{

String tempRegEx = "\\s([A-Z][a-z]+|[0-9]+)";

String defaultRegEx = "[A-Z][a-z]+";

String numRegEx = "[0-9]+";

//String descRegEx = "(^).\*?(?=\\n|$)"; - Initially used to check Movie Description.

//if (txtMovieTitle.getText().matches("\\d.\*")) - Used to check if string started with a number.

// System.out.println("Contains number!");

for (int x = 0; x < txtMovieTitle.getText().length(); x++)

{

if (txtMovieTitle.getText().charAt(x) == ' ')

{

//System.out.println("Space found at: " + x);

defaultRegEx += tempRegEx;

numRegEx += tempRegEx;

}

}

boolean movTitleCheck = ((Pattern.matches(defaultRegEx, txtMovieTitle.getText().trim()) || (Pattern.matches(numRegEx, txtMovieTitle.getText().trim()))));

//System.out.println("Matched Pattern for Title: " + movTitleCheck);

if ((taMovieDesc.getText().length() > 15))

{

if (movTitleCheck)

{

//System.out.println("Matched Pattern: " + txtMovieTitle.getText().trim());

String temp = txtMovieTitle.getText().trim() + "|" +

taMovieDesc.getText().trim() + "|" +

cbxMovieGenre.getSelectedItem().toString().trim();

//System.out.println(temp);

boolean insertCheck = ACTION\_INSERT(temp);

if (insertCheck)

{

//System.out.println(insertCheck);

ACTION\_SET\_CBX("Movie");

JOptionPane.showMessageDialog(null, "Added Movie \"" + txtMovieTitle.getText() + "\" !", "Alert!", 1);

txtMovieTitle.setText(null);

taMovieDesc.setText(null);

}

else

{

JOptionPane.showMessageDialog(null, "Adding Movie \"" + txtMovieTitle.getText() + "\" was not successful!\nPlease check that the Movie doesn't already exist and that the genre exists!", "Alert!", 2);

txtMovieTitle.setText(null);

taMovieDesc.setText(null);

}

}

else

{

JOptionPane.showMessageDialog(null, "Movie Title doesn't start with a capital letter! OR\neach word in the movie title isn't capitalised!", "ALERT!", 2);

}

}

else

{

JOptionPane.showMessageDialog(null, "Description is not greater than 15 characters!", "ALERT!", 2);

}

}

});

btnAddGen.addActionListener(new java.awt.event.ActionListener() {

@Override

public void actionPerformed(java.awt.event.ActionEvent evt)

{

String temp = txtGenre.getText().trim();

boolean genreCheck = Pattern.matches("[A-Z][a-z]+", temp);

//System.out.println(temp);

if (genreCheck)

{

boolean insertCheck = ACTION\_INSERT(temp);

if (insertCheck)

{

//System.out.println(insertCheck);

JOptionPane.showMessageDialog(null, "Added Genre \"" + txtGenre.getText() + "\" !", "Alert!", 1);

txtGenre.setText(null);

ACTION\_SET\_CBX("Genre");

}

else

{

JOptionPane.showMessageDialog(null, "Adding Genre \"" + txtGenre.getText() + "\" was\nnot successful!", "Alert!", 2);

txtGenre.setText(null);

}

}

else

{

JOptionPane.showMessageDialog(null, "Please Check that the Genre Name is capitalised\nand that there are no spaces!", "Alert!", 2);

}

}

});

btnDeleteMov.addActionListener(new java.awt.event.ActionListener() {

@Override

public void actionPerformed(java.awt.event.ActionEvent evt)

{

cbxMovieList.getSelectedItem();

if (ACTION\_DELETE(cbxMovieList.getSelectedItem().toString()))

{

JOptionPane.showMessageDialog(null, "Movie: \"" + cbxMovieList.getSelectedItem().toString() + "\" \nwas deleted!","ALERT!", 2);

ACTION\_SET\_CBX("Movie");

}

else

{

JOptionPane.showMessageDialog(null, "Unable to DELETE Movie: \n\"" + cbxMovieList.getSelectedItem().toString() + "\"!","ALERT!", 2);

}

}

});

AdminMainWindow.addWindowListener(new java.awt.event.WindowAdapter(){

@Override

public void windowClosing(java.awt.event.WindowEvent windowEvent)

{

ACTION\_CLOSE(AdminMainWindow);

}

});

}

//------------------------------------------------------------------------------

/\*\*

\* This method accepts a JFrame as a parameter and based on what the user chose

\* from the JOptionPane.showOptionDialog, it will either minimize,

\* or exit the program.

\* @param frame

\*/

public static void ACTION\_CLOSE(JFrame frame)

{

String[] closingOptions = {"Minimize", "Exit", "Cancel"};

int response = JOptionPane.showOptionDialog(null,

"Are you sure you want to exit?", "Exit?", 0,

JOptionPane.WARNING\_MESSAGE, null, closingOptions, closingOptions[1]);

if (response == 0)

{

frame.setState(frame.ICONIFIED);

}

else if (response == 1)

{

System.exit(0);

}

}

//------------------------------------------------------------------------------

/\*\*

\* This method returns an ArrayList of String[] with the movies which were

\* selected from the parameters passed. It accepts the query of what the user

\* is trying to find from the database, populates the ArrayList with String[]

\* of the movie information that matches the query and then returns the ArrayList.

\* @param query

\* @param topic

\* @return ArrayList of String[] containing Movie Information.

\*/

public static ArrayList<String[]> ACTION\_SELECT\_QUERY(String query, String topic)

{

ArrayList<String[]> MovieList = new ArrayList<>();

String queryString;

try

{

String data = "jdbc:odbc:VideoStoreDb";

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Connection conn = DriverManager.getConnection(data, "", "");

Statement st = conn.createStatement();

if (topic.equals("Movie"))

{

queryString = "SELECT movie\_name, movie\_description, genre\_name FROM Movie, Genre WHERE Movie.genre\_id = Genre.genre\_id AND movie\_name LIKE '%" + query + "%';";

}

else

{

queryString = "SELECT movie\_name, movie\_description, genre\_name FROM Movie, Genre WHERE Genre.genre\_id = Movie.genre\_id AND genre\_name LIKE '%" + query + "%';";

}

ResultSet rec = st.executeQuery(queryString);

while (rec.next())

{

//System.out.println(rec.getString(1) + "\t" + rec.getString(2) + "\t" + rec.getString(3));

String[] tempArr = new String[3];

tempArr[0] = rec.getString(1); tempArr[1] = rec.getString(2); tempArr[2] = rec.getString(3);

MovieList.add(tempArr);

}

st.close();

return MovieList;

}

catch (SQLException E)

{

System.out.println(E);

}

catch(Exception X)

{

System.out.println(X);

}

return null;

}

//------------------------------------------------------------------------------

/\*\*

\* This method accepts a single string parameter, to determine whether

\* the user is searching for a movie or a specific genre. It then will call

\* the ACTION\_SELECT\_QUERY() method, and assigns the returned value to an

\* ArrayList, it then uses that ArrayList to populate the JTable and update

\* the JTable to show the result of the query, which the user entered.

\*IF there are no results the method shows a JOptionPane.showMessageDialog,

\* to the user to display that there are no movies, or genres matching the query.

\* @param topic

\*/

public static void ACTION\_SEARCH(String topic)

{

String userQuery = txtSearch.getText().trim();

//System.out.println("get string");

ArrayList<String[]> MovieList = ACTION\_SELECT\_QUERY(userQuery, topic);

if (MovieList.size() > 0)

{

//System.out.println("arraylist create");

int rowCount = MovieList.size();

//System.out.println("int create");

String[][] MovieItems = new String[rowCount][3];

//System.out.println("2d Array Create");

for (int i = 0; i < MovieList.size(); i++)

{

String[] movieInfoArr = MovieList.get(i);

for (int j = 0; j < movieInfoArr.length; j++)

{

MovieItems[i][j] = movieInfoArr[j];

//System.out.println(movieInfoArr[j]);

}

}

//System.out.println("after for loops");

dtResults = new JTable(MovieItems, tableCols);

//System.out.println("after jtable");

spResults.getViewport().add(dtResults);

spResults.setViewportView(dtResults);

}

else

{

dtResults = new JTable();

spResults.getViewport().add(dtResults);

spResults.setViewportView(dtResults);

if (topic.equals("Movie"))

{

JOptionPane.showMessageDialog(null, "No Movie Results found for : \"" + txtSearch.getText() + "\"","ALERT!", 2);

}

else

{

JOptionPane.showMessageDialog(null, "No Genre Results found for : \"" + txtSearch.getText() + "\"","ALERT!", 2);

}

}

}

//------------------------------------------------------------------------------

/\*\*

\* This method accepts two parameters, the user's username and password from

\* the username and password textboxes on the LoginWindow, and checks these

\* values against the database and compares if the information given is correct.

\*

\* If the information is correct, it will return true.

\* @param username

\* @param password

\* @return true if matches the username and password given, otherwise false.

\*/

public static boolean ACTION\_LOGINCHECK(String username, String password)

{

try

{

String[] tempArr = new String[2];

String data = "jdbc:odbc:VideoStoreDb";

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Connection conn = DriverManager.getConnection(data, "", "");

Statement st = conn.createStatement();

ResultSet rec = st.executeQuery("SELECT user\_name, user\_password FROM User\_Accounts");

while (rec.next())

{

//System.out.println(rec.getString(1) + "\t" + rec.getString(2) + "\t" + rec.getString(3));

tempArr[0] = rec.getString(1); tempArr[1] = rec.getString(2);

}

//System.out.println(tempArr[0] + "|" + tempArr[1]);

st.close();

if (tempArr[0].equals(username))

{

return tempArr[1].equals(password);

}

}

catch (SQLException E)

{

System.out.println(E);

}

catch(Exception X)

{

System.out.println(X);

}

return false;

}

//------------------------------------------------------------------------------

/\*\*

\* This method accepts a single String parameter to determine what JCombobox to

\* populate, if 'Movie' is passed, it populates the Movie JComobobox and if

\* 'Genre' is passed, it populates the Genre JComboboxes.

\* @param topic

\*/

public static void ACTION\_SET\_CBX(String topic)

{

ArrayList<String> List = new ArrayList<>();

String queryString;

try

{

if (topic.equals("Movie"))

{

cbxMovieList.removeAllItems();

queryString = "SELECT movie\_name,movie\_description FROM Movie";

}

else

{

cbxGenreList.removeAllItems();

cbxMovieGenre.removeAllItems();

queryString = "SELECT genre\_name FROM Genre";

}

String data = "jdbc:odbc:VideoStoreDb";

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Connection conn = DriverManager.getConnection(data, "", "");

Statement st = conn.createStatement();

ResultSet rec = st.executeQuery(queryString);

while (rec.next())

{

//System.out.println(rec.getString(1) + "\t" + rec.getString(2) + "\t" + rec.getString(3));

if (topic.equals("Movie"))

{

List.add(rec.getString(1) + " - " + rec.getString(2).substring(0, 10) + "...");

}

else

{

List.add(rec.getString(1));

}

}

st.close();

if (List.size() > 0)

{

if (topic.equals("Movie"))

{

for (String movie : List)

{

cbxMovieList.addItem(movie);

cbxMovieList.setEnabled(true);

btnDeleteMov.setEnabled(true);

}

}

else

{

for (String genre : List)

{

cbxGenreList.addItem(genre);

cbxMovieGenre.addItem(genre);

cbxGenreList.setEnabled(true);

cbxMovieGenre.setEnabled(true);

}

}

}

}

catch (SQLException E)

{

System.out.println(E);

}

catch(Exception X)

{

System.out.println(X);

}

}

//------------------------------------------------------------------------------

/\*\*

\* This method accepts a single parameter to determine which table in the database

\* it must insert into, if 'Movie' is passed, it inserts into the Movie Table,

\* and if 'Genre' is passed, it inserts into the Genre Table.

\* @param topicInfo

\* @return true if insert is successful, otherwise false;

\*/

public static boolean ACTION\_INSERT(String topicInfo)

{

int rowsAdded;

String genre;

String queryString;

try

{

//System.out.println(topicInfo);

if (topicInfo.contains("|"))

{

//System.out.println("Before split");

String[] movie = topicInfo.split("\\|");

//System.out.println("After split: " + movie[0] + "|" + movie[1] + "|" + movie[2]);

if (!CheckMovie(movie[0] + "|" + movie[1]))

{

//System.out.println("CheckGenre: True");

String tempGenre = getGenreID(movie[2]);

//System.out.println("Method got ID");

//System.out.println(tempGenre);

//System.out.println(movie[0] + "|" + movie[1] + "|" + tempGenre);

queryString = "INSERT INTO Movie(movie\_name, movie\_description, genre\_id) VALUES('" + movie[0] + "', '" + movie[1] + "', '" + tempGenre + "')";

}

else

{

return false;

}

}

else

{

genre = topicInfo;

if (!CheckGenre(genre))

{

//System.out.println("GenreCheck was False");

queryString = "INSERT INTO Genre(genre\_name) VALUES( '" + genre + "')";

}

else

{

return false;

}

}

String data = "jdbc:odbc:VideoStoreDb";

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Connection conn = DriverManager.getConnection(data, "", "");

Statement st = conn.createStatement();

rowsAdded = st.executeUpdate(queryString);

//System.out.println("After SQL INSERT");

st.close();

return true;

}

catch (SQLException E)

{

System.out.println(E);

}

catch(Exception X)

{

System.out.println(X);

}

return false;

}

//------------------------------------------------------------------------------

/\*\*

\* This method accepts a single String parameter which contains the movie information,

\* allowing that specified movie to be deleted, and removed from the Database

\* To ensure that the correct movie is deleted, it calls the getMovieInfo() method

\* before deleting, to ensure that it is, in fact, the correct movie it should delete.

\* @param MovieInfo

\* @return true if delete was successful, otherwise false.

\*/

public static boolean ACTION\_DELETE(String MovieInfo)

{

int rowsAdded;

String queryString;

try

{

//System.out.println(MovieInfo);

String tempMovieInfo = getMovieInfo(MovieInfo);

if (tempMovieInfo.contains("|"))

{

//System.out.println("Before split");

String[] movie = tempMovieInfo.split("\\|");

//System.out.println("After split: " + movie[0] + "|" + movie[1] + "|" + movie[2]);

queryString = "DELETE FROM Movie WHERE movie\_name = '" + movie[0] + "' AND movie\_description = '" + movie[1] + "';";

}

else

{

return false;

}

String data = "jdbc:odbc:VideoStoreDb";

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Connection conn = DriverManager.getConnection(data, "", "");

Statement st = conn.createStatement();

rowsAdded = st.executeUpdate(queryString);

//System.out.println("After SQL DELETE");

st.close();

return true;

}

catch (SQLException E)

{

System.out.println(E);

}

catch(Exception X)

{

System.out.println(X);

}

return false;

}

//------------------------------------------------------------------------------

/\*\*

\* This method accepts a single String parameter, which contains movie information

\* it then finds the movie specified and returns the Movie Name and Full Description.

\* @param Movie

\* @return Movie information in a single String.

\*/

public static String getMovieInfo(String Movie)

{

String[] temp = Movie.split(" - ");

//System.out.println(temp[0] + "|" + temp[1]);

ArrayList<String> tempResults = new ArrayList<>();

try

{

String data = "jdbc:odbc:VideoStoreDb";

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Connection conn = DriverManager.getConnection(data, "", "");

Statement st = conn.createStatement();

ResultSet rec = st.executeQuery("SELECT movie\_name,movie\_description FROM Movie WHERE movie\_name = '" + temp[0] + "' AND movie\_description LIKE '" + temp[1].substring(0, temp[1].length()-3) + "%';");

if (rec.next())

{

//System.out.println(rec.getString(1) + "\t" + rec.getString(2) + "\t" + rec.getString(3));

tempResults.add(rec.getString(1) + "|" + rec.getString(2));

}

st.close();

//System.out.println("ArrSize: " + tempResults.size());

if (tempResults.size() == 1)

return tempResults.get(0);

return null;

}

catch (SQLException E)

{

System.out.println(E);

}

catch(Exception X)

{

System.out.println(X);

}

return null;

}

//------------------------------------------------------------------------------

/\*\*

\* This method accepts a single String parameter, which contains the movie information,

\* it uses this information to check in the database that no other information for this movie

\* exists, if so it returns true. Otherwise false;

\* @param movieInfo

\* @return true if movie exists, otherwise false.

\*/

public static boolean CheckMovie(String movieInfo)

{

String movieCompare = "";

try

{

String[] tempMovieInfo = movieInfo.split("\\|");

String data = "jdbc:odbc:VideoStoreDb";

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Connection conn = DriverManager.getConnection(data, "", "");

Statement st = conn.createStatement();

ResultSet rec = st.executeQuery("SELECT movie\_name, movie\_description FROM Movie WHERE movie\_name = '" + tempMovieInfo[0] + "' AND movie\_description = '" + tempMovieInfo[1] + "';");

while (rec.next())

{

//System.out.println(rec.getString(1) + "\t" + rec.getString(2) + "\t" + rec.getString(3));

movieCompare = rec.getString(1) + "|" + rec.getString(2);

}

st.close();

//System.out.println(movieInfo.toUpperCase());

//System.out.println(movieCompare.toUpperCase());

return movieInfo.toUpperCase().equals(movieCompare.toUpperCase());

}

catch (SQLException E)

{

System.out.println(E);

}

catch(Exception X)

{

System.out.println(X);

}

return false;

}

//------------------------------------------------------------------------------

/\*\*

\* This method accepts a single String parameter, and returns a string parameter.

\* This method is used to get the GenreID for that specified Genre.

\* @param genre

\* @return Genre ID

\*/

public static String getGenreID(String genre)

{

try

{

String data = "jdbc:odbc:VideoStoreDb";

String genreID = "";

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Connection conn = DriverManager.getConnection(data, "", "");

Statement st = conn.createStatement();

ResultSet rec = st.executeQuery("SELECT genre\_id FROM Genre WHERE genre\_name = '" + genre + "';");

while (rec.next())

{

//System.out.println(rec.getString(1) + "\t" + rec.getString(2) + "\t" + rec.getString(3));

genreID = rec.getString(1);

}

st.close();

//System.out.println(genreID);

return genreID;

}

catch (SQLException E)

{

System.out.println(E);

}

catch(Exception X)

{

System.out.println(X);

}

return null;

}

//------------------------------------------------------------------------------

/\*\*

\* This method accepts a single String parameter, which is the Genre Name, it then

\* uses this to check if it exists within the database, and if it does returns false.

\* @param genre

\* @return true if genre already exists.

\*/

public static boolean CheckGenre(String genre)

{

try

{

String data = "jdbc:odbc:VideoStoreDb";

String tempGenre = "";

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Connection conn = DriverManager.getConnection(data, "", "");

Statement st = conn.createStatement();

ResultSet rec = st.executeQuery("SELECT genre\_name FROM Genre WHERE genre\_name = '" + genre + "';");

while (rec.next())

{

//System.out.println(rec.getString(1) + "\t" + rec.getString(2) + "\t" + rec.getString(3));

tempGenre = rec.getString(1);

}

st.close();

System.out.println(tempGenre + "|" + genre);

return genre.toUpperCase().equals(tempGenre.toUpperCase());

}

catch (SQLException E)

{

System.out.println(E);

}

catch(Exception X)

{

System.out.println(X);

}

return false;

}

//------------------------------------------------------------------------------

/\*\*

\* This method displays a JOptionPane.showMessageDialog, detailing who created the program

\* as well as when it was created.

\*/

public static void ACTION\_ABOUT()

{

JOptionPane.showMessageDialog(null, "Author: Donovan van Heerden\nEL2014-0043\nDate Created: 20-01-2014", "About", 1);

}

//------------------------------------------------------------------------------

/\*\*

\* This method displays a JOptionPane.showMessageDialog, detailing how a user may search

\* for a movie with a specified name or movies with similar names, otherwise movies with

\* certain genres.

\*/

public static void BROWSE\_HELP()

{

JOptionPane.showMessageDialog(null, "Searching Database:\n----------------------------------------------------------------------------------------------------\n" +

"Users may enter anything into the search textbox provided, " +

"the program\nwill then search the database to partially match what was typed into the\nsearch textbox and show the results below.\n\n" +

"If the movie or genre specified cannot be found a message will appear.", "Browse Help", 1);

}

//------------------------------------------------------------------------------

/\*\*

\* This method displays a JOptionPane.showMessageDialog, detailing how an Administrator, may

\* add a new Movie, Genre.

\*/

public static void ADMIN\_HELP()

{

JOptionPane.showMessageDialog(null, "Movie Section:\n--------------------------------------------------------------------\n" +

"1. Movie Title must start with a capital letter.\n" +

"2. If the Movie Title starts with a number, and has a space following the number, the next word must be capitalised.\n" +

"3. If the Movie Title has mutliple spaces and words, each word in the title must be capitalised.\n" +

"4. The Movie Description must be 15 characters in length or longer.\n\n" +

"Examples: 9, The Hobbit, 28 Days Later, Super 8\n\n" +

"Genre Section:\n--------------------------------------------------------------------\n" +

"1. The Genre Name must start with a capital letter.\n" +

"2. No spaces are allowed for the Genre Name.\n\n" +

"Examples: Comedy, Romance, Family", "Admin Help", 1);

}

}

**User Documentation:**

**Author:** Donovan van Heerden

**Student No:** EL2014-0043

**Date:** 20/01/2014

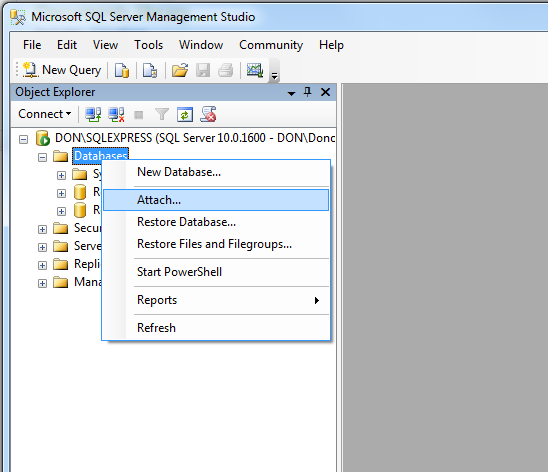
**Lecturer:** Jason Smith

**Campus:** CTI East London

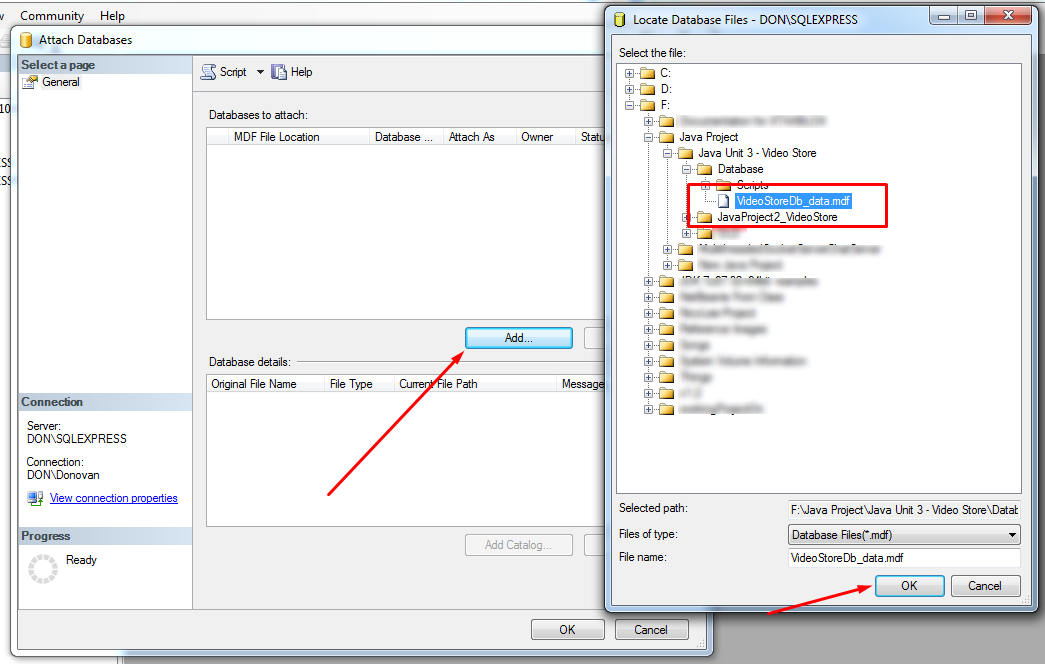
The purpose of this project is to create a java application for users to browse a video/movie database, and for an administrator to add videos/movies or remove videos/movies, as well as add new genres. In other words, the administrator is able to manage the database from the “backend” of the java application by means of a login system.

**How to add the Database to MS SQL Server 2008:**

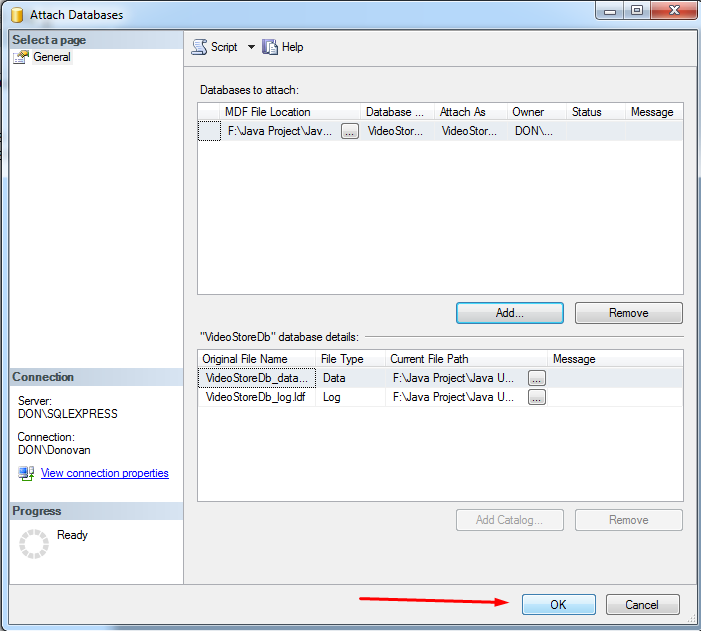
* Open Microsoft SQL Server 2008
* Right click on the Database folder and select **Attach…**



* A new window should appear, click on **Add…** Another new window should appear, now locate the database **.mdf** file and select it, followed by clicking the **OK** button.



* The previous window should now have some information in it, similar to the image below, now select **OK.**



* You should now see the database has been added to the Databases Folder in MS SQL Server.

**How to setup the MS SQL Server 2008 Database:**

Go to **Start > Control Panel > Administrative Tools > ODBC Data Source**.

• Select the **System DSN** tab at the top.

• Click the **Add** button to the right.

• In the list box, select **SQL Server Native Client 10.0** and then click **Finish**.

• Enter **“VideoStoreDb”** in the **Name** field, **“Video Store Database”** in the **Description** field, and select the name of your **SQL Server** from the combo box.(See image below for example of where to find this information.) Click the **Next** button.



• Make sure the **With Windows NT authentication……**, and the **Connect to SQL Server ….** items are selected. Click the **Next** button.

• Make sure the **Change the default database to…...** box has a tick in it, and select **“VideoStoreDb”** form the drop-down list. Leave the rest of the checkboxes as they are. Click **Next**.

• Click **Finish**.

• A new frame will be displayed showing the chosen settings. Click the **Test Data Source** button to see if you have set it up correctly. It should say ‘Test Completed successfully’. If not, go through the setup again.

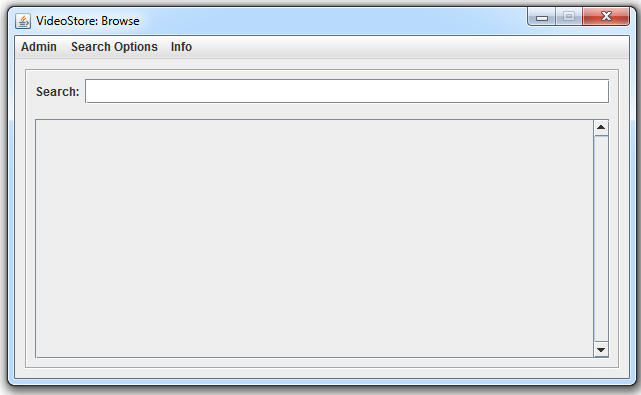
• Click **OK**.

• You should see the name of your database in the **System DSN** tab.

**Application Images & Guidelines:**

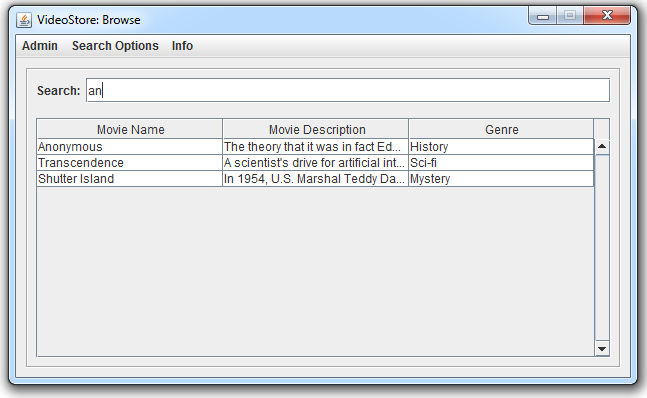
Browsing the VideoStore Application.

Below is the first window you will see when running the application.

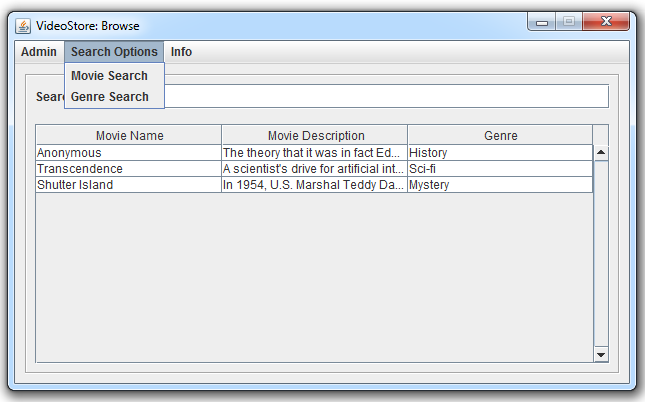


*VideoStore – Browse Window*

A user can type in the **Search textbox** provided, and select from the **Search Options** on the Menubar, **Movie Search** or **Genre Search**, to search according to how they so desire.

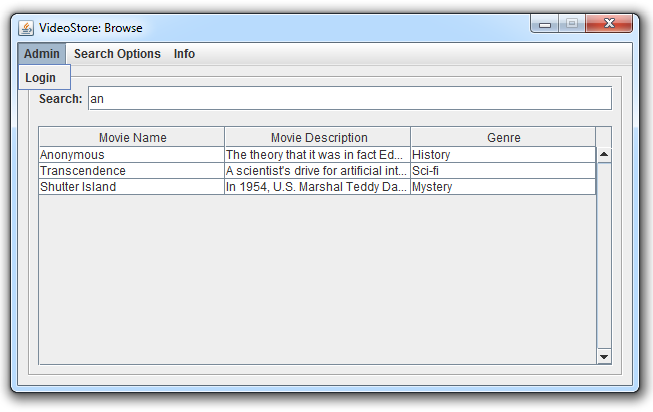


*VideoStore – Browse Window: Searching “an”*

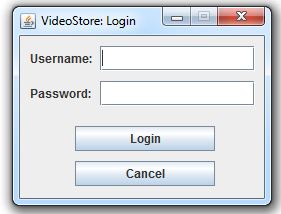


*VideoStore – Browse Window: Search Options*

The **Admin Menu** contains the button called **Admin**, this will take the user to the **Login** **Window**, allowing the Administrator to access the **Administrator** **Window**.

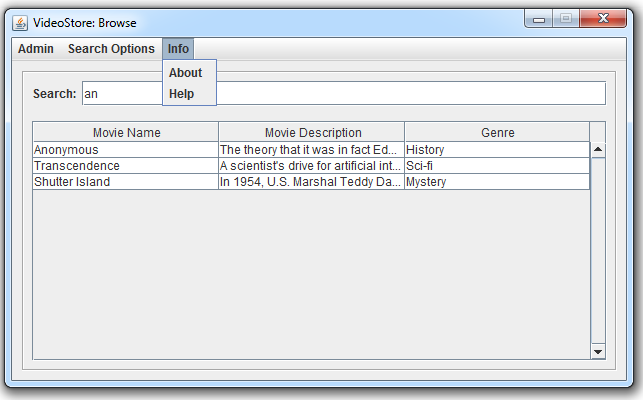


*VideoStore – Browse Window: Admin Option*

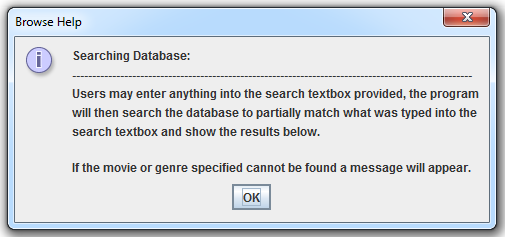


*VideoStore – Login Window*

The **Info** **Menu** contains two buttons, the **About** button and Help button.



*VideoStore – Browse Window: Info Options*

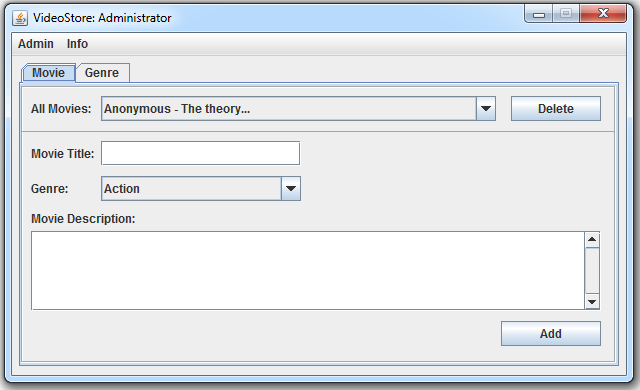


*VideoStore – Browse Help*



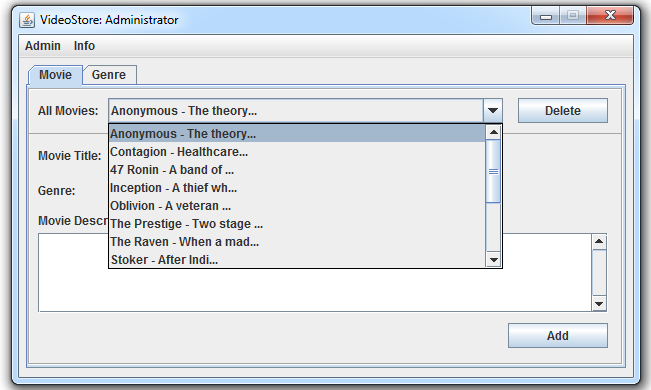
*VideoStore – About*

The Administrator Window appears once the user/administrator has logged in successfully.

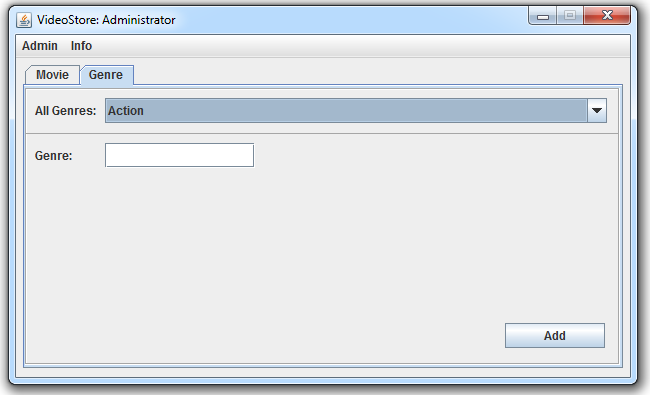


*VideoStore – Administrator Window: Movie Tab*

From here, the Administrator can add new movies to the database, remove movies from the database and add new genre’s allowing new movies of those genres to be added.

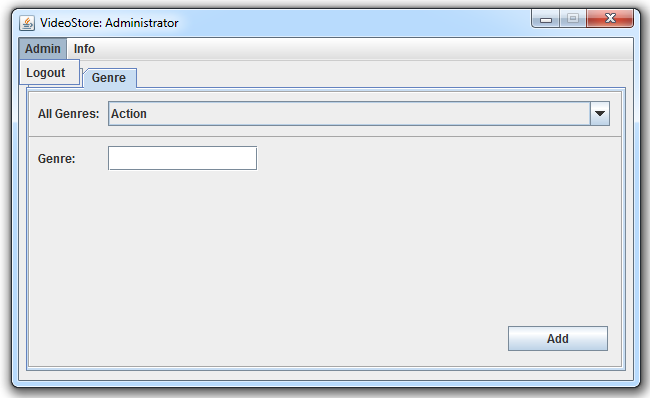


*VideoStore – Administrator Window: Movies List*

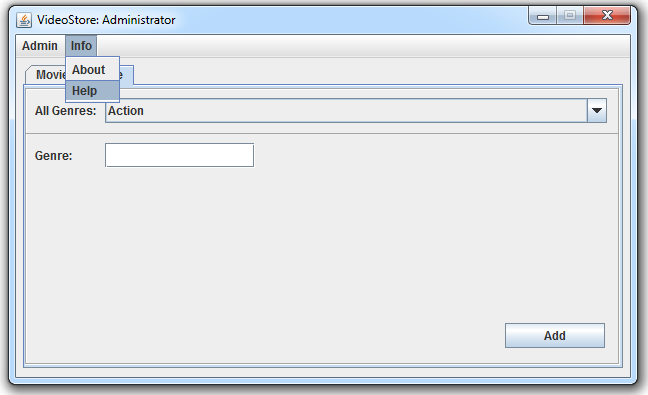


*VideoStore – Administrator Window: Genre Tab*

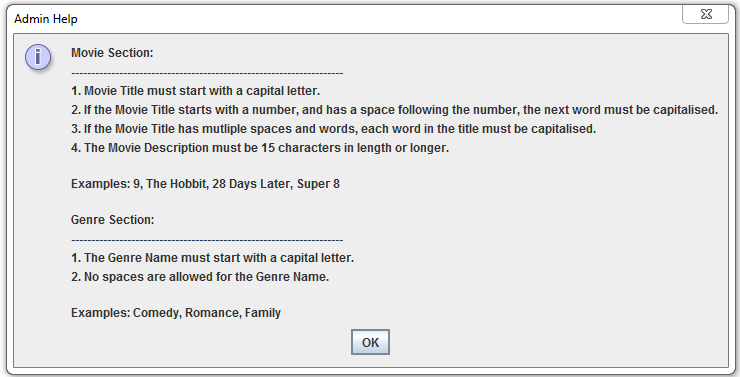
The **Admin Option** in the Menu contains a Logout button and the **Info** Options, contains the **About** button, which displays the same as before, and a **Help** button which displays a Help message for Administrators attempting to add new movies/genres.



*VideoStore – Administrator Window: Admin Option*

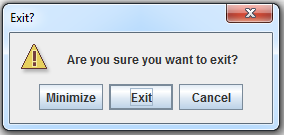


*VideoStore – Administrator Window: Info Option*



*VideoStore – Admin Help*

Upon trying to exit any window by means of pressing the **“X”** the user will get prompted with a message asking if they would like to **Minimize**, **Exit** or **Cancel**.



*VideoStore – Exit Prompt*