CS 250 - Fall 2021 - Lab 05: Javax Swing and Event-Driven Programming

Available: Sep 22, 2021.
Due date: **Sep 27, 2021, 11:59 PM**

Objectives:

The objectives of this lab are:

- To develop a Java GUI application using Swing framework: We will develop an application to manage the courses for students. This application requires logging in before use. In case the student does not have an account, they can register for an account
- To gain additional practice using AWT, Swing components, and layouts.
- To gain additional practice event-driven programming by handling different events on the GUI.

Prerequisite:

- Read section 13 of the free textbook *Java, Java, Java, 3E* and the supplemented slide of lecture 5
- This lab requires you to install the WindowsBuilder plugin for your Eclipse. This plugin allows creating Swing JFrame, JDialog, etc., interactively, so we do not need to code the form by typing the code. Please go to this Install WindowBuilder and Swing Designer (need to scroll down to the third question) to install those packages.
- Read online tutorials of Oracle - Java Swing tutorial
- Read online tutorials of javatpoint - Java Swing tutorial
- Read online tutorials of Event Handling in Java Tutorial

Part 1: Design and Implement the Login Form

1.1. Run the Eclipse IDE program and Create the project

1.1.1 Run the Eclipse IDE program

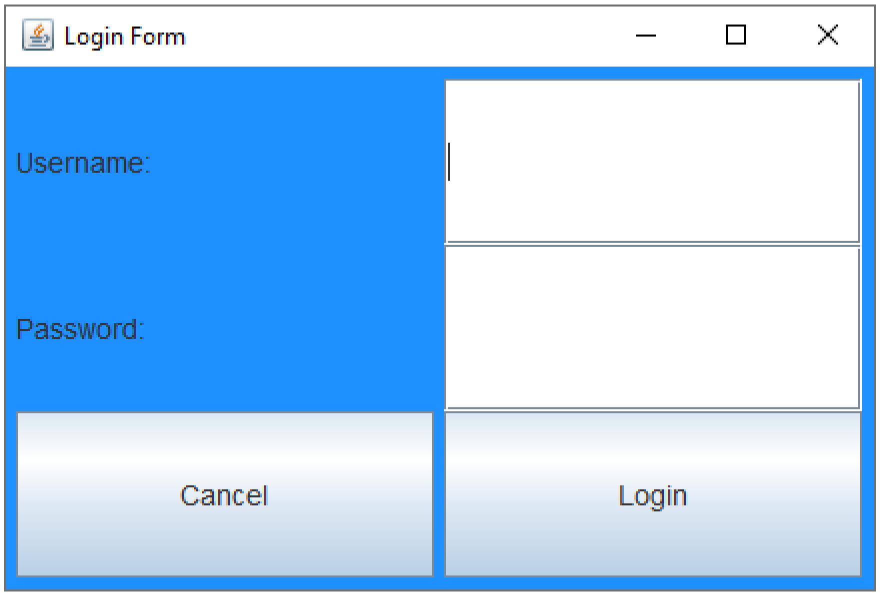
NOTE: Remember how to find this program because you're going to need it every week.

1.1.2. Create the Lab05 project.

- Click on the "Create a java project" link in the Package Explorer window or right-click on that window, select NEW, then Java project. Enter the name of your project, i.e., "Lab05". The "Location" text box should indicate that your project will be stored on the desktop. In the project JRE setting, make sure at least one Java SE version is selected, default to JavaSE-1.8 if you followed instructions on the Prerequisites section. Click on Finish to create the project.

1.2. Design and Create the Login Form

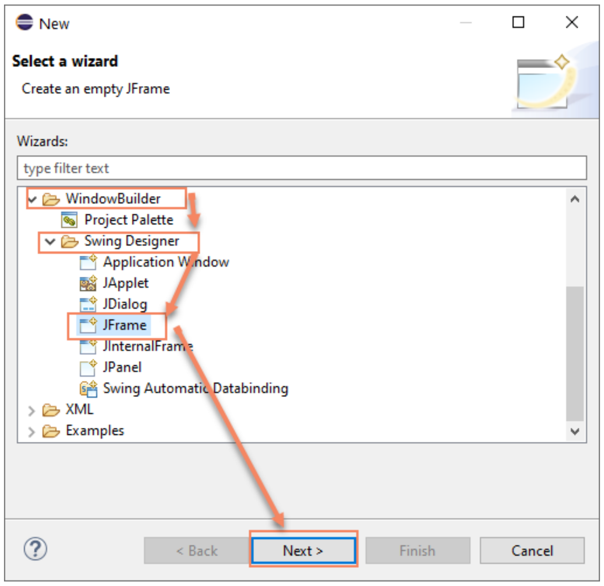
In this part, we will first design and handle the events on the Login Form for the users to log in if there is an existing account or switch to the Register Form to create a new account. The sample of the form is displayed below:



The following steps guide you to design this form interactively using **WindowBuilder**:

1.2.1 Create the Login Form using Swing Designer wizard

First, create a Java project for this Lab 05 on Eclipse. Now, right-click on the `"src"` item of the newly created project on the Package Explorer, select `New`, then `File`, then `Other`. Next, you need to open the `WindowBuilder`, then choose `Swing Designer`, then choose `JFrame`. Finally, click on `Next`. On the next window, enter `LoginForm` on the `Name:` textbox, and then click on `Finish`.


New

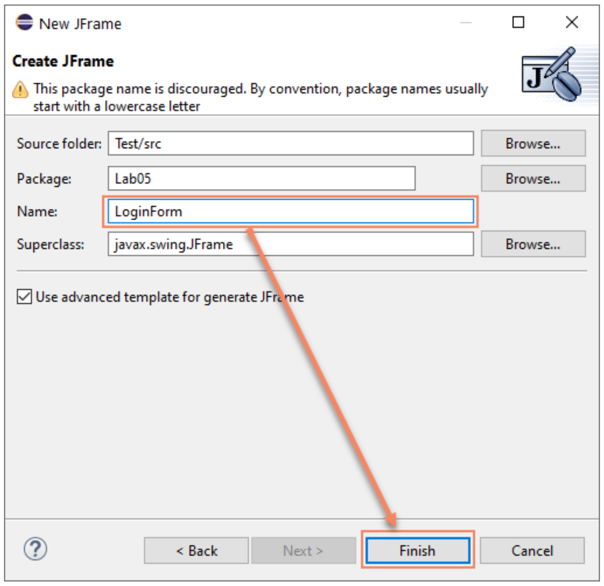
Select a wizard

Create an empty JFrame

Wizards:

type filter text

- WindowBuilder
  - Project Palette
  - Swing Designer
    - Application Window
    - JApplet
    - JDialog
    - JFrame
    - JInternalFrame
    - JPanel
    - Swing Automatic Databinding
- XML
- Examples


New JFrame

Create JFrame

This package name is discouraged. By convention, package names usually start with a lowercase letter

Source folder: Test/src

Package: Lab05

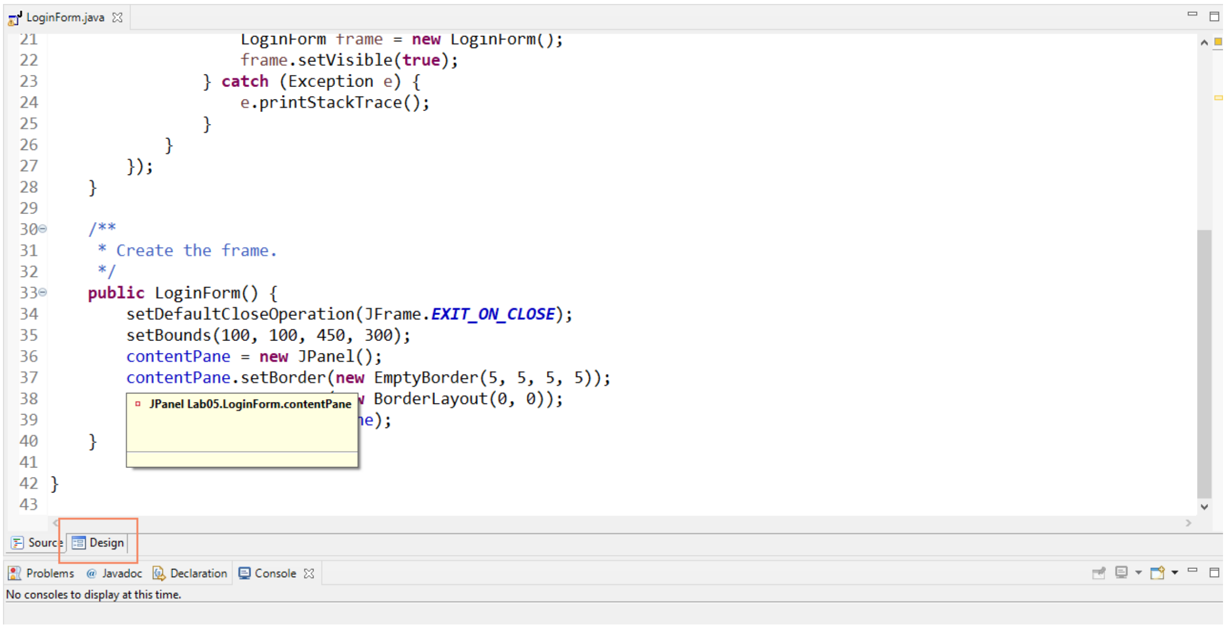
Name: LoginForm

Superclass: javax.swing.JFrame

☑ Use advanced template for generate JFrame

1.2.2. Open Designer View

A new Java class LoginForm is automatically created, and Eclipse opens it for you. Now click on the `Design` on the bottom to open the Designer View of this class.


LoginForm.java

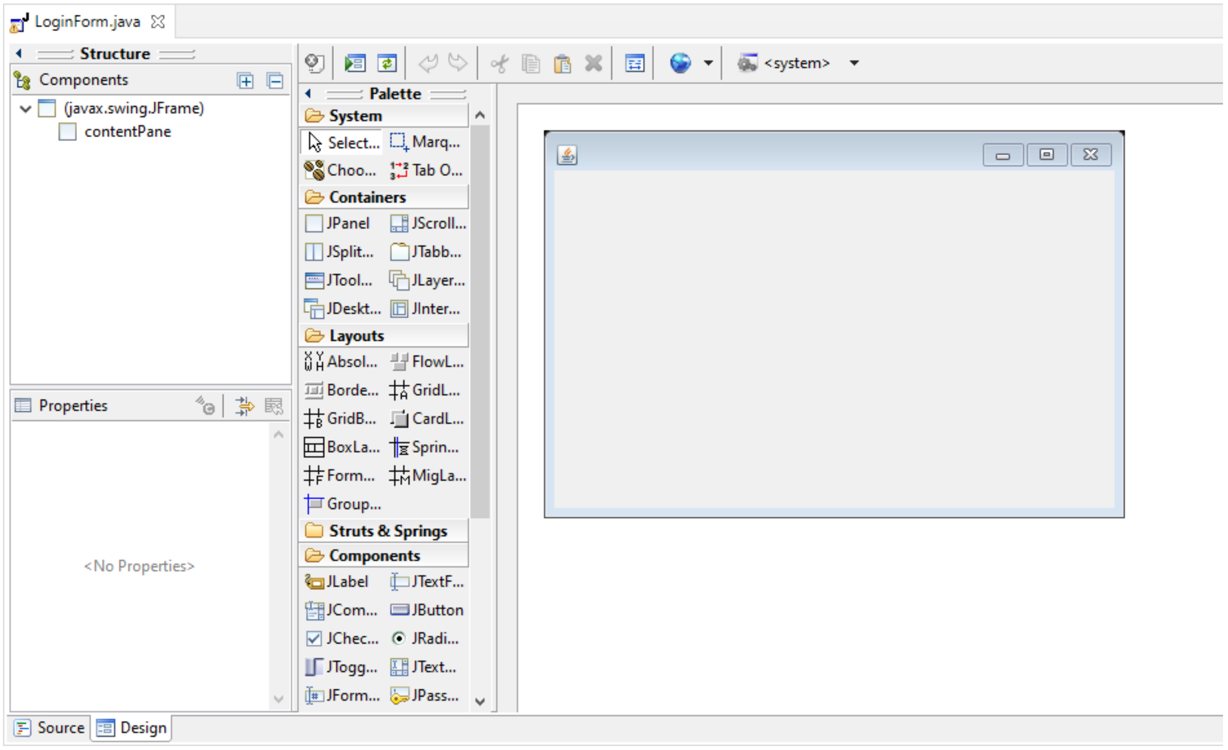
```java
                        LoginForm frame = new LoginForm();
                        frame.setVisible(true);
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
            });
        }

        /**
         * Create the frame.
         */
        public LoginForm() {
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setBounds(100, 100, 450, 300);
            contentPane = new JPanel();
            contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
                                      w BorderLayout(0, 0));
                                      ne);
        }

    }
```

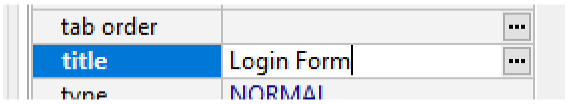
JPanel Lab05.LoginForm.contentPane

No consoles to display at this time.



1.2.3. Change title of the Login Form

Choose the topmost item in the `Components` window. Then enter "Login Form" in the `title` row on the `Properties` subwindow to set/change the title of the Login Form.



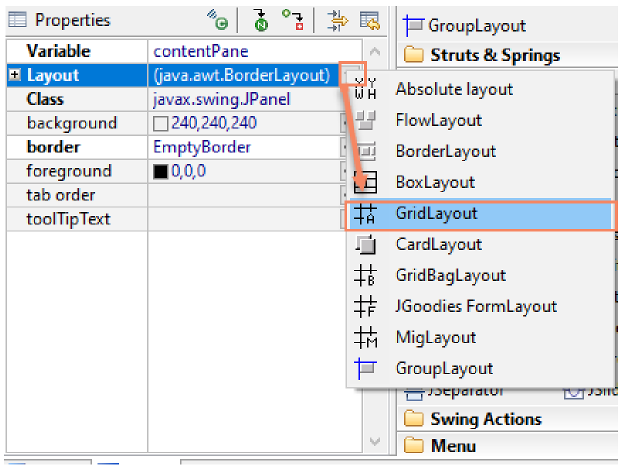
tab order	
title	Login Form
type	NORMAL

1.2.4. Change the layout to GridLayout

Click on the `contentPane` inside the `Components` subwindow. The properties will be loaded on the `Properties` window below. You can see that the Layout is set to `BorderLayout` as default. Now, we need to change the layout to `GridLayout` as follows: click on the small down triangle button next to `BorderLayout` and change to `GridLayout`.

1.2.5. Change the background color of the Login Form

Now, you need to change the background color of the Login Form. Click on the "..." button on the row `background` on the `Properties` window, set with the default color (240, 240, 240). The color picker window will be opened. Then you need to select the tab `Named color` and choose the light blue color as follows:

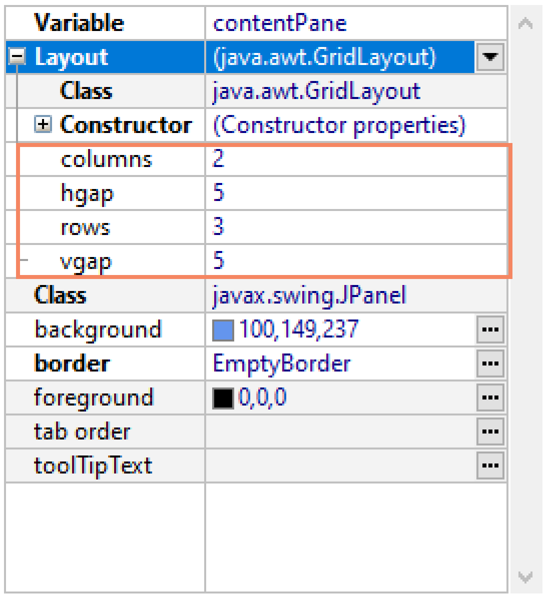


Properties

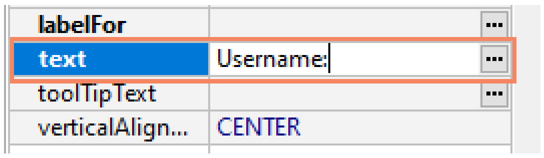
Variable	contentPane
Layout	(java.awt.BorderLayout)
Class	javax.swing.JPanel
background	240,240,240
border	EmptyBorder
foreground	■ 0,0,0
tab order	
toolTipText	

1.2.6. Create the Username label and Username text field

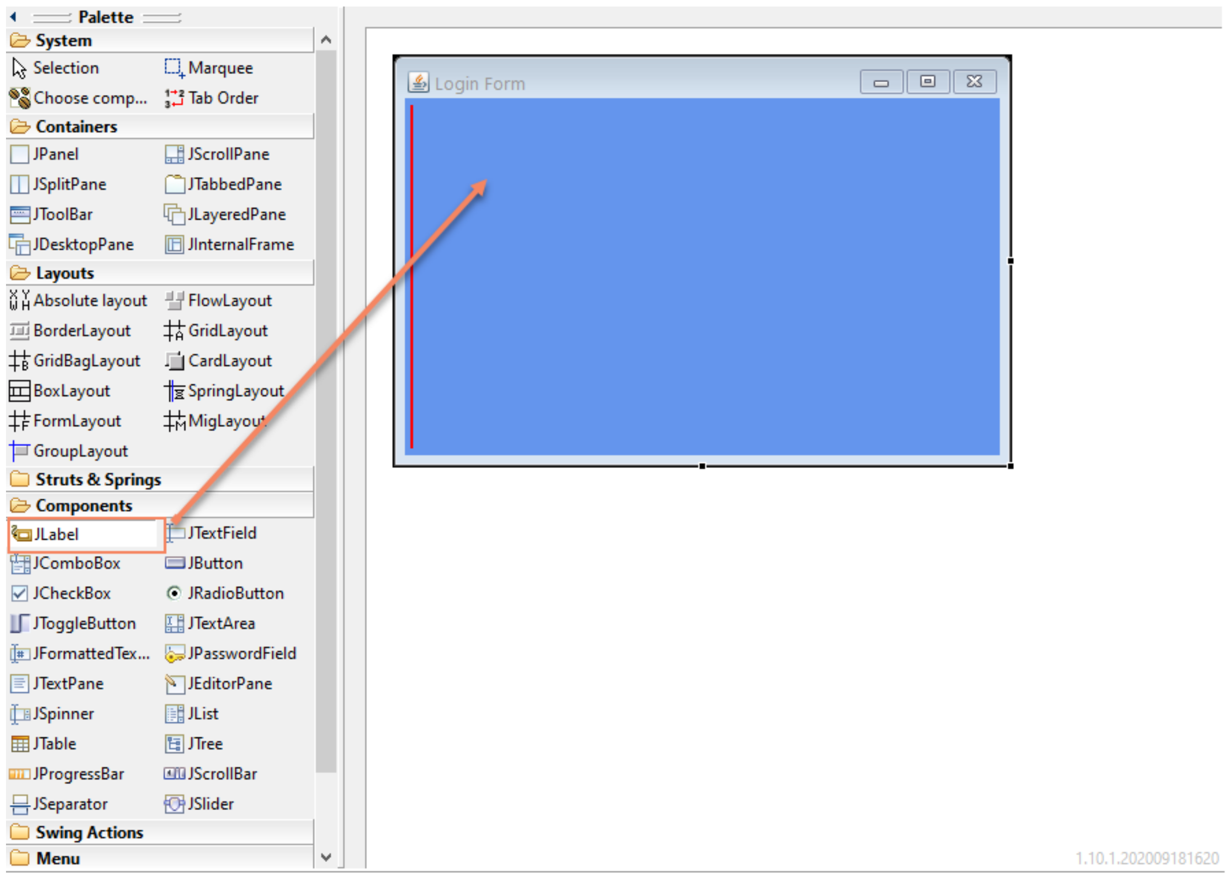
In the `Properties` window, click to expand `Layout (GridLayout)` and modify the columns to 2, rows to 3, and both hgap and vgap to 5 pixels. In the `Palette` window, click on `JLabel` to select this component and move the mouse click on the top left portion of the light blue background to create a new instance of JLabel. Now you need to change the `text` property of this new JLabel to `Username` and change the `font` property to a bigger font size: change from Tahoma 11 to Tahoma 14 as an example.



Variable	contentPane
Layout	(java.awt.GridLayout)
Class	java.awt.GridLayout
Constructor	(Constructor properties)
columns	2
hgap	5
rows	3
vgap	5
Class	javax.swing.JPanel
background	100,149,237
border	EmptyBorder
foreground	■ 0,0,0
tab order	
toolTipText	



labelFor	
text	Username:
toolTipText	
verticalAlign...	CENTER



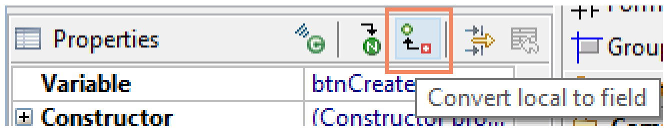
Now, create the Username text field by similarly select the `JTextField` component on the Pallette window. Then click on the right side of the "Username:" label on the form (see a vertical red line). You need to change the variable name in the Properties window from "textField" to `"txtUsername"`.

1.2.7. Now, do the similar as above to create the following components:

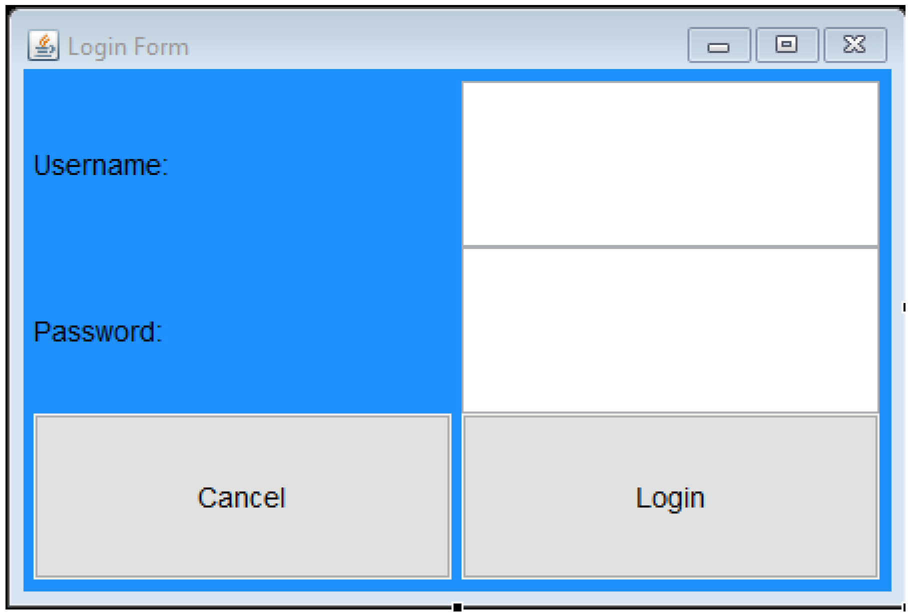
Create additional GUI components for this Login Form:

- A new JLabel component displaying "Password".
- A new JPasswordField component after the label "Password:" (it will appear on the second row, the second column). Rename this component to `txtPassword` on the Properties window
- Two JButton variables are named "btnLogin" and "btnCancel" in the third row. Remember to change the `text` property of these two buttons accordingly. Also, change their font to Tahoma 14.
- As we need to handle the events of the two buttons **btnCancel** and **btnLogin** later, we need to convert two buttons from local variables to class fields. You can do this by selecting the button component, then on the **Properties** window, click on an icon button in the same row with **Properties** and has the tooltip `"Convert local to field"`.

You should have a similar design view of the Login Form as on the right image. Now click on the `Source` tab on the bottom to check the source code of the "Login Form" class. You can see that WindowBuilder and Swing Designer plugin helps you design the form and generate code for you. If you run the project, it should display a form similar to the first image of this lab.



Variable	btnCreate
Constructor	(Constructor pro...



If you click on different places on the form now, the buttons are not working as we have not handled the events that users click on them. The only thing work is the group of 3 buttons minimize, maximize and exit form.

1.2.8. Handle the event when the user clicks on the button "Login".

For the sake of the simplicity of this lab, as we have limited lab time, we will allow any user to login into our application if they provide the same username and password. Complete the following steps:

- Modify the class header of `LoginForm` to add `implements ActionListener` after `extends JFrame`
- Hover the mouse on the class name "LoginForm" which is showing a red underline. Click on the suggestion "Add unimplemented methods". There should be an empty `actionPerformed(ActionEvent e)` generated at the end of this LoginForm class.

- Now, we need to implement code inside the `actionPerformed(ActionEvent e)` method. The `e` parameter encapsulates the information of the event. As there are two buttons that the users can click, we need to check `e.getSource()` to handle differently for each case. If the button Login was clicked, we verify if the input username from the textbox "username" and the input password from the textbox "Password" are equals and both not null or empty. Below is the pseudocode:

```
if(e.getSource() == btnLogin) {
   - String username = get the input text from text field "Username" (Hint use getText() method)
   - String password = get the input password from the text field "Password"
   - If either username or password is empty, use JOptionPage to show an error message "Username or password could not be empty! "
   - Verify that the username and the password match, if not show an error message "The username and password do not match!". If yes, open the Main Form. At this time, the Main Form is not yet designed. You can put a TODO comment here, and we will go back after finishing the Main Form in the following parts.
}
```

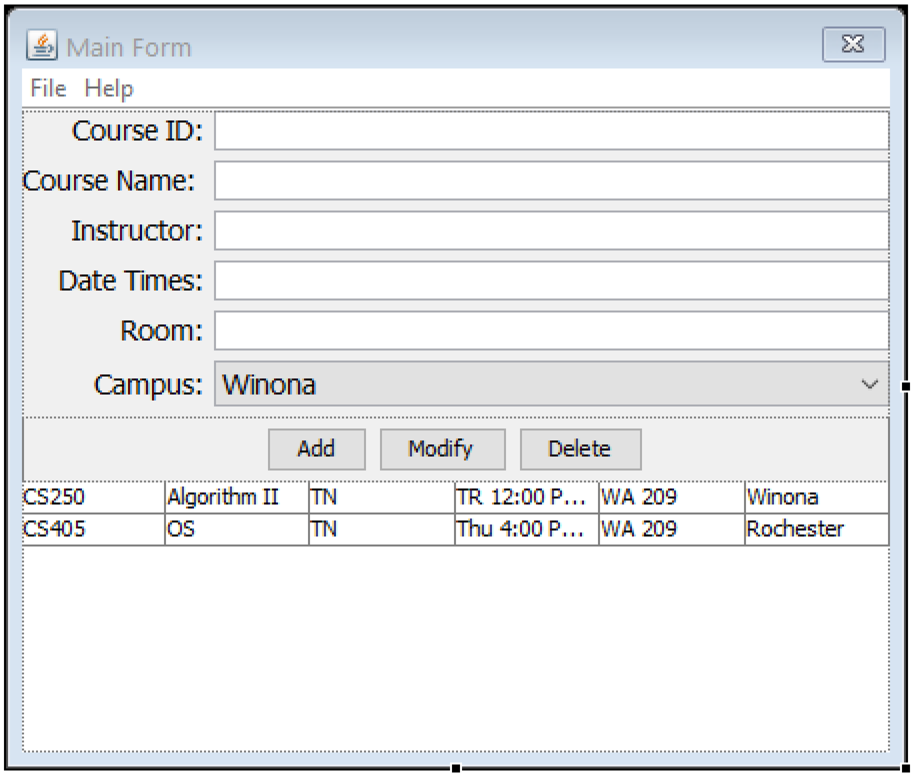
- Register the action listener of `btnLogin` in the constructor method:
  `btnLogin.addActionListener(this);`

1.2.9 Similarly, we need to handle the event that users click on the button "Cancel".

Here, we need to close the Login Form. Add and complete the following code Inside the generated actionPerformed():
Simply dispose the current Login Form

Part 2: Design and Implement the Main Form Dialog

This part will design and handle events for the Main form dialog that manage the list of courses. The design view of this form is displayed below.



CS250	Algorithm II	TN	TR 12:00 P...	WA 209	Winona
CS405	OS	TN	Thu 4:00 P...	WA 209	Rochester

Complete the following steps:

2.1 Create the MainForm class with Swing Designer

Create a new Java class named `MainForm` that extends from `javax.swing.JFrame` similar to Login Form: Right-click on "src", choose New, File, Other, then WindowBuilder, Swing Designer, choose `JFrame`, and press Next. Enter "MainForm" in the text field "Name:". A generated code of this class will be created. Then, click on the Design tab to open the Design View of this form.

2.2 Change title, background color, and layout of MainForm:

First, we need to set the title, change the background color, and change the layout of this form to use `GridBagLayout`. The following substeps guide you to do:

- Change the `title` property of this form to "Main Form" in the Properties windows.
- Change the `defaultCloseOperation` property of this form to "DISPOSE_ON_CLOSE" in the Properties windows as we do not want to exit the program when we close this form.
- Select the "contenPane" in the Components. Its properties will be displayed on the Properties window. Like the Login form, you need to change the font to "Tahoma, 14".

- Now, change the Layout property to `GridBagLayout`.

2.3 Create the GUI components on the Main Form dialog

The following steps guide you through adding components to this MainForm to display a form similar to the above figure:

- Create a JLabel on the first row, the first column, and set the text property "Course ID:"
- Create a JTextField on the first row, the second column and rename it to `txtCourseID`
- Create a JLabel on the second row, the first column, and set the text property "Course Name:"
- Create a JTextField on the second row, the second column and rename it to `txtCourseName`
- Create a JLabel on the third row, the first column, and set the text property "Instructor:"
- Create a JTextField on the third row, the second column and rename it to `txtInstructor`
- Create a JLabel on the fourth row, first column and set the text property "Date Times:"
- Create a JTextField on the fourth row, the second column and rename it to `txtDateTime`
- Create a JLabel on the fifth row, the first column, and set the text to "Room:"
- Create a JTextField on the fifth row, the second column and rename it to `txtRoom`
- Create a JLabel on the sixth row, the first column, and set the text property "Campus:"
- Create a JComboBox on the sixth row, the second column, and rename the variable to cboCampus. We need to enter some values of this combo box: On the Properties window, click on "model", then a new window form is opened. Enter the following string in separated lines: "Winona", "Rochester", and "Other". Then click the OK button to close it. Convert this combo box to the class fields by selecting the component. On the Properties window, click on an icon button in the same row with Properties and has the tooltip `"Convert local to field".`
- Create a JPanel on the 7th row. In the `Constraints` property, set `gridwidth` to 2 to expand this panel to two columns
- Create a JButton named `btnAdd` inside the panel on the 7th row. Change the text property to "Add".
- Create a JButton named `btnModify` inside the panel on the 7th row. Change the text property to "Modify".

- Create a JButton named btnDelete inside the panel on the 7th row. Change the text property to "Delete".
- Convert three buttons btnAdd, btnModify, and btnDelete to the class fields by selecting the button component. On the Properties window, click on an icon button in the same row with Properties and has the tooltip "Convert local to field".
- Create a JTable named table on the 8th row, below the panel containing three buttons. Also, in the Constraints property, set gridwidth to 2 to expand this table to two columns.
- Drop a JMenuBar between the form title and "Course ID:" component. Now, insert two JMenu on this menu bar. Change the text of the first JMenu object to File and the second JMenu to Help. Next, create the menu items for each menu as follows:
  - Inside the File menu, create three JMenuItem components with the text Add, Modify, Delete and renamed three menu items to mnuAdd, mnuModify, mnDelete
  - Inside the Help menu, create one JMenuItem component with the text About, and rename that menu item to mnuAbout Convert all menu items above (mnuAdd, mnuModify, mnDelete, and mnuAbout) from local to fields
- Change all components in this form to use the font "Tahoma 14".
- Resize the design form to fit the components neatly.

If you do correctly, the design view should be similar to the image on the right above.

2.4 Implement a method to write an array of students to a text file:

In this step, we will load some data to the JTable and add a JScrollPane to wrap this table object. To do these things, first, switch from the Design view to Source View. Complete the following steps:
- Declare an instance variable of the type DefaultTableModel named tableModel after the declaration of the table variable:
`final DefaultTableModel tableModel = new DefaultTableModel();`
- Find the line that initializes the table variable in the constructor method (table = new JTable();), insert the following code before that line:

```
// Initialize column headings.
String[] Headings = { "ID", "Name", "Instructor", "DateTime", "Room", "Campus"
};
tableModel.setColumnIdentifiers(Headings);
// Initialize data.
String[][] data = {
    { "CS250", "Algorithm II", "TN", "MW 10:00 AM - 11:50 AM", "WA 209",
"Winona" },
    { "CS405", "OS", "TN", "TH 10:00 AM - 11:20 AM", "WA 108", "Winona" }
};
for(String[] row : data)
    tableModel.addRow(row);
```

- Insert this code `table.setModel(tableModel);` after the line that initialize the `table` variable
- Go down, find the line of code `getContentPane().add(table, gbc_table);` and modify it to

```
    getContentPane().add(new JScrollPane(table), gbc_table);
```

Now, open the Design View again. You should see the imported data rows on the table.

2.5 Handle the events on the Main Form:

This section will guide you through handling some events when users click on buttons, table, and menu items:

- On the Designer view, double click on the button Delete.
- Now, add code to the `actionPerformed(ActionEvent e)` method to handle the event when **button Delete or menu item Delete** are clicked. Here, we want to delete the selected rows on the table. Use the following pseudocode to complete this step:

```
if( e.getSource() == btnDelete || e.getSource() == mnuDelete ) {
    int[] selectedRows = get selected rows of the table (Hint: check
getSelectedRows() method)
    if(selectedRows.length > 0) { //Make sure the rows are selected
        use a for loop from selectedRows.length - 1 down to 0 do
           call tableModel.removeRow() with argument selectedRows[i]
    } else {
        call JOptionPane.showMessageDialog() with arguments this and "Please
select at least one row of the table"
    }
}
```

- On the Designer view, double click on the button Add. We need to add code to the `actionPerformed(ActionEvent e)` method to handle the event when **button Add, or menu item Add are clicked**. Here, we want to validate the user's input on the text fields, combo box and insert a new row after the last row of the table. Use the following pseudocode to complete this step:

```
else if( e.getSource() == btnAdd || e.getSource() == mnuAdd ) {
    String[] newRow = new String[table.getModel().getColumnCount()]; //string
array with length = #columns
    newRow[0] = txtCourseID.getText();  //get input on Course ID text field
    newRow[1] = txtCourseName.getText();
    newRow[2] = txtInstructor.getText();
    newRow[3] = txtDateTime.getText();
    newRow[4] = txtRoom.getText();
    newRow[5] = (String) cboCampus.getSelectedItem();
    call tableModel.addRow() with argument newRow
}
```

- On the Designer view, double click on the button Modify. Now, we need to add code to the `actionPerformed(ActionEvent e)` method to handle the event when the **button Modify or menu item Modify** are clicked. Here, we want to validate the user's input on the text fields and combo box and update the selected row on the table. Use the following pseudocode to complete this step:

```
else if( e.getSource() == btnModify || e.getSource() == mnuModify ) {
     int selIdx = get selected row of the table (Hint: check getSelectedRow()
method)
    if(selIdx >= 0) { //Make sure a row is selected
        //update each cell values in the TABLE MODEL
        call tableModel.setValueAt() with arguments txtCourseID.getText(),
selIdx and 0
        call tableModel.setValueAt() with arguments txtCourseName.getText(),
selIdx and 1
        call tableModel.setValueAt() with arguments txtInstructor.getText(),
selIdx and 2
        call tableModel.setValueAt() with arguments txtDateTime.getText(),
selIdx and 3
        call tableModel.setValueAt() with arguments txtRoom.getText(), selIdx
and 4
        call tableModel.setValueAt() with arguments
cboCampus.getSelectedItem().toString(), selIdx and 5
    } else {
        call JOptionPane.showMessageDialog() with arguments this and "You need
to select one row in the table to update"
    }
}
```

- Before the case when the button Modify or menu item Modify is clicked, the user needs to select a row on the table. Here, we want to load the data on the selected

row to the text fields and combo box on the form to edit them. Use the following pseudocode to complete this step:
- Open the Designer view again. Right click on the table component and choose "Add event handler" -> "mouse" => "mouseClick". A new code will be generated in MainForm class that register a MouseListener for the table object. Here, we will add the code to the method `public void mouseClicked(MouseEvent arg0)`:

```
if(arg0.getSource() == table) { //make sure the mouse is clicked on the table
    int selIdx = get selected row of the table (Hint: check getSelectedRow()
method)
    if(selIdx >= 0) { //Make sure a row is selected
        //set each text field text with the corresponding cell's value of the
selected row
        call txtCourseID.setText with arguments tableModel.getValueAt(selIdx,
0).toString()
        call txtCourseName.setText() with arguments
tableModel.getValueAt(selIdx, 1).toString()
        call txtInstructor.setText() with arguments
tableModel.getValueAt(selIdx, 2).toString()
        call txtDateTime.setText() with arguments tableModel.getValueAt(selIdx,
3).toString()
        call txtRoom.setText() with arguments tableModel.getValueAt(selIdx,
4).toString()
        //for the combobox, you need to check the value and set the
corresponding selected index
        String campus = tableModel.getValueAt(selIdx, 5).toString();
        if( the lower case of campus contains "winona" )
            call cboCampus.setSelectedIndex() with argument 0
        else if( the lower case of campus contains rochester" )
            call cboCampus.setSelectedIndex() with argument 1
        else
            call cboCampus.setSelectedIndex() with argument 2
    } else {
        call JOptionPane.showMessageDialog() with arguments this and "You need
to select one row in the table to update"
    }
}
```

Go back to the Login Form, add code to display this Main Form in the code that handles the event of button Login is clicked, and username and password are matched.

```
MainForm mainForm = new MainForm();
mainForm.show();
```

Now, run the main form, test it by clicking on the buttons, inputting the text fields, clicking menu items, etc. Ask the instructor if you have trouble.

Lab Assignment

Complete the following tasks:

a. Modify the code that handles the events when button Add, or menu item Add are clicked so that a new row is only added to the table if the user has entered at least one text field. If all text fields are empty, show an error message using JOptionPage with this text "You need to enter the value of at least one text field".

b. Modify the code that handles the events when button Modify or menu item Modify are clicked so that the selected row is only updated if the user has entered at least one text field. If all text fields are empty, show an error message using JOptionPage with this text "You need to enter the value of at least one text field".

c. Add a new About dialog that extends from Dialog named "AboutForm". This dialog should display your full name, student id, current course number, and the date-time you create this About dialog. Use the method to create the Login Form and Main Form to create this dialog. In the Main Form, add code to show this About dialog when users click on the Help - About menu item.

Lab Result Submission:

You need to submit the results of the following sections to the D2L assignment item of Lab#5:

- Complete the required Lab Assignment above

- Compress the whole Eclipse source code of your Lab#5 project in zip format using the Export function