

Chapter 8:

GUI Dialog & Table

Informatics Practices
Class XII

By- Rajesh Kumar Mishra

PGT (Comp.Sc.)

KV No.1, AFS, Suratgarh

e-mail : rkmalld@gmail.com

Objective

In this presentation, you will learn about the following-

- ❑ **JDilog Swing Control:**
Design & implementation.
 - ❑ **JOptionPane:**
Using pre-defined, built-in dialog control.
 - ❑ **JTable Swing Control:**
Displaying Data in Tabular Form using JTable Control.
-

GUI Dialogs in JAVA- Introduction

- ❑ A Dialog box is a small separate sub-window that appears to either provide or request information to/from the users.
 - ❑ Dialog Boxes are used in the application for user-friendly interaction.
 - ❑ Java provides extensive support for dialogs. It offers the following dialog types-
 - ❖ JDialog: General purpose dialog using JDialog class.
 - ❖ JOptionPane: Predefined and ready-to-use built-in dialog box.
 - ❖ JFileChooser: Dialog for choosing files.
 - ❖ JColorChooser: Dialog for choosing color.
-

JDialog – A Simple General purpose dialog

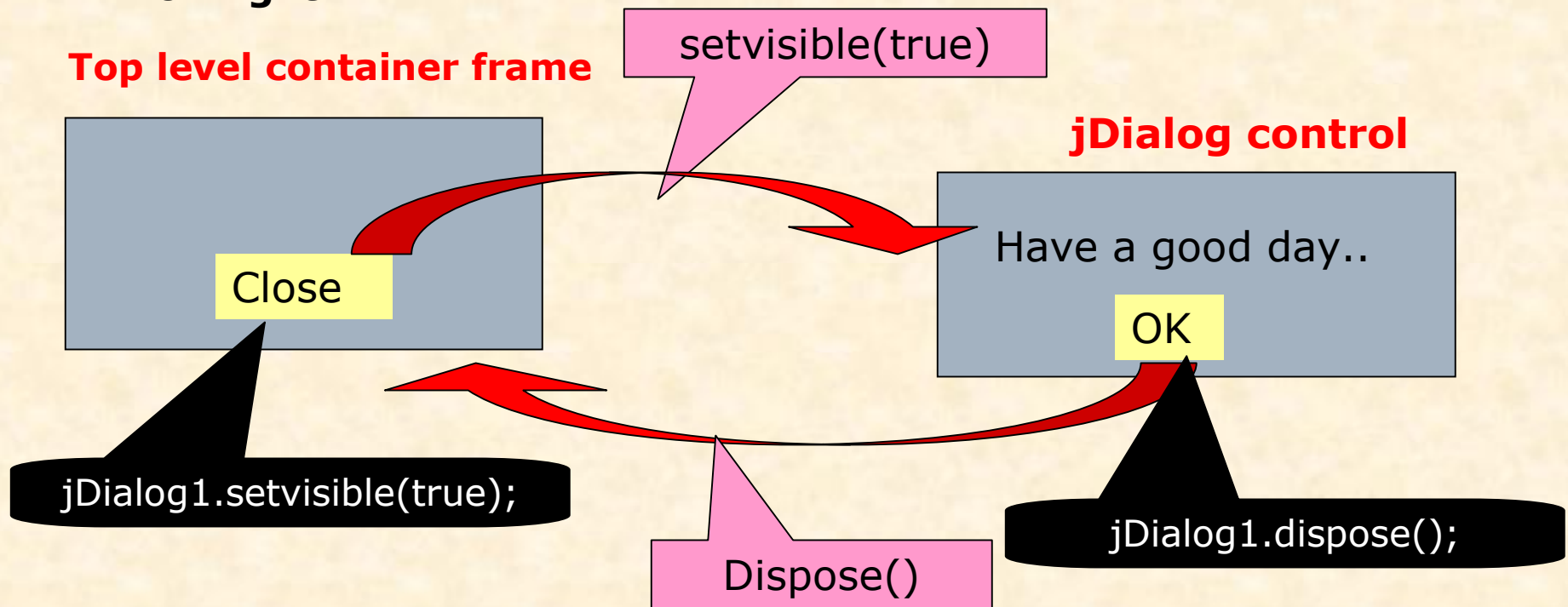
- ❑ The JDialog is a swing window dialog equipped with Minimise, Maximise and Close functionality.
 - ❑ JDialog creates a standard pop-up window for displaying desired information related to your application.
 - ❑ Steps to Attach a JDialog Control-
 1. Design an application as required and drag Dialog control from swing palette and drop it into the application.
 2. To open Dialog frame, double click on jDialog node under other component in Inspector window.
 3. Attach message button and title in Dialog control as per requirement.
 4. Attach the following ActionPerformed event handler of attached button.

jDialog1.dispose();
 5. Now switch to Frame by double clicking on JFrame node in Inspector window.
 6. Attach the following code on button from where you want to run the dialog.

jDialog.setVisible(true);
-

Working with Dialog Control

- A dialog control is a control that can be used to display messages in the application. It may contains message , image and buttons etc.
- ❑ Add `JDialog` control from swing controls and customize it as per you requirement with text, image and buttons.
 - ❑ It can be invoked in `ActionPerformed` Event of a button in top level container `JFrame` by `JDialog1.setVisible(true)` command.
 - ❑ You can set text at run time by `JDialog1.setText()` method before invoking it.



JOptionPane : Built-In Dialog of JAVA

JOptionPane allows to create pop-up window or dialog with predefined style.

A JOptionPane Dialog window comprises the following elements-

- ❑ **Icon:**

A JOptionPane provides four types of Icons, these are- Error (X), Information (i), Warning (!) and Question (?)

- ❑ **Message:**

A text to convey the information to user.

- ❑ **Input Area:**

It allows user to give input using Text box or combo.

- ❑ **Button(s):**

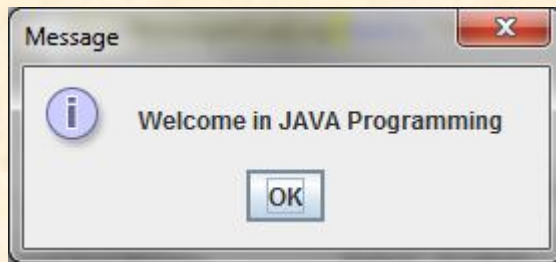
Set of buttons like OK, Cancel, Yes, No etc.

JOptionPane Dialog Types:

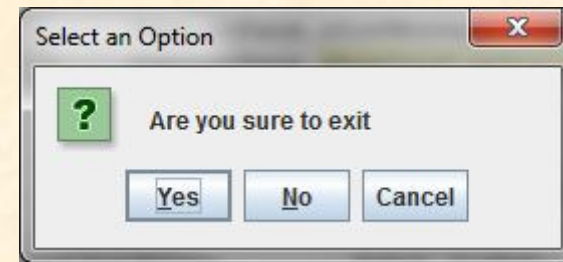
Java provides four types of Dialogs which can be used as per requirement.

Dialog Types	Method	Description
Message Dialog	showMessageDialog()	Used to inform user by displaying a message. It includes OK button only.
Input Dialog	showInputDialog()	Used to get user input using JTextField, JComboBox etc.
Confirm Dialog	showConfirmDialog()	Used to ask a user to confirm some information with Yes/No or OK/Cancel buttons.
Option Dialog	showOptionDialog()	You may customize these dialog with text messages and buttons as per need

JOptionPane Dialog Types:



Message Dialog



Confirm Dialog



Input Dialog

Working with JOptionPane

To use the JOptionPane dialog control in the application, you must import the following class(s).

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

In general, the following syntax of methods along with optional parameters can be used-

**JOptionPane.show.....([frame name],<"Message">
[,<"Title">] [,Option Type] [,Message Type])**

❑ **Frame Name:**

Generally null is used, since current frame is assumed to display dialog boxes.

❑ **Message:**

User given string to convey the message.

❑ **Title:**

Text to be displayed as Title on the Title bar.

Working with JOptionPane

❑ Option Type:

It determines the Buttons to be displayed on the dialog. The following values or equivalent code may be used.

Value	Equivalent code	Buttons
-1	<code>JOptionPane.DEFAULT_OPTION</code>	OK button only.
0	<code>JOptionPane.YES_NO_OPTION</code>	YES and No button.
1	<code>JOptionPane.YES_NO_CANCEL_OPTION</code>	YES , NO and CANCEL
2	<code>JOptionPane.OK_CANCEL_OPTION</code>	OK and CANCEL button

❑ Message Type:

It determines the Icon to be displayed. The values are-

Value	Equivalent code	Buttons
-1	<code>JOptionPane.PLAIN_MESSAGE</code>	No Icon.
0	<code>JOptionPane.ERROR_MESSAGE</code>	X Icon.
1	<code>JOptionPane.INFORMATION_MESSAGE</code>	i Icon.
2	<code>JOptionPane.WARNING_MESSAGE</code>	! Icon
3	<code>JOptionPane.QUESTION_MESSAGE</code>	? Icon

How to use JOptionPane dialog

You can invoke JOptionPane Dialogs by using .show.....Dialog() method in the coding where you want to display the dialog box.

Examples:

```
JOptionPane.showMessageDialog(null,"JAVA Welcomes You");
```

```
JOptionPane.showMessageDialog(null,"Hi.. I am Amit");
```

```
JOptionPane.showMessageDialog(null,"JAVA Welcomes You","Hi..");
```

```
String n= JOptionPane.showInputDialog(null,"Enter your Name ?");
```

```
String n= JOptionPane.showInputDialog(null,"Enter your Name ?",  
                                         "Input Name");
```

```
String n= JOptionPane.showInputDialog(null,"Enter your Name ?", 0,3);
```

```
JOptionPane.showConfirmDialog(null,"Want to exit ?");
```

```
JOptionPane.showConfirmDialog(null,"Want to exit ?",0,3);
```

```
JOptionPane.showConfirmDialog(null,"Are you sure to delete",1,2);
```

Return Value of JOptionPane dialog

❑ Value returned by Input Dialog:

ShowInputDialog() returns a **string** value which can be directly assigned on a string type variable. You may use **parse...()** method to convert it into other data types like **int** or **float** etc.

Once value has been converted on a variable, it can be used in the application.

```
String n=JOptionPane.showInputDialog(null, "Enter Name? ");
```

```
String age=JOptionPane.showInputDialog(null, "Enter Age ? ");  
int a=Integer.parseInt(age);
```

Return Value of JOptionPane dialog

❑ Value returned by Confirm & Option Dialog:

Sometimes it is required to check the response of user i.e. which button is pressed, so that application can respond accordingly.

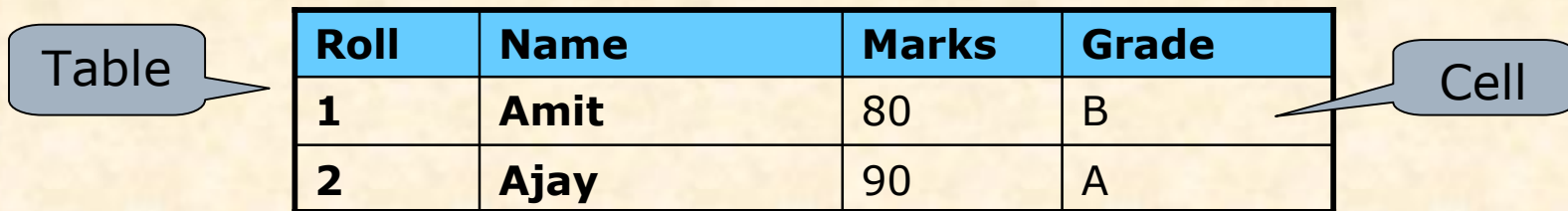
Both `showConfirmDialog()` and `showOptionDialog()` returns int type value which can be compared with the following constants to determine the status of button pressed by the user.

Returned Value	Indication
<code>JOptionPane.YES_OPTION</code>	YES button is pressed
<code>JOptionPane.OK_OPTION</code>	OK button is pressed
<code>JOptionPane.NO_OPTION</code>	NO button is pressed
<code>JOptionPane.CANCEL_OPTION</code>	CANCEL button is pressed
<code>JOptionPane.CLOSE_OPTION</code>	User closed the dialog using X button

```
int ans=JOptionPane.showConfirmDialog(null, "Want to exit ?");  
If (ans==JOptionPane.YES_OPTION)  
    System.exit(0);
```


JTable Swing Control

Sometimes it is required to represent information in tabular form. Java provides JTable swing control to handle data in a table. A table consists of certain rows and columns.



The diagram shows a JTable with four columns: Roll, Name, Marks, and Grade. The first row contains the values 1, Amit, 80, and B. The second row contains the values 2, Ajay, 90, and A. A callout bubble labeled 'Table' points to the entire table structure. Another callout bubble labeled 'Cell' points to the cell containing the value 'B' in the first row, fourth column.

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.

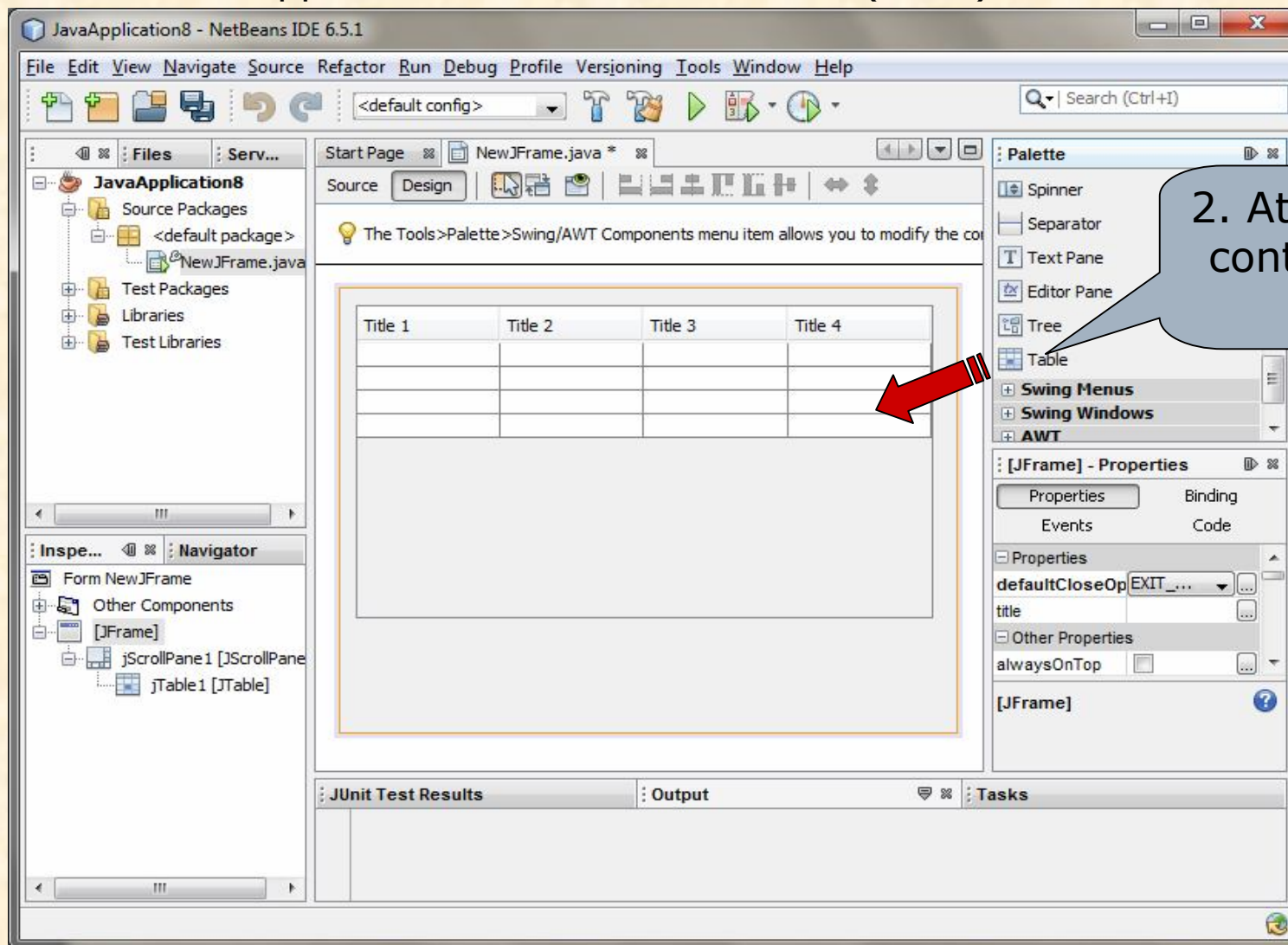
❑ Properties & Methods of JTable control:

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, Tool Tip Text** and **Enabled** etc.

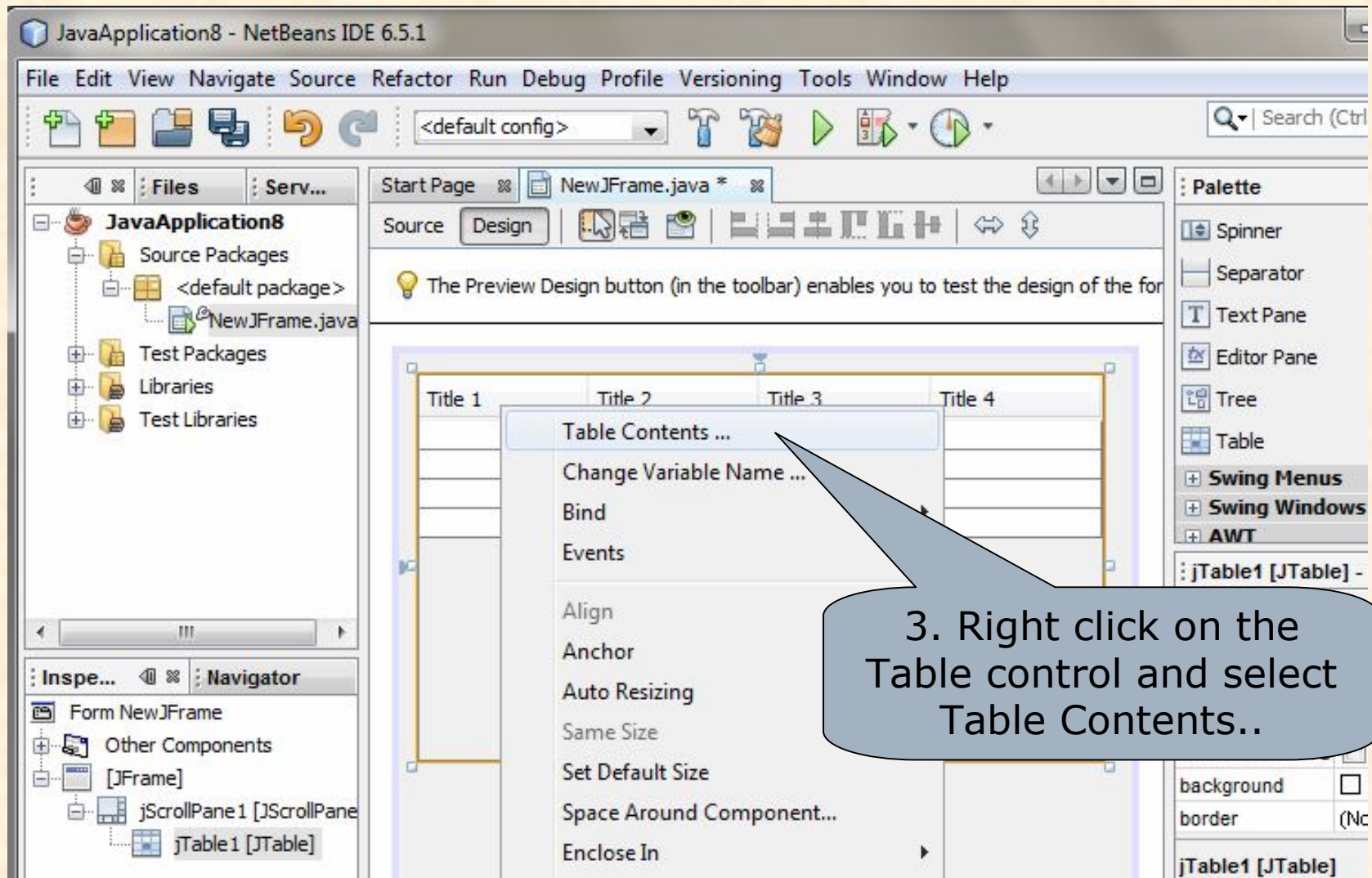
Designing a Simple Table

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



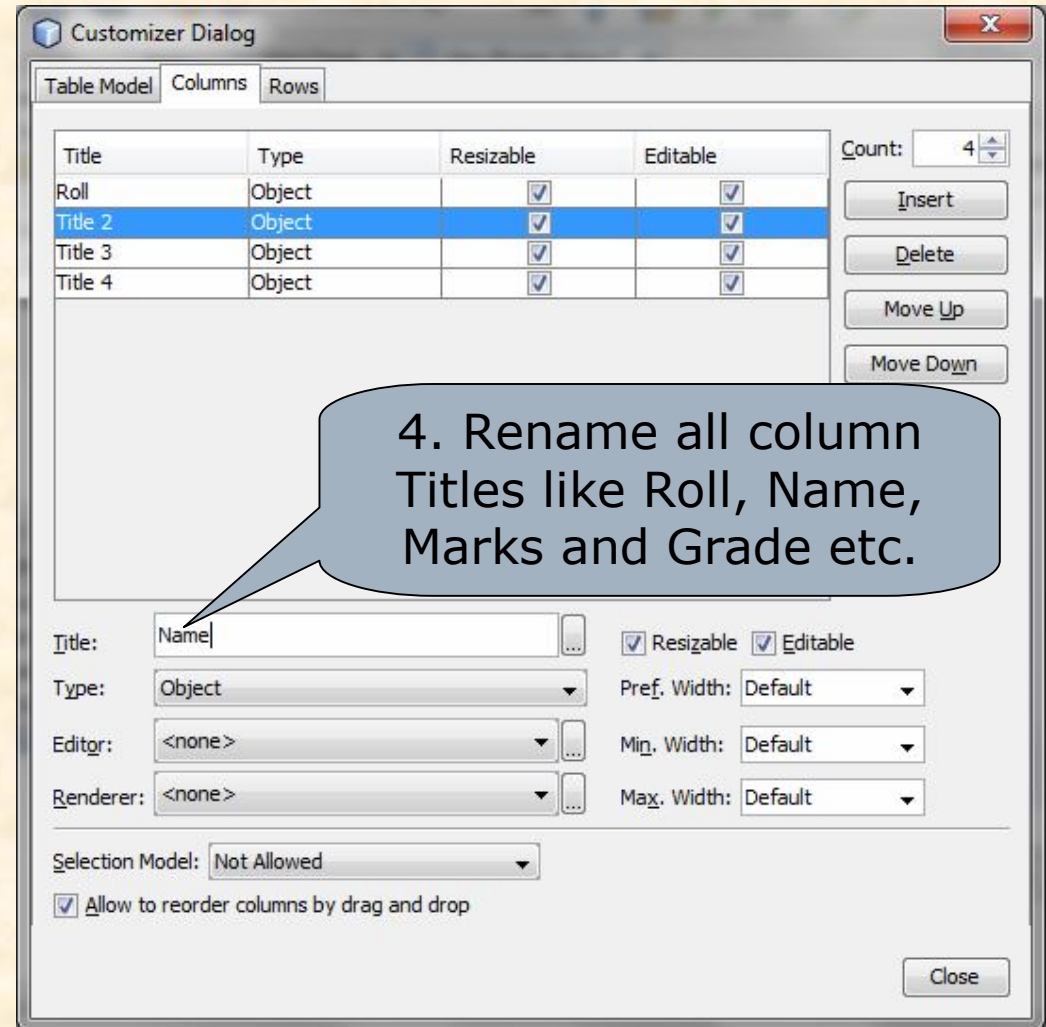
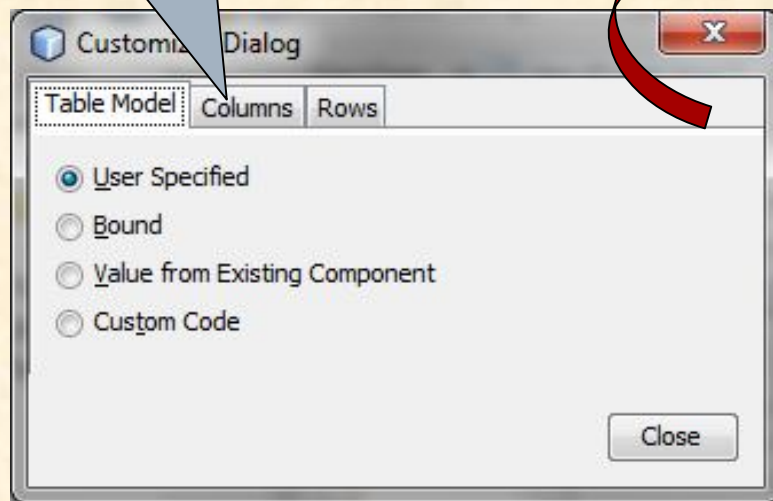
2. Attach Table control to the frame

Designing a Simple Table



Designing a Simple Table

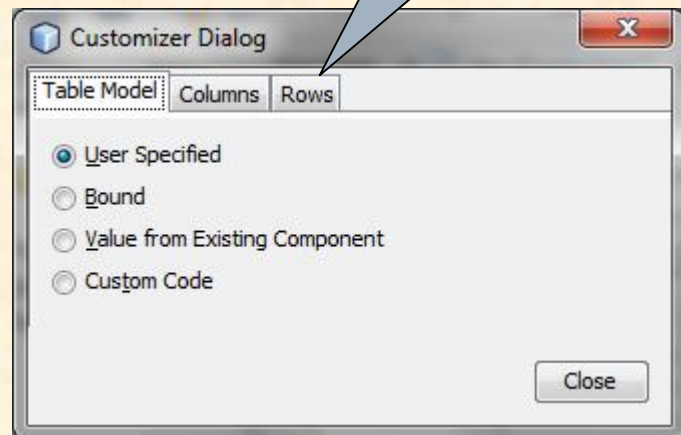
3. Select Column tab



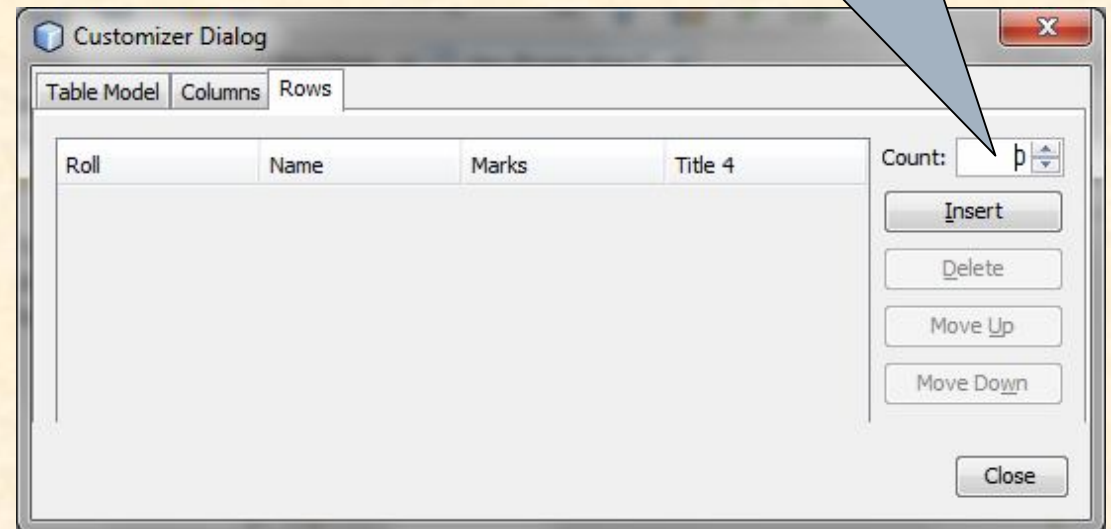
4. Rename all column Titles like Roll, Name, Marks and Grade etc.

Designing a Simple Table

5. Select Row tab



6. Set Count as 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);
```

Accessing Cell Values in the Application

Once records have been added in a table, you can access cell value directly using Object type variable.

```
// Example - to get sum of a column of the table
/* Asume a Table (jTable1) contains Roll, Name, Marks and
Grade column*/
/*The following code can be added in TODO section of a button*/
```

```
int row = jTable1.getRow();
int sum=0;
Object val= new Object();
for (int i=0; i<row ; i++)
    { val=jTable.getValueAt(i,2);
      sum = sum + Integer.parseInt(val);
    }
JOptionPane.showMessageDialog(null, "Sum of Marks:"+sum);
```