Swing

A Quick Tutorial on Programming Swing Applications

MVC – Model View Controller

- Swing is based on this design pattern
- It means separating the implementation of an application into layers or components:
  - The **Model** - the data structure that represents something (like a customer info rec)
  - The **Controller** - the user interface logic for manipulating it
  - The **View** - the display of that data structure to the user.

What is Swing?

- A set of classes (part of JFC) that support platform independent GUI (Graphical User Interface)
- Successor to the original Java GUI classes (AWT) which didn’t work very well (they had platform dependencies that really made it a difficult API to use)
- AWT wasn’t very “sexy”

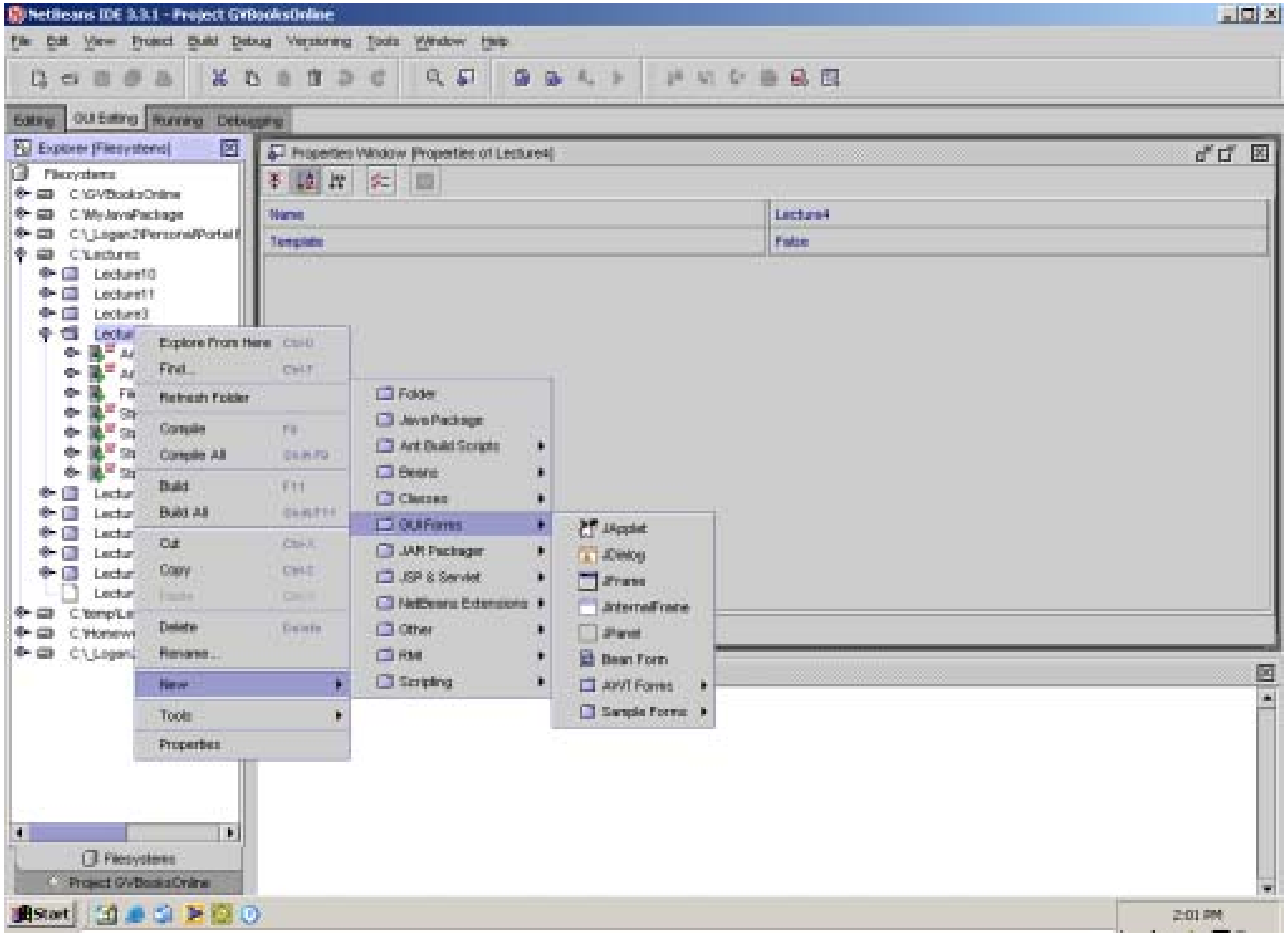
Swing

- Visible “widgets” - windows, buttons, combo boxes, trees, tables, checkboxes, text fields, menus, …
- Containers of components – applets, dialogs, windows and frames
- Supporting classes and utility methods

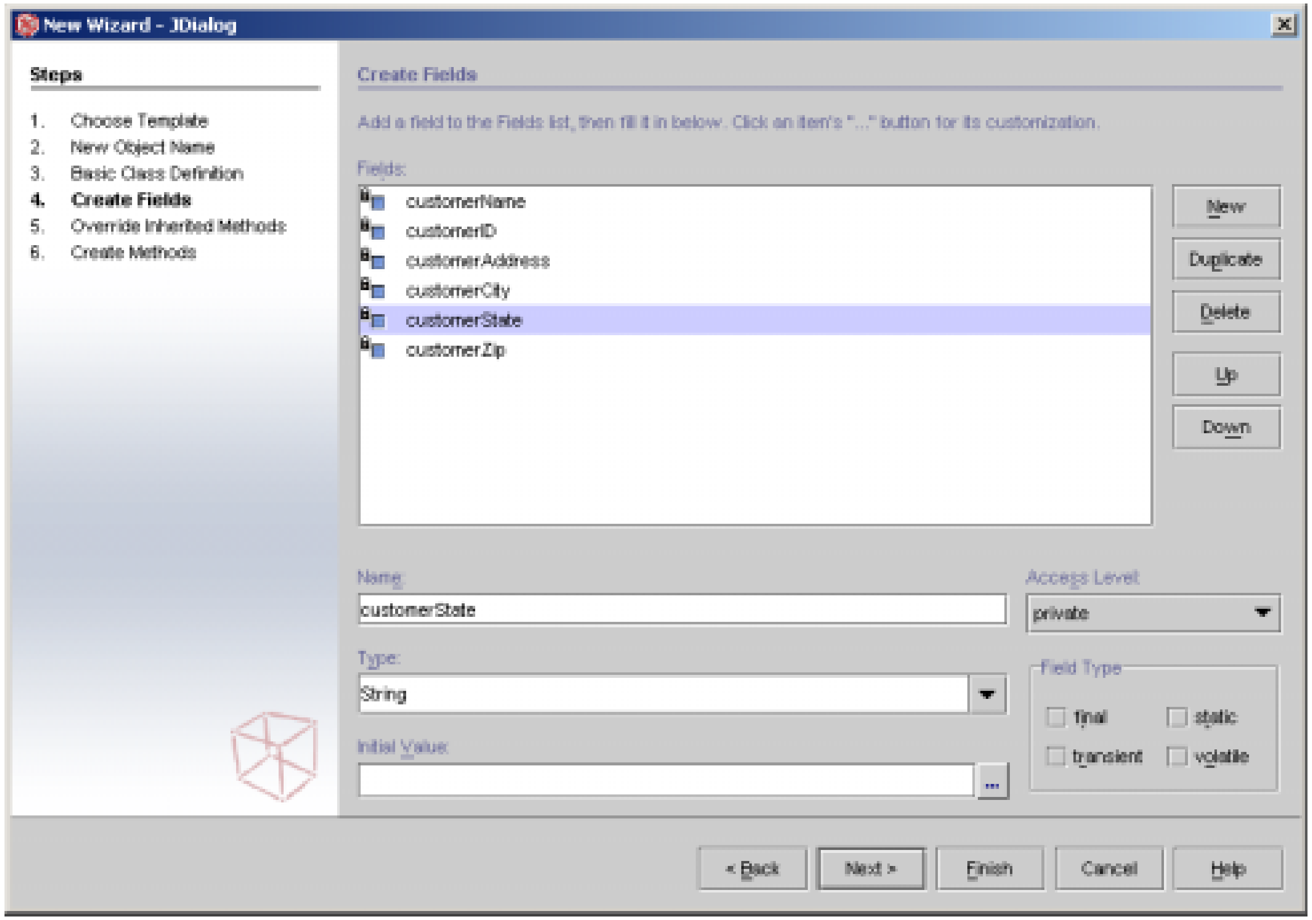
Some important Swing visible component classes

- JApplet **
- JButton
- JCheckBox
- JColorChooser
- JComboBox
- JDialog **
- JFileChooser
- JFormattedTextField
- JFrame **
- JLabel
- JList
- JMenu
- JMenuBar
- JMenuItem
- JPanel
- JPasswordField
- JPopupMenu
- JProgressBar
- JRadioButton
- JScrollBar
- JSlider
- JSpinner
- JTable
- JTextArea
- JTextField
- JToggleButton
- JToolBar
- JTree
- JWindow **
- *** means a top level containers*

Using netbeans to create a JDialog



adding fields



New Wizard - JDialog

Steps

1. Choose Template
2. New Object Name
3. Basic Class Definition
4. **Create Fields**
5. Override Inherited Methods
6. Create Methods

Create Fields

Add a field to the Fields list, then fill it in below. Click an item's "..." button for its customization.

Fields:

customerName
customerID
customerAddress
customerCity
customerState
customerZip

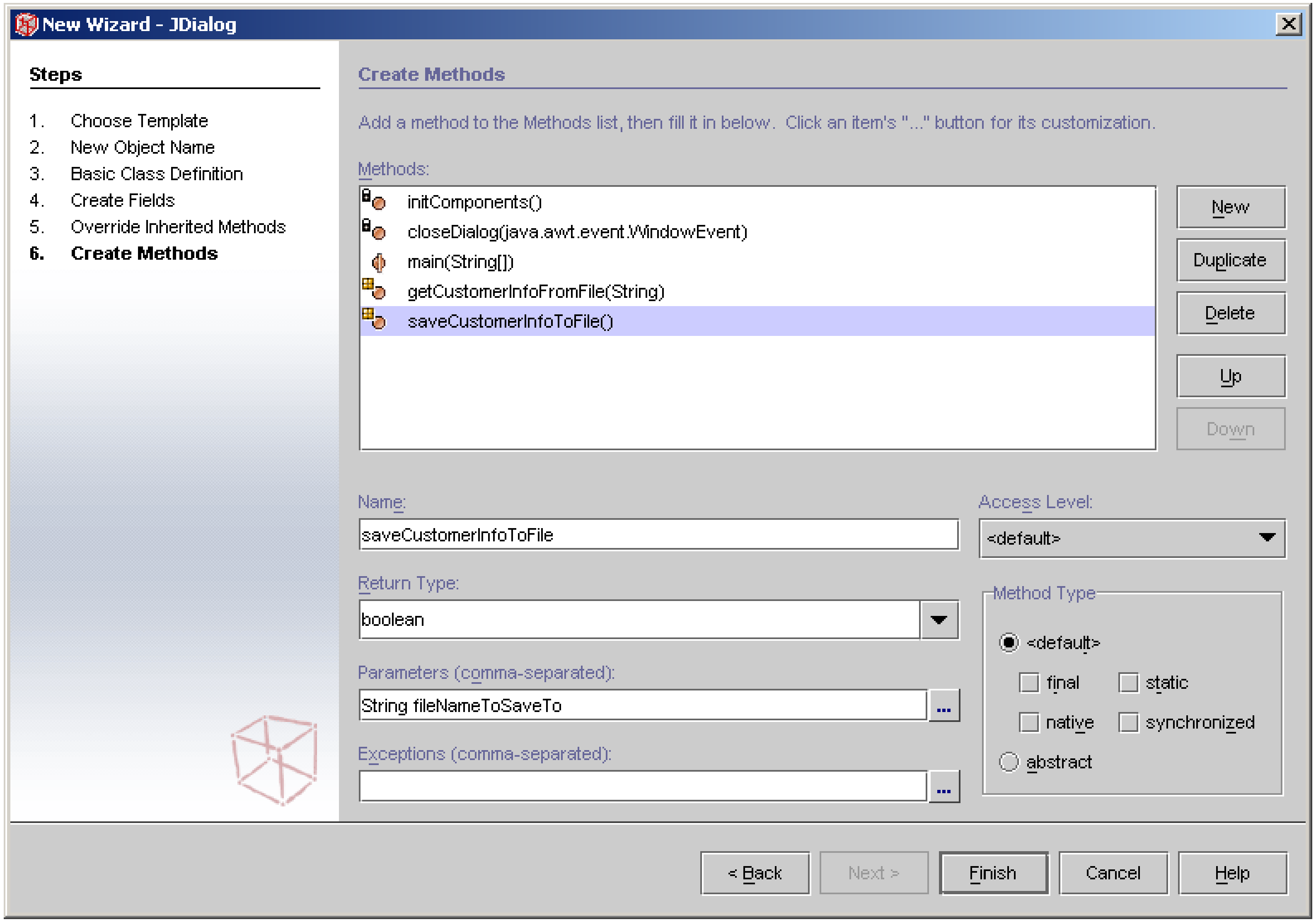
Name: customerState

Access Level: private

Type: String

Initial Value:

Field Type: final, static, transient, volatile (none checked)


New Wizard - JDialog

Steps

1. Choose Template
2. New Object Name
3. Basic Class Definition
4. Create Fields
5. Override Inherited Methods
6. **Create Methods**

Create Methods

Add a method to the Methods list, then fill it in below. Click an item's "..." button for its customization.

Methods:

initComponents()
closeDialog(java.awt.event.WindowEvent)
main(String[])
getCustomerInfoFromFile(String)
saveCustomerInfoToFile()

Name: saveCustomerInfoToFile

Access Level: <default>

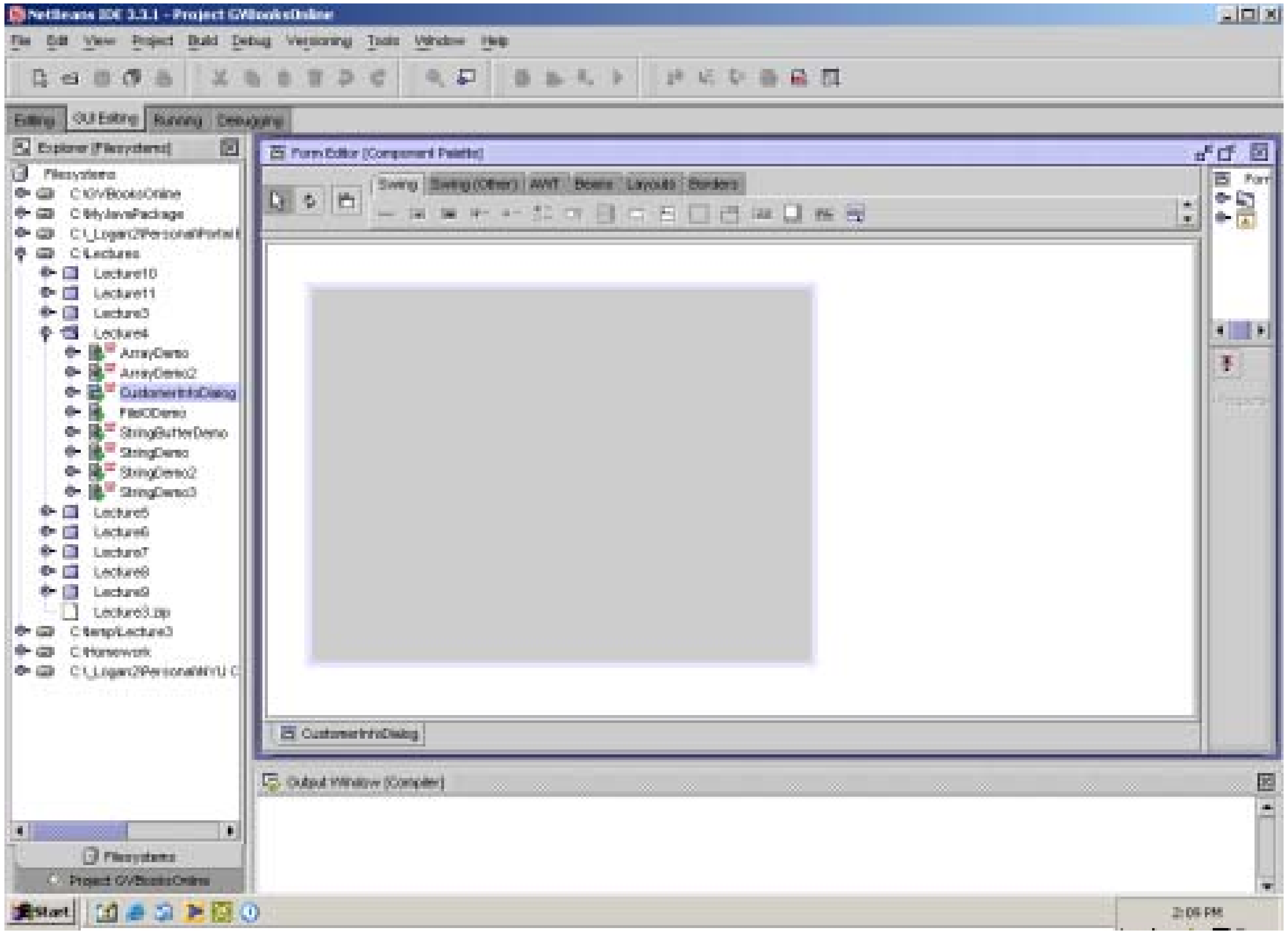
Return Type: boolean

Parameters (comma-separated): String fileNameToSaveTo

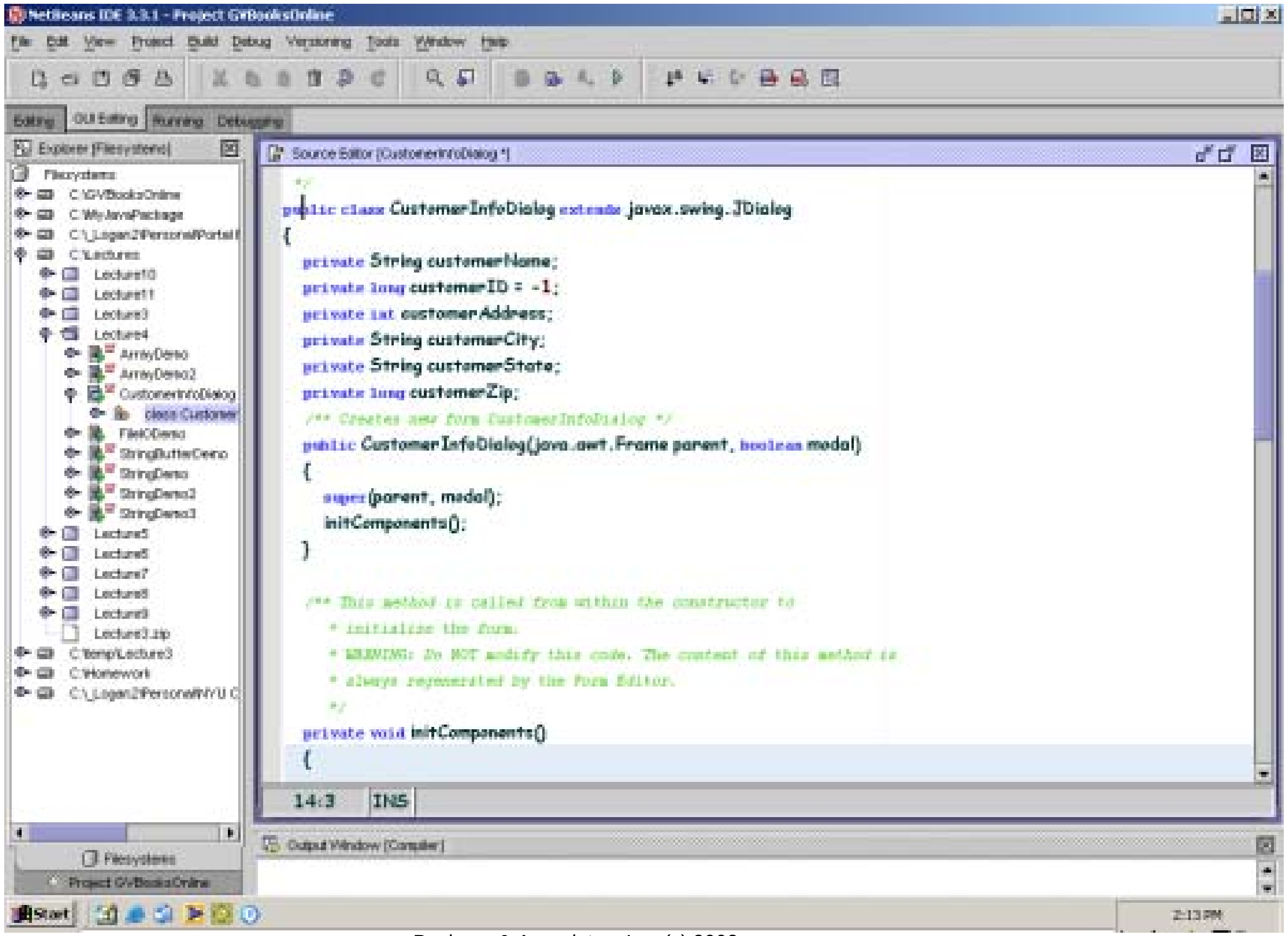
Exceptions (comma-separated):

Method Type: [✓ <default>] final static native synchronized abstract

my empty CustomerInfoDialog:JDialog

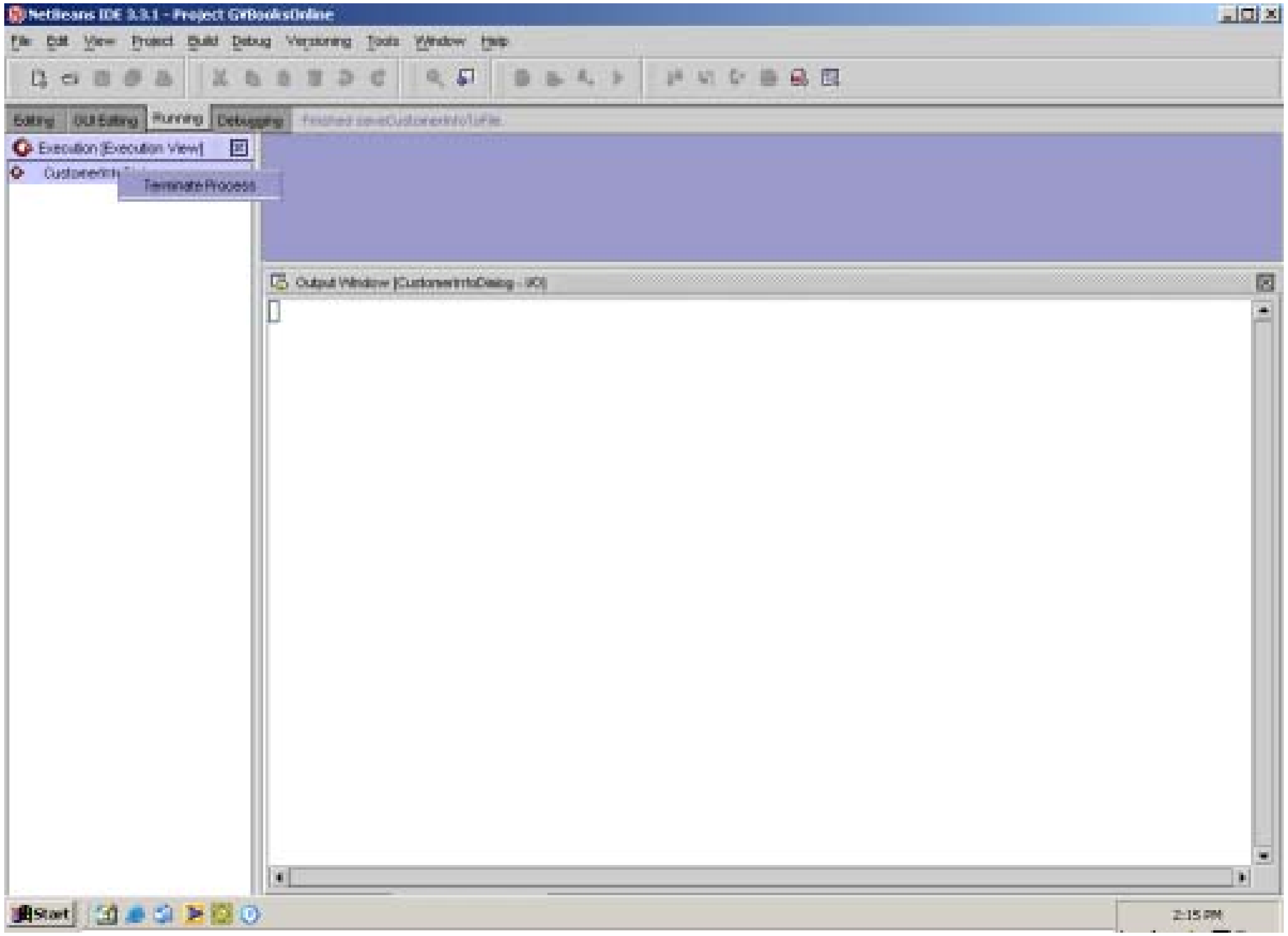


code created

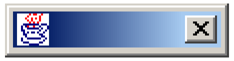


```
  */
public class CustomerInfoDialog extends javax.swing.JDialog
{
  private String customerName;
  private long customerID = -1;
  private int customerAddress;
  private String customerCity;
  private String customerState;
  private long customerZip;
  /** Creates new form CustomerInfoDialog */
  public CustomerInfoDialog(java.awt.Frame parent, boolean modal)
  {
    super(parent, modal);
    initComponents();
  }

  /** This method is called from within the constructor to
    * initialize the form.
    * WARNING: Do NOT modify this code. The content of this method is
    * always regenerated by the Form Editor.
    */
  private void initComponents()
  {
```

To kill a zombie or running process in netbeans right click and choose: ”terminate”

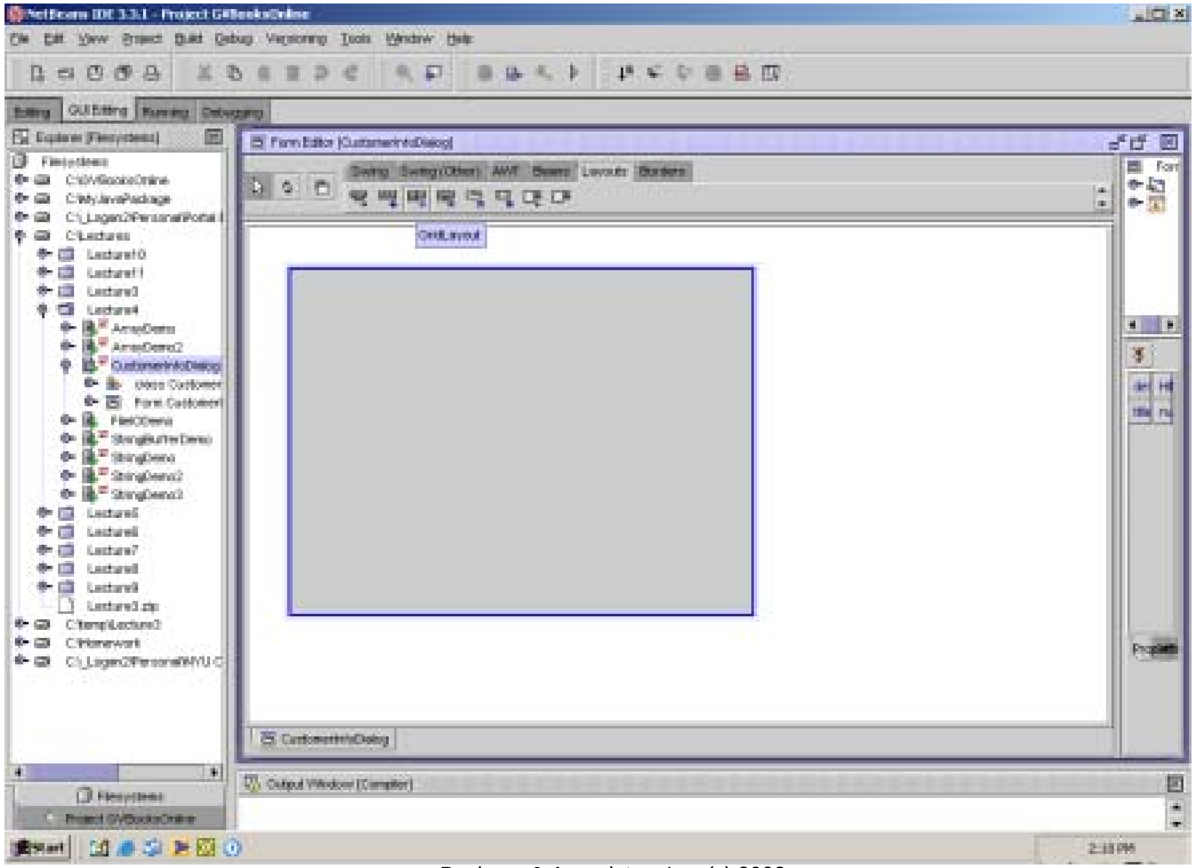

NetBeans IDE 3.3.1 - Project GVBooksOnline
File Edit View Project Build Debug Versioning Tools Window Help
Editing GUI Editing Running Debugging
Execution [Execution View]
Terminate Process
Output Window [CustomerInfoDialog - I/O]
Start
2:15 PM

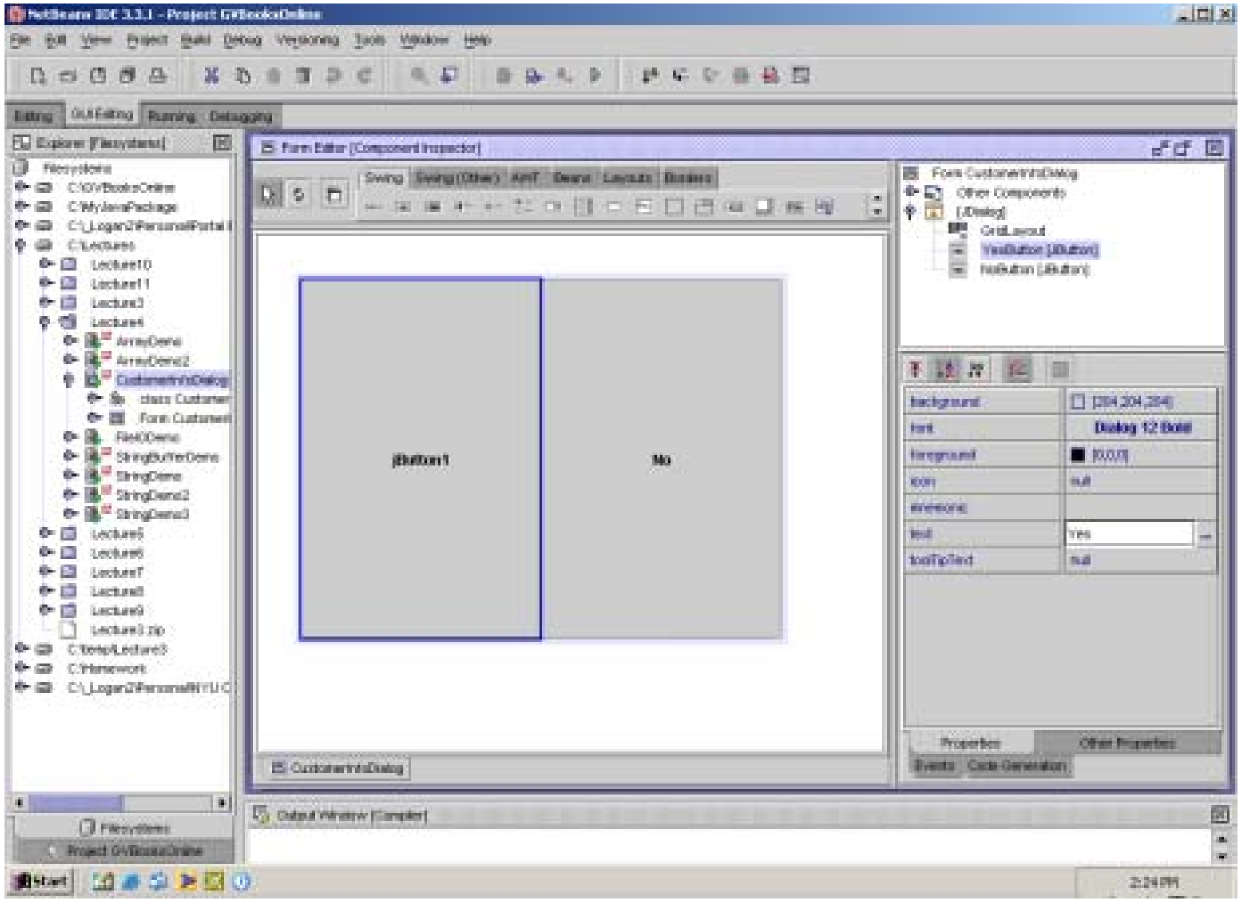
executing the class displays:



Editing a dialog

- 1St select a layout manager for the dialog


NetBeans IDE 3.3.1 - Project G4BooksOnline
File Edit View Project Build Debug Versioning Tools Window Help
Editing GUI Editing Running Debugging
Explorer [Filesystems]
Form Editor [CustomerInfoDialog]
Swing Swing (Other) AWT Beans Layouts Borders
GridLayout
CustomerInfoDialog
Output Window [Compiler]
Start

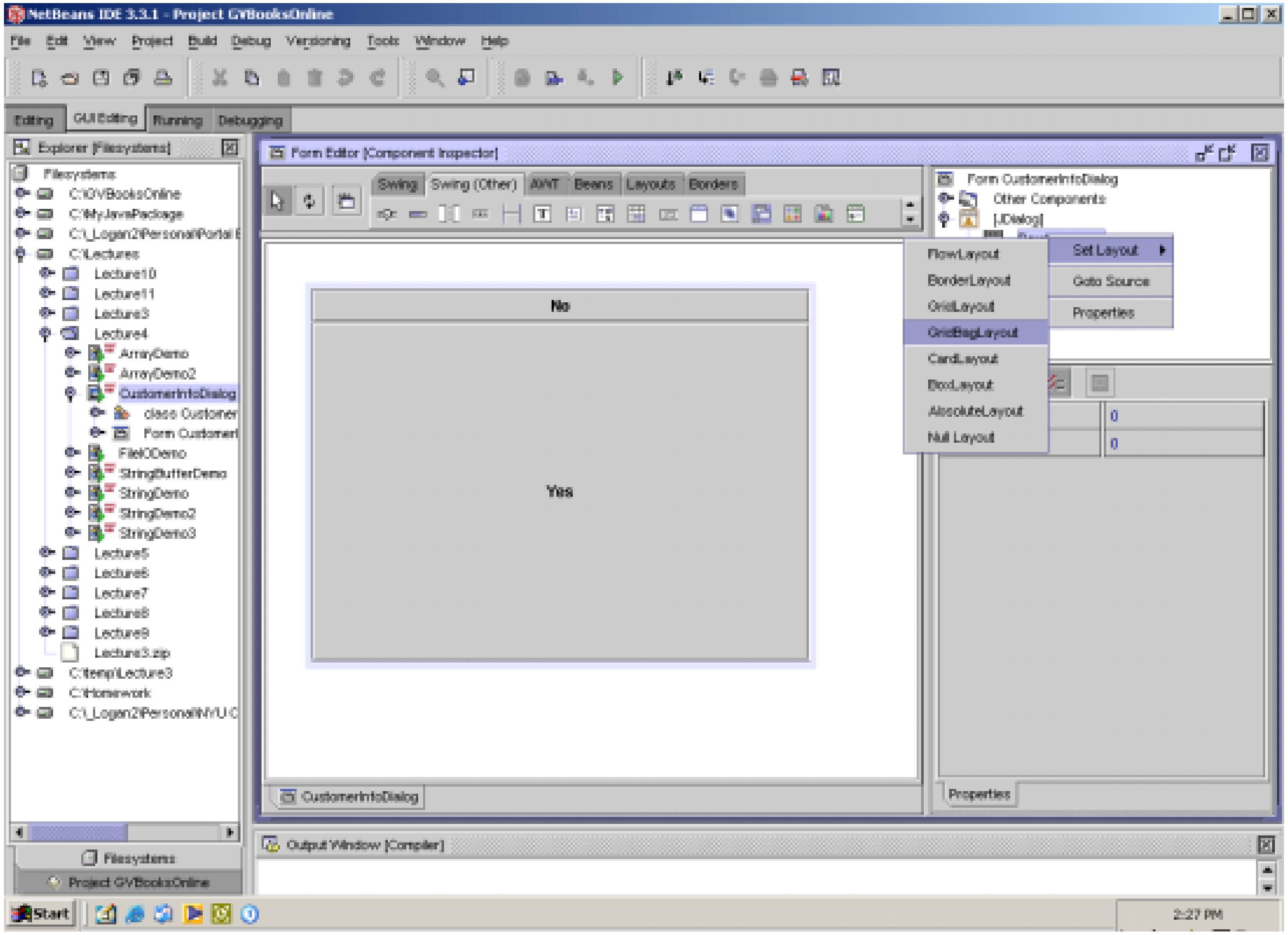

NetBeans IDE 3.3.1 - Project GVBooksOnline
File Edit View Project Build Debug Versioning Tools Window Help
Editing GUI Editing Running Debugging
Explorer [Filesystems]
Form Editor [Component Inspector]
Swing Swing (Other) AWT Beans Layouts Borders
jButton1
No
Form CustomerInfoDialog
Other Components
[JDialog]
GridLayout
YesButton [JButton]
NoButton [JButton]
Properties Other Properties
Events Code Generation
Output Window [Compiler]

background	[204,204,204]
font	Dialog 12 Bold
foreground	[0,0,0]
icon	null
mnemonic	
text	Yes
toolTipText	null

select component

edit properties

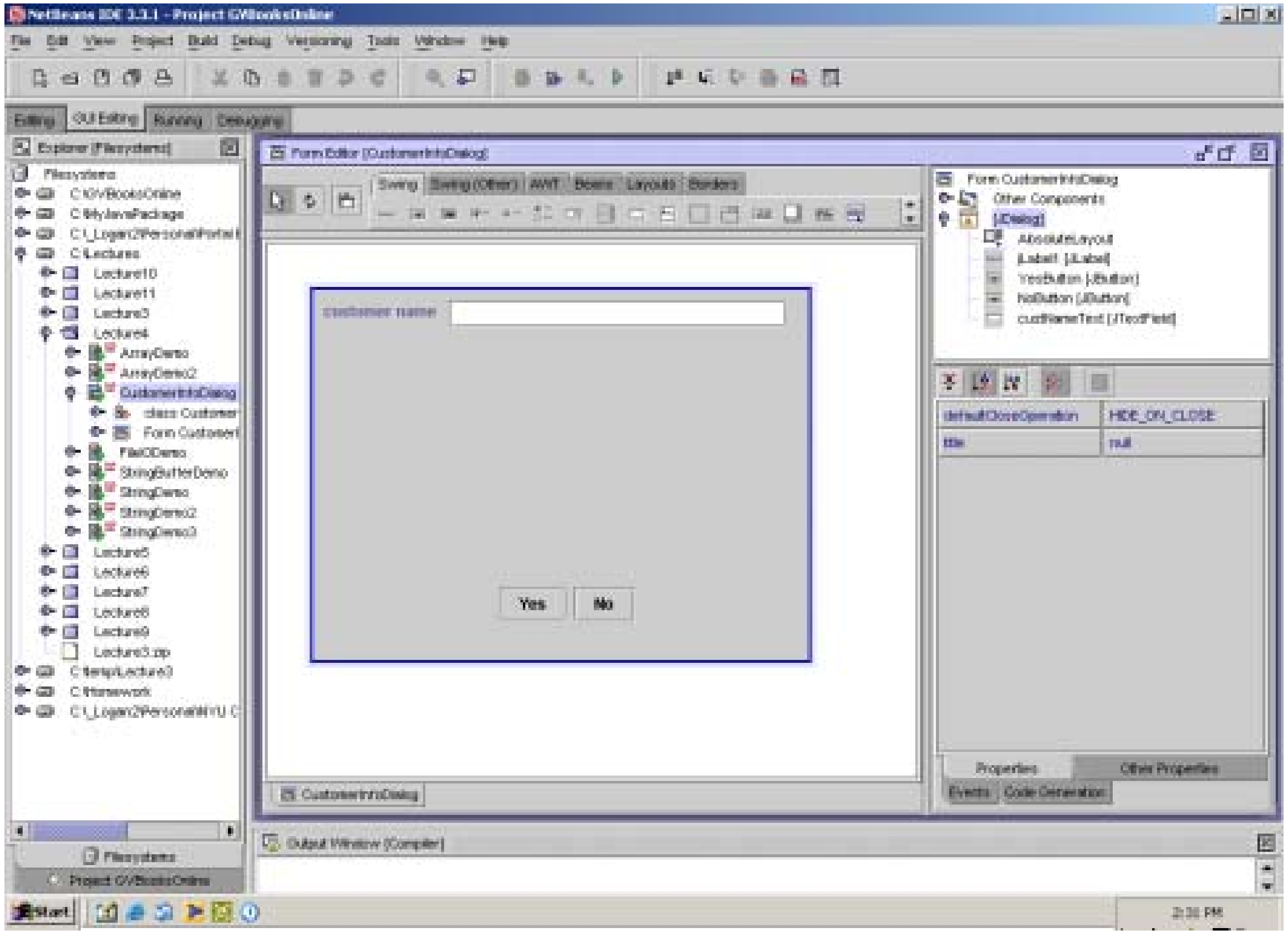
changing the layout manager



what layout manager should I use?

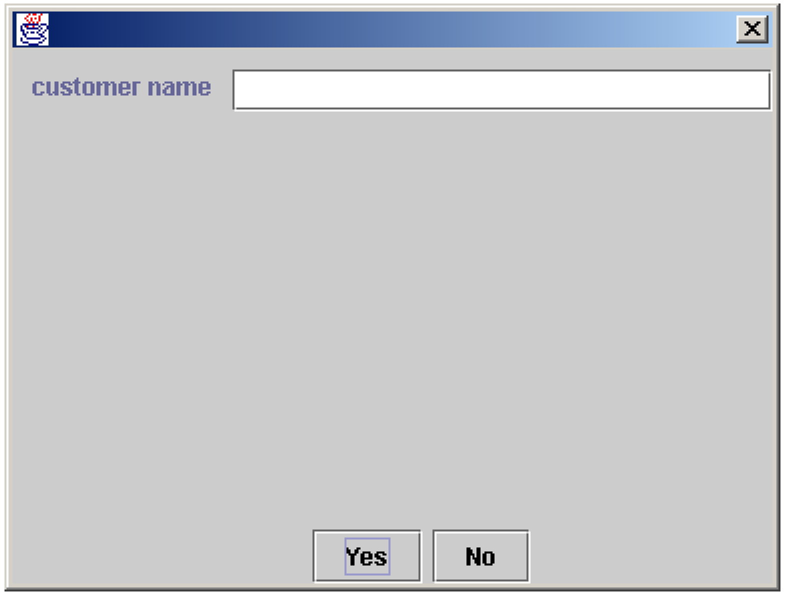
- Start with the absolute and then experiment when you feel comfortable (or hire a graphic artist and let them worry about it ;-).

Adding other components to the view - JTextFields

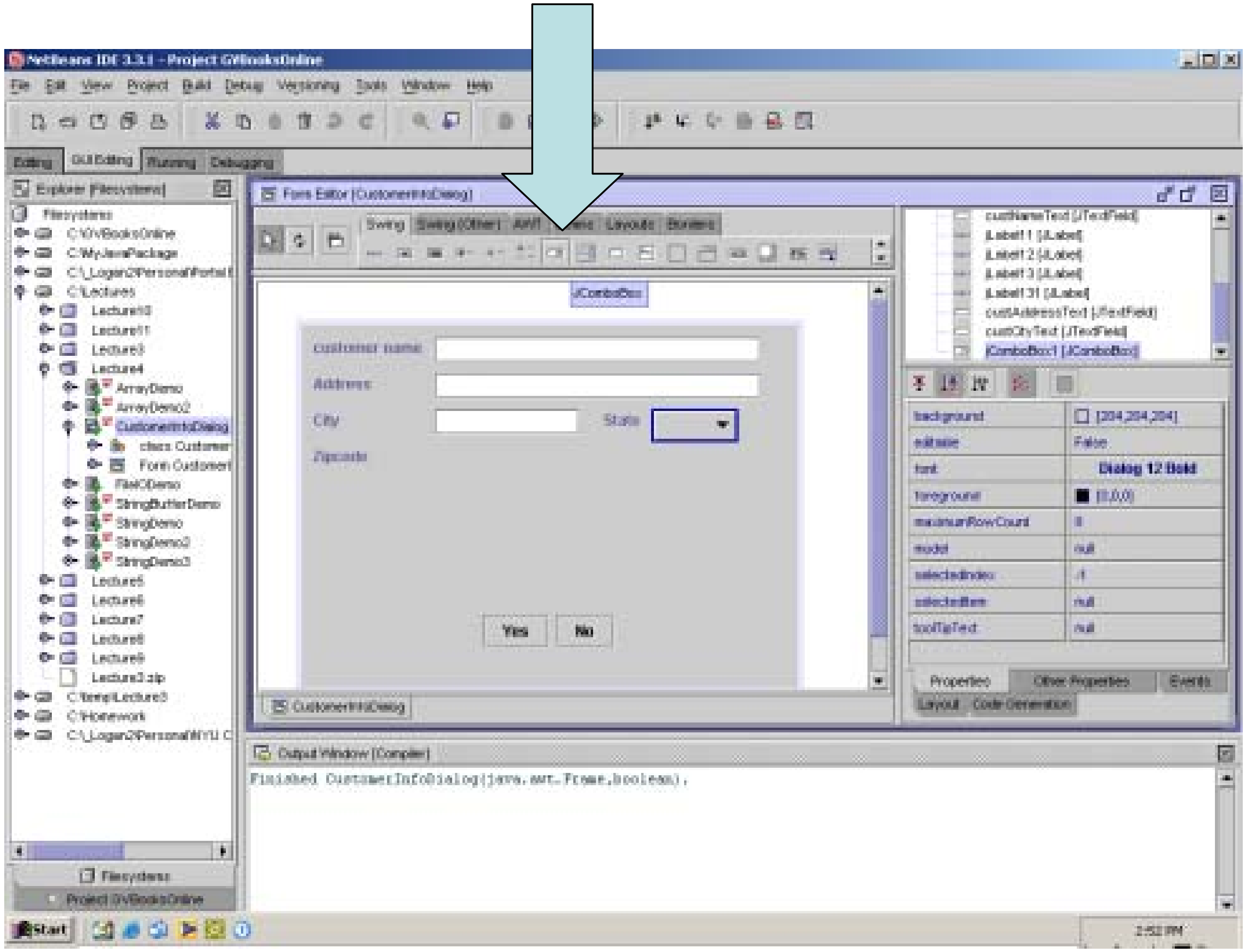

NetBeans IDE 3.3.1 - Project GVBooksOnline
File Edit View Project Build Debug Versioning Tools Window Help
Editing GUI Editing Running Debugging
Explorer [Filesystems]
Form Editor [CustomerInfoDialog]
Swing Swing (Other) AWT Beans Layouts Borders
customer name
Yes
No
Form CustomerInfoDialog
Other Components
[JDialog]
AbsoluteLayout
jLabel1 [JLabel]
YesButton [JButton]
NoButton [JButton]
custNameText [JTextField]
Properties Other Properties Events Code Generation
Output Window [Compiler]
Start

defaultCloseOperation	HIDE_ON_CLOSE
title	null

execute the class



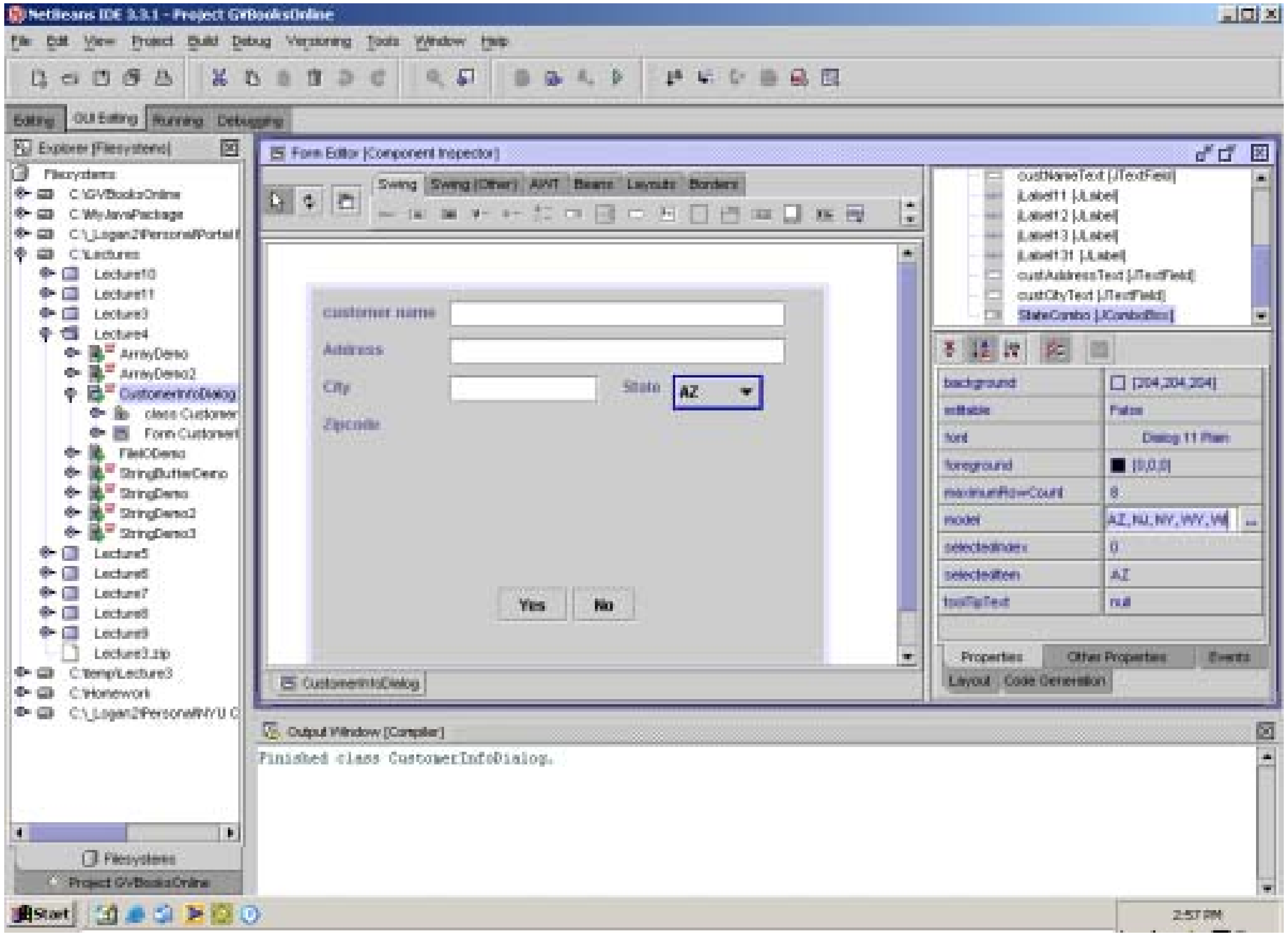
Adding a combo box



background	☐ [204,204,204]
editable	False
font	Dialog 12 Bold
foreground	■ [0,0,0]
maximumRowCount	8
model	null
selectedIndex	-1
selectedItem	null
toolTipText	null

```
Finished CustomerInfoDialog(java.awt.Frame,boolean).
```

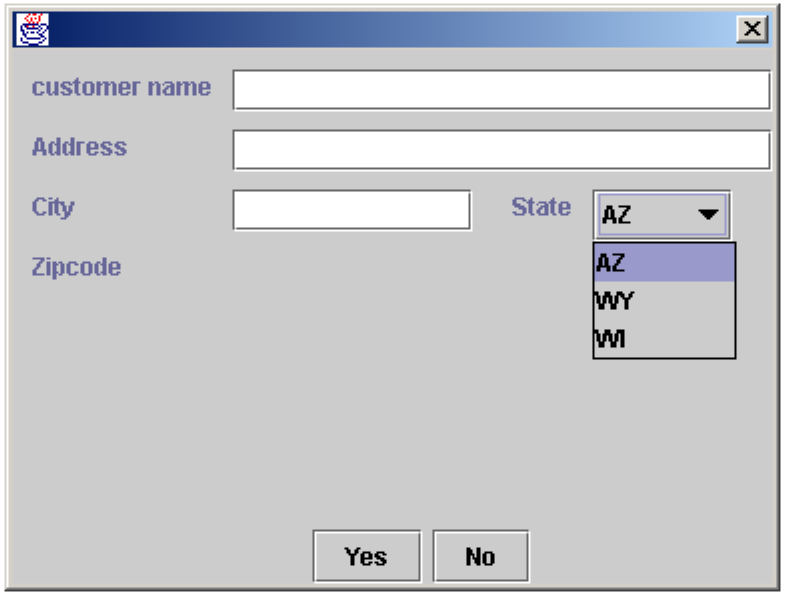
edit the model property for the combo box


NetBeans IDE 3.3.1 - Project GVBooksOnline
File Edit View Project Build Debug Versioning Tools Window Help
Editing GUI Editing Running Debugging
Explorer [Filesystems]
Form Editor [Component Inspector]
Swing Swing (Other) AWT Beans Layouts Borders
customer name
Address
City
State
Zipcode
Yes
No
custNameText [JTextField]
jLabel11 [JLabel]
jLabel12 [JLabel]
jLabel13 [JLabel]
jLabel131 [JLabel]
custAddressText [JTextField]
custCityText [JTextField]
StateCombo [JComboBox]
Properties Other Properties Events Layout Code Generation
Output Window [Compiler]

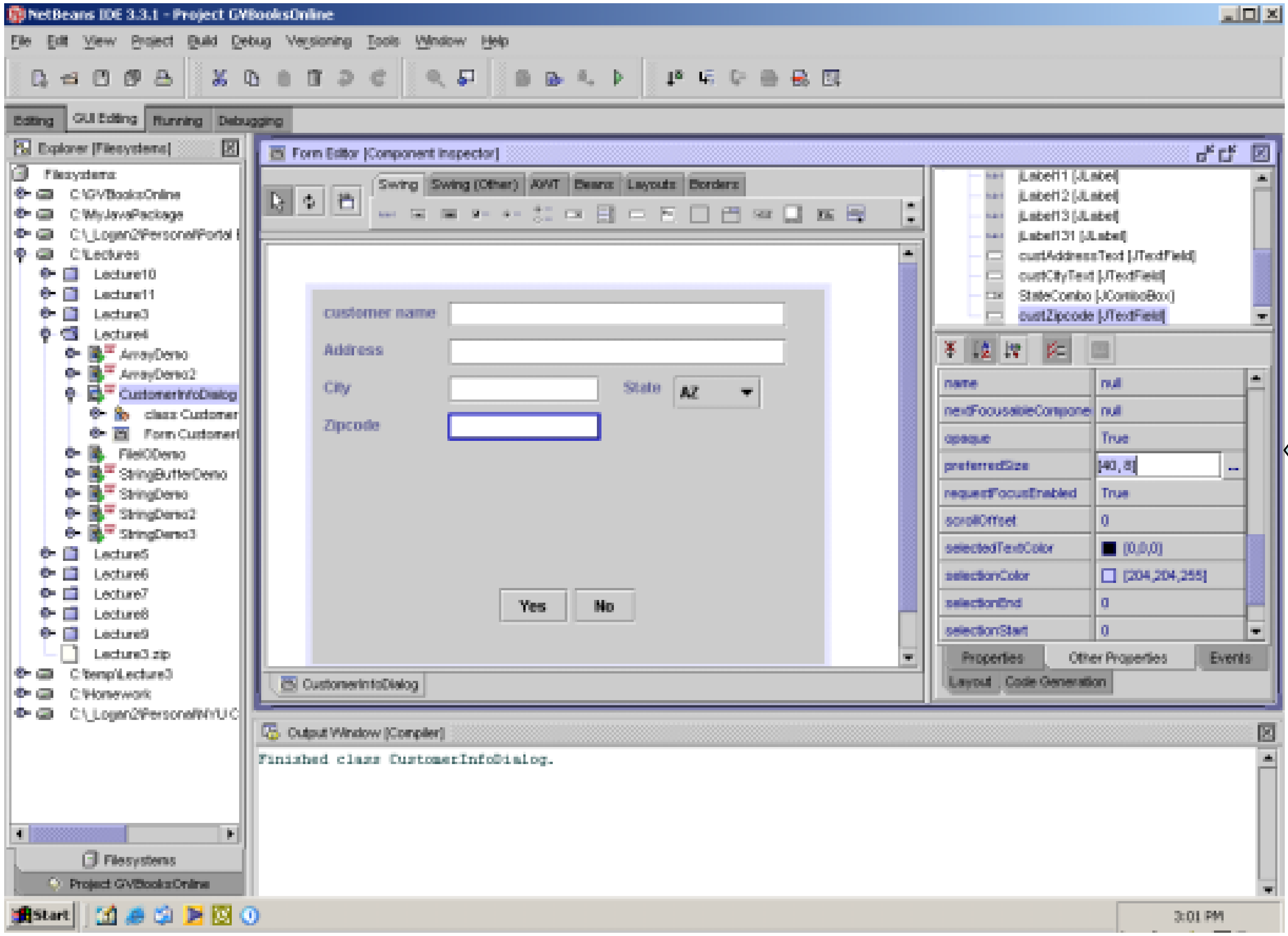
background	☐ [204,204,204]
editable	False
font	Dialog 11 Plain
foreground	■ [0,0,0]
maximumRowCount	8
model	AZ,NJ,NY,WY,WI
selectedIndex	0
selectedItem	AZ
toolTipText	null

```
Finished class CustomerInfoDialog.
```

type in state abbreviations separated by commas


customer name
Address
City
State
AZ
AZ
WY
WI
Zipcode
Yes
No

preferred size property


NetBeans IDE 3.3.1 - Project GVBooksOnline
File Edit View Project Build Debug Versioning Tools Window Help
Editing GUI Editing Running Debugging
Explorer [Filesystems]
Form Editor [Component Inspector]
Swing Swing (Other) AWT Beans Layouts Borders
customer name
Address
City
State
AZ
Zipcode
Yes
No
CustomerInfoDialog
Properties Other Properties Events
Layout Code Generation
Output Window [Compiler]
Filesystems
Project GVBooksOnline
Start
3:01 PM

name	null
nextFocusableComponent	null
opaque	True
preferredSize	[40, 8]
requestFocusEnabled	True
scrollOffset	0
selectedTextColor	■ [0,0,0]
selectionColor	□ [204,204,255]
selectionEnd	0
selectionStart	0

```
Finished class CustomerInfoDialog.
```

hor , vert

MVC

Model – View – Controller Design Pattern

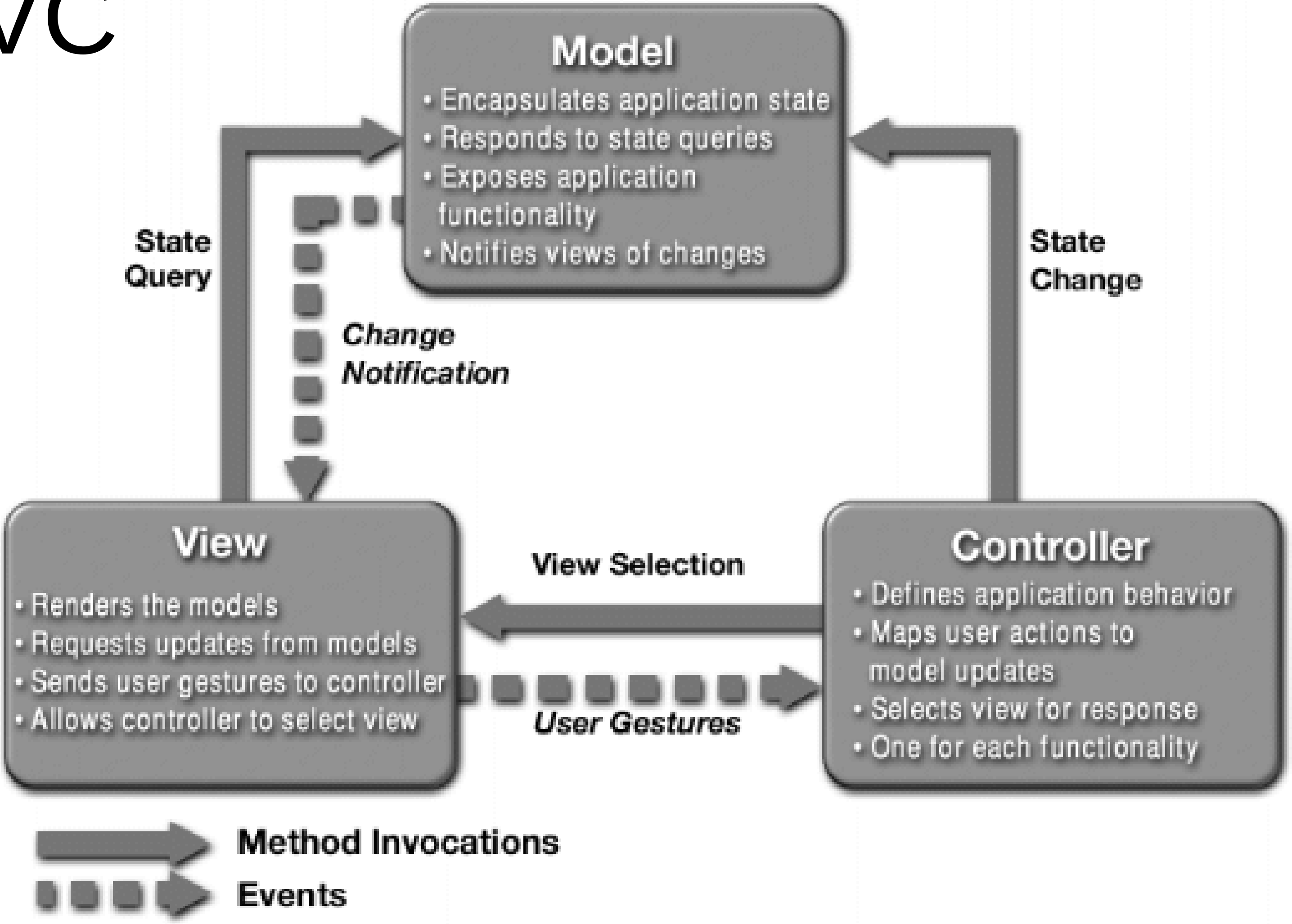
Design Patterns

- A design pattern is a way of designing code that benefits from experience of other developers – see GoF (Gang of Four) on Patterns
- Design patterns are “rules of thumb” & best practices
- A GUI is based on many design patterns
  - 3D Pliancy
  - Feedback
  - Icons
  - Menus
  - Pointing
  - Mnemonics & Accelerators
  - Many more …
- A pattern usually has a name (and several aliases), a context, a problem it addresses, a description of the solution, hints of when to use it and when not to.
- See http://www.csc.calpoly.edu/~dbutler/tutorials/winter96/patterns/ , http://choices.cs.uiuc.edu/sane/dpatterns.html#dp and http://www.stanford.edu/~borchers/hcipatterns

MVC – Model View Controller pattern

- Swing components are designed as MVC components
  - **Model** = data or object that is the to be visually represented
  - **View** = one or more visual representations of that data/object
  - **Controller** = code to manage input to the model

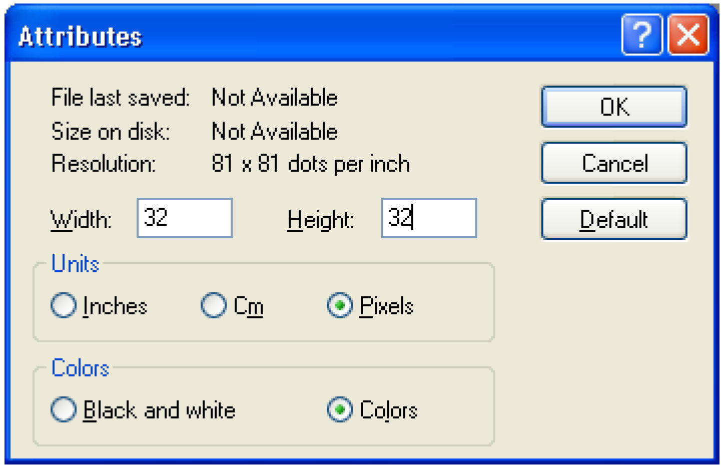
MVC


Model
• Encapsulates application state
• Responds to state queries
• Exposes application functionality
• Notifies views of changes
State Query
Change Notification
State Change
View
• Renders the models
• Requests updates from models
• Sends user gestures to controller
• Allows controller to select view
View Selection
User Gestures
Controller
• Defines application behavior
• Maps user actions to model updates
• Selects view for response
• One for each functionality
Method Invocations
Events

- © Sun 2002

MVC in Swing Components

- The Swing component class is the **view** and **controller**
- A separate class is the **model**
- Most components come with a default model
- You can set the model to your own model for a control
- Several controls could **share** a model!

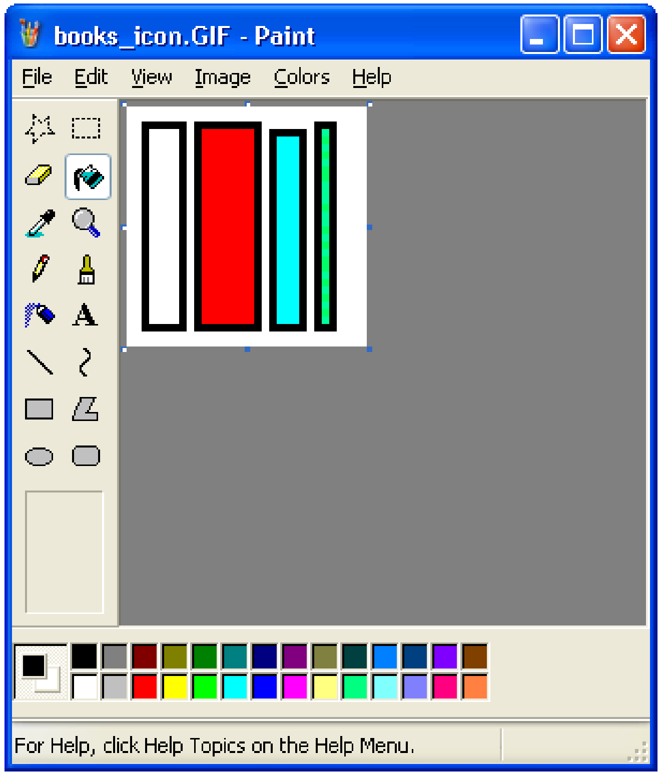

Attributes

File last saved: Not Available
Size on disk: Not Available
Resolution: 81 x 81 dots per inch

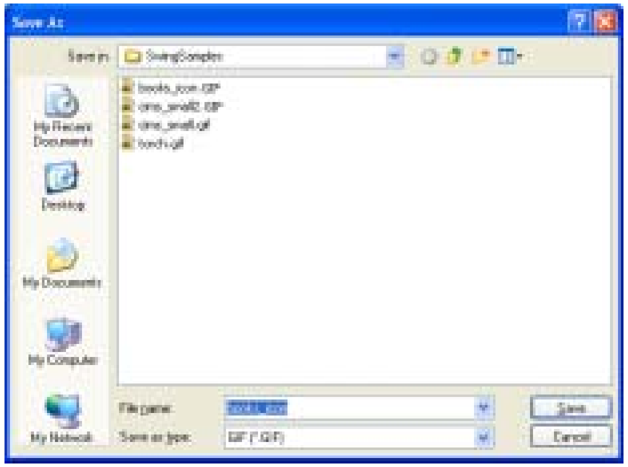
Width: 32 Height: 32

Units: Inches Cm [✓ Pixels]

Colors: Black and white [✓ Colors]


books_icon.GIF - Paint

For Help, click Help Topics on the Help Menu.

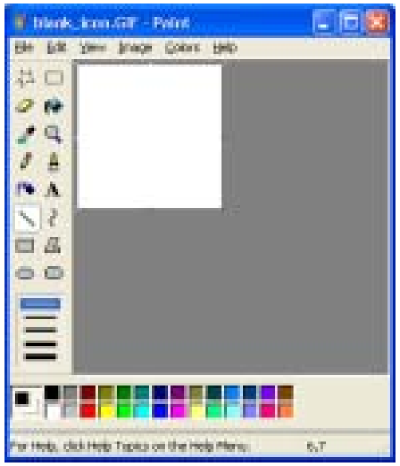

Save As

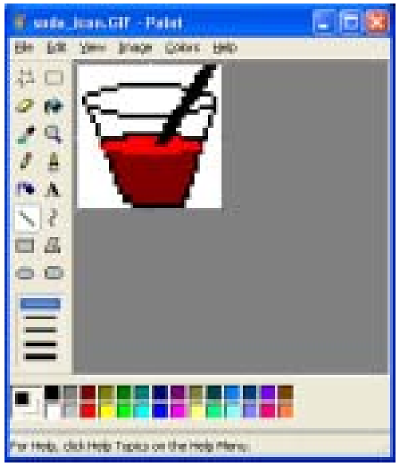
Save in: SwingComplex

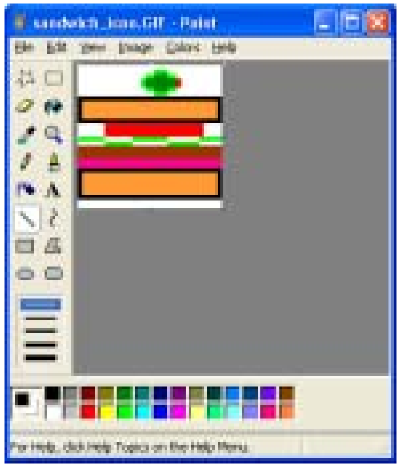
books_icon.GIF
cms_small2.GIF
cms_small.gif
torch.gif

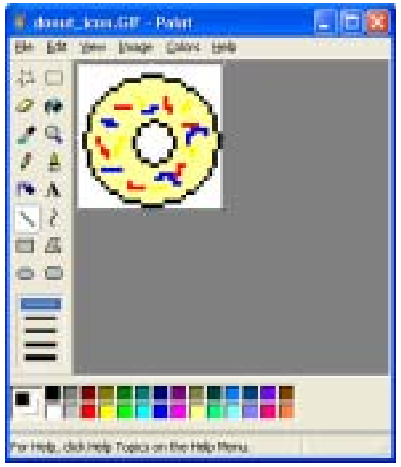
File name: [illegible]
Save as type: GIF (*.GIF)

Creating icons using the blank icon to start with

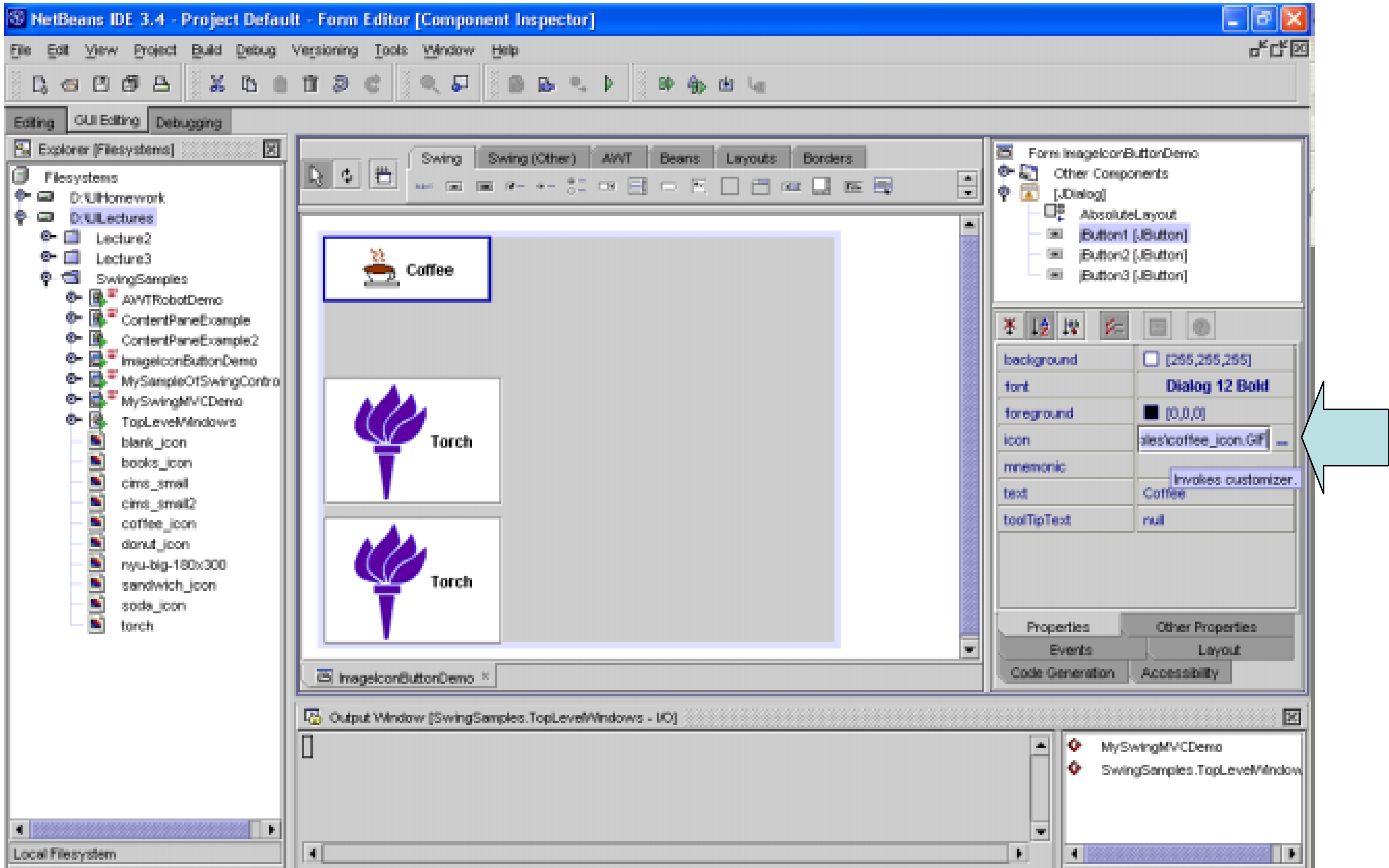

blank_icon.GIF - Paint
File Edit View Image Colors Help
For Help, click Help Topics on the Help Menu.


soda_icon.GIF - Paint
File Edit View Image Colors Help
For Help, click Help Topics on the Help Menu.


sandwich_icon.GIF - Paint
File Edit View Image Colors Help
For Help, click Help Topics on the Help Menu.


donut_icon.GIF - Paint
File Edit View Image Colors Help
For Help, click Help Topics on the Help Menu.

Change the properties of the button to use you icon


NetBeans IDE 3.4 - Project Default - Form Editor [Component Inspector]
File Edit View Project Build Debug Versioning Tools Window Help
Editing GUI Editing Debugging
Explorer [Filesystems]
Swing Swing (Other) AWT Beans Layouts Borders
Coffee
Torch
Torch
ImageIconButtonDemo
Properties Other Properties
Events Layout
Code Generation Accessibility
Output Window [SwingSamples.TopLevelWindows - I/O]
Local Filesystem

Filesystems
- D:\UIHomework
- D:\UILectures
  - Lecture2
  - Lecture3
  - SwingSamples
    - AWTRobotDemo
    - ContentPaneExample
    - ContentPaneExample2
    - ImageIconButtonDemo
    - MySampleOfSwingContro
    - MySwingMVCDemo
    - TopLevelWindows
    - blank_icon
    - books_icon
    - cims_small
    - cims_small2
    - coffee_icon
    - donut_icon
    - nyu-big-180x300
    - sandwich_icon
    - soda_icon
    - torch

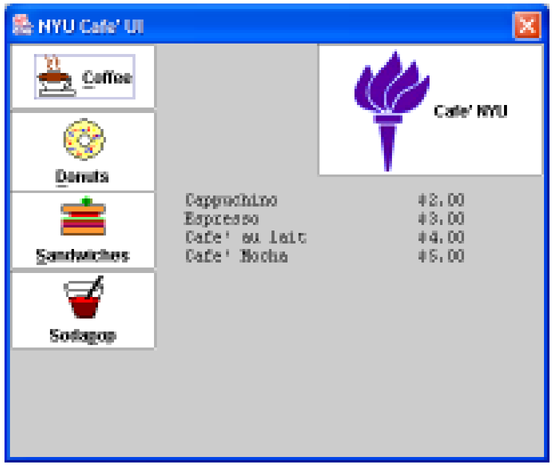
Form ImageIconButtonDemo
- Other Components
- [JDialog]
  - AbsoluteLayout
  - jButton1 [JButton]
  - jButton2 [JButton]
  - jButton3 [JButton]

background	[255,255,255]
font	Dialog 12 Bold
foreground	[0,0,0]
icon	ples\coffee_icon.GIF ...
mnemonic	
text	Coffee
toolTipText	null

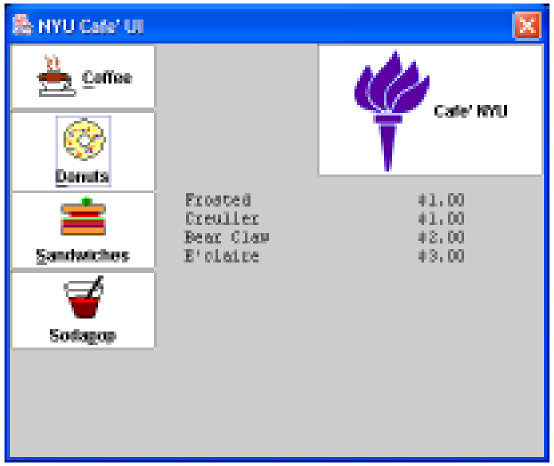
Invokes customizer.

MySwingMVCDemo
SwingSamples.TopLevelWindow

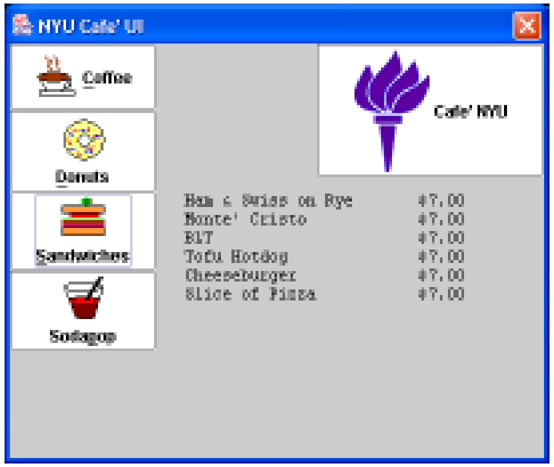
Pushing the buttons changes the displayed prices.



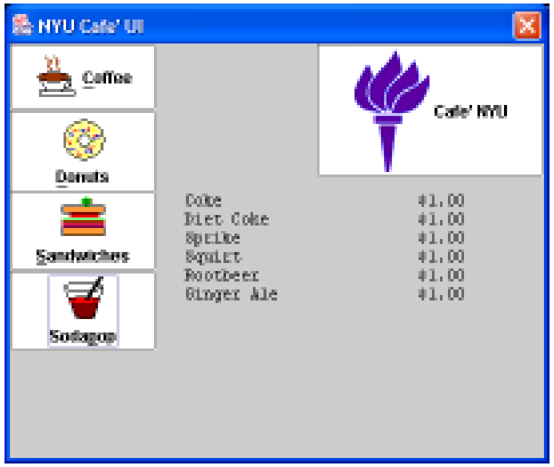
```
Cappuchino          $2.00
Espresso            $3.00
Cafe' au lait       $4.00
Cafe' Mocha         $5.00
```



```
Frosted             $1.00
Creuller            $1.00
Bear Claw           $2.00
E'claire            $3.00
```

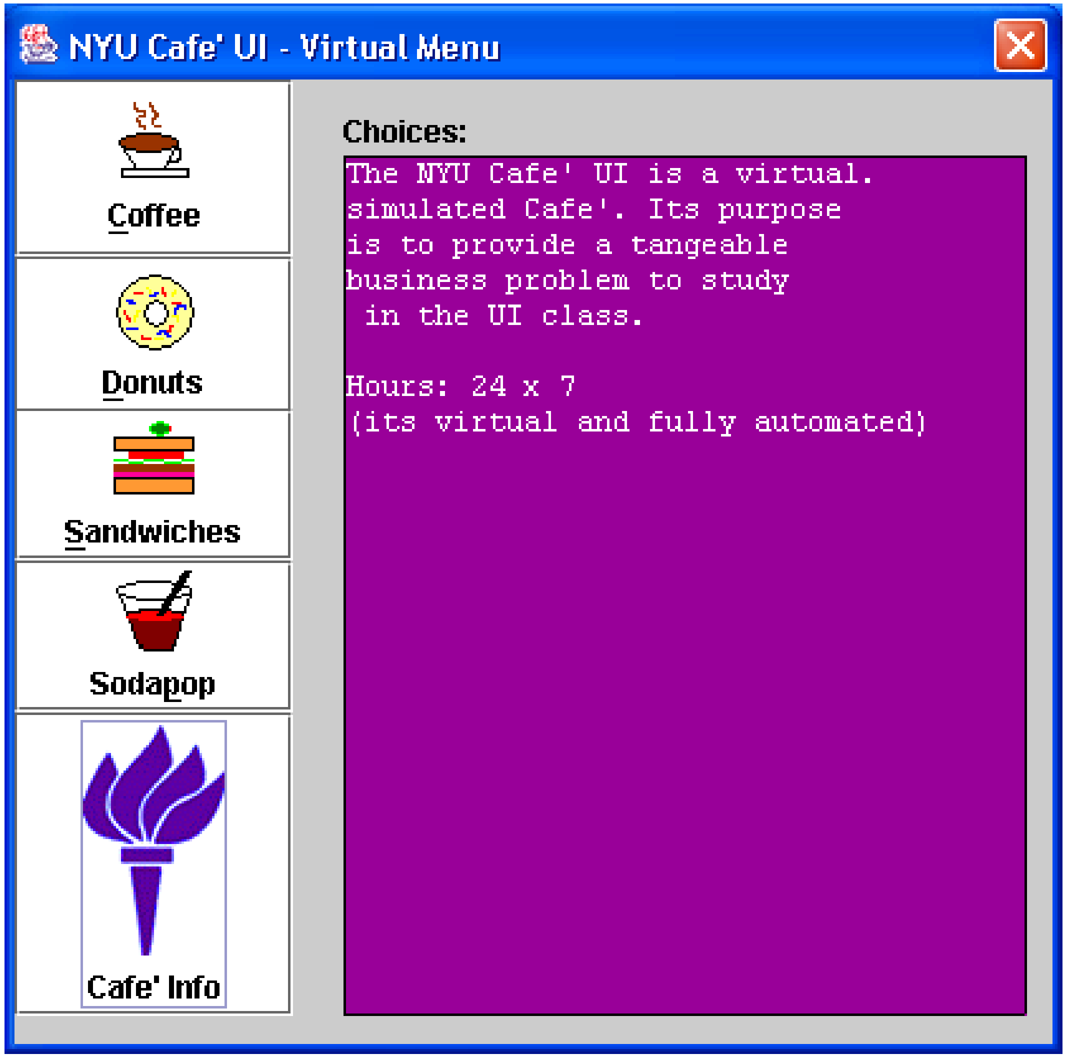


```
Ham & Swiss on Rye  $7.00
Monte' Cristo       $7.00
BLT                 $7.00
Tofu Hotdog         $7.00
Cheeseburger        $7.00
Slice of Pizza      $7.00
```

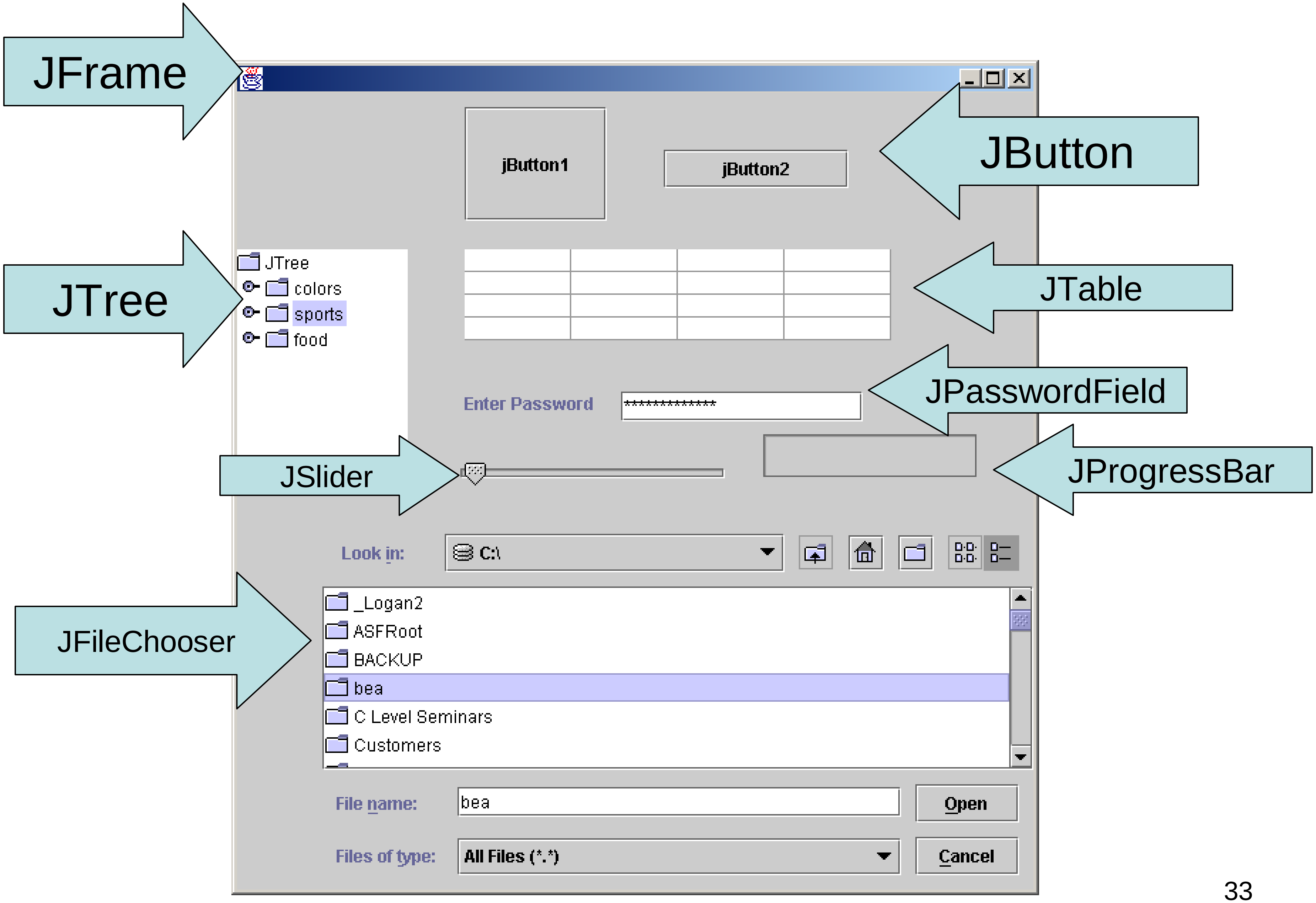


```
Coke                $1.00
Diet Coke           $1.00
Sprite              $1.00
Squirt              $1.00
Rootbeer            $1.00
Ginger Ale          $1.00
```

A different and better layout

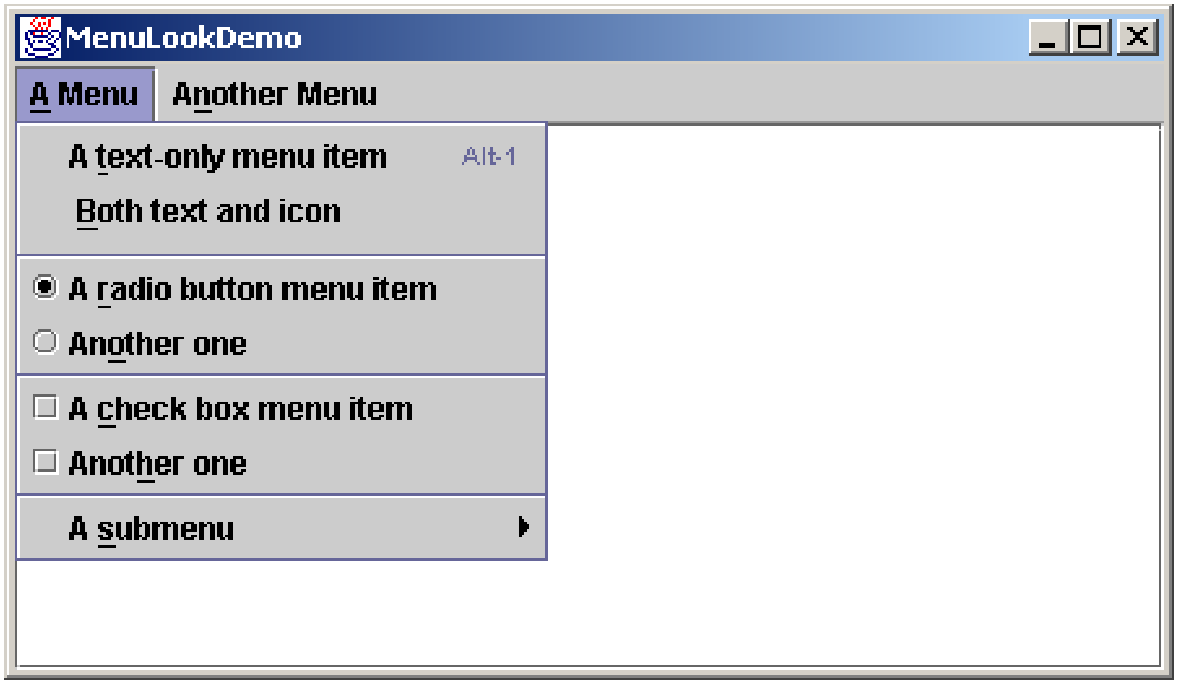


```
The NYU Cafe' UI is a virtual.
simulated Cafe'. Its purpose
is to provide a tangeable
business problem to study
 in the UI class.

Hours: 24 x 7
(its virtual and fully automated)
```

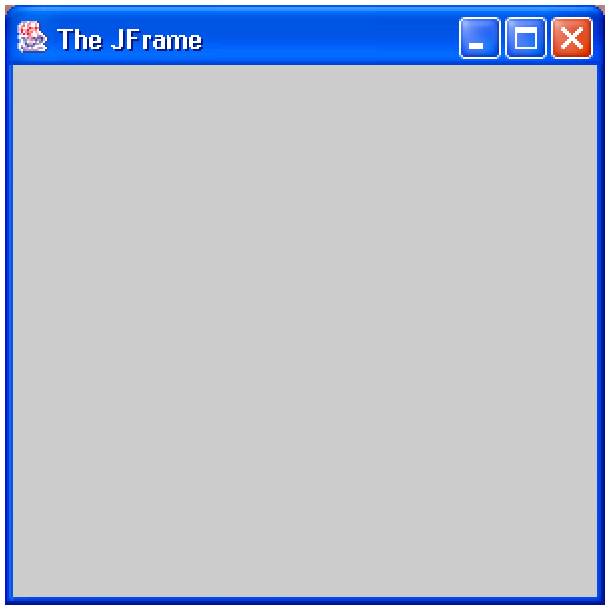

JFrame
JButton
jButton1
jButton2
JTree
colors
sports
food
JTable
Enter Password
****
JPasswordField
JSlider
JProgressBar
Look in:
C:\
JFileChooser
_Logan2
ASFRoot
BACKUP
bea
C Level Seminars
Customers
File name:
bea
Open
Files of type:
All Files (*.*)
Cancel

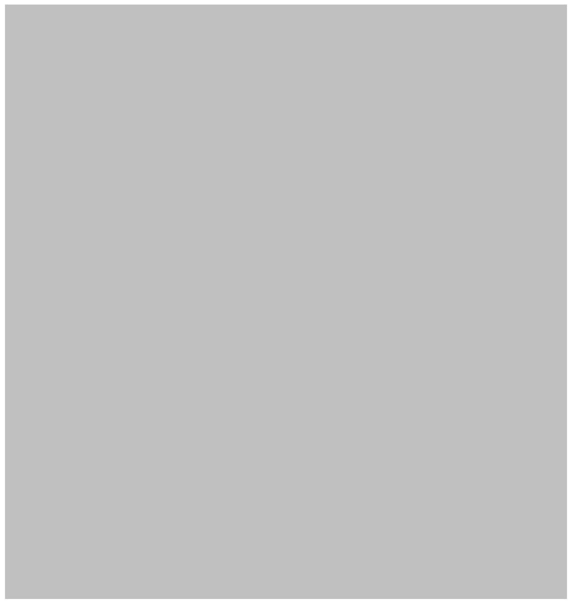
```
private javax.swing.JPasswordField     jPasswordField1;
private javax.swing.JTree              jTree1;
private javax.swing.JSlider            jSlider1;
private javax.swing.JProgressBar       jProgressBar1;
private javax.swing.JTable             jTable1;
private javax.swing.JButton            jButton2;
private javax.swing.JButton            jButton1;
private javax.swing.JFileChooser       jFileChooser1;
private javax.swing.JLabel             jLabel1;
```

Swing based `MenuLookDemo`



TopLevelWindows.java





TopLevelWindows.java

```
package SwingSamples;
import javax.swing.*;
public class TopLevelWindows
{
    public static void main(String args[])
    {
        JFrame myJFrame = new JFrame("The JFrame");
        myJFrame.setSize(300,300);
        myJFrame.setLocation(100,100);

        JWindow myJWindow = new JWindow();
        myJWindow.setSize(300,300);
        myJWindow.setLocation(500, 100);

        myJFrame.setVisible(true);
        myJWindow.setVisible(true);

    }
}
```

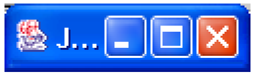
Top Level Containers

- Must have a top level container in Swing
- You must add components to the associated content pane

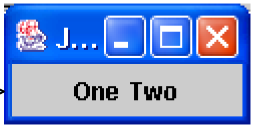
ContentPaneExample.java

```
package SwingSamples;

import java.awt.*;
import javax.swing.*;

public class ContentPaneExample
{
    public static void main(String args[])
    {
        JFrame myJFrame = new JFrame("JFrame");
        myJFrame.setLocation(100,100);

        Container myContentPane = myJFrame.getContentPane();
        myContentPane.setLayout(new FlowLayout());
        myContentPane.add(new JLabel("One"));
        myContentPane.add(new JLabel("Two"));

        myJFrame.pack(); //reformats the layout to the minimum size to fit
   everything
        myJFrame.setVisible(true);
    }

}
```

Without the pack()

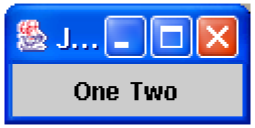


With the pack()



ContentPaneExample2.java

```
package SwingSamples;

import java.awt.*;
import javax.swing.*;

public class ContentPaneExample2
{

    public static void main(String args[])
    {
        JFrame myJFrame = new JFrame("JFrame");
        myJFrame.setLocation(100,100);

        Container myContentPane = new JPanel();

        myContentPane.add(new JLabel("One"));
        myContentPane.add(new JLabel("Two"));

        myJFrame.setContentPane(myContentPane);
        myJFrame.pack();
        myJFrame.setVisible(true);
    }
}
```



Events

- Swing uses them to communicate between swing components.
- An event is just a method call on the receiving object by the sending object. The method passes the event object.

  ```
  addActionListener(ActionListener listener);
  removeActionListener(ActionListener listener);
  ```

- An object registers to receive events. The method that gets called is:

  ```
  actionPerformed(ActionEvent e);
  ```

Events

- In Swing they are multicast – 1 to many possible. Manes multiple method calls by the send basically.
- Order isn't defined, though.
- Events are immutable to the receiver.
- Events may be queued as in the keyboard event queue.
- Multiple events maybe compressed into one as in mouse movements.

Event Modifier Flags

- `SHIFT_MASK`
- `CTRL_MASK`
- `META_MASK`
- `ALT_MASK`
- `BUTTON1_MASK`
- `BUTTON2_MASK`
- `BUTTON3_MASK`
- Detect when certain keys are also pressed.

```
int modifierFlags = myEvent.getModifiers();
if ((modifierFlags & InputEvent.CRTL_MASK)!=0)
    System.println.out(“Pressing the contrl key”);
```

Event Types

- `ComponentEvent  //resized,moved, shown, hidden`
- `FocusEvent      //gained, lost`
- `KeyEvent        //typed, pressed, released`
- `MouseEvent      //clicked, pressed, released,`
  `                //entered, exited`
- `ContainerEvent  //componentAdded  componentRemoved`
- `ActionEvent     //fired by: JButton, JChekBox, …`
- `AdjustmentEvent //fired by: JScrollBar`
- Many more ….

Event Adapter Classes

- Map incoming events to a method to invoke on the model to achieve the function.
- Separates the View & Controller from the Model (MVC)
- Prebuilt adapter has stubbed out methods for events. You only implement the ones you are interested. You do this by extending the adapter and overiding the methods you need.
- Follows a general design pattern of called “adapter”.
- `MouseAdapter, MouseInputAdapter, MouseMotionAdapter, KeyAdapter, ComponentAdapter, ContainerAdapter, DragSourceAdapter, DropTargetAdapter, FocusAdapter, WindowAdapter, …`

AWT Robot!

- Used to simulate keyboard and mouse programmatically.
- It places events in the native system queues for the platform you are on (not just the java queue).
- Used for recording and replaying activities in regression testing and other uses.

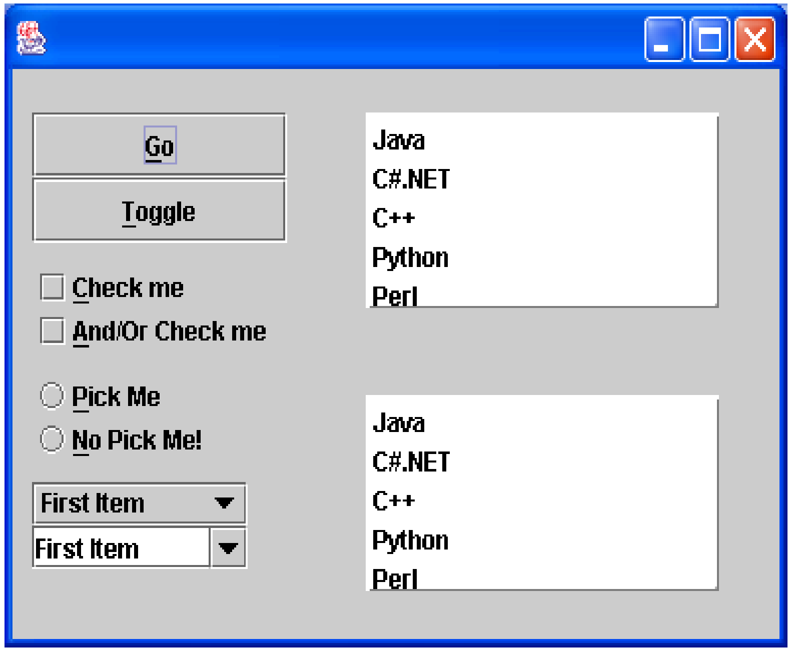
Multithreading and Swing

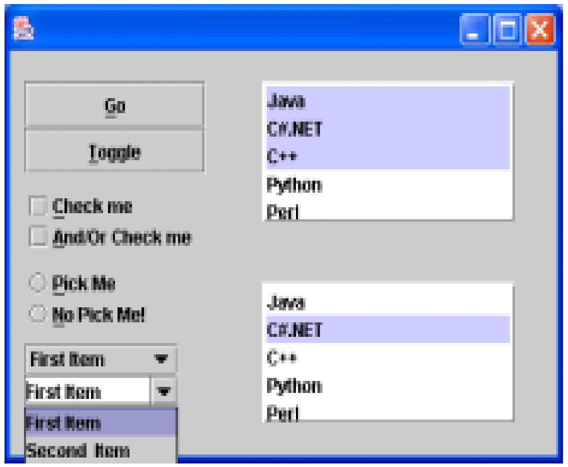
- Swing components always execute on a single thread within your application. Not the main thread of your application, either.
- Swing components are NOT multithread safe!
- This is done for speed but influences how you must design for them.
- We can ignore this for protoyping UIs but not for design of applications.

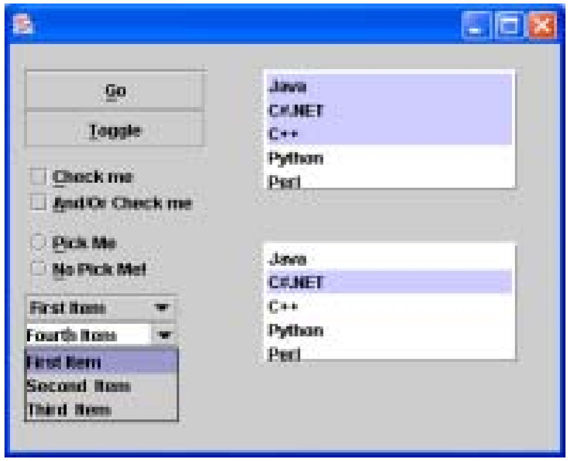
Swing Components

Sample dialog with a few controls.

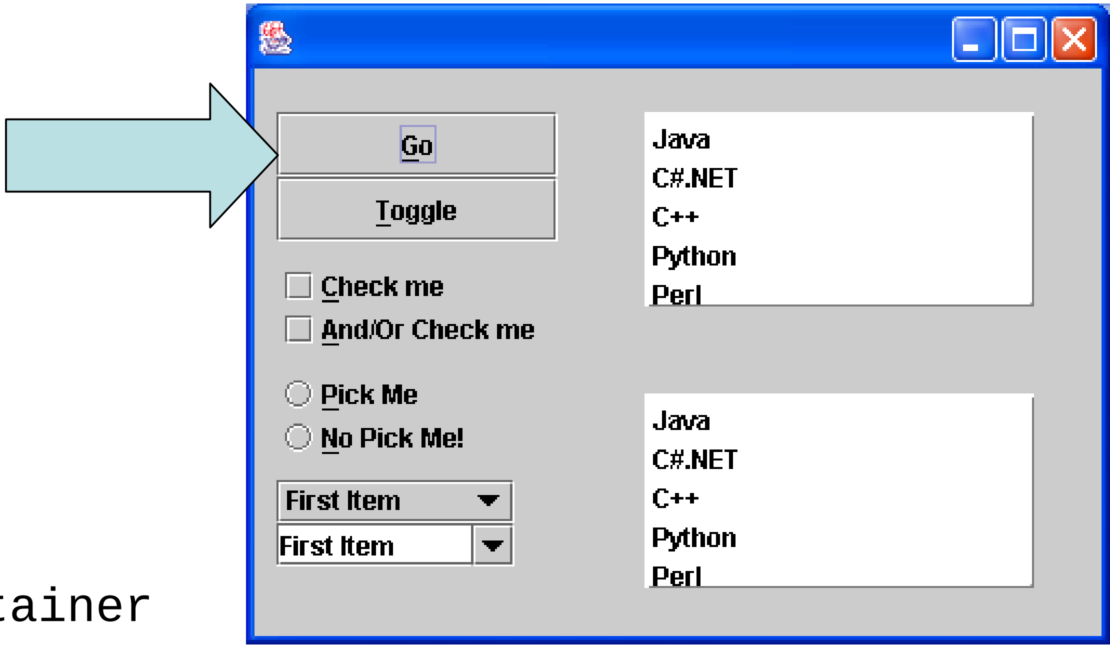
`MySampleOfSwingControls1.java`







JButton

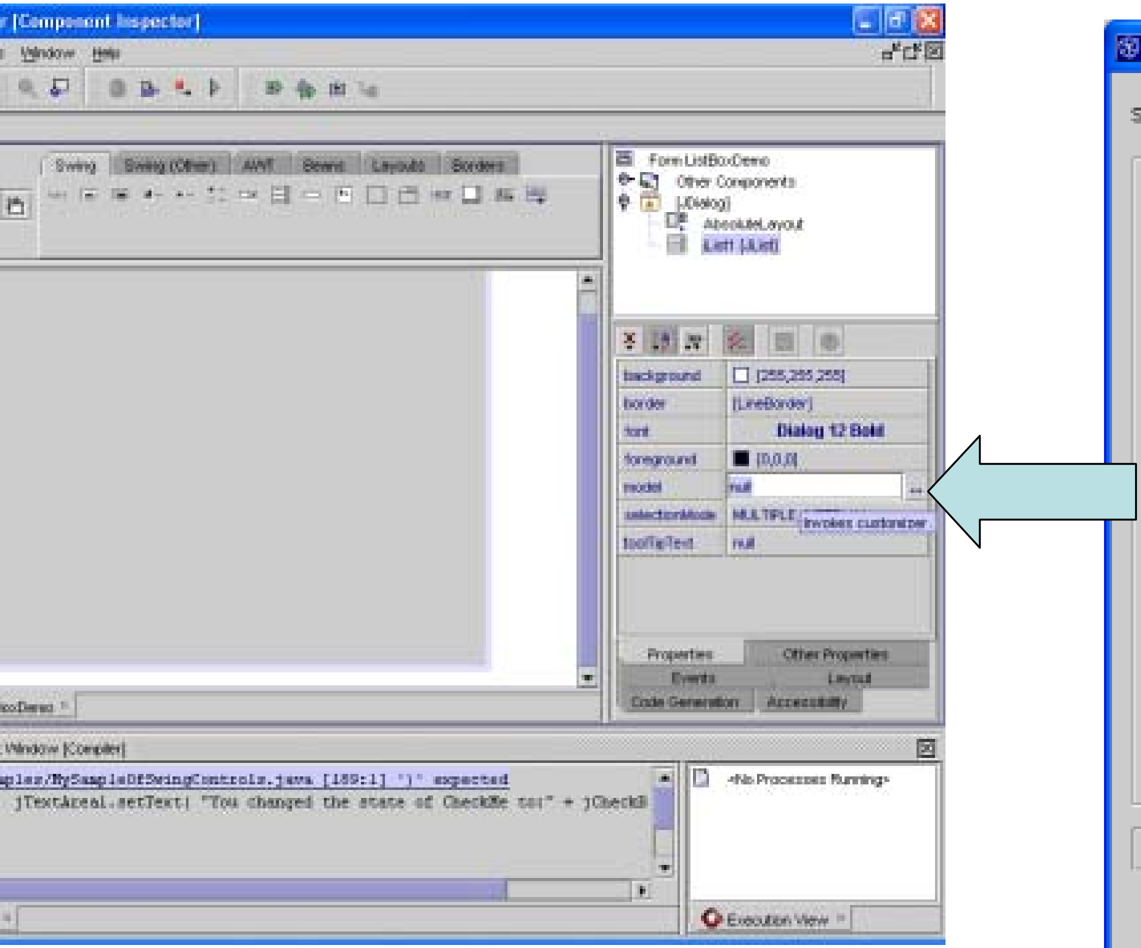


```
java.lang.Object
   |
   +--java.awt.Component
         |
         +--java.awt.Container
               |
               +--javax.swing.JComponent
                     |
                     +--javax.swing.AbstractButton
                           |
                           +--javax.swing.JButton
```

JButton

- Used for a command
- Push and shows a state change visually (pliancy)
- Has a name, label text,

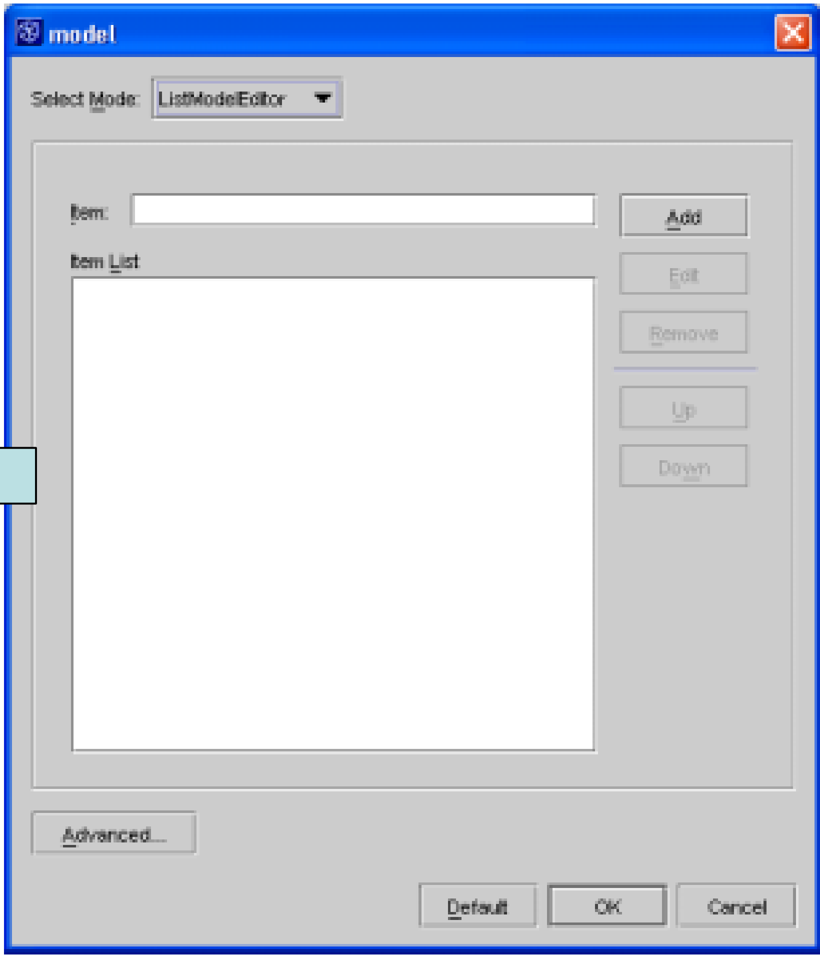
Adding items to the List



background	[255,255,255]
border	[LineBorder]
font	Dialog 12 Bold
foreground	[0,0,0]
model	null
selectionMode	MULTIPLE
toolTipText	null

Invokes customizer

```
mples/MySampleOfSwingControls.java [189:1] ')' expected
 jTextArea1.setText( "You changed the state of CheckMe to:" + jCheckB
```

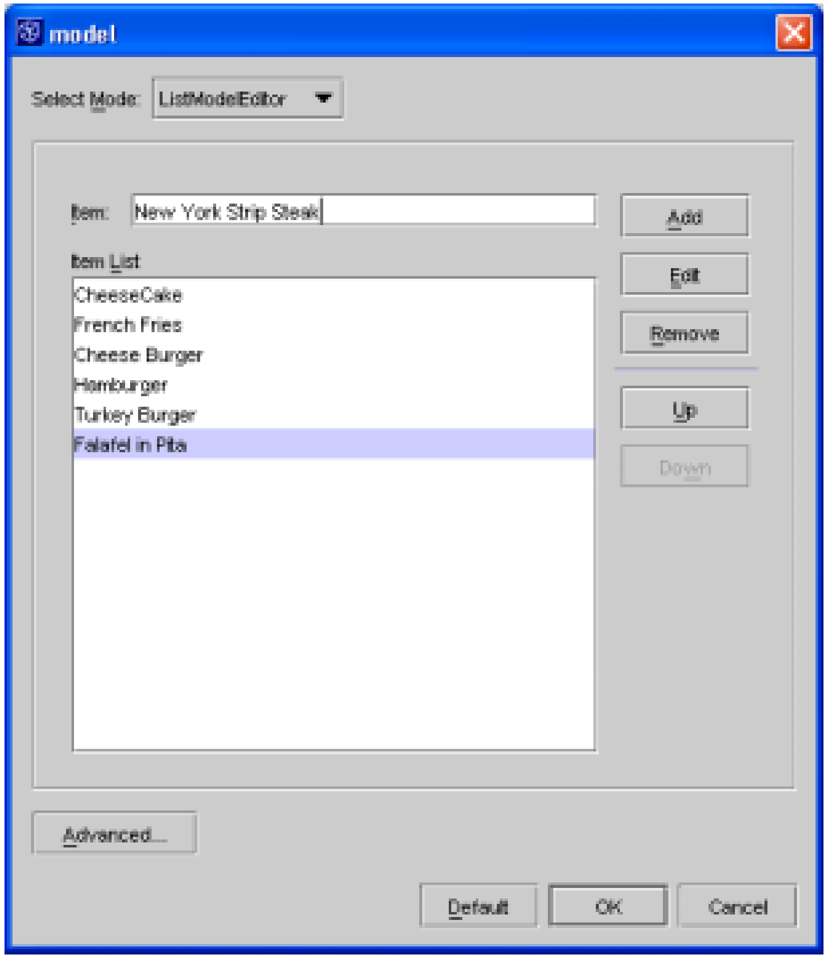


Select Mode: ListModelEditor

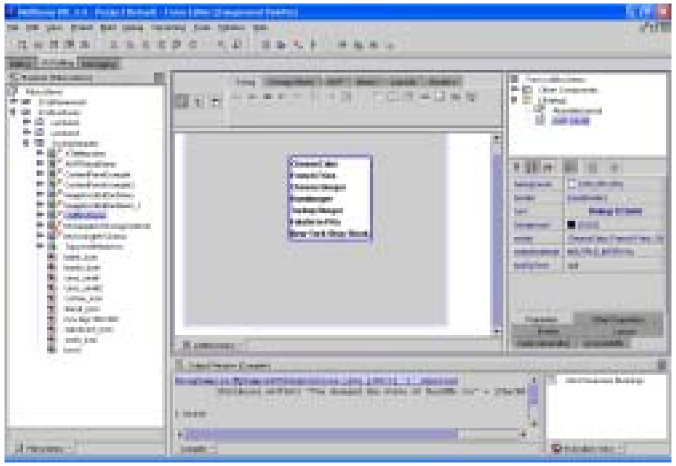
Item:

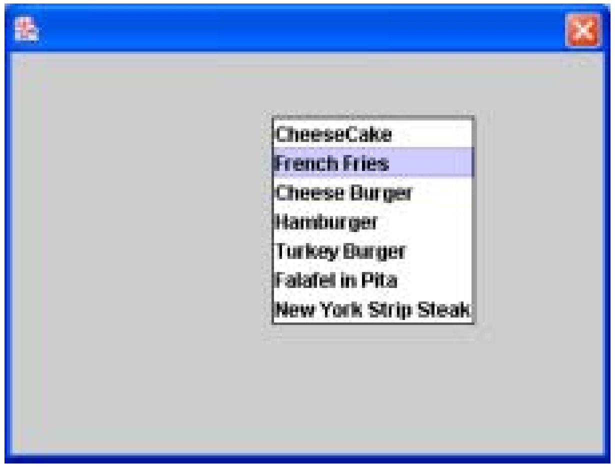
Item List

Add items to the model for the list



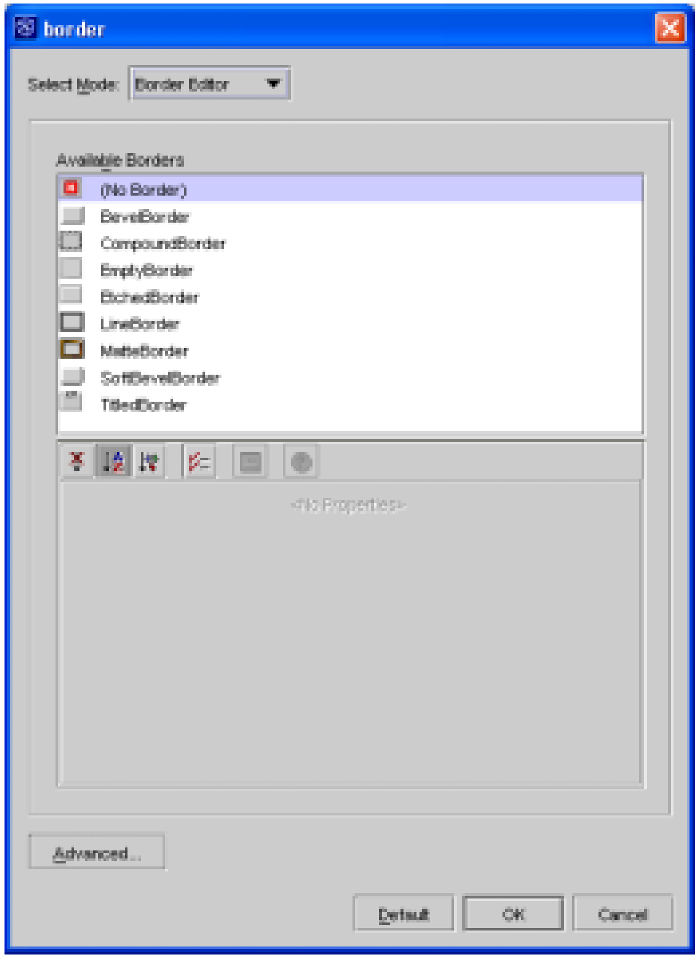
CheeseCake
French Fries
Cheese Burger
Hamburger
Turkey Burger
Falafel in Pita

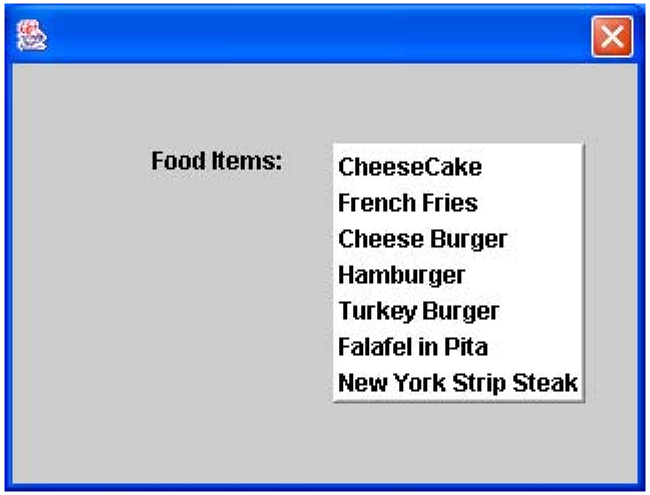




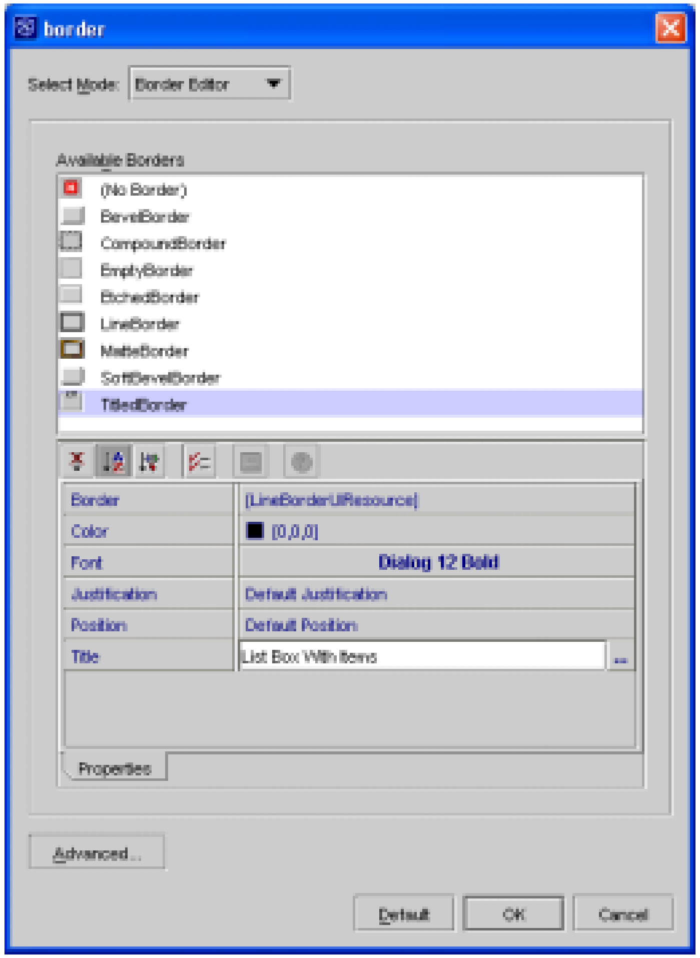
CheeseCake
French Fries
Cheese Burger
Hamburger
Turkey Burger
Falafel in Pita
New York Strip Steak

Changing the border of a list box





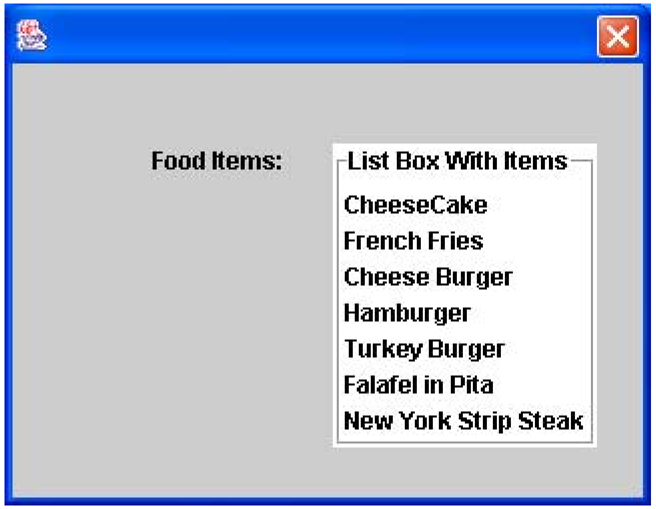
A Titled Border for a List Box


border

Select Mode: Border Editor

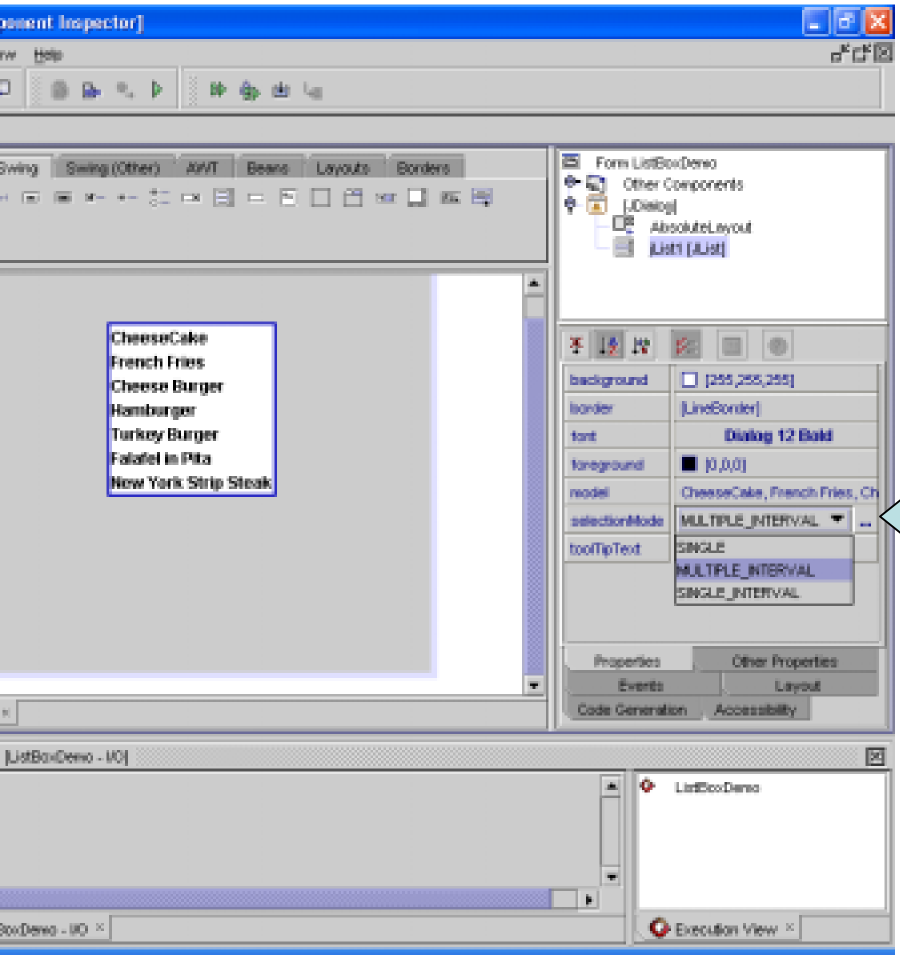
Available Borders
- (No Border)
- BevelBorder
- CompoundBorder
- EmptyBorder
- EtchedBorder
- LineBorder
- MatteBorder
- SoftBevelBorder
- TitledBorder

Border	[LineBorderUIResource]
Color	■ [0,0,0]
Font	Dialog 12 Bold
Justification	Default Justification
Position	Default Position
Title	List Box With Items


Food Items:

List Box With Items
- CheeseCake
- French Fries
- Cheese Burger
- Hamburger
- Turkey Burger
- Falafel in Pita
- New York Strip Steak

List Box Selection Modes



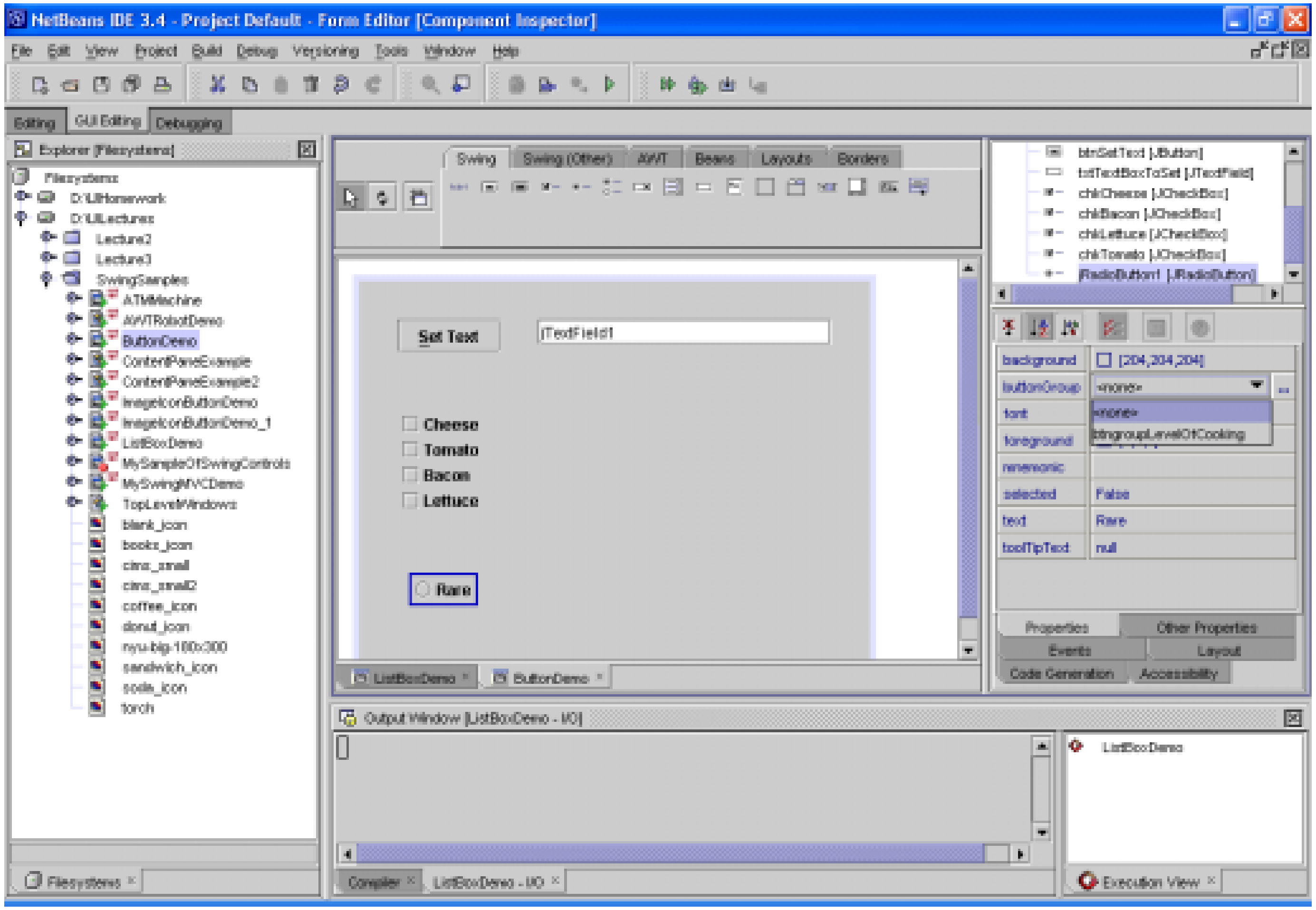
CheeseCake
French Fries
Cheese Burger
Hamburger
Turkey Burger
Falafel in Pita
New York Strip Steak

background	[255,255,255]
border	[LineBorder]
font	Dialog 12 Bold
foreground	[0,0,0]
model	CheeseCake, French Fries, Ch
selectionMode	MULTIPLE_INTERVAL
toolTipText	

SINGLE
MULTIPLE_INTERVAL
SINGLE_INTERVAL

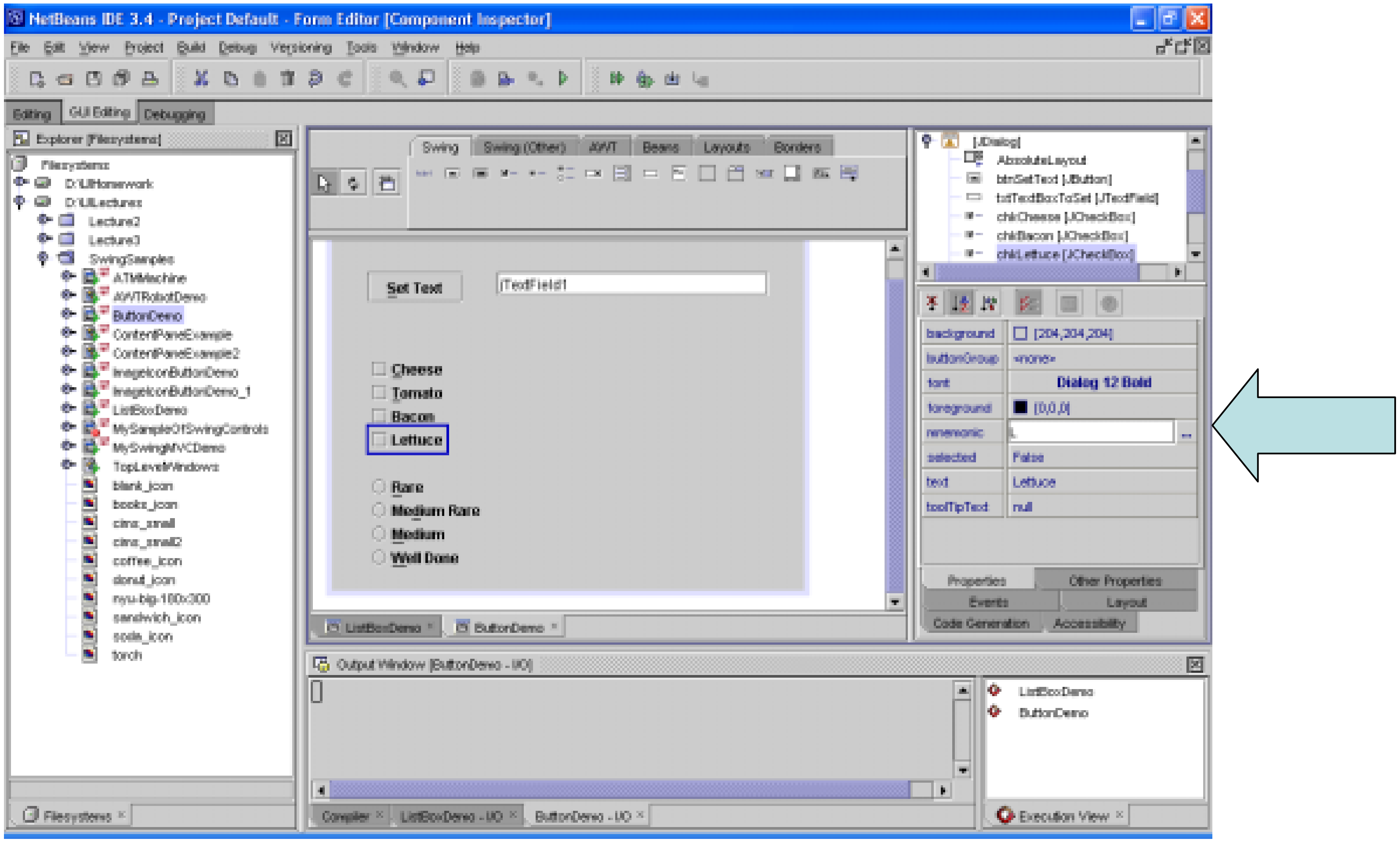
- Single
- Multiple_Interval
- Single_Interval

Setting the Button group on a radio button



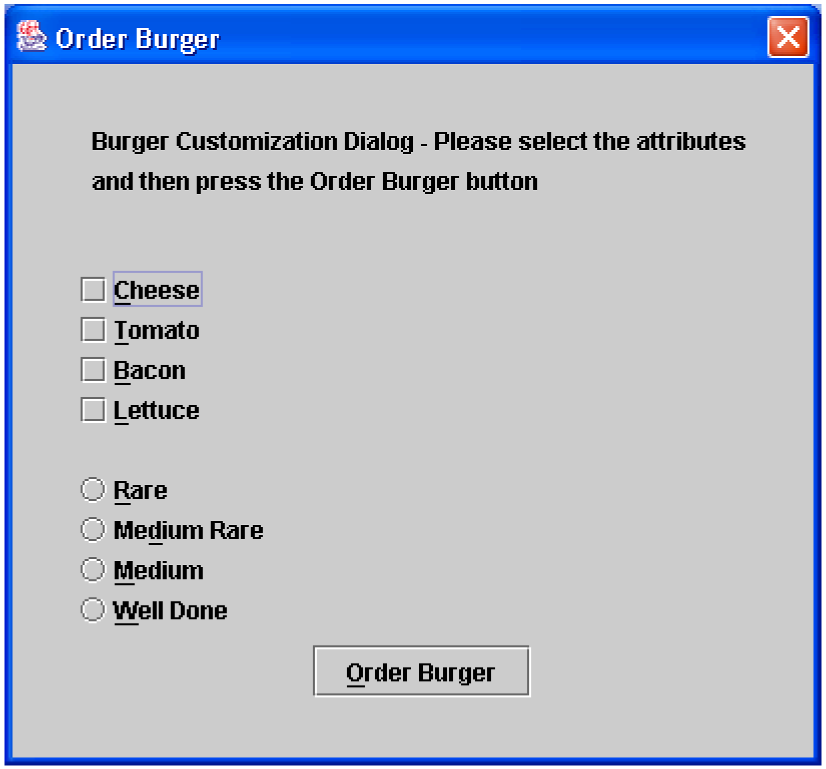
background	[204,204,204]
buttonGroup	<none>
	<none>
	btngroupLevelOfCooking
font	
foreground	
mnemonic	
selected	False
text	Rare
toolTipText	null

Setting the Mnemonics



background	[204,204,204]
buttonGroup	<none>
font	Dialog 12 Bold
foreground	[0,0,0]
mnemonic	L
selected	False
text	Lettuce
toolTipText	null

Final Dialog Version



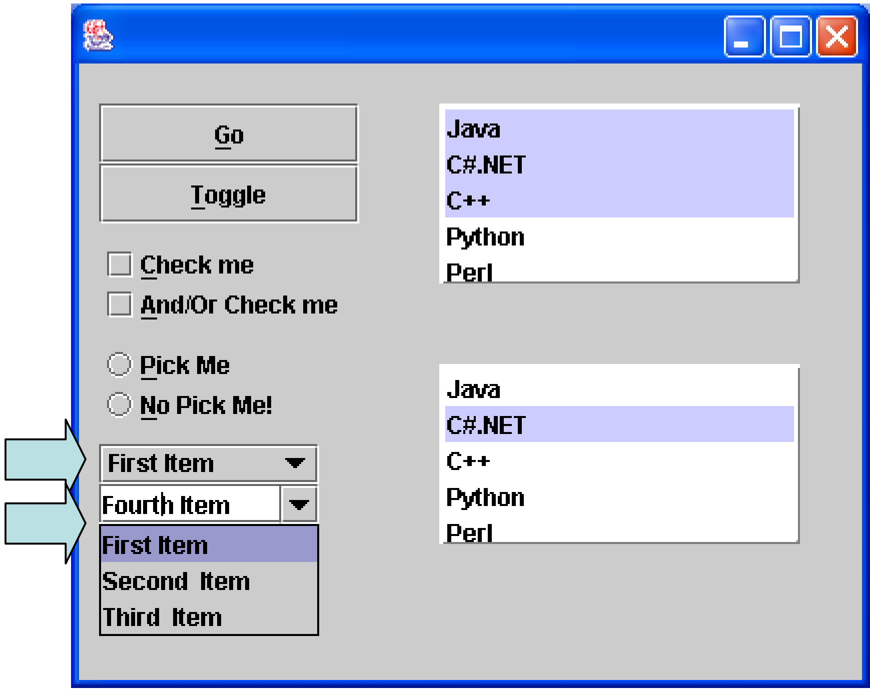
Burger Customization Dialog - Please select the attributes and then press the Order Burger button

- [ ] Cheese
- [ ] Tomato
- [ ] Bacon
- [ ] Lettuce

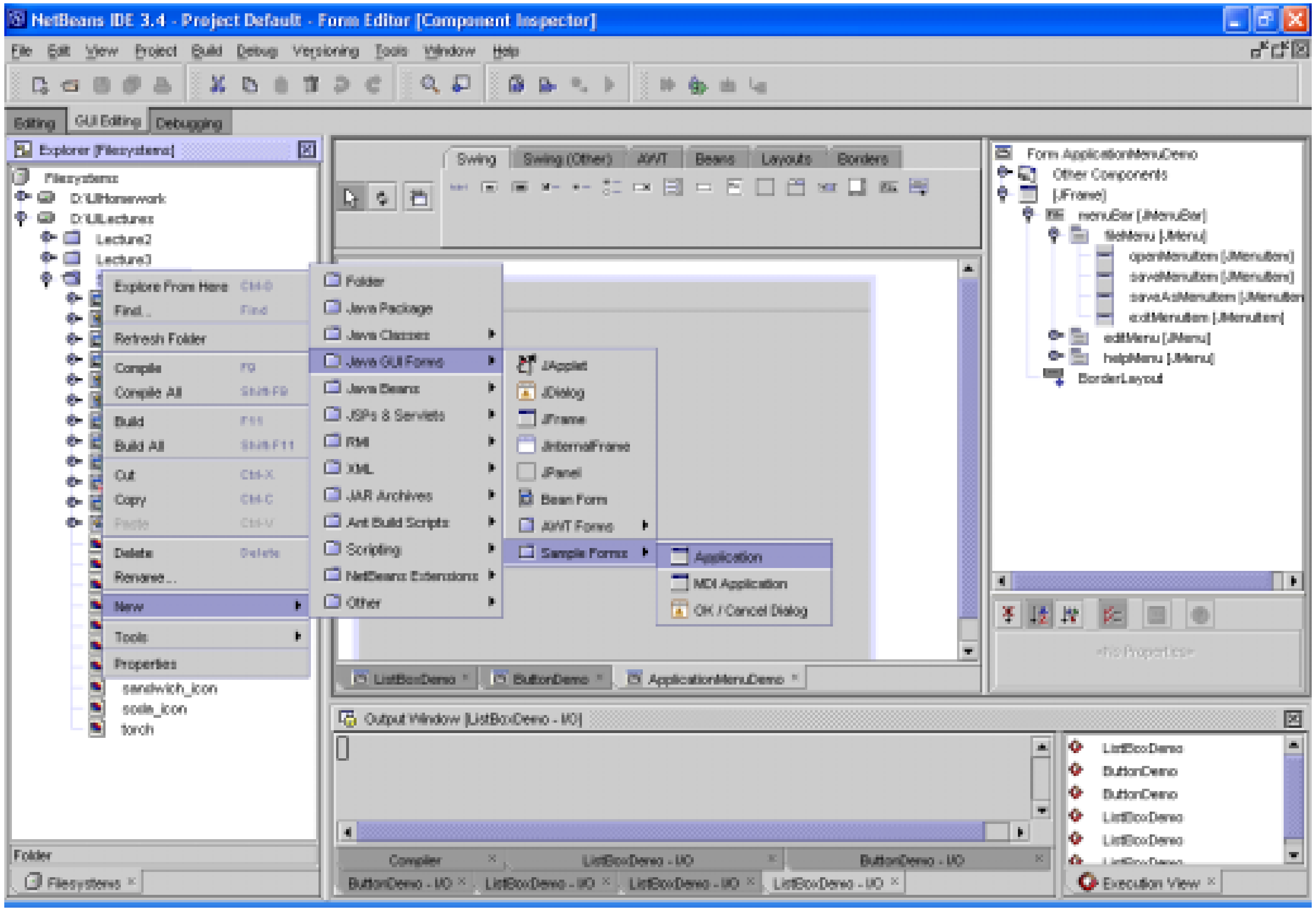
○ Rare
○ Medium Rare
○ Medium
○ Well Done

JComboBox

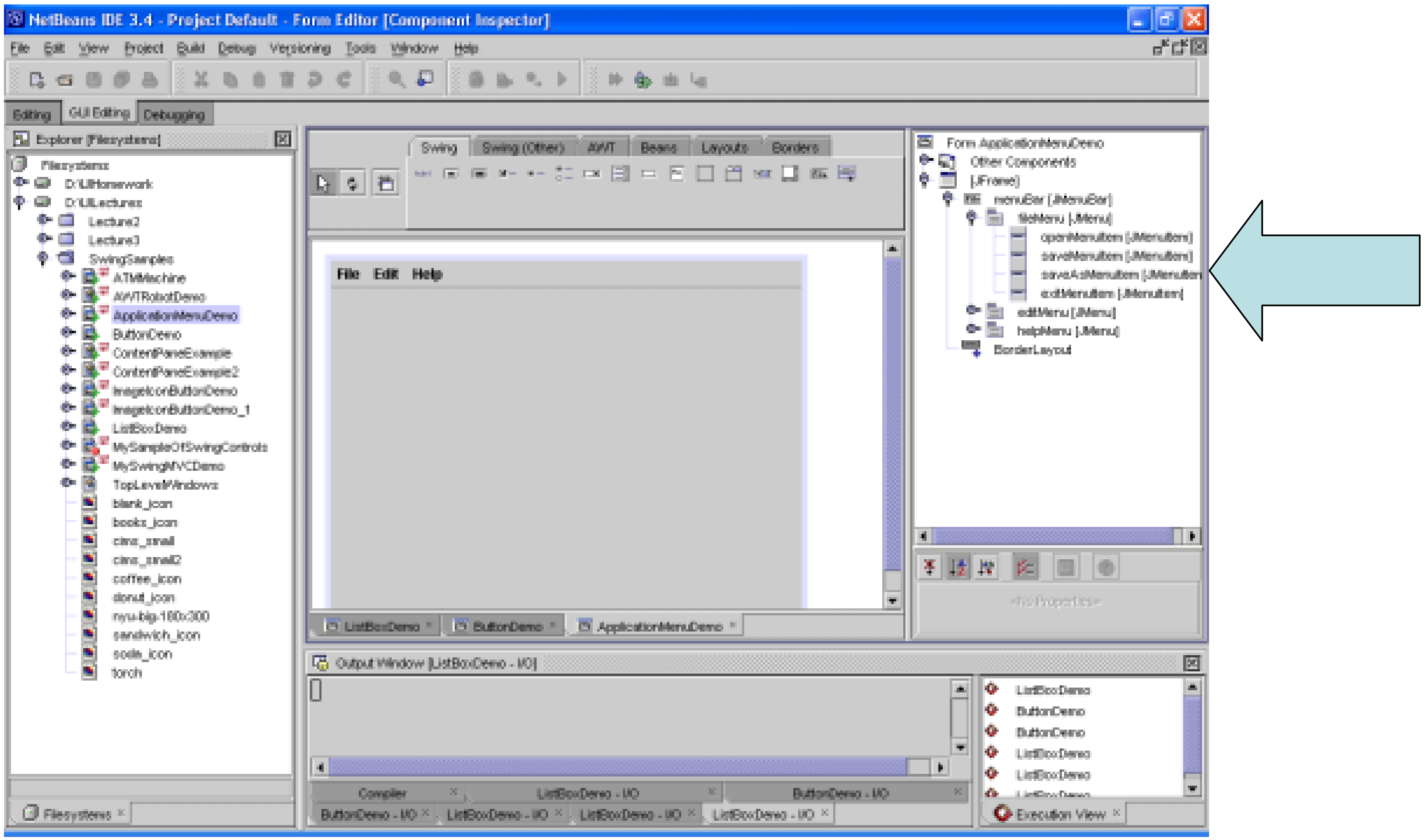
- Two styles in the app – non-editable and editable
- If you use the editable type you should check the input of the user to make sure it is acceptable.
- You can change the style by changing the `editable` property.



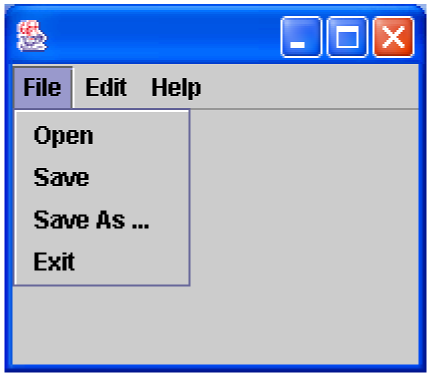
Creating a
JavaGUI -> SampleForms -> Application

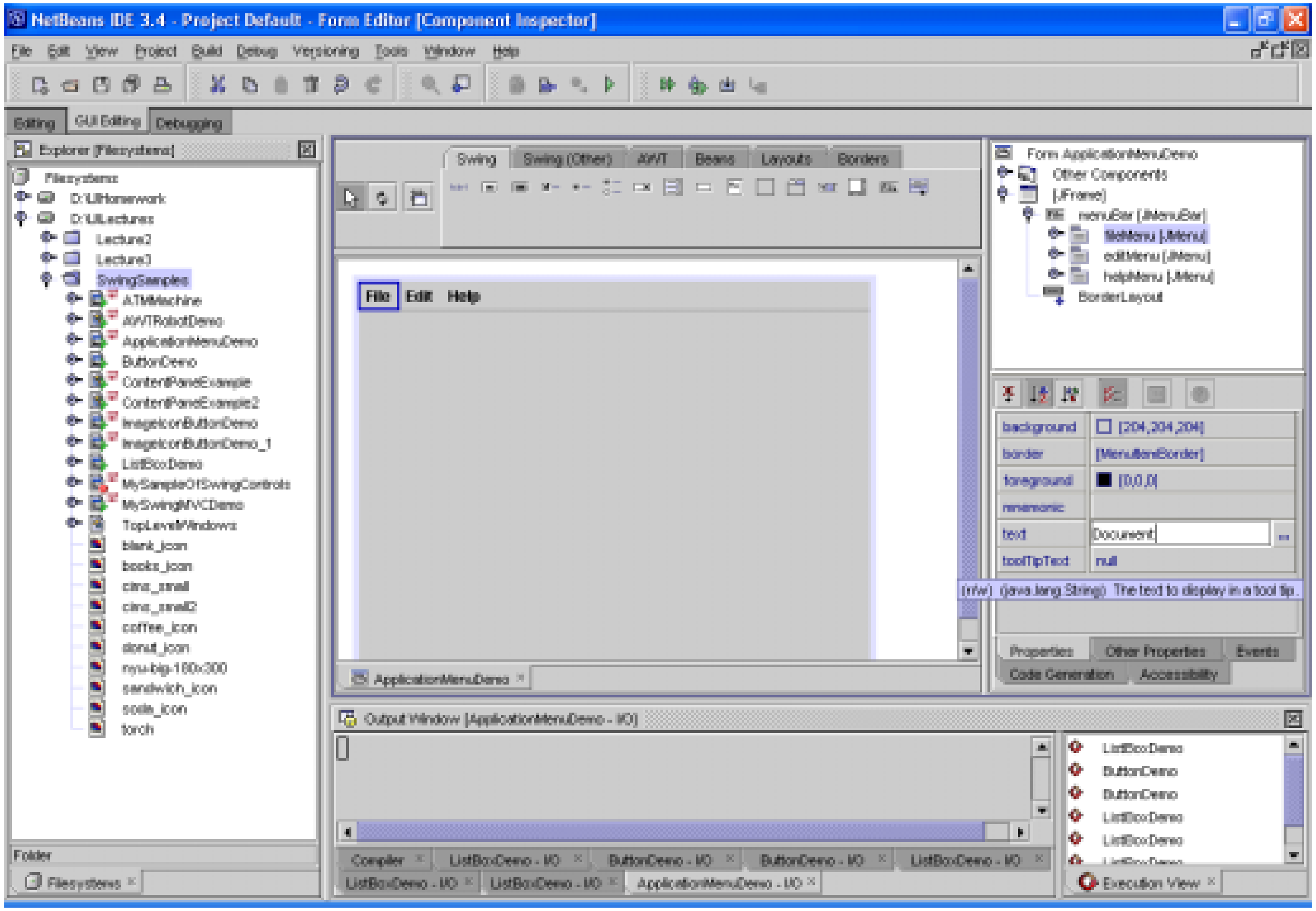


Menu and Menu item hierarchy that defines the menus for the app


NetBeans IDE 3.4 - Project Default - Form Editor [Component Inspector]
File Edit View Project Build Debug Versioning Tools Window Help
Editing GUI Editing Debugging
Explorer [Filesystems]
Swing Swing (Other) AWT Beans Layouts Borders
File Edit Help
Form ApplicationMenuDemo
Other Components
[JFrame]
menuBar [JMenuBar]
fileMenu [JMenu]
openMenuItem [JMenuItem]
saveMenuItem [JMenuItem]
saveAsMenuItem [JMenuItem]
exitMenuItem [JMenuItem]
editMenu [JMenu]
helpMenu [JMenu]
BorderLayout
Output Window [ListBoxDemo - I/O]
Execution View

The default menus

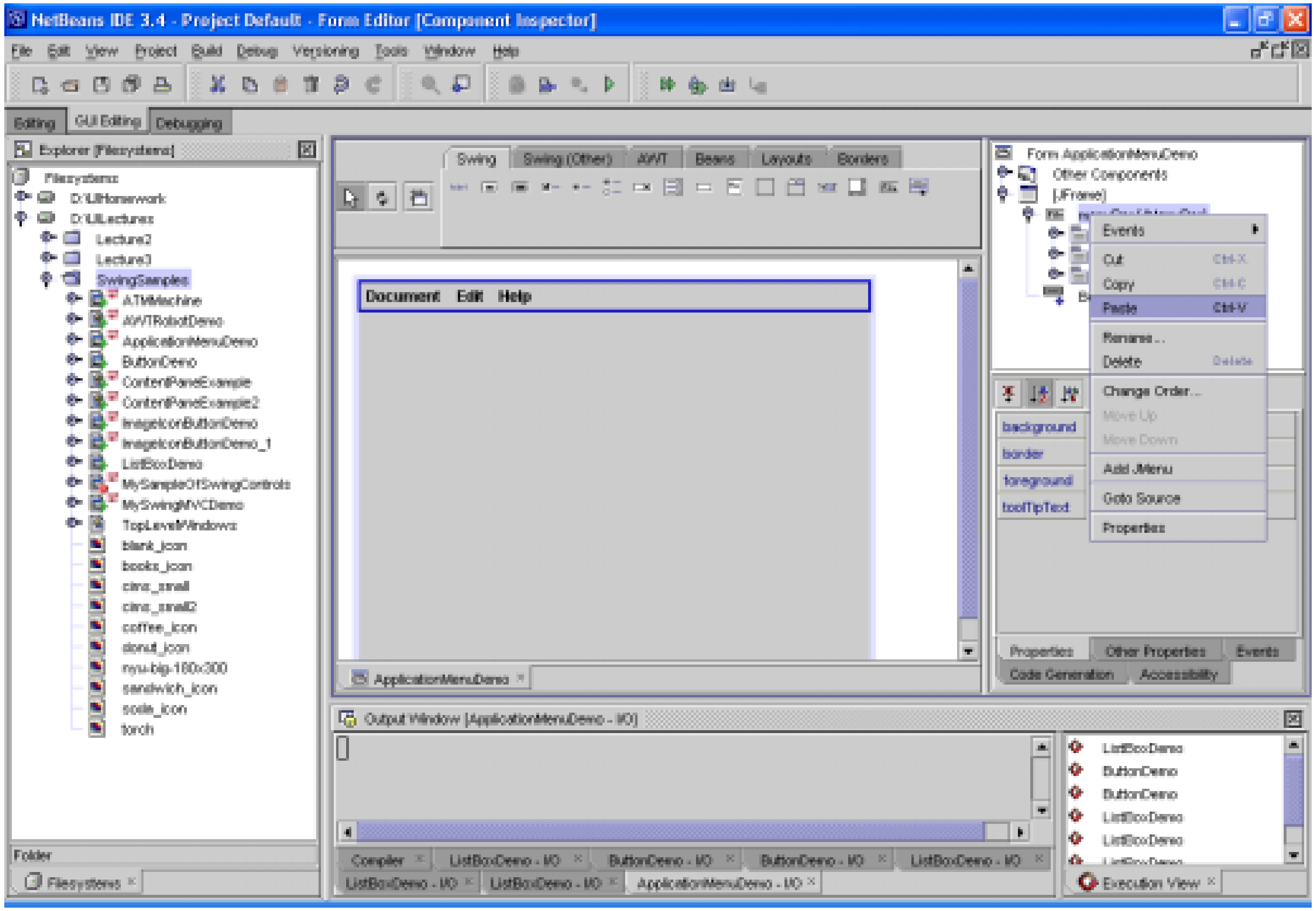




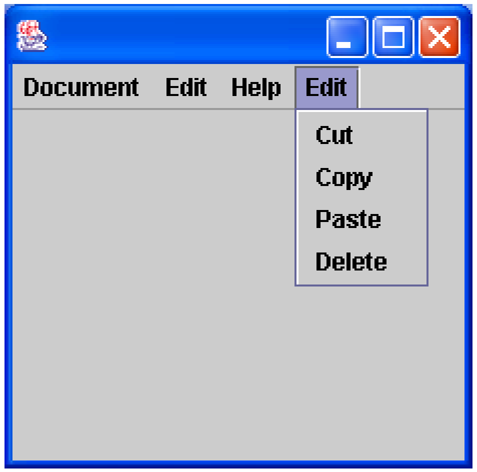
background	[204,204,204]
border	[MenuItemBorder]
foreground	[0,0,0]
mnemonic	
text	Document
toolTipText	null

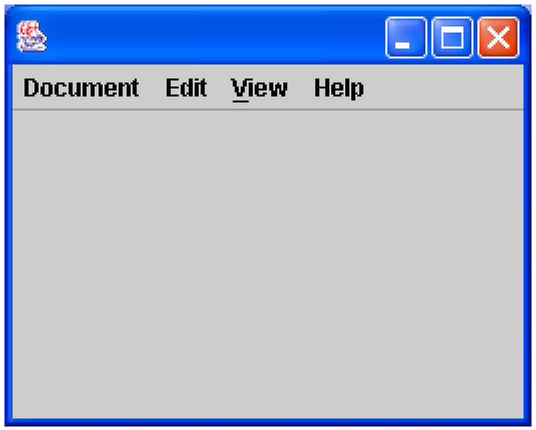
(r/w) (java.lang.String) The text to display in a tool tip.

Copy and paste a menu into the hierarchy



This shows 2 Edit menus


Document
Edit
Help
Edit
Cut
Copy
Paste
Delete


Document
Edit
View
Help