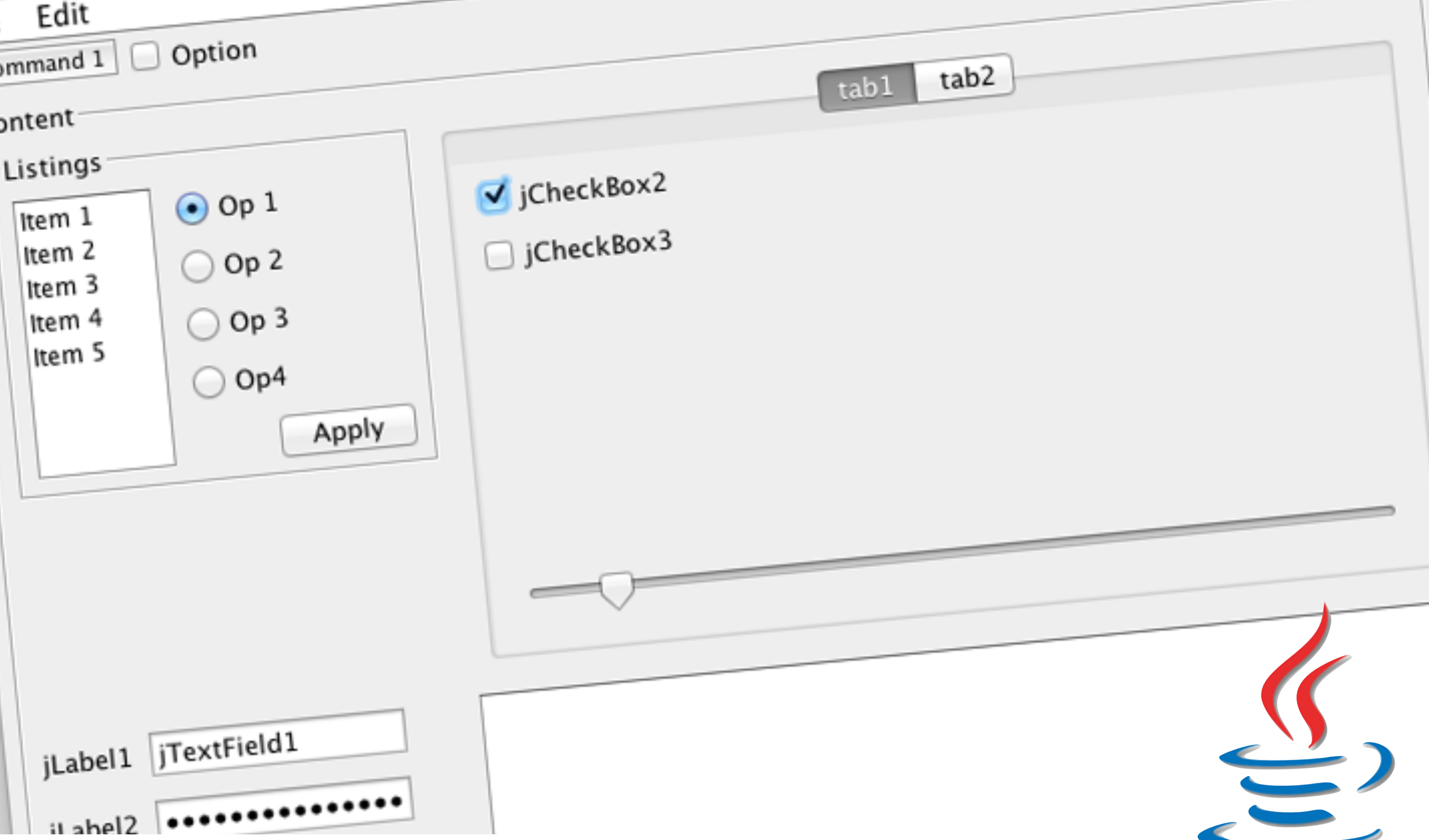


Java swing

Rui Couto
rui.couto@di.uminho.pt

Outline

- Whats Java Swing
- Setup
- Building GUIs
- The MVC pattern
- Example

What is Java swing

A brief introduction

What is Java swing

- A Java library for designing user interfaces
  - The official GUI Toolkit
- Lightweight (platform independent)
- Several kind of widgets
  - Simpler: buttons, checkboxes, etc.
  - Complex: lists, tables, trees, etc.

Setup

How to start

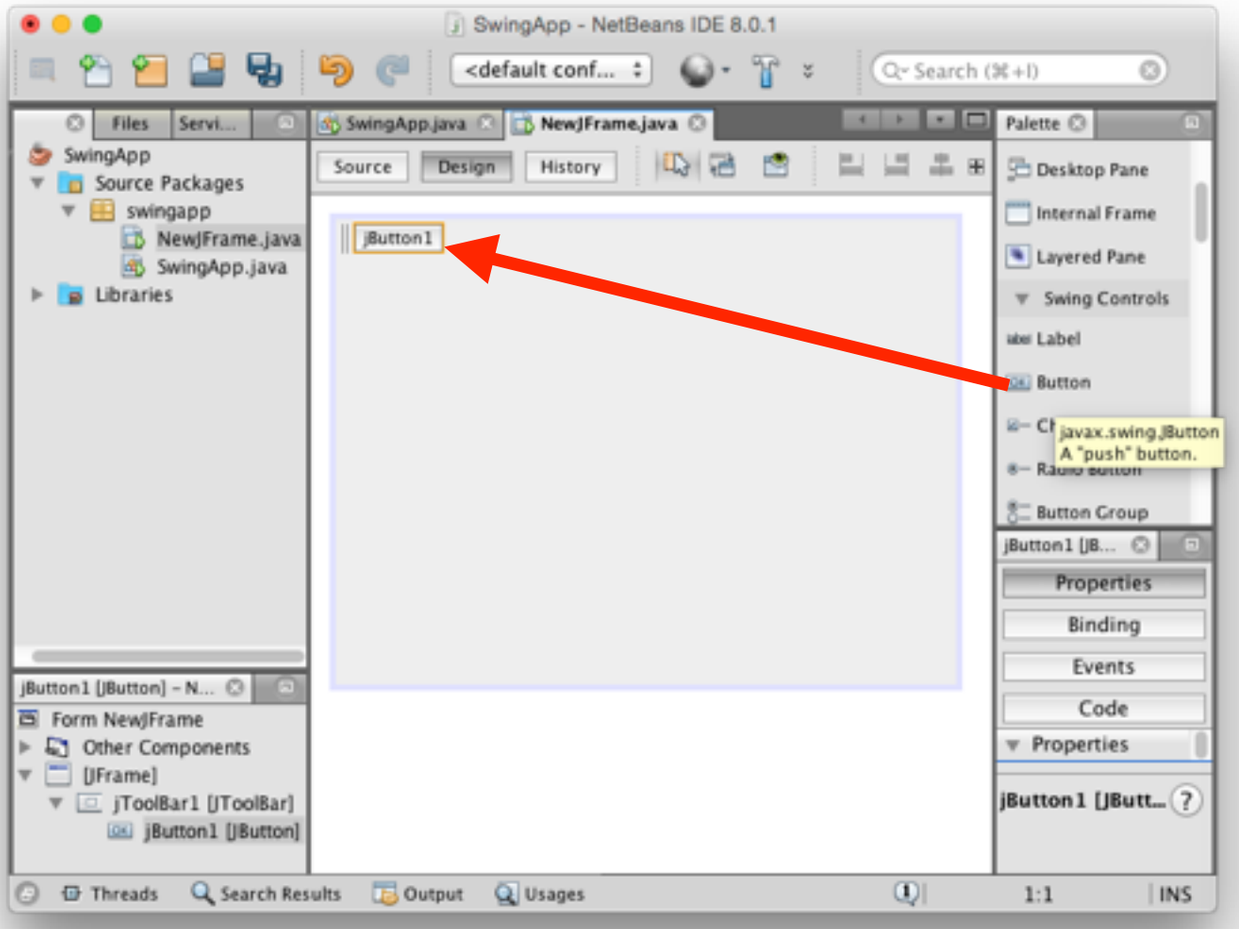
Setup

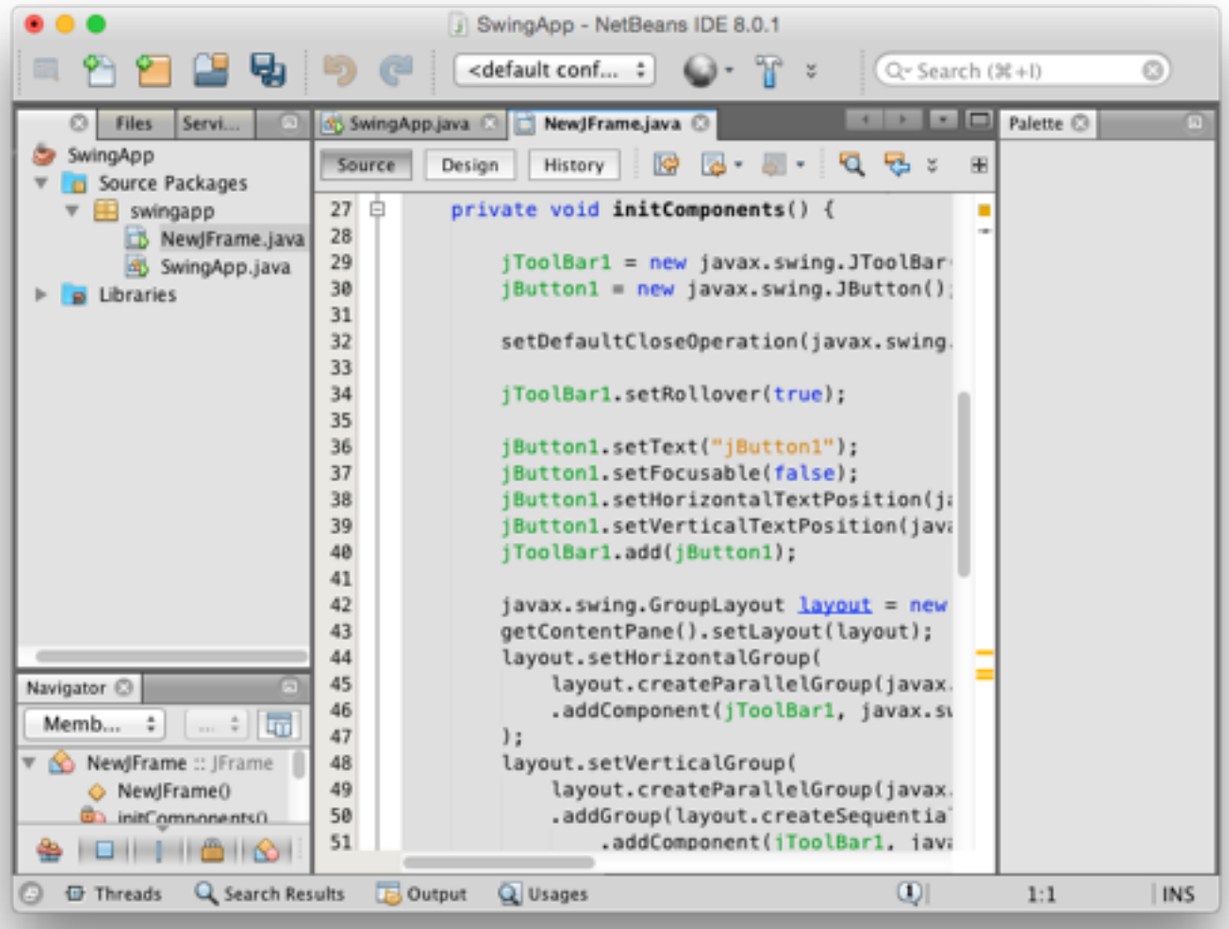
- Two alternatives (at least):
  - Manual coding:

```
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
jButton1 = new javax.swing.JButton();
jButton2 = new javax.swing.JButton();
jButton1.setText("Command 1");
jButton1.setFocusable(false);
jButton1.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
jButton1.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});
jToolBar1.add(jButton1);
```

Setup

- Using an IDE:





Setup

- The hand made code is more personalized (i.e. custom content)
- The IDE approach is easier
- The best approach depends on the application context
- Combining both approaches might be a good tradeoff
- NetBeans includes a form editor
- In eclipse it is possible to install 3rd party plugins

Building GUIs

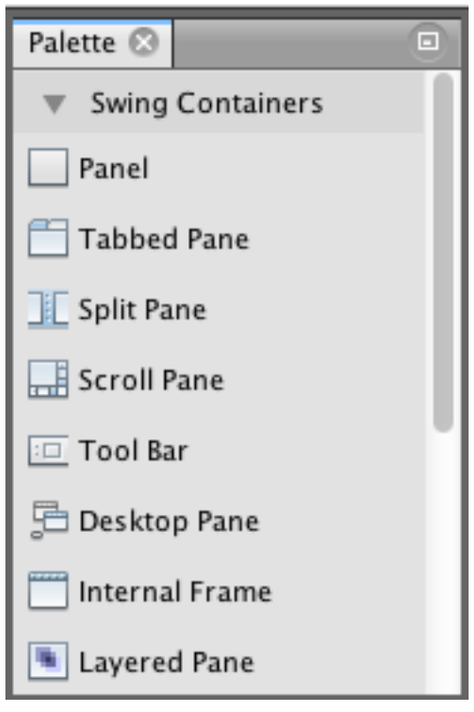
Different components

Building GUIs

- Using NetBeans
- Main component is a JFrame
- JFrame can have other components:
  - (Other) Containers
  - Controls
  - Menus
  - Windows
  - Fillers

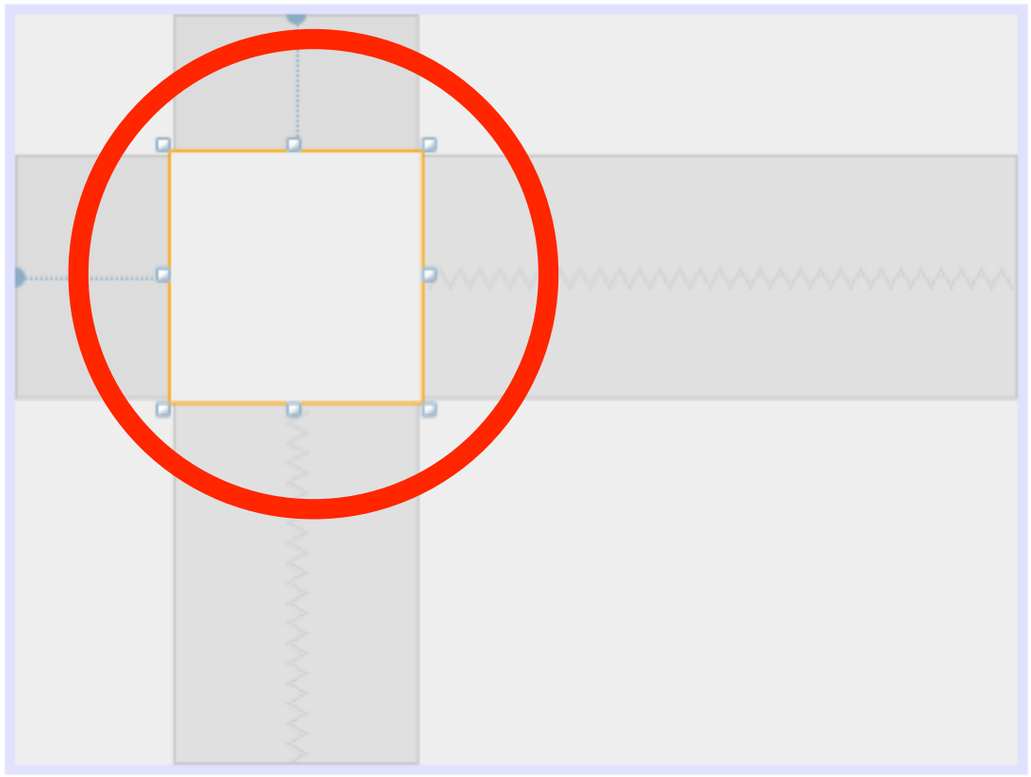
Building GUIs - Containers

- Composite elements to contain other elements
- Are the base of the interfaces
- Provide easier arrangement of the GUI elements
- Have several kind of elements



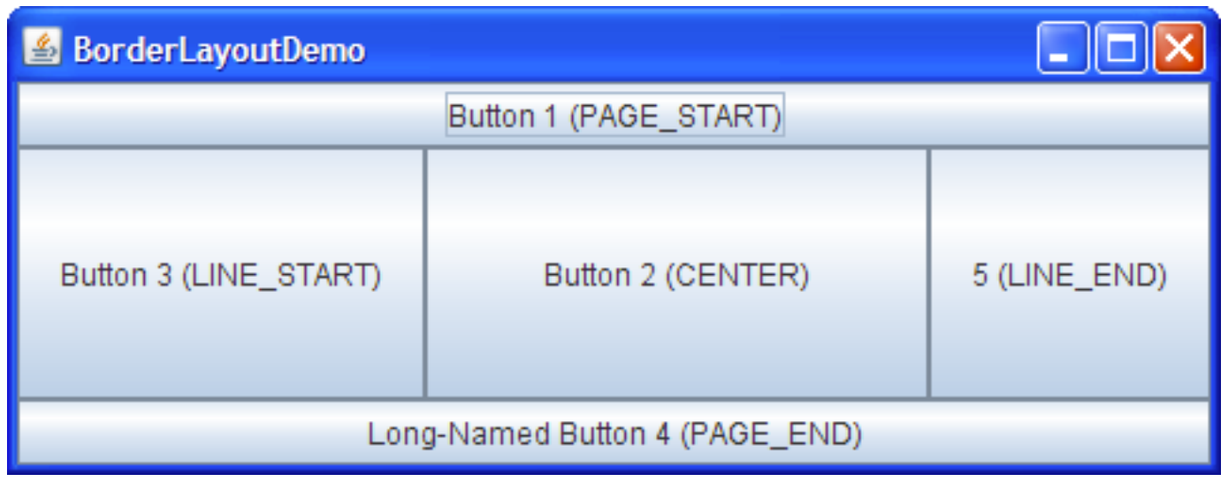
Building GUIs - Containers - Panel

- Primarily container
- Have a *layout*
- Typically do not have listeners (but might have!)
- Example:

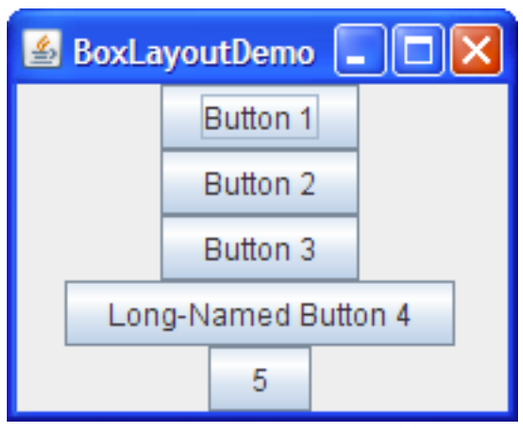


Building GUIs - Containers - Panel

- Layouts are content managers
- Border layout:



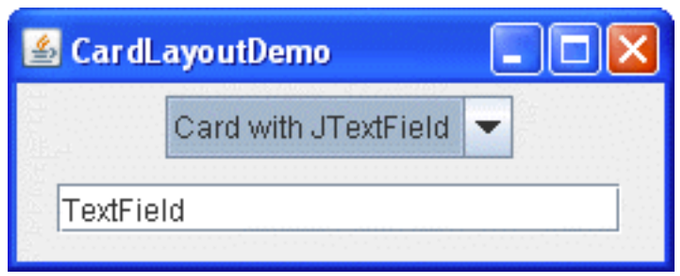
- Box layout:



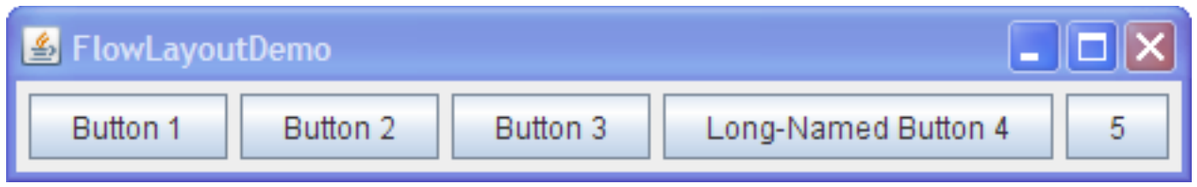
https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html

Building GUIs - Containers - Panel

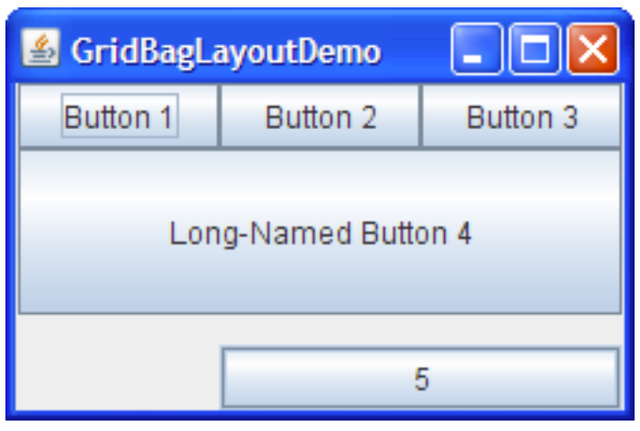
- Card layout:



- Flow layout:



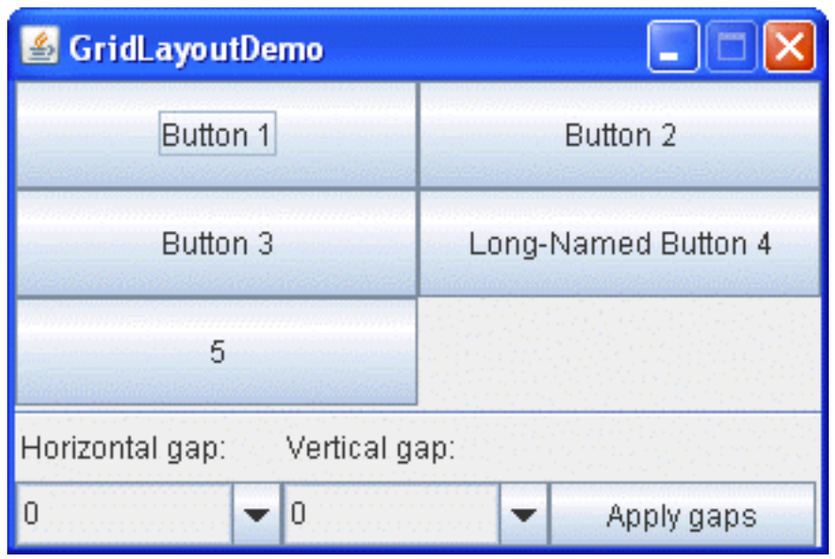
- GridBag layout:



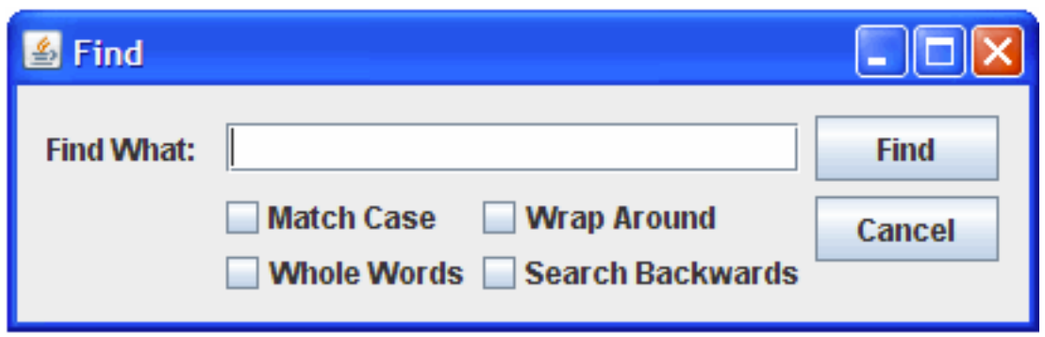
https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html

Building GUIs - Containers - Panel

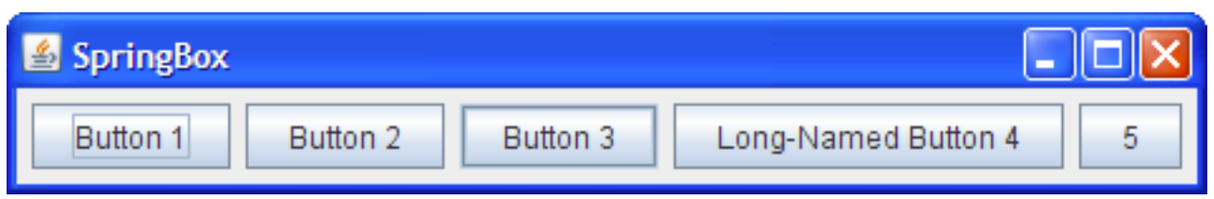
- Grid layout:



- Group layout:



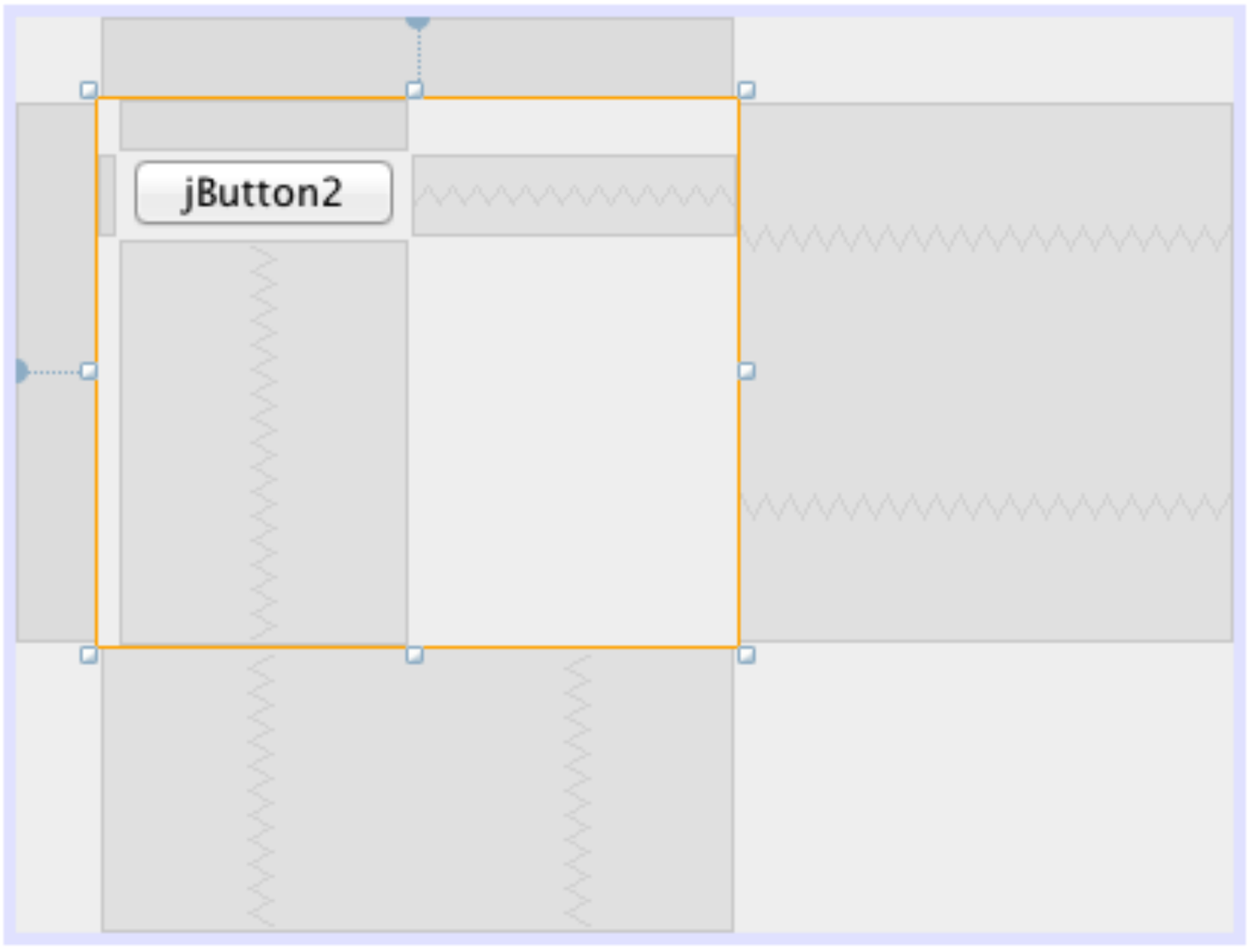
- Spring layout:



https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html

Building GUIs - Containers - Panel

- Free design (NetBeans):

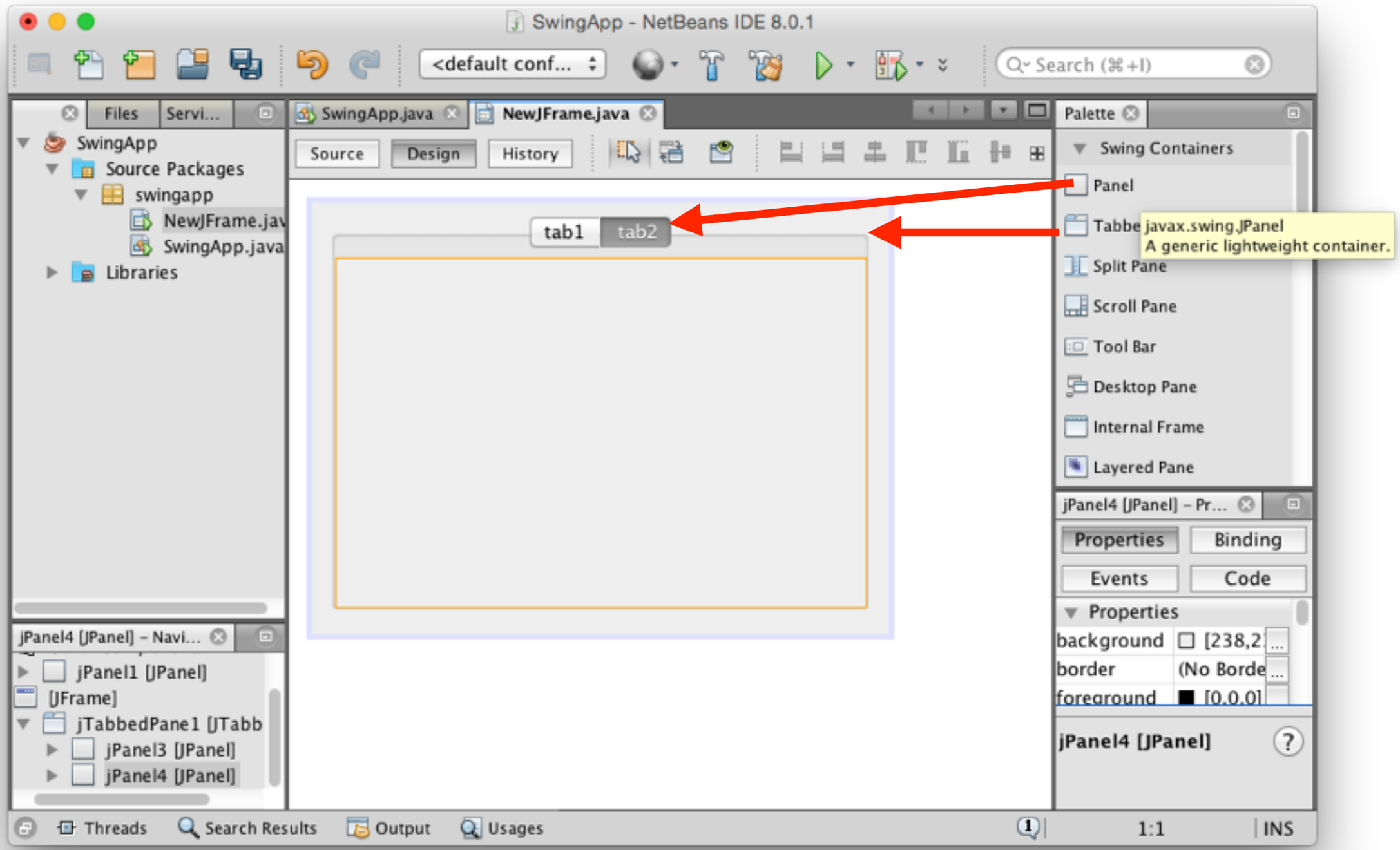


Building GUIs - Containers - TabbedPane

- Tab host for Panels
- Easy way to have several pages
- It is possible to change, switch, add (, etc) tabs by code
- A single TabbedPane holds several panels

Building GUIs - Containers - TabbedPane

- Example:

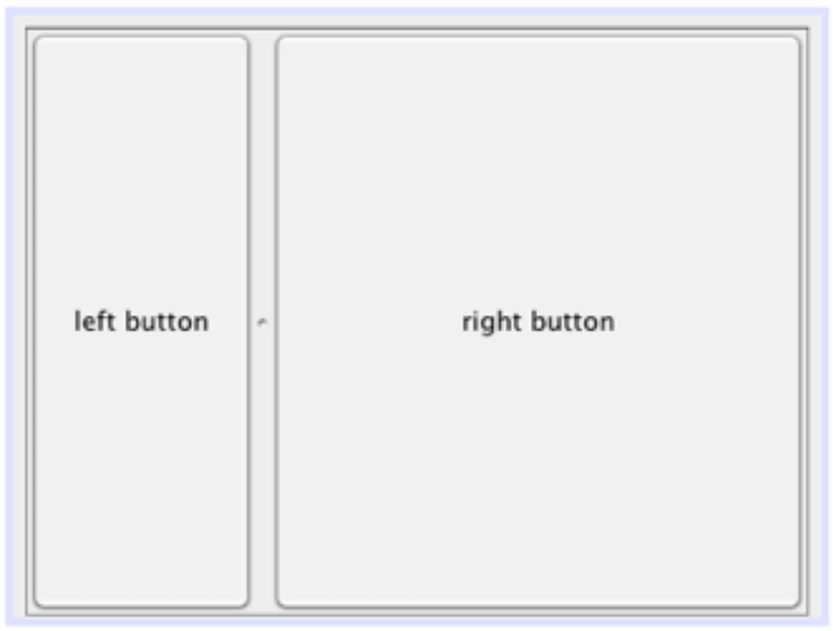


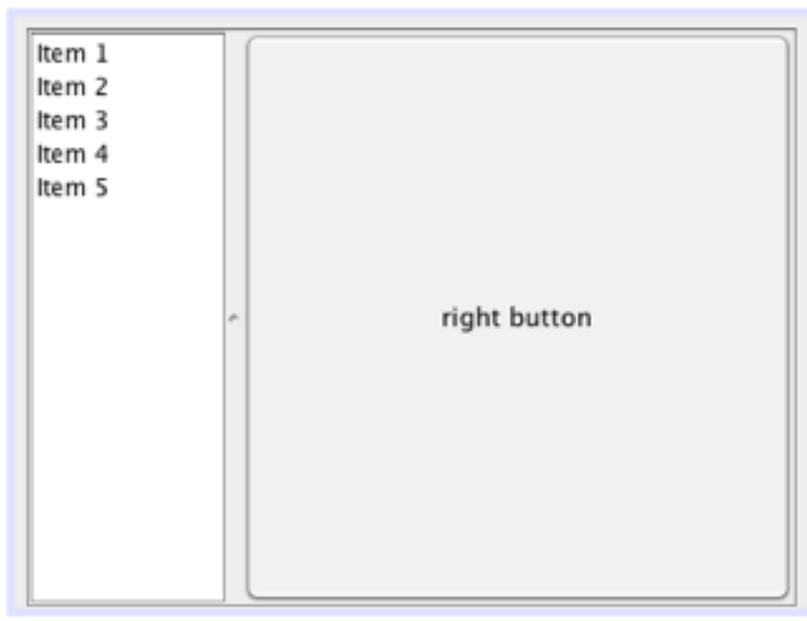
Building GUIs - Containers - SplitPane

- Element to split horizontally or vertically
- Default elements are buttons, but typically a panel should be placed instead
- The divider location can be changed at runtime

Building GUIs - Containers - SplitPane

- Example:



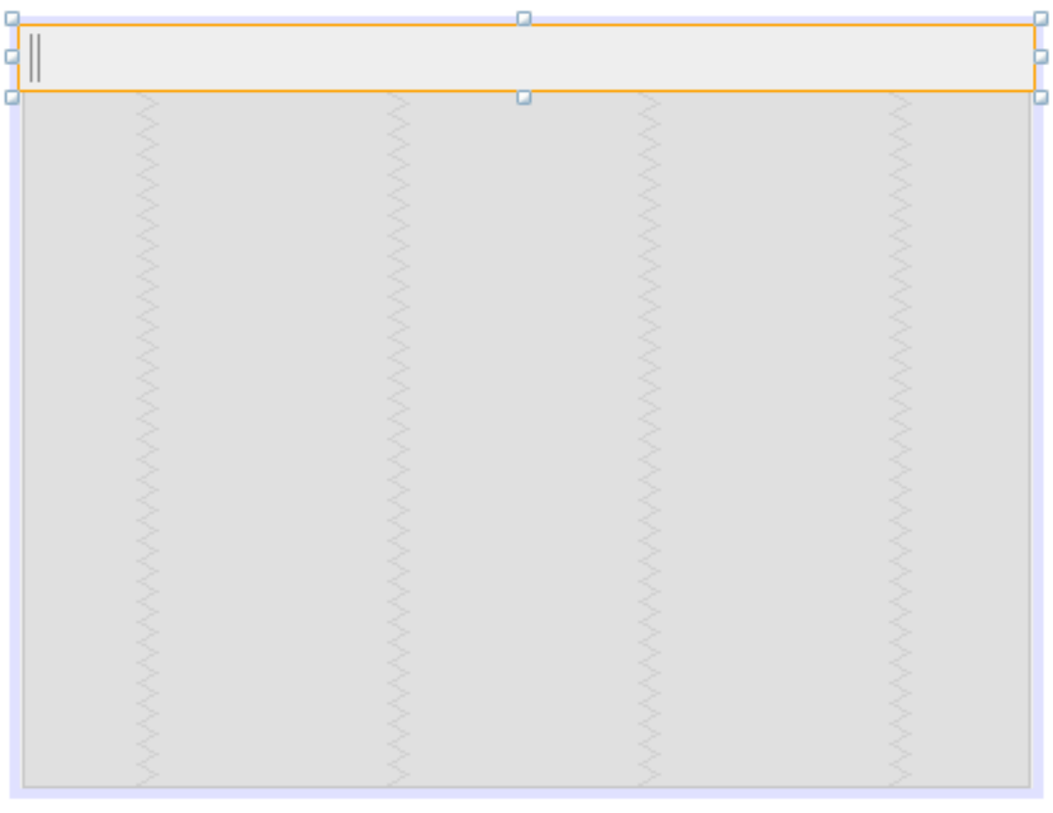


Building GUIs - Containers - ScrollPane & LayeredPane

- Behaves like a regular panel
- ScrollPane creates automatically scrollbars to handle large content
- LayeredPane allows to sort the contained elements in layers

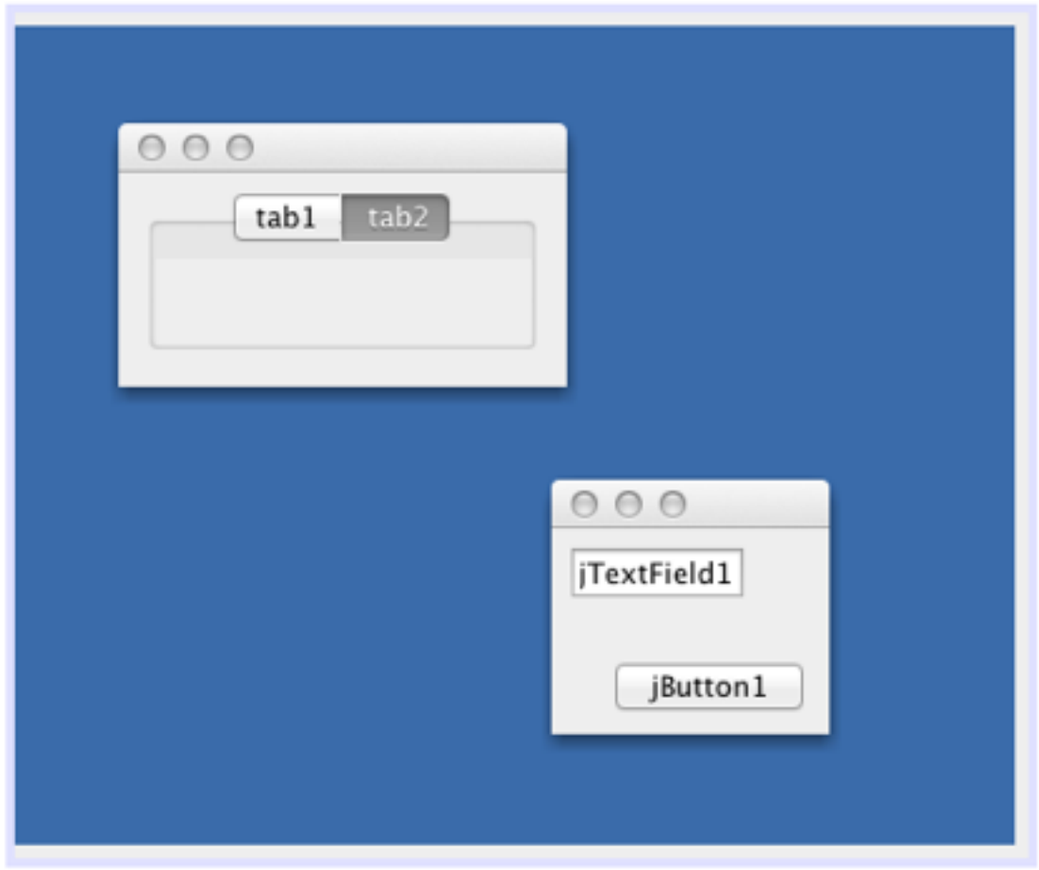
Building GUIs - Containers - ToolBar

- A toolbar to hold common components (e.g. buttons)
- Can be either fix or floatable



Building GUIs - Containers - DesktopPane & Internal frame

- Used to create a virtual desktop (with several windows).
- Allows better results when the windows are created by code
- Example:

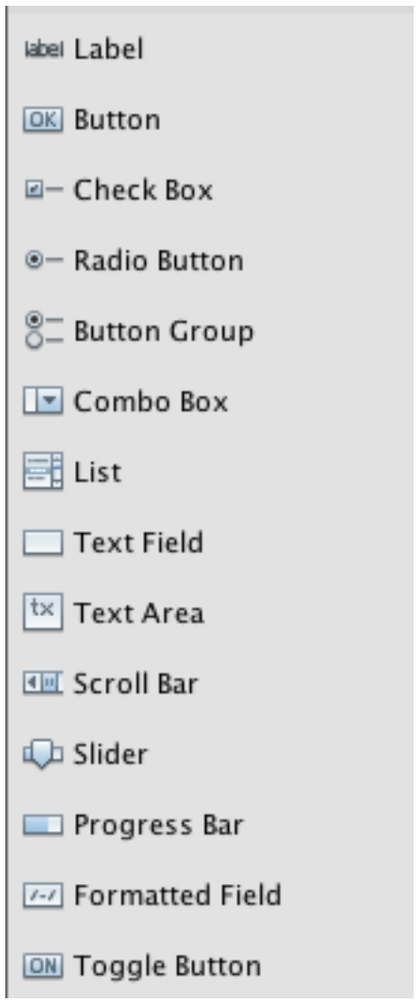


Building GUIs - Controls

- Are elements for input/output.
- Typically have associated actions.
  - Label - Element to show text
  - Button - Element to respond to actions
  - Checkbox - Boolean input element
  - RadioButton - Exclusive selection element
  - ComboBox - Pop out select element
  - List - (Multi) selection list

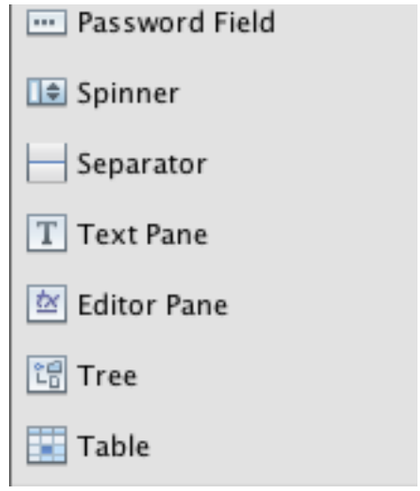
Building GUIs - Controls

- TextField - Simple text input element
- TextArea - Complex text input element
- Scrollbar - Scrollable bar
- Slider - Fixed value select element
- Progress bar - Progress indicator
- Formatted Field - Pattern input element
- Toggle button - On/off button



Building GUIs - Controls

- Password field - Masked input text field
- Spinner - Numeric input field
- Separator - Separator line
- TextPane - Formatted text input
- Editor Pane - Complex editing operations
- Tree - Hierarchical component display
- Table - Table container

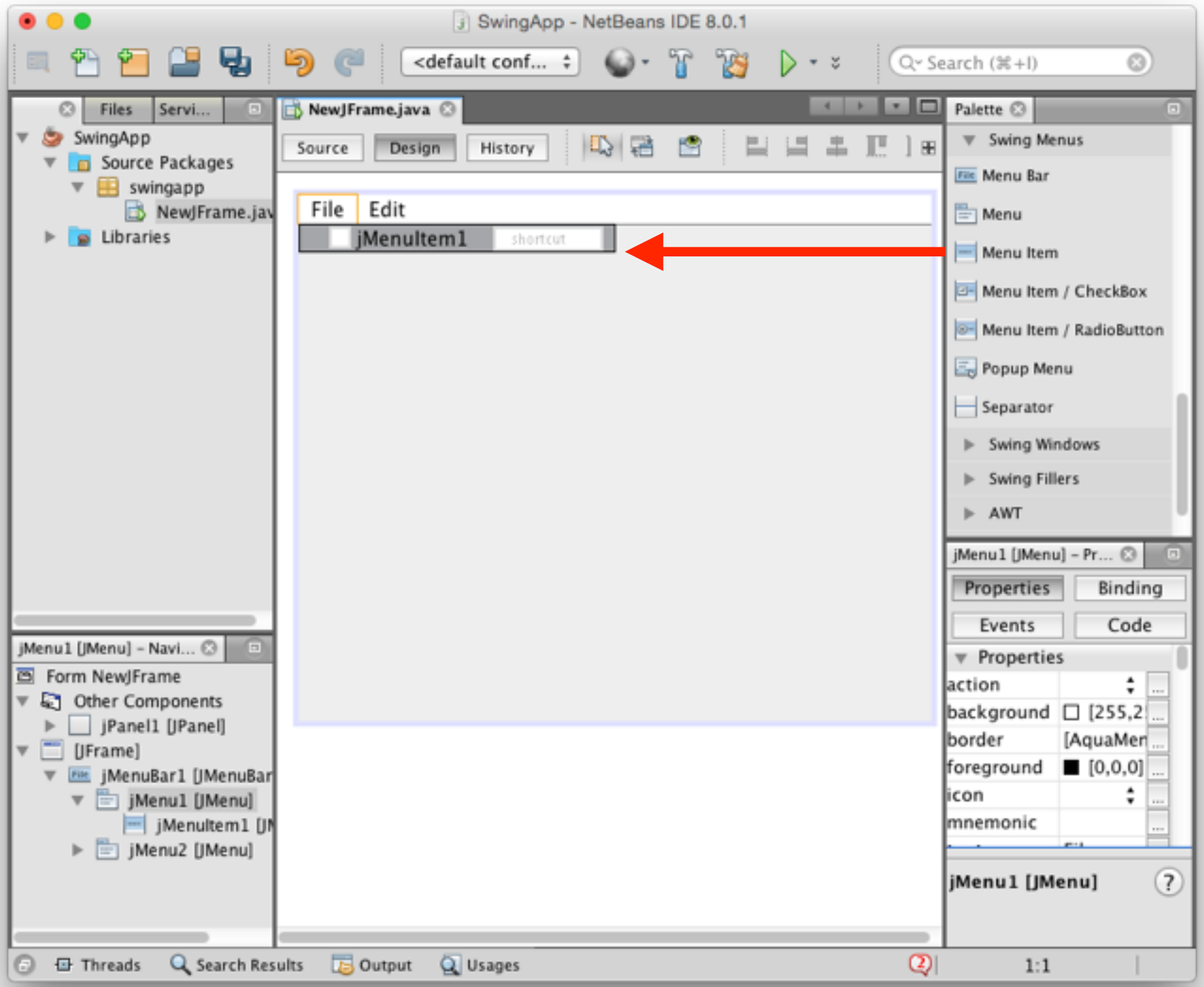


Building GUIs - Menus

- Used to create menu bars
- Have a fixed position (not floatable)
    - Menu bar - The menu bar container
    - Menu - A new entry
    - Menu Item (+ checkbox/radio) - A new item inside a menu
    - Popup - Window that pops up to show choices
    - Separator - Separator line

Building GUIs - Menus

- Example:



Building GUIs - Swing Windows

- Meant to be created by code
- Provide UI facilities
- Example for color chooser:

```
JColorChooser cs = new JColorChooser();
JColorChooser.createDialog(this, "Pick a color", true, cs ,
    new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            //code to hanlde ok
          }
    }, new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            //code to handle cancel
        }
    }).setVisible(true);
```

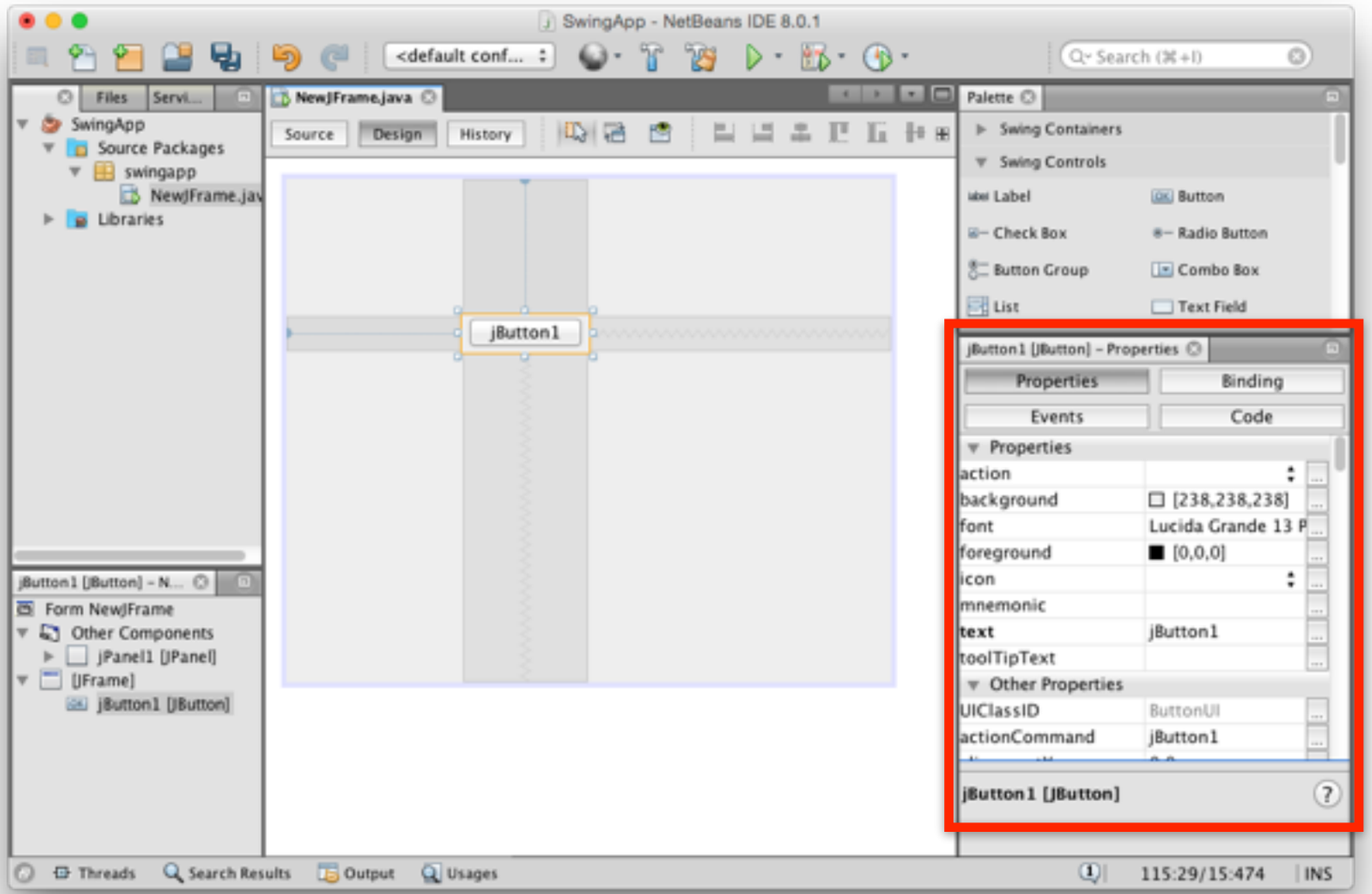
Building GUIs

- Swing fillers - space organization components
- AWT - Deprecated window building components
- Beans & persistence - Advanced GUI elements

Building GUIs - Customization

- Elements have several properties which can be customized

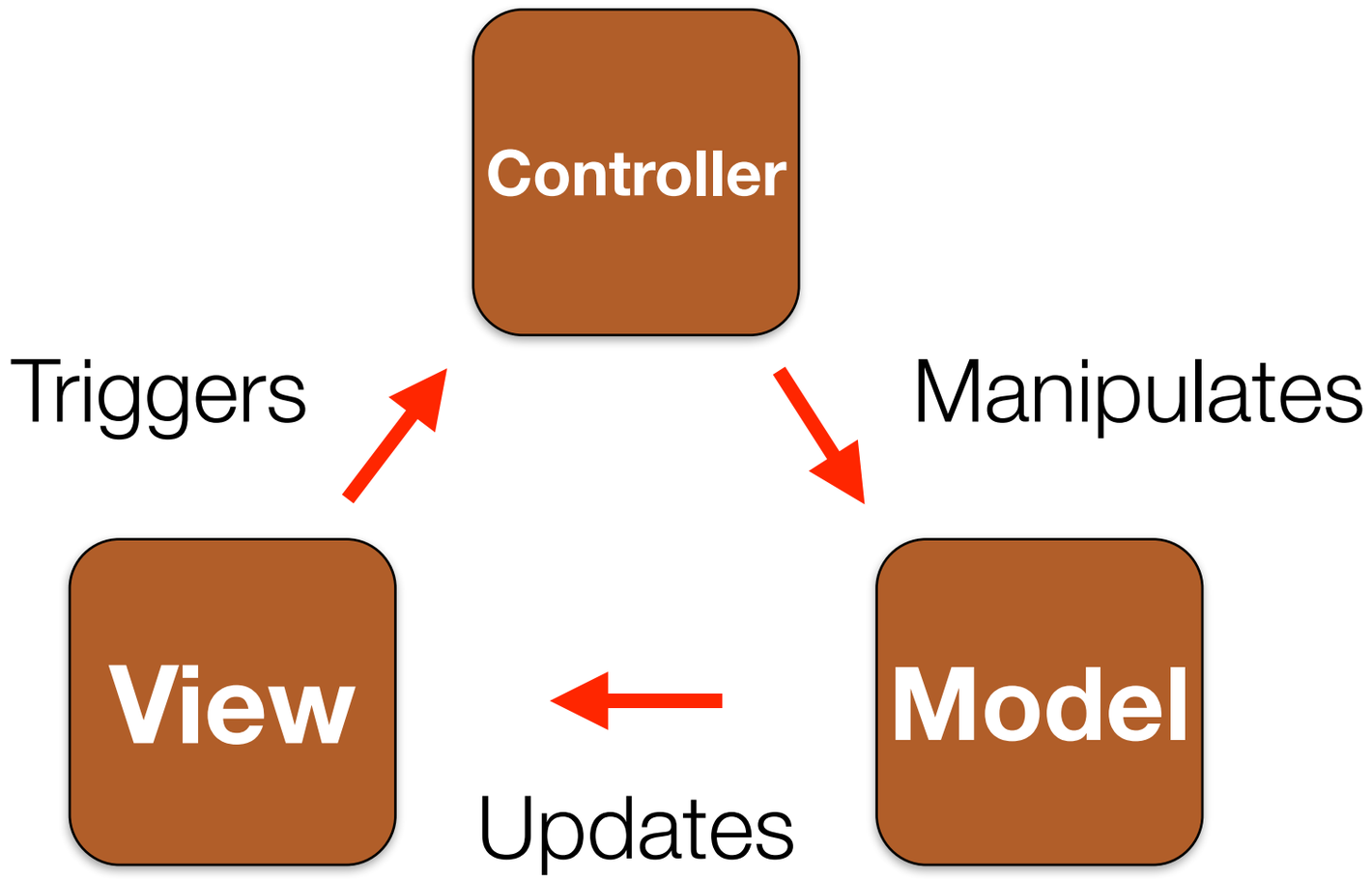
- Properties tab shows available properties:



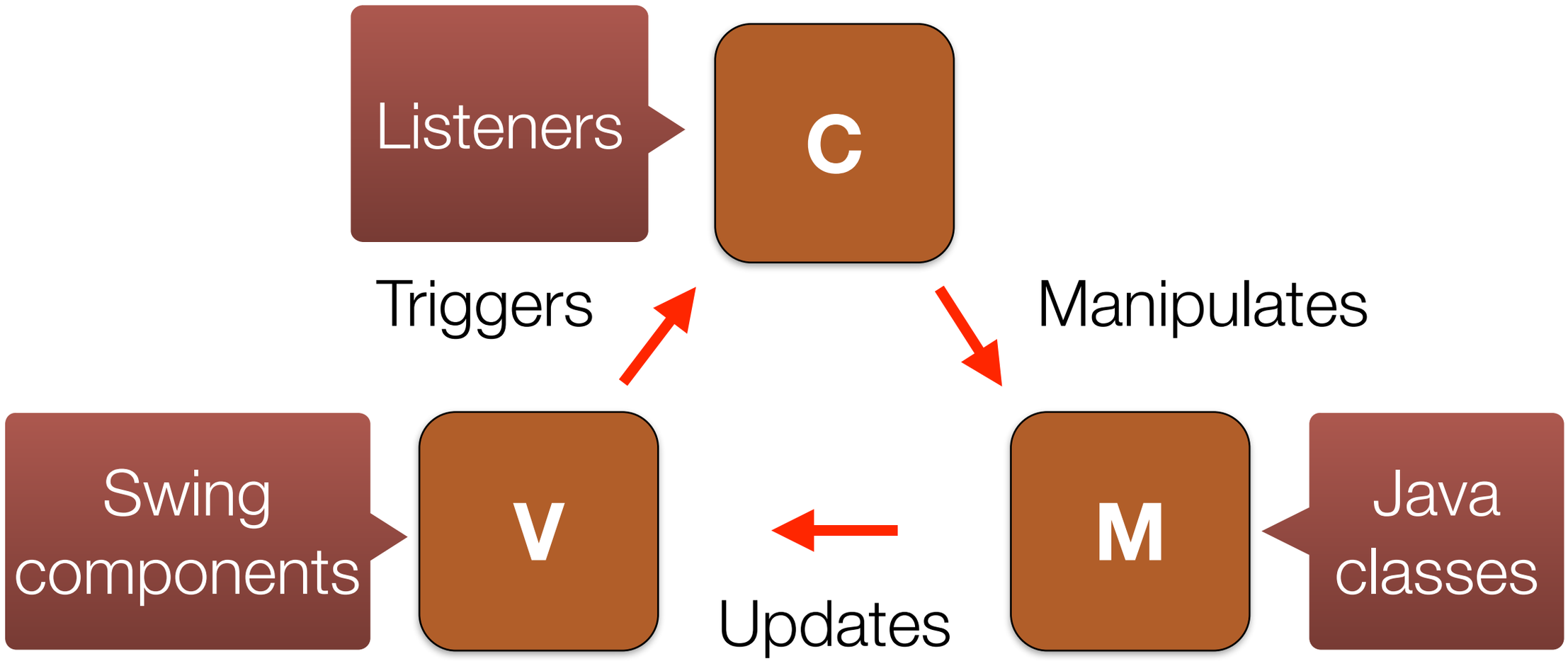
The MVC Pattern

Displaying and handling events

The MVC Pattern



The MVC Pattern

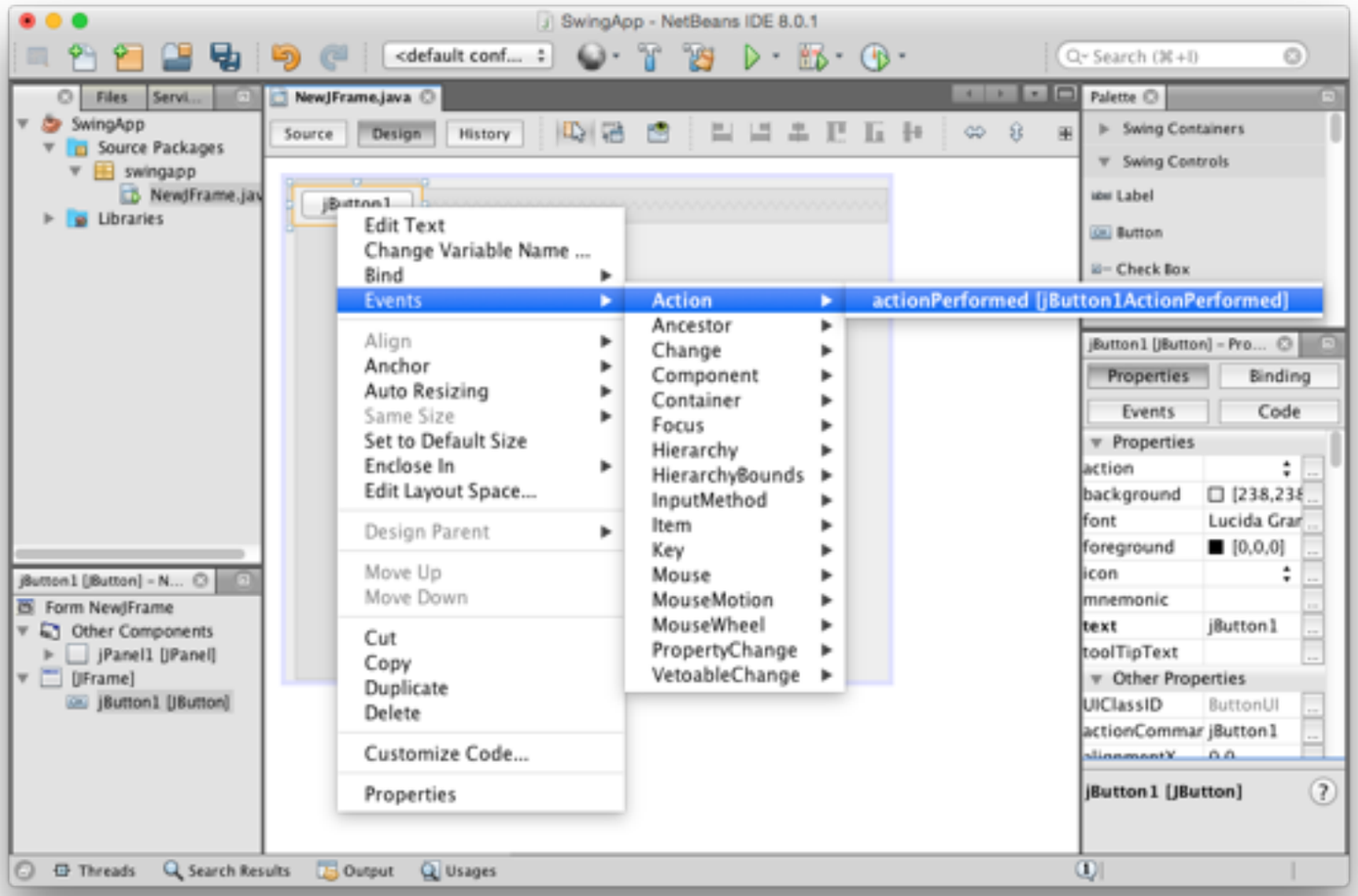


The MVC Pattern

- Model:
  - Are the business classes (e.g. User, Account, Car, etc.)
- View:
  - Are the swing components (e.g. Buttons, lists, etc.)
- Controller
  - Are the handlers to the operations

The MVC Pattern

- To add an action to a component:
  - Double click (for instance in a button)
  - Right click > Events > Select one



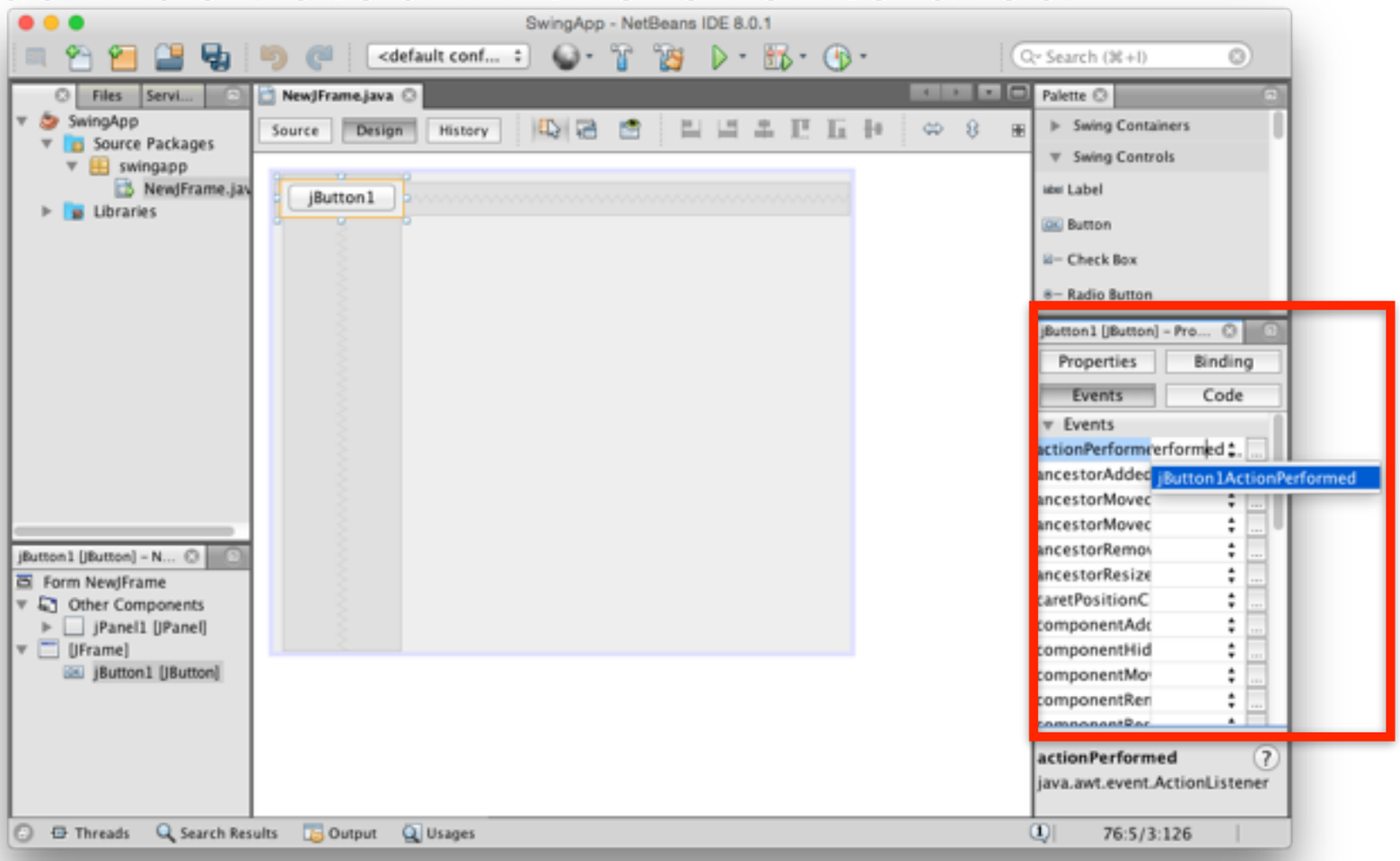
The MVC Pattern

- The code is automatically generated:

```
//...
jButton1 = new javax.swing.JButton();

jButton1.setText("jButton1");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});
//...

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    //write your code here
}
```

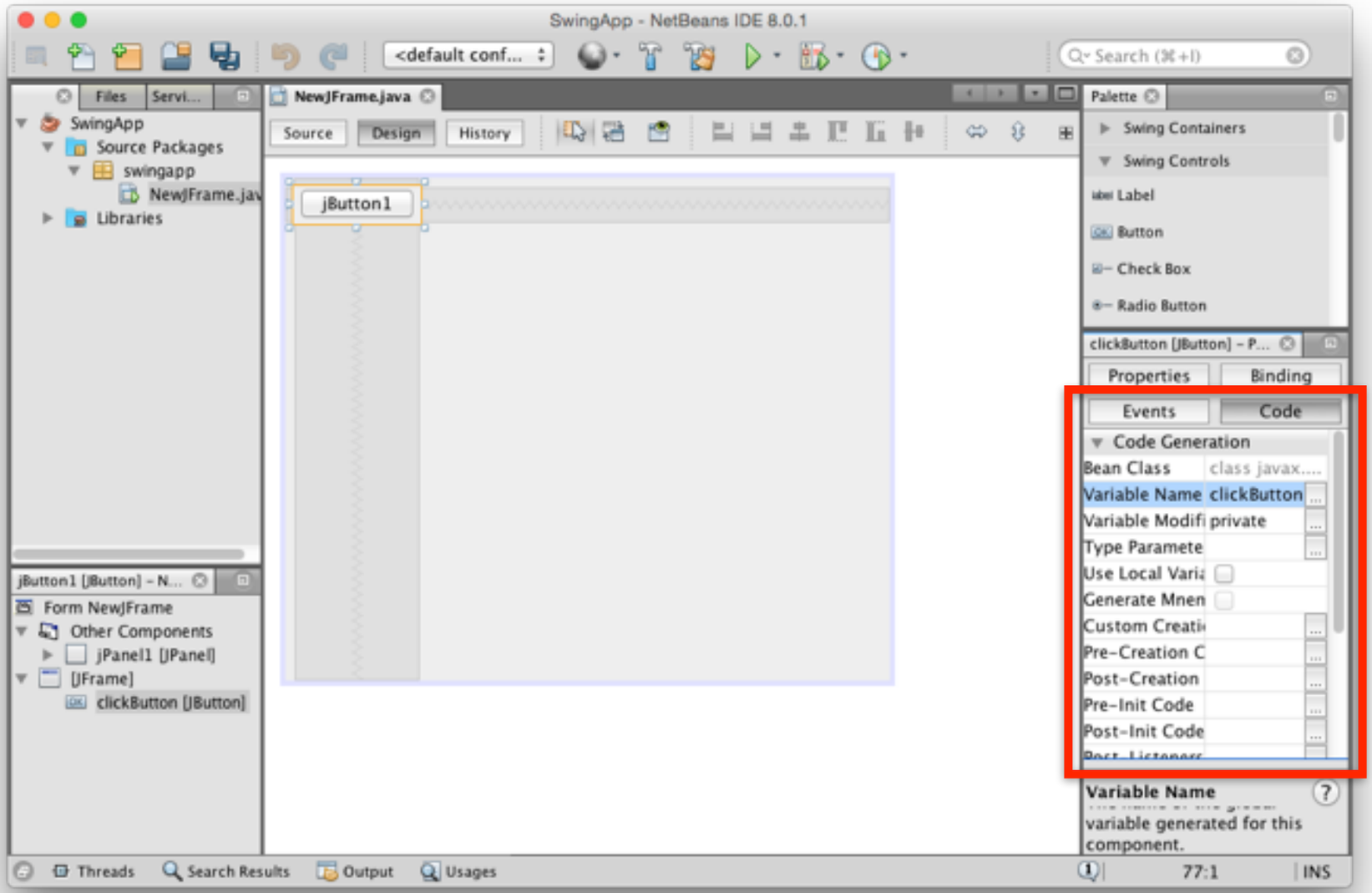
The MVC Pattern

- An element can have several actions
  - Just repeat the process
- Actions can be listed in the events area



The MVC Pattern

- The elements are to be referred from the code
- Set the variable name in the Code area



The MVC Pattern

- Elements are referred as any other instance inside the code

```
//...
clickButton.setText("Click here");
//...
```

Example

How to create an interface

Example

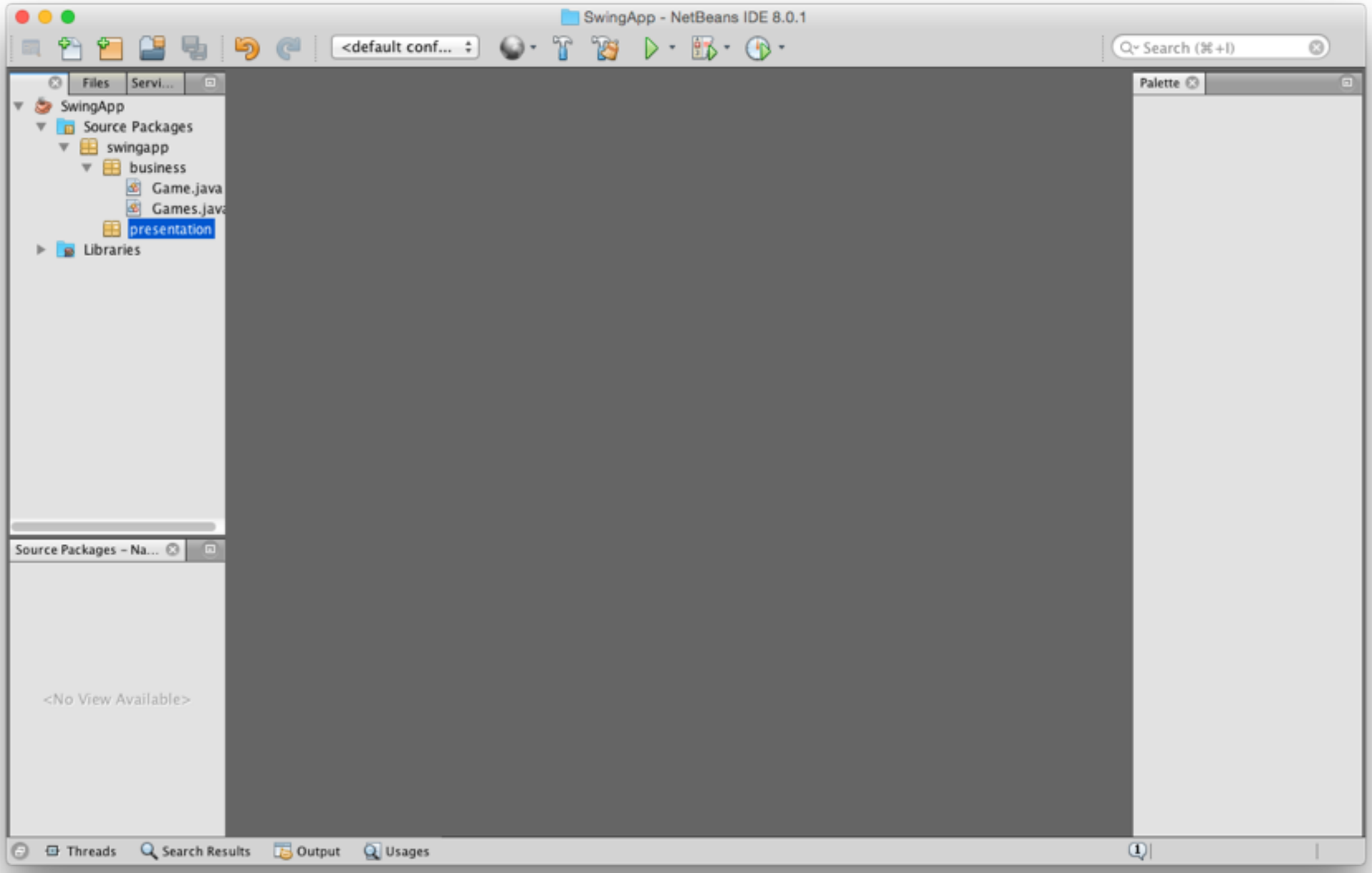
- How to create a window for input/output.
- Example of an interface to manage a videogame database
- Business layer contains a **Game** class, plus a container class, **Games**.

Example

```java
public class Game {
    private String platform;
    private String name;
    private int rating;
    private boolean finished;
    private boolean have;
    private boolean want;

    public Game(String platform, String name, int rating, boolean finished, boolean
        this.platform = platform;
        this.name = name;
        this.rating = rating;
        this.finished = finished;
        this.have = have;
        this.want = want;
    }
```

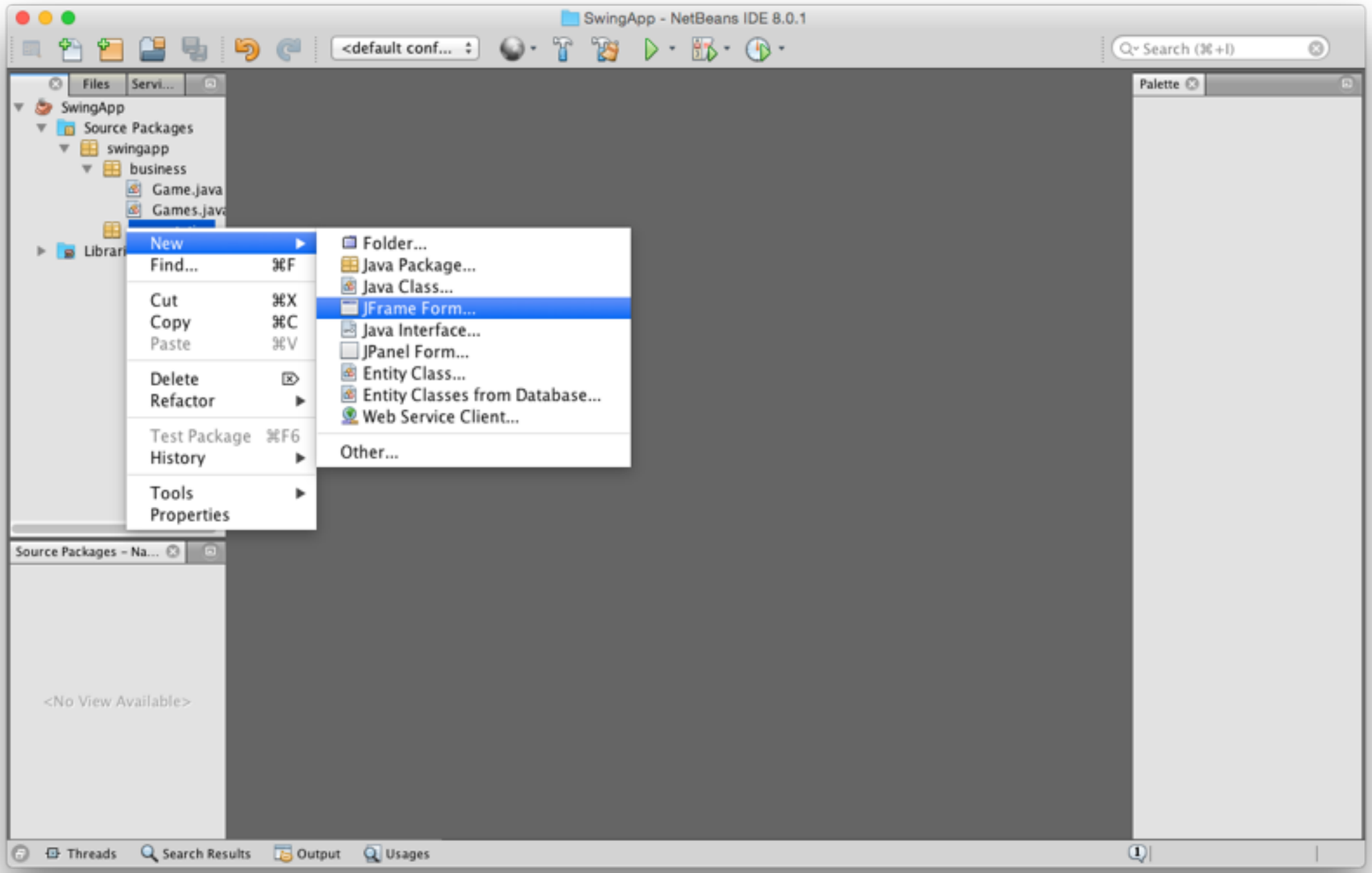
```java
public class Games {
    private ArrayList<Game> games;

    public Games() {
        games = new ArrayList<Game>();
    }

    public ArrayList<Game> getGames() {
        return games;
    }
```

Example

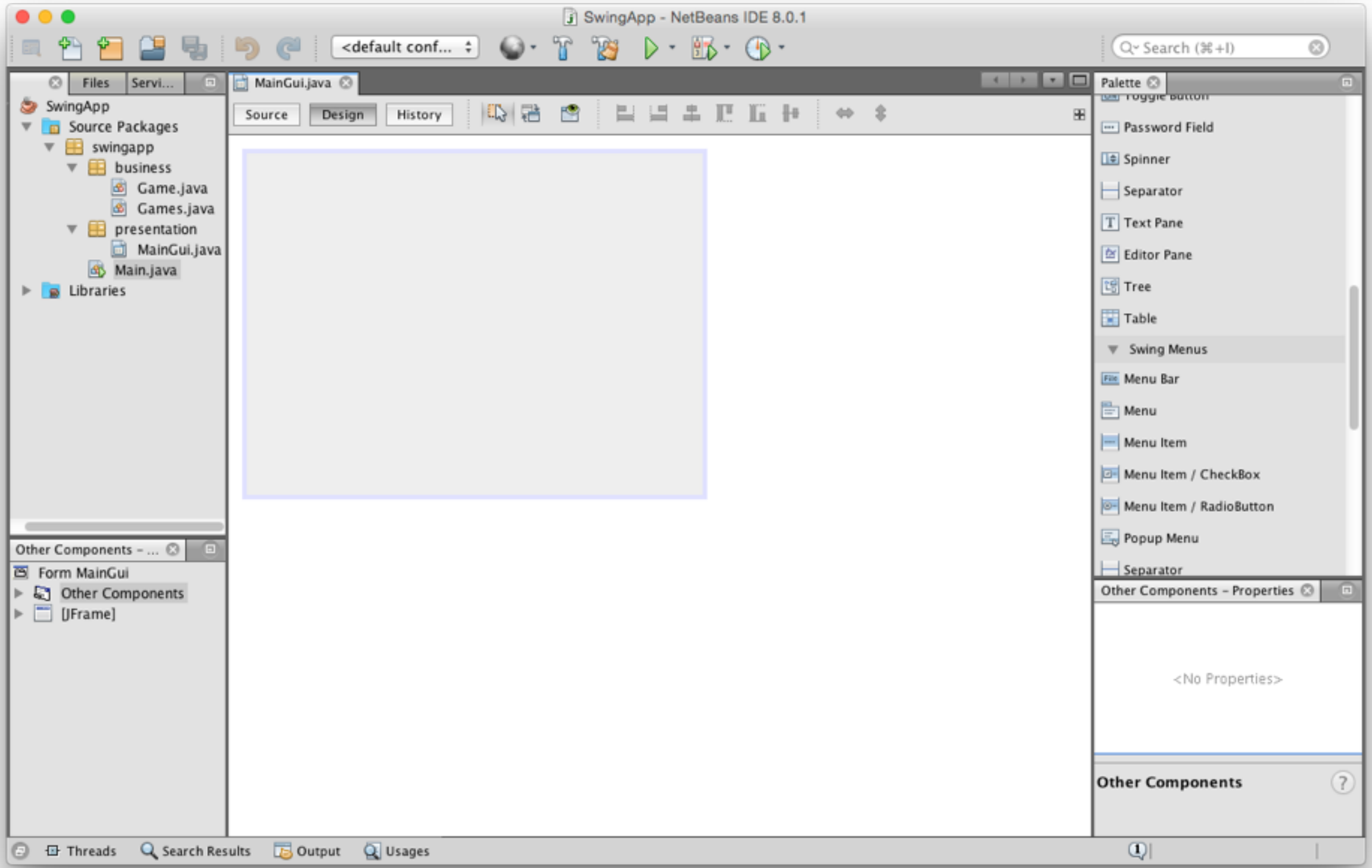

SwingApp - NetBeans IDE 8.0.1
<default conf...
Search (⌘+I)
Files
Servi...
SwingApp
Source Packages
swingapp
business
Game.java
Games.java
presentation
Libraries
Palette
Source Packages – Na...
<No View Available>
Threads
Search Results
Output
Usages

Example

- Create a new JFrameForm

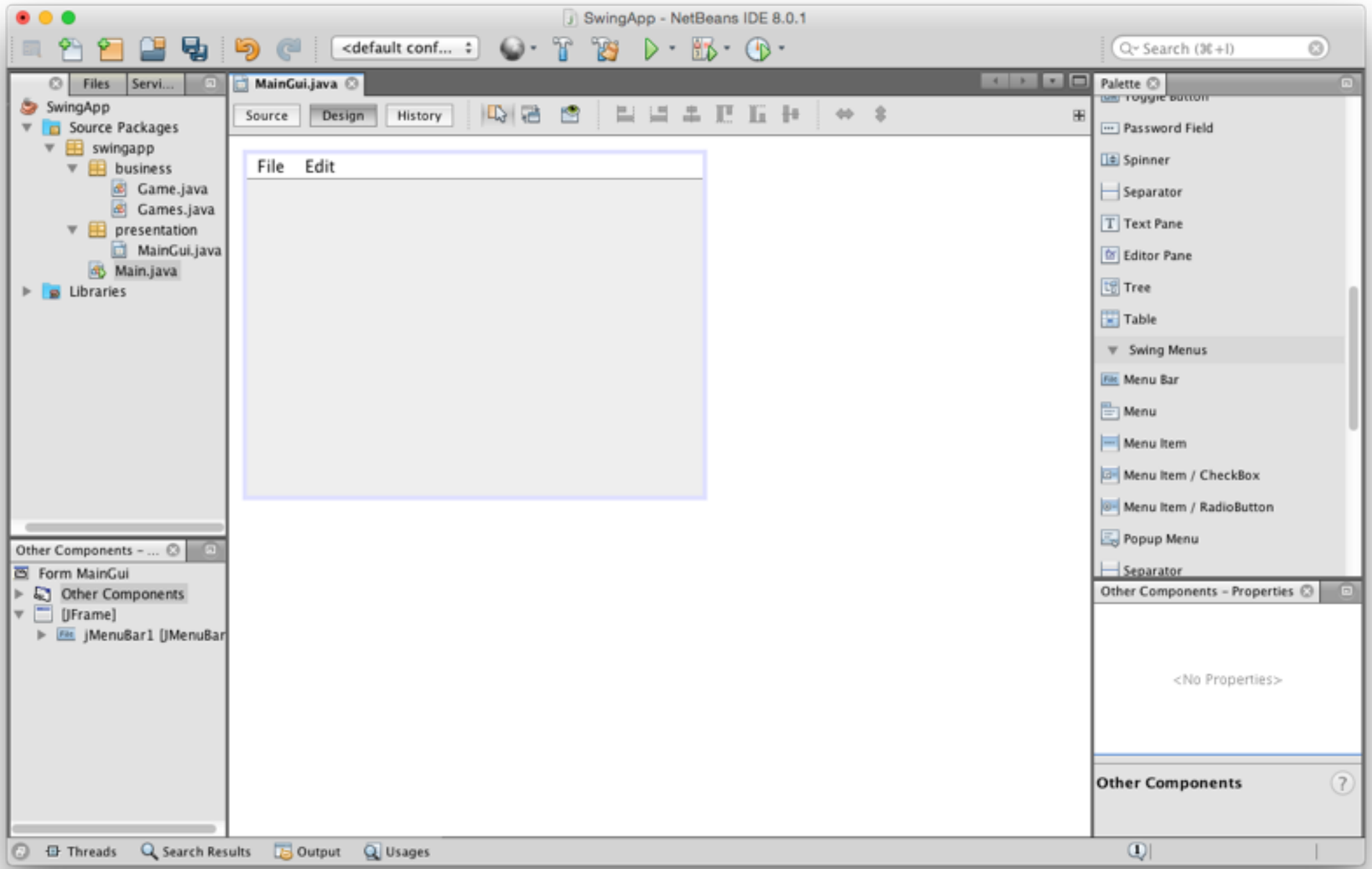


Example


SwingApp - NetBeans IDE 8.0.1
<default conf...
Search (⌘+I)
Files
Servi...
SwingApp
Source Packages
swingapp
business
Game.java
Games.java
presentation
MainGui.java
Main.java
Libraries
MainGui.java
Source
Design
History
Palette
Password Field
Spinner
Separator
Text Pane
Editor Pane
Tree
Table
Swing Menus
Menu Bar
Menu
Menu Item
Menu Item / CheckBox
Menu Item / RadioButton
Popup Menu
Separator
Other Components - ...
Form MainGui
Other Components
[JFrame]
Other Components - Properties
<No Properties>
Other Components
Threads
Search Results
Output
Usages

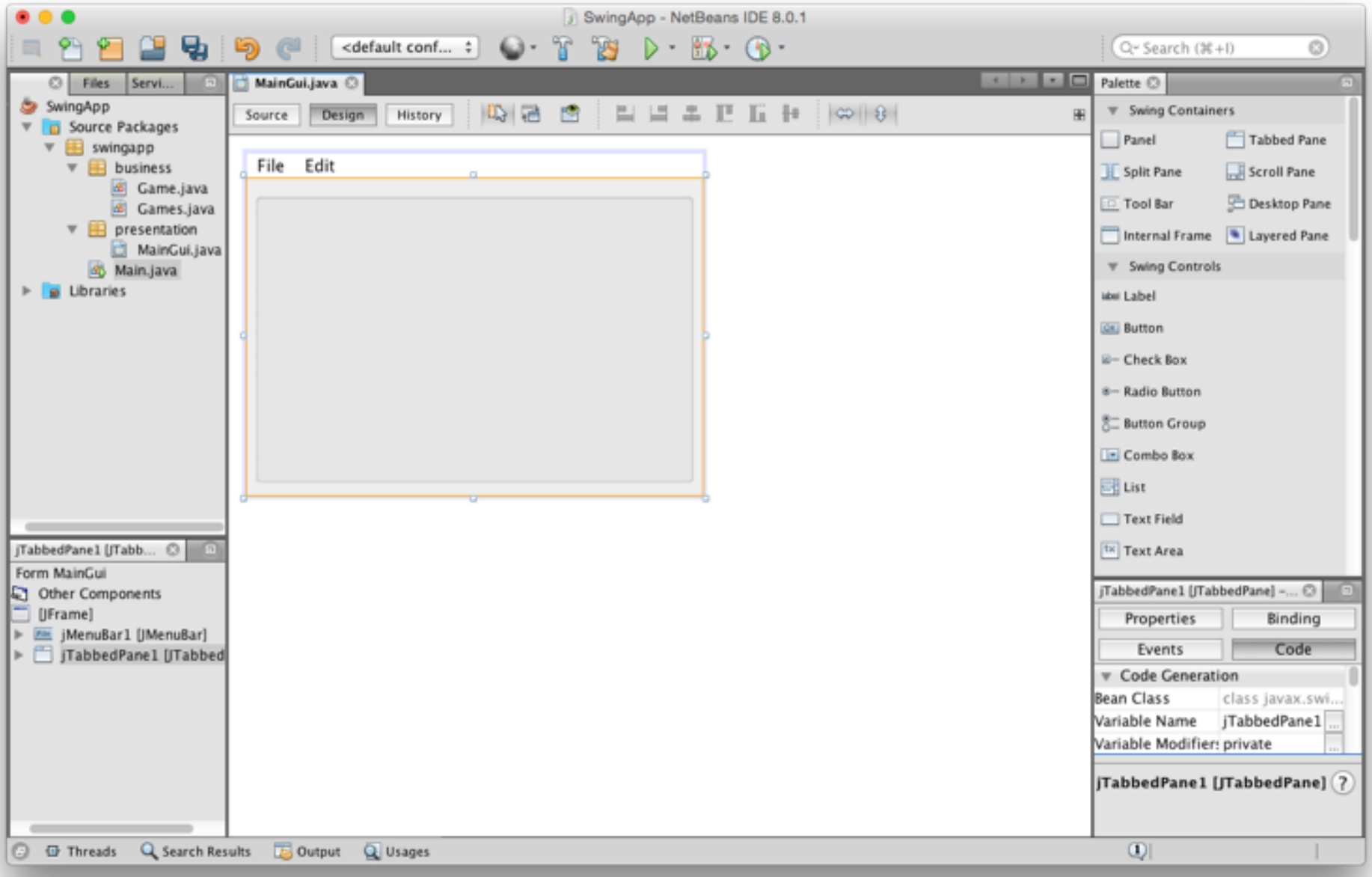
Example

- Add a menu



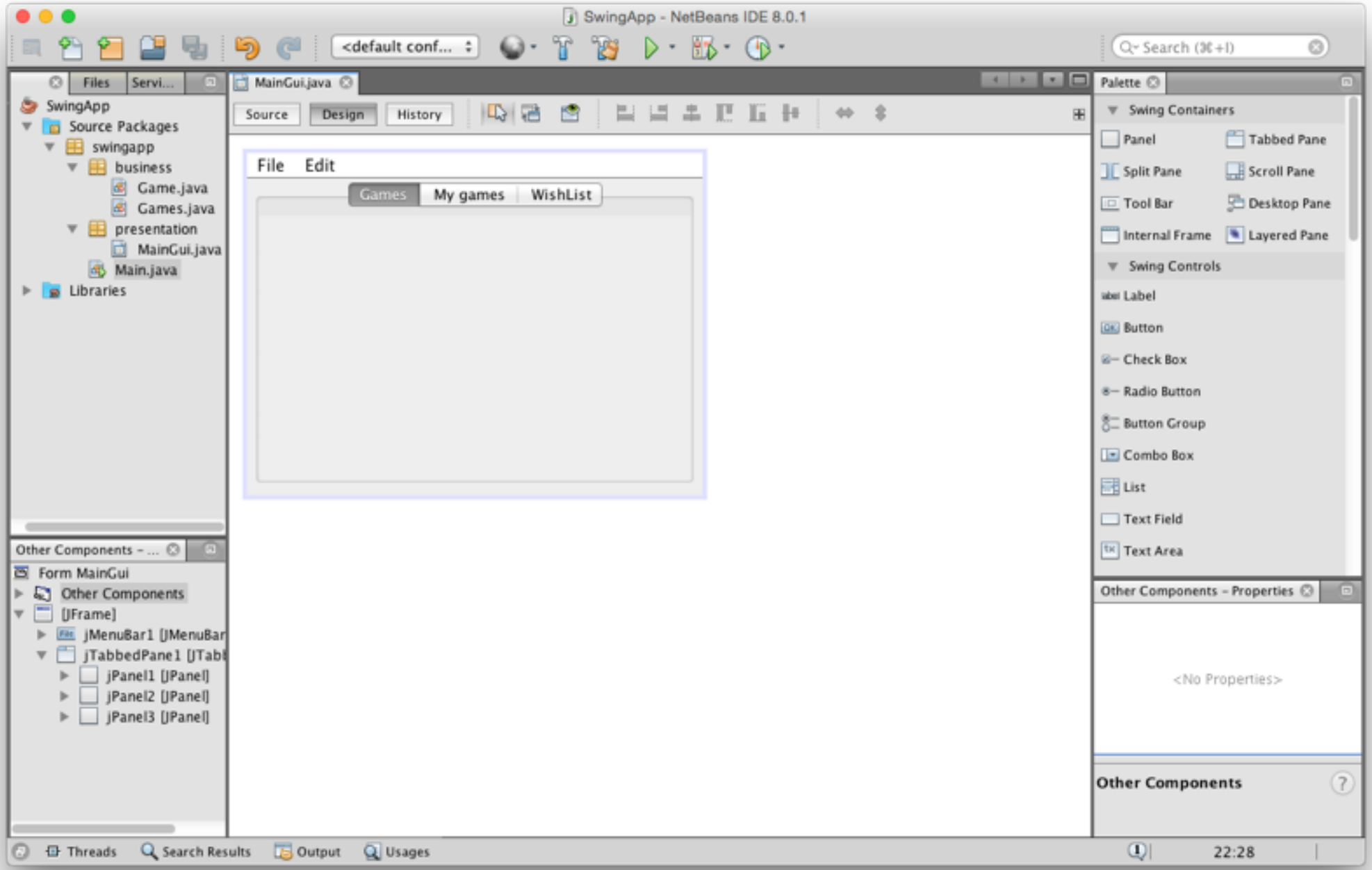
Example

- Add a tabbed pane



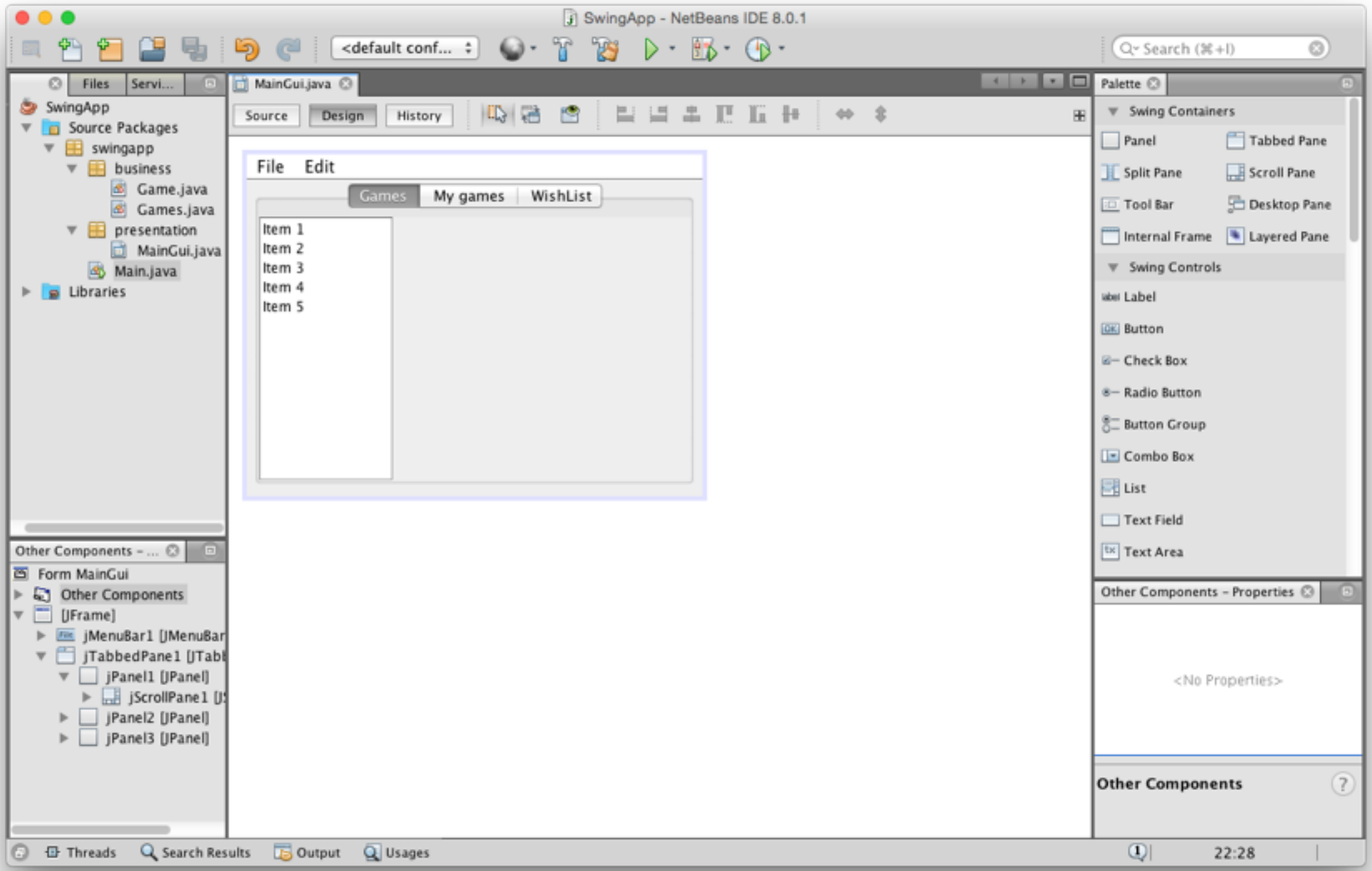
Example

- Add three Panels



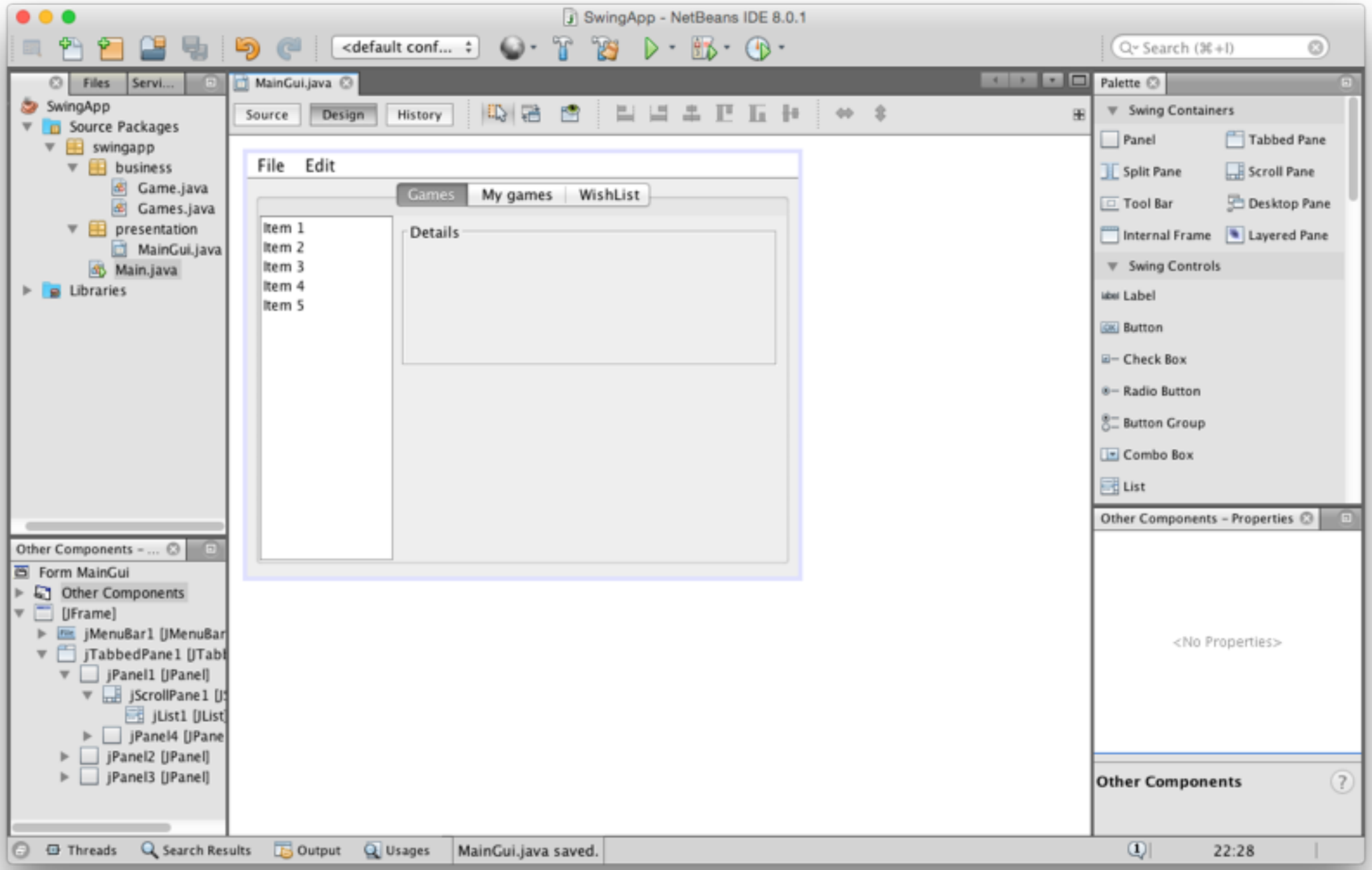
Example

- Add a list



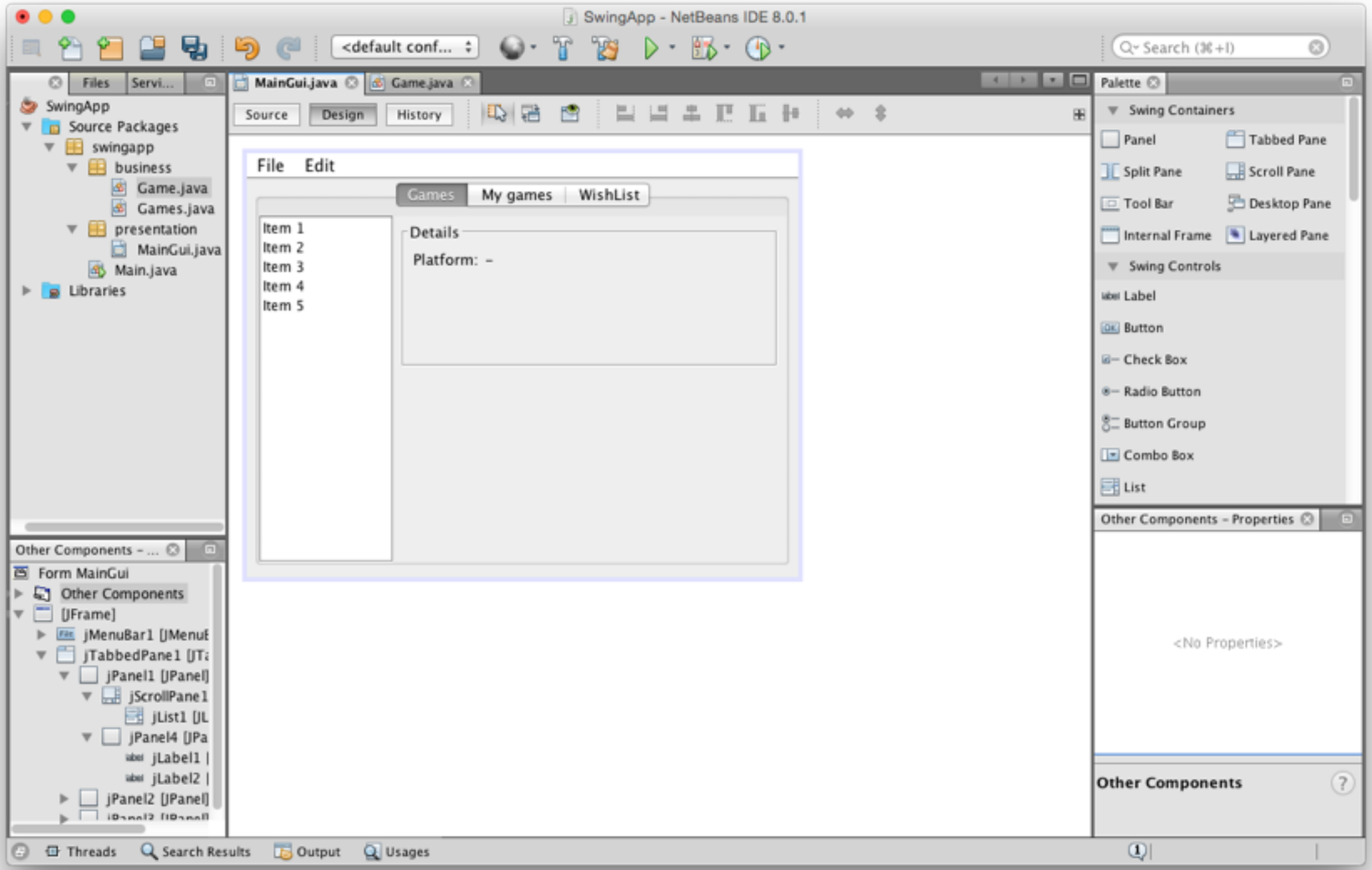
Example

- Add another panel, set border to labeled



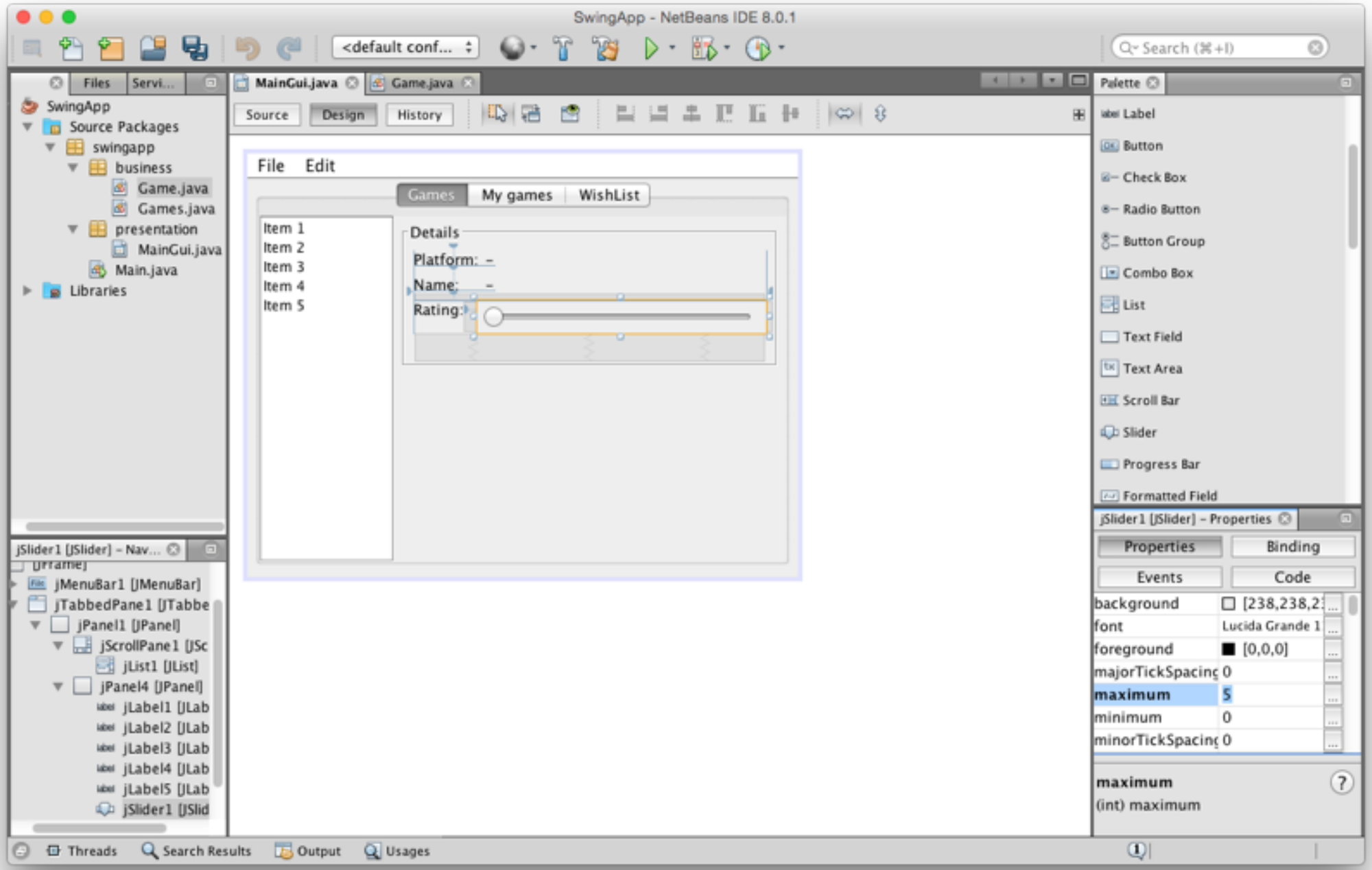
Example

- Add two labels



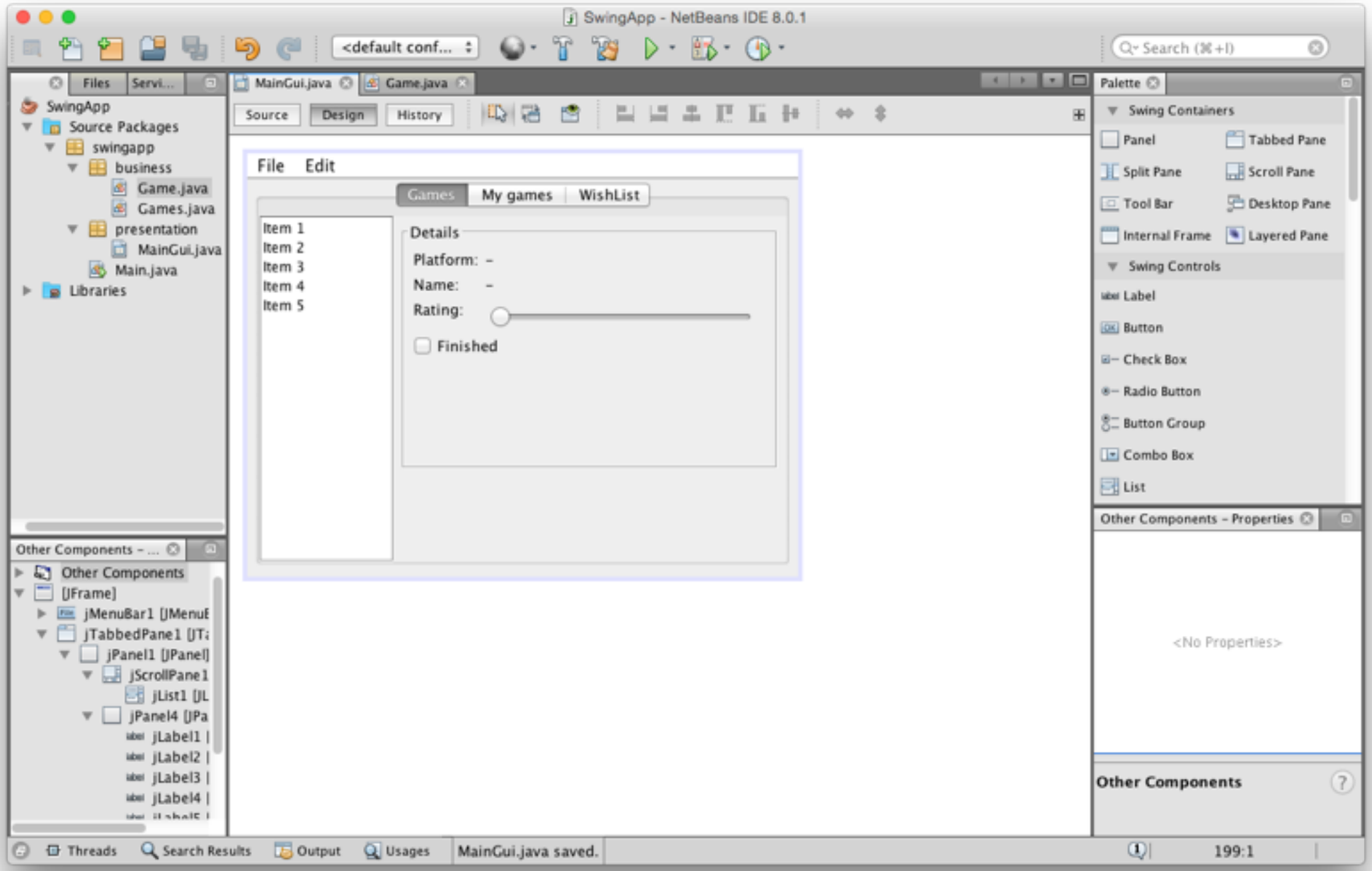
Example

- Add some more labels and a slider



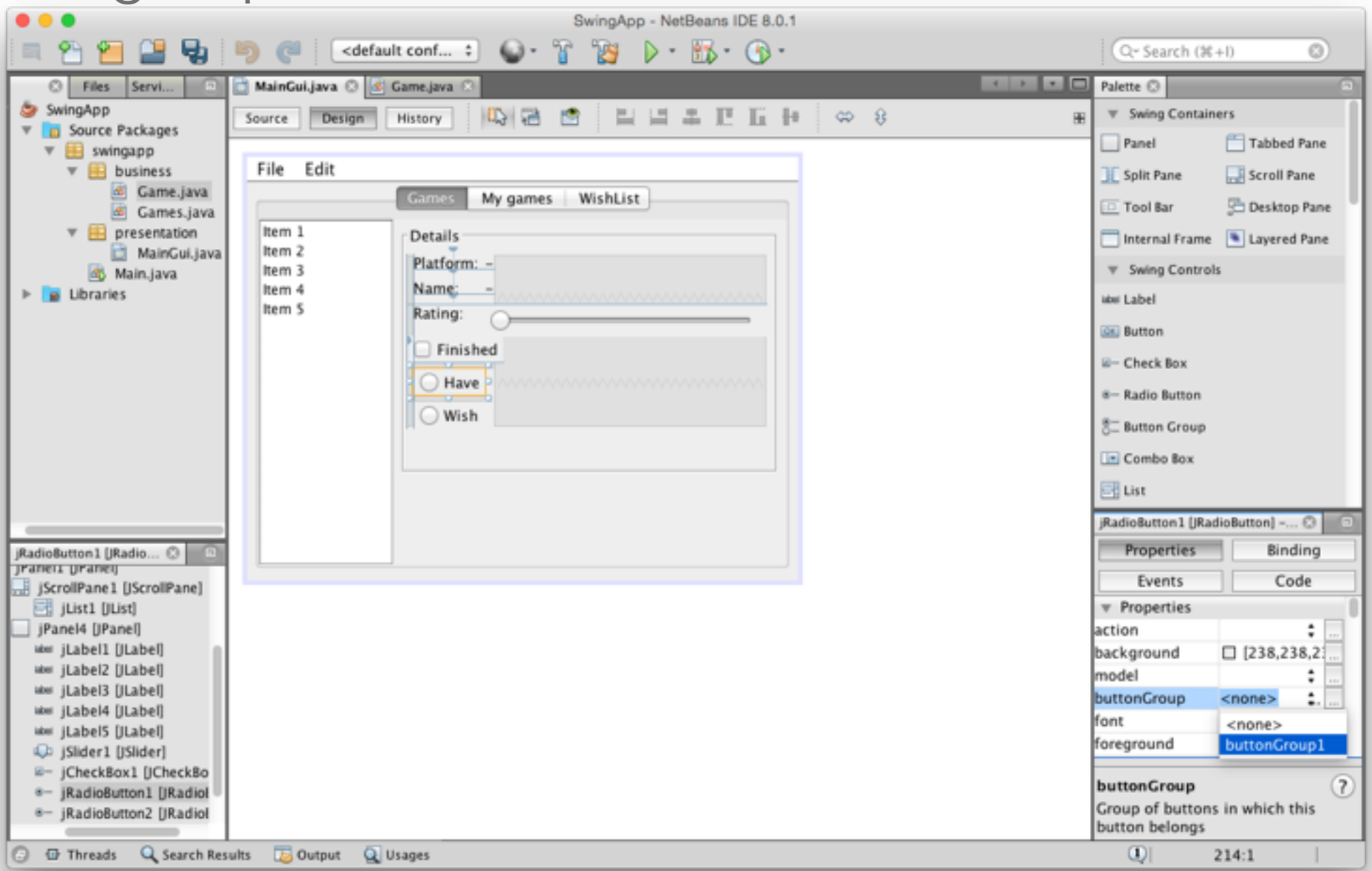
Example

- Add checkbox



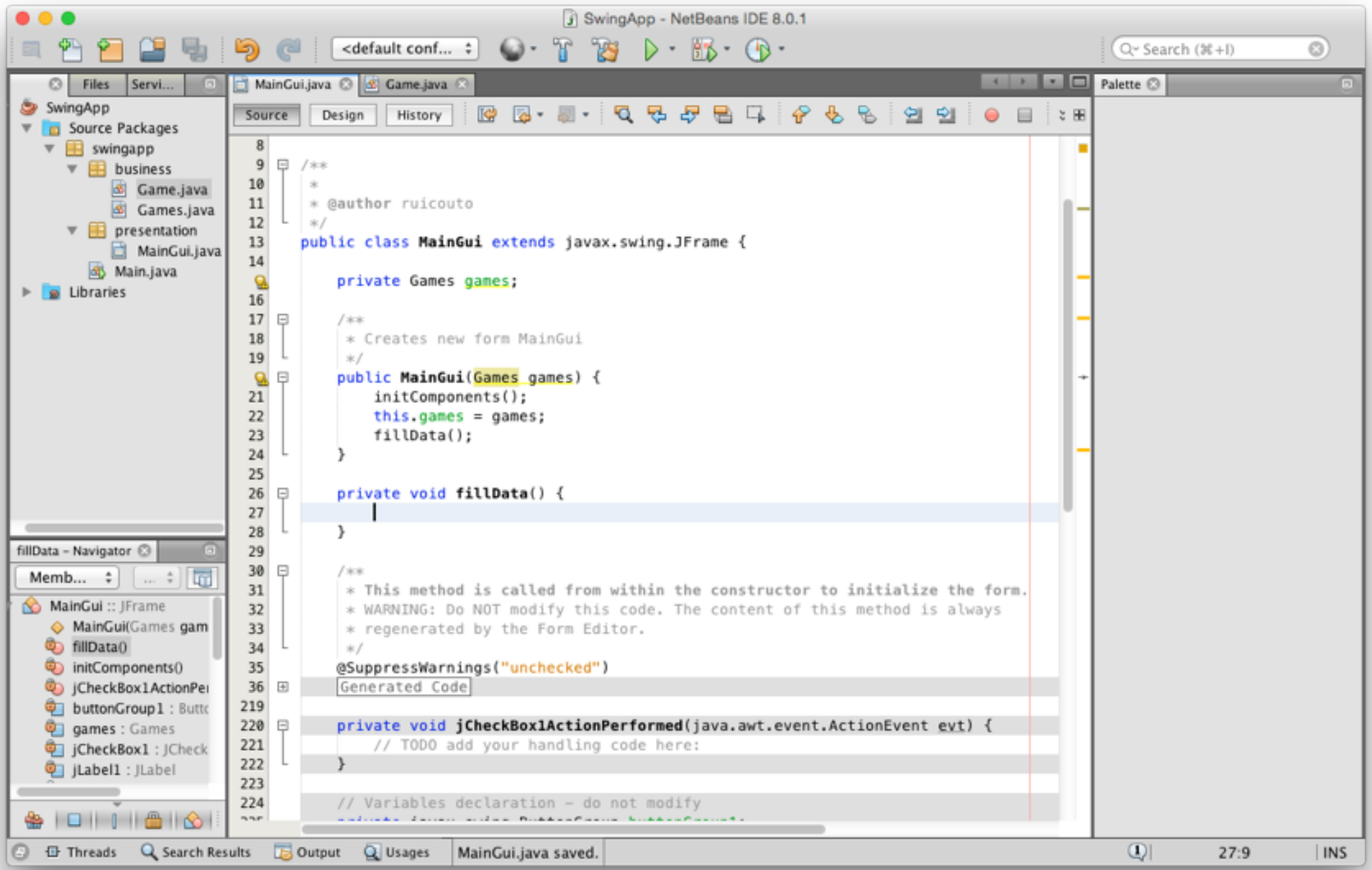
Example

- Add radioButton group and two radio buttons
- Set the group



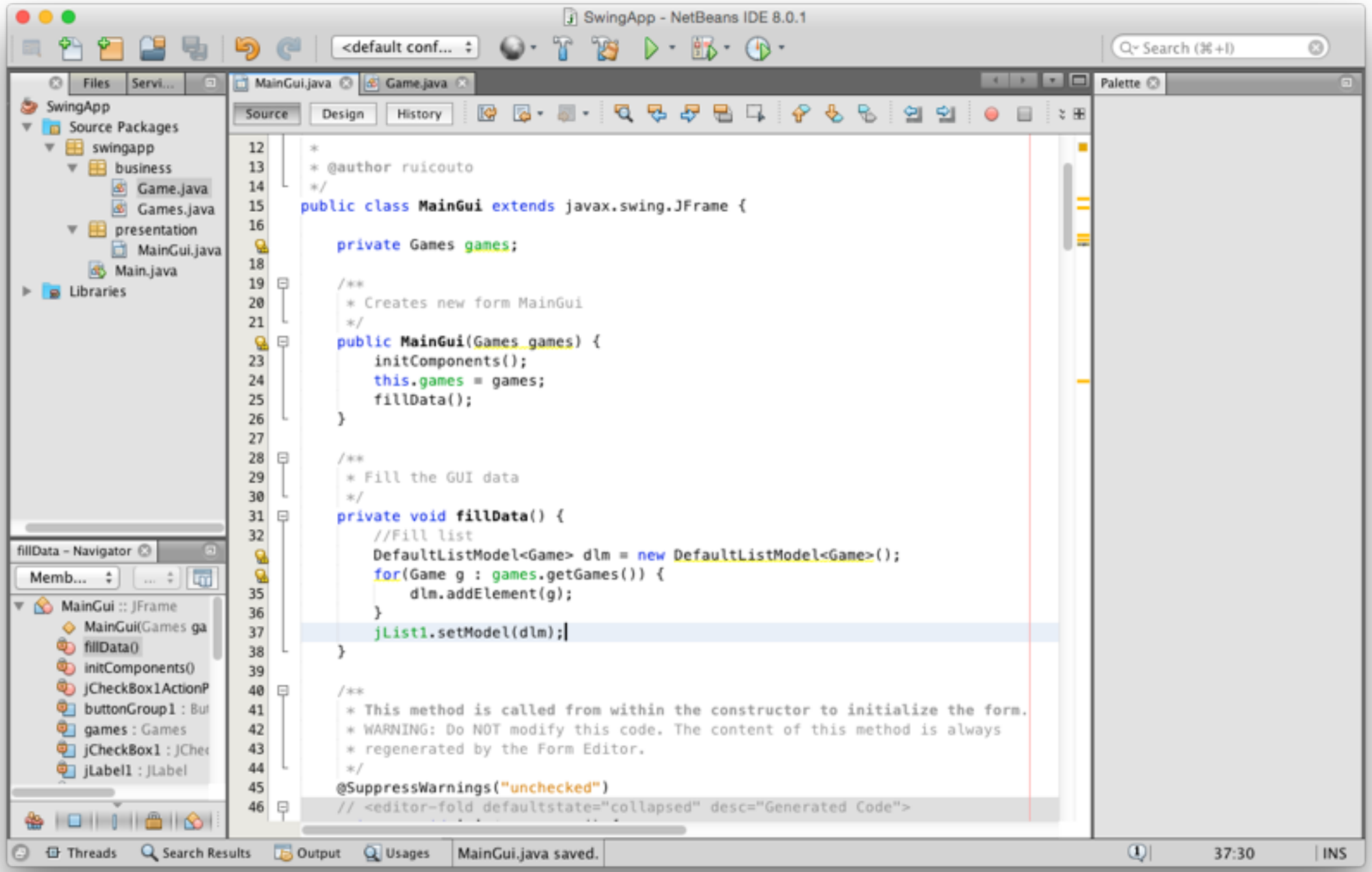
Example

- Create method to initialize data



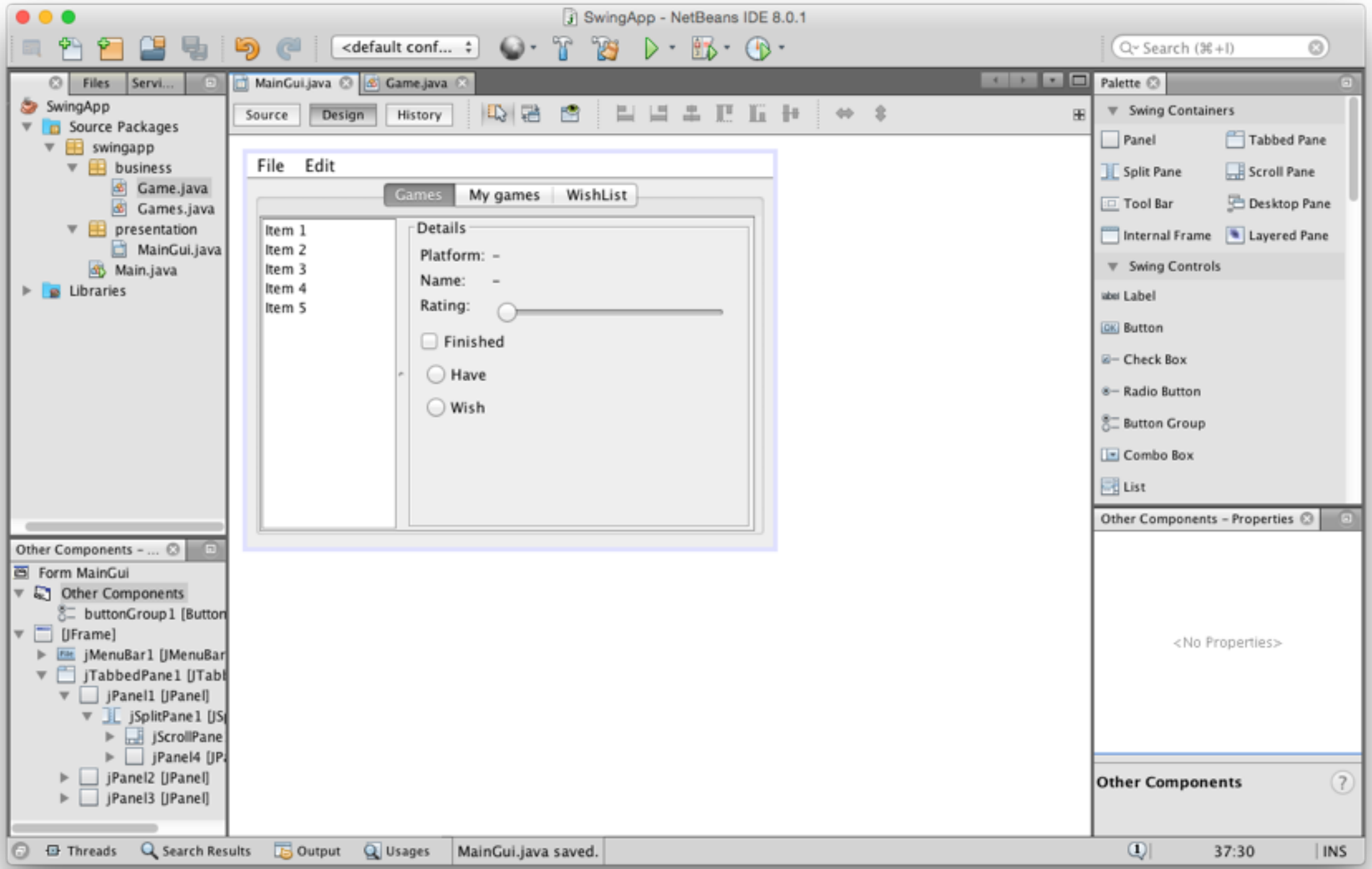
Example

- Create list model to fill the list and set data



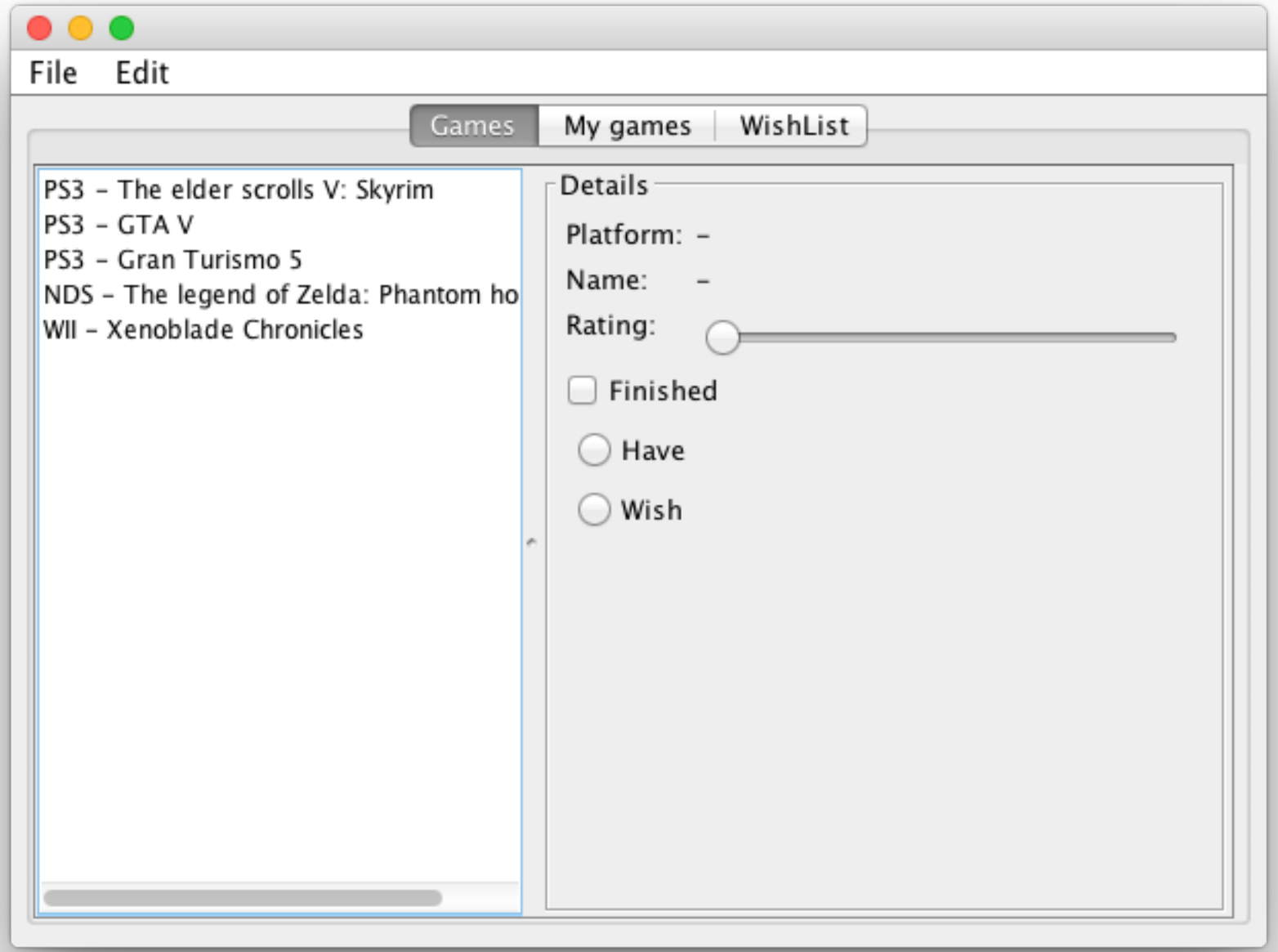
Example

- Added split pane



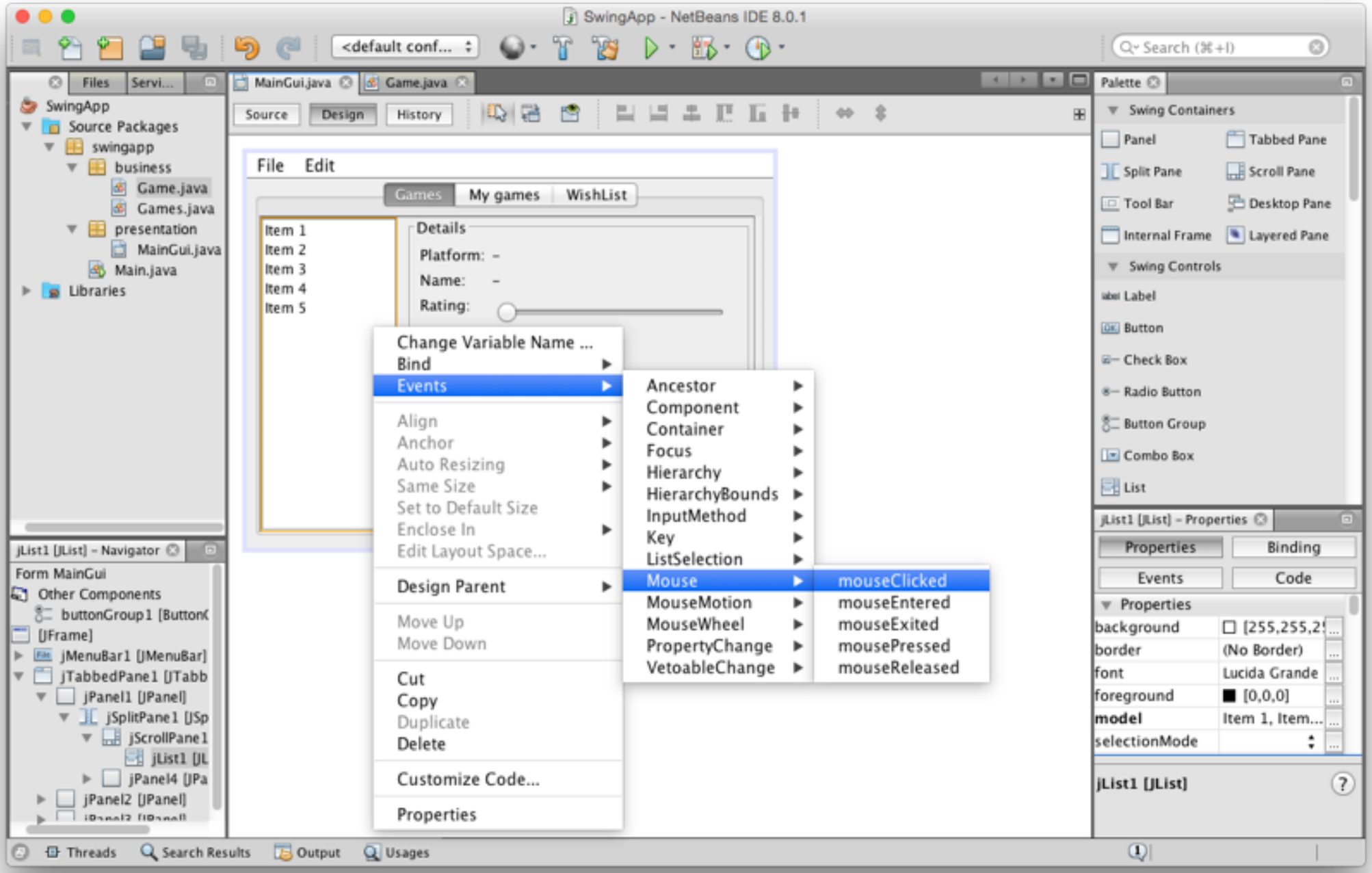
Example

- Running example



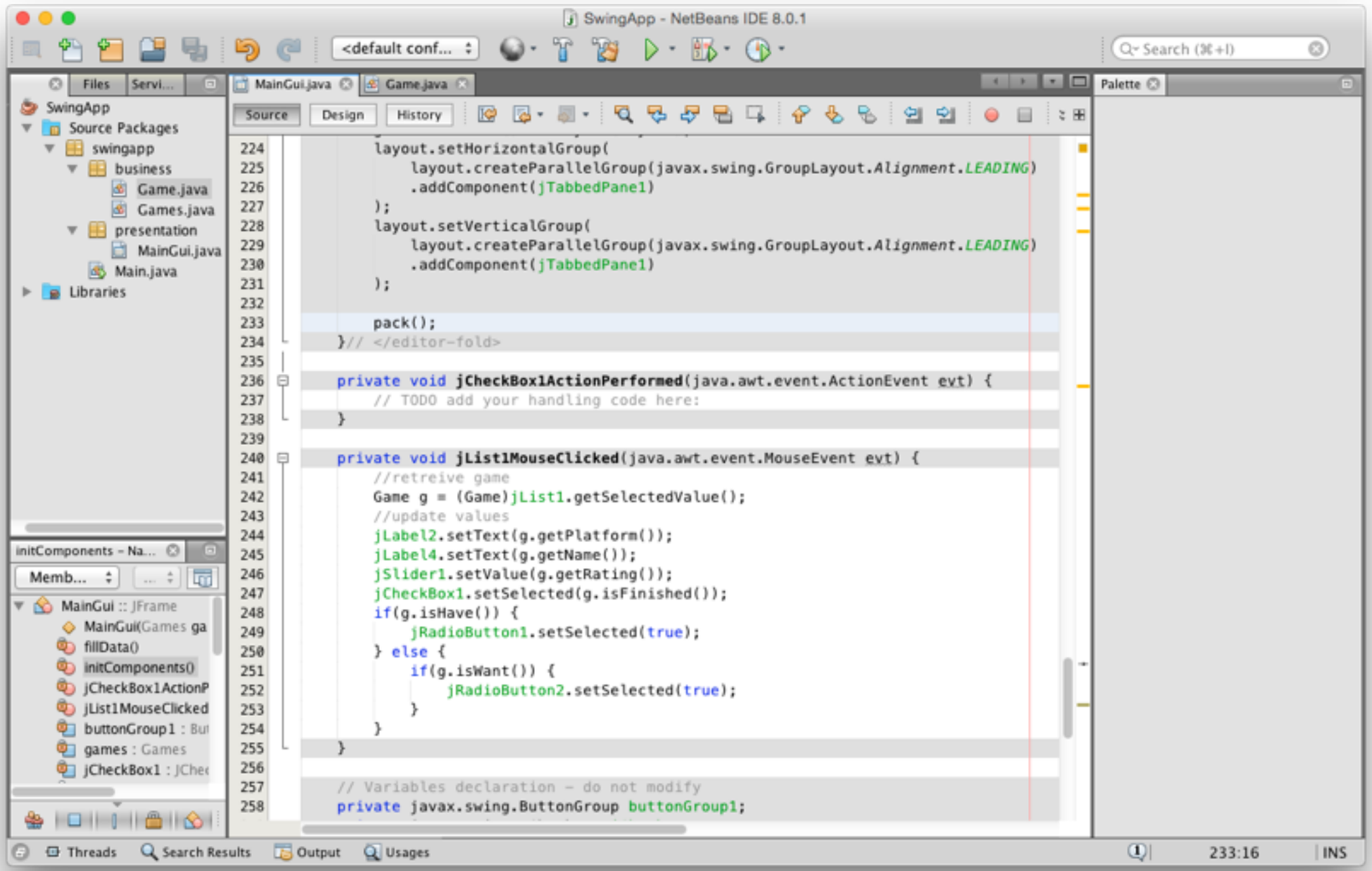
Example

- Add list click event



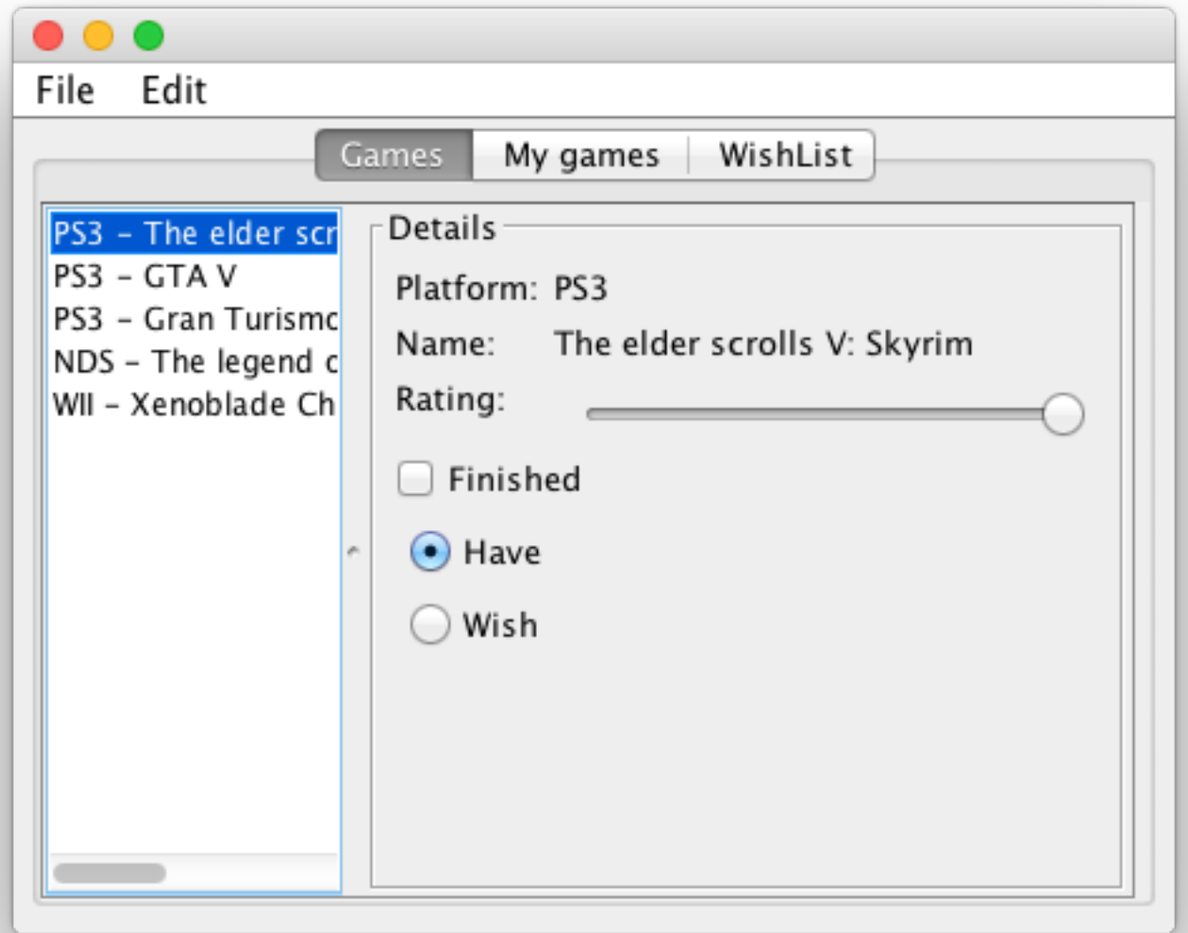
Example

- Create code to display data



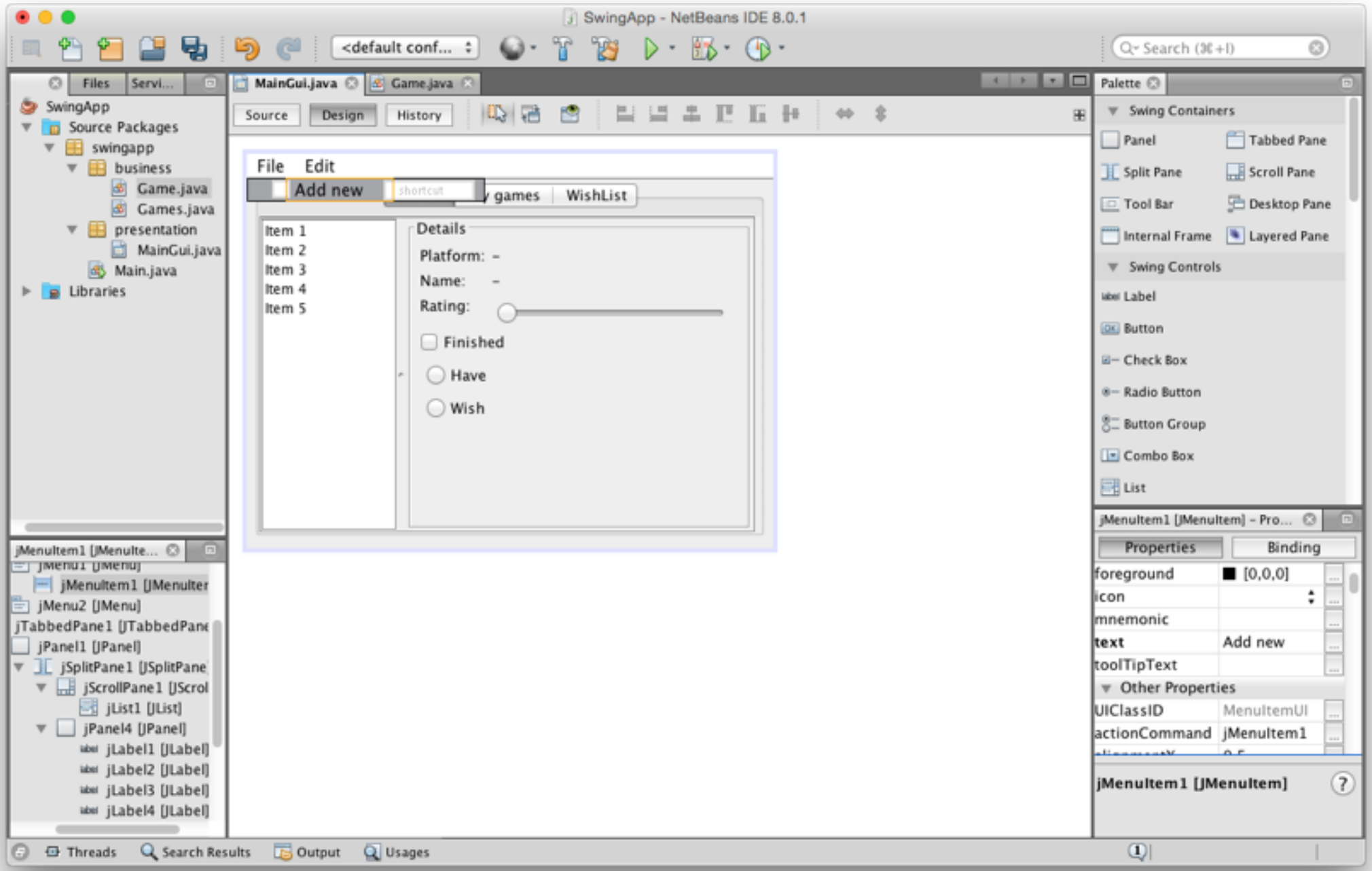
Example

- Second running example



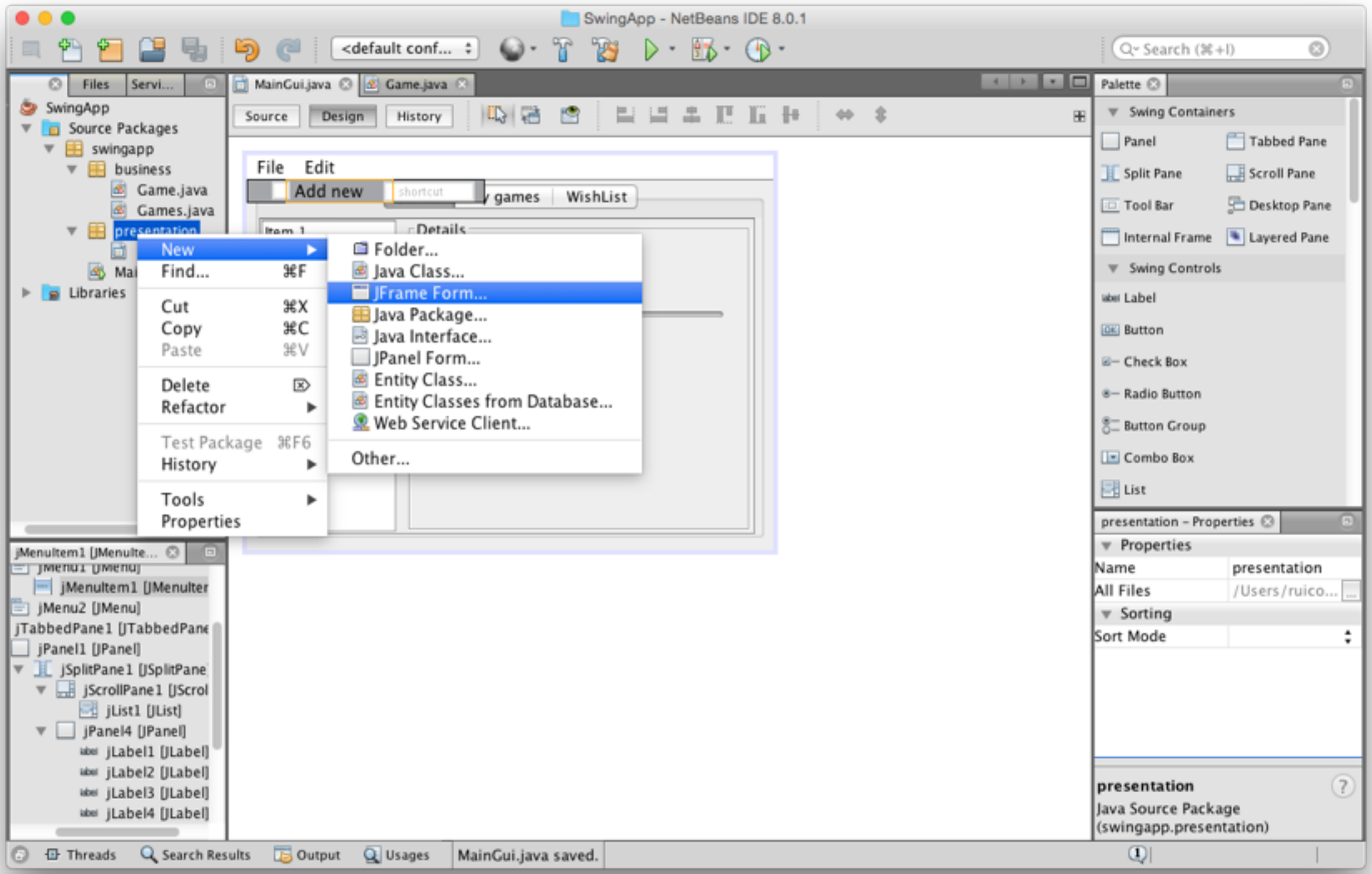
Example

- Add menu entry to create game



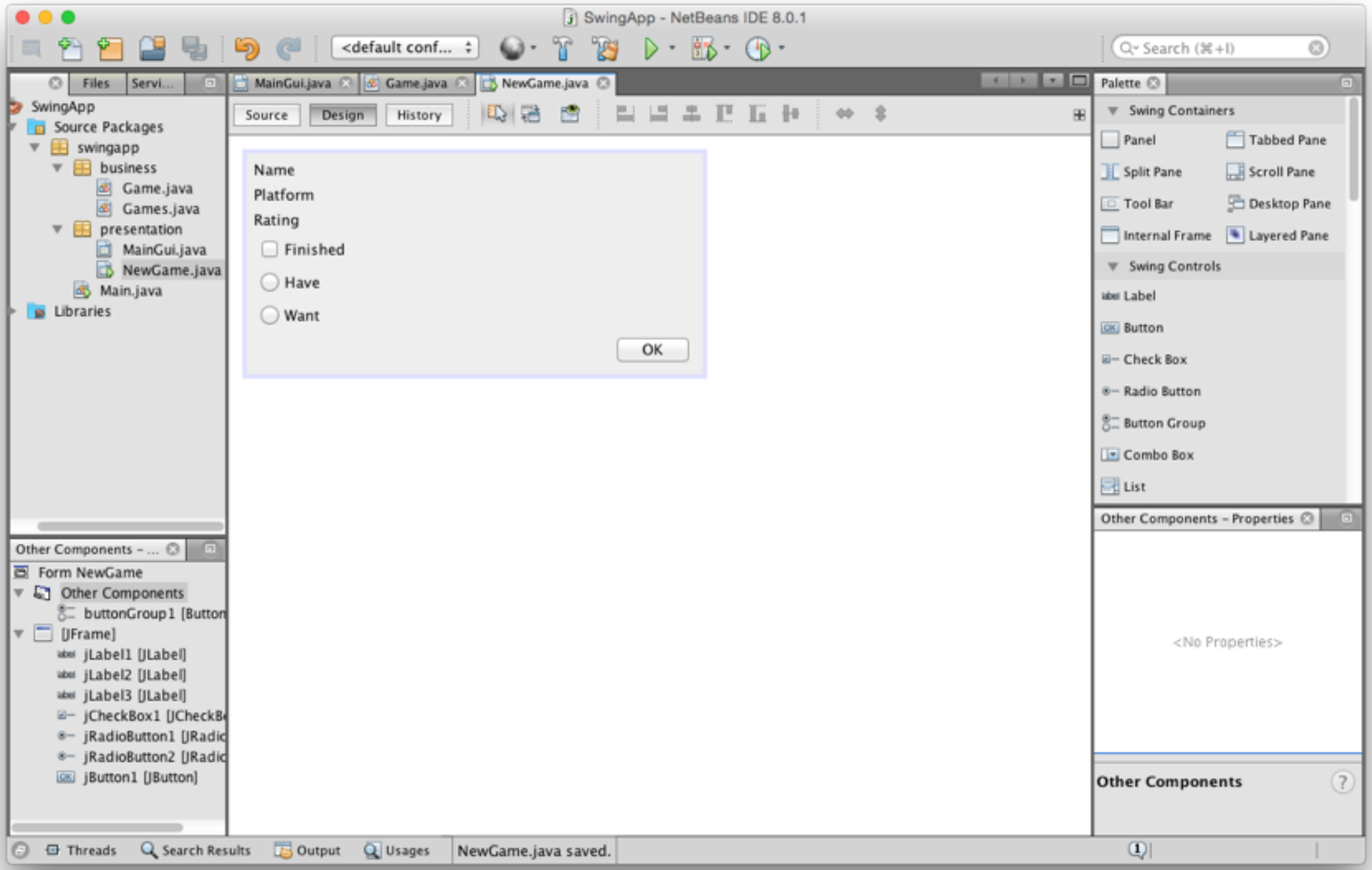
Example

- Create a new JFrame



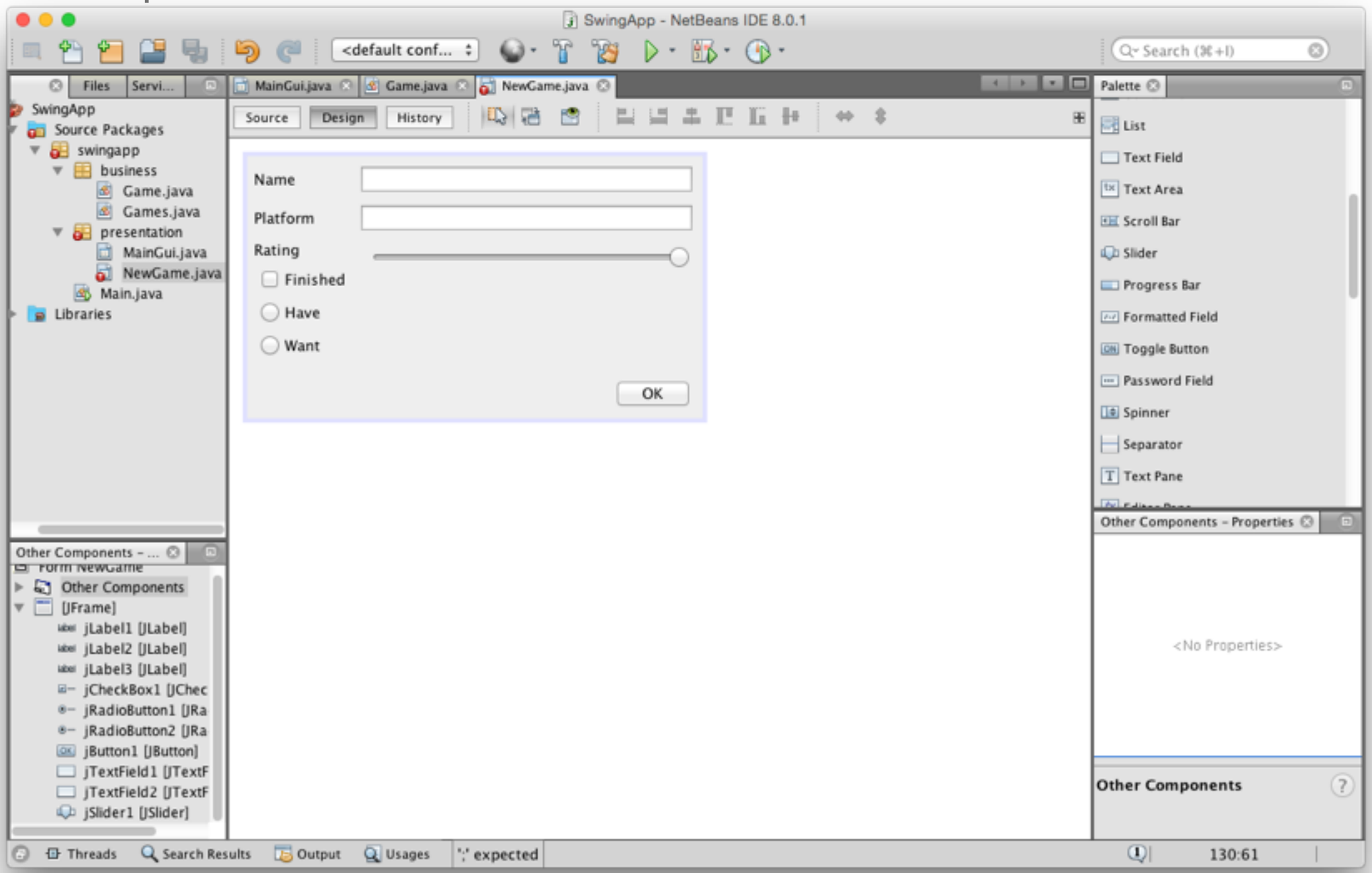
Example

- Add required components



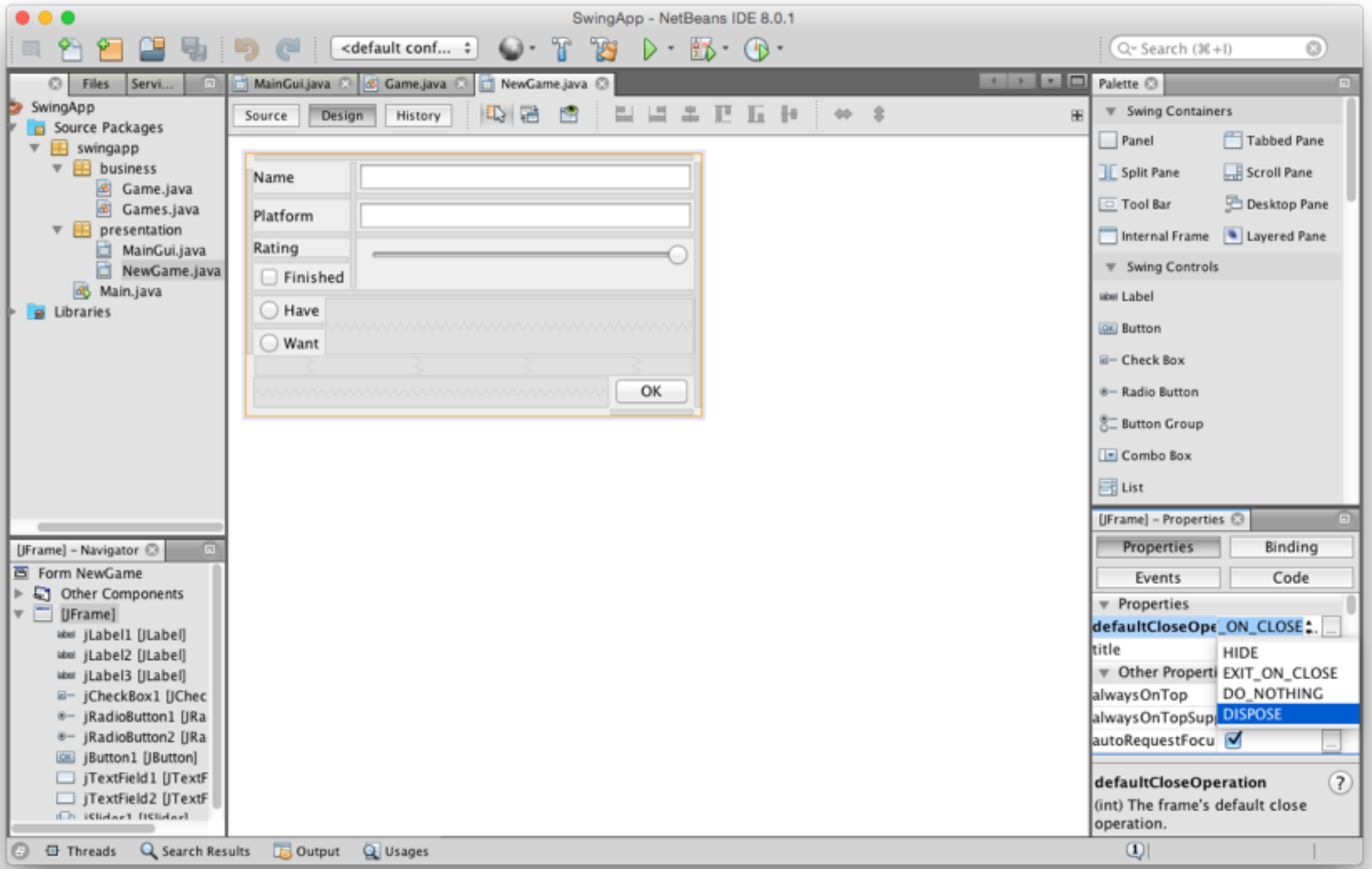
Example

- Add input fields



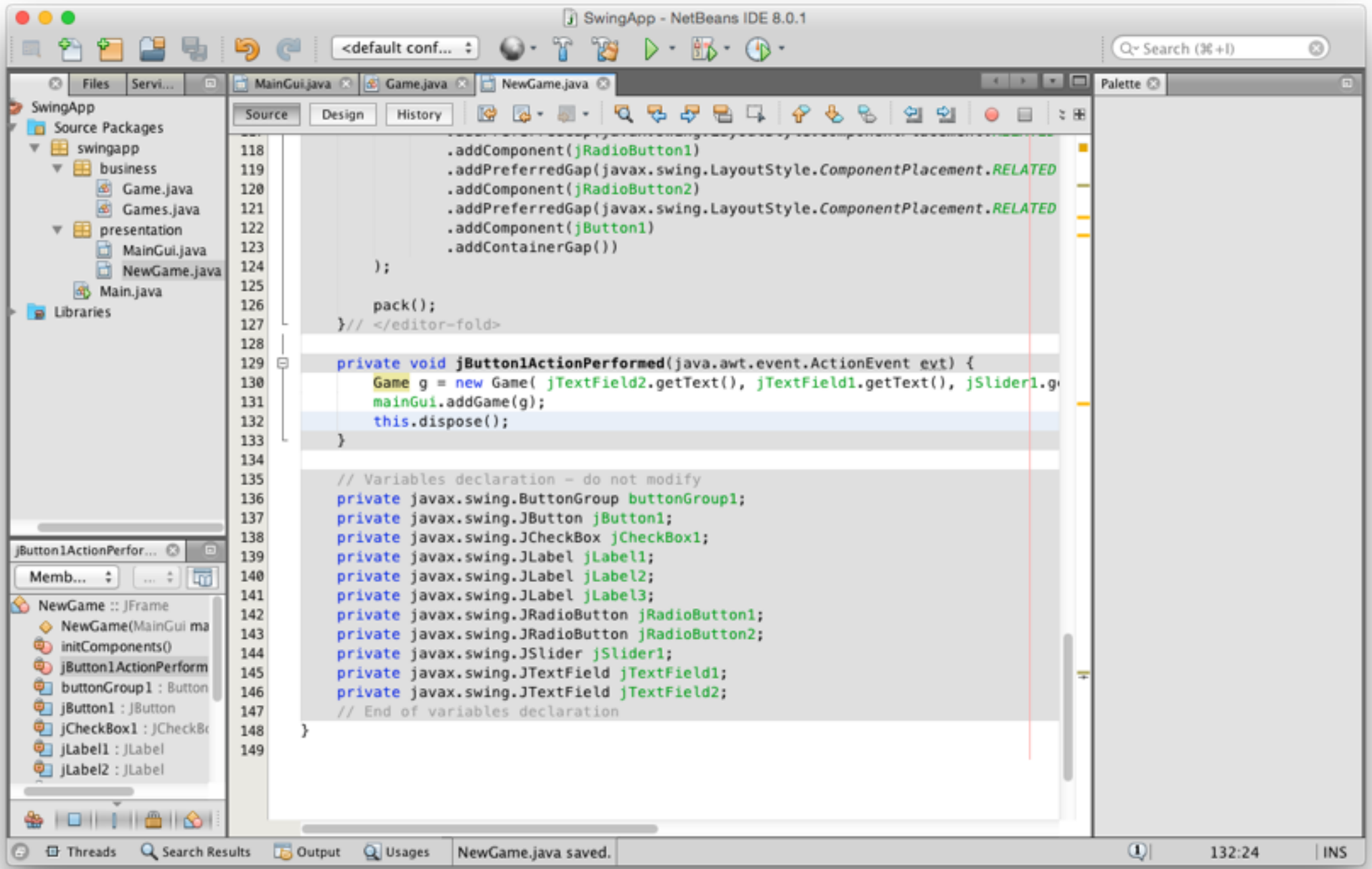
Example

- Set dispose action



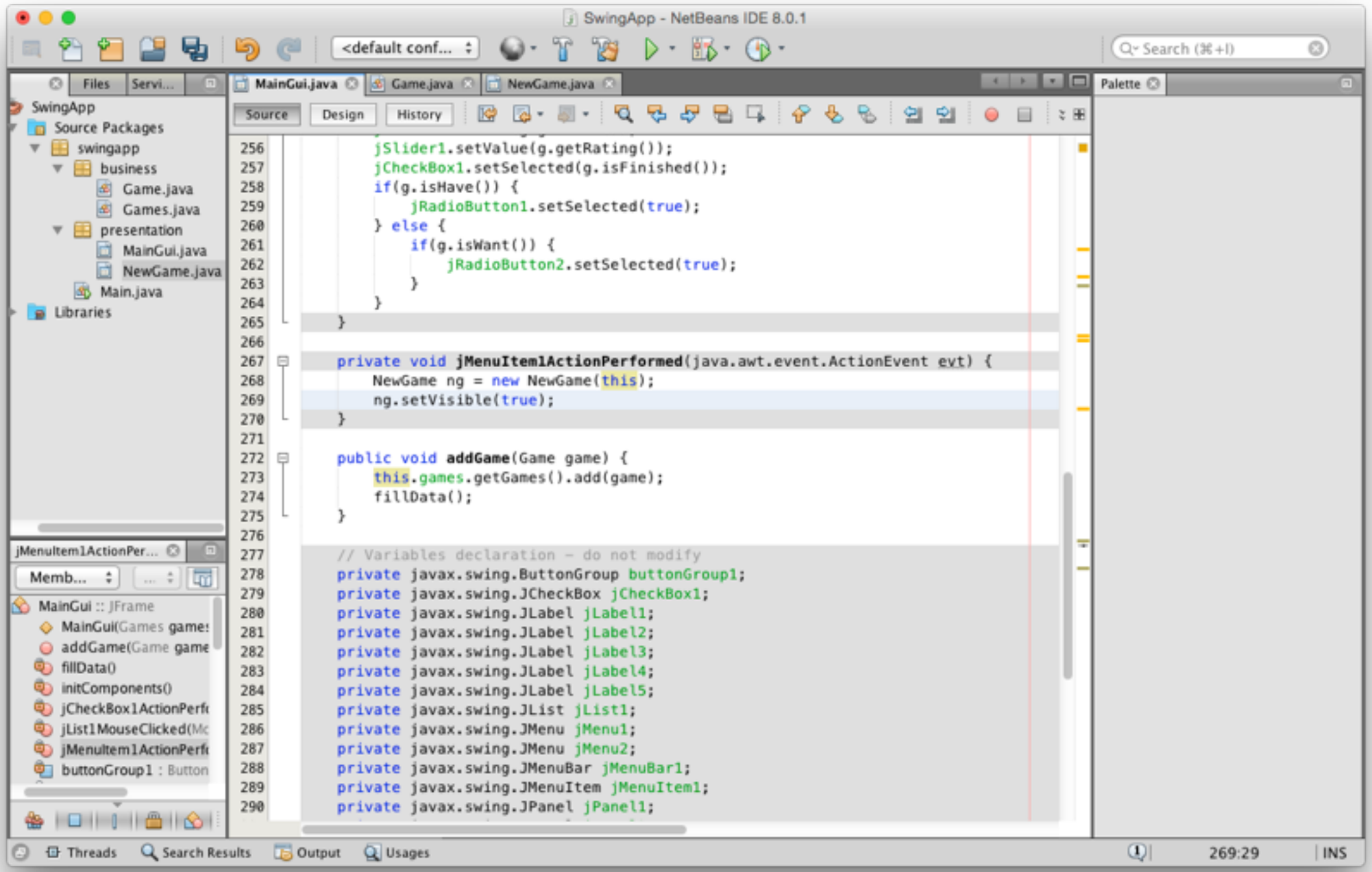
Example

- Create method to handle new game in **MainGui**



Example

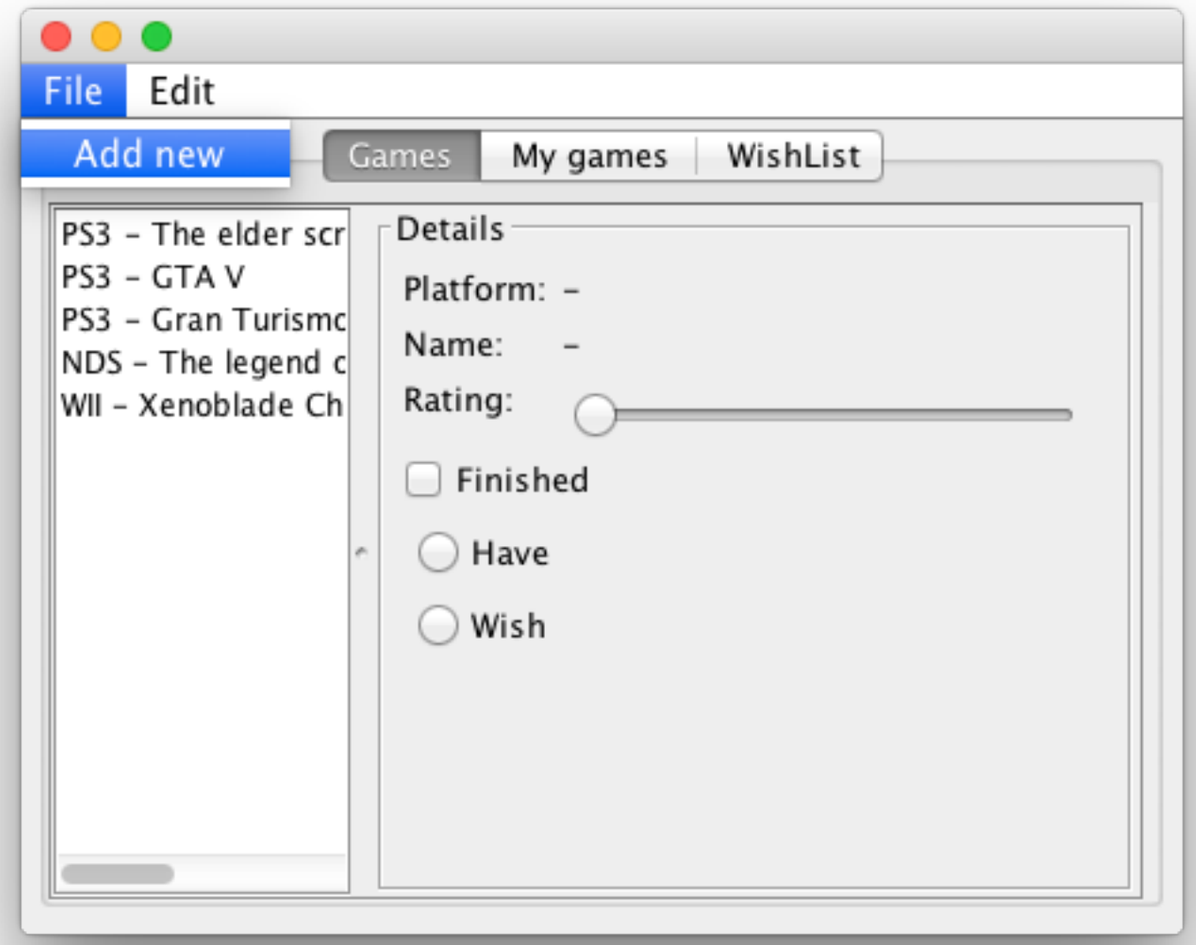
- Add event to show new window



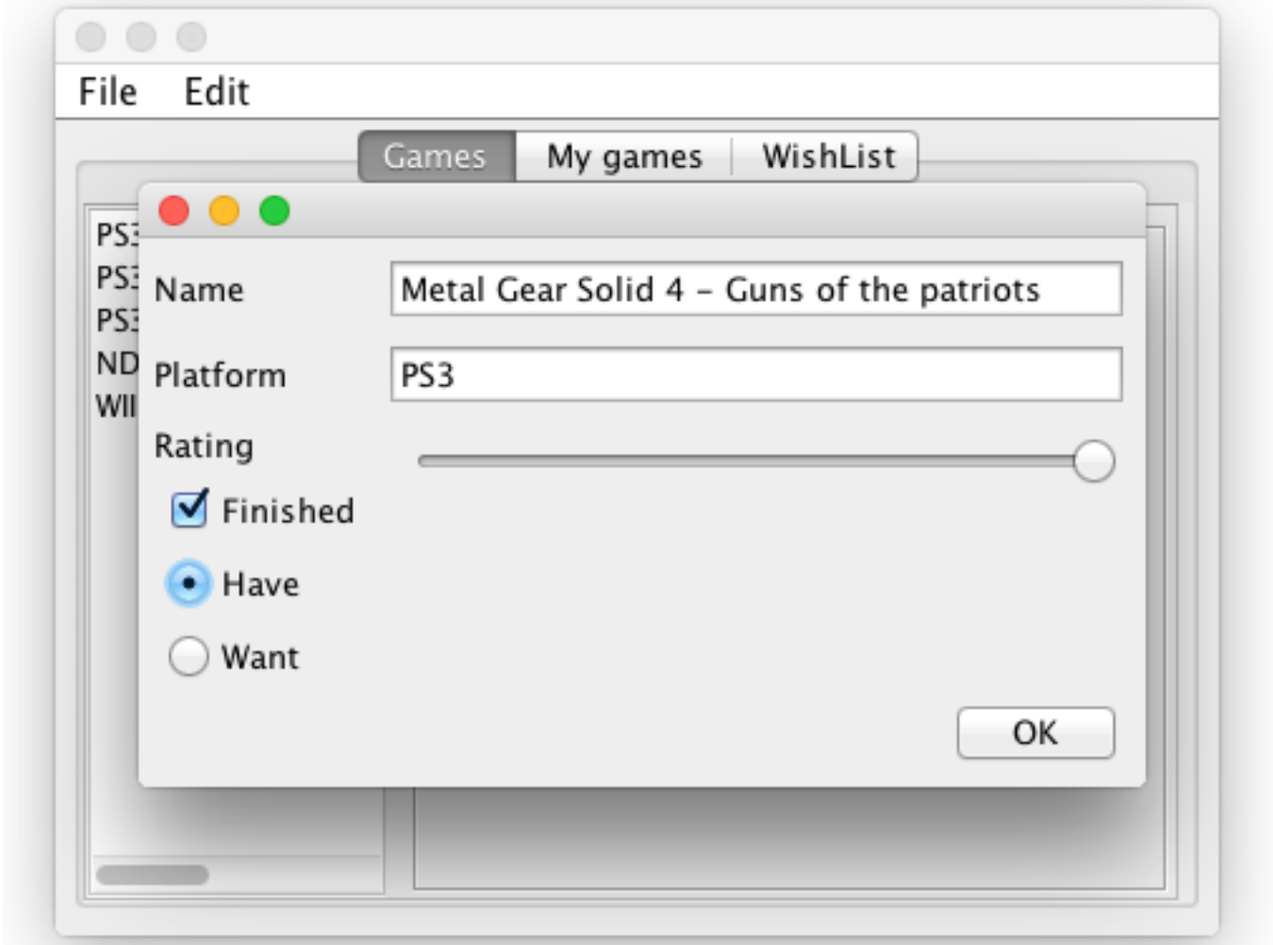

```
256             jSlider1.setValue(g.getRating());
257             jCheckBox1.setSelected(g.isFinished());
258             if(g.isHave()) {
259                 jRadioButton1.setSelected(true);
260             } else {
261                 if(g.isWant()) {
262                     jRadioButton2.setSelected(true);
263                 }
264             }
265         }
266
267         private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
268             NewGame ng = new NewGame(this);
269             ng.setVisible(true);
270         }
271
272         public void addGame(Game game) {
273             this.games.getGames().add(game);
274             fillData();
275         }
276
277         // Variables declaration - do not modify
278         private javax.swing.ButtonGroup buttonGroup1;
279         private javax.swing.JCheckBox jCheckBox1;
280         private javax.swing.JLabel jLabel1;
281         private javax.swing.JLabel jLabel2;
282         private javax.swing.JLabel jLabel3;
283         private javax.swing.JLabel jLabel4;
284         private javax.swing.JLabel jLabel5;
285         private javax.swing.JList jList1;
286         private javax.swing.JMenu jMenu1;
287         private javax.swing.JMenu jMenu2;
288         private javax.swing.JMenuBar jMenuBar1;
289         private javax.swing.JMenuItem jMenuItem1;
290         private javax.swing.JPanel jPanel1;
```

Example

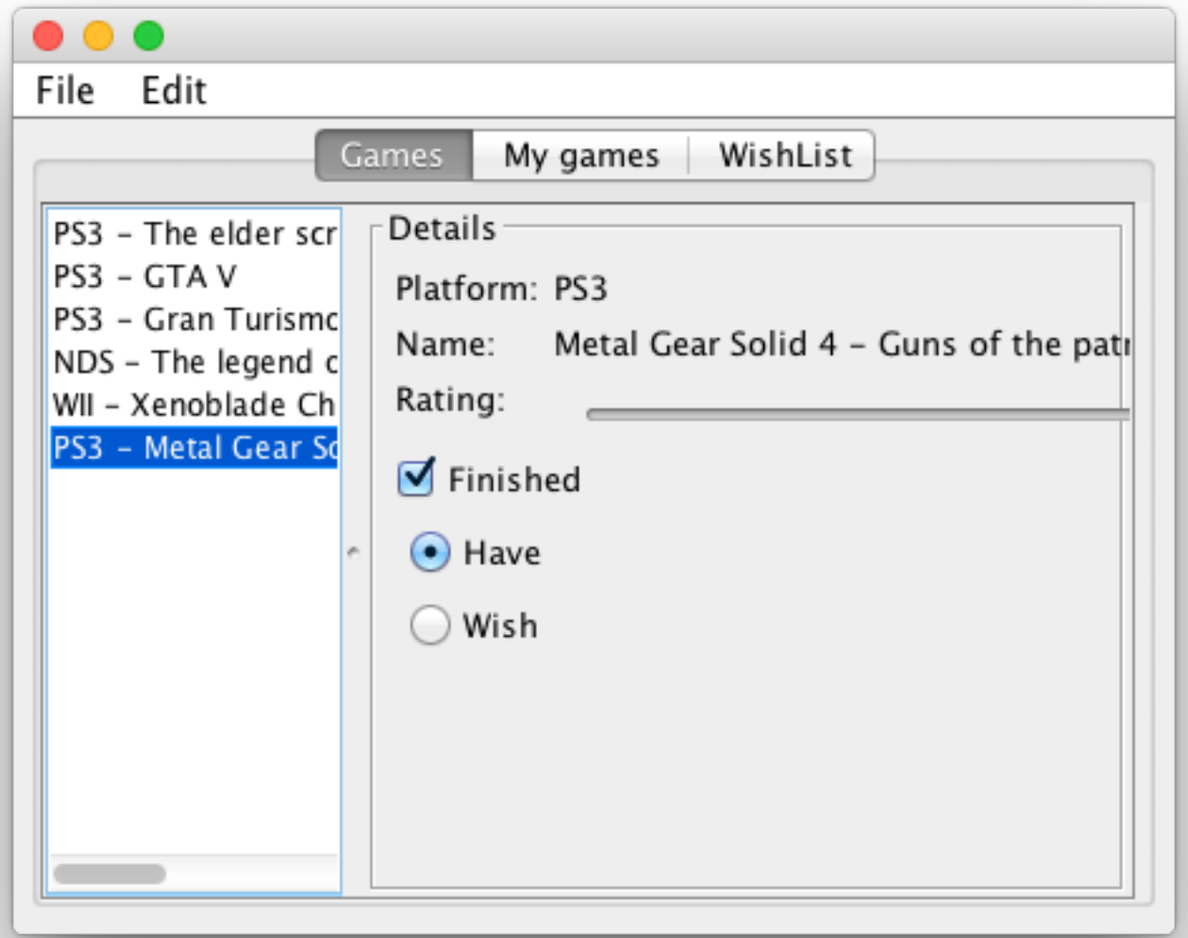
- Final running example

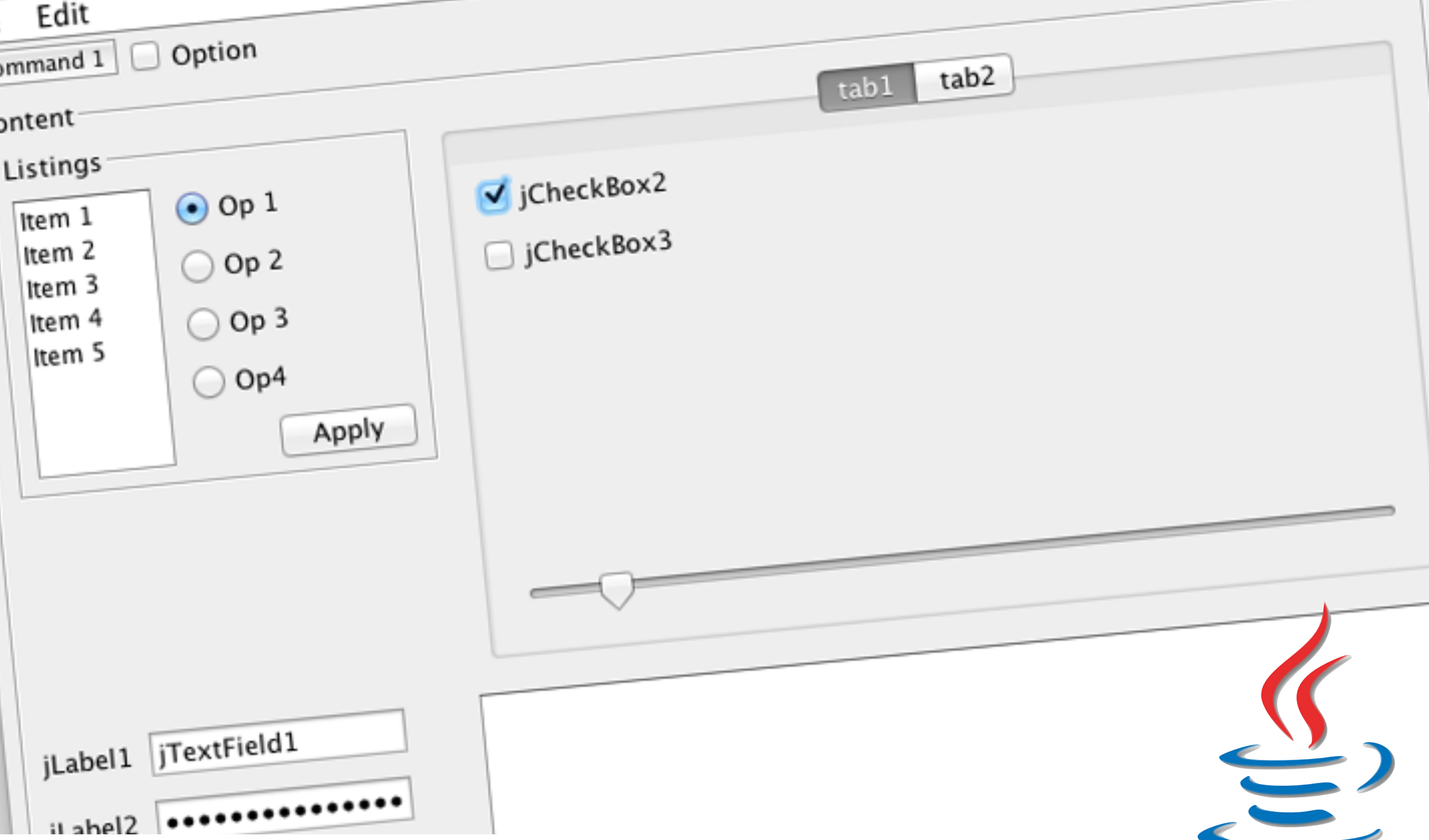


Example


File
Edit
Games
My games
WishList
Name
Metal Gear Solid 4 - Guns of the patriots
Platform
PS3
Rating
Finished
Have
Want
OK

Example


File
Edit
Games
My games
WishList
PS3 - The elder scr
PS3 - GTA V
PS3 - Gran Turismc
NDS - The legend c
WII - Xenoblade Ch
PS3 - Metal Gear Sc
Details
Platform: PS3
Name: Metal Gear Solid 4 - Guns of the patr
Rating:
Finished
Have
Wish



Java swing

Rui Couto
rui.couto@di.uminho.pt