

61FIT3JSD

Fall 2022

Lecture 7

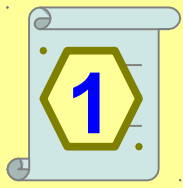
GUI programming (2)
Intermediate issues

Lecture outline

- Basic components
- GUI containers
- Content container
- Menu (container)
- More display components

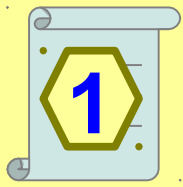
Lecture outline

- Basic components
- GUI containers
- Content container
- Menu (container)
- More display components



Basic display components

- Label
- Text field



Basic display components

- Label
- Text field

Labelled text field

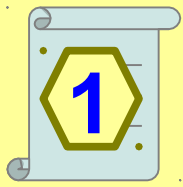
```
lect07.LabellingTextFieldDemo
```

A screenshot of a Java Swing window titled "Labelled text". The window contains two rows of labels and text fields. The first row has a label "name:" followed by an empty text field. The second row has a label "address:" followed by an empty text field. The labels "name:" and "address:" are underlined, indicating mnemonic characters. Three yellow callout boxes are present: one pointing to the "name:" label with the text "mnemonics", one pointing to the "name:" label with the text "labels", and one pointing to the text field in the second row with the text "text fields".

Label

- Class: JLabel
- Displays a single line text
- Example:

```
JLabel label = new JLabel("name:");
```



Basic display components

- Label
- Text field

Text field (1)

- Class: `JTextField`
- Captures a single line text input
- Can contain any number of characters, but only a limited number will be visible
- Number of visible characters (the visible area) can be set

Text field (2)

- Create a text field:

```
JTextField tf = new JTextField();
```

- with a fixed visible area:

```
JTextField tf = new JTextField(30);
```

- Change text:

```
tf.setText("some name");
```

- Get text:

```
String t = tf.getText();
```

Bounding a label to a text field

- A label can be bound to a text field:

```
label.setLabelFor(tf);
```

- Activate the bounded text field through the label:

```
label.setDisplayMnemonic('n');
```

*a character of
the label*

- To retrieve the bounded text field:

```
Component comp = label.getLabelFor();
```

```
JTextField tf = (JTextField) comp;
```

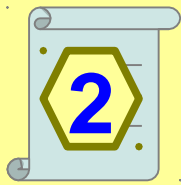
DEMO

Code

```
lect07.LabellledTextFieldDemo
```

Lecture outline

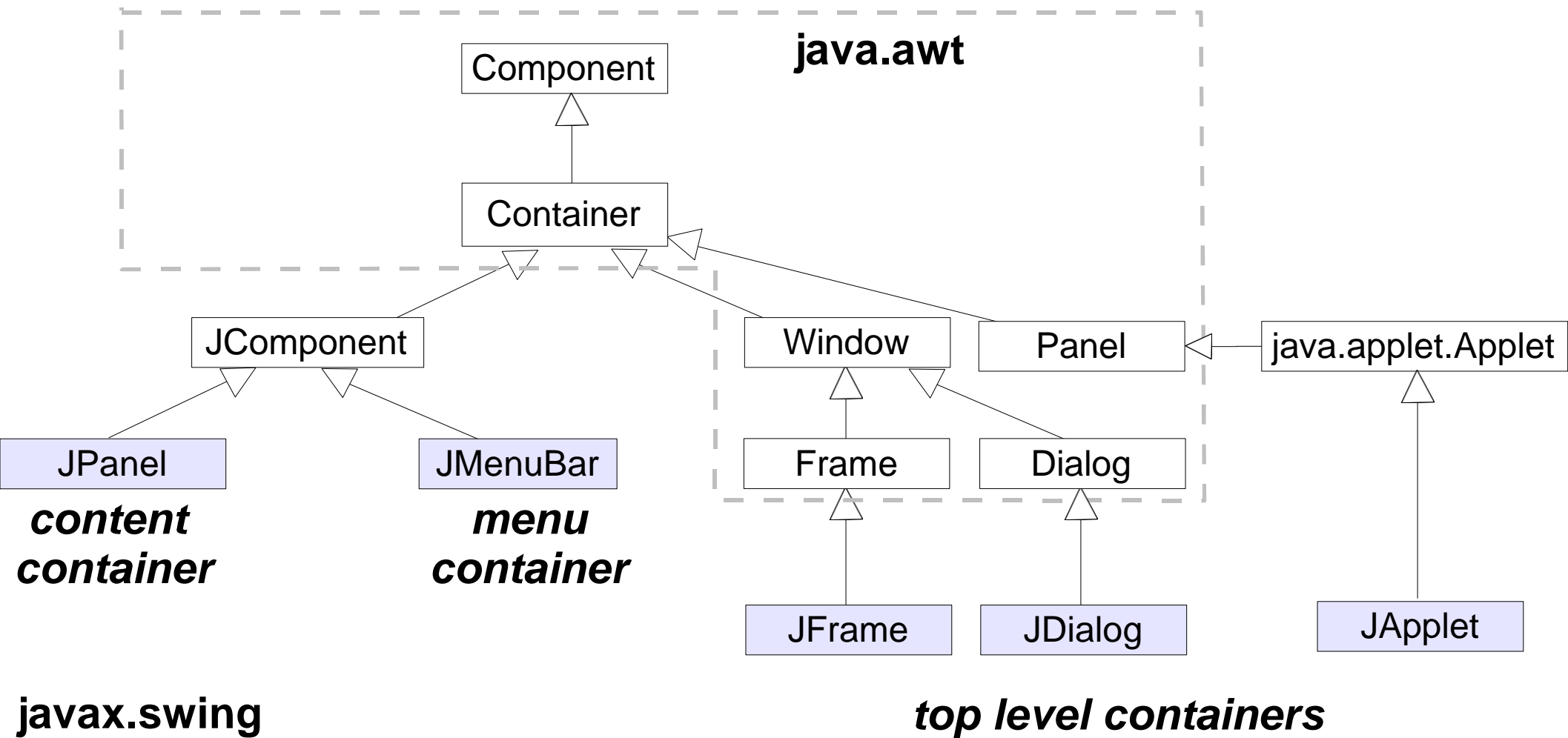
- Basic components
- **GUI containers**
- Content container
- Menu (container)
- More display components



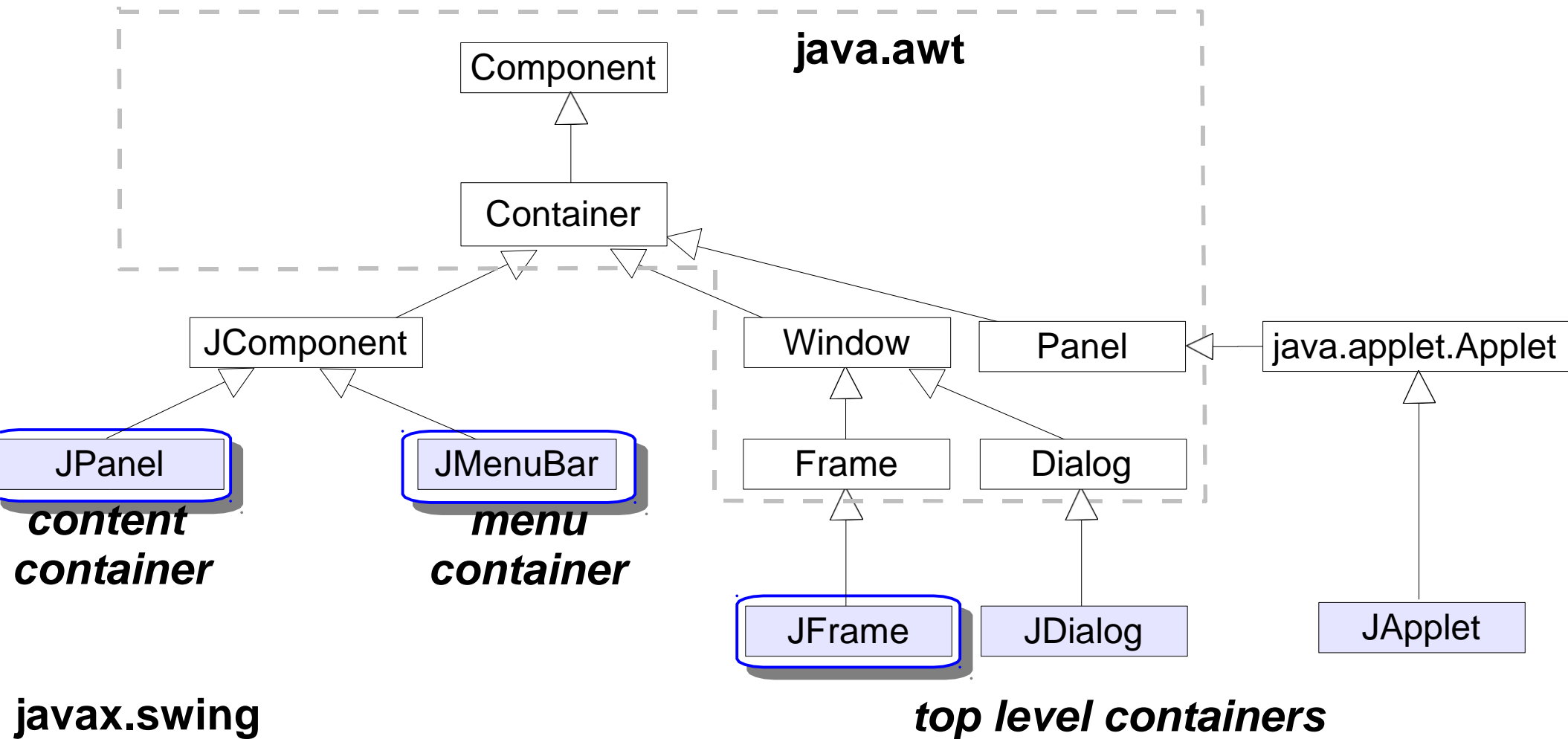
GUI containers

- All **Swing** components are sub classes of `Container`: can contain other components
- GUI components are arranged in a nested structure called ***containment hierarchy*** (CH)
- Container-specific components:
 - ***top level*** container: contains components & other containers
 - ***content*** container: contains non-menu components
 - **menu** container: contains *menu* components
- Display-specific components: label, text field, etc.

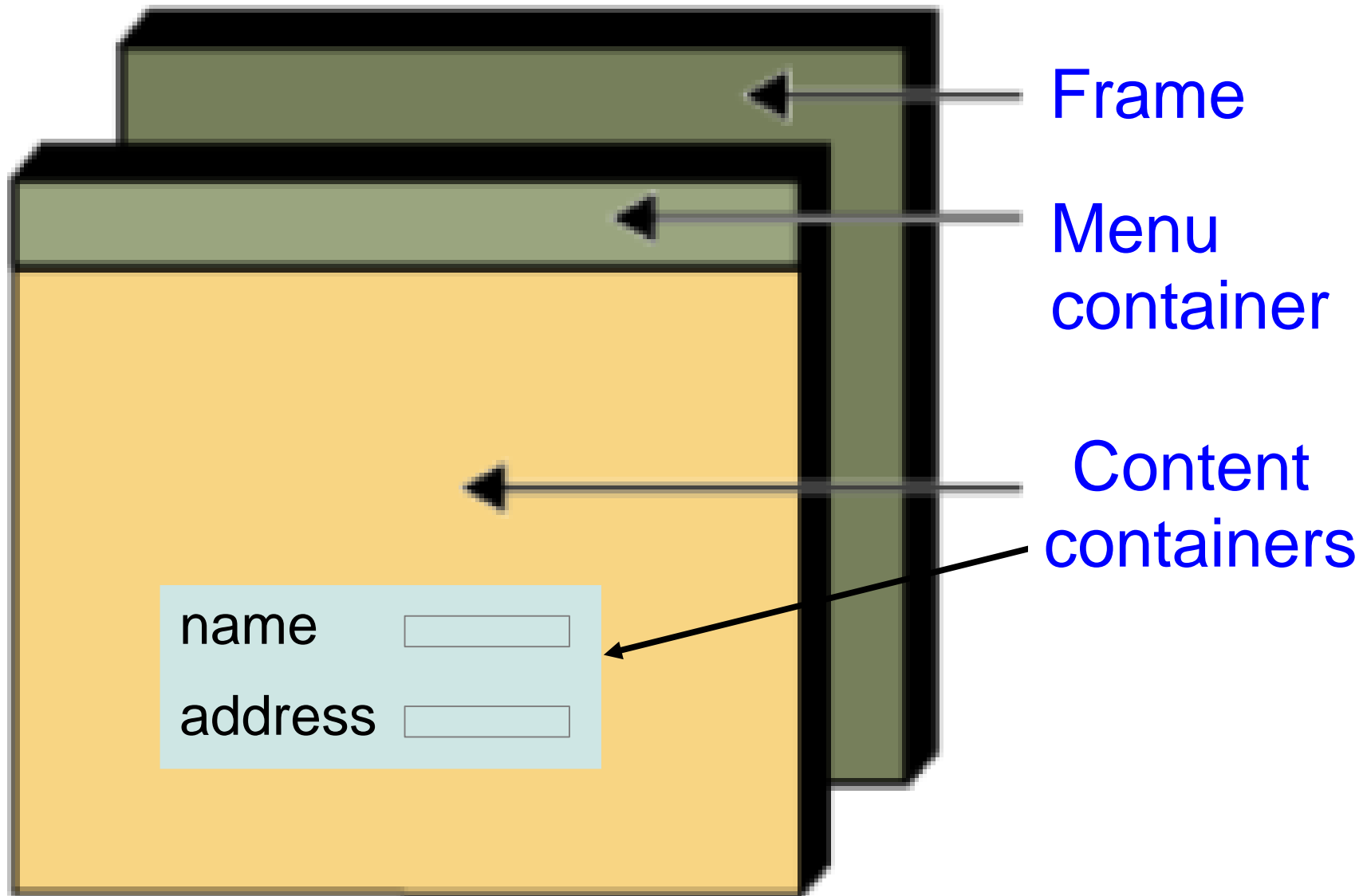
Swing CH diagram



Our application scope

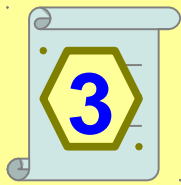


Containers



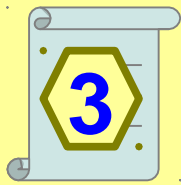
Lecture outline

- Basic components
- GUI containers
- **Content container**
- Menu (container)
- More display components



Content container

- Types of content container
- Content pane
- Panel
- Layout manager
- Working with the container's components



Content container

- Types of content container
- Content pane
- Panel
- Layout manager
- Working with the container's components

Types of content container

- The **content pane** of a window:
 - the top-level content container
- A **panel** of components:
 - to group the related display components together
- A content pane may contain one or more panels



Content container

- Types of content container
- **Content pane**
- Panel
- Layout manager
- Working with the container's components

Content pane

- Every Swing window (`JWindow` or `JFrame`) has a content pane
- All non-menu, displayable components are added to the content pane by default
- To access the content pane:

```
Container c = w.getContentPane();
```

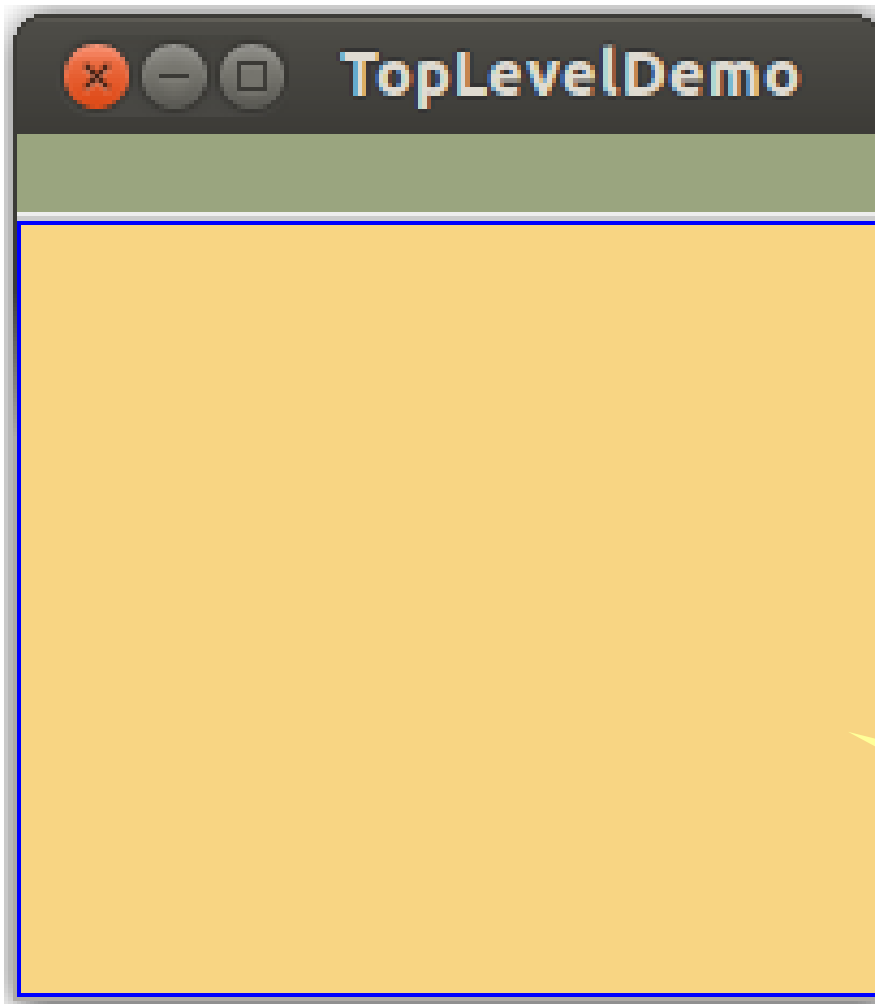
- To change a content pane:

```
Container c = new JPanel();
```

```
w.setContentPane(c);
```

Window and containers

- `lect07.TopLevelDemo`



menu bar

content pane with
a yellowish label



Content container

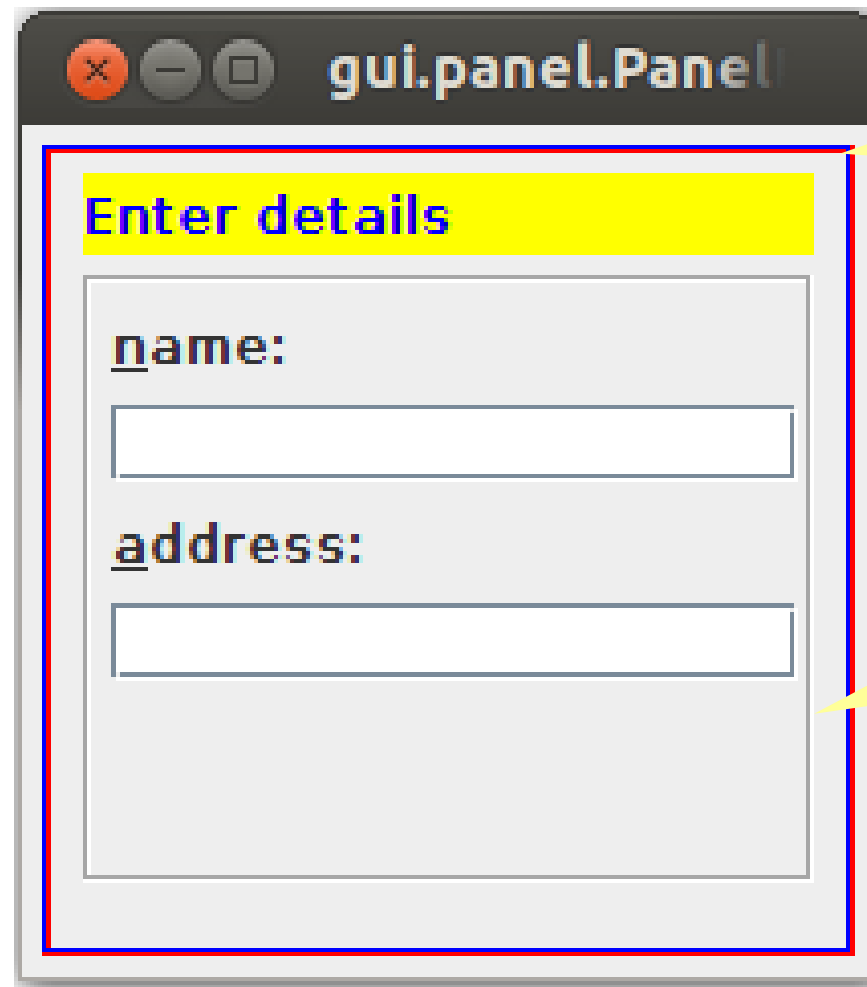
- Types of content container
- Content pane
- **Panel**
- Layout manager
- Working with the container's components

Panel

- A sub-container to organise related display components
- A panel can be nested inside another panel
- Components in a panel are arranged using *layouts*

Panel

lect07.PanelDemo



Top-level
panel

Nested (sub)
panel

Working with panels (1)

- Class: `JPanel`
- Create a panel (with a default layout):

```
JPanel p = new JPanel();
```

- Add components to a panel:

```
p.add(label);
```

```
p.add(tf);
```

- Create a nested panel:

```
JPanel subPanel = new JPanel();
```

```
p.add(subPanel);
```

Working with panels (2)

- Add a panel to a JFrame (its content pane):

```
JFrame w = new JFrame ("My GUI") ;
```

```
w.add(p) ;
```

```
// or w.getContentPane().add(p)
```



Content container

- Types of content container
- Content pane
- Panel
- **Layout manager**
- Working with the container's components

Layout manager

- Defines the *layout* of the components in a container:
 - specifies the relative positions of the components
- Implement interface `java.awt.LayoutManager` and/or `LayoutManager2`
- Pre-defined layout managers:
 - Flow layout
 - Border layout
 - Grid layout

Layout manager (2)

- Change the layout manager of a container:

```
LayoutManager lm = new FlowLayout();  
w.setLayout(lm);
```

- Obtain the layout manager of a container:

```
LayoutManager lm = w.getLayout();
```


Flow layout

- Class: `java.awt.FlowLayout`
- Default layout for panels
- Arranges components in rows:
 - direction depends on container's orientation
 - number of rows are determined by container's size
- Components alignment: left, right, center, leading, trailing
 - default: center
- Respect the components size

Flow layout

lect07.FlowLayoutDemo

The screenshot shows a Java Swing window titled "gui.layout.FlowLayoutDemo". The window contains a flow layout with the following elements:

- A label **name:** followed by a text input field.
- A label **address:** followed by a text input field.
- A button labeled **Orientation [L->R]**.

The components are arranged in a top-to-bottom flow, demonstrating how a FlowLayout organizes GUI elements.

Refresh a container's GUI

- Required when a container's layout or composition has been changed:
 - components are added/removed
 - layout information changed
- Method: `Container.validate()`

Border layout

- **Class:** `java.awt.BorderLayout`
- Default layout for windows (e.g. `JFrame`)
- Arranges components in five regions: NORTH, SOUTH, EAST, WEST, CENTER
 - defined as constants of `BorderLayout`
- Each region can only contain one component:
 - use a panel if multiple components are needed
- Spaces are allocated in the above order:
 - CENTER region fills up the remaining space

Border layout

lect07.BorderLayoutDemo

gui.layout.BorderLayoutDemo

Enter details

name:

address:

OK

NORTH

CENTER

SOUTH

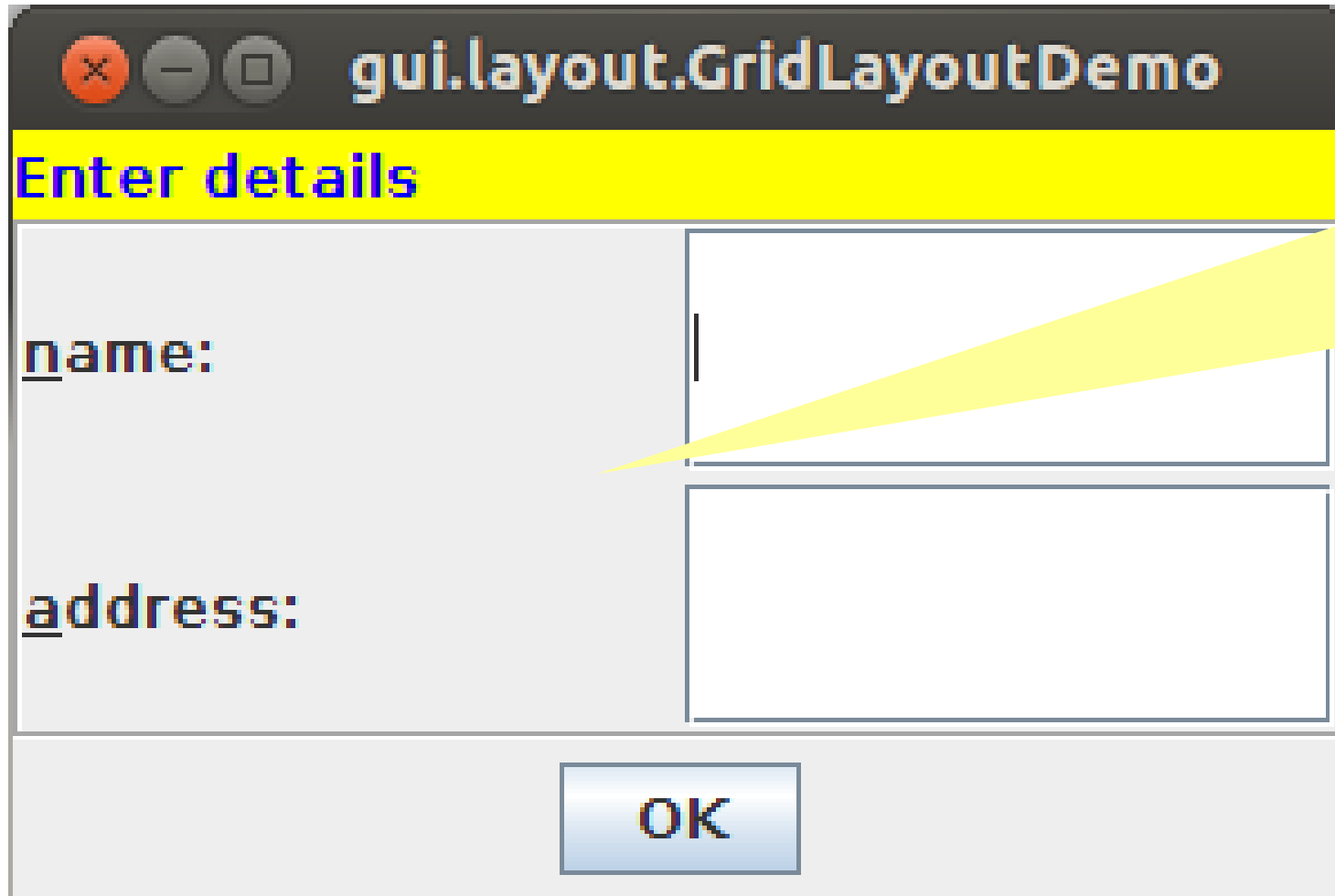
Grid layout

- **Class:** `java.awt.GridLayout`
- Arranges components in a table $N \times M$
 - N : number of rows
 - M : number of columns
 - all cells have equal size
- Components are added sequentially: from left to right, one row at a time
- Does not respect the components size:
 - components are stretched to fill up their cells

DEMO

Grid layout

lect07.GridLayoutDemo



The screenshot shows a Java Swing window titled "gui.layout.GridLayoutDemo". The window has a yellow title bar. Below the title bar, the text "Enter details" is displayed in blue. The main content area is divided into two rows by a grid layout. The first row contains the label "name:" and a text input field. The second row contains the label "address:" and another text input field. At the bottom of the window, there is an "OK" button. A yellow callout box points to the input fields with the text "Grid layout (2,2) components are stretched".

Grid layout
(2,2)
components
are stretched



Content container

- Types of content container
- Content pane
- Panel
- Layout manager
- Working with the container's components

Working with a container's components

- Get all the display components (of a container):

```
Container c = w.getContentPane();
```

```
Component[] comps = c.getComponents();
```

- Get a display component at a given position:

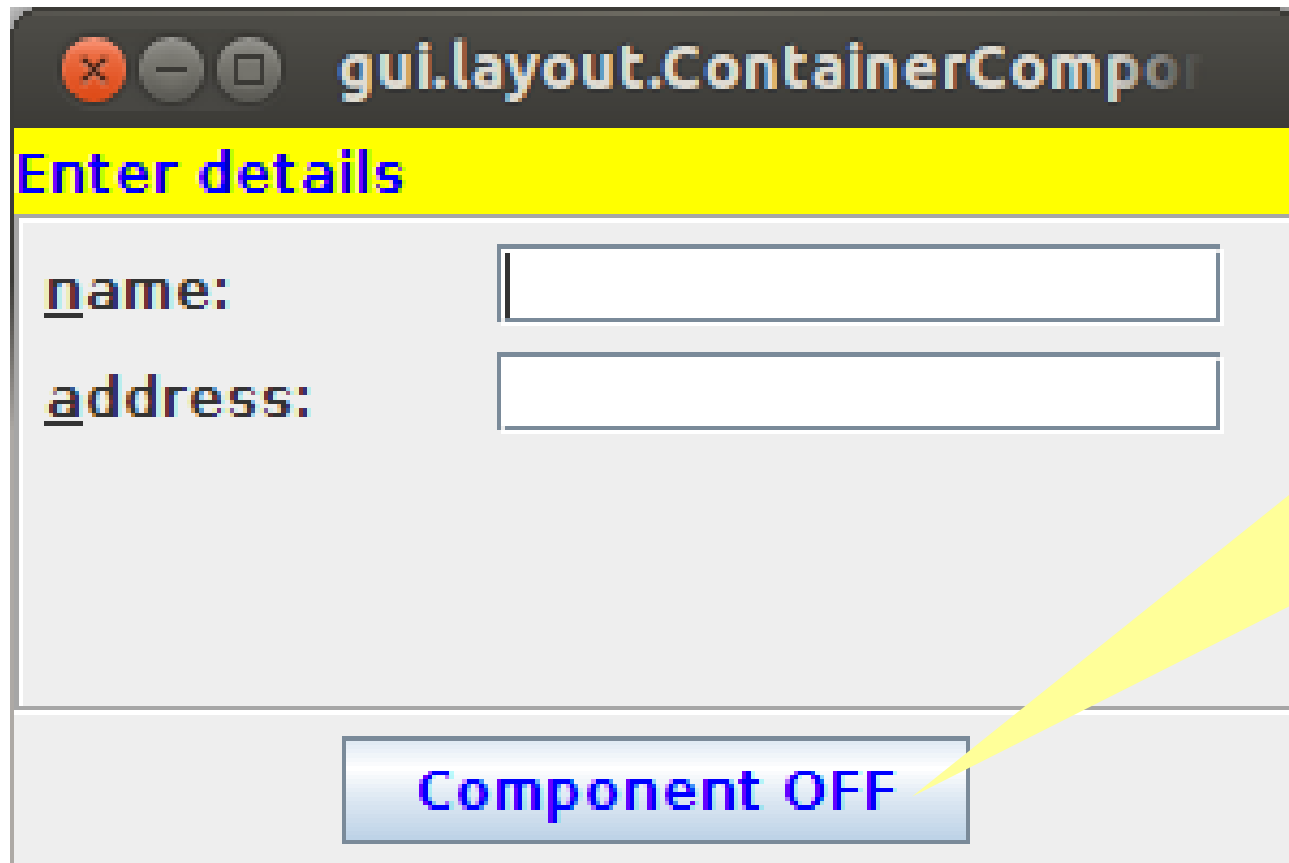
```
Component co = c.getComponent(i);
```

- Search for a component:
 - loop over the component array

DEMO

Container components

lect07.ContainerComponentDemo



gui.layout.ContainerComponentDemo

Enter details

name:

address:

Component OFF

To turn ON/OFF
each component
of the
middle panel

Lecture outline

- Basic components
- GUI containers
- Content container
- **Menu (container)**
- More display components



Menu (container)

- Menu bar, menu, menu item
- Menu component hierarchy
- Working with menus



Menu (container)

- Menu bar, menu, menu item
- Menu component hierarchy
- Working with menus

Menu bar, menu, menu item

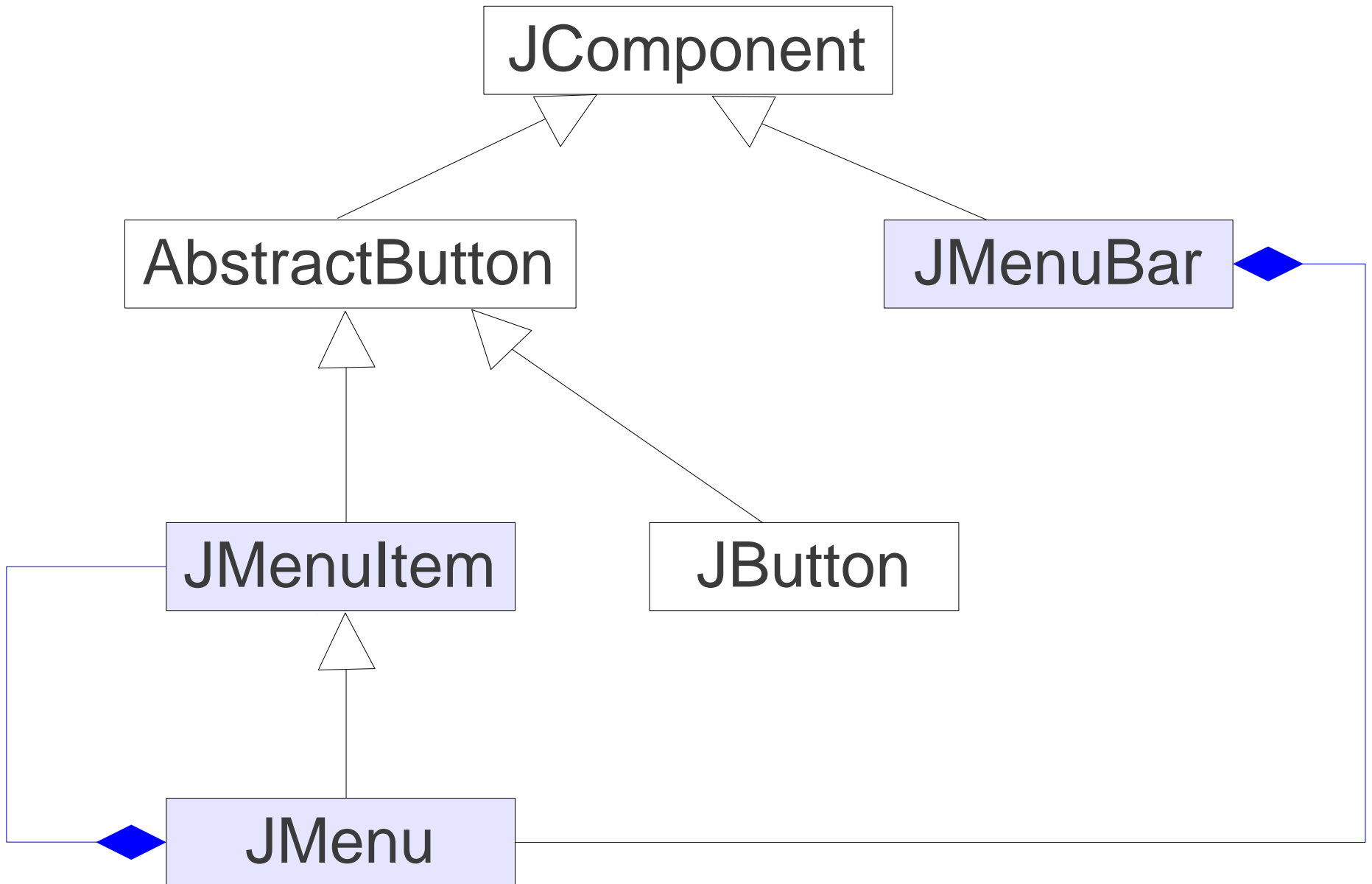
- Every `JFrame` has a menu bar displayed above the content pane
- A **menu bar** contains zero or more menus
- A **menu** contains zero or more menu items
- A **menu item** is a user action (same as that on a button)
- Swing classes:
 - `JMenuBar`
 - `JMenu`
 - `JMenuItem`



Menu (container)

- Menu bar, menu, menu item
- **Menu component hierarchy**
- Working with menus

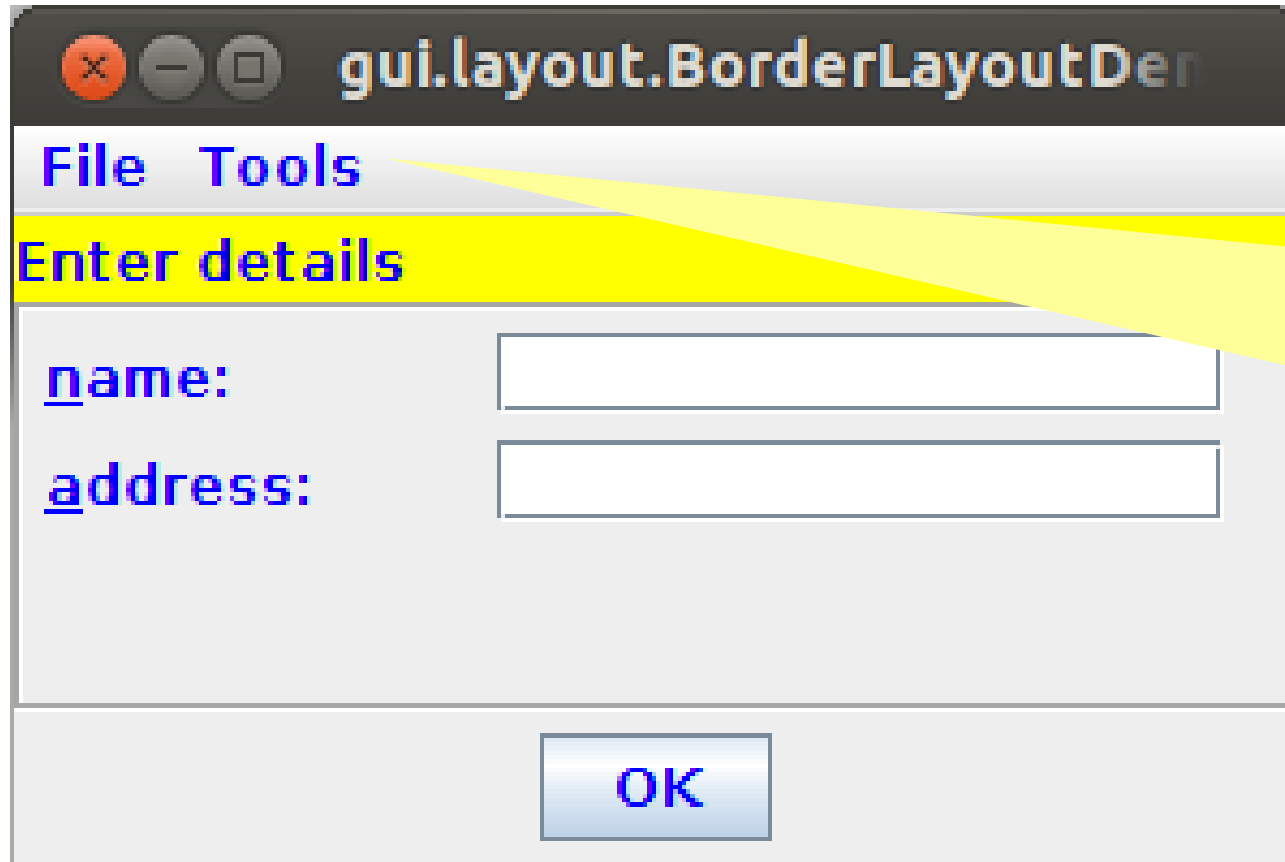
Menu component/container hierarchy



DEMO

Menu

lect07.MenuDemo



gui.layout.BorderLayoutDemo

File Tools

Enter details

name:

address:

OK

Menu bar/
Options:
to change
text colours



Menu (container)

- Menu bar, menu, menu item
- Menu component hierarchy
- Working with menus

Working with menus (1)

- Create a menu bar:

```
JMenuBar menuBar = new JMenuBar();
```

- Create one or more menus:

```
JMenu fileMenu = new JMenu("File");
```

- Create one or more menu items under a menu:

```
JMenuItem exit = new JMenuItem("Exit");
```

- Set up action handler for menu items:

```
exit.addActionListener(...);
```

Working with menus (2)

- Add menu items to a menu:

```
fileMenu.add(exit);
```

- Add menus to a menu bar:

```
menuBar.add(fileMenu);
```

```
menuBar.add(toolsMenu);
```

- Set menu bar on the JFrame:

```
w.setJMenuBar(menuBar);
```

Create a nested menu

- Add a menu to another menu:

```
JMenu saveMenu = new JMenu("Save");  
JMenuItem toFile =  
    new JMenuItem("to file");  
JMenuItem toDB =  
    new JMenuItem("to database");  
saveMenu.add(toFile);  
saveMenu.add(toDB);  
fileMenu.add(saveMenu);
```

Obtain menu components

- **Use** `getMenuComponents()`:

```
Component[] menuItems =
```

```
    fileMenu.getMenuComponents();
```

Lecture outline

- Basic components
- GUI containers
- Content container
- Menu (container)
- **More display components**

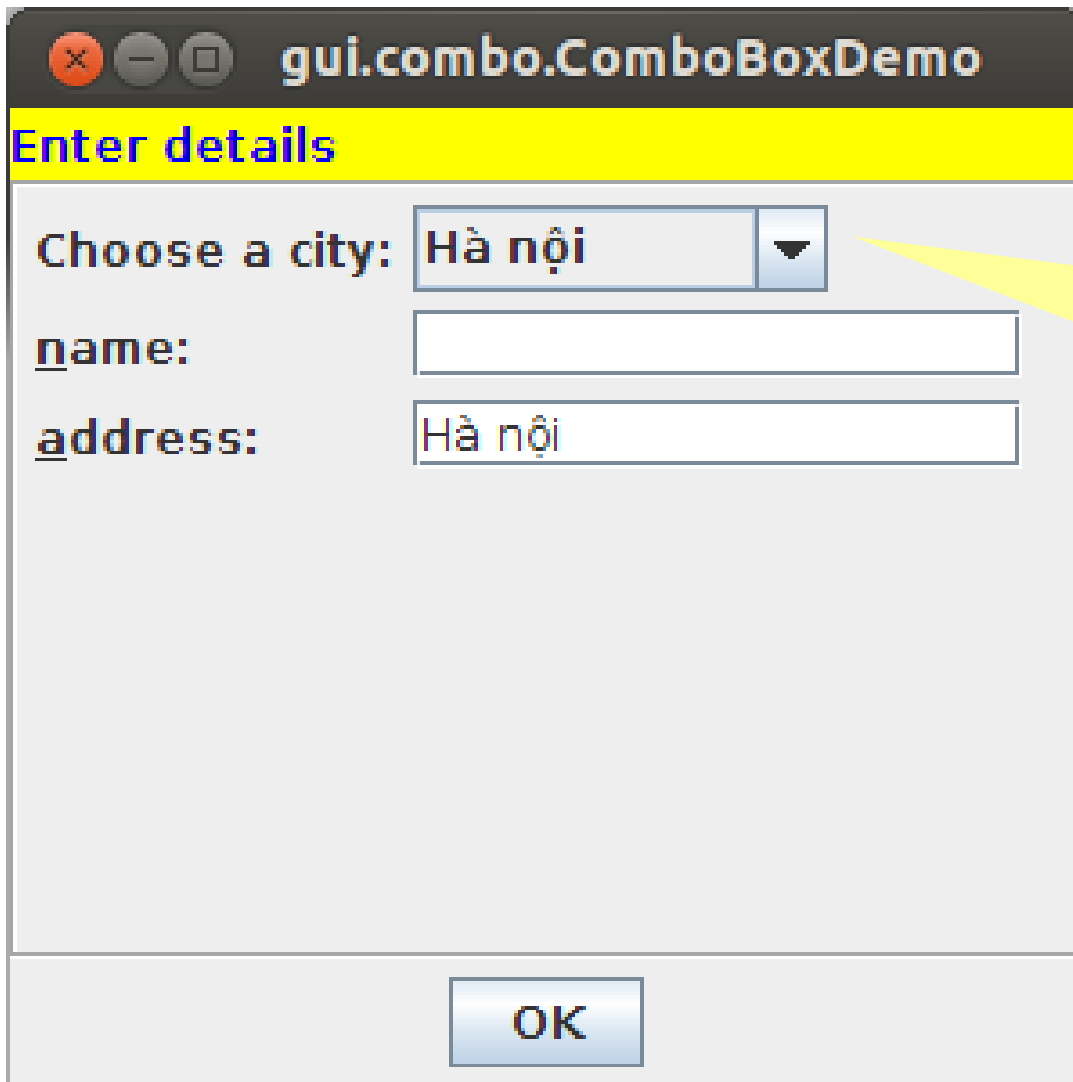


More display components

- Combo box

Combo box

lect07.ComboBoxDemo



The screenshot shows a Java Swing window titled "gui.combo.ComboBoxDemo". The window has a yellow title bar and a light gray content area. At the top, there is a yellow header bar with the text "Enter details" in blue. Below the header, there is a label "Choose a city:" followed by a combo box containing the text "Hà nội". To the right of the combo box is a small downward arrow. Below the combo box, there are two text fields. The first text field is labeled "name:" and is empty. The second text field is labeled "address:" and contains the text "Hà nội". At the bottom of the window, there is an "OK" button.

Choose a city
to update
address field

Combo box (1)

- Class: `JComboBox`
- Displays a drop-down list of objects:
 - typically strings
- Read only (by default), but can be made editable

Combo box (2)

- Create a combo box:

```
String[] strings = { "", "Hà nội" };  
JComboBox combo = new JComboBox(strings);
```

- Get the selected object:

```
String s = (String) combo.getSelectedItem();
```

- Change the selected object:

```
combo.setSelectedIndex(1);  
combo.setSelectedItem("Hà nội");
```

Handle combo box events

- Create event handler for combo box
- Create an event handler object
- Register event handler object to the combo box
 - invoke an `addXListener()` method

Create event handler for combo box

- **Event:** `ActionEvent` **or** `ItemEvent`
- **Listener:** `ActionListener` **or** `ItemListener`
- **Create a handler as `ActionListener`:**
 - implements `ActionListener`
 - in `actionPerformed()`:
 - get combo box object from source
 - invoke `getSelectedItem` to retrieve current item
 - process item

Additional components & issues

- Technical guide for Swing components:
 - Java tutorial > Creating a GUI With JFC/Swing > Using Swing Components
- Handle text field events:
 - Java tutorial > ... > How to Use Text Fields

Summary

- Display components are organised on a GUI using containers
- All containers are sub-class of Container class
- All Swing components are containers
- Layout managers are used to layout components in a container
- Top level containers are JFrame, JDialog and JApplet
- JFrame uses a content pane to organise components and an (optional) menu bar

References

Savitch W., Absolute Java, 6th, Pearson, 2015

- Chapter 17

Oracle, The Java Tutorial, Oracle,
<http://docs.oracle.com/javase/tutorial>

-Lesson: Creating a GUI With JFC/Swing, Using
Swing Components