

LẬP TRÌNH JAVA

Đỗ Ngọc Như Loan - Nguyễn Thị Hồng Anh

Chương 3. Lập trình giao diện đồ họa

Nội dung



- Frame, Panel
- Label, Textfield, Button
- Các đối tượng đồ họa thường dùng khác

Chương 3. Lập trình giao diện đồ họa

Khái niệm



- Lập trình giao diện GUI (Graphic User Interface) là việc sử dụng các đối tượng trong Java để thiết kế thành các giao diện trực quan giúp người dùng có thể tương tác để thực hiện các công việc.
- Trong một bài toán, có thể có 1 hoặc nhiều giao diện người dùng.

Chương 3. Lập trình giao diện đồ họa

Các thư viện lập trình giao diện

- AWT
- SWING

Chương 3. Lập trình giao diện đồ họa

Thư viện lập trình AWT

(Abstract Window Toolkit)



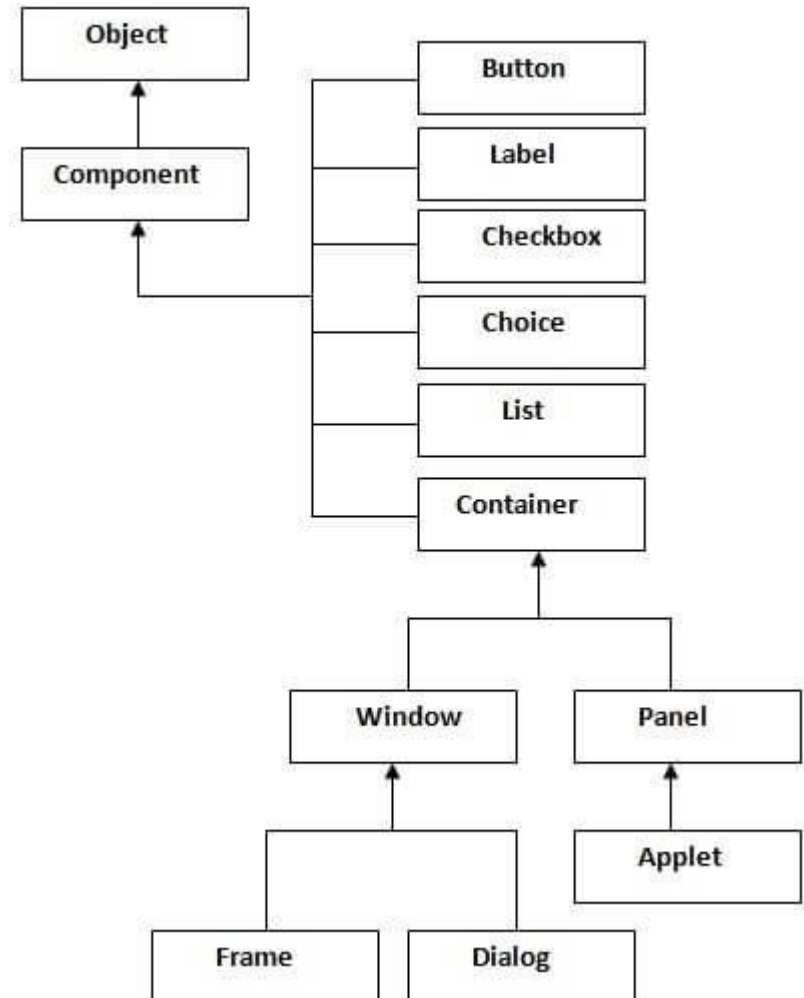
- AWT là bộ thư viện được Java xây dựng từ phiên bản JDK 1.0 để hỗ trợ thiết kế các giao diện và đồ họa người dùng.
- Đây được xem là bộ thư viện cồng kềnh, gặp khó khăn và giao diện không ổn định trên các hệ điều hành khác nhau

Chương 3. Lập trình giao diện đồ họa

AWT



- Kiến trúc một số đối tượng AWT



Chương 3. Lập trình giao diện đồ họa

Thư viện lập trình Swing



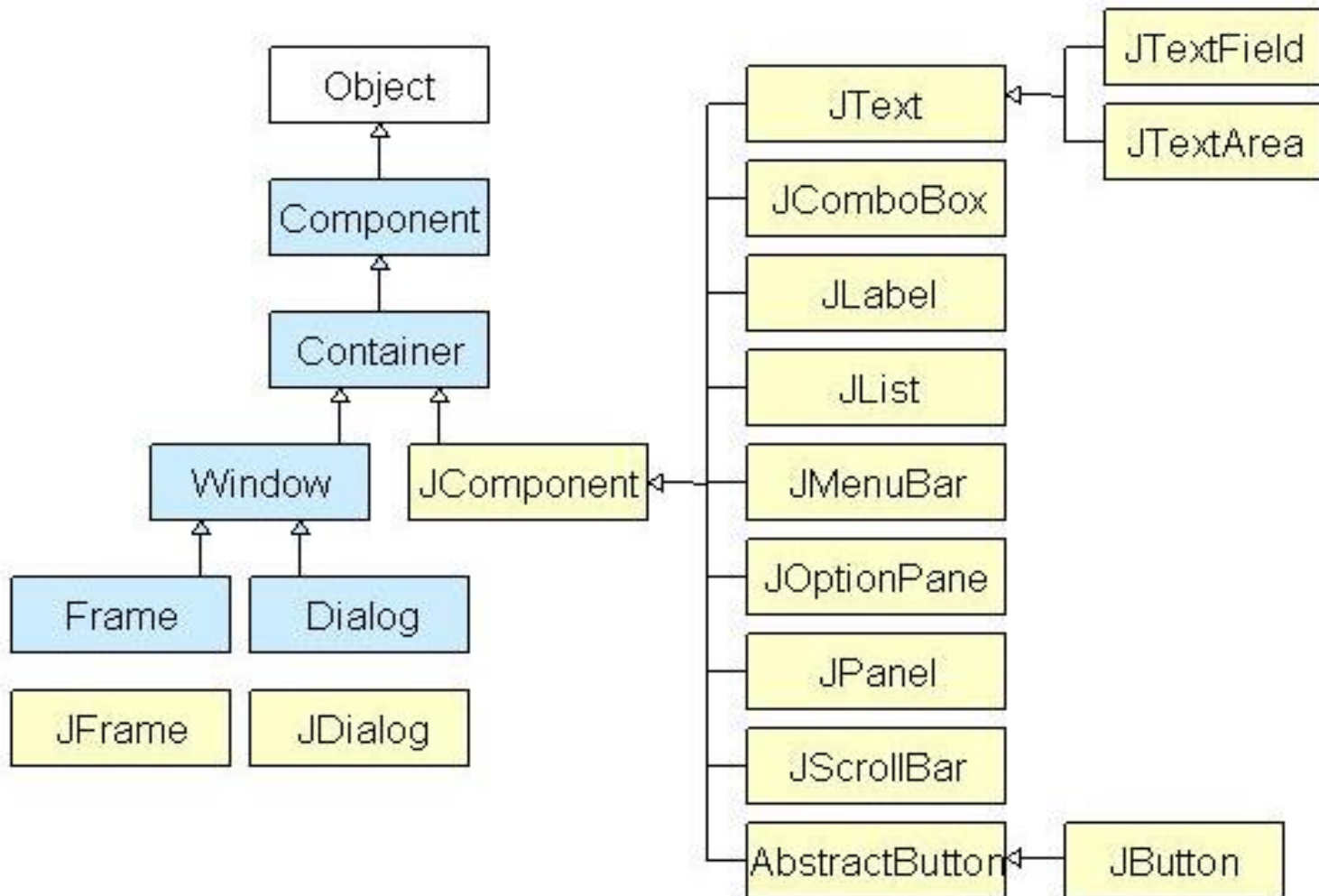
- Swing được xây dựng và tích hợp vào phiên bản JDK 1.1.2
- Được đánh giá là bộ thư viện tối ưu và thuận tiện cho việc thiết kế các giao diện người dùng, thân thiện và phù hợp cho nhiều hệ điều hành khác nhau.
- Java Swing là một phần của Java Foundation Classes (JFC) được sử dụng để tạo các ứng dụng Window-Based. Nó được xây dựng ở trên cùng của AWT API và được viết hoàn toàn bằng Java.

Chương 3. Lập trình giao diện đồ họa

Swing



Kiến trúc một số đối tượng Swing



Chương 3. Lập trình giao diện đồ họa

Điểm khác nhau giữa AWT và SWING



Java AWT	Java SWING
Các thành phần AWT là phụ thuộc nền tảng	Các thành phần Java Swing là độc lập nền tảng
Các thành phần AWT là nặng	Các thành phần Swing là gọn nhẹ
AWT không hỗ trợ pluggable look & feel	Swing hỗ trợ pluggable look & feel
AWT cung cấp ít thành phần hơn Swing	Swing cung cấp các thành phần mạnh mẽ hơn như table, list, scrollpanes, colorchooser, tabbedpane ...
AWT không theo sau MVC (Model View Controller), ở đây model biểu diễn dữ liệu, view biểu diễn sự trình bày và controller hoạt động như một Interface giữa model và view	Swing theo sau MVC

Chương 3. Lập trình giao diện đồ họa

Các thành phần GUI



- Các phần tử UI: : Đó là các phần tử nhìn thấy chủ yếu mà người dùng cuối cùng nhìn thấy và tương tác với. Swing cung cấp rất nhiều các phần tử đa dạng từ cơ bản tới nâng cao.
- Layout: Chúng định nghĩa cách các phần tử UI nên được tổ chức trên màn hình và cung cấp đối tượng L&F (là viết tắt của Look and Feel) cuối cùng tới GUI (Graphical User Interface).
- Hành vi: Đó là các sự kiện xảy ra khi người dùng tương tác với các phần tử UI.

Chương 3. Lập trình giao diện đồ họa

Các phần tử Swing UI



- **Lớp JLabel** Một đối tượng JLabel là một thành phần để đặt text vào trong một Container
- **Lớp JButton** Lớp này tạo một button đã được gán nhãn
- **Lớp JTable** Lớp JTable được sử dụng để hiển thị dữ liệu trên các ô của bảng hai chiều
- **Lớp Graphics** Lớp này cung cấp nhiều phương thức để lập trình đồ họa
- **Lớp JColorChooser** Một JColorChooser cung cấp một pane gồm các control được thiết kế để cho phép một người dùng thao tác và lựa chọn màu

Chương 3. Lập trình giao diện đồ họa

Các phần tử Swing UI



- **Lớp JCheckBox** Một JCheckBox là một thành phần đồ họa mà có thể trong trạng thái on (true) hoặc off (false)
- **Lớp JRadioButton** Lớp JRadioButton là một thành phần đồ họa mà có thể trong trạng thái on (true) hoặc off (false) trong một nhóm
- **Lớp JList** Một thành phần JList biểu diễn cho người dùng một danh sách các item
- **Lớp JComboBox** Một thành phần JComboBox biểu diễn cho người dùng một menu các lựa chọn

Chương 3. Lập trình giao diện đồ họa

Các phần tử Swing UI



- **JTextField** Một đối tượng JTextField là một thành phần text cho phép chỉnh sửa một dòng text đơn
- **Lớp JTextArea** Một đối tượng JTextArea là một thành phần text cho phép sửa đổi một text có nhiều dòng
- **Lớp ImageIcon** Một ImageIcon control là một trình triển khai của Icon Interface mà tô màu các Icon từ Image
- **Lớp JScrollbar** Một Scrollbar control biểu diễn một thành phần scroll bar để cho người dùng khả năng lựa chọn từ trong một dãy các giá trị



Chương 3. Lập trình giao diện đồ họa

Các phần tử Swing UI



- **Lớp JOptionPane** JOptionPane cung cấp tập hợp các dialog box chuẩn mà gợi ý người dùng về một giá trị hoặc thông báo cho họ một cái gì đó
- **JFileChooser** Một JFileChooser control biểu diễn một dialog window từ đó người dùng có thể lựa chọn một file
- **Lớp JProgressBar** Thanh tiến trình hiển thị phần trăm hoàn thành tác vụ đang diễn ra
- **Lớp JSlider** Một JSlider cho phép người dùng lựa chọn một giá trị từ một dãy cụ thể
- **Lớp JSpinner** Một JSpinner là một trường input dòng đơn, cho phép người dùng lựa chọn một số hoặc một giá trị đối tượng từ dãy đã qua sắp xếp

Chương 3. Lập trình giao diện đồ họa

Các đối tượng Container trong Java



- **Container** là vùng để đặt các thành phần giao diện vào đó. Bất cứ vật gì mà kế thừa từ lớp Container sẽ là vật chứa.
- Một vật chứa có thể chứa nhiều phần tử, các phần tử này có thể được vẽ hay được tô màu tùy thích. Bạn hãy xem vật chứa như một cửa sổ.
- Một số container thường gặp:
 - Frames
 - Panels
 - Dialogs
 - ...

Chương 3. Lập trình giao diện đồ họa

Một số phương thức thường dùng của Java Swing trong Component class



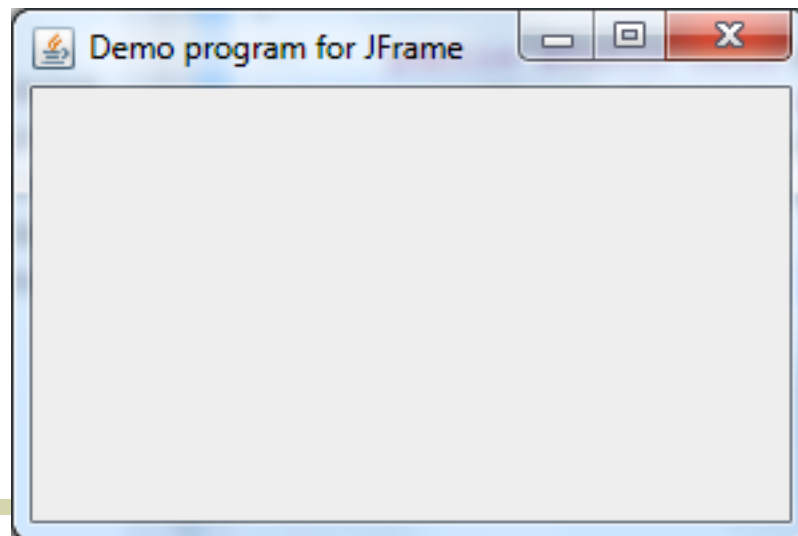
- `public void add(Component c)` : thêm 1 component trên 1 component khác
- `public void setSize(int width, int height)`: thiết lập kích thước Component
 - Ví dụ: `setSize(300,100);`
- `public void setVisible(Boolean b)`: thiết lập thuộc tính ẩn hay hiện cho Component (giá trị ngầm định là false).
- `public void setLayout(layoutManager m)`: thiết lập layout chính cho **Container**

Chương 3. Lập trình giao diện đồ họa

JFrame



- JFrame trong gói Swing kế thừa từ Frame của AWT. JFrame giống như cửa sổ chính có thể chứa trong nó các thành phần để tạo Gui như: labels, buttons, textfields, ...
- Ưu điểm của JFrame hơn Frame: nó có thêm tùy chọn hide và close cửa sổ thông qua phương thức *setDefaultCloseOperation(int)*.



Chương 3. Lập trình giao diện đồ họa



- Code tạo JFrame có thể được viết trong các constructor.
 - JFrame(): Tạo một Frame mới, ban đầu là không nhìn thấy (invisible)
 - JFrame(String title): Tạo một Frame mới, ban đầu là không nhìn thấy (invisible) với title đã cho.
- Chúng ta cũng có thể kế thừa lớp JFrame, nên không cần tạo thể hiện của các JFrame

Chương 3. Lập trình giao diện đồ họa

JFrame – Một số phương thức thường dùng



- setTitle(String Title): định nghĩa tiêu đề cho khung giao diện
 - Ví dụ: setTitle("My first program");
- setSize(int width, int height): định nghĩa kích thước chiều rộng và chiều cao của khung giao diện.
 - Ví dụ: setSize(300,100);
- setBackground(color c); định nghĩa màu sắc cho nền của JFrame
 - Ví dụ: getContentPane().setBackground(Color.RED);

Chương 3. Lập trình giao diện đồ họa

JFrame – Một số phương thức thường dùng



- `setLocation(int x, int y)`: định nghĩa vị trí hiển thị của khung giao diện trong màn hình.
 - Ví dụ: `setLocation(20,20);`
- `setBounds(int x, int y, int width, int height)`: định nghĩa vị trí và kích thước cho khung giao diện.
 - Ví dụ: `setBounds(20,20,300,100);`
- `setResizable(Boolean value)`; `value=true` cho phép người dùng có thể kéo dãn kích thước khung giao diện, ngược lại thì kích thước khung giao diện bị cố định.
 - Ví dụ: `setResizable(true);`

Chương 3. Lập trình giao diện đồ họa

JFrame – Một số phương thức thường dùng



- `setDefaultCloseOperation(int mode)`; định nghĩa hành động của khung giao diện khi người dùng click vào dấu X ở góc phải trên cùng.
 - `DISPOSE_ON_CLOSE`: đóng giao diện đang thao tác mà không ảnh hưởng tới các giao diện hiển thị khác. Nếu chương trình chỉ có 1 giao diện thì chương trình sẽ tự động kết thúc khi giao diện được đóng.
 - `HIDE_ON_CLOSE`: ẩn giao diện đang sử dụng xuống, giống như khi ta click vào – thu nhỏ màn hình. Nó không làm tắt giao diện và các trạng thái trên giao diện vẫn giữ nguyên. (**chế độ mặc định**)

Chương 3. Lập trình giao diện đồ họa



- **DO_NOTHING_ON_CLOSE**: không thực hiện hành động gì. Trong trường hợp này, thông thường chúng ta sẽ xử lý bắt thao tác của người dung và tự định nghĩa hành vi cho khung giao diện.
- **EXIT_ON_CLOSE**: đóng giao diện đang thao tác, đồng thời tắt luôn chương trình. Thường được sử dụng cho giao diện chính của chương trình.
- Ví dụ:
`setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);`

Chương 3. Lập trình giao diện đồ họa



- `setLayout(LayoutManager layout);` định nghĩa việc hiển thị các đối tượng trong khung giao diện. Layout mặc định của Frame là `BorderLayout`.
 - `null`: cho phép các đối tượng trong khung giao diện được hiển thị ở các vị trí tùy ý, phụ thuộc vào việc định nghĩa `x`, `y`, `width`, `height` của các đối tượng.
 - Ví dụ: `setLayout(null);`
`setLayout(new CardLayout());`

Chương 3. Lập trình giao diện đồ họa

JFrame – Một số phương thức thường dùng



- `setVisible(Boolean true)`: định nghĩa việc ẩn/hiện của khung giao diện. Nếu `true`: khung giao diện hiển thị ngược lại thì bị ẩn.
 - Ví dụ: `setVisible(true)`;
- `setLocationRelativeTo(Component comp)`: định nghĩa việc hiển thị của một đối tượng khung giao diện trong màn hình chứa nó, giúp khung giao diện luôn hiển thị chính giữa.
 - Ví dụ: `setLocationRelativeTo(null)`;
- `pack()`: hiển thị Frame ôm sát nội dung trên form

Chương 3. Lập trình giao diện đồ họa

JFrame – Một số phương thức thường dùng



- `setIconImage(Image icon);` định nghĩa icon của khung giao diện.
 - Ví dụ:
 - Khởi tạo đối tượng Image:
`Image icon=new ImageIcon (Ten_doi_tuong.class.getResource(String path).getImage());`
 - Set Icon cho khung giao diện:
`setIconImage(Image icon);`

Chương 3. Lập trình giao diện đồ họa

JFrame – Một số phương thức thường dùng



- `addWindowListener(WindowListener lis);` là bộ sự kiện giúp khung giao diện lắng nghe được khi trạng thái của nó bị thay đổi như: bị đóng lại, bị mở ra, bị ẩn, ...
- `addMouseListener(MouseListener m);` là bộ sự kiện giúp khung giao diện lắng nghe khi người dùng sử dụng chuột để tương tác: click, press, move, di chuột vào khung giao diện, ...
- `addKeyListener(KeyListener k);` là bộ sự kiện giúp khung giao diện người dùng lắng nghe được khi người dùng nhấn các phím cứng trên bàn phím để tương tác với nó.

Chương 3. Lập trình giao diện đồ họa

JFrame – Một số phương thức thường dùng



- **WindowEvent:** là đối tượng chứa tất cả các thông số cần thiết của khung giao diện tương ứng với trạng thái lắng nghe được. Có thể sử dụng đối tượng này để lấy các dữ liệu liên quan của khung giao diện tại thời điểm lắng nghe được.
- **MouseEvent:** là đối tượng chứa các thông số cần thiết về chuột như (x, y, ...) tương ứng với trạng thái khung giao diện đang lắng nghe được.
- **KeyEvent:** đối tượng chứa các thông tin về phím mà người dùng vừa thao tác trong các trạng thái lắng nghe được, chúng ta có thể thông qua đối tượng này để lấy các thông tin như mã, nội dung phím vừa nhấn là gì.
- **add(Component com);** phương thức dùng để add một đối tượng giao diện vào khung chứa.

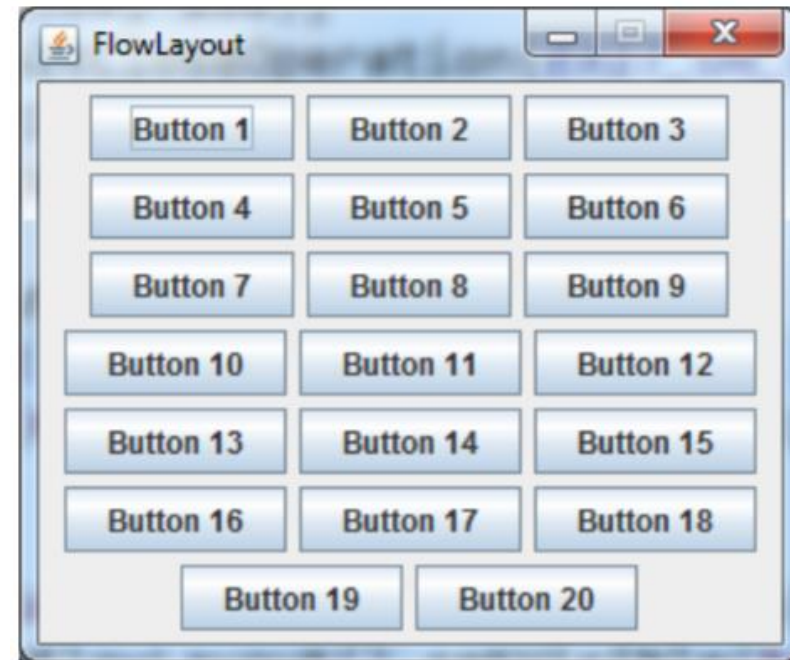
Chương 3. Lập trình giao diện đồ họa

Layout



FlowLayout

- Đây là loại layout đơn giản nhất, layout này thường được dùng kết hợp với các layout khác.
- Layout này quy định kích thước của các component con là vừa đủ với nội dung hiển thị của component.
- Mặc định các component sẽ được sắp xếp trên một hàng từ trái sang phải, nếu không vừa đủ một hàng thì chúng sẽ xuống hàng.



Chương 3. Lập trình giao diện đồ họa

Layout



GridLayout

- Layout này sắp xếp các component theo dạng bảng.
- Các component sẽ có kích thước bằng nhau.
- `GridLayout(int rows, int cols)`
- `GridLayout(int rows, int cols, int hgap, int vgap)`



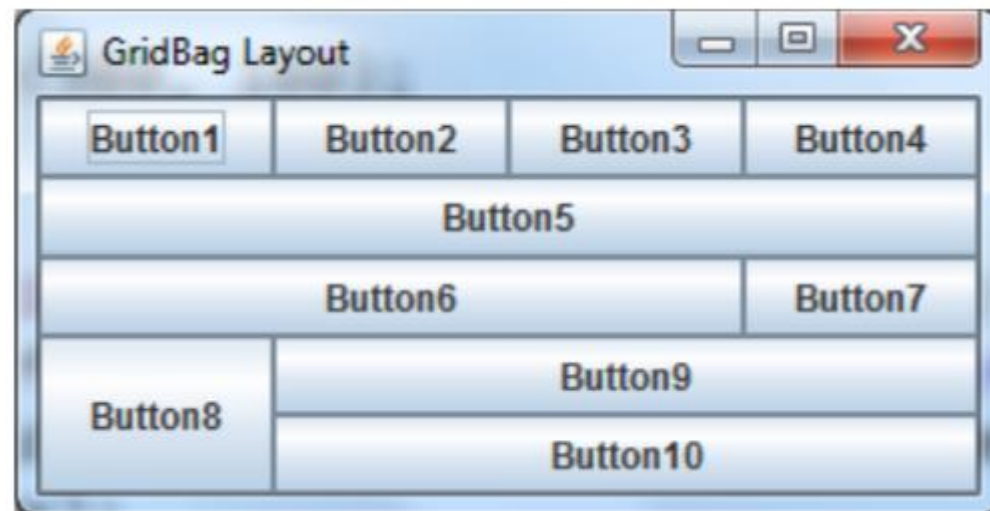
Chương 3. Lập trình giao diện đồ họa

Layout



GridBagLayout

- Đây là một layout có cơ chế sắp xếp giống như GridLayout, các phần tử xếp thành một lưới hình chữ nhật. Tuy nhiên nó linh động hơn GridLayout cho phép chúng ta chỉ định vị trí, kích thước của các phần tử trong lưới hình chữ nhật.
- Mỗi thành phần con trong GridBagLayout đều gắn liền với một GridBagConstraints object chỉ định các thuộc tính như tọa độ dòng và cột, độ rộng của một thành phần con.



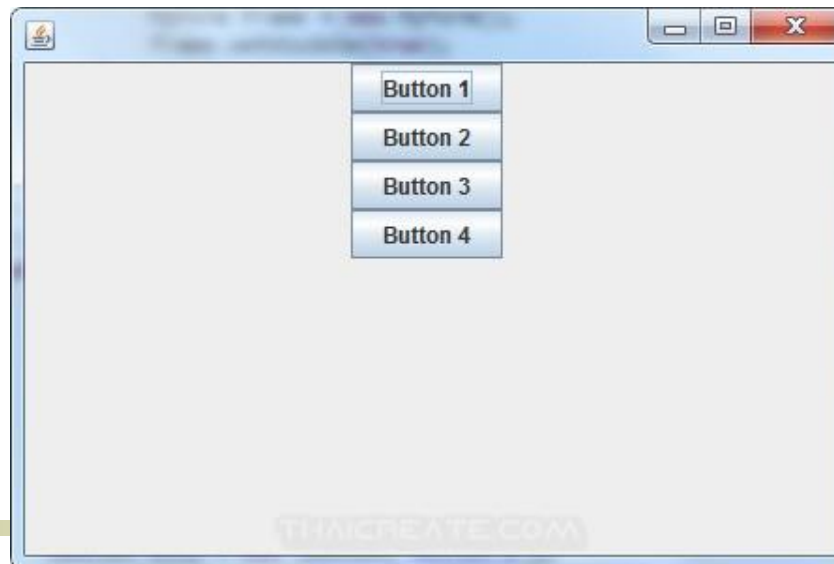
Chương 3. Lập trình giao diện đồ họa

Layout



BoxLayout

- BoxLayout cho phép tạo các giao diện phức tạp.
- Sắp xếp các component theo chiều ngang hoặc chiều dọc, khác với FlowLayout, các control sẽ không tự động xuống dòng nếu như không đủ chỗ cho chúng.
- Có thể lồng một BoxLayout vào một BoxLayout khác.

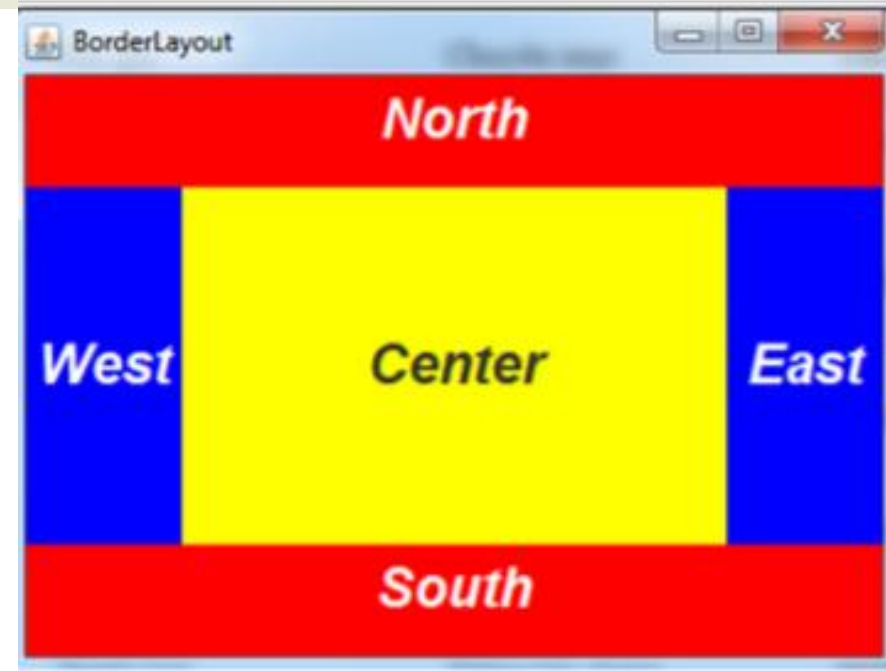


Chương 3. Lập trình giao diện đồ họa



BorderLayout

- Sắp xếp các component theo vùng, ở đây có 5 vùng là Đông (EAST), Tây (WEST), Nam (SOUTH), Bắc (NORTH), và Chính giữa (CENTER).
- Mỗi vùng chỉ được chứa một component, muốn đưa nhiều component vào một vùng thì đặt một layout khác vào vùng đó rồi đặt các component vào layout mới đó.
- Kích thước của các component trong mỗi vùng là do chúng ta thiết lập chứ không phải do layout tự co giãn, ngoại trừ vùng CENTER sẽ có kích thước thay đổi tùy thuộc vào 4 vùng còn lại.



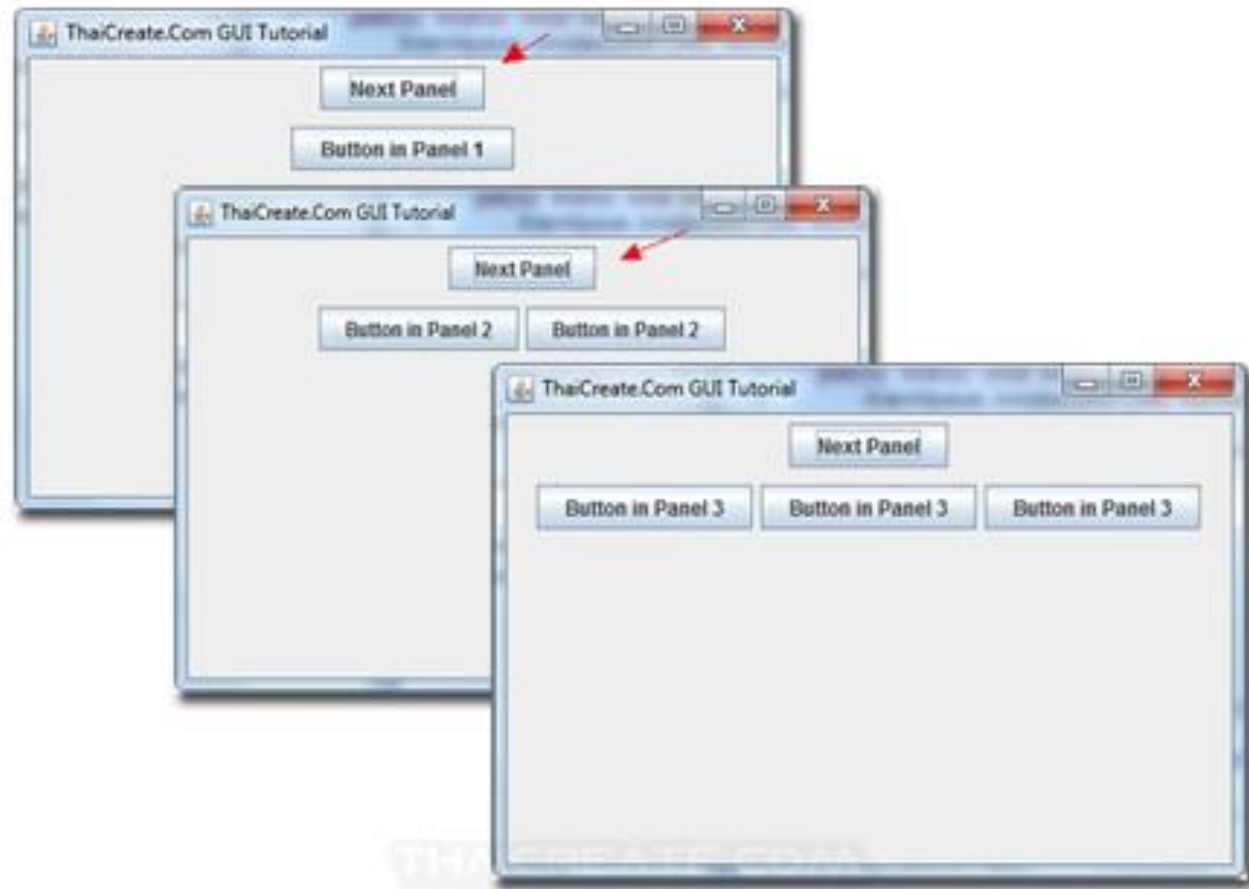
VD: Thêm button b vào frame f tại vị trí
WEST `f.add(b, BorderLayout.WEST);`

Chương 3. Lập trình giao diện đồ họa



CardLayout

Cho phép các đối tượng giao diện được hiển thị thành các tầng layer, mỗi đối tượng là một tầng. Đối tượng được thêm vào đầu tiên sẽ ở trên cùng



Chương 3. Lập trình giao diện đồ họa

Cách xây dựng đối tượng JFrame



Ví dụ 3.1 File đặt trong tệp: FirstSwingExample.java

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

import thư
viện swing

```
public class FirstSwingExample {
```

```
public static void main(String[] args) {
```

```
JFrame f=new JFrame();//creating instance of JFrame
```

```
JButton b=new JButton("click");//creating instance of JButton
```

```
b.setBounds(80,80,100, 40);//x axis, y axis, width, height
```

```
f.add(b);//adding button in JFrame
```

```
f.setSize(300,200);//300 width and 200 height
```

```
f.setLayout(null);//using no layout managers
```

```
f.setVisible(true);//making the frame visible
```

```
}
```

```
}
```

Chương 3. Lập trình giao diện đồ họa



- Kết quả



Chương 3. Lập trình giao diện đồ họa

Cách xây dựng đối tượng JFrame



■ Ví dụ 3.2:

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

import thư viện
awt và swing

```
public class GUI extends JFrame{  
    public static final int WIDTH =400;  
    public static final int HEIGHT =200;  
    public GUI(String title){  
        initGUI(title); // gọi phương thức định nghĩa khung giao diện  
    }  
    private void initGUI(String title){  
        setTitle(title);  
        setSize(WIDTH,HEIGHT);  
        setLocationRelativeTo(null);
```

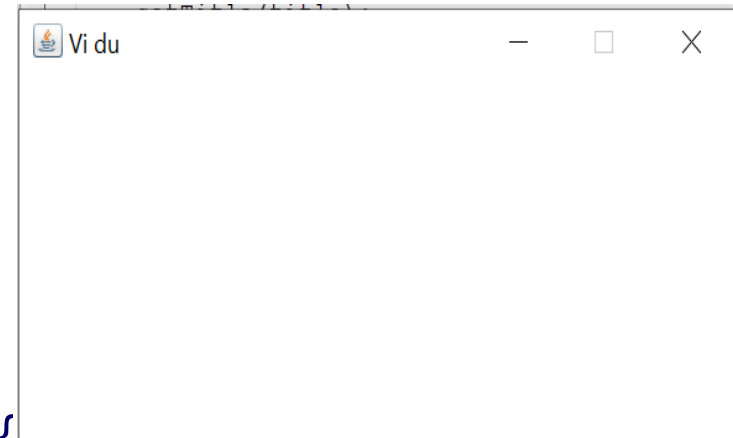
Chương 3. Lập trình giao diện đồ họa



```
getContentPane().setBackground(Color.WHITE);  
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
setResizable(false);  
setLayout(new FlowLayout());  
}
```

```
}
```

```
public class Main{  
    public static void main(String[] args){  
        GUI gui = new GUI("Vi du");  
        gui.setVisible(true);  
    }  
}
```



Chương 3. Lập trình giao diện đồ họa

JPanel



- JPanel là một lớp container đơn giản nhất. Nó cung cấp không gian để các ứng dụng có thể thêm các component vào. Nó được kế thừa từ lớp JComponent.
- JPanel không có thanh tiêu đề (title bar).

Chương 3. Lập trình giao diện đồ họa

JPanel – Một số constructor phổ biến



- **JPanel():** Tạo một JPanel mới mặc định sử dụng FlowLayout.
- **JPanel(LayoutManager layout):** Tạo một JPanel mới với Layout Manager đã cho

Chương 3. Lập trình giao diện đồ họa

JPanel – Ví dụ 1



Ví dụ 3.3

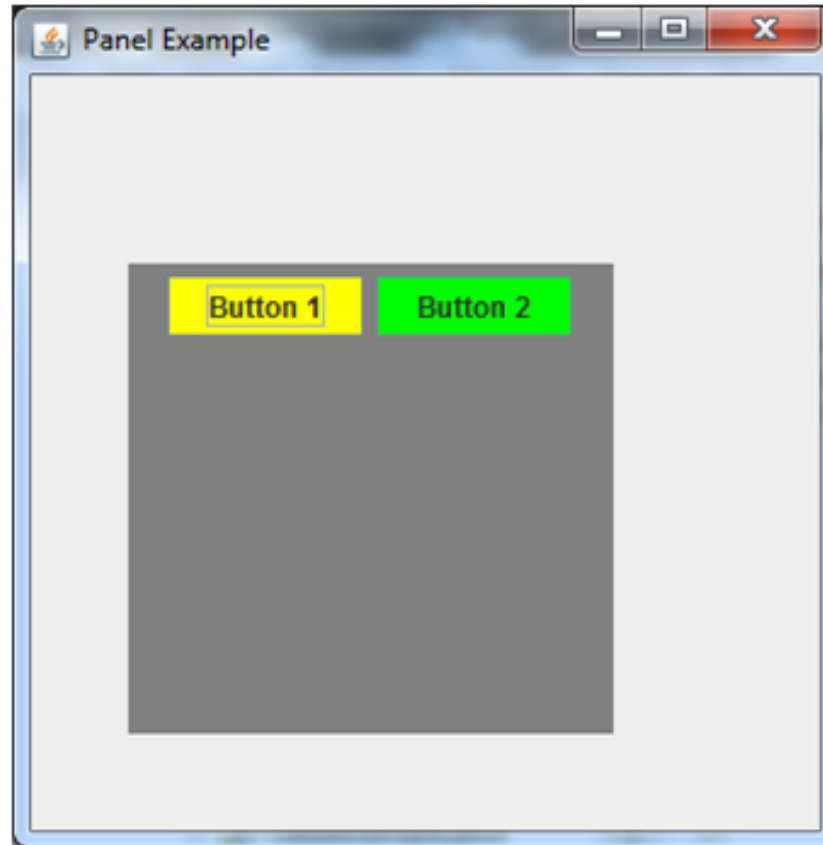
```
import java.awt.*;
import javax.swing.*;
public class PanelExample {
    PanelExample()
    {
        JFrame f= new JFrame("Panel Example");
        JPanel panel=new JPanel();
        panel.setBounds(40,80,200,200);
        panel.setBackground(Color.gray);
        JButton b1=new JButton("Button 1");
        b1.setBounds(50,100,80,30);
    }
}
```


Chương 3. Lập trình giao diện đồ họa



```
b1.setBackground(Color.yellow);
JButton b2=new JButton("Button 2");
b2.setBounds(100,100,80,30);
b2.setBackground(Color.green);
panel.add(b1); panel.add(b2);
f.add(panel);
f.setSize(400,400);
f.setLayout(null);
f.setVisible(true);      }
public static void main(String args[]) {
new PanelExample();
} }
```

Chương 3. Lập trình giao diện đồ họa



Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JPanel – Ví dụ 2



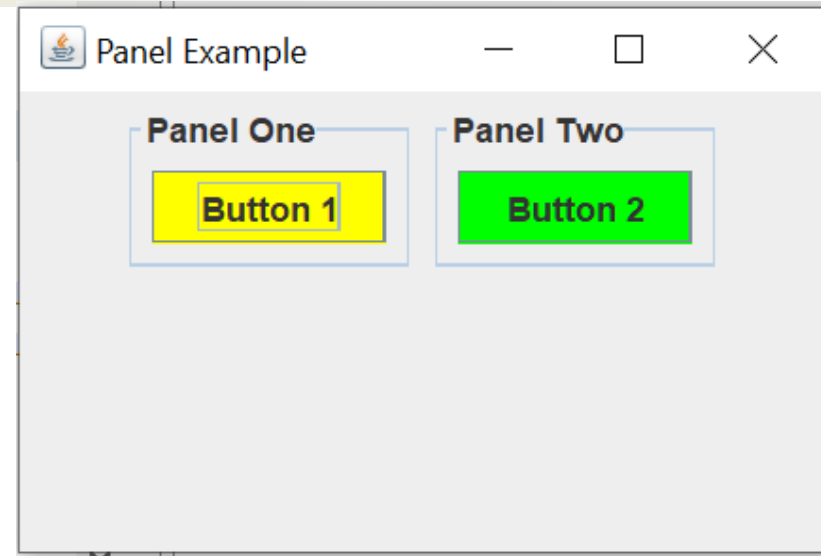
```
PanelExample() {  
    JFrame f= new JFrame("Panel Example");  
    JPanel panel1=new JPanel();  
    JPanel panel2=new JPanel();  
    panel1.setSize(100,100);  
    panel2.setSize(100,100);  
    JButton b1=new JButton("Button 1");  
    b1.setBackground(Color.yellow);  
    JButton b2=new JButton("Button 2");  
    b2.setBackground(Color.green);  
}
```

Chương 3. Lập trình giao diện đồ họa



```
panel1.add(b1);
panel2.add(b2);
f.setLayout(new FlowLayout());
f.add(panel1);
f.add(panel2);
f.setSize(300,200);
f.setVisible(true);

panel1.setBorder(BorderFactory.createTitledBorder("Panel One"));
panel2.setBorder(BorderFactory.createTitledBorder("Panel Two"));
}
```



Chương 3. Lập trình giao diện đồ họa

JLabel



- JLabel: là đối tượng giao diện hiển thị một tiêu đề có tác dụng chú thích hoặc thông báo trên giao diện
- JLabel: không sửa được văn bản hiển thị đối với người dùng.

Chương 3. Lập trình giao diện đồ họa

JLabel – Một số constructor phổ biến



- `JLabel()`: tạo một thể hiện JLabel không có hình ảnh và tiêu đề trống.
- `JLabel(String s)`: tạo một thể hiện JLabel với text cụ thể.
- `JLabel(Icon i)`: tạo một thể hiện JLabel với 1 ảnh xác định.
- `JLabel(String s, Icon i, int horizontalAlignment)`: tạo một thể hiện JLabel với text xác định có ảnh và được căn chỉnh theo chiều ngang.

Chương 3. Lập trình giao diện đồ họa

JLabel – Một số phương thức thường dùng



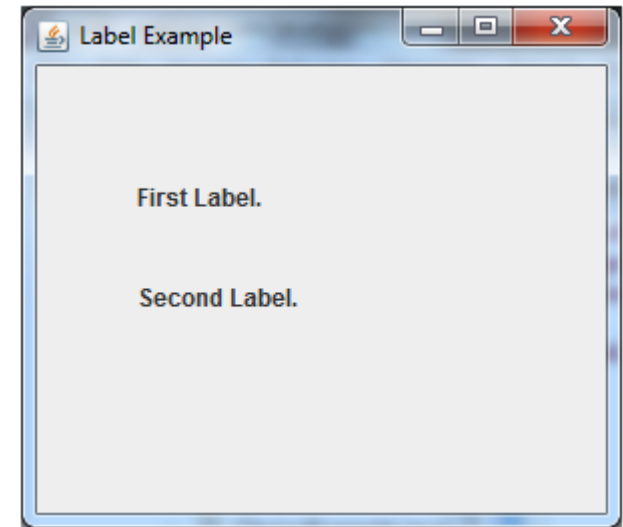
- `String getText()`: trả về chuỗi mà label đang hiển thị.
- `void setText(String text)`: thiết lập 1 dòng đơn chuỗi text hiển thị trên component.
- `void setHorizontalAlignment(int alignment)`: thiết lập nội dung hiển thị của label theo hàng ngang trục X.
- `Icon getIcon()`: trả về hình ảnh mà label hiển thị.
- `int getHorizontalAlignment()`: trả về dạng căn chỉnh của nội dung label theo trục X.

Chương 3. Lập trình giao diện đồ họa

JLabel – Ví dụ 3.5



```
import javax.swing.*;
class LabelExample{
public static void main(String args[])    {
    JFrame f= new JFrame("Label Example");
    JLabel l1,l2;
    l1=new JLabel("First Label.");
    l1.setBounds(50,50, 100,30);
    l2=new JLabel("Second Label.");
    l2.setBounds(50,100, 100,30);
    f.add(l1); f.add(l2);
    f.setSize(300,300);
    f.setLayout(null);
    f.setVisible(true);    } }
```

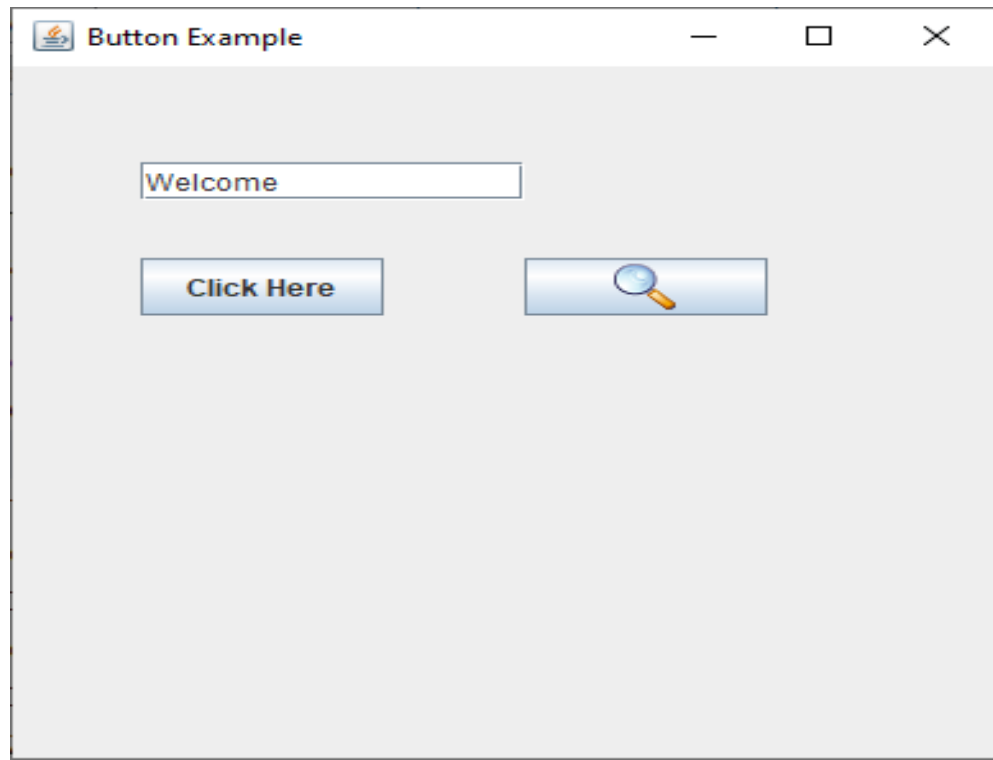


Chương 3. Lập trình giao diện đồ họa

JTextField



- JTextField: đối tượng dùng để nhập dòng dữ liệu đơn.



Chương 3. Lập trình giao diện đồ họa

JTextField – Một số constructor thường dùng



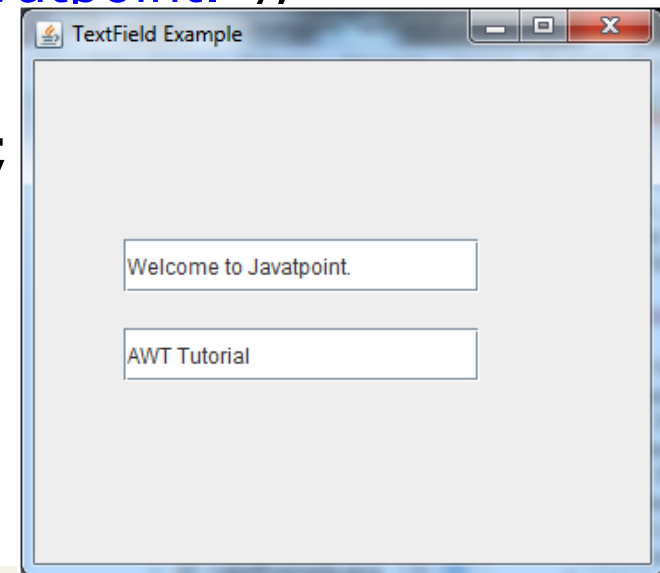
- `JTextField()`: tạo 1 textfield trống mới
- `JTextField(String text)`: tạo 1 textfield có nội dung là text.

Chương 3. Lập trình giao diện đồ họa

JTextField – Ví dụ



```
import javax.swing.*;  
class TextFieldExample {  
public static void main(String args[]) {  
    JFrame f= new JFrame("TextField Example");  
    JTextField t1,t2;  
    t1=new JTextField("Welcome to Javatpoint.");  
    t1.setBounds(50,100, 200,30);  
    t2=new JTextField("AWT Tutorial");  
    t2.setBounds(50,150, 200,30);  
    f.add(t1); f.add(t2);  
    f.setSize(400,400);  
    f.setLayout(null);  
    f.setVisible(true); } }
```



Chương 3. Lập trình giao diện đồ họa

JButton



- JButton: là một component được sử dụng để tạo ra một nút có nhãn được thực thi một cách độc lập. Thực thi 1 hành động nào đó khi người dùng nhấn nút.

Chương 3. Lập trình giao diện đồ họa

JButton – Một số constructor phổ biến



- JButton(): tạo ra 1 nút không có text cũng như ảnh.
- JButton(String s): tạo ra nút có text.
- JButton(Icon i): tạo ra nút có ảnh

Chương 3. Lập trình giao diện đồ họa

JButton – Một số method thường dùng



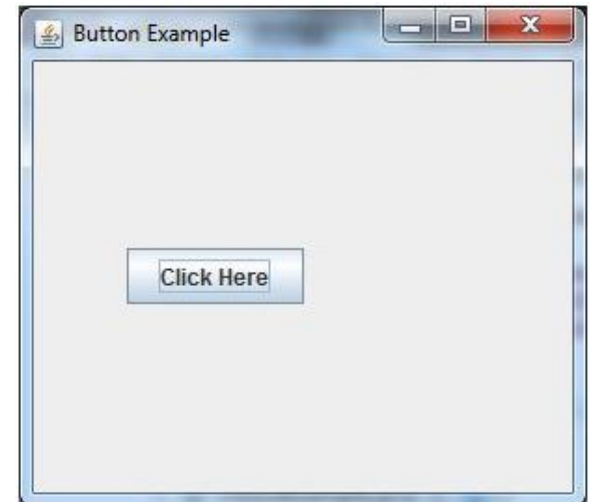
- `void setText(String s)`: dùng để thiết lập text cho nút .
- `String getText()`: trả về text của nút
- `void setEnabled(boolean b)`: để thiết lập nút được kích hoạt hay không.
- `void setIcon(Icon b)`: thiết lập ảnh cho nút.
- `Icon getIcon()`: trả về icon đang dung của nút.
- `void setMnemonic(int a)`: thiết lập mnemonic (phím tắt) cho nút.
- `void addActionListener(ActionListener a)`: dùng để thêm action listener cho nút.

Chương 3. Lập trình giao diện đồ họa

JButton – Ví dụ



```
import javax.swing.*;  
public class ButtonExample {  
  public static void main(String[] args) {  
    JFrame f=new JFrame("Button Example");  
    JButton b=new JButton("Click Here");  
    b.setBounds(50,100,95,30);  
    f.add(b);  
    f.setSize(400,400);  
    f.setLayout(null);  
    f.setVisible(true);  
  }  
}
```



Chương 3. Lập trình giao diện đồ họa

Bài tập



1. Tạo 1 frame 250×200 với tiêu đề “HelloWorld” hiển thị ở tọa độ (300, 200) trên màn hình. Trong Frame có nút click. Tắt chương trình khi người dùng click vào dấu X ở góc phải trên cùng.
2. Thiết kế giao diện sau (sử dụng FlowLayout).



Chương 3. Lập trình giao diện đồ họa

Bài tập



3. Thiết kế giao diện sau (sử dụng GridLayout)



Chương 3. Lập trình giao diện đồ họa



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

public class GridLayout_Example {
    public static void main(String[] args) {
        JFrame frame = new JFrame("GridLayout Test");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new GridLayout(3, 3, 5, 5));
        frame.add(new JButton("Press 1"));
        frame.add(new JButton("Press 2"));
        frame.add(new JButton("Press 3"));
        frame.add(new JButton("Press 4"));
    }
}
```

Chương 3. Lập trình giao diện đồ họa

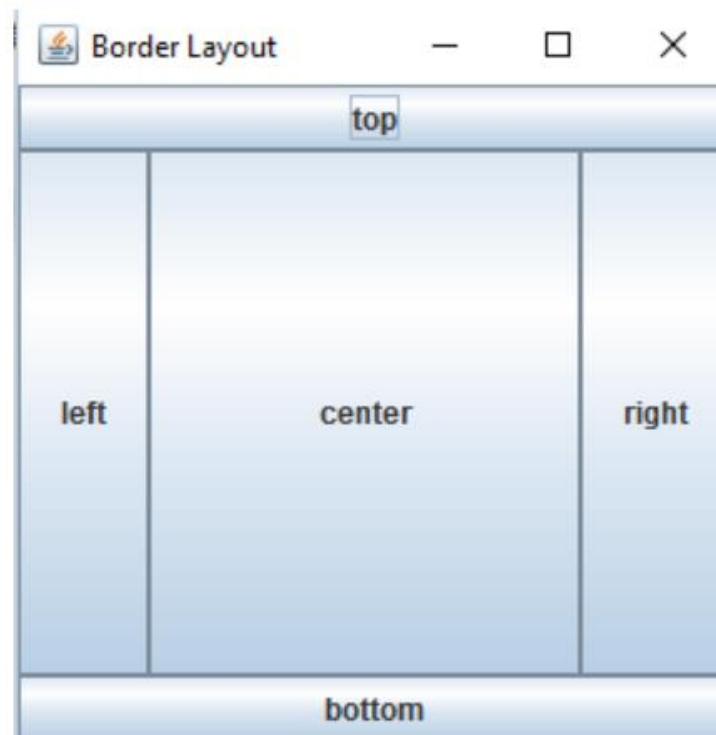


```
frame.add(new JButton("Press 5"));  
frame.add(new JButton("Press 6"));  
frame.add(new JButton("Press 7"));  
frame.add(new JButton("Press 8"));  
frame.setSize(300,200);  
frame.setVisible(true);  
  
}  
}
```

Chương 3. Lập trình giao diện đồ họa



4. Thiết kế giao diện sau (sử dụng BorderLayout)



Chương 3. Lập trình giao diện đồ họa



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

public class BorderLayout_Example {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Border Layout");
        JButton button, button1, button2, button3, button4;
        button = new JButton("left");
        button1 = new JButton("right");
        button2 = new JButton("top");
        button3 = new JButton("bottom");
        button4 = new JButton("center");
```


Chương 3. Lập trình giao diện đồ họa



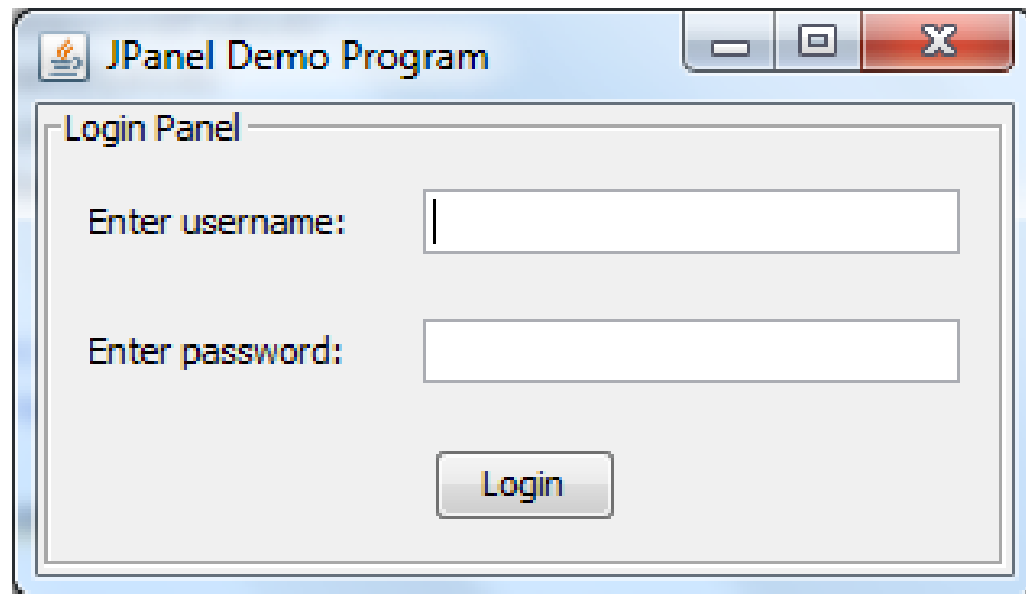
```
frame.add(button, BorderLayout.WEST);  
frame.add(button1, BorderLayout.EAST);  
frame.add(button2, BorderLayout.NORTH);  
frame.add(button3, BorderLayout.SOUTH);  
frame.add(button4, BorderLayout.CENTER);  
frame.setSize(300,300);  
frame.setVisible(true);  
  
}  
}
```

Chương 3. Lập trình giao diện đồ họa

Bài tập



5. Thiết kế giao diện sau, kết thúc chương trình khi tắt cửa sổ.

A screenshot of a Java Swing window titled "JPanel Demo Program". The window has a standard Mac OS X-style title bar with minimize, maximize, and close buttons. Inside the window, there is a "Login Panel" with a title bar. The panel contains two text input fields: "Enter username:" and "Enter password:". Below the password field is a "Login" button. The window is styled with a light blue border and a white background for the panel.