Bài 2

Form Layout

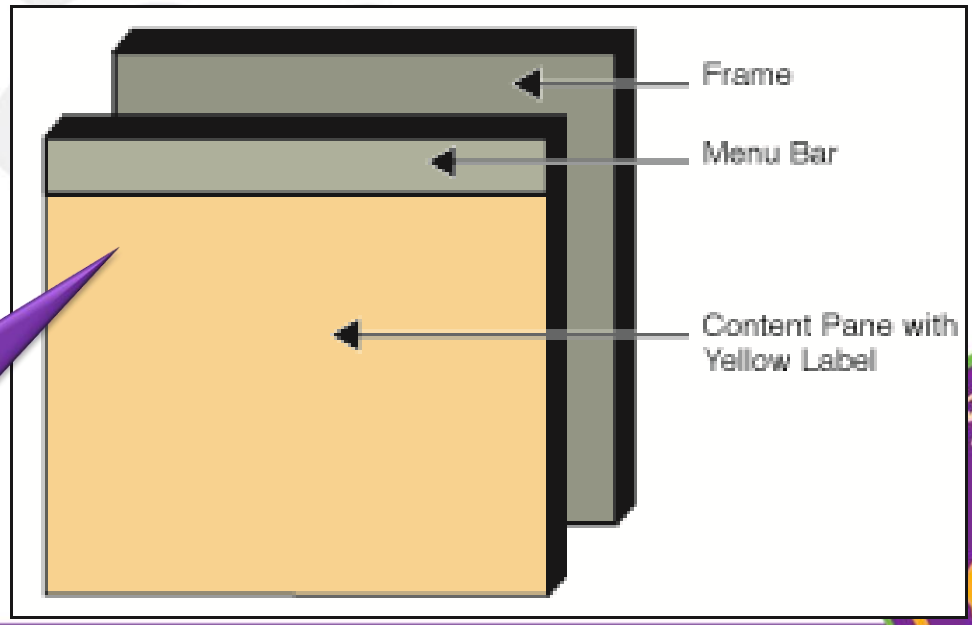
Mục tiêu

- Giới thiệu Layout
- FlowLayout
- BorderLayout
- GridLayout
- CardLayout
- Layout bổ sung
- Layout mới
- Kích thước

Giới thiệu

Cấp cao nhất của đối tượng containers như là Frame, Window là:

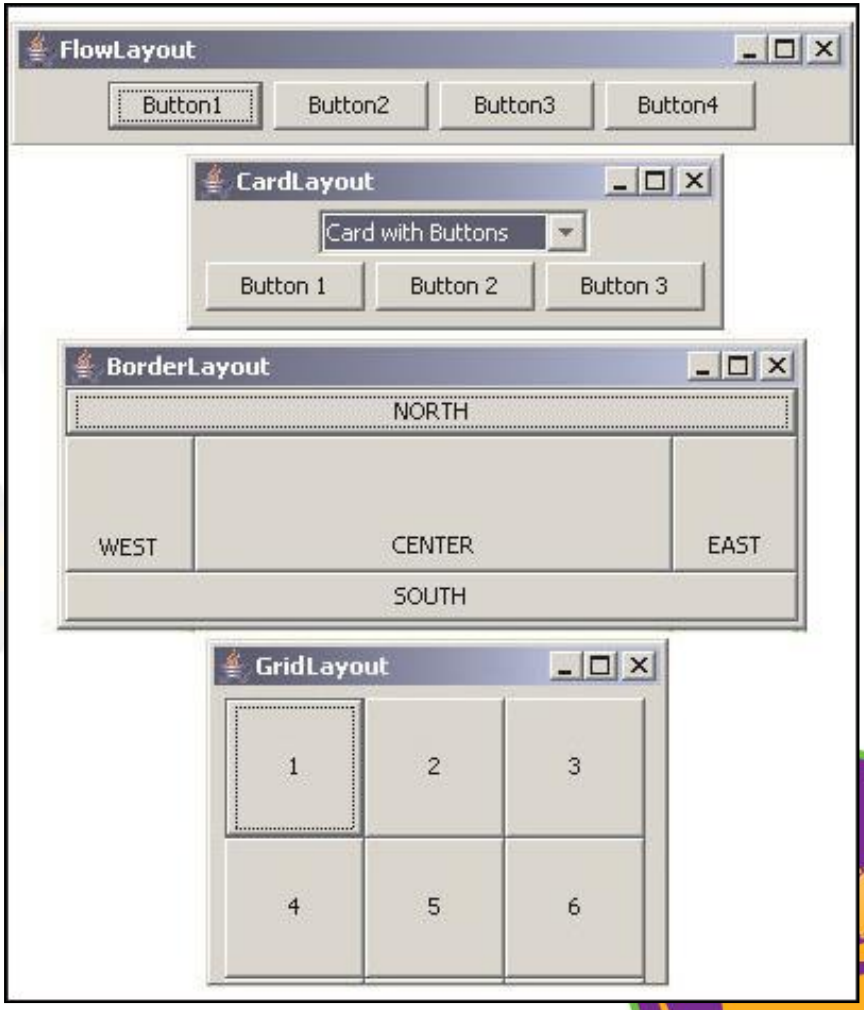
- Chịu trách nhiệm cung cấp sắp xếp mặc định cho thành phần được thêm vào chúng.
- Liên kết với khung nội dung chứa thành phần có thể nhìn thấy.



Trình quản lý bố cục ảnh hưởng tới khung nội dung

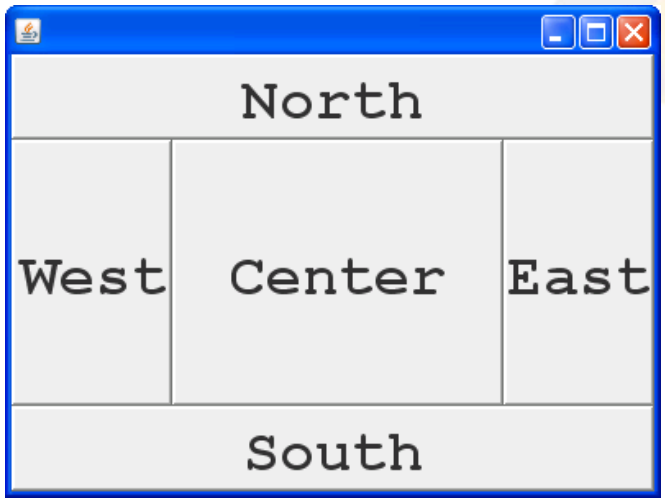
Giới thiệu

- Một **Layout Manager**:
  - Là đối tượng Java kết hợp với container.
  - **Quản lý** vị trí và kích thước của thành phần.
  - **Đơn giản hóa** việc thêm thành phần vào container.
- Tất cả các **container** đều có layout mặc định
- Khi container thay đổi thì **layout manager** tự động co dãn theo…



Giới thiệu

- Có thể thay đổi bố cục bằng Layout Manager nếu muốn.
- Có 2 bước để gọi và và sử dụng layout manager:
  - B1: gọi phương thức **setLayout()** để cài đặt bố cục khác.
  - B2: xác định các kích cỡ của các thành phần.

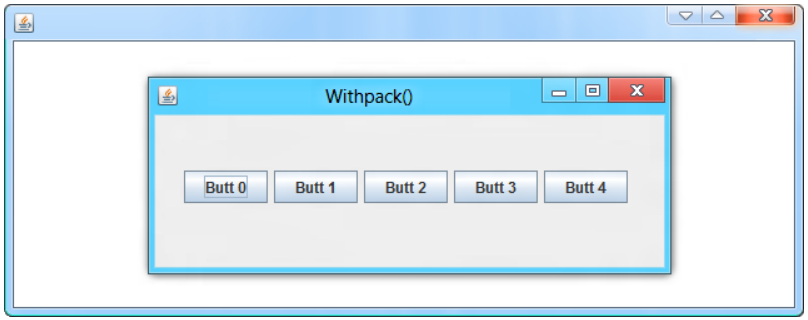


Giới thiệu

- Các phương thức sau chỉ định kích thước:
  - void **setPreferredSize**(Dimension preferredSize): xác định kích thước ưu tiên của thành phần, layout manager sử dụng kích thước này khi thành phần thêm vào khung chứa.
  - void **setMinimumSize**(Dimension minimumSize): sử dụng để định kích thước tối thiểu của thành phần, layout manager tham chiếu để định kích thước tối thiểu.
  - void setMaximumSize(Dimension maximumSize): cài đặt kích thước tối đa cho thành phần, layout manager sẽ tham chiếu để định kích thước tối đa.

Giới thiệu

- Phương thức **pack**() sẽ gọi tới **getPreferredSize**() để xác định kích thước tốt nhất cho các thành phần.
- Phương thức **getPreferredSize**() được gọi đệ quy cho tất cả thành phần đến khi tổng kích thước của khung (**frame**) được tính toán.
- Từ kích thước này, kích thước của cạnh khung (**frame**) và thanh menu (nếu có) được thêm vào.
- Khi có kích thước cuối cùng, **layout manager** sẽ trình bày các thành phần lên khung.



Giới thiệu

Các layout manager có sự khác biệt như sau:

- **FlowLayout Manager**: sắp xếp các thành phần từ trái qua phải.
- **BorderLayout Manager**: chỉ cho phép các thành phần nằm tại 5 vị trí trong container. Các vị trí là East, West, North, South và Center.
- **GridLayout Manager**: bố cục các thành phần theo hàng và cột.
- **CardLayout Manager**: bố cục các thành phần như bộ bài. Thành phần trên cùng nhìn thấy được, các thành phần còn lại có thể xáo trộn.

FlowLayout

FlowLayout Manager:

- Các thành phần nằm trên một hàng sắp xếp từ trái qua phải trong cùng container.
- Nếu không đủ chỗ, nó tự động xuống dòng.
- Kích thước preferred của thành phần được sử dụng.
- Nếu container rộng hơn mức cần thiết của dãy thành phần, mặc định tự động căn giữa.
- Có thể chỉ định căn trái hoặc phải và các padding dọc, ngang xung quanh thành phần.

FlowLayout

FlowLayout Manager: code demo

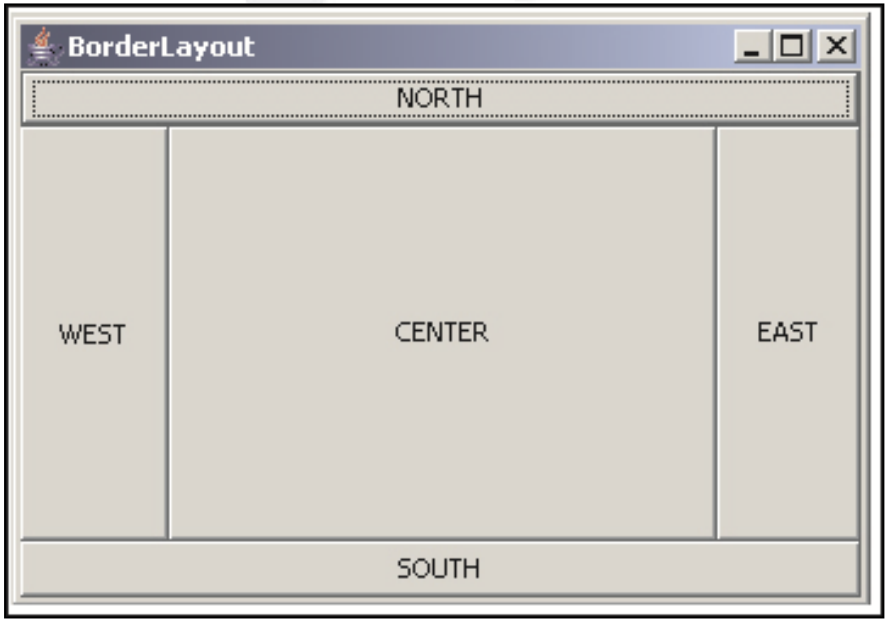
Code Snippet

```
public class Login extends JFrame {
// GUI components declaration
JLabel lblName;
JTextField txtName;
 public Login() {
   // Sets the layout of the frame to the flow layout
     setLayout(new FlowLayout());
  // Creates a label with the name “Name”
     lblName = new JLabel(“Name”);
  // Adds the label to the frame
     getContentPane().add(lblName);
  // Creates a textfield.
     txtName  = new JTextField();
 // Adds the textfield to the frame
     getContentPane().add(txtName);
 }
}
```

BorderLayout

BorderLayout Manager:

- Cho phép các thành phần đặt tại các vị trí Đông, Tây, Nam, Bắc và Trung tâm.
- Jframe và Dialog sử dụng BorderLayout như là mặc định.

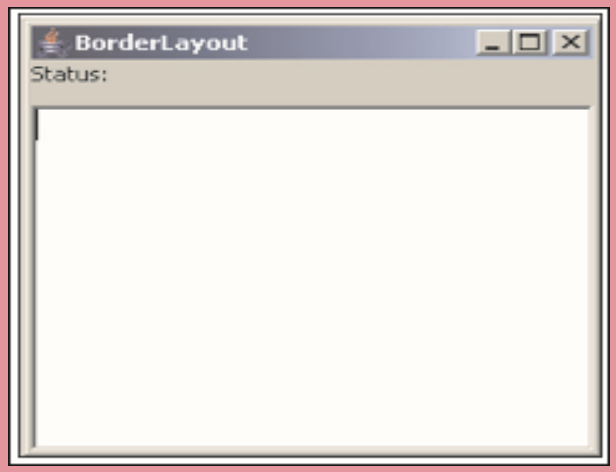


BorderLayout

BorderLayout Manager: code demo

Code Snippet

```
JPanel  pnlPanel;
JLabel lblStatus;
JTextArea txaNotes;
pnlPanel = new JPanel();
// Changes the layout of the panel to BorderLayout
   pnlPanel.setLayout(new BorderLayout());
// Creates a label with the name “Status”
   lblStatus = new JLabel(“Status”);
// Adds the label to the panel in the north direction
   pnlPanel.add(lblStatus, BorderLayout.NORTH);
// Creates a textarea
   txaNotes = new JTextArea(); /
/ Adds the textarea to the panel in the center.
  pnlPanel.add(txaNotes, BorderLayout.CENTER);
. . .
```



BorderLayout

GridLayout Manager:

- Các thành phần bố cục theo hàng và cột.
- Các cell có cùng kích cỡ.
- Nếu container thay đổi kích thước thì sẽ được phân bố đều lại giữa các thành phần.

BorderLayout

GridLayout Manager: code demo

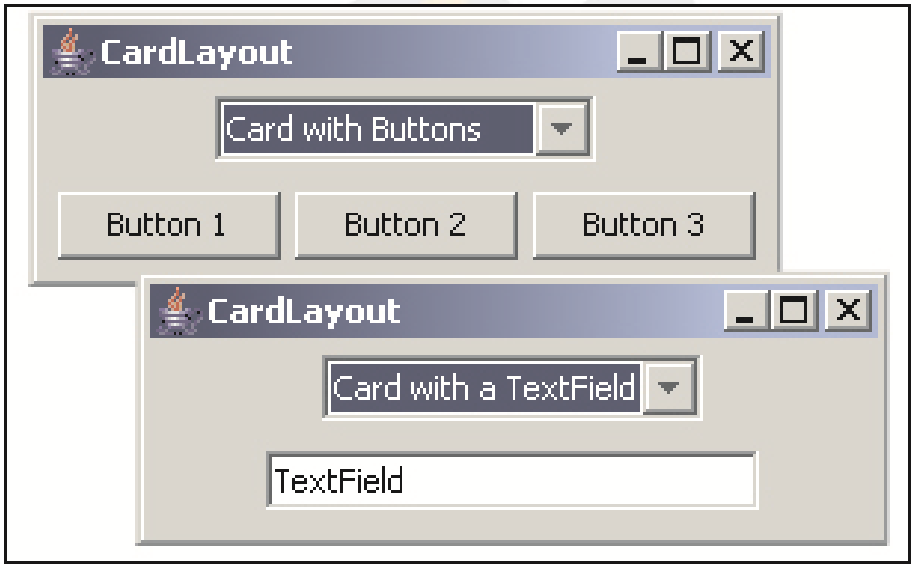
Code Snippet

```
JPanel pnlNumericPad;
JButton  btnOne, btnTwo, btnThree, btnFour;
// Creates a panel
   pnlNumericPad = new JPanel();
// 2 rows and 2 columns
   pnlNumericPad.setLayout(new Gridlayout(2, 2));
// Create buttons 1 to 4
   btnOne = new JButton(“1”);
   btnTwo = new JButton(“2”);
   btnThree = new JButton(“3”);
   btnFour = new JButton(“4”);
// Add the buttons 1 to 4 to the panel
   pnlNumericPad.add(btnOne);
   pnlNumericPad.add(btnTwo);
   pnlNumericPad.add(btnThree);
   pnlNumericPad.add(btnFour);
```

BorderLayout

CardLayout Manager:

- Các thành phần bố cục theo dạng stack như bộ bài.
- Chỉ có 1 thành phần hiển thị tại 1 thời điểm.
- Để hiển thị thành phần khác phải lật nó dựa trên sự kiện cài đặt của thành phần khác (dropdown, radio…).



BorderLayout

CardLayout Manager: code demo

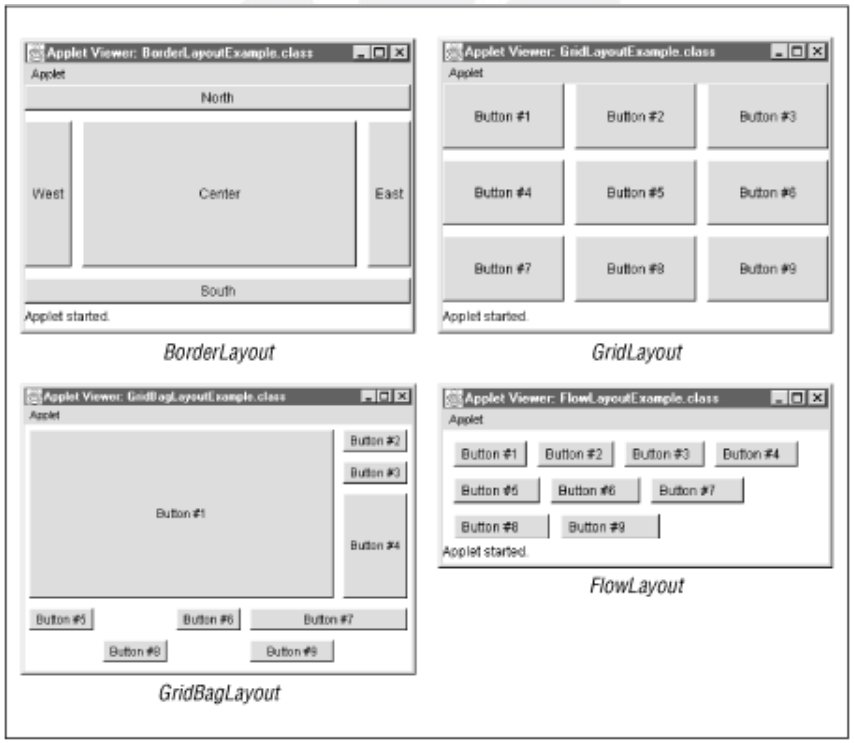
Code Snippet

```
CardLayout cardLayout;
JPanel pnlSubjects;
JPanel pnlEnglish, pnlScience, pnlMaths;
// Creates the card layout manager
   cardLayout = new CardLayout();
// Creates a main panel
   pnlSubjects = new JPanel();
// Sets the layout of the panel
   pnlSubjects.setLayout(cardLayout);
// Creates panels representing three cards
pnlEnglish = new JPanel();
pnlScience = new JPanel();
     pnlMaths = new JPanel();
// Code to add the components to each card
   . . . . . .
// Add the card to the main panel
   pnlSubjects.add(pnlEnglish, “English”);
   pnlSubjects.add(pnlScience, “Science”);
   pnlSubjects.add(pnlMaths, “Maths”);
```

Layout bổ sung

Các layout bổ sung trong Swing:

- **GridBagLayout**
- **Absolute Positioning**



Layout bổ sung

GridBagLayout:

- Bố cục phức tạp nhưng linh hoạt
- Đặt các thành phần trong hàng và cột.
- Tuy nhiên có thể trộn hàng, cột.
- Tất cả các hàng và cột không nhất thiết phải cùng chiều cao, rộng.
- Xác định ràng buộc trên mỗi thành phần. Ràng buộc này xác định thông qua lớp GridBagConstraints.

Layout bổ sung

Code demo:

Code Snippet

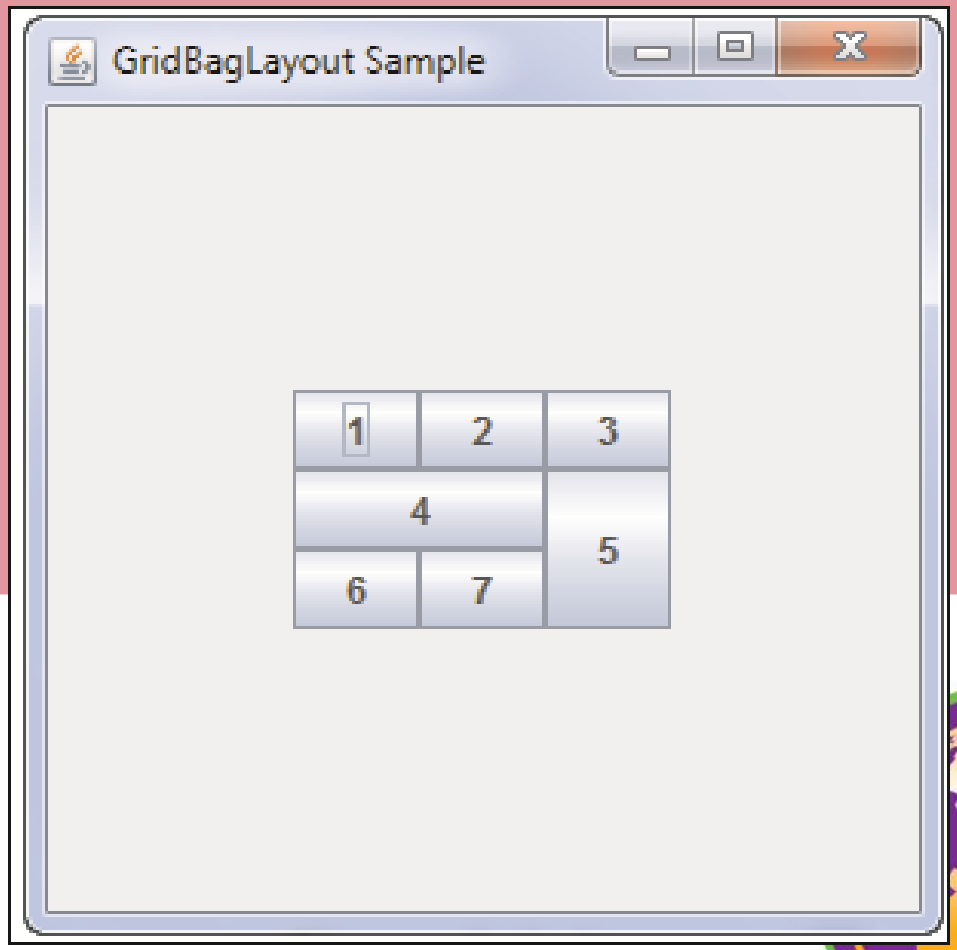
```
Container  c;
JButton btnButton1, btnButton2,btnButton3,btnButton4;
JButton btnButton5,btnButton6,btnButton7;

GridBagConstraints gbc;
. . .
c = getContentPane();

c.setLayout(new GridBagLayout());
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 0;

gbc.gridwidth = 1;
gbc.gridheight = 1;
```

Layout bổ sung

```
btnButton1 = new JButton(“1”);
c.add(btnButton1, gbc);
gbc.gridx = 1;

btnButton2 = new JButton(“2”);
c.add(btnButton2, gbc);
gbc.gridx = 2;

btnButton3 = new JButton(“3”);
c.add(btnButton3, gbc);
gbc.gridx = 0;
gbc.gridy = 1;
gbc.gridwidth = 2;
gbc.gridheight = 1;
gbc.fill = GridBagConstraints.HORIZONTAL;

btnButton4 = new JButton(“4”);
c.add(btnButton4, gbc);  gbc.gridx = 2;
gbc.gridwidth = 1;
gbc.gridheight = 2;
gbc.fill = GridBagConstraints.VERTICAL;
```

Layout bổ sung

```
btnButton5 = new JButton(“5”);
c.add(btnButton5, gbc);
gbc.gridx = 0;
gbc.gridy = 2;
gbc.gridwidth = 1;
gbc.gridheight = 1;

btnButton6 = new JButton(“6”);
c.add(btnButton6, gbc);
gbc.gridx = 1;

btnButton7 = new JButton(“7”);
c.add(btnButton7, gbc);
setVisible(true);
```



Layout bổ sung

Absolute Positioning:

- Là kiểu bố cục tuyệt đối hoặc null, nó cho phép xác định giới hạn thành phần trước khi thêm vào container.
- Các giới hạn như là vị trí x, y theo pixel, chiều rộng/cao.
- Cú pháp: **setLayout(null)**;

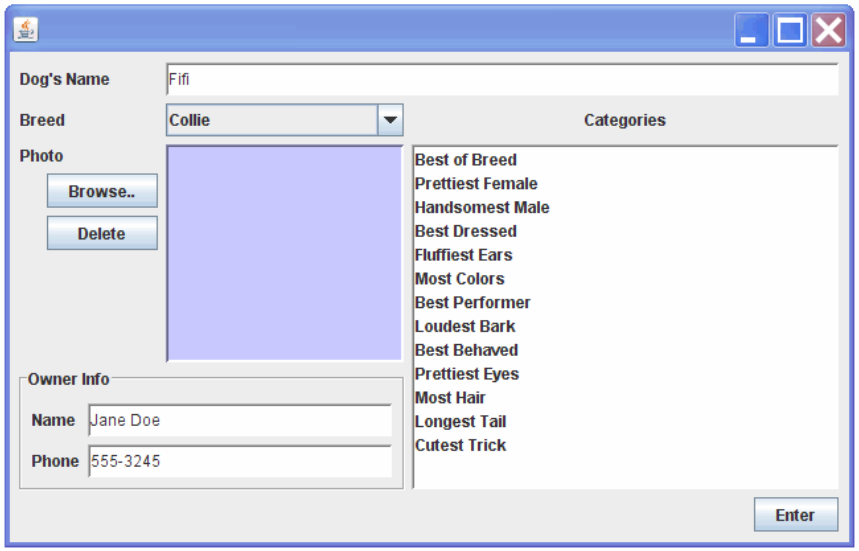
Code Snippet

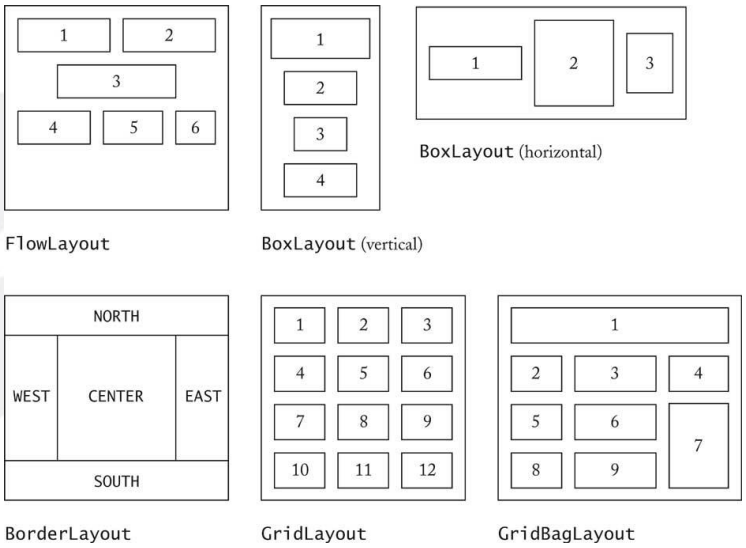
```
Container container;
JButton btnClick;
JTextField txtText;
. . .
container = getContentPane();
container.setLayout(null);
btnClick = new JButton(“Click”);
btnClick.setBounds(50,50,75,25);
container.add(btnClick);
txtText = new JTextField();
txtText.setBounds(130,50,200,25);
container.add(txtText);
```

Layout mới

Các layout mới trong Swing:

- BoxLayout
- Spring Layout
- GroupLayout.

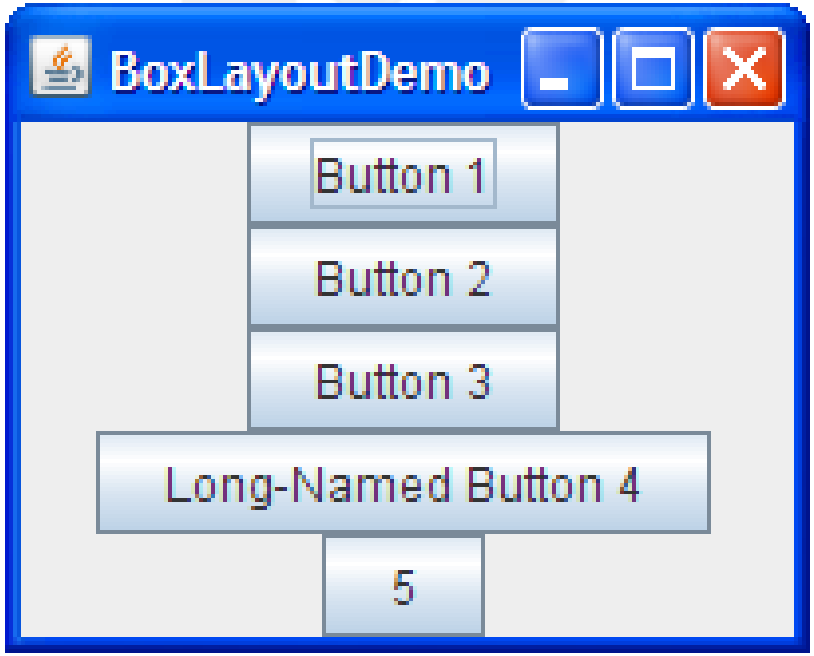




Layout mới

BoxLayout:

- Tương tự FlowLayout, ngoại trừ việc đặt các thành phần lên nhau trong một cột hoặc cạnh nhau theo chiều ngang trong một hàng.



Layout mới

Các tham số:

1. • `X_AXIS`: Thành phần được sắp xếp ngang theo chiều trái qua phải.
2. • `Y_AXIS`: Thành phần được sắp xếp dọc theo chiều từ trên xuống.
3. • `Line_AXIS`: Thành phần sắp xếp theo hướng dòng văn bản (xác định bởi thuộc tính ComponentOrientation của container).
4. • `Page_AXIS`: Thành phần sắp xếp theo hướng dòng chảy trên trang (xác định bởi thuộc tính ComponentOrientation của container).

Layout mới

BoxLayout:

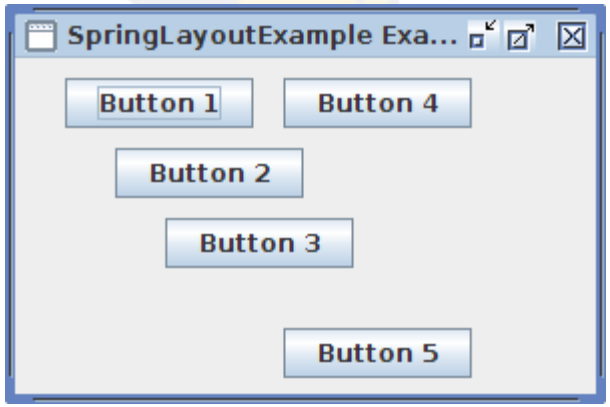
Code Snippet

```
. . .
public class BoxLayoutManager {
  public BoxLayoutManager(){  . . .  }
  public static void main(String args[]) {
   JPanel container = new JPanel();
   container.setBorder(BorderFactory.createTitledBorder(“ BoxLayout”));
   BoxLayout layout = new BoxLayout(container, BoxLayout.Y_AXIS);
   container.setLayout(layout);
   JButton button1 = new JButton(“Button1”);
   container.add(button1);
   button2 = new JButton(“Button2”);
   container.add(button2);
   button3 = new JButton(“Button3”);
   container.add(button3);
   . . .
  }
}
```

Layout mới

SpringLayout:

- Hoạt động bằng cách định nghĩa mối quan hệ giữa các thành phần.
- Khoảng cách giữa các cạnh thể hiện bởi đối tượng Spring.
- Mỗi Spring có 4 đặc tính: minimum, preferred, maximum và giá trị actual (hiện tại) của nó.



Layout mới

Code Snippet

```
. . .
/**  Defining SpringLayout manager **/
public class SpringLayoutManager extends JFrame{
 public SpringLayoutManager(String title) {
  super(title);
  Container content_pane = this.getContentPane();
  // Create an instance of SpringPanel.
     SpringPanel spring_panel = new SpringPanel();
  // Add it to frame
     content_pane.add (spring_panel);
```

Layout mới

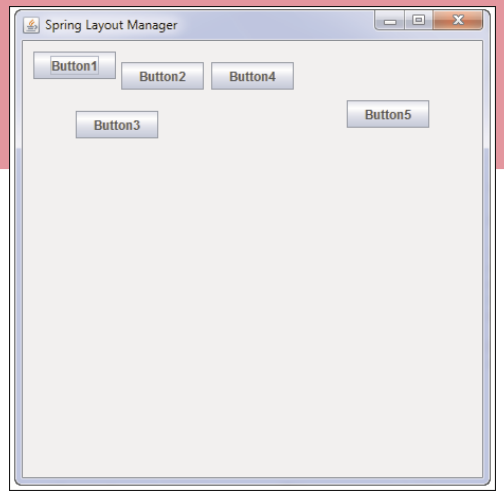
```
   setSize(300, 200);
       setVisible(true);
 }
 public static void main(String[] args){
  SpringLayoutManager slm = new SpringLayoutManager(“Spring Layout
   Manager”);
 }
}

/** Layout five buttons using a SpringLayout Manager. **/

class SpringPanel extends JPanel {
/**
     * The Constructor creates 5 buttons
     * and constrains each to a particular position relative  to the
     * panel.
 **/
    SpringPanel () {
     SpringLayout layout = new SpringLayout ();
     setLayout (layout);
     JButton btn1  = new JButton (“Button1”);
```

Layout mới

```
   JButton btn2  = new JButton (“Button2”);
   JButton btn3  = new JButton (“Button3”);
   JButton btn4  = new JButton (“Button4”);
   JButton btn5  = new JButton (“Button5”);
   add (btn1);
   add (btn2);
   add (btn3);
   add (btn4);
   add (btn5);

// Set the distances between the edges . Put the first button at pixel
// co-ordinates (10,10) relative to the panel’s frame
   layout.putConstraint (SpringLayout.WEST, btn1, 10, SpringLayout.WEST, this);
   layout.putConstraint (SpringLayout.NORTH, btn1, 10, SpringLayout.NORTH, this);

// Put the second button 5 pixels to the right of the first button and 20
// pixels below the top panel edge.
   layout.putConstraint (SpringLayout.WEST, btn2, 5, SpringLayout.EAST, btn1);
   layout.putConstraint (SpringLayout.NORTH, btn2, 20,SpringLayout.NORTH, this);
```

Layout mới

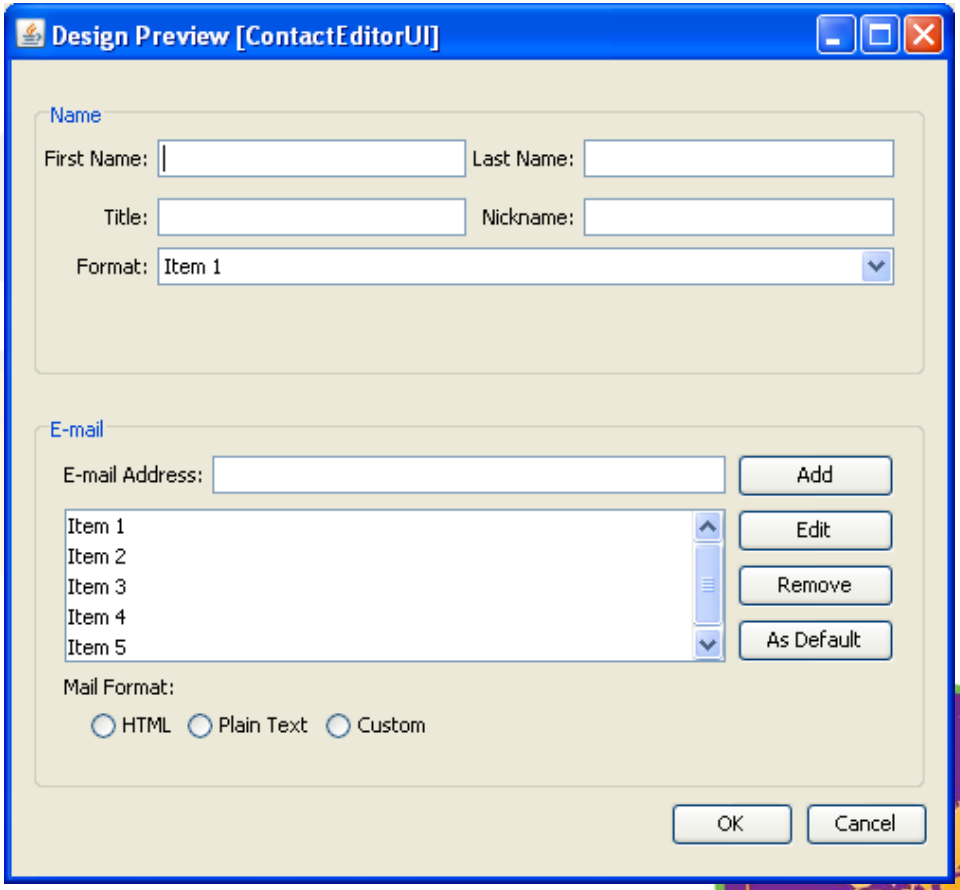
```
// Put the third button 50 pixels to the left of the
// panel edge and 20 pixels above the second button.
   layout.putConstraint (SpringLayout.WEST, btn3,50, SpringLayout.WEST, this);
   layout.putConstraint (SpringLayout.NORTH, btn3, 20, SpringLayout.SOUTH, btn2);

// Put the fourth button 50 pixels to the right of the
// third button  and 20 pixels below the top panel edge.
   layout.putConstraint (SpringLayout.WEST, btn4,50, SpringLayout.EAST, btn3);
   layout.putConstraint (SpringLayout.NORTH, btn4,20, SpringLayout.NORTH, this);
   layout.putConstraint (SpringLayout.WEST, btn5, 50, SpringLayout.EAST, btn4);
   layout.putConstraint (SpringLayout.NORTH, btn5, 10,SpringLayout.SOUTH, btn4);
 }
}
```



Layout mới

GroupLayout:

- Giúp nhóm các thành phần để định vị tốt hơn trong container.
- Các thành phần có thể được phân loại theo thứ bậc trong nhóm tuần tự hoặc song song.
- Bố cục được định nghĩa độc lập cho mỗi chiều.



Layout mới

Code demo 3 thành phần tuần tự:

Code Snippet

```
horizontal layout = sequential group { c1, c2, c3 }
vertical layout = parallel group (BASELINE) { c1, c2, c3}
```

Code demo thành phần song song:

Code Snippet

```
horizontal layout = sequential group { c1, c2, parallel group (LEFT) { c3, c4
} }
vertical layout = sequential group { parallel group (BASELINE) { c1, c2, c3
}, c4 }
```

Kích thước

- Nhiều phương thức của các thành phần trong Swing nhận hoặc trả về kích thước sử dụng một đối tượng của lớp Dimension.
- Đây là lớp thuận tiện gói gọn chiều rộng, cao của thành phần.
- Lớp Dimension lấy 2 số nguyên là tham số: rộng và cao.

Code Snippet

```
JButton btnOk;
. . .
btnOk = new JButton("OK");
btnOk.setMinimumSize(new Dimension(50,20));
btnOk.setMaximumSize(new Dimension(70,25));
btnOk.setPreferredSize(new Dimension(60,25));
```

Tóm tắt bài học

- ✓ Các **layout manager** hỗ trợ bố cục các thành phần trong bộ chứa container.
- ✓ Mỗi layout manager có tính chất, đặc điểm khác biệt.
- ✓ Bên cạnh đó việc cài đặt và sử dụng khá dễ dàng tuy mỗi loại layout manager có khác nhau về ràng buộc.
- ✓ Lập trình **Java Swing** với **Netbeans** được hỗ trợ giao diện đồ họa kéo thả kiểu "What you see is what you get" (**WYSiWYG**).
- ✓ Sử dụng cửa sổ **Navigator** tiện lợi theo dõi, cài đặt bố cục cho các container và thành phần bên trong nó.

HẾT
XIN CẢM ƠN!