

# Chương 5

## Tạo giao tiếp người dùng **Graphic User Interface- GUI**

# Mục tiêu

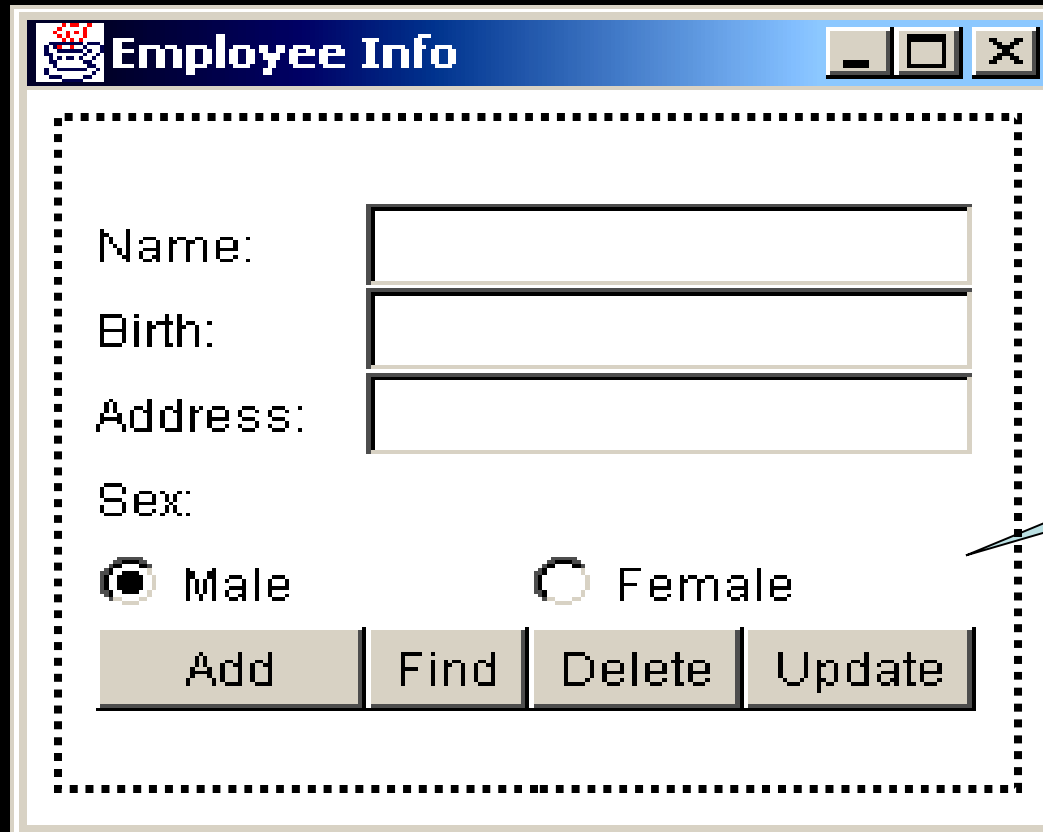
- Hiểu mục đích của gói AWT và cấu trúc của gói này.
- Biết cách sử dụng các đối tượng của gói AWT.
- Biết cách bố cục một GUI.
- Biết cách thiết kế một GUI.

## 5.2- GUI là gì?

- **GUI = Graphic User Interface** – mô hình giao tiếp kiểu tương tác giữa ứng dụng và user dạng đồ họa.
- Mỗi ngôn ngữ hỗ trợ cách tạo GUI khác nhau: C# , VB , VC++ lập trình viên phải phụ thuộc vào công cụ , trong khi Java , Python , Flex thì việc viết code lại cực kỳ đơn giản rất ít khi cần sử dụng công cụ kéo thả .

# GUI là gì?...

- GUI= Container + Components



Employee Info

Name:

Birth:

Address:

Sex:

☒ Male ☐ Female

Add Find Delete Update

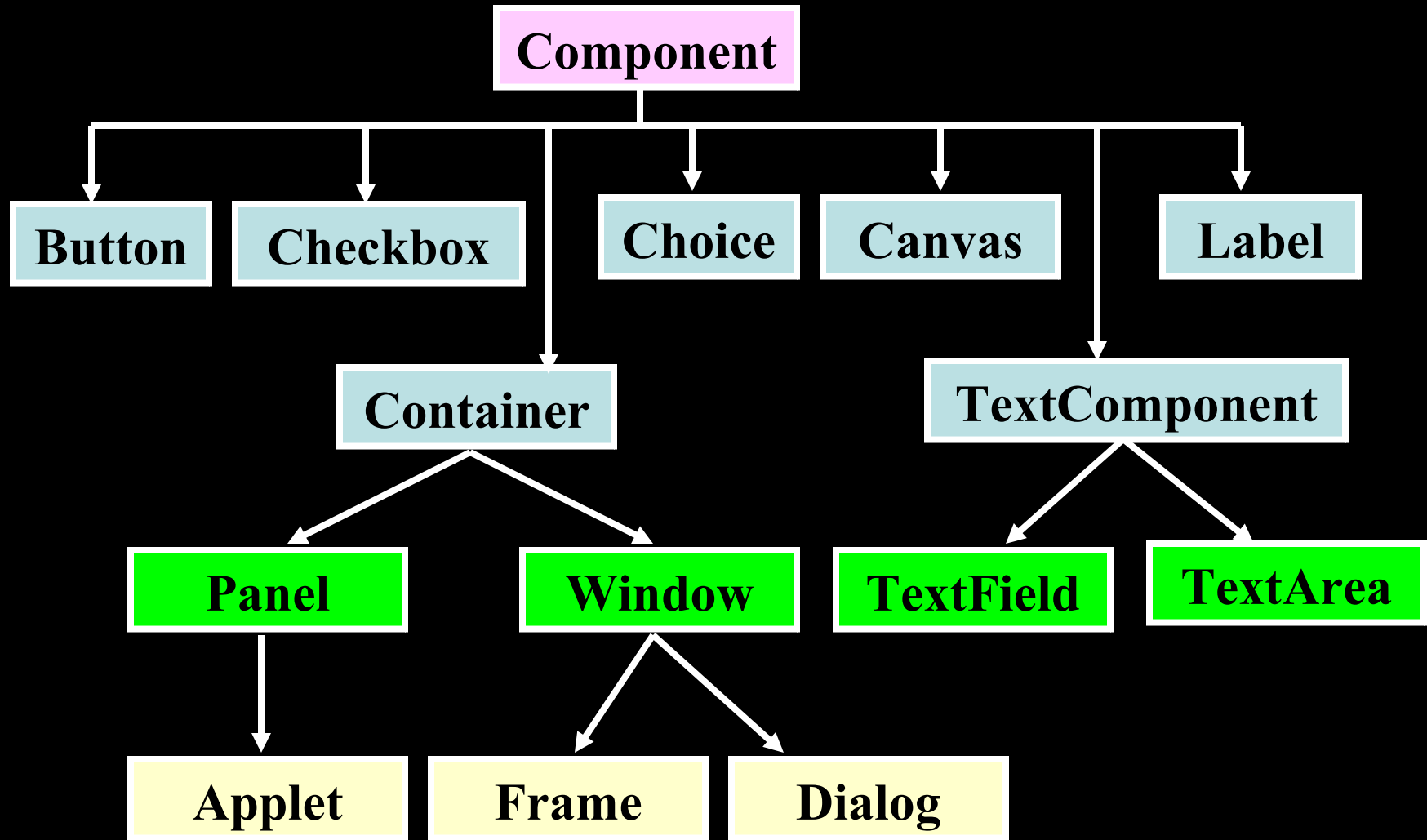
**Container**

**Components**

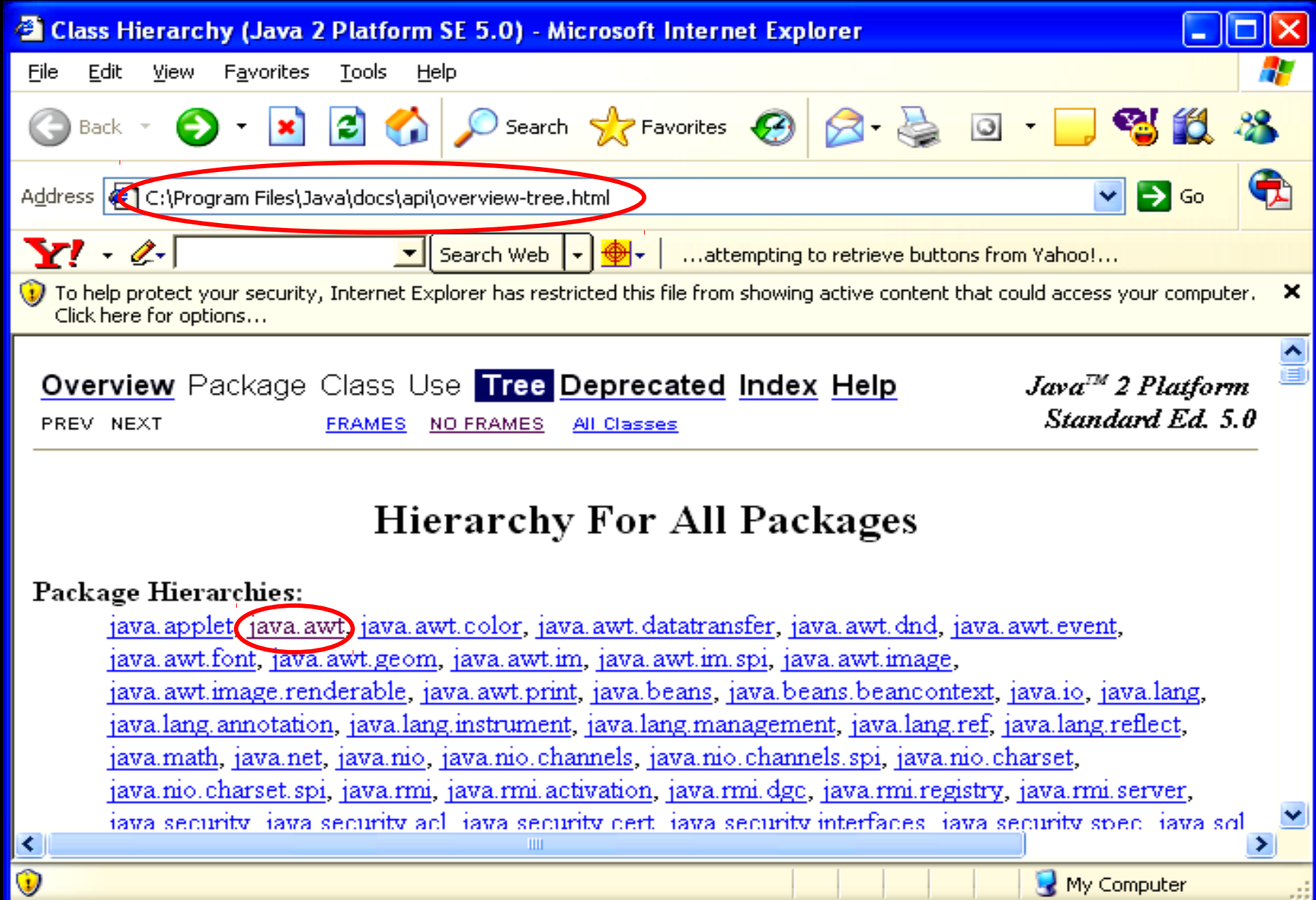
## 5.3- Gói AWT của Java

- AWT : abstract windowing toolkit - bộ công cụ chứa các lớp để tạo cửa sổ.
- AWT là 1 phần của JFC- Java Foundation Classes.
- Sử dụng: `import java.awt.*;`
- Gồm nhiều phần tử (class) để tạo GUI.
- Có các lớp quản lý việc bố trí các phần tử.
- Có (event-oriented application) mô hình ứng dụng hướng sự kiện.
- Có các công cụ xử lý đồ họa và hình ảnh.
- Các lớp sử dụng các tác vụ với clipboard (vùng nhớ đệm) như cut, paste.

# Cấu trúc gói AWT



# Tham khảo gói java.awt



# Tham khảo gói java.awt

The screenshot shows a Microsoft Internet Explorer window titled "java.awt Class Hierarchy (Java 2 Platform SE 5.0) - Microsoft Internet Explorer". The address bar displays the URL "C:\Program Files\Java\docs\api\java\awt\package-tree.html". A security warning from Internet Explorer is visible, stating that active content is restricted to protect the user's security. The main content area is titled "Hierarchy For Package java.awt" and includes a link to "All Packages". Below this, the "Class Hierarchy" is listed as follows:

- java.lang.[Object](#)
  - javax.accessibility.[AccessibleContext](#)
    - java.awt.[Component.AccessibleAWTComponent](#) (implements javax.accessibility.[AccessibleComponent](#), java.io.[Serializable](#))
      - java.awt.[Button.AccessibleAWTButton](#) (implements javax.accessibility.[AccessibleAction](#), javax.accessibility.[AccessibleValue](#))

The status bar at the bottom shows "Done" and "My Computer".



# Yêu cầu của GUI

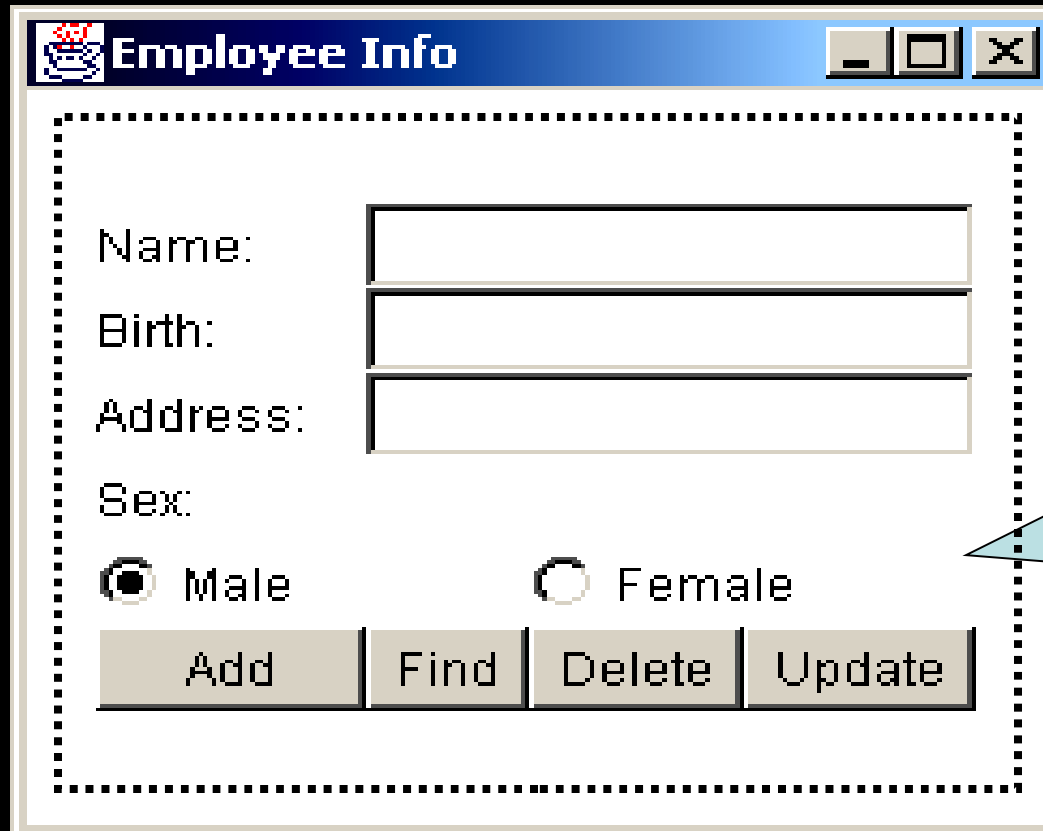
- Thân thiện với user.
- Số phần tử (element, component) trên GUI thay đổi tùy thuộc vào ứng dụng.
- Khi user tương tác với phần tử của GUI, ứng dụng phải có phản ứng.
- Lập trình sự kiện sẽ bàn đến trong chương sau.

## 5.4- Đưa 1 component vào GUI

Các bước để đưa 1 component vào GUI  
( viết code )

- Tạo 1 đối tượng component phù hợp.
- Xác định hình thức bên ngoài lúc đầu của component.
- Định vị component này trên GUI.
- Thêm component này vào GUI.

## 5.5- Một thí dụ



The image shows a graphical user interface window titled "Employee Info". The window has a blue title bar with standard minimize, maximize, and close buttons. Inside the window, there is a dashed rectangular border containing the following elements: three text labels ("Name:", "Birth:", "Address:") aligned to the left, each followed by a text input field; a "Sex:" label followed by two radio buttons labeled "Male" and "Female"; and a row of four buttons labeled "Add", "Find", "Delete", and "Update" at the bottom.

**Container**

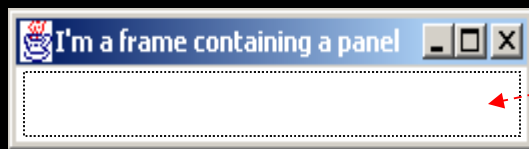
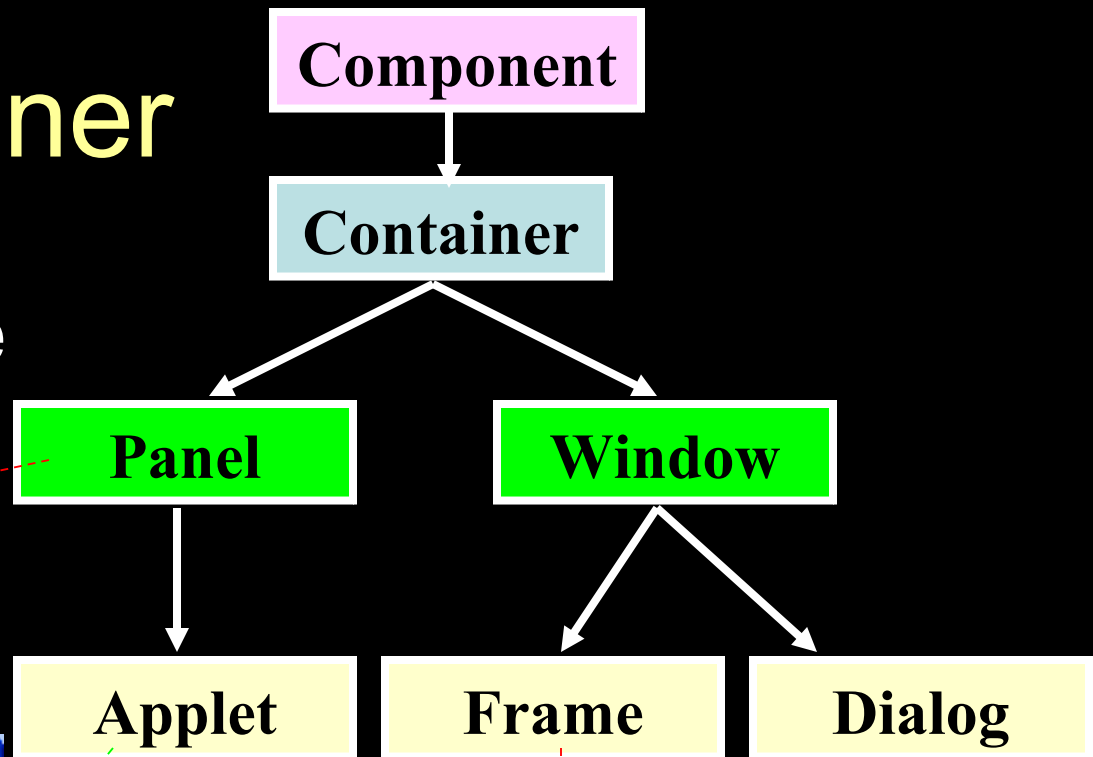
### Components

- 3 label,
- 3 text-field
- 1 checkboxgroup chứa 2 check-box
- 4 button

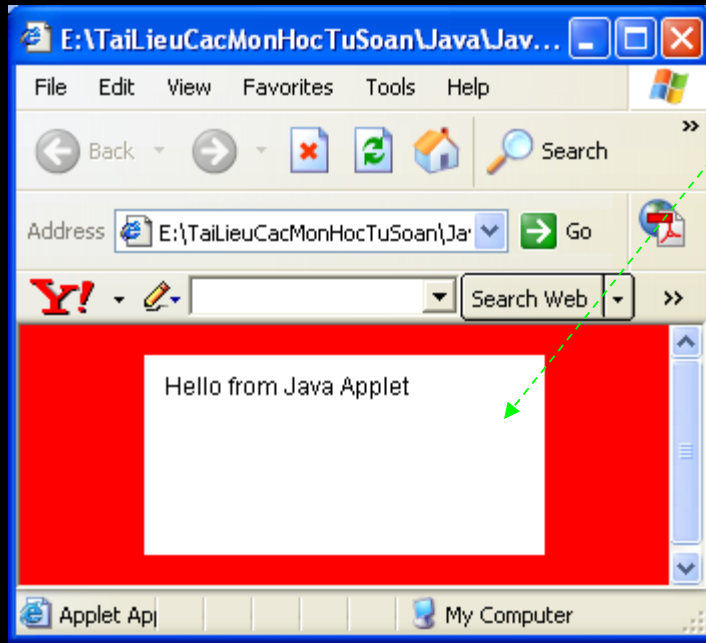
## 5.6- Sử dụng các lớp của awt

# 5.6.1- Container

## Phân cấp thừa kế



Panel là 1 vùng chữ nhật, không có đường viền



Panel là 1 khung chữ nhật, có đường viền, có các nút điều khiển cửa sổ

# Container...

- Container: Đối tượng chứa các element, cho phép vẽ, tô màu lên container.
- Frame và Panel là các class thường dùng.
- Panel thường dùng để chứa các element trong 1 GUI phức tạp, 1 Frame có thể chứa nhiều Panel.
- Panel, Applet thường dùng để tạo 1 ứng dụng nhúng vào Browser.

## 5.6.2- Frame

java.awt

### Class Frame

java.lang.Object

|

+--java.awt.Component

|

+--java.awt.Container

|

+--java.awt.Window

|

+--java.awt.Frame

Constructors:

**Frame()** → Make invisible frame

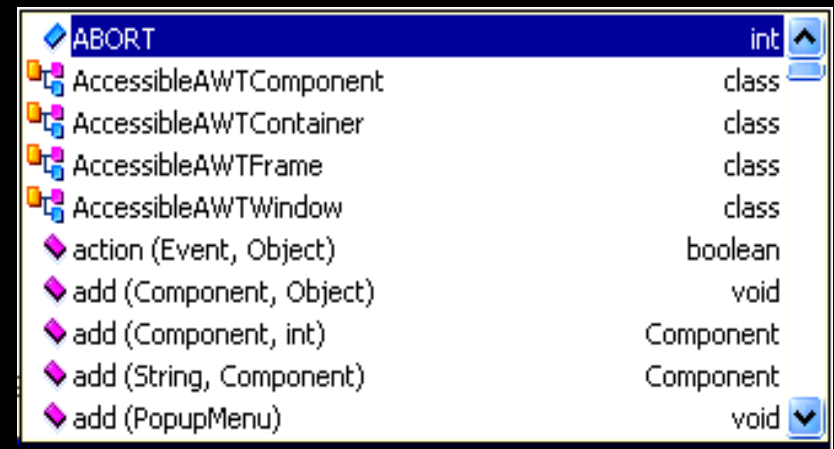
**Frame(String)** → Make a visible frame with title

# Frame...

## Common methods

```
void SetSize( int width, int Height)
public String getTitle();
public void setTitle(String title);
public void setResizable(boolean resizable)
public boolean isResizable()
public void setVisible(boolean)
public boolean isShowing()
void show(boolean)
void add (...) // add component
...
```

**Click for Demo**





## 5.6.3- Panel

Panel phải được đưa vào Frame khi viết application vì Frame mới có border

### Constructors

**Panel():** tạo 1 panel với bố cục mặc định.

**Panel(LayoutManager layout):** tạo 1 panel với bố cục đã biết.

### **Methods:**

**add(component)** // thêm 1 component vào panel

**setLayout(LayoutManager layout)** //chọn kiểu bố trí components

java.awt

### Class Panel

java.lang.Object

|

+--java.awt.Component

|

+--java.awt.Container

|

+--java.awt.Panel

[Click for Demo](#)

## 5.6.4- Label

- Nhãn nhằm giải thích, chứa dữ liệu chỉ xuất.

### Constructor:

**Label()** : tạo label trống

**Label(String)** : tạo label có chuỗi

**Label (String, int Align)** Tạo label có gióng hàng:  
*Align=LEFT,RIGHT,CENTER*

### Common Methods:

**void setFont (Font f)**

**void setText(String S)**

**String getText()**

## 5.6.5- TextField

- Chứa dữ liệu nhập 1 chuỗi ký tự.
- User chỉ được nhập 1 dòng.

### Constructors

**TextField()** : tạo text field trống

**TextField(int cols)** : tạo text field trống có n cột chữ

**TextField(String S)** : tạo text field có chuỗi S

**TextField (String S, int cols)** : text có chuỗi S, n cột

### Common Methods:

**void setEchoChar (char c)** - ấn định ký tự thể hiện (password)

**void setText(String S)**

**String getText()**

**void setEditable(boolean b)**

**boolean isEditable()**

## 5.6.6- TextArea

- Chứa dữ liệu nhập 1 chuỗi ký tự.
- User chỉ được nhập nhiều dòng.

### Constructors

TextArea()

TextArea(int cols, int rows)

TextArea(String S)

TextArea(String S, int rows, int cols)

TextArea(String,int cols, int rows, int Scrollbars)

### Common methods

void **setText**(String)

String **getText**()

void **setEditable**(boolean)

boolean **isEditable**()

void **insert**(String S, int Index)

void **replaceRange**(String S, int begin, int end)

## 5.6.7- Button

- Công cụ để user chọn 1 tác vụ.

### Constructors

**Button()** - tạo nút không có nhãn

**Button(String S)** - tạo nút có nhãn

### Common Methods

**void setLabel(String)** - đổi nhãn

**String getLabel()** - lấy nội dung nhãn

## 5.6.8- Checkbox/CheckboxGroup

- **Công cụ nhập yes/no**
- **Checkbox: multi-option (cho phép chọn nhiều) thể hiện dạng hộp vuông.**
- **CheckboxGroup chứa nhiều checkbox nhưng chỉ cho phép chọn 1/n. Phần tử trong CheckboxGroup là đối tượng thuộc lớp Checkbox nhưng lại thể hiện dạng nút tròn (radio button)**

# Checkbox/CheckboxGroup...

## Checkbox constructors

**Checkbox()**            Tạo checkbox không nhãn

**Checkbox(String)**    Tạo text box có nhãn

**Tạo 1 checkbox có nhãn, có ấn định trị chọn lựa đưa vào 1 nhóm trong 1 nhóm**

**Checkbox(String label , boolean state, CheckboxGroup group)**

**Checkbox(String label, CheckboxGroup group, boolean state)**

## Common methods

void **setLabel**(String);            String **getLabel**();

void **setState**(boolean);        boolean **getState**();

## 5.6.9- List

- Công cụ nhập bằng cách chọn trong 1 danh sách chuỗi ký tự.
- Có thể chọn 1 hoặc nhiều.

### Constructor

Choice() – tạo 1 danh sách trống

### Common methods

void **add**(String) ;      void **addItem**(String);

void **insert**(String item, int index)

int **CountItems**(); int **getItemCount**(); - lấy số phần tử

String **getItem**(int Index);

int **getSelectedIndex**();

String **getSelectedItem**();

void **remove**(int position)

void **removeAll**()

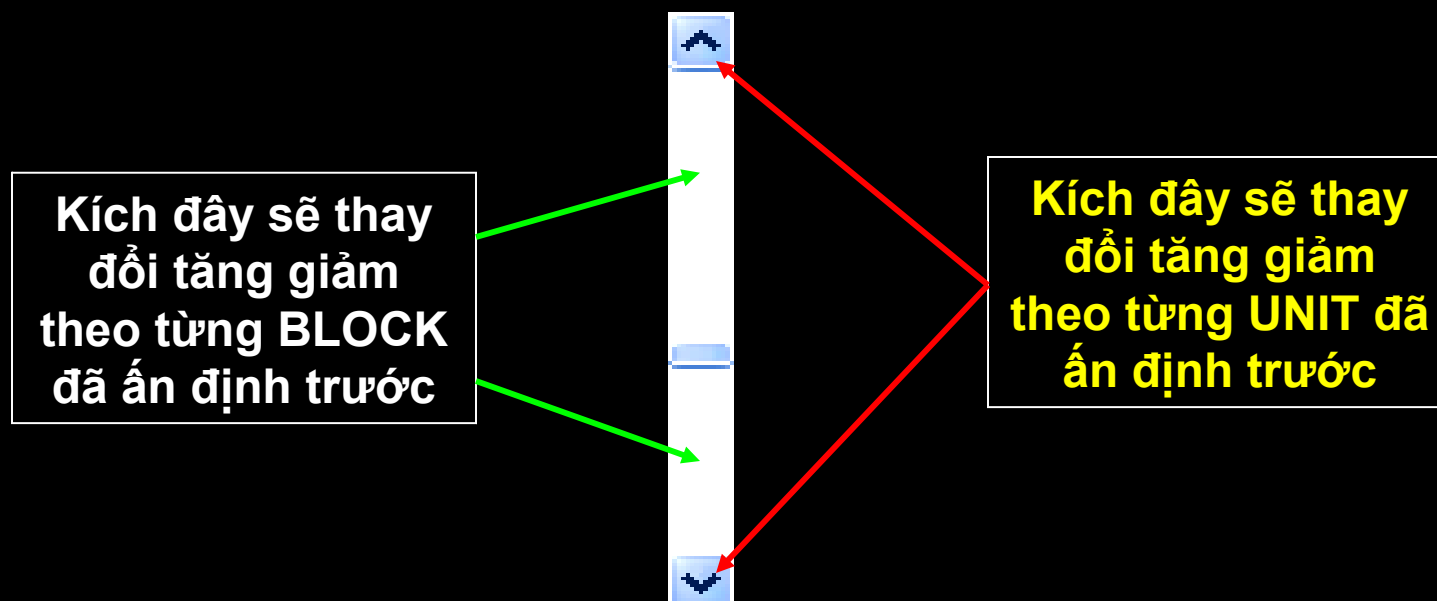
void **select**(int pos) – áp đặt 1 mục chọn theo vị trí

void **select**(String str) – áp đặt chọn 1 mục chọn theo nội dung



## 5.6.10- Scrollbar- Thanh cuộn

- Công cụ nhập 1 trị trong 1 khoảng số ( biểu diễn bằng Maximum, Minimum) bằng cách kéo con trượt.
- Tại 1 thời điểm, con trượt ở tại vị trí mô tả cho trị hiện hành (Value)
- Có thể có hướng ngang hoặc dọc (Orientation)



# Scrollbar...

## Constructors

Scrollbar() - tạo thanh cuộn dọc

Scrollbar(int orientation) // VERTICAL|HORIZONTAL

Scrollbar(int orientation, int value, int visible, int minimum, int maximum)

## Common methods

void **setMaximum**(int v)

void **setMinimum**(int v)

int **getOrientation**()

void **setUnitIncrement**(int v)

void **setBlockIncrement**(int v)

void **setValue**(int v)

void **setVisibleAmount**(int newAmount)

int **getVisibleAmount**()

int **getMaximum**() ;

int **getMinimum**()

int **getUnitIncrement**()

int **getBlockIncrement**()

int **getValue**()

# Minh họa

**Click for Demonstration**

## 5.7- Bố trí các components lên GUI

- Layout : Cách bố trí các components lên container.
- Không dễ dàng gì để tự quản lý vị trí của các components trên GUI.
- LayoutManager là interface mô tả về các layout.
- Trong gói AWT có hiện thức sẵn một số layout, các lớp layout này đều implement LayoutManager interface.

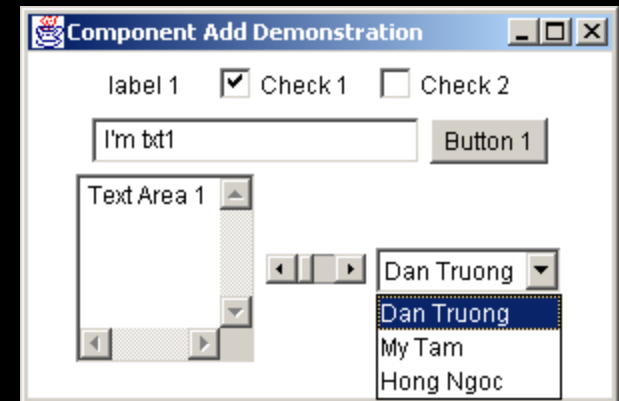
## 5.7.1-Layouts có sẵn trong AWT

- `java.awt.FlowLayout` (bố trí dạng dòng chảy)
- `java.awt.BorderLayout` (bố trí về biên khung)
- `java.awt.GridLayout` (bố trí dạng lưới đều nhau)
- `java.awt.GridBagLayout` (bố trí dạng lưới không đều)
- `java.awt.CardLayout` (bố trí dạng lưng quân bài)
- **Tham khảo**

**`docs\api\java\awt\package-tree.html`**

vớ docs là thư mục Documentation của Java 2

## 5.7.2- FlowLayout



- Bố trí các component theo dạng chảy xuôi theo thứ tự mà phần tử này được add vào container.
- Đây là layout mặc định của Panel.
- Nếu có nhiều component trên container → Các component có thể được đặt trên nhiều dòng → Vấn đề giống hàng (Align)
- Giữa các component, chúng hõ nhau theo chiều dọc (vgap) bao nhiêu, theo chiều ngang (hgap) bao nhiêu?

# FlowLayout...

## Constructors

### FlowLayout()

Tạo FlowLayout mặc định: align= center, vgap=hgap=5 unit.

### FlowLayout(int align)

Tạo FlowLayout với align đã biết, vgap=hgap=5 unit (default).

### FlowLayout(int align, int hgap, int vgap)

Tạo FlowLayout với 3 tham số

## Trị của align: các dữ liệu static của class FlowLayout

LEFT CENTER RIGHT

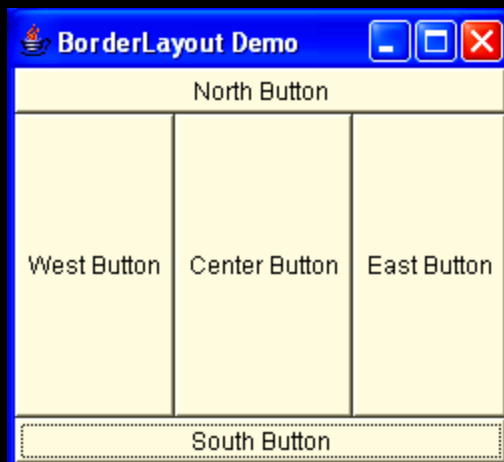
LEADING (phía đầu, tương tự LEFT)

TRAILING (phía đuôi, tương tự RIGHT)

[Click for Demo](#)

## 5.7.3- BorderLayout

- Bố trí các component theo dạng ra biên của khung tạo ra 5 vị trí: EAST, WEST, SOUTH, NORTH, CENTER.
- Đây là layout MẶC ĐỊNH của Frame.
- Nếu container chỉ có 1 component và đặt nó ở vị trí CENTER thì component này phủ kín container.
- Cú pháp thêm 1 component vào container tại 1 vị trí:  
`Container.add("East", componentName);` // hoặc  
`Container.add(BorderLayout.EAST, componentName);`
- Tương tự cho “West”, “South”, “North”, “Center”




**Click for Demo**



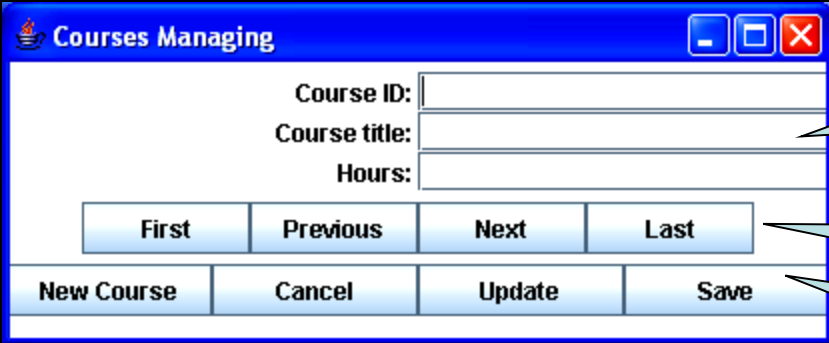
## 5.7.4- GridLayout

- Bố trí các component thành 1 lưới rows dòng, cols cột đều nhau.

Lưới 4x4



13	11	8	7
4	5	6	2
3	9	1	10
12		14	15



Lưới 3x2

Lưới 1x4

Lưới 1x4

# GridLayout...

## Constructor

### **GridLayout()**

Tạo grid layout mặc định 1x1

### **GridLayout(int rows, int cols)**

Tạo grid layout rows x cols

### **.GridLayout(int rows, int cols, int hgap, int vgap)**

[Click for Demo](#)

## 5.7.5- GridBagLayout

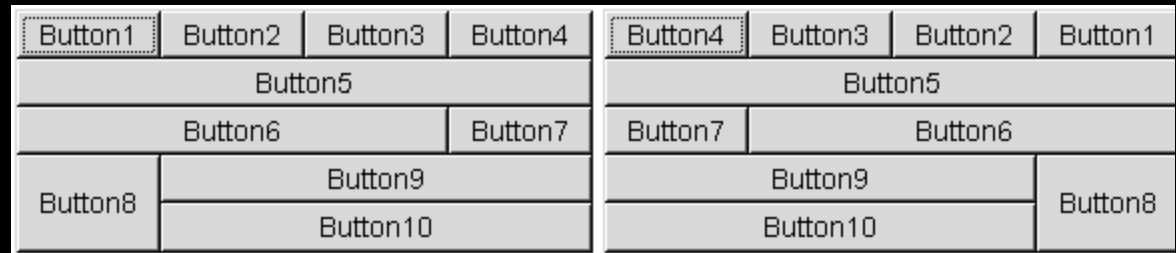
- Layout dạng lưới cho phép 1 component chiếm 1 số ô kề nhau theo cả 2 chiều.
- Hình Employee Info sau là GridBagLayout 6x4, các label bên trái chiếm 1 ô, các textbox chiếm 3 ô ngang. Dòng “Sex” chiếm 4 ô ngang, 2 checkbox chiếm 2 ô ngang.



The screenshot shows a window titled "Employee Info" with a blue title bar. The window contains a form with the following elements:

- Name: [text box]
- Birth: [text box]
- Address: [text box]
- Sex: ☒ Male ☐ Female
- Buttons: Add, Find, Delete, Update

The layout is a 6x4 grid. The first three rows are for Name, Birth, and Address. The fourth row is for Sex. The fifth row contains the buttons. The sixth row is empty.



**Các thí dụ khác**

# Class java.awt.GridBagLayout

- **Constructors:**    `GridBagLayout()`
- **Áp đặt GridBagLayout cho 1 container:**

```
GridBagLayout gb= new GridBagLayout();
```

```
FrameName.setLayout(gb);
```

```
PanelName.setLayout(gb);
```

**Viết ngắn gọn:**

```
FrameName.setLayout(new GridBagLayout());
```

```
PanelName.setLayout(new GridBagLayout());
```

# class GridBagConstraints

- Làm sao để có thể đưa 1 component vào 1 vị trí nhưng trải dài trên nhiều ô kề nhau?
- 1 component vào 1 vị trí nhưng trải dài trên nhiều ô kề nhau là 1 sự “ràng buộc” 1 component vào các ô này. Một đối tượng thuộc lớp **GridBagConstraints** sẽ đảm nhiệm việc này.

# class GridBagConstraints...

- **Properties – Đa số là static data**

**int gridx, gridy** : ô sẽ đặt component vào

**int gridwidth, gridheight** : số ô sẽ phủ theo 2 chiều khi thêm 1 component vào ô <gridx,gridy>

**double weightx, weighty** : Khoảng hở của lưới, mặc định là 0.

**int anchor** : Vị trí đặt component, mặc định là CENTER, các static int được khai báo sẵn: GridBagConstraints.NORTH, EAST, WEST, SOUTH, NORTHEAST, SOUTHEAST, NORTHWEST, SOUTHWEST.

**int fill**: Xác định kiểu đặt khi component có kích thước lớn hơn ô sẽ được đặt vào. Các hằng được dùng: GridBagConstraints.NONE, HORIZONTAL, VERTICAL, BOTH.

**Insets insets** : Đặc tả khoảng hở <top, bottom, left, right> giữa các phần tử được đưa vào, mặc định là 0.

**int ipadx, ipady**: Khoảng đệm (số pixel trống) bên trong của phần tử theo 2 chiều. Mặc định là 0. Khi vẽ phần tử, sẽ thêm 2\*ipadx và 2\*ipady vào chiều rộng tối thiểu và chiều cao tối thiểu của phần tử.

# class GridBagConstraints...

- **Constructor:**

`GridBagConstraints()` // tạo object với các dữ liệu mặc định.

`GridBagConstraints(int gridx, int gridy, int gridwidth, int gridheight, double weightx, double weighty, int anchor, int fill, Insets insets, int ipadx, int ipady)`

- Thao tác với `GridBagConstraints` object bằng static data

- **Làm sao kết hợp `GridBagConstraints` với `GridBagLayout`?**

`GridBagLayout gbLayout = new GridBagLayout();`

`GridBagConstraints gbc = new GridBagConstraints();`

`gbLayout.setConstraints(gbc);`

# class GridBagConstraints...

Method sau đây sẽ thêm component c vào vị trí ô (row,col) kéo tràn dọc nrow ô và tràn ngang ncol ô. Trong đó gbc là GridBagConstraints và gb là GridBagLayout

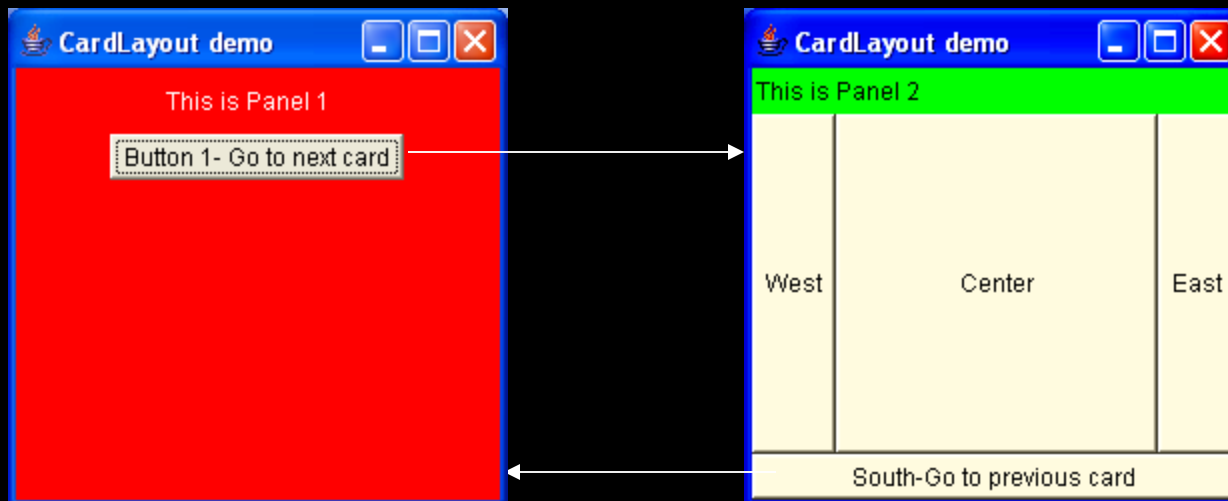
```
void addComponent(Component c,int row, int col,  
                                     int nrow, int ncol)  
{ gbc.gridx= col;   gbc.gridy=row;    // định vị ô sẽ thao tác  
  gbc.gridwidth=ncol; gbc.gridheight=nrow; // định vùng tràn  
  // ràng buộc c vào lưới thông qua gbc  
  gb.setConstraints(c,gbc);  
  add(c);  
  
}
```

[Click for Demo](#)



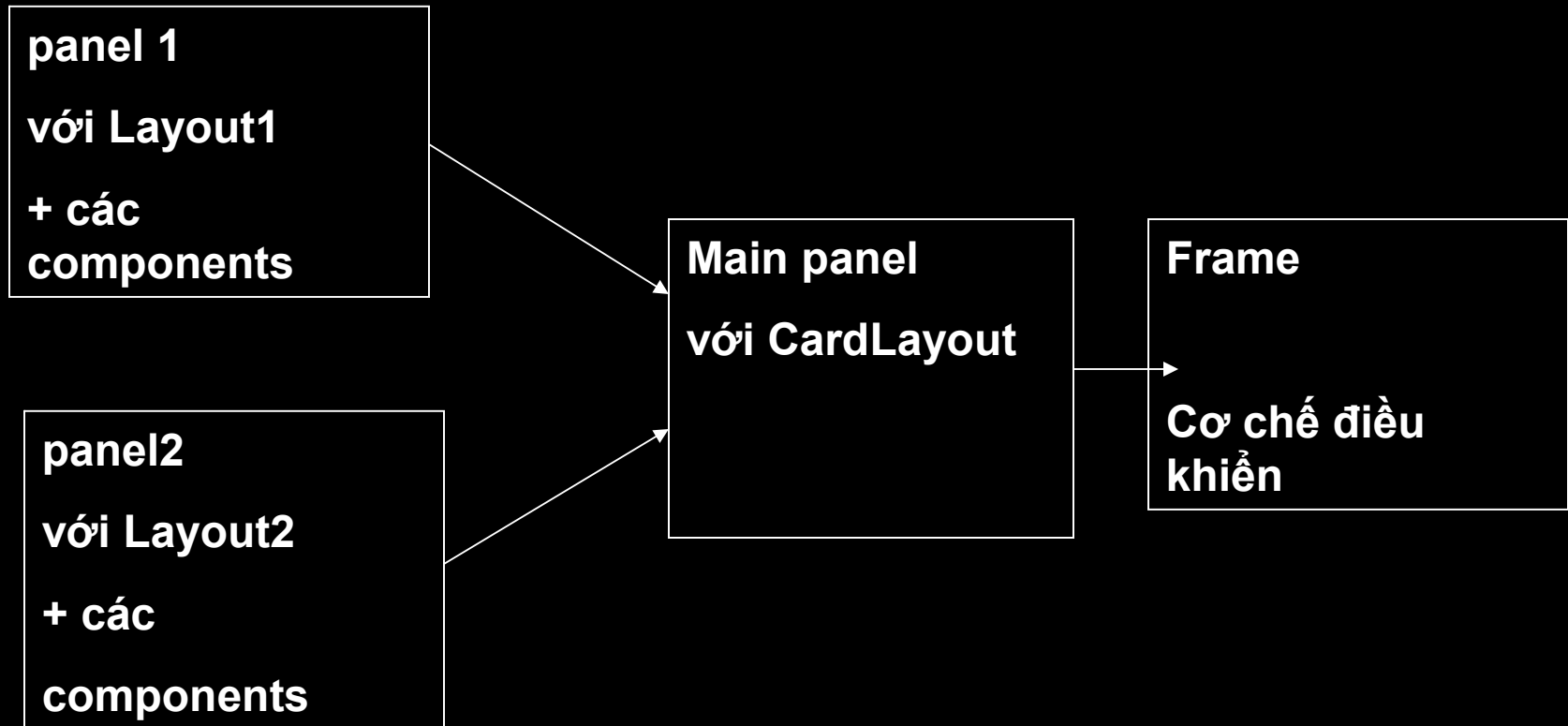
## 5.7.6- CardLayout

- Bố trí các component thành từng lớp như lưng các quân bài (card).
- Thường dùng Panel để chứa các component.
- Tại 1 thời điểm chỉ có 1 panel hiện hành (quân bài trên cùng).
- Có thể chuyển qua lại giữa các Panel.



# CardLayout...

- Cách tạo GUI với card layout



# CardLayout...

- **CardLayout Constructors:**

**CardLayout()**

**CardLayout(int hgap, int vgap)**

- **Đưa 1 panel con vào panel cha**

FatherPanel.add (sonPanel);

FatherPanel.add (“Alias”,sonPanel);

- **Chọn 1 panel sẽ hiển thị**

Card.first(FatherPanel);

Card.last(FatherPanel);

Card.next(FatherPanel);

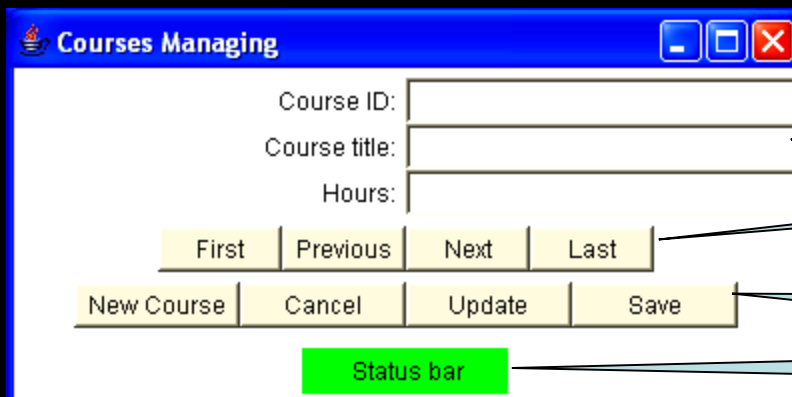
Card.previous(FatherPanel);

Card.show( FatherPanel, “Alias\_of\_sonPanel”);

[Click for Demo](#)

## 5.7.7- Layout phức tạp

- Có thể phải kết hợp nhiều Layout trên 1 GUI.
- Chia GUI thành nhiều Panel, mỗi panel 1 Layout riêng.



Panel p1, lưới 3x2

Panel p2, lưới 1x4

Panel p3, lưới 1x4

Panel p4

[Click for Demo](#)

## 5.8- Hướng dẫn tạo GUI cho ứng dụng

- GUI là khuynh hướng của các ứng dụng hiện nay. Nhờ GUI, giao diện với người sử dụng đẹp hơn và có được cơ hội kiểm tra dữ liệu nhập trước khi chuyển giao vào biến. Ta nói rằng “**tách biệt cơ chế điều khiển của chương trình và dữ liệu**”).
- User  $\leftrightarrow$  GUI  $\leftrightarrow$  Biến

# Hướng dẫn tạo GUI: Chọn components

- **Bài toán**

→ Dữ liệu nhập/xuất → Chọn component cho việc nhập/xuất

→ Các tác vụ → Mỗi tác vụ là 1 nút lệnh

Dữ liệu	Đối tượng
Chuỗi nhập 1 dòng	TextField
Chuỗi nhiều dòng	TextArea
Chọn trong nhiều chuỗi	Choice
Chọn Yes/No (nhiều)	Checkbox
Chọn Yes/No (1/n)	CheckboxGroup + Checkbox
Dữ liệu chỉ xuất 1 dòng	Label, TextField-cấm nhập
Dữ liệu chỉ xuất nhiều dòng	TextArea – cấm nhập
Chuỗi nhập + xuất	TextField/TextArea

# Hướng dẫn tạo GUI: Chọn Layout

- **Các cơ sở chọn Layout**

- Thân thiện :Ưu tiên tạo Layout giống mẫu hồ sơ mà user thường dùng.
- Trật tự nhập liệu tự nhiên của bài toán.
- Nếu GUI phức tạp thì phân bổ các component vào nhiều panel, mỗi panel có một layout khác nhau.

- **Thói quen tốt khi đặt tên đối tượng**

- Dùng tiếp đầu ngữ  
txt cho **TextField**,  
lbl cho **Label**,  
chk cho **Checkbox**,  
btn cho **Button**, ...

# Một thí dụ: -Phân tích bài toán

## Bài toán:

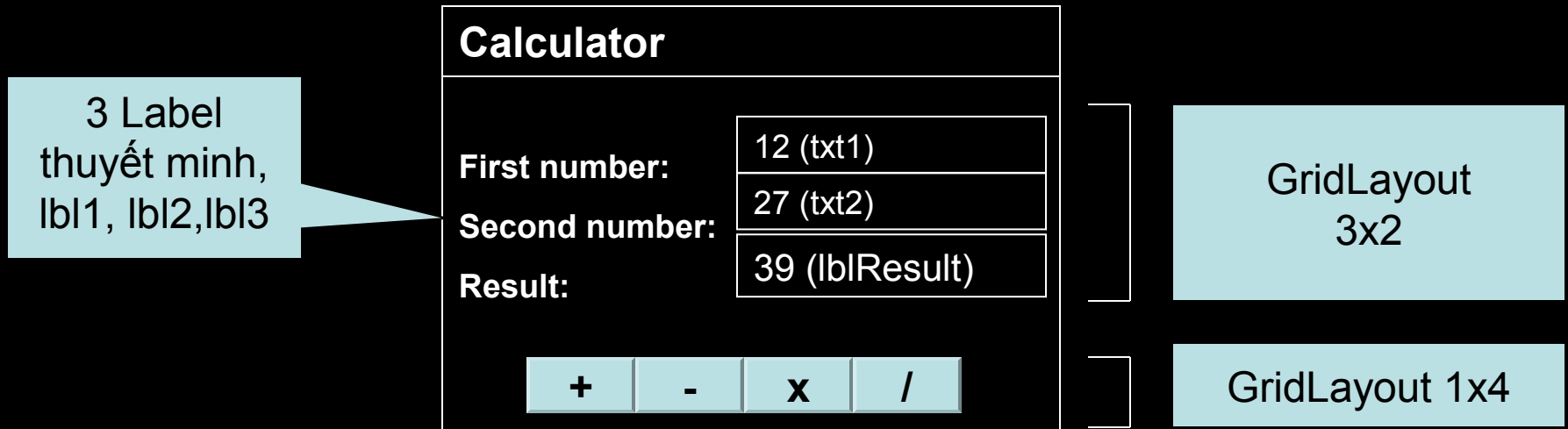
Xây dựng ứng dụng cho phép làm các phép tính  $+$ ,  $-$ ,  $*$ ,  $/$

- Dữ liệu nhập: 2 số  $\rightarrow$  2 TextFiled, tên txt1, txt2.
- Dữ liệu xuất: Kết quả của phép tính: Label, tên lblResult
- 4 tác vụ: Cộng, trừ, nhân, chia  $\rightarrow$  4 Button, tên btnAdd, btnSub, btnMul, btnDiv
- 3 lời thuyết minh: Label, tên lbl1, lbl2, lbl3

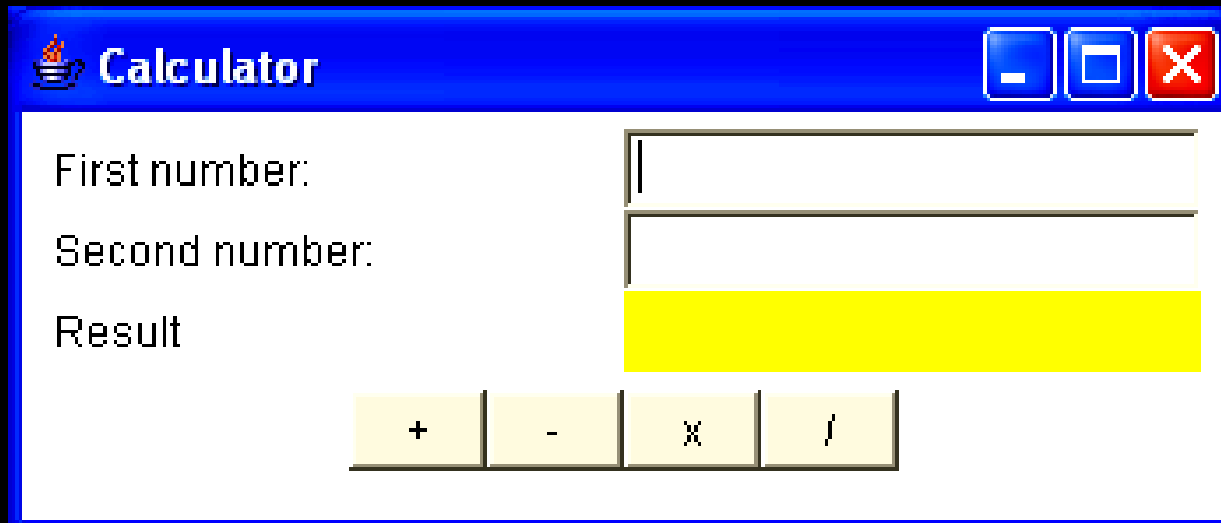


# Một thí dụ: THIẾT KẾ GUI

- Hình thức GUI



# Kết quả



A screenshot of a Java Swing window titled "Calculator" with a blue title bar. The window contains three text input fields labeled "First number:", "Second number:", and "Result". The "Result" field is highlighted in yellow. Below the fields is a row of four buttons: "+", "-", "x", and "/".

[Click for Demo](#)

## 5.9- Tóm tắt

- GUI- Graphic User Interface.
- GUI là khuynh hướng của các ứng dụng hiện nay.
- Nhờ GUI, người lập trình tách được điều khiển của chương trình và dữ liệu của chương trình.
- Gói AWT của Java bao gồm một tập các lớp cho phép người lập trình tạo ra GUI của ứng dụng.
- Một component đặt trên GUI có thể được user nhìn thấy (visible) hoặc không nhìn thấy (invisible) và có thể được thay đổi kích thước (resize)
- Frame và Panel là các container thường được dùng để tạo ra ứng dụng chạy độc lập (stand-alone application).
- Một panel thường được dùng để nhóm một số components lại với nhau.

# Tóm tắt...

- Các lớp Layout manager giúp bố trí các component lên GUI.
- Trong gói awt, có 5 loại layout khác nhau: FlowLayout, BorderLayout, GridLayout, GridBagLayout, CardLayout.
- Thiết lập layout cho 1 container bằng hành vi `setLayout(aLayout)` của lớp container này.
- FlowLayout là bố cục mặc định của Panel, Applet. Các component được thêm vào tuần tự từ trên xuống dưới, từ trái sang phải.
- BorderLayout là bố cục mặc định của Frame. Cửa sổ sẽ được chia thành các phần: “East”, “West”, “South”, “North”, “Center”.
- GridLayout là cách bố trí các component vào container dạng 1 lưới rows hàng cols cột. Các component cũng được đưa vào các ô theo thứ tự: trên → dưới, trái → phải. Các component sẽ có cùng kích thước.

# Tóm tắt...

- GridBagLayout là cách bố trí cho phép 1 component trải rộng trên nhiều ô kề nhau. Các component có thể có kích thước khác nhau.
- Để dùng GridBagLayout, cần có sự kết hợp của GridBagConstraints để “ràng buộc” 1 component vào lưới.
- CardLayout là cách bố trí các component trên 1 số Panel. Các panel hình thành 1 chồng (stack). Tại 1 thời điểm chỉ có 1 panel hiện hành.
- Một GUI có bố cục phức tạp có thể được phân tích thành nhiều thành phần, mỗi thành phần là 1 panel có layout riêng.
- BorderLayout là bố cục mặc định của Frame. Cửa sổ sẽ được chia thành các phần: “East”, “West”, “South”, “North”, “Center”.
- GridLayout là cách bố trí các component vào container dạng 1 lưới rows hàng cols cột. Các component cũng được đưa vào các ô theo thứ tự: trên → dưới, trái → phải.

# Tóm tắt...

- Trình tự thiết kế GUI:

(1) Đọc kỹ yêu cầu.

(2) Xác định dữ liệu nhập/xuất, chọn đối tượng phù hợp.

(3) Xác định các tác vụ, mỗi tác vụ là 1 nút lệnh.

(4) Xác định các nội dung thuyết minh, mỗi lời thuyết minh là 1 label.

(5) Vẽ các đối tượng lên giấy → Hình dáng GUI.

(6) Phân tích hình dáng GUI để xác định số nhóm panel.