NNLT JAVA

GV: Trần Thanh Tuấn

CHƯƠNG 7

LẬP TRÌNH GIAO DIỆN

NỘI DUNG

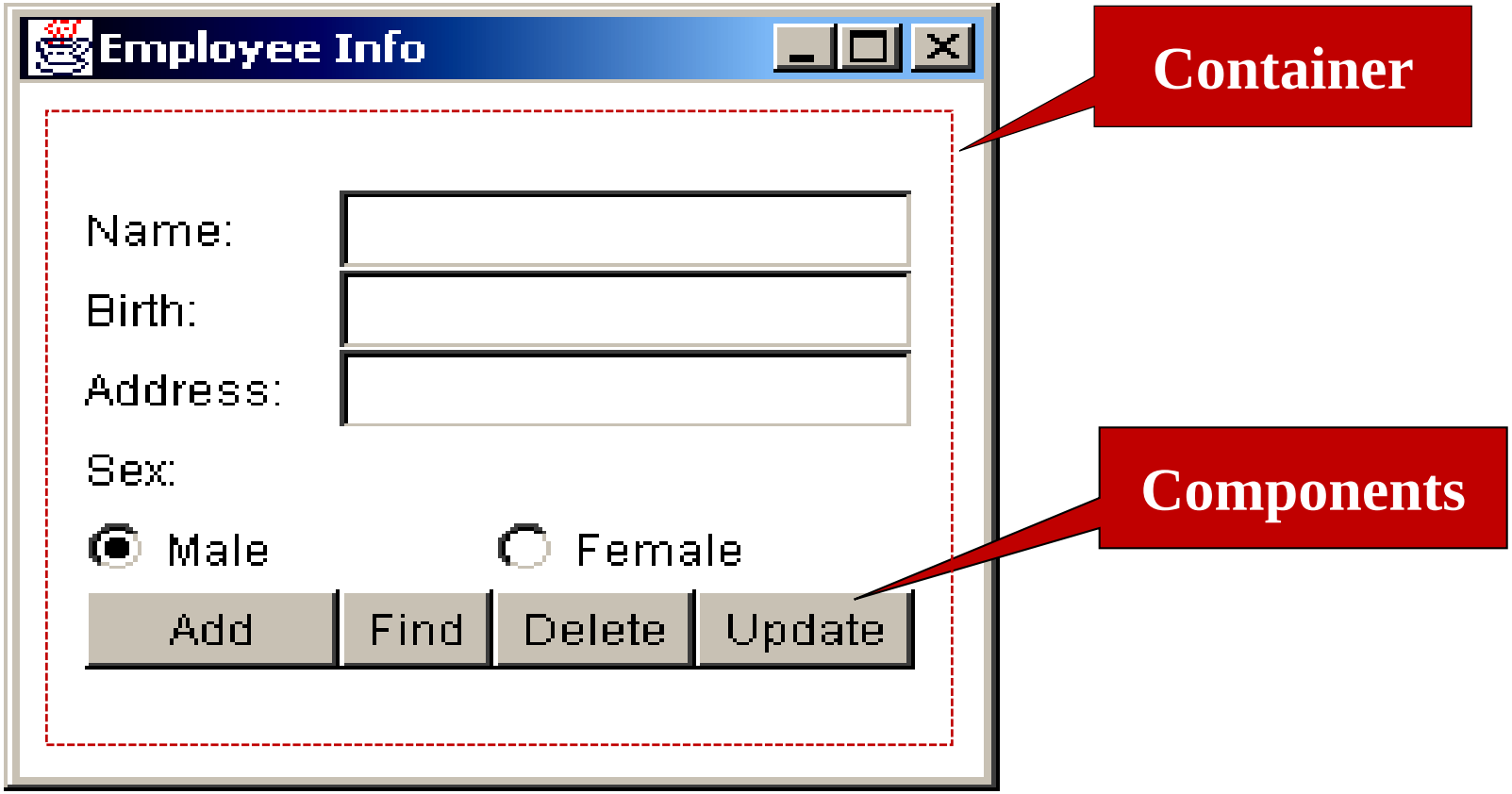
- Giới thiệu về lập trình giao diện (GUI)
- Thư viện AWT
- Container
- Components
- Layout Manager
- Xây dựng ứng dụng GUI
- Lập trình xử lý sự kiện
- MENU
- Thư viện SWING

GIỚI THIỆU LẬP TRÌNH GUI

- **G**raphic **U**ser **I**nterface – mô hình giao tiếp kiểu tương tác giữa ứng dụng và người dùng dạng đồ họa.
- Hầu hết các ngôn ngữ lập trình hiện nay được cung cấp các đối tượng đồ họa
- Mỗi ngôn ngữ hỗ trợ cách tạo GUI khác nhau:
  - VB, VC++, VC# dùng dạng kéo thả (drag and drop)
  - C++ đòi hỏi lập trình viên viết toàn bộ code để tạo GUI
  - Java hỗ trợ sẵn các lớp tạo GUI cho lập trình viên sử dụng

GIỚI THIỆU LẬP TRÌNH GUI

- GUI = Container + Components



THƯ VIỆN AWT

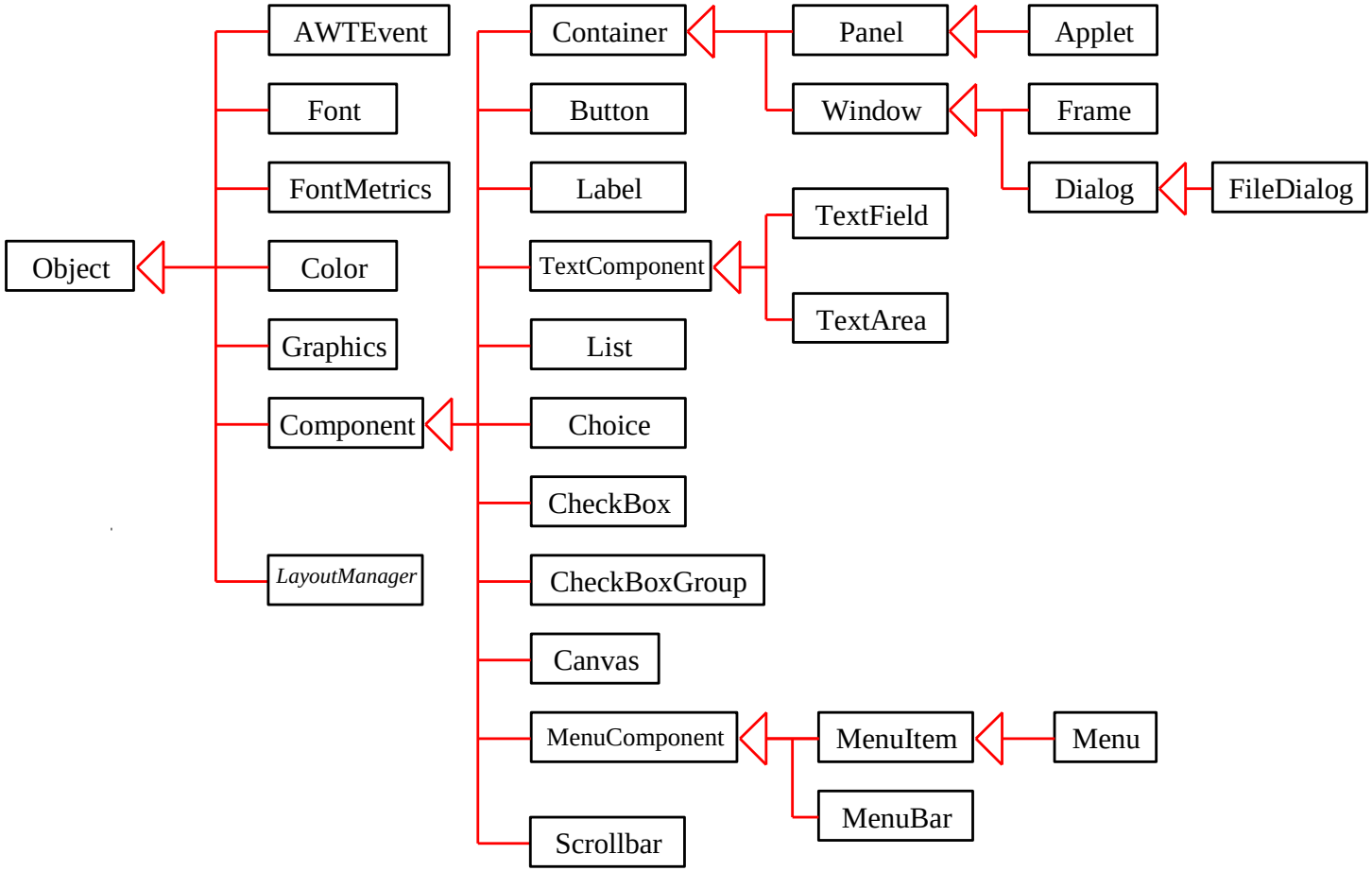
- AWT: **A**bstract **W**indowing **T**oolkit
- AWT là một bộ các lớp trong Java cho phép chúng ta tạo một GUI và chấp nhận các nhập liệu của người dùng thông qua bàn phím và chuột.
- AWT là thư viện nền tảng, cơ sở giúp cho chúng ta tiếp cận với thư viện mở rộng JFC hiệu quả hơn.
- Sử dụng:
  - import java.awt.*;
  - import java.awt.event.*;

THƯ VIỆN AWT

- AWT cung cấp các item khác nhau để tạo một GUI hiệu quả và lôi cuốn người sử dụng. Các item này có thể là:
  - Container
  - Component
  - Trình quản lý cách trình bày (Layout manager)
  - Đồ họa (Graphic) và các tính năng vẽ (Draw)
  - Phông chữ (Font)
  - Sự kiện (Event)

THƯ VIỆN AWT

❑Cấu trúc thư viện AWT



CONTAINER

- Container có khả năng quản lý và nhóm các đối tượng khác lại
  - Những đối tượng con thuộc thành phần awt như: button, checkbox, choice, scrollbar,… chỉ sử dụng được khi ta đưa nó vào khung chứa (container).
- Gói java.awt chứa một lớp có tên là Container. Lớp này trực tiếp hay gián tiếp phát sinh ra hai container được sử dụng phổ biến nhất là Frame và Panel

CONTAINER

- Frame (java.awt.Frame)
  - Khung chứa Frame là một cửa sổ window
    - Là lớp con của Window (java.awt.Window)
    - Bao gồm một tiêu đề (title) và một đường biên (border) như các ứng dụng windows thông thường khác
    - Thường được sử dụng để tạo cửa sổ chính của các ứng dụng

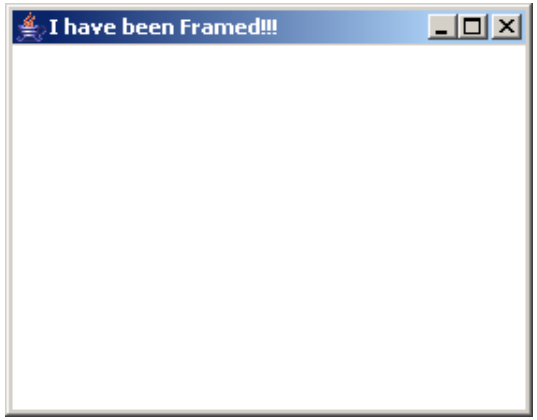
CONTAINER

- Frame (java.awt.Frame)
  - Frame có thể hoạt động như một container hay như một thành phần (component)
  - Chúng ta có thể sử dụng một trong những constructor sau để tạo một frame:
    - Frame();
    - Frame(String title);
    - ...
  - Các phương thức:
    - setSize(int width, int height);
    - setVisible(boolean visible);
    - ...

CONTAINER

Frame (java.awt.Frame)

Ví dụ tạo Frame




```
import java.awt.*;
class FrameDemo extends Frame{
        public FrameDemo(String title){
                super (title);
        }
        public static void main (String args[])     {
                FrameDemo ObjFr =
                        new FrameDemo("I have been Framed!!!");
                ObjFr.setSize(500,500);
                ObjFr.setVisible(true);
        }
}
```

CONTAINER

- Panel (java.awt.Panel)
  - Đối tượng khung chứa đơn giản nhất, dùng để nhóm các đối tượng, thành phần con lại
  - Một Panel có thể chứa bên trong một Panel khác
  - Một panel không có sẵn vì thế chúng ta cần phải thêm nó vào Frame
  - Hàm khởi tạo:
    - Panel ( );
    - Panel (LayoutManager);

Panel (java.awt.Panel)

Ví dụ tạo Panel

```
import java.awt.*;
class PanelTest extends Panel{
        public static void main(String args[])
        {
                PanelTest p = new PanelTest();
                Frame f = new Frame("Testing a Panel");
                f.add(p);
                f.setSize(300,200);
                f.setVisible(true);
        }
        public PanelTest()
        {
        }
}
```

CONTAINER

- Dialog (java.awt.Dialog)
  - Là một lớp con của lớp Window (java.awt.Window)
  - Là một cửa sổ dạng hộp hội thoại, cửa sổ dạng này thường được dùng để đưa ra thông báo, hay dùng để lấy dữ liệu nhập từ ngoài vào thông qua các đối tượng, thành phần trên dialog như TextField…
  - Dialog cũng là một cửa sổ nhưng không đầy đủ chức năng như đối tượng khung chứa Frame

- Dialog (java.awt.Dialog)
  - Ví dụ tạo Dialog

```
Frame myframe = new Frame(“My frame”);
String title = “Title”;
boolean modal = true;
Dialog dlg = new Dialog(myframe, title, modal);
```

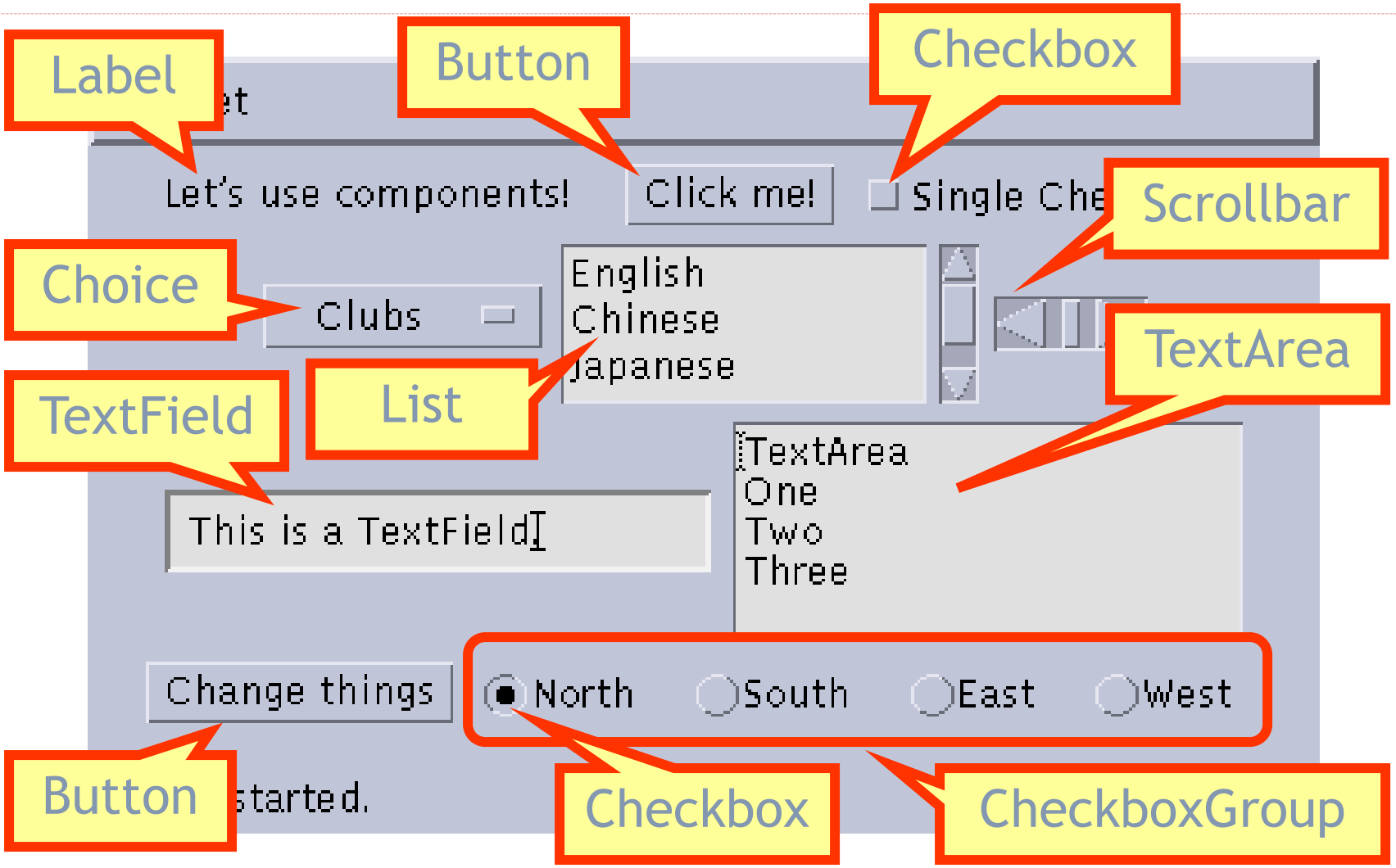
CONTAINER

- ScrollPane (java.awt.ScrollPane)
  - Là một khung chứa tương tự khung chứa Panel, nhưng có thêm 2 thanh trượt giúp ta tổ chức và xem được các đối tượng lớn chiếm nhiều chỗ trên màn hình như những hình ảnh hay văn bản nhiều dòng

COMPONENTS

- Tất cả các thành phần cấu tạo nên chương trình GUI được gọi là component
- Người dùng tương tác với các thành phần thông qua con trỏ chuột hay bàn phím
- Ví dụ:
  - Containers
  - TextFields, Labels, CheckBoxes, TextAreas…
  - Scrollbars, Scrollpanes, Dialog…

COMPONENTS


Label
Button
Checkbox
Let's use components!
Click me!
Single Che
Scrollbar
Choice
Clubs
English
Chinese
Japanese
TextArea
TextField
List
This is a TextField
TextArea
One
Two
Three
Change things
North
South
East
West
Button
Checkbox
CheckboxGroup

COMPONENTS

- Label (java.awt.Label)
  - Dùng để hiển thị chuỗi
  - Constructors:
    - Label();
    - Label(String text);
    - Label(String text, int alignment);
  - Một số phương thức thông dụng :
    - setFont(Font f);
    - setText(String text);
    - getText();
    - …

COMPONENTS

- TextField (java.awt.TextField)
  - Một Textfield là một vùng chỉ chứa một dòng đơn, trong đó văn bản có thể được trình bày hay được nhập vào bởi người dùng
  - Constructors:
    - TextField();
    - TextField(int columns);
    - TextField(String s);
    - TextField(String s, int columns);
  - Một số phương thức thông dụng :
    - setEchoChar(char c);
    - setText(String s);
    - getText();
    - setEditable(boolean editable);
    - isEditable();
    - ...

COMPONENTS

- TextArea (java.awt.TextArea)
  - Là điều khiển text có thể soạn thảo với nhiều dòng
  - Constructors:
    - TextArea();
    - TextArea(int rows, int cols);
    - TextArea(String text);
    - TextArea(String text, int rows, int cols);
  - Một số phương thức thông dụng :
    - setText(String text);
    - getText( );
    - setEditable(boolean editable);
    - isEditable( );
    - insertText(String text, int pos);
    - replaceText(String text, int rows, int cols);
    - ...

COMPONENTS

- Button (java.awt.Button)
  - Nút nhấn hay còn gọi là nút lệnh là một phần nguyên của bất kỳ GUI nào
  - Sử dụng button là cách dễ nhất để chặn các tác động của người dùng
  - Constructors:
    - Button();
    - Button(String S);
  - Một số phương thức thông dụng :
    - void setLabel(String text);
    - String getLabel();

COMPONENTS

- Checkbox (java.awt.Checkbox) và RadioButton
  - CheckBox được sử dụng khi cho phép user chọn một hay nhiều chọn lựa.
  - RadioButton
    - Được dùng để người dùng chỉ ra một lựa chọn duy nhất
    - CheckboxGroup (java.awt.CheckboxGroup) chứa nhiều checkbox nhưng chỉ cho phép chọn một chọn lựa
    - Phần tử trong CheckboxGroup là đối tượng thuộc lớp Checkbox nhưng lại thể hiện dạng nút tròn (radio button)

COMPONENTS

- Checkbox và RadioButton
  - Constructors:
    - Checkbox();
    - Checkbox(String text);
    - Checkbox(String text, CheckboxGroup group, boolean value);
  - Để tạo các RadioButton, trước hết chúng ta phải tạo đối tượng CheckBoxGroup

```
CheckboxGroup cg = new CheckboxGroup();
Checkbox radMale = new Checkbox(“male”, cg, true);
Checkbox radFemale = new Checkbox(“female”, cg, false);
```

- Checkbox và RadioButton
  - Một số phương thức thông dụng:
    - void setLabel(String text);
    - String getLabel();
    - void setState(boolean state);
    - boolean getState();
    - ...

COMPONENTS

- Choice (java.awt.Choice)
  - Tạo danh sách có nhiều chọn lựa
  - Khi list được tạo lần đầu tiên, nó được khởi tạo là empty
  - Constructors:
    - Choice();

- Choice (java.awt.Choice)
  - Một số phương thức thông dụng:
    - void addItem(String item);
    - void insert(String item, int index);
    - int CountItems();
    - int getItemCount();
    - String getItem(int index);
    - int getSelectedInddex();
    - String getSelectedItem();
    - void remove(int pos);
    - void removeAll();
    - void select(int pos);
    - void select(String text);

COMPONENTS

- Scrollbar (java.awt.Scrollbar)
  - Công cụ nhập một trị trong 1 khoảng số (biểu diễn bằng Maximum, Minimum) bằng cách kéo thanh trượt.
  - Tại một thời điểm, thanh trượt ở tại vị trí mô tả cho trị hiện hành (Value).
  - Có thể có hướng ngang hoặc dọc (Orientation)
  - Constructors:
    - Scrollbar();
    - Scrollbar(int orientation); // VERTICAL|HORIZONTAL
    - Scrollbar(int orientation, int value, int visible, int minimum, int maximum);

- Scrollbar (java.awt.Scrollbar)
  - Một số phương thức thông dụng :
    - void setMaximum(int v);
    - int getMaximum();
    - void setMinimum(int v) ;
    - int getMinimum();
    - int getOrientation();
    - void setUnitIncrement(int v);
    - int getUnitIncrement();
    - void setBlockIncrement(int v);
    - int getBlockIncrement();
    - void setValue(int v);
    - int getValue() ;
    - ...

LAYOUT MANAGER

- Container nhận các đối tượng từ bên ngoài đưa vào và nó phải biết làm thế nào để tổ chức sắp xếp "chỗ" cho các đối tượng đó
- Mỗi đối tượng khung chứa đều có một bộ quản lý chịu trách nhiệm thực hiện công việc trên gọi là bộ quản lý trình bày (Layout Manager)
- Layout Manager được thiết lập bằng phương thức được gọi là setLayout()

LAYOUT MANAGER

- Các bộ quản lý trình bày mà thư viện AWT cung cấp bao gồm:
  - FlowLayout
  - BorderLayout
  - GridLayout
  - GridBagLayout
  - Null Layout

LAYOUT MANAGER

- FlowLayout
  - Các đối tượng được sắp xếp từ trái qua phải và từ trên xuống dưới.
    - Các đối tượng đều giữ nguyên kích thước của mình.
    - Nếu chiều rộng của Container không đủ chỗ cho các component thì chúng tự động tạo ra một dòng mới.
    - Chúng ta có thể điều chỉnh khoảng cách giữa các component.
    - Giữa các component, chúng cách nhau theo chiều dọc (vgap) bao nhiêu, theo chiều ngang (hgap) bao nhiêu?

LAYOUT MANAGER

- FlowLayout
  - Constructors:
    - FlowLayout();
    - FlowLayout(int align);
    - FlowLayout(int align, int hgap, int vgap);
  - Các điều khiển có thể được canh về bên trái (LEFT), bên phải (RIGHT) hay ở giữa (CENTER)

    Ví dụ: để canh các điều khiển về bên phải, bạn sử dụng cú pháp sau:

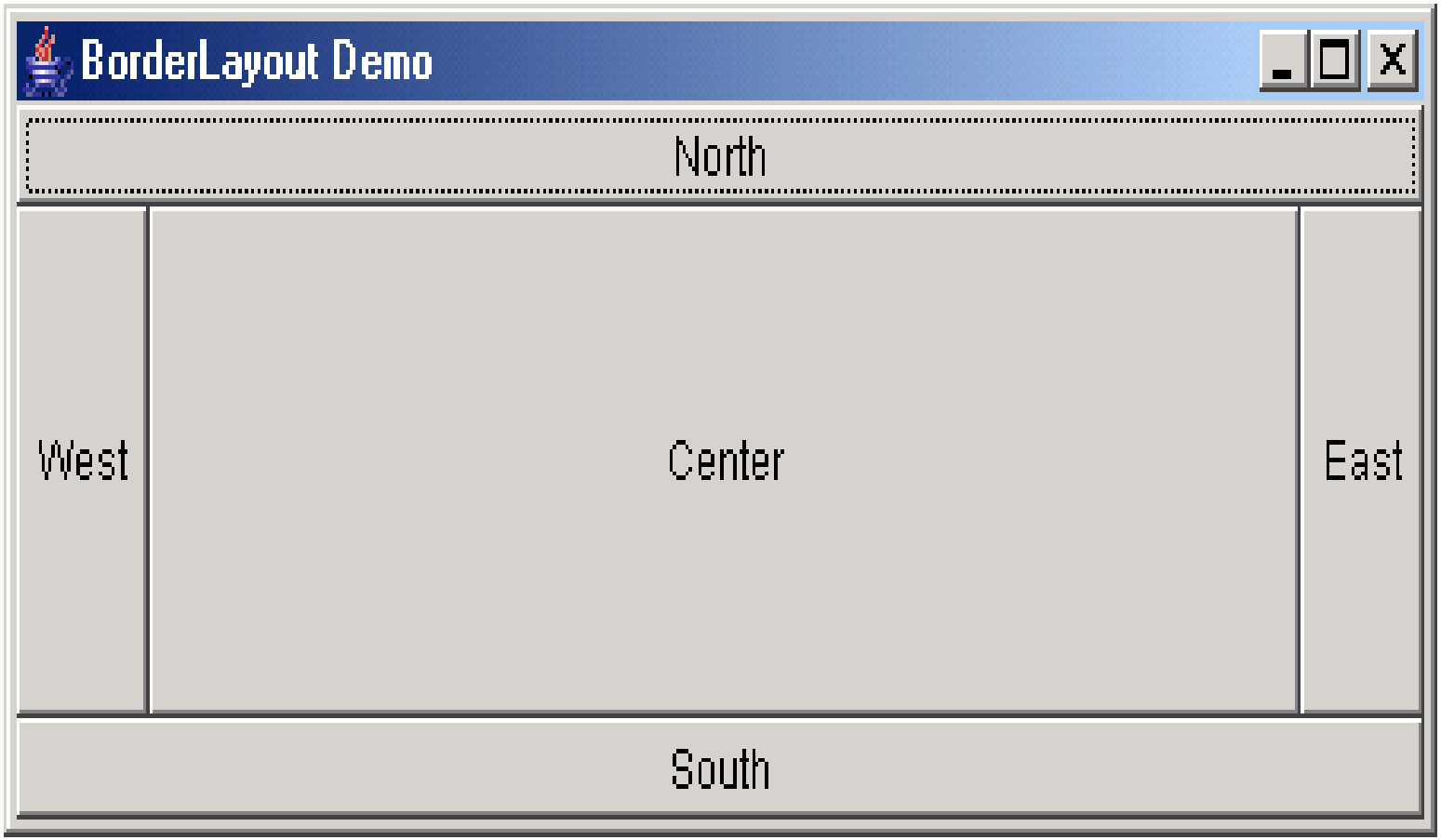
    ```
    setLayout(new FlowLayout(FlowLayout.RIGHT));
    ```

LAYOUT MANAGER

- BorderLayout
  - Là Layout Manager mặc định cho Window, Frame và Dialog
  - Các đối tượng được đặt theo các đường viền của khung chứa theo các cạnh West, East, South, North và Center tức Đông, Tây, Nam, Bắc và Trung tâm hay Trái, Phải, Trên, Dưới và Giữa tùy theo cách nhìn của chúng ta.
  - Nếu container chỉ có 1 component và đặt nó ở vị trí CENTER thì component này phủ kín container.

LAYOUT MANAGER

- BorderLayout



LAYOUT MANAGER

- GridLayout
  - Bố trí các component thành 1 lưới rows dòng, cols cột đều nhau
  - Kiểu layout này được sử dụng khi tất cả các thành phần có cùng kích thước
  - Thứ tự sắp xếp cũng từ trái qua phải và từ trên xuống dưới

LAYOUT MANAGER

- GridLayout
  - Constructors:
    - GridLayout( );
    - GridLayout(int rows, int cols);
    - GridLayout(int rows, int cols, int hgap, int vgap);

LAYOUT MANAGER

- GridBagLayout
  - Tương tự như GridLayout, các đối tượng khung chứa cũng được đưa vào một lưới vô hình
  - Tuy nhiên kích thước các đối tượng không nhất thiết phải vừa với một ô mà có thể là 2, 3 ô hay nhiều hơn tùy theo các ràng buộc mà ta chỉ định thông qua đối tượng GridBagConstraint

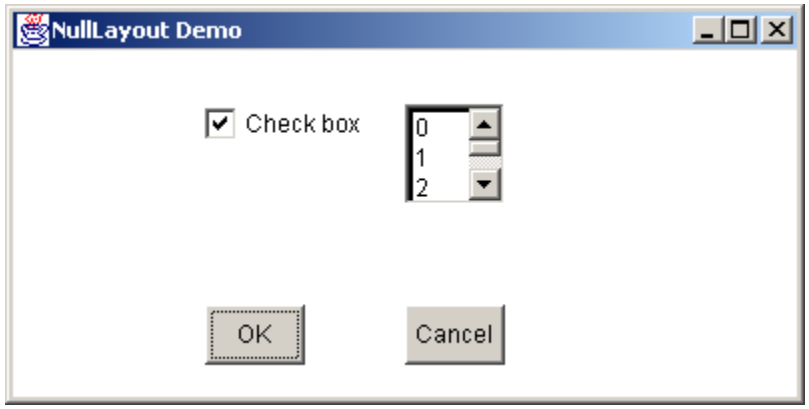
LAYOUT MANAGER

- GridBagLayout
  - Lớp GridBagConstraints kế thừa từ lớp Object.
    - int gridx, gridy: Vị trí ô của khung lưới vô hình mà ta sẽ đưa đối tượng con vào
    - int gridwidth, gridheight: Kích thước hay vùng trình bày cho đối tượng con.
    - Insets: Là một biến đối tượng thuộc lớp Inset dùng để qui định khoảng cách biên phân cách theo 4 chiều (trên, dưới, trái, phải).
    - double weightx, weighty: Khoảng hở của lưới, mặc định là 0

LAYOUT MANAGER

- Null Layout
  - Cách trình bày tự do.
  - Đối với cách trình bày này người lập trình phải tự làm tất cả từ việc định kích thước của các đối tượng, cũng như xác định vị trí của nó trên màn hình.

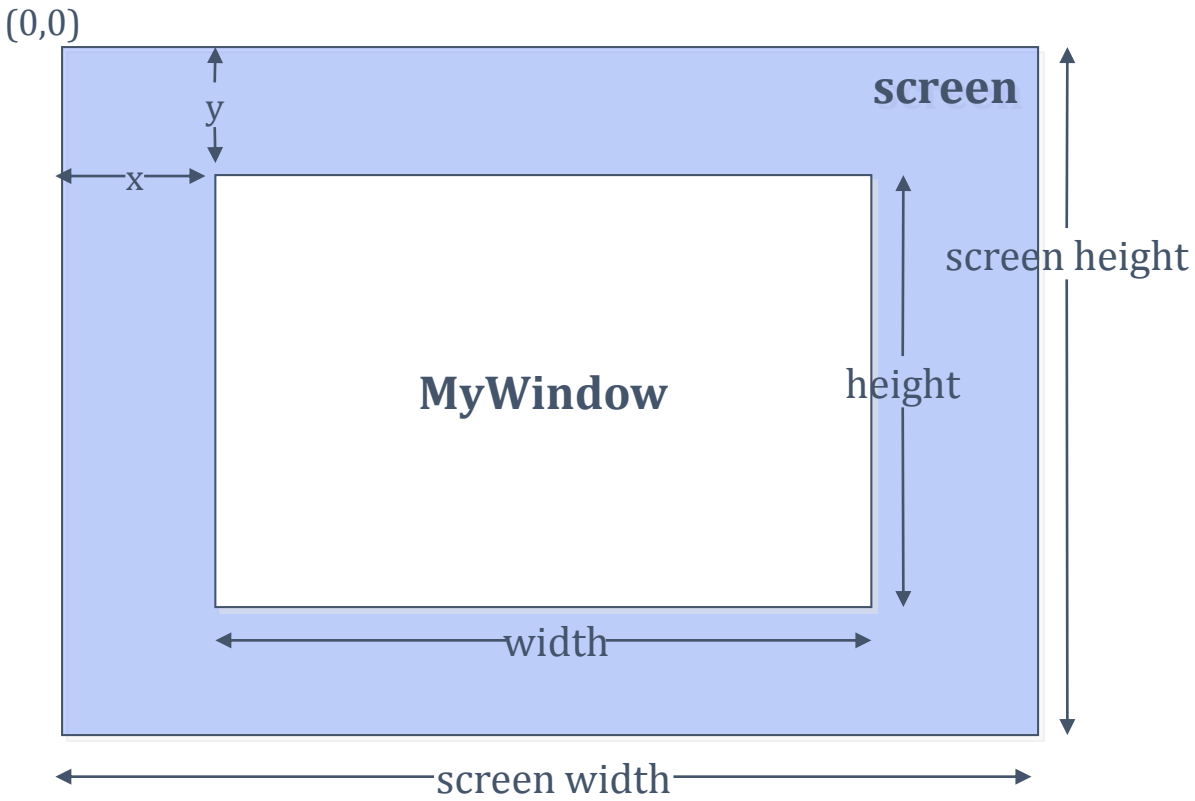
```
Frame fr = new Frame("NullLayout Demo");
fr.setLayout(null);
```



XÂY DỰNG ỨNG DỤNG GUI

- Nguyên tắc xây dựng GUI:
  - Lựa chọn 1 container: Frame, Panel, Dialog,…
  - Tạo các điều khiển: Label, Button, TextArea…
  - Đưa các điều khiển vào vùng chứa
  - Sắp xếp các điều khiển (layout)
  - Thêm các xử lý sự kiện (Listeners)
    - Sẽ được giới thiệu trong phần sau

- Cách tính tọa độ:



XÂY DỰNG ỨNG DỤNG GUI

- Dữ liệu nhập/xuất → Chọn component?
- Các tác vụ → Mỗi tác vụ là một 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

XÂY DỰNG ỨNG DỤNG GUI

- 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, ...

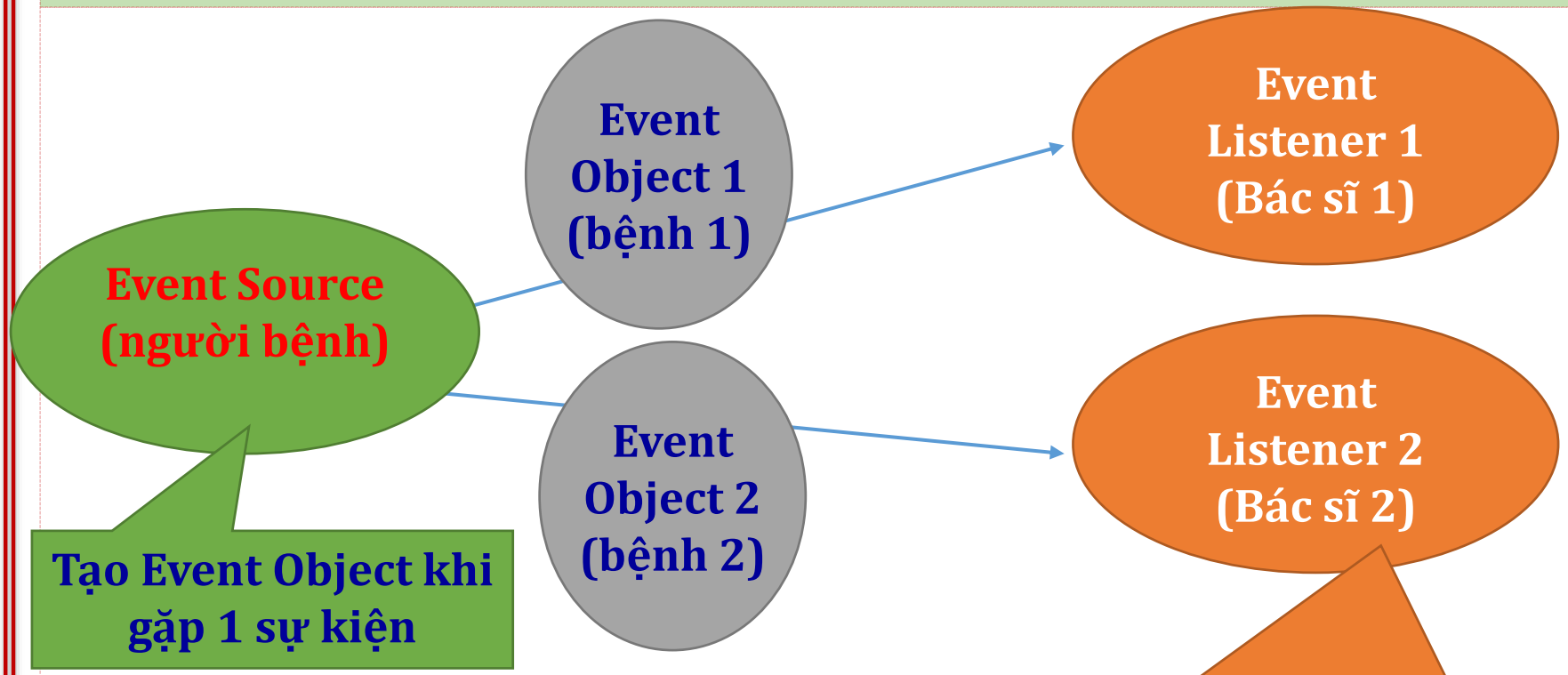
LẬP TRÌNH XỬ LÝ SỰ KIỆN

❑Giới thiệu mô hình ứng dụng hướng sự kiện:

- ❑Event-Oriented Application Model: Chương trình có GUI, user tương tác với GUI qua chuột, bàn phím,…, chương trình xử lý, trạng thái mới lại xuất ra cho user xem ➔ thân thiện.
- ❑Event: một tín hiệu mà ứng dụng nhận biết có sự thay đổi trạng thái của 1 đối tượng.
- ❑3 nguồn phát xuất event:
  - ❑(1) User (gõ phím, kích chuột vào 1 phần tử,…)
  - ❑(2) Do hệ thống (do định thời 1 tác vụ)
  - ❑(3) Do 1 event khác (các event kích hoạt nhau)
- ❑Hiện nay, đa số các ngôn ngữ đều cung cấp mô hình này, VC++ cung cấp MFC *(Microsoft Foundation Classes)*, Java cung cấp JFC *(Java Foundation Classes)*.

LẬP TRÌNH XỬ LÝ SỰ KIỆN

❑Ví dụ minh họa việc ủy thác xử lý sự kiện:



LẬP TRÌNH XỬ LÝ SỰ KIỆN

❑Ba thành phần chính của mô hình:

- ❑ **Event source**: nguồn gây ra sự kiện, thường là các thành phần GUI trong chương trình
- ❑ **Event object**: đối tượng lưu thông tin về sự kiện đã xảy ra
- ❑ **Event listener**: đối tượng sẽ nhận được thông tin khi có sự kiện xảy ra

LẬP TRÌNH XỬ LÝ SỰ KIỆN

- Sự kiện (event) được phát sinh khi người dùng tương tác với GUI, ví dụ: di chuyển chuột, ấn nút, nhập dữ liệu văn bản, chọn menu...
- Thông tin về sự kiện được lưu trong một đối tượng sự kiện thuộc lớp con của lớp AWTEvent (java.awt.event).
- Chương trình có thể xử lý các sự kiện bằng cách đặt *"lắng nghe sự kiện"* trên các thành phần GUI.

LẬP TRÌNH XỬ LÝ SỰ KIỆN

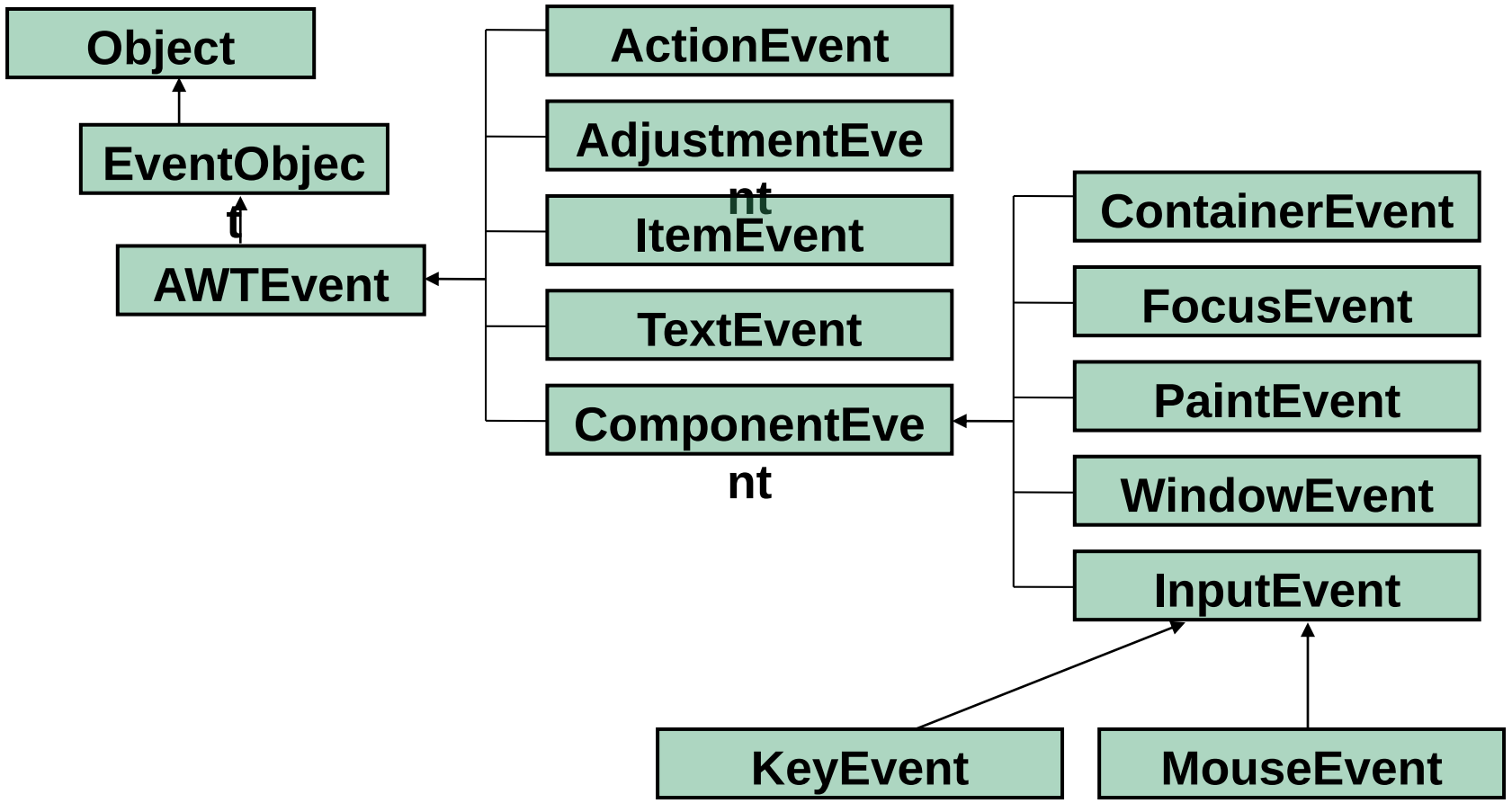
- Việc thông báo sự kiện xảy ra thực chất là việc gọi một phương thức của EventListener với đối số truyền vào là EventObject.
- Các lớp con của EventListener có thể cài đặt các phương thức để xử lý sự kiện

LẬP TRÌNH XỬ LÝ SỰ KIỆN

- Nguồn sự kiện
  - Các lớp thành phần GUI mà người sử dụng tương tác.
  - Bạn có thể đăng ký "Listener" đáp ứng với những sự kiện nhất định
- Bộ lắng nghe (Listener)
  - Nhận đối tượng sự kiện khi được thông báo và thực hiện đáp ứng thích hợp.
  - Nhiều kiểu của bộ lắng nghe tồn tại cho các sự kiện cụ thể như MouseListener, ActionListener, KeyListener,…
  - Các giao tiếp được hiện thực và cài đặt các hành động
- Đối tượng sự kiện (Event)
  - Đóng gói thông tin về sự kiện xuất hiện
  - Các đối tượng sự kiện được gửi tới bộ lắng nghe khi sự kiện xuất hiện trên thành phần GUI

LẬP TRÌNH XỬ LÝ SỰ KIỆN

- Gói **java.awt.event.***



LẬP TRÌNH XỬ LÝ SỰ KIỆN

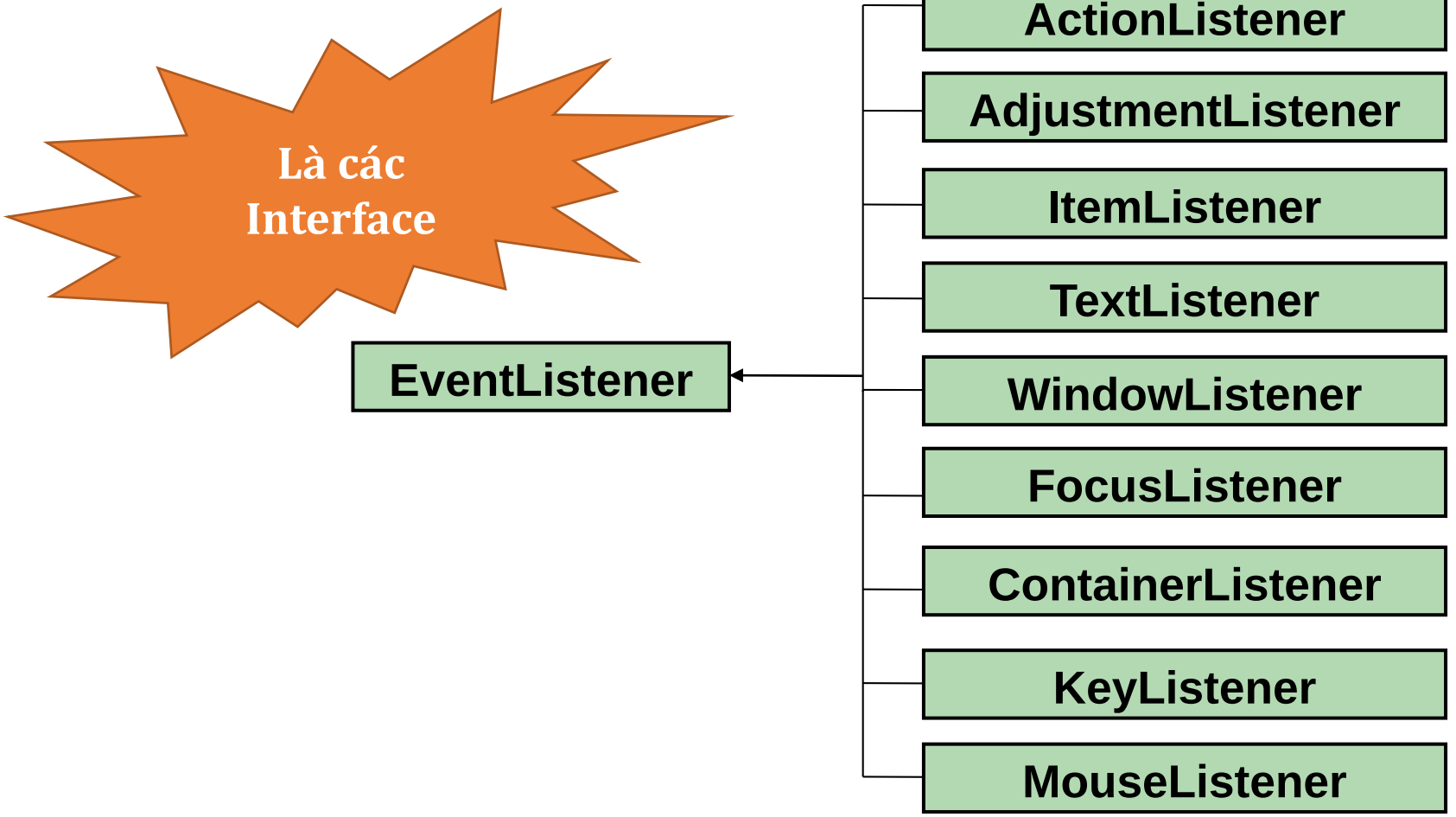
- Một số lớp sự kiện:
  - Sự kiện cấp thấp: dùng cho hầu hết các thành phần
    - FocusEvent: đặt/chuyển focus
    - InputEvent: sự kiện phím (KeyEvent) hoặc chuột (MouseEvent)
    - ContainerEvent: thêm hoặc xoá các component
    - WindowEvent: đóng, mở, di chuyển cửa sổ
    - ...

LẬP TRÌNH XỬ LÝ SỰ KIỆN

- Một số lớp sự kiện:
  - Sự kiện cấp cao: dùng cho một số thành phần đặc thù
    - ActionEvent: sự kiện sinh ra từ các thành phần giao tiếp với người dùng như nhấn một nút, chọn menu…
    - ItemEvent: lựa chọn một item trong danh sách
    - TextEvent: thay đổi giá trị của hộp text
    - ...

LẬP TRÌNH XỬ LÝ SỰ KIỆN

- Một số bộ lắng nghe sự kiện



LẬP TRÌNH XỬ LÝ SỰ KIỆN

- Cài đặt và quản lý sự kiện:
  - Xác định đối tượng sẽ gây ra sự kiện (event source).

    Ví dụ: nút bấm.
  - Xác định sự kiện cần xử lý trên đối tượng gây sự kiện.

    Ví dụ: ấn nút.
  - Xác định đối tượng nghe sự kiện (event listener) và cài đặt các phương thức tương ứng.

    Ví dụ: chính applet sẽ nghe sự kiện.
  - Đăng ký đối tượng nghe trên đối tượng gây ra sự kiện.

    Ví dụ: `button.addActionListener(...);`

LẬP TRÌNH XỬ LÝ SỰ KIỆN

❑Các Event Source và Event Object:

Event source	Event	Chú thích
Button	ActionEvent	Nhấn nút
Checkbox	ItemEvent	Chọn, bỏ chọn một item
Choice	ItemEvent	Chọn, bỏ chọn một item
Component	ComponentEvent	Ẩn, hiện, di chuyển
	FocusEvent	Được chọn
	MouseEvent	Tương tác chuột
	KeyEvent	Tương tác bàn phím
Container	ContainerEvent	Thêm, bớt component
List	ActionEvent	Nhấp kép chuột một item
	ItemEvent	Chọn, bỏ chọn một item

LẬP TRÌNH XỬ LÝ SỰ KIỆN

- Các Event Source và Event Object:

Event source	Event	Chú thích
MenuItem	ActionEvent	Chọn một menu item
Scrollbar	AdjustmentEvent	Di chuyển thanh cuộn
TextComponent	TextEvent	Thay đổi văn bản
TextField	ActionEvent	Kết thúc thay đổi văn bản
Window	WindowEvent	Thay đổi cửa sổ

LẬP TRÌNH XỬ LÝ SỰ KIỆN

- Các Listener Method:

Event Class	Listener Interface	Listener Methods
ActionEvent	ActionListener	actionPerformed()
AdjustmentEvent	AdjustmentListener	adjustmentValueChanged()
ComponentEvent	ComponentListener	componentHidden()
		componentMoved()
		componentResized()
		componentShown()
ContainerEvent	ContainerListener	componentAdded()
		componentRemoved()
FocusEvent	FocusListener	focusGained()
		focusLost()
ItemEvent	ItemListener	itemStateChanged()

LẬP TRÌNH XỬ LÝ SỰ KIỆN

- Các Listener Method:

Event Class	Listener Interface	Listener Methods
KeyEvent	KeyListener	keyPressed()
		keyReleased()
		keyTyped()
MouseEvent	MouseListener	mouseClicked()
		mousePressed()
		mouseReleased()
	MouseMotionListener	mouseDragged()
		mouseMoved()
TextEvent	TextListener	textValueChanged()
WindowEvent	WindowListener	windowClosed()
		windowActivated()

❑Đăng ký đối tượng lắng nghe:

- ❑Để đăng ký đối tượng nghe ta sử dụng tên phương thức có cấu trúc như sau:

  add + loại sự kiện + Listener(lớp nghe sự kiện)

- ❑Ví dụ với nút Button

  addActionListener(ActionListener)

- ❑Ví dụ với danh sách List

  addActionListener(ActionListener)

  addItemListener(ItemListener)

PHỤ LỤC: CÁC LỚP SỰ KIỆN

ActionEvent Class

- Một ActionEvent object được sinh ra khi: 1 nút lệnh bị kích, một mục chọn trong danh sách bị kích đôi, 1 mục menu bị kích.
- Các hằng kiểm tra có 1 phím bị nhấn khi kích chuột hay không: ALT_MASK (phím Alt), CTRL_MASK (phím Ctrl), META_MASK (phím meta, ký tự mô tả về 1 ký tự khác - ký tự escape), SHIFT_MASK (phím Shift).

PHỤ LỤC: CÁC LỚP SỰ KIỆN

- **ActionEvent Class**
  - Event source: Button, List Item, Menu
  - Constructor:
    - ActionEvent(Object source, int id, String command)
    - ActionEvent(Object source, int id, String command, int modifiers)
  - Các phương thức thường dùng:
    - public String getActionCommand() *Lấy tên tác vụ kết hợp với Source*
    - public int getModifiers() *Lấy bit mask của phím điều khiển đi kèm (Shift,Alt,Ctrl)*
    - public Object getSource() *Lấy nguồn gây event*
  - ActionListener interface có 1 event handler:
    - void actionPerformed (ActionEvent e)

PHỤ LỤC: CÁC LỚP SỰ KIỆN

AdjustmentEvent Class

- Được sinh ra khi 1 thanh cuộn bị thao tác.
- Các hằng int:BLOCK_DECREMENT, BLOCK_INCREMENT: Độ giảm/tăng theo khối khi user kích chuột vào vùng giữa con trượt và 1 biên của thanh cuộn, UNIT_DECREMENT, UNIT_INCREMENT: Đơn vị giảm/tăng khi user kích chuột vào mũi tên ở 2 đầu thanh cuộn. TRACK: Giá trị mô tả thanh cuộn khi bị user kéo.
- Các phương thức thường dùng:
  - Adjustable getAdjustable() *Lấy đối tượng Source*
  - int getAdjustableType() *Lấy trị hằng mô tả ở trên*
  - int getValue() *Lấy trị hiện hành của thanh cuộn*
- AdjustmentListener interface có 1 event handler:
  - void adjustmentValueChange(AdjustmentEvent e)

PHỤ LỤC: CÁC LỚP SỰ KIỆN

- **ComponentEvent Class**
  - Được sinh ra khi 1 componet bị ẩn đi, được hiển thị, bị di chuyển, bị thay đổi kích thước.
  - Các hằng mô tả trạng thái gồm: COMPONENT_HIDDEN, COMPONENT_MOVED, COMPONENT_RESIZED, COMPONENT_SHOWN.
  - Phương thức:
    - Component getComponent() *Lấy đối tượng phát sinh sự kiện*
  - ComponentListener interface có các event handler:
    - void componentHidden(ComponentEvent e)
    - void componentMoved(ComponentEvent e)
    - void componentResized(ComponentEvent e)
    - void componentShown(ComponentEvent e)

PHỤ LỤC: CÁC LỚP SỰ KIỆN

- **ContainerEvent Class**
  - Được sinh ra khi 1 component được thêm/xóa khỏi 1 container.
  - Các hằng mô tả sự kiện: COMPONENT_ADDED, COMPONENT_REMOVED.
  - Các phương thức thường dùng:
    - Component getChild() *Lấy component được added/removed*
    - Container getContainer() *Lấy source container của sự kiện*
  - ContainerListener interface có 2 event handler:
    - void componentAdded(ContainerEvent e)
    - void componentRemoved(ContainerEvent e)

PHỤ LỤC: CÁC LỚP SỰ KIỆN

- **FocusEvent Class**
  - Được sinh ra khi 1component có/mất focus.
  - Các hằng: FOCUS_GAINED, FOCUS_LOST.
  - Phương thức:
    - boolean isTemporary() *Trả về true nếu việc mất focus là tạm thời. Việc mất focus là tạm thời khi focus ở tại 1 phần tử trên GUI như thanh cuộn, pop-up menu*.
  - FocusListener interface có 2 event handler:
    - void focusGained (FocusEvent e)
    - void focusLost (FocusEvent e)

PHỤ LỤC: CÁC LỚP SỰ KIỆN

ItemEvent Class

- Được sinh ra khi 1 mục được chọn/bỏ chọn trong 1 danh sách Listbox, Combobox, checkbox menuitem.
- Các hằng:
  - SELECTED, DESELECTED,
  - ITEM_STATE_CHANGED
- Các phương thức hay dùng:
  - Object getItem() — *Lấy đối tượng bị thao tác*
  - ItemSelectable getItemSelectable() — *Lấy source của sự kiện*
  - int getStateChange() — *Lấy loại sự kiện (SELECTED/DESELECTED)*
- ItemListener interface có 1 event handler:
  - void itemStateChanged (ItemEvent e)

PHỤ LỤC: CÁC LỚP SỰ KIỆN

InputEvent Class

- Là lớp cha của 2 lớp con: KeyEvent và MouseEvent.
- Các hằng khai áo trong lớp này mô tả các bit mặt nạ truy xuất phím đi kèm sự kiện hoặc nút chuột nào bị nhấn: ALT_MASK, CTRL__MASK, META_MASK, SHIFT_MASK, BUTTON1_MASK, BUTTON2_MASK, BUTTON3_MASK.
- Meta character : Ký tự mô tả về 1 ký tự khác – Ví dụ: Ký tự backslash (\) chỉ thị rằng ký tự sau nó là thành phần của chuỗi escape trong C, Java
- Các phương thức thường dùng:
  - int getModifier() *Lấy bit mặt nạ*
  - boolean isAltDown() *Kiểm tra có phím bấm đi kèm*
  - boolean isMetaDown()
  - boolean isShiftDown()
  - boolean isControlDown()

PHỤ LỤC: CÁC LỚP SỰ KIỆN

KeyEvent Class

- Được sinh ra khi user thao tác với bàn phím .
- Các hằng kiểu int KEY_PRESSED, KEY_RELEASED, KEY_TYPED. Nếu phím chữ, phím số được gõ, cả 3 loại sự kiện được sinh ra (pressed, released, typed). Nếu phím đặc biệt được thao tác (phím Home, End, PageUp, PageDown-modifier key), chỉ có 2 sự kiện được sinh ra: pressed, released.
- Hai phương thức thường dùng để truy cập phím bị thao tác:
  - char getKeyChar()
  - int getKeyCode()
- KeyListener interface có 3 event handler:
  - void keyPressed(KeyEvent e)
  - void keyReleased(KeyEvent e)
  - void keyTyped(KeyEvent e)

PHỤ LỤC: CÁC LỚP SỰ KIỆN

MouseEvent Class

- Được sinh ra khi user thao tác chuột với 1 component.
- Các hằng int:MOUSE_CLICKED, MOUSE_DRAGGED, MOUSE_ENTERED, MOUSE_EXITED, MOUSE_MOVED, MOUSE_PRESSED, MOUSE_RELEASED.
- Các phương thức thường dùng:
  - Point getPoint() *Lấy vị trí của mouse lúc sự kiện xảy ra*
  - int getX(), int getY() *Lấy tọa độ x,y của vị trí chuột*
- MouseListener interface có 5 event handler:
  - void mouseClicked(MouseEvent e)
  - void mouseEntered(MouseEvent e)
  - void mouseExited(MouseEvent e)
  - void mousePressed(MouseEvent e)
  - void mouseReleased(MouseEvent e)

PHỤ LỤC: CÁC LỚP SỰ KIỆN

- **TextEvent Class**
  - Được sinh ra khi các ký tự trong 1 TextField hay 1 TextArea bị đổi.
  - Hằng int: TEXT_VALUE_CHANGED
  - TextListener interface có 1 event handler:
    - void textValueChanged(TextEvent e)

PHỤ LỤC: CÁC LỚP SỰ KIỆN

- **WindowEvent Class**
  - Được sinh ra khi 1 cửa sổ: activated, deactivated, iconified,deiconified, opened, closed, closing
  - Các hằng int: WINDOW_ACTIVATED, WINDOW_DEACTIVATED, WINDOW_OPENED, WINDOW_CLOSED, WINDOW_CLOSING, WINDOW_ICONIFIED, WINDOW_DEICONIFIED.
  - Phương thức:
    - Window getWindow() *Lấy source window*
  - WindowListener interface có 7 event handler:
    - void windowActivated(WindowEvent e)
    - void windowDeactivated(WindowEvent e)
    - void windowOpened(WindowEvent e)
    - void windowClosed(WindowEvent e)
    - void windowClosing(WindowEvent e)
    - void windowIconified(WindowEvent e)
    - void windowDeiconified(WindowEvent e)

THAM KHẢO

- Java document (online)
- http://docs.oracle.com/javase/7/docs/api/
- Java document (offline)
- http://www.oracle.com/technetwork/java/javase/documentation/java-se-7-doc-download-435117.html

HỎI – ĐÁP

THANK YOU…