Chương 5
Abstract Window Toolkit

GV biên soạn:
Ths Nguyễn Thị Đoan Trang

Nội dung

- Hiểu về AWT
- Sử dụng các Component
- Sử dụng các Container
- Sử dụng các Layout Manager
- Xử lý sự kiện với các Component

5.1 GIỚI THIỆU VỀ AWT

- Phần mềm ứng dụng hiện nay ngày càng thân thiện vì được thể hiện bởi giao diện đồ họa đẹp mắt.
- Các ngôn ngữ lập trình được hỗ trợ các đối tượng đồ họa, chúng có thể được điều khiển bởi người lập trình, hay người sử dụng.
- Đa số ngôn ngữ hiện nay được dựa trên Giao diện người dùng đồ họa (Graphical User Interface - GUI).
- Trong chương này, nghiên cứu về Java hỗ trợ tính năng GUI cùng tiến trình thực thi của chúng.

5.1 GIỚI THIỆU VỀ AWT

- GUI cung cấp chức năng nhập liệu theo cách thân thiện với người dùng.
- GUI có thể chứa nhiều đối tượng điều khiển như textbox, label, listbox, …
- Các ngôn ngữ lập trình khác nhau cung cấp các cách khác nhau để tạo GUI.
- Các phần mềm giống như VB hay VC++ có thể cung cấp chức năng kéo và thả, nhưng C++ yêu cầu người lập trình phải viết toàn bộ mã để xây dựng một GUI.

5.1 GIỚI THIỆU VỀ AWT

- Trong Java, để tạo GUI, chúng ta cần sử dụng các lớp tồn tại trong gói **java.awt**.
- AWT (Abstract Window Toolkit): Là một bộ các lớp trong Java cho phép chúng ta tạo một GUI và cho phép thao tác thông qua bàn phím và chuột. AWT cung cấp các đối tượng và phương thức để tạo một GUI hiệu quả và lôi cuốn người sử dụng. Bao gồm:
  - Không gian chứa (**Container**)
  - Thành phần (**Component**)
  - Trình quản lý cách trình bày (**Layout manager**)
  - Đồ họa (**Graphic**) và các tính năng vẽ (**draw**)
  - Phông chữ (**Font)**
  - Sự kiện (**Event**)

5.2 Container

- Container là vùng có thể đặt các thành phần vào đó. Bất cứ đối tượng nào kế thừa từ lớp Container sẽ là một container.
- Một container có thể chứa nhiều đối tượng, các đối tượng này có thể được thể hiện tuỳ ý. Có thể container như một cửa sổ: Có khung (frame), viền, kích thước,...
- Gói java.awt chứa một lớp gọi là Container. Lớp này trực tiếp hay gián tiếp phát sinh ra hai container được sử dụng phổ biến nhất là Frame và Panel.
- Frame và Panel là các container thường được sử dụng. Frame là các cửa sổ được tách riêng nhau nhưng ngược lại panel là các vùng được chứa trong một cửa sổ.

5.2.1 Frame

- Frame không phụ thuộc vào applet và trình duyệt. Frame có thể hoạt động như một container hay như một thành phần (component).
- Sử dụng một trong những constructor sau để tạo một frame:

✓ **Frame()**: Tạo một frame vô hình (không nhìn thấy được)

✓ **Frame(String title)**: Tạo một frame với tiêu đề

5.2.1Frame

- Chương trình demo 5.2.1

```
package frame;
import java.awt.*;
public class FrameDemo extends Frame{
        public FrameDemo(String title)
    {
            super(title);
    }
    public static void main(String args[])
    {
            FrameDemo f=new FrameDemo("I have been Frameed!!!");
            f.setSize(300,200); //Thiết lập kích thước Frame
            f.setVisible(true); // Cho phép Frame hiển thị
        }
}
```

5.2.1 Frame

- Lớp được định nghĩa Framedemo là một lớp con của lớp Frame.
- Lớp FrameDemo này có một constructor, trong contructor này ta cho gọi hàm super(). Mục đích của super() là gọi constructor của lớp cha mẹ.
- Tiến trình này sẽ tạo một đối tượng của lớp con, lớp con này sẽ tạo frame. Thêm vào đó, nó cũng sẽ cho phép đối tượng frame nhìn thấy được thông qua phạm vi lớp.
- Tuy nhiên, frame vẫn không nhìn thấy được và không có kích thước. Để làm được điều này, ta sử dụng hai phương thức nằm trong phương thức main: setSize() và setVisible().

5.2.2 Panel

- Panel được sử dụng để nhóm một số các thành phần lại với nhau.
- Để tạo một panel là sử dụng hàm constructor của nó, hàm Panel().

5.2.2 Panel

- Chương trình 5.2.2 chỉ ra cách tạo một panel:

```
package panel;
import java.awt.*;
public class PanelDemo extends Panel{
    public static void main(String args[])
    {
       PanelDemo p=new PanelDemo();
       Frame f=new Frame("Testing a Panel");
       f.add(p); //Thêm panel vào frame
       f.setSize(300,200); //Thiết lập kích thước frame
       f.setVisible(true); // Hiển thị frame
    }
    public PanelDemo()
    {
    }
}
```

5.2.2 Panel

- Panel không thể được nhìn thấy trực tiếp.
- Do đó, chúng ta cần thêm panel đến một frame. Vì vậy ta cần tạo một frame mới và thêm Panel mới được tạo này vào nó.
- Tuy nhiên, frame sẽ không nhìn thấy được, và không có kích thước. Sử dụng setSize() và setVisible() để thiết lập kích thước và hiển thị frame.

5.2.3 Dialog (Hộp thoại)

- Lớp ‘Dialog’ tương tự như lớp Frame, nghĩa là Dialog là lớp con của lớp Window.
- Đối tượng dialog được tạo như sau:

```
Frame myframe=new Frame(“My frame”); // calling frame
String title = “Title”;
boolean modal = true; // whether modal or not
Dialog dlg=new Dialog(myframe, title, modal);
```

- Tham số modal=true không cho phép làm việc với các cửa sổ đang mở khác; modal=false ngược lại. Kiểu hộp thoại này ngăn chặn người dùng tương tác với các cửa sổ khác trên màn hình, cho tới khi dialog được đóng lại.

5.2.3 Dialog (Hộp thoại)

- Chương trình demo 5.2.3 về lớp Dialog

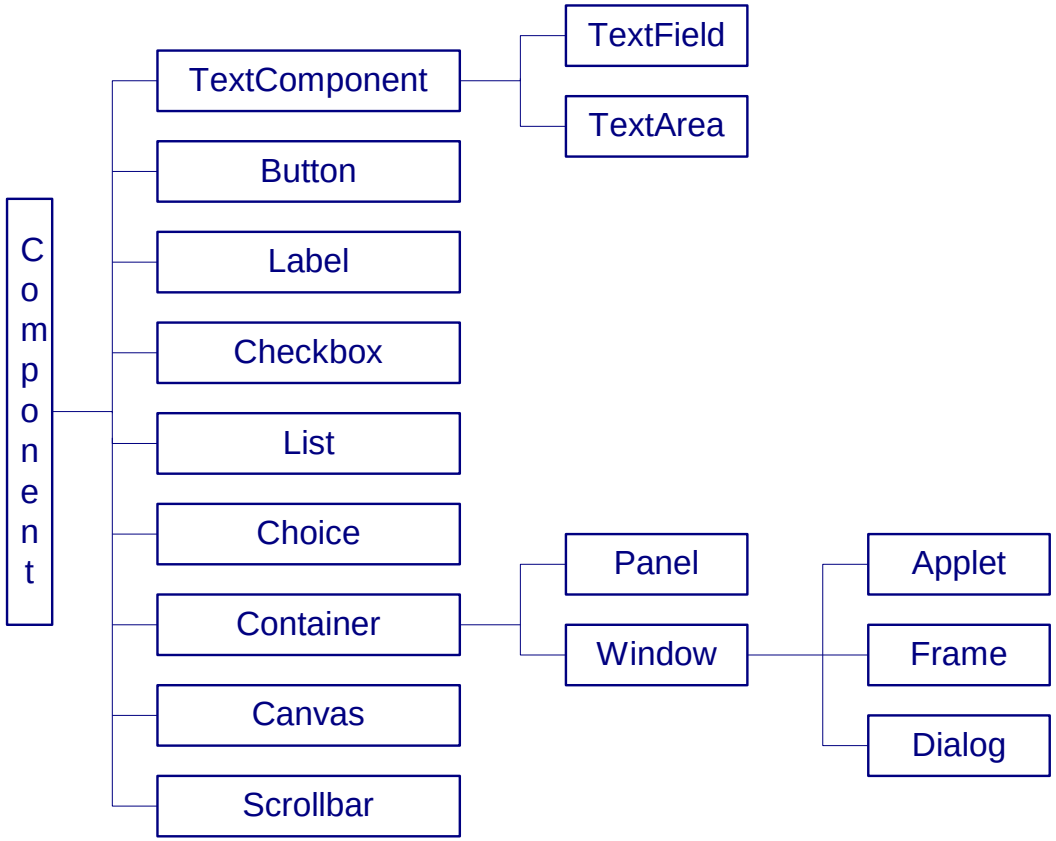
```
package dialog;
import java.awt.*;
public class DialogDemo extends Frame{
    public static void main(String[] args) {
        Frame myframe=new Frame("My frame"); // calling frame
        String title = "Dialog Demo";
        boolean modal = true; // whether modal or not
        Dialog dlg=new Dialog(myframe, title, modal);
        dlg.setSize(300,200);
         dlg.setVisible(true);
    }
}
```

5.3 Component (Thành phần)

- Một component có thể được đặt trên giao diện người dùng, có thể được thay đổi kích thước, ẩn hoặc hiện hay di chuyển.
- Các thành phần được sử dụng thường xuyên là textfield, label, checkbox, textarea v.v… Các thành phần khác như scrollbar và dialog

5.3 Component (Thành phần)

- Sơ đồ phân cấp thành phần



5.3.1Label (Nhãn)

- Lớp này được sử dụng để trình bày một chuỗi. Nó không cho phép sửa đổi bởi người sử dụng.
- Sử dụng một trong những constructor sau đây để tạo một label:
  - **Label():** Tạo một Label trống.
  - **Label(String labeltext):** Tạo một Label với văn bản được cho.
  - **Label(String labeltext, int alignment):** Tạo một Label với một chế độ canh lề alignment được cho, alignment có thể là Label.LEFT, Label.RIGHT hay Label.CENTER.

5.3.1Label (Nhãn)

- Các phương thức được sử dụng phổ biến của label:

Phương thức	**Chức năng**
setFont(Font f)	Thay đổi phông chữ đang được chọn của Label
setText(String s)	Thiết lập nhãn cho Label
getText()	Lấy nội dung hiện hành của Label

5.3.1Label (Nhãn)

- Chương trình Demo 5.3.1

```
package label;
import java.awt.*;
public class Labeltest extends Frame {
    Label label1=new Label("This is just a label");
    public Labeltest(String title)
    {
        super(title);
        add(label1);//Thêm label vào một container
    }
    public static void main(String args[])
    {
        Labeltest f=new Labeltest("Label");
        f.setSize(300,200);
        f.show();
    }
}
```

5.3.2 Ô văn bản (TextField)

- Một textfield là một vùng chỉ chứa một dòng đơn, trong đó văn bản có thể được trình bày hay được nhập vào từ bàn phím bởi người dùng.
- Các constructor khởi tạo textfield:
  - TextField(): Tạo một textfield mới.
  - TextField(int columns): Tạo một textfield mới với số cột được cho.
  - TextField(String s): Tạo một textfield mới với chuỗi văn bản được cho.
  - TextField(String s, int columns): Tạo một textfield mới với nhãn và số cột được cho.

5.3.2 Ô văn bản (TextField)

- Các phương thức thường được sử dụng của đối tượng TextField

Phương thức	Chức năng
setEchoChar(char)	Thiết lập các kí tự được trình bày trong dạng của một kí tự được cho.
setText(String s)	Thiết lập nhãn cho TextField.
getText()	Trả về nhãn của TextField.
setEditable(boolean)	Xác định trường có thể được soạn thảo hay không. Trường chỉ được soạn thảo khi giá trị này được đặt là True.
isEditable()	Xác định xem trường có đang trong mode soạn thảo hay không. Giá trị trả về kiểu Boolean.

5.3.2 Ô văn bản (TextField)

- Chương trình Demo 5.3.2

```
package textbox;
import java.awt.*;
class TextFieldtest extends Frame
{
    TextField tf1=new TextField(30);
    public TextFieldtest(String title)
    {
        super(title);
        setLayout(new FlowLayout()); //Sắp xếp các thành phần trong một container
        add(tf1);
    }
    public static void main(String args[])
    {
        TextFieldtest f=new TextFieldtest("TextField");
        f.setSize(300,200);
        f.show();
    }
}
```

5.3.3 Vùng văn bản (TextArea)

- Một Textarea được sử dụng khi văn bản nhập vào trên hai hay nhiều dòng. Textarea có một scrollbar. Thành phần TextArea là một trường văn bản có thể được soạn thảo với đặc tính nhiều dòng.
- Sử dụng các constructor sau để tạo TextArea:
  - **TextArea()**: Tạo một TextArea mới.
  - **TextArea(int rows, int cols)**: Tạo một TextArea mới với số lượng cột và dòng được cho.
  - **TextArea(String text)**: Tạo một TextArea mới với nhãn được cho.
  - **TextArea(String text, int rows, int cols)**: Tạo một TextArea mới với nhãn, số dòng và số cột được cho.

5.3.3 Vùng văn bản (TextArea)

- Các phương thức thường được sử dụng của TextArea:

Phương thức	**Chức năng**
setText(String)	Thiết lập văn bản cho TextArea.
getText()	Trả về văn của TextArea.
setEdiable(boolean)	Xác định xem trường có thể được soạn thảo hay không. Trường có thể được soạn thảo khi giá trị này là True.
isEdiable()	Xác định xem trường có đang trong mode soạn thảo được không. Trả về giá trị là kiểu Boolean.
insertText(String, int)	Chèn String vào vị trí index được cho.
replaceText(String, int, int)	Thay thế văn bản nằm giữa vị trí int, int được cho.

5.3.3 Vùng văn bản (TextArea)

- Chương trình demo 5.3.3

```
package textarea;
import java.awt.*;
class TextAreatest extends Frame
{
    Label lbl=new Label("Details");//Tạo một label
    TextArea ta1=new TextArea(5,20);//Tạo một vùng văn bản
    public TextAreatest(String title)
    {
      super(title);
      setLayout(new FlowLayout());
      add(lbl);// Thêm label vào container
      add(ta1);//Thêm vùng văn bản vào container
    }
    public static void main(String args[])
    {
      TextAreatest t=new TextAreatest("TextArea");
      t.setSize(300,200);
      t.show();
    }
}
```

5.3.4 Button

- Nút nhấn hay còn gọi là nút lệnh
- Sử dụng một trong hai constructor sau để tạo các button trong Java:
  - **Button()**
  - **Button(String text)**
- Sử dụng setLabel() và getLabel() để thiết lập và nhận về nhãn của button.

5.3.4 Button

- Chương trình demo 5.3.4

```
package button;
import java.awt.*;
class Buttontest extends Frame
{
    Button b1 = new Button("red"); //Tạo mới một button
    Button b2 = new Button("Green");    Button b3 = new Button("Blue");
    public Buttontest(String title)
    {
       super(title);  setLayout(new FlowLayout());
       add(b1);  add(b2);  add(b3);
    }
    public static void main(String args[])
    {
       Buttontest t= new Buttontest("Button");
        t.setSize(300,200);  t.show();
    }
}
```

5.3.5 Checkbox và RadioButton

- Checkbox được sử dụng khi người dùng tiến hành chọn một hay nhiều tùy chọn.
- Một radiobutton chỉ cho phép người dùng chọn một lựa chọn trong số nhiều lựa chọn.
- Sử dụng các constructor sau để tạo các checkbox trong Java:
  - **Checkbox()**: Tạo một checkbox trống.
  - **Checkbox(String text)**: Tạo một checkbox với nhãn được cho.
- Để tạo các radiobutton, đầu tiên chúng ta tạo đối tượng CheckboxGroup như sau:
  - **CheckboxGroup cg=new CheckboxGroup();**
- Sau đó chúng ta tạo các button:
  - **Checkbox male=new Checkbox(“male”, cg, true);**
  - **Checkbox female=new Checkbox(“female”, cg, false);**
- Sử dụng các phương thức setState() và getState() để thiết lập và nhận về trạng thái của checkbox.

5.3.5 Checkbox và RadioButton

- Chương trình demo 5.3.5

```
package checkboxandradiobutton;
import java.awt.*;
class Checkboxtest extends Frame
{
        Label l1=new Label("CheckBoxes");Checkbox b1=new Checkbox("red",true);
        Checkbox b2=new Checkbox("Green",false);Checkbox b3=new Checkbox("Blue",false);
        Label l2=new Label("Radiobuttons");CheckboxGroup cb=new CheckboxGroup();
        Checkbox b4=new Checkbox("small",cb,true);Checkbox b5=new Checkbox("medium",cb,false);
        Checkbox b6=new Checkbox("large",cb,false);
     public Checkboxtest(String title)
    {
        super(title);setLayout(new GridLayout(8,1));
        add(l1);add(b1);add(b2);add(b3);add(l2);add(b4);          add(b5);add(b6);
     }
     public static void main(String args[])
     {
        Checkboxtest t=new Checkboxtest("Checkbox and radiobutton");
        t.setSize(300,200); t.show();
     }
}
```

5.3.6 Danh sách chọn lựa (Choice List)

- Đôi khi cần trình bày một danh sách để người dùng chọn lựa. Người dùng có thể click vào một hay nhiều item từ danh sách. Một danh sách chọn lựa được tạo bằng cách sử dụng một số các chuỗi (String) hay các giá trị văn bản.
- Java hỗ trợ lớp Choice cho phép chúng ta tạo các danh sách chứa nhiều item. Khi danh sách vừa được tạo ra, nó sẽ rỗng.
  - **Choice colors=new Choice();**
- Mỗi thời điểm chỉ thêm được một item bằng cách sử dụng phương thức addItem:
  - **colors.addItem(“Red”);**
  - **colors.addItem(“Green”);**

5.3.6 Danh sách chọn lựa (Choice List)

- Chương trình demo 5.3.6

```
package choicelist;
import java.awt.*;
class Choicetest extends Frame
{
 Label l1=new Label("What is your favorite color");
 Choice colors=new Choice();
 public Choicetest(String title)
 {
  super(title);   setLayout(new FlowLayout());    add(l1);
  colors.addItem("White");    colors.addItem("Red");    colors.addItem("Orange");
  colors.addItem("Green");    colors.addItem("Yellow");    colors.addItem("Blue");
  colors.addItem("Black");    add(colors);
 }
 public static void main(String args[])
 {
   Choicetest t=new Choicetest("Choice list");
    t.setSize(300,200);      t.show();
 }
}
```

5.4 Trình quản lý cách trình bày (Layout manager)

- Layout manager điều khiển cách trình bày vật lý của các phần tử GUI như là button, textbox, option button v.v… Một layout manager tự động bố trí các thành phần này trong container.
- Các kiểu trình bày khác nhau:
  - Flow layout
  - Border layout
  - Card layout
  - Grid layout
  - GridBag Layout

5.4 Trình quản lý cách trình bày (Layout manager)

- Tất cả các thành phần mà chúng ta vừa tạo sử dụng layout manager mặc định.
- Layout manager này sẽ tự động xắp xếp các thành phần.
- Tất cả các thành phần được đặt trong một container, và được xắp xếp đến layout manager tương ứng.
- Layout manager được thiết lập bằng phương thức được gọi là ‘setLayout()’.

5.4.1 FlowLayout manager

- 'FlowLayout' là layout manager mặc định cho các applet và các panel. Các thành phần được xắp xếp từ góc trái trên đến góc phải dưới của màn hình. Khi một số thành phần được tạo, chúng được xắp xếp theo hàng, từ trái sang phải.
- Các constructor của FlowLayout:
  - FlowLayout mylayout = new FlowLayout() // constructor
  - FlowLayout exLayout=new FlowLayout(FlowLayout.RIGHT);
  - setLayout(exLayout); //setting the layout to Flowlayout
- Các điều khiển có thể được canh về bên trái, bên phải hay ở giữa. Để canh các điều khiển về bên phải, bạn sử dụng cú pháp sau:
  - **setLayout(new FlowLayout(FlowLayout.RIGHT));**

5.4.1 FlowLayout manager

- Chương trình demo 5.4.1

```
package flowlayout;
import java.awt.*;
class Fltest extends Frame
{
    Button b1=new Button("Center Aligned Button 1");
    Button b2=new Button("Center Aligned Button 2");
    Button b3=new Button("Center Aligned Button 3");
    public Fltest(String title)
    {
        super(title);          setLayout(new FlowLayout(FlowLayout.CENTER));
        add(b1);  add(b2);  add(b3);
    }
    public static void main(String args[])
    {
        Fltest t=new Fltest("Flow Layout");
        t.setSize(300,200);  t.show();
    }
}
```

5.4.2 BorderLayout Manager

- 'BorderLayout' là layout manager mặc định cho 'Window', 'Frame' và 'Dialog'.
- Layout này xắp xếp tối đa 5 thành phần trong một container. Những thành phần này có thể được đặt ở các hướng 'North', 'South', 'East', 'West' và 'Center' của container.
  - **NORTH** – Đặt ở đỉnh của container.
  - **EAST** – Đặt phía bên phải của container.
  - **SOUTH** – Đặt ở phía dưới của container.
  - **WEST** – Đặt phía bên trái của container.
  - **CENTER** – Đặt ở giữa của container.

5.4.2 BorderLayout Manager

- Để thêm một thành phần vào vùng ‘North’, sử dụng cú pháp sau:
  - **Button b1=new Button(“North Button”); // khai báo thành phần**
  - **setLayout(new BorderLayout()); // thiết lập layout**
  - **add(b1,BorderLayout.NORTH); // thêm thành phần vào layout**
- Các thành phần vẫn giữ nguyên vị trí tương đối của chúng kể cả khi container bị thay đổi kích thước.
- Các thành phần được đặt trong vùng ‘center’ sẽ được dàn đều vào những khu vực nằm giữa của container.
  - **add(b2,BorderLayout.CENTER); // thêm thành phần vào vùng ‘center’**

5.4.3 CardLayout Manager

- CardLayout có thể lưu trữ một ngăn xếp (stack) các layout. Mỗi layout giống như một bảng (card), thường là đối tượng Panel.
- Một thành phần độc lập như button sẽ điều khiển cách trình bày các bảng ở lớp trên cùng.

5.4.3 CardLayout Manager

- Trước tiên, chúng ta bố trí tập hợp các thành phần được yêu cầu trên các panel tương ứng. Mỗi panel sẽ được bố trí vào các layout khác nhau. Cho ví dụ:
  - **panelTwo.setLayout(new GridLayout(2,1));**
- Panel chính sẽ chứa những panel này. Thiết lập layout của panel chính là Cardlayout như sau:
  - **CardLayout card=new CardLayout();**
  - **panelMain.setLayout(card);**
- Bước kế tiếp là thêm các panel khác vào panel chính:
  - **panelMain.add("Red Panel", panelOne);**
  - **panelMain.add("Blue Panel", panelTwo);**
- Phương thức 'add()': Tham số String làm nhãn của panel và tham số thứ hai là tên đối tượng Panel.

5.4.4. GridLayout Manager

- GridLayout' trợ giúp việc chia container vào trong ô lưới.
- Các thành phần được đặt trong các dòng và các cột. Mỗi khung lưới nên chứa ít nhất một thành phần.
- Một khung lưới được sử dụng khi tất cả các thành phần có cùng kích thước.
- Constructor GridLayout được tạo như sau:
  - **Gridlayout g1=new GridLayout(4,3);**
  - 4 là số dòng và 3 là số cột.

5.4.4. GridLayout Manager

Chương trình demo 5.4.4

```
package gridlayout;
import java.awt.*;
class Gltest extends Frame
{
	Button btn[];
	String str[]={"1", "2", "3", "4", "5", "6", "7", "8", "9"};
	public Gltest(String title)
	{
		super(title);
		setLayout(new GridLayout(3,3));
		btn=new Button[str.length];
		for (int I=0; I<str.length;I++)
		{
			btn[I]=new Button(str[I]);
			add(btn[I]);
		}
	}
	public static void main(String args[])
	{
		Gltest t=new Gltest("Grid Layout");
		t.setSize(300,200);
		t.show();
	}
}
```

5.5 Xử lý các sự kiện

- Các hệ thống GUI xử lý các tương tác người dùng với sự trợ giúp của mô hình event-driven.
- Tương tác: Di chuyển chuột, nhấn phím, nhả phím v.v…Tất cả các thao tác này thiết lập một sự kiện tương ứng.
- Việc xử lý những sự kiện này phụ thuộc vào ứng dụng. AWT xử lý một vài sự kiện.

5.5 Xử lý các sự kiện

- Ứng dụng cần đăng ký một hàm xử lý sự kiện với một đối tượng tương ứng.
- Hàm xử lý sự kiện này sẽ được gọi bất cứ khi nào sự kiện tương ứng phát sinh.
- Trong tiến trình này, ứng dụng cho phép ta đăng ký các handler, hay gọi là listener với các đối tượng.
- Những handler này tự động được gọi khi một sự kiện thích hợp phát sinh.

5.5 Xử lý các sự kiện

- Các bước xây dựng sự kiện:
- Thực hiện giao diện listener thích hợp:

  public class MyApp extends Frame implements ActionListener
- Xác định các thành phần tạo ra sự kiện: button, label, menu item, hay window.
- Đăng ký một thành phần với listener, ta có thể sử dụng:

  exitbtn.addActionListener(This);
- Xác định tất cả các sự kiện được xử lý: Có thể là một 'ActionEvent' nếu một button được click hay một 'mouseEvent' nếu như chuột được kéo đi.
- Thi hành các phương thức của listener và viết hàm xử lý sự kiện tương ứng với các phương thức.

5.5 Xử lý các sự kiện

- Bảng sau đây chỉ ra các sự kiện khác nhau:

Lớp sự kiện	Mô tả
ActionEvent	Phát sinh khi một button được nhấn, một item trong danh sách chọn lựa được nhắp đôi hay một menu được chọn.
AdjustmentEvent	Phát sinh khi một thanh scrollbar được sử dụng.
ComponentEvent	Phát sinh khi một thành phần được thay đổi kích thước, được di chuyển, bị ẩn hay làm cho hoạt động được.
FocusEvent	Phát sinh khi một thành phần mất hay nhận focus từ bàn phím.
ItemEvent	Phát sinh khi một menu item được chọn hay bỏ chọn; hay khi một checkbox hay một item trong danh sách được click.
WindowEvent	Phát sinh khi một cửa sổ được kích hoạt, được đóng, được mở hay thoát.
TextEvent	Phát sinh khi giá trị trong thành phần text field hay text area bị thay đổi.
MouseEvent	Phát sinh khi chuột di chuyển, được click, được kéo hay bị thả ra.
KeyEvent	Phát sinh khi input được nhận từ bàn phím.

5.5 Xử lý các sự kiện

- Các giao diện tương ứng được thi hành để xử lý một trong số những sự kiện này là:
  - ActionListener
  - AdjustmentListener
  - ComponentListener
  - FocusListener
  - ItemListener
  - WindowListener
  - TextListener
  - MouseListener
  - MouseMotionListener
  - KeyListener

5.5 Xử lý các sự kiện

- Chương trình demo 5.5 minh họa cách xây dựng và sử dụng sự kiện

5.6 Thực đơn (menu)

- Ngôn ngữ Java có một tập hợp các lớp đối tượng để tạo các menu.
- Có hai loại menu – pull down và pop-up.
- Menu làm cho ứng dụng dễ sử dụng hơn.
- Chỉ duy nhất một thanh menubar được đặt trong một frame.
- Menubar là một thanh nằm ngang được đặt tại đỉnh của frame.
- Một menu độc lập có thể chứa các submenu gọi là menuitem.
- Java cung cấp các checkbox menuitem, chúng có thể được bật hay mở, phụ thuộc vào trạng thái.

5.6 Thực đơn (menu)

- Chương trình demo 5.6 minh họa cách xây dựng và sử dụng menu