LẬP TRÌNH JAVA NÂNG CAO

Chương 01: Lập trình giao diện

Lê Tân

Bộ môn: Lập trình máy tính

Nội dung chương 01

- Sơ đồ phân cấp lớp GUI
- JFrames
- Layout Managers
- Drawing on JPanels: Lines, Rectangles, Ovals, Arcs, Polygons
- Event-Driven Programming: Event Source, Listener, Listener Interface
- Nút nhấn và menu
- JCheckbBox và JRadioButton
- Các lớp văn bản và JScrollBar

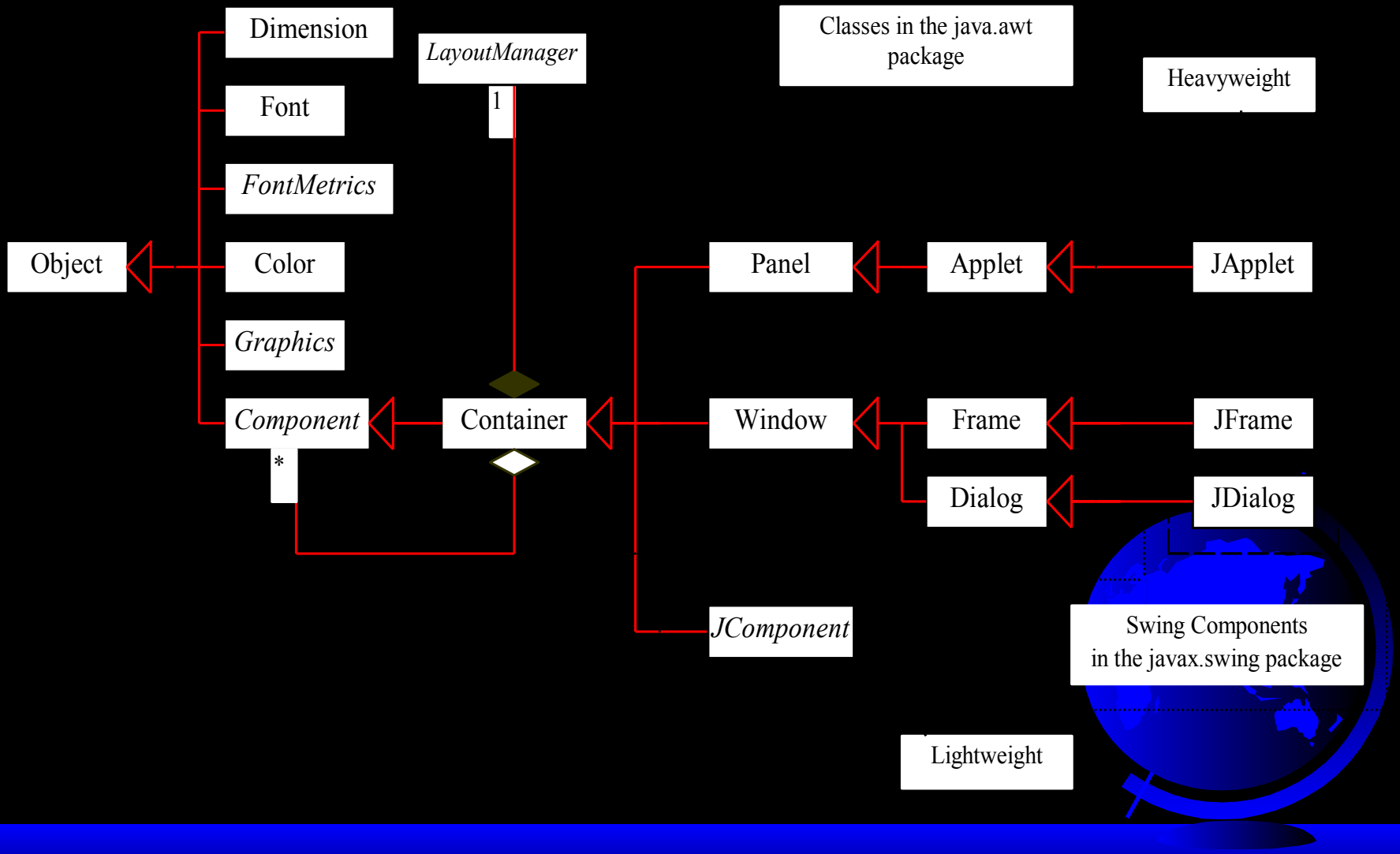
Các thành phần GUI

- Các đối tượng GUI: button, label, text field, check box, radio button, combo box, …
- Mỗi loại đối tượng được xác định trong 1 lớp: JButton, JLabel, JTextField, JCheckBox, JRadioButton, JComboBox, …
- Mỗi lớp thành phần GUI có một số constructor để tạo các đối tượng thành phần GUI.

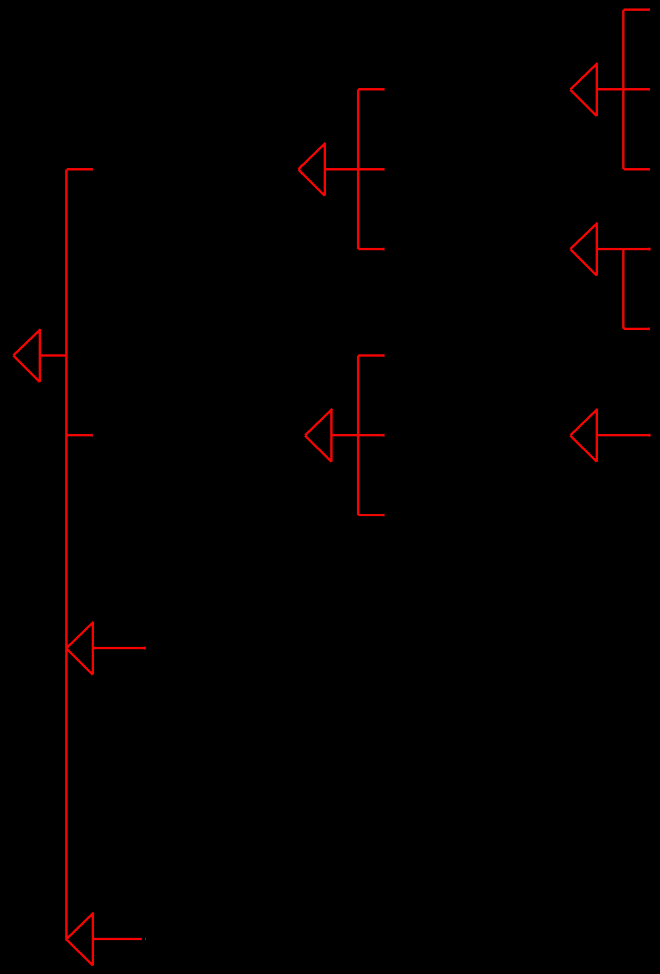
Swing vs. AWT

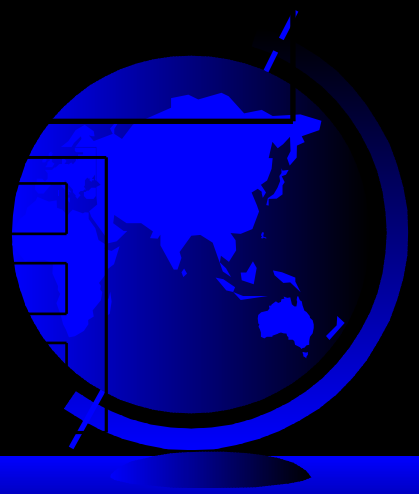
- AWT: Abstract Windows Toolkit:
  - Java 1
  - Được gắn với platform xác định
  - Thích hợp với việc phát triển các ứng dụng GUI đơn giản.
- Swing components:
  - Java 2
  - Không gắn với platform cố định
  - Mạnh, đa năng, linh hoạt

Sơ đồ phần cấp lớp GUI (Swing)



JComponent





Các lớp GUI: nhóm container

- Được dùng để chứa các thành phần khác.
- Các lớp container (Swing):
  - Container
  - JFrame
  - JDialog
  - JApplet
  - JPanel

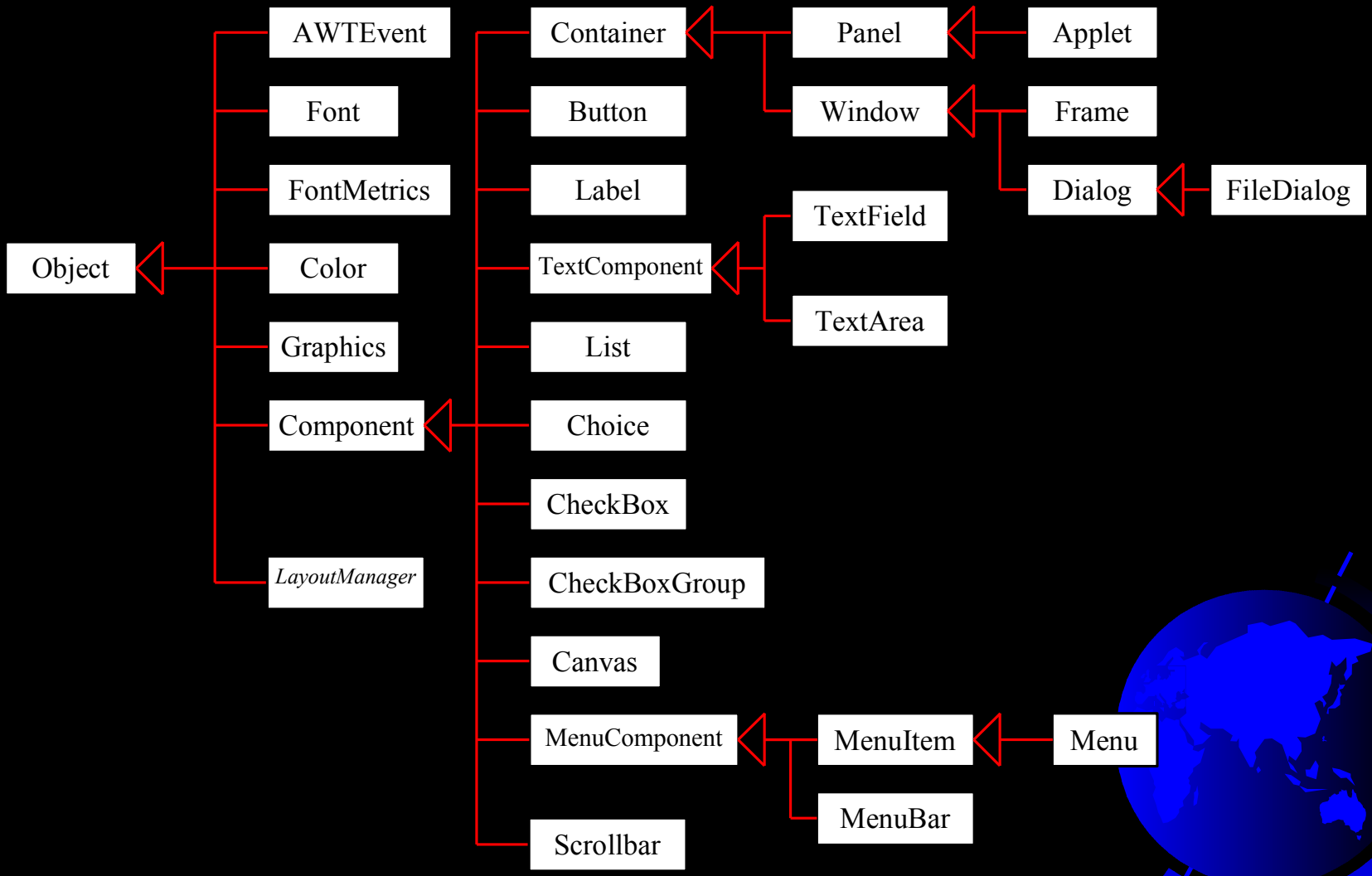
Các lớp GUI: nhóm component

- Gồm các subclass của lớp JComponent.
- Các lớp GUI component (Swing):
  - JButton
  - JLabel
  - JTextField
  - JTextArea
  - JComboBox
  - JList
  - JRadioButton
  - JMenu
  - …

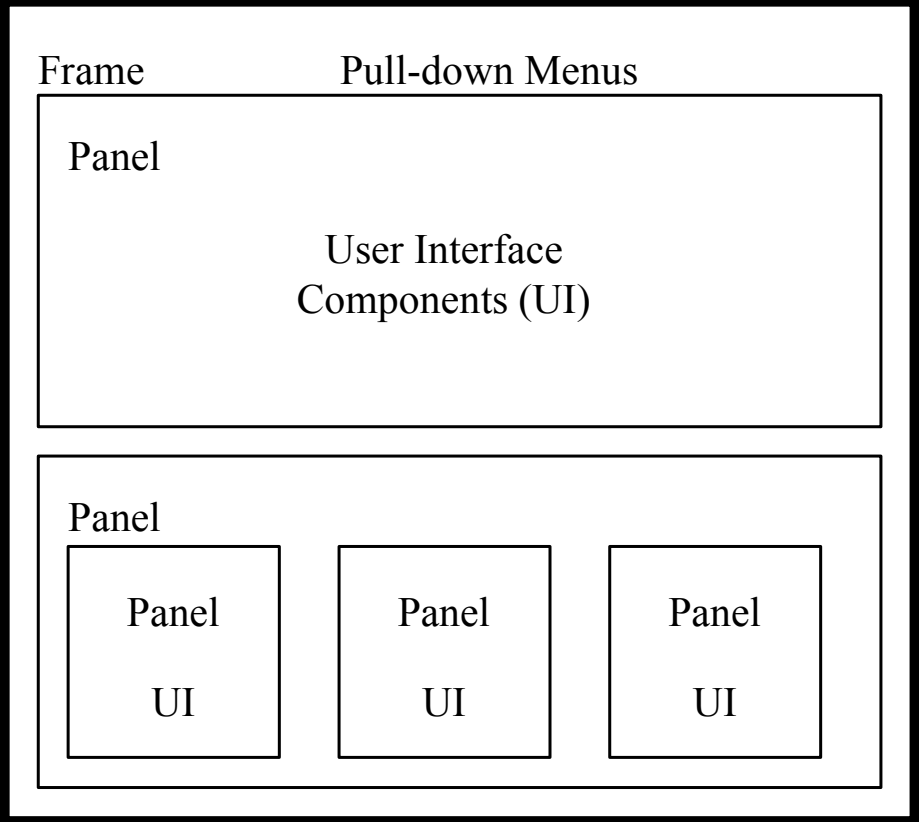
Các lớp GUI: nhóm helper

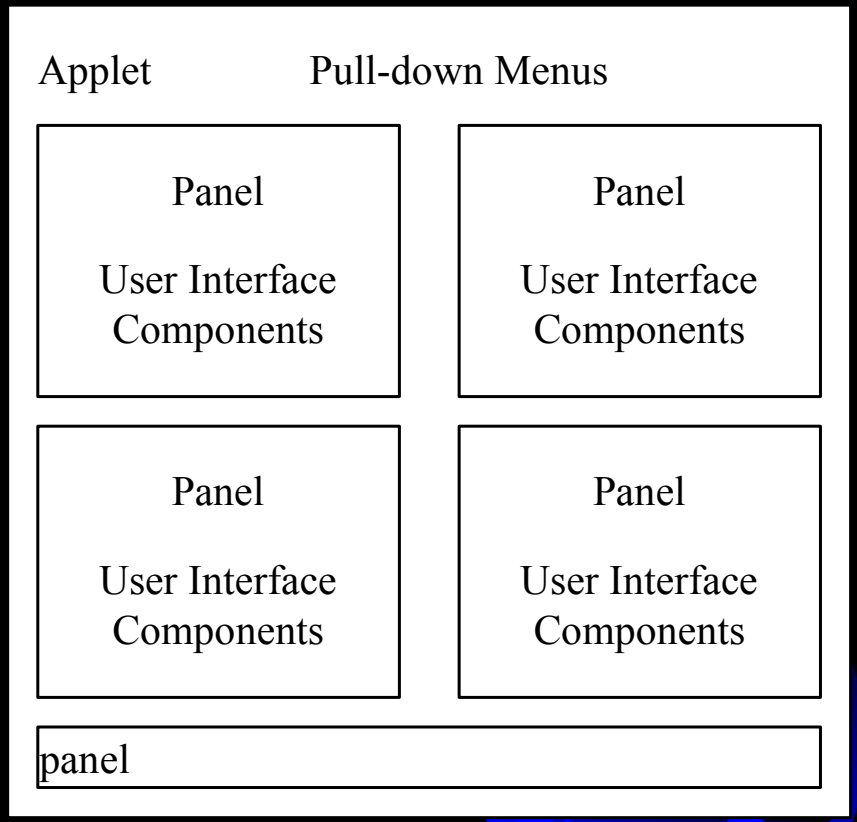
- Được các component và container dùng để vẽ và đặt các đối tượng.
- Các lớp helper (Swing):
  - Graphics
  - Color
  - Font
  - FontMetrics
  - Dimension
  - LayoutManager

AWT (Optional)



Các thành phần giao diện người sử dụng





Frames

- Frame là một cửa sổ không chứa trong cửa sổ khác.
- Frame là nền tảng để chứa các thành phần GUI khác trong các ứng dụng Java GUI.
- Trong các chương trình Swing GUI, sử dụng lớp JFrame để tạo các cửa sổ.

Tạo Frame

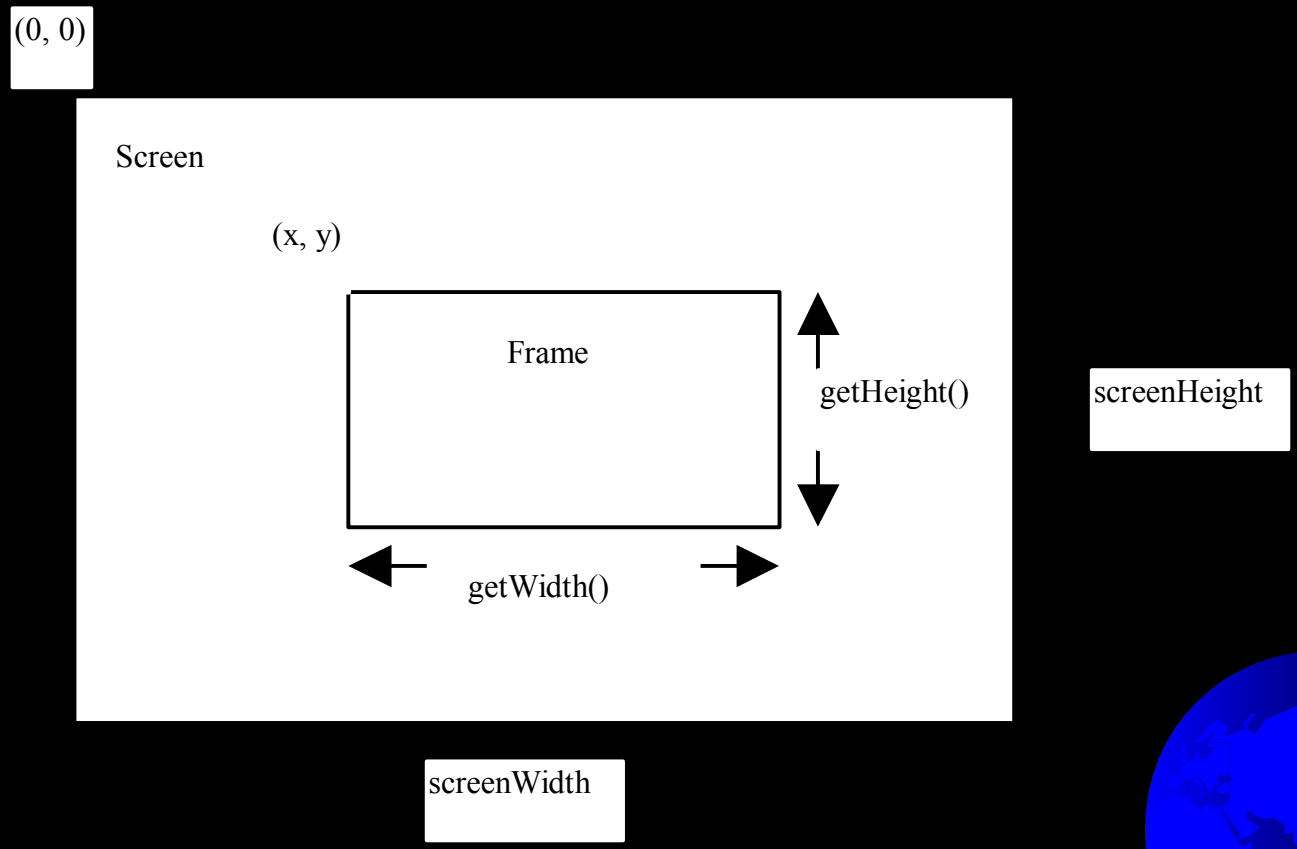
```
import javax.swing.*;
public class MyFrame {
  public static void main(String[] args) {
    JFrame frame = new JFrame("Test Frame");
    frame.setSize(400, 300);
    frame.setVisible(true);
    frame.setDefaultCloseOperation(
      JFrame.EXIT_ON_CLOSE);
  }
}
```

Chú ý: Chạy chương trình cần JDK 1.3 hoặc cao hơn

Căn giữa Frame

- Mặc định, frame được hiển thị ở góc trên bên trái của màn hình.
- Để hiển thị frame ở một vị trí xác định, sử dụng phương thức `setLocation(x, y)` trong lớp `JFrame`.
- Phương thức này đặt góc trái trên của frame tại vị trí có tọa độ (x, y).

Căn giữa Frame (tiếp)



Đưa các thành phần vào trong Frame

```
// Dua nut bam vao trong frame
frame.getContentPane().add(
  new JButton("OK"));
```

LƯU Ý

Content pane là một lớp con của Container. Câu lệnh ở slide trước tương đương với 2 câu lệnh sau:

```
Container container = frame.getContentPane();
container.add(new JButton("OK"));
```

Content pane được sinh ra khi một đối tượng JFrame được tạo. Đối tượng JFrame sử dụng content pane để chứa các thành phần trong frame.

Layout Managers

- Các layout manager của Java cung cấp cơ chế để tự động ánh xạ các thành phần GUI của bạn trên tất cả các hệ thống cửa sổ.
- Các thành phần GUI được đặt trong các container. Mỗi container có một layout manager để sắp xếp các thành phần đó.

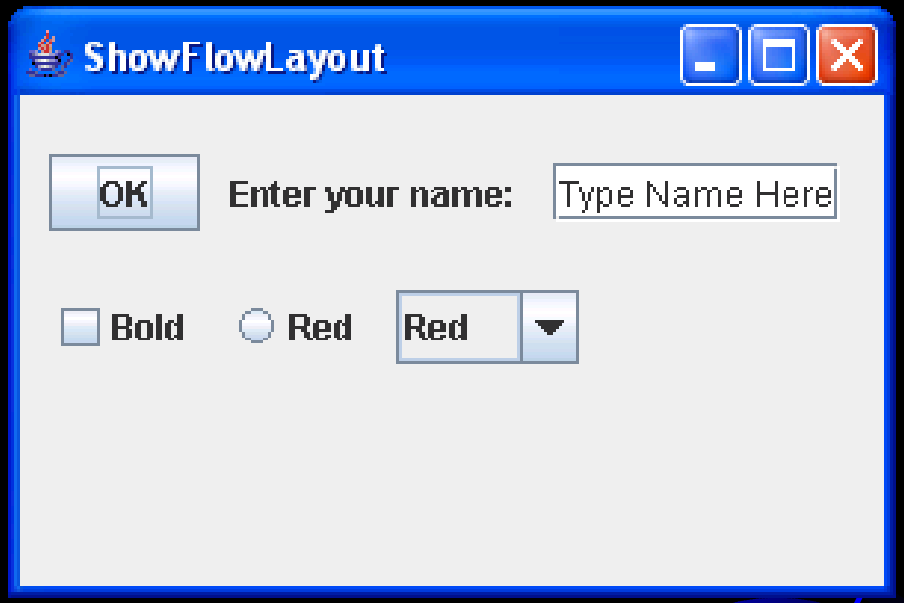
Thiết lập Layout Manager

```
LayoutManager layMan = new
  XLayout();
container.setLayout(layMan);
```

- XLayout:
  - FlowLayout
  - GridLayout
  - BorderLayout

Ví dụ 9.1: FlowLayout Manager

Các thành phần được sắp xếp trong container từ trái sang phải, từ trên xuống dưới theo thứ tự chúng được đưa vào.



FlowLayout Constructors

- public FlowLayout(int align, int hGap, int vGap)

  Xây dựng một FlowLayout mới có cách sắp hàng (alignment), khoảng trống ngang (horizontal gap), khoảng trống dọc (vertical gap) xác định. Các khoảng trống giữa các thành phần được tính bằng pixel.

- public FlowLayout(int alignment)

  Xây dựng một FlowLayout mới có alignment xác định, khoảng trống ngang và dọc đều có mặc định bằng 5 pixel.

- public FlowLayout()

  Xây dựng một FlowLayout mới có cách sắp hàng mặc định căn giữa và khoảng trống ngang và dọc mặc định bằng 5 pixel.

Ví dụ 9.2: GridLayout Manager

GridLayout manager sắp xếp các thành phần trong một lưới (ma trận) với số hàng và số cột được xác định bởi constructor. Các thành phần được đặt trong lưới từ trái sang phải, từ trên xuống dưới.

GridLayout Constructors

- public GridLayout(int rows,int columns)

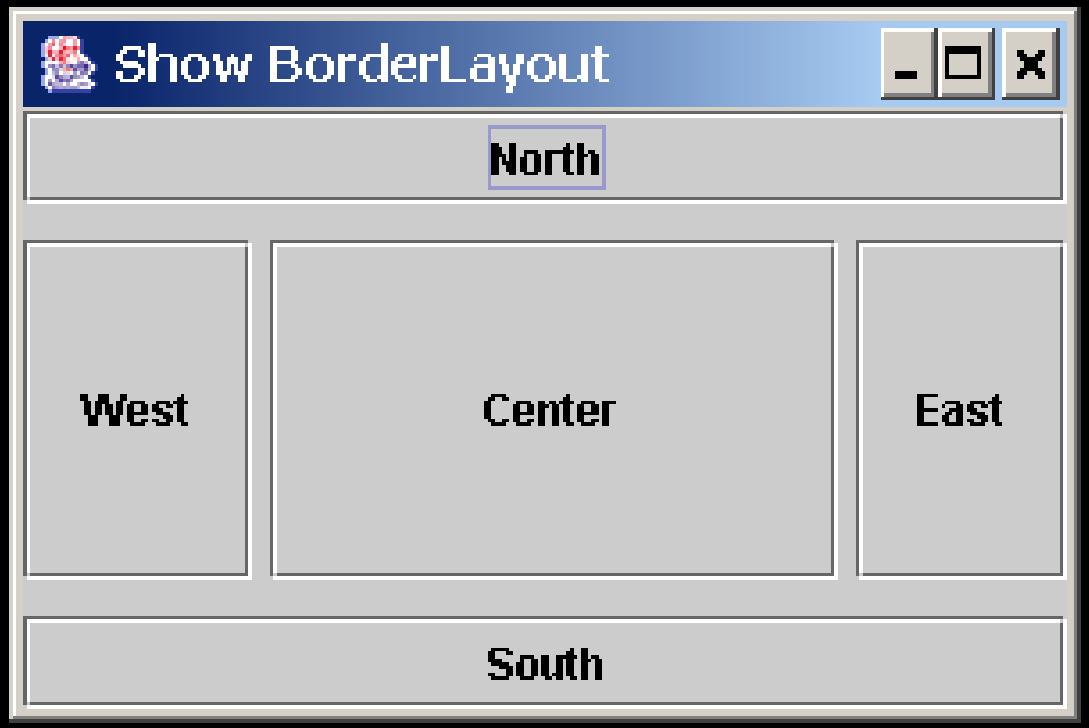
  Xây dựng một GridLayout mới có số hàng và số cột xác định.

- public GridLayout(int rows, int columns, int hGap, int vGap)

  Xây dựng một GridLayout mới có số hàng và số cột xác định, và khoảng trống ngang và dọc giữa các thành phần được xác định.

Ví dụ 9.3: BorderLayout Manager

- BorderLayout manager chia container thành 5 khu vực: East, South, West, North, và Center.



Ví dụ 9.3 (tiếp)

- Các thành phần được đưa vào BorderLayout bằng phương thức add:

  add(Component, constraint)

- constraint:
  - BorderLayout.EAST,
  - BorderLayout.SOUTH,
  - BorderLayout.WEST,
  - BorderLayout.NORTH,
  - BorderLayout.CENTER.

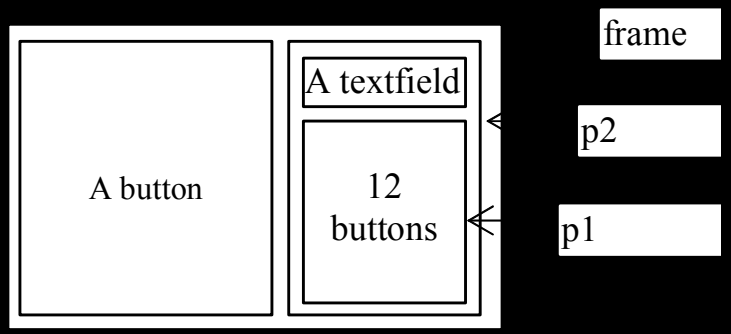
Sử dụng Panel làm Container

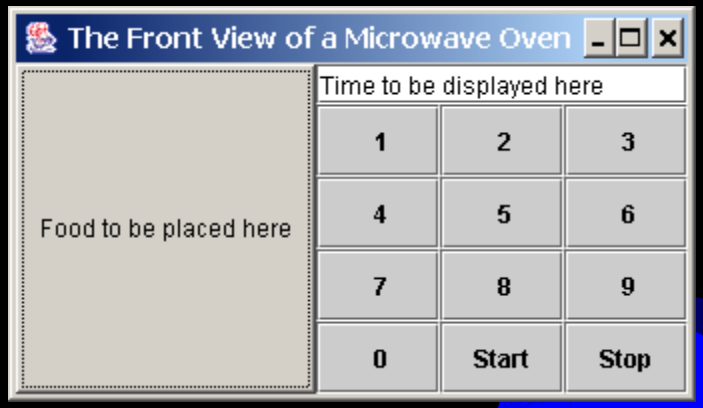
- Các panel đóng vai trò như các container nhỏ để nhóm các thành phần GUI.
- Bạn nên đặt các thành phần GUI trong các panel và đặt các panel trong một frame, hoặc cũng có thể đặt panel trong panel.

```
JPanel p = new JPanel();
p.add(new JButton("OK");
frame.getContentPanel().add(p);
```

Ví dụ 9.4: Panel

Chương trình tạo một giao diện cho lò vi sóng, sử dụng các panel để tổ chức các thành phần.





Vẽ trên Panel

- JPanel còn có thể được sử dụng để vẽ đồ họa, văn bản và cho phép tương tác với người sử dụng.

- Để vẽ trên panel:
  - Tạo một lớp subclass của `JPanel`
  - Chồng phương thức `paintComponent`.
  - Sau đó có thể hiển thị các chuỗi ký tự, vẽ các khối hình học và hiển thị ảnh trên panel.

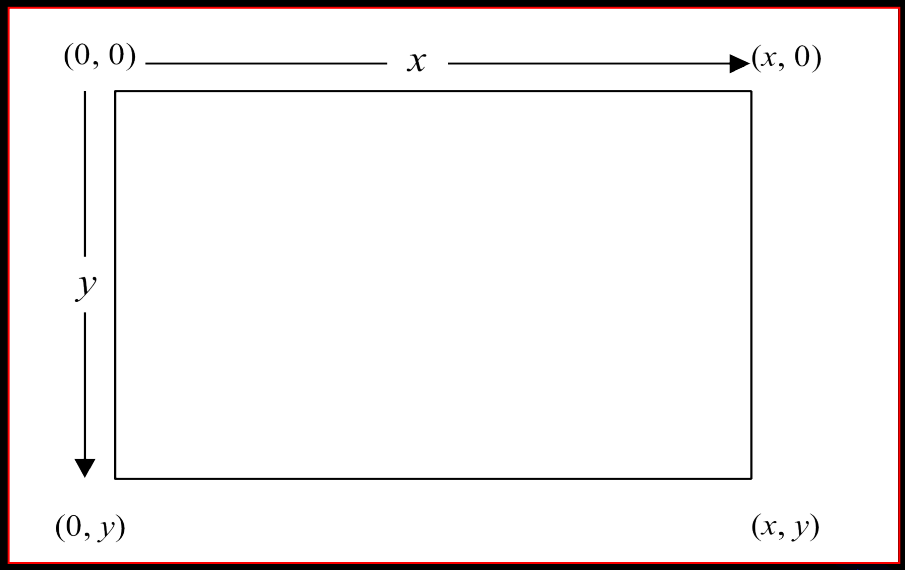
Vẽ trên Panel (tiếp)

```
public class DrawMessage extends JPanel {
  /** Main method */
  public static void main(String[] args) {
    JFrame frame = new JFrame("DrawMessage");
    frame.getContentPane().add(new
DrawMessage());

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CL
OSE);
    frame.setSize(300, 200);
    frame.setVisible(true);
  }

  /** Paint the message */
  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    g.drawString("Welcome to Java!", 40, 40);
```

Vẽ trên Panel (tiếp)



LƯU Ý

- Lớp Graphics là một lớp trừu tượng để hiển thị hình vẽ và ảnh trên màn hình trên các platform khác nhau.
- Lớp Graphics gói gọn các chi tiết platform và cho phép bạn vẽ các thứ theo cách giống nhau không liên quan đến các platform cụ thể.
- Lời gọi super.paintComponent(g) là cần thiết để đảm bảo rằng vùng hiển thị được xóa sạch trước khi hiển thị một bản vẽ.

LƯU Ý

Để vẽ các hình, thông thường bạn tạo một lớp con của JPanel và chồng phương thức paintComponent để "nói" cho hệ thống phải vẽ như thế nào. Thực tế bạn có thể vẽ các thứ trên bất kỳ thành phần GUI nào.

Lớp Color

- Bạn có thể thiết lập màu cho các thành phần GUI bằng cách sử dung lớp java.awt.Color. Các màu được tạo từ 3 màu cơ bản là red, green, blue; mỗi màu đó được biểu diễn bởi một giá trị byte (0-255) miêu tả cường độ. Đây được gọi là hệ màu RGB (*RGB model*).

```
Color c = new Color(r, g, b);
```

r, g, b xác định một màu được tạo bởi các thành phần tương ứng red, green, blue.

Ví dụ:

```
Color c = new Color(228, 100, 255);
```

Thiết lập màu

- Bạn có thể sử dụng các phương thức sau để thiết lập màu background và foreground của các thành phần:

```
setBackground(Color c)

setForeground(Color c)
```

Ví dụ:

```
JButton jbtOK = new JButton();
jbtOK.setBackground(Color.yellow);
jbtOK.setForeground(new Color(255,0,0));
```

Lớp Font

```
Font myFont = Font(name, style, size);
```

Ví dụ:

```
Font font1 = new Font("SansSerif", Font.BOLD, 16);
Font font2 = new Font("Serif",
                      Font.BOLD+Font.ITALIC, 12);
```

Tìm tất cả tên Font khả dụng

```
import java.awt.GraphicsEnvironment;

public class testAllFonts {
  public static void main(String[] args) {
  GraphicsEnvironment e =

  GraphicsEnvironment.getLocalGraphicsEnvironment();
  String[] fontnames =
  e.getAvailableFontFamilyNames();
  for (int i = 0; i < fontnames.length; i++)
  System.out.println(fontnames[i]);
  }
}
```

Thiết lập Font

```
public void paint(Graphics g) {
  Font myFont = new Font("Times", Font.BOLD, 18);
  g.setFont(myFont);
  g.drawString("Welcome to Java", 20, 40);

  //set a new font
  g.setFont(new Font("Courier",Font.BOLD+Font.ITALIC,16));
  g.drawString("Welcome to Java", 20, 70);
}
```

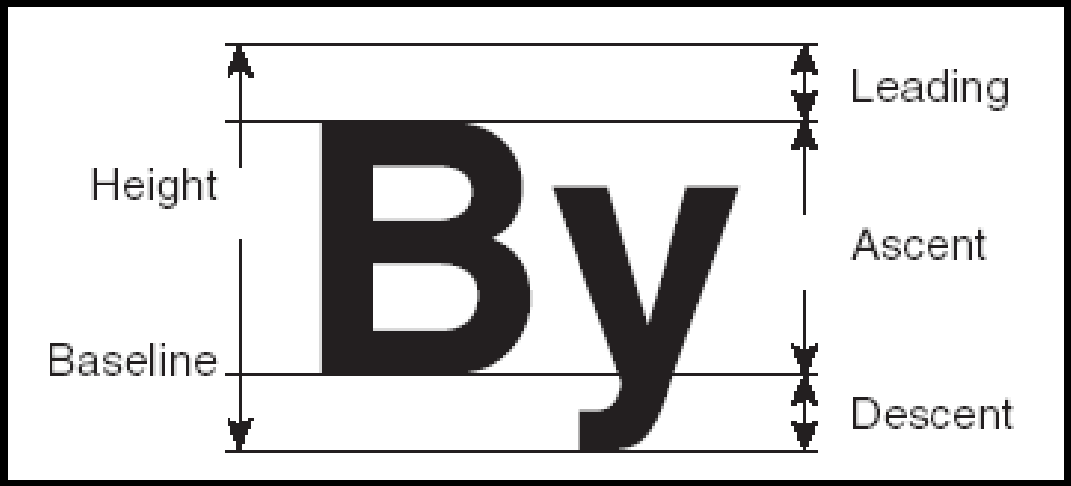
Lớp FontMetrics

- Bạn có thể hiển thị một chuỗi ký tự tại vị trí bất kỳ trong panel bằng cách sử dụng lớp FontMetrics.
- Để nhận đối tượng FontMetrics cho một font xác định, sử dụng phương thức getFontMetrics:

```
public void paint(Graphics g) {
   g.getFontMetrics(Font f); // hoặc
   g.getFontMetrics();
}
```

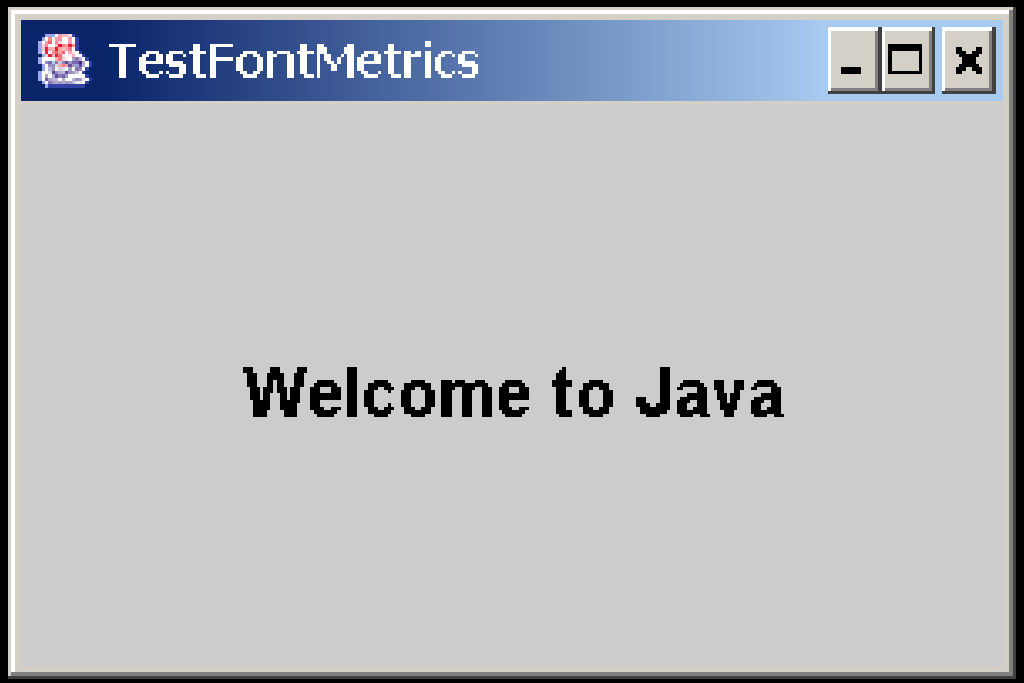
Các phương thức lấy thuộc tính chuỗi của lớp `FontMetrics`

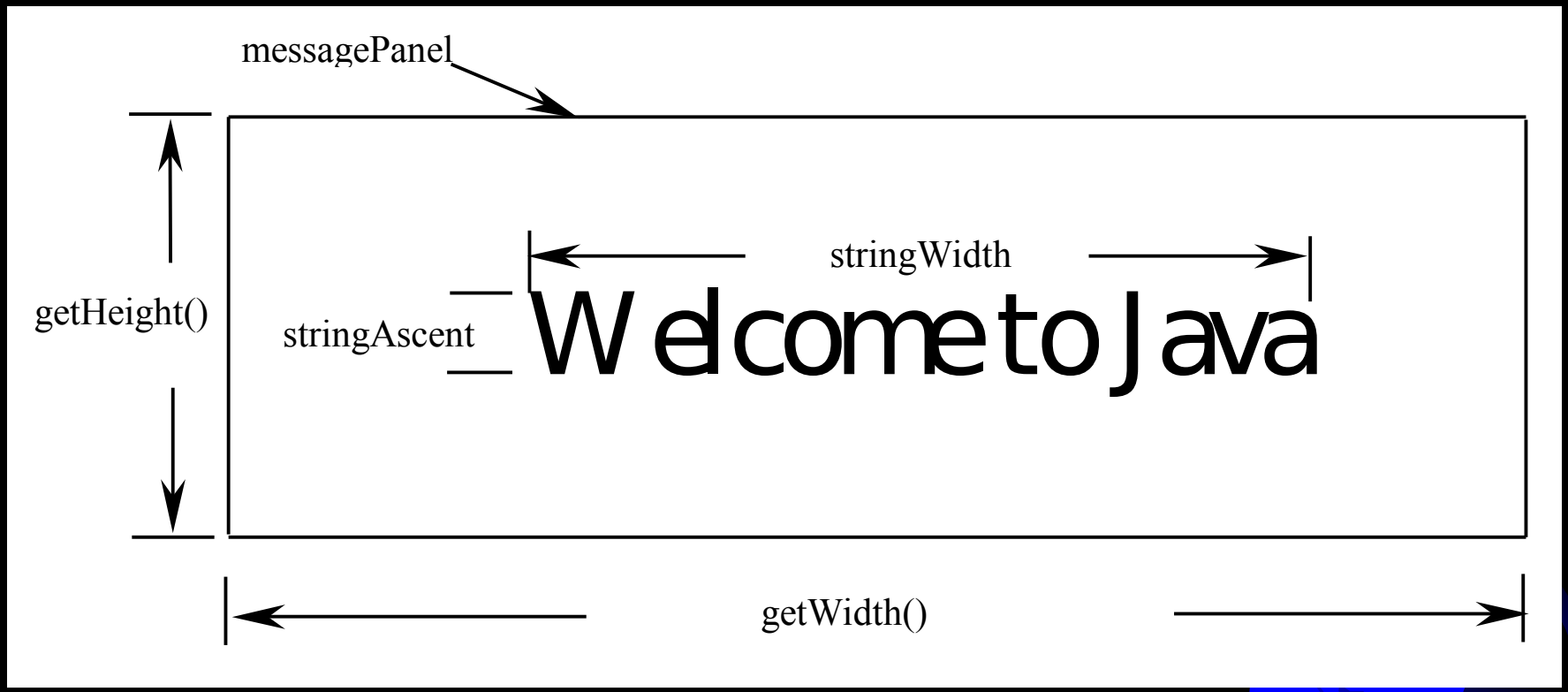
- `public int getAscent()`
- `public int getDescent()`
- `public int getLeading()`
- `public int getHeight()`
- `public int stringWidth(String str)`



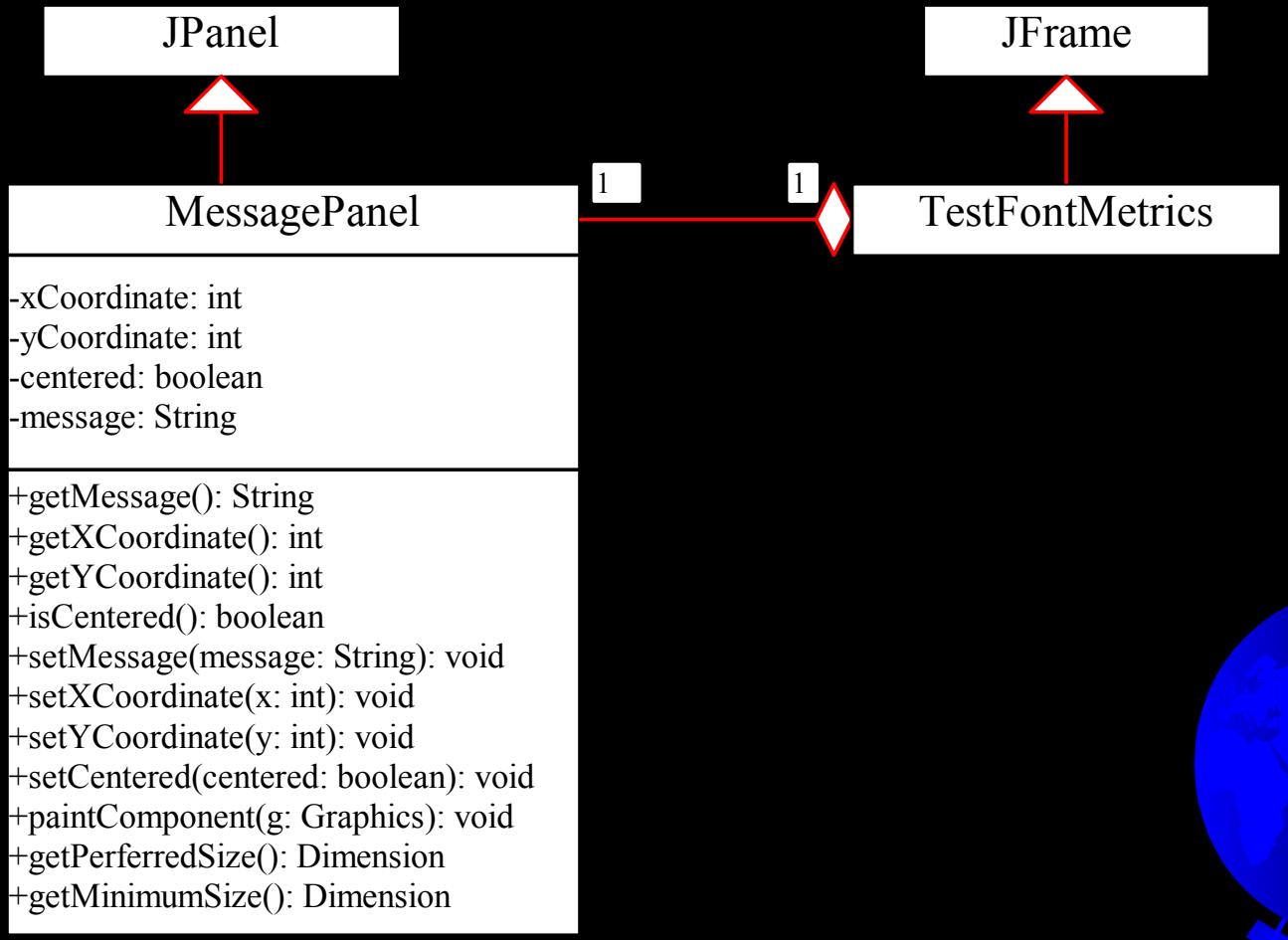
Ví dụ 9.5: Sử dụng FontMetrics

- Mục tiêu: Hiển thị "Welcome to Java" căn giữa trong frame.




messagePanel
getHeight()
stringAscent
stringWidth
Welcome to Java
getWidth()

Ví dụ 9.5 (tiếp)

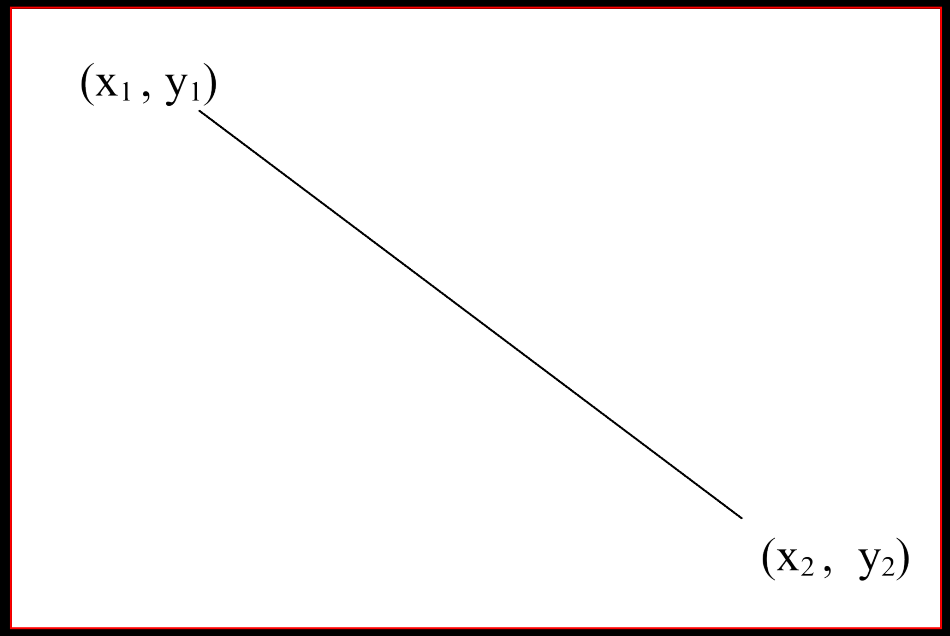

JPanel
JFrame
MessagePanel
1
1
TestFontMetrics
-xCoordinate: int
-yCoordinate: int
-centered: boolean
-message: String
+getMessage(): String
+getXCoordinate(): int
+getYCoordinate(): int
+isCentered(): boolean
+setMessage(message: String): void
+setXCoordinate(x: int): void
+setYCoordinate(y: int): void
+setCentered(centered: boolean): void
+paintComponent(g: Graphics): void
+getPerferredSize(): Dimension
+getMinimumSize(): Dimension

Vẽ các hình hình học trên Panel

- Vẽ đường thẳng
- Vẽ hình chữ nhật
- Vẽ hình bầu dục
- Vẽ cung tròn
- Vẽ đa giác

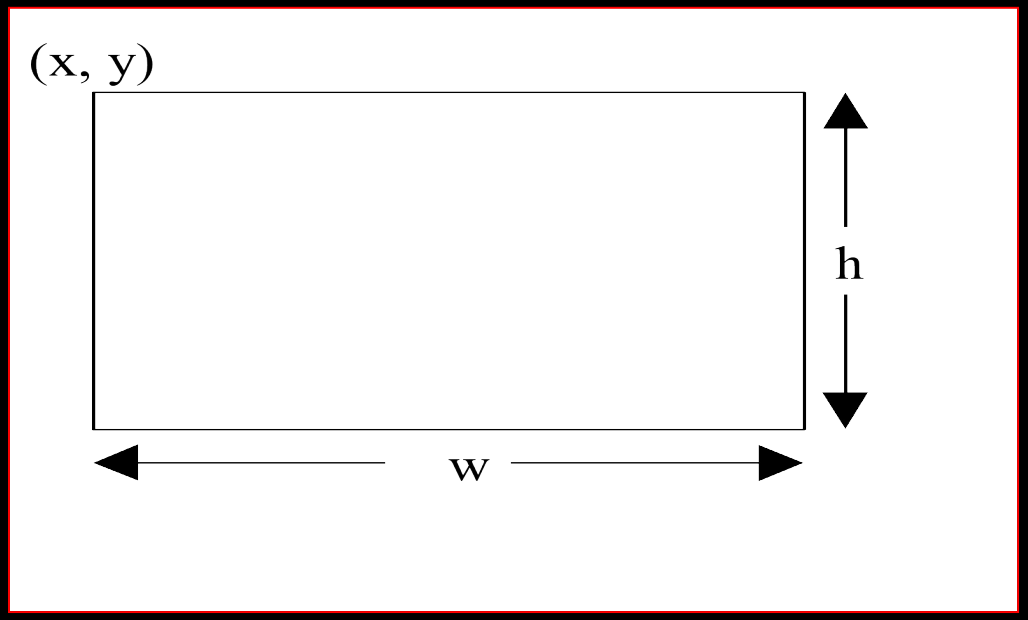
Vẽ đường thẳng

```
drawLine(x1, y1, x2, y2);
```



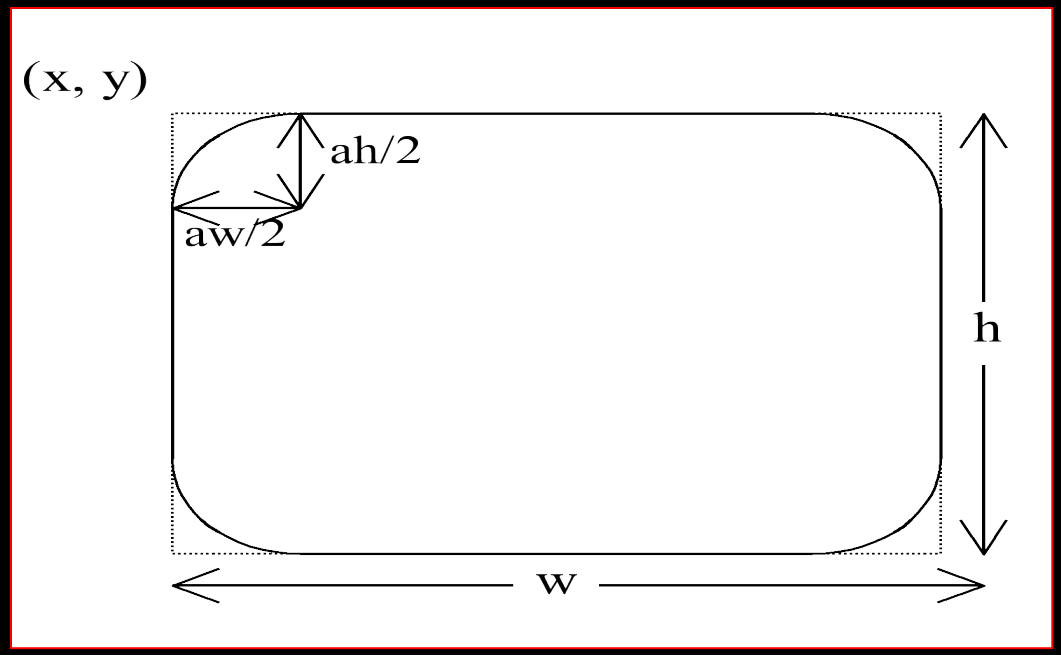
Vẽ hình chữ nhật

- `drawRect(x, y, w, h);`
- `fillRect(x, y, w, h);`



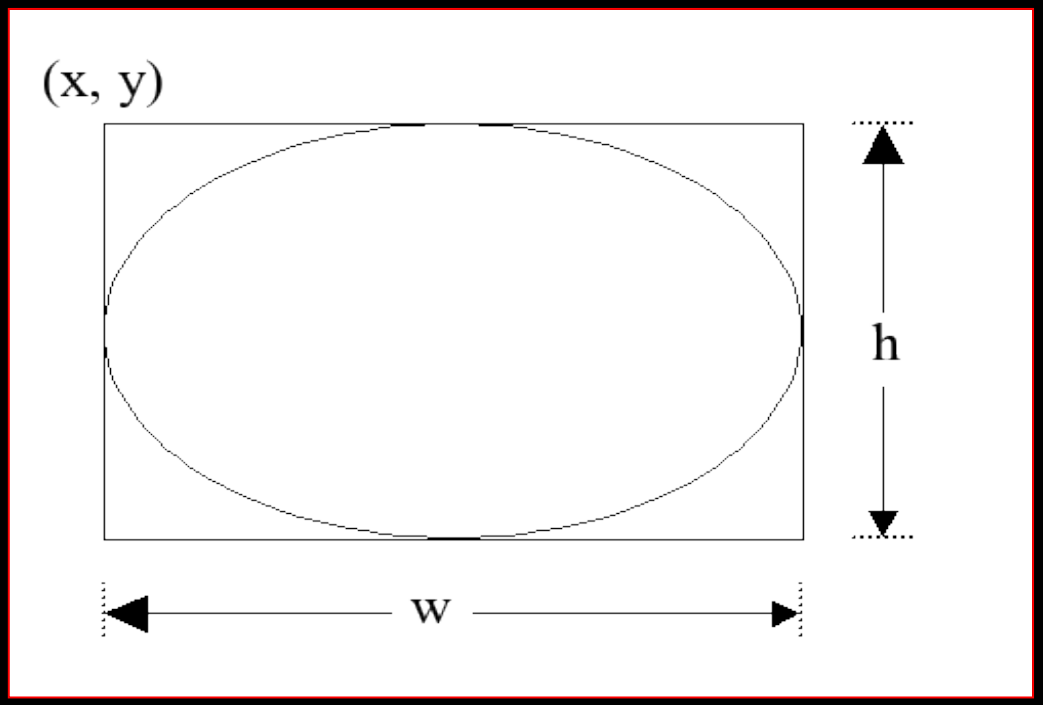
Vẽ hình chữ nhật góc tròn

- `drawRoundRect(x, y, w, h, aw, ah);`
- `fillRoundRect(x, y, w, h, aw, ah);`



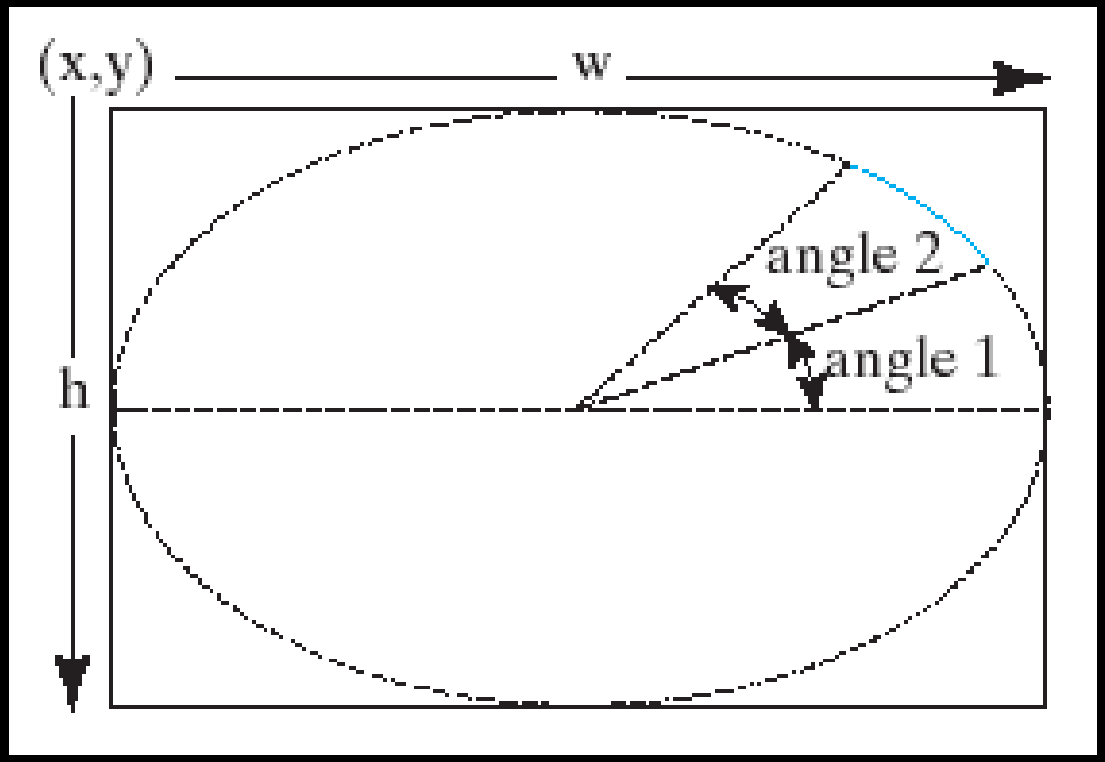
Vẽ hình bầu dục

- `drawOval(x, y, w, h);`
- `fillOval(x, y, w, h);`



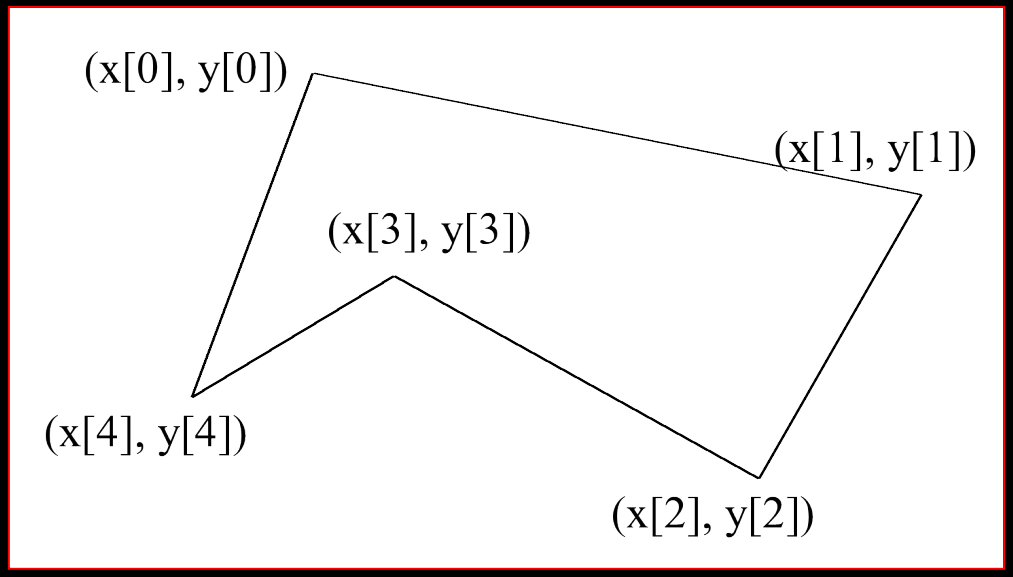
Vẽ cung tròn

- `drawArc(x, y, w, h, angle1, angle2);`
- `fillArc(x, y, w, h, angle1, angle2);`



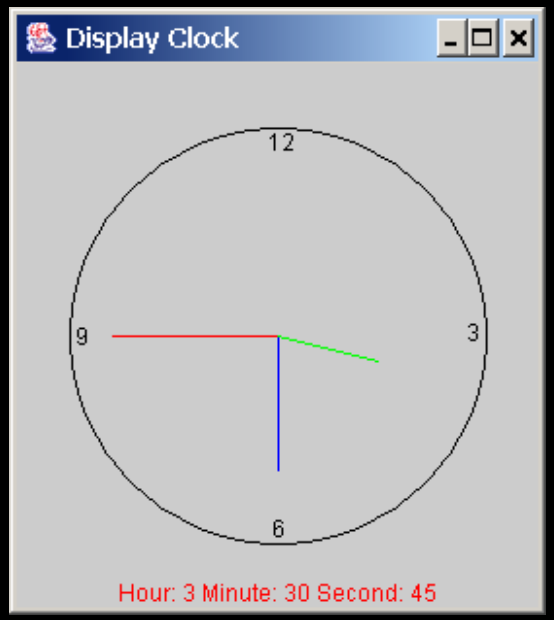
Vẽ đa giác

```
int[] x = {40, 70, 60, 45, 20};
int[] y = {20, 40, 80, 45, 60};
g.drawPolygon(x, y, x.length);
g.fillPolygon(x, y, x.length);
```



Ví dụ 9.6: Vẽ chiếc đồng hồ

- Mục tiêu: Sử dụng các phương thức vẽ và lượng giác để vẽ một chiếc đồng hồ hiển thị giờ, phút, giây hiện tại trong một frame.



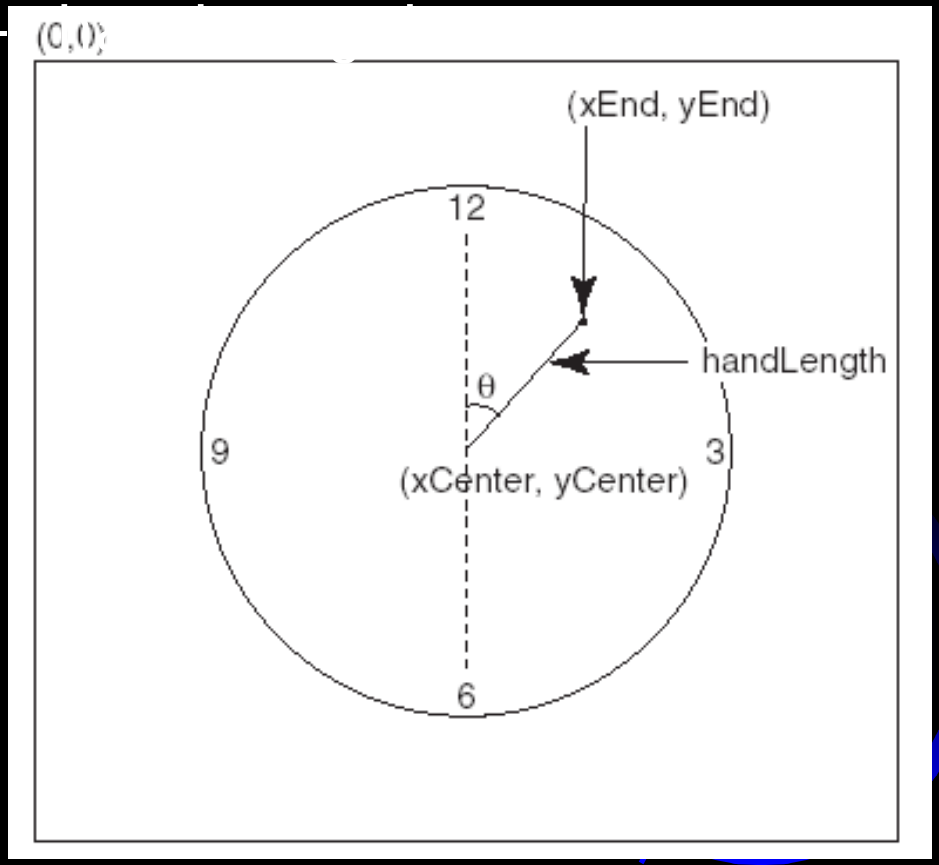
Ví dụ 9.6 (tiếp)

```
xEnd = xCenter + handLength ×
sin(θ)

yEnd = yCenter -
cos(θ)
```

Vì 1 phút có 60 giây, góc của kim giây là:

```
second × (2π/60)
```



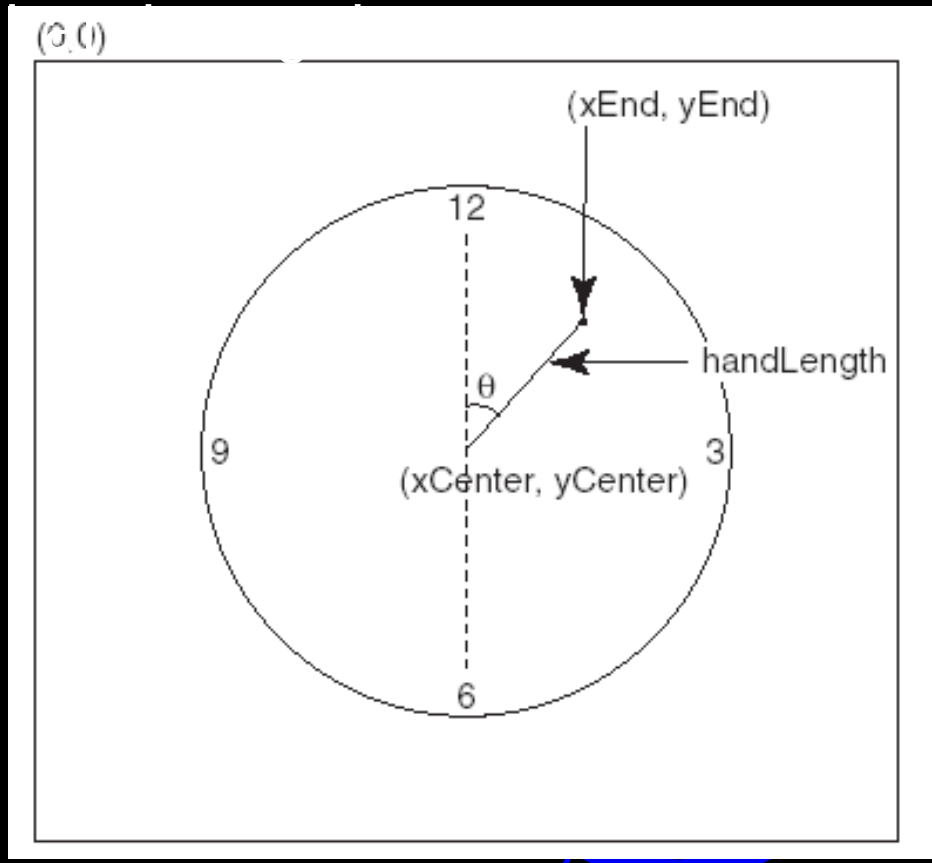
Ví dụ 9.6 (tiếp)

```
xEnd = xCenter + handLength ×
sin(θ)

yEnd = yCenter -
cos(θ)
```

Vị trí của kim phút được xác định bởi phút và giây theo công thức minute + second/60. Ví dụ, nếu thời gian là 3 phút 30 giây. Tổng số phút là 3.5. Vì 1 giờ có 60 phút, góc của kim phút là:

`(minute + second/60) × (2π/60)`



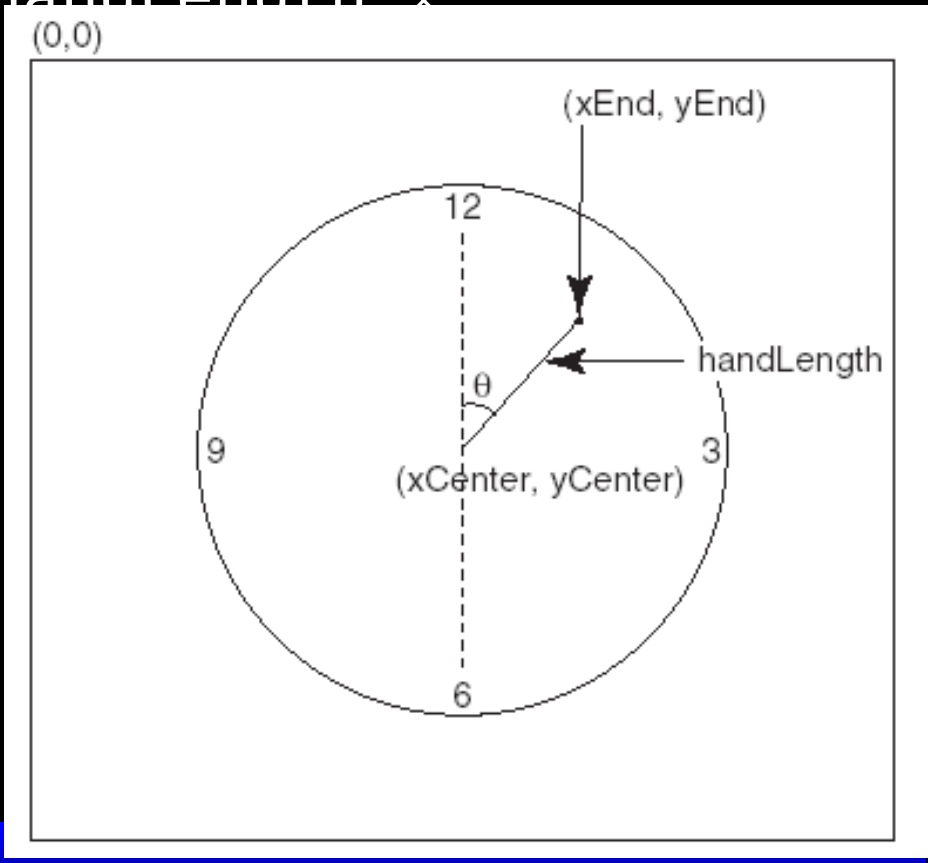
Ví dụ 9.6 (tiếp)

```
xEnd = xCenter + handLength ×
sin(θ)

yEnd = yCenter - handLength ×
cos(θ)
```

Vì hình tròn được chia thành 12 giờ, góc của kim giờ là:

```
(hour + minute/60 +
second/(60 × 60))) ×
(2π/12)
```



Lập trình hướng sự kiện

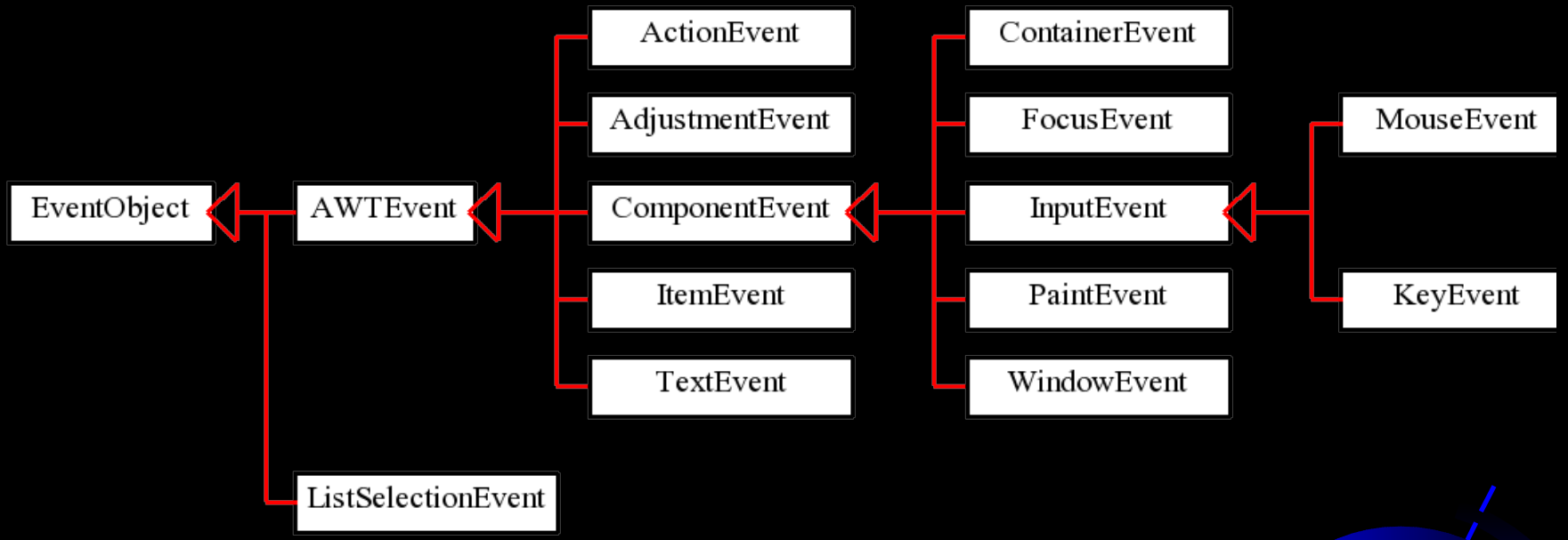
- *Lập trình hướng thủ tục (Procedural programming)* chương trình được thực hiện theo thứ tự thủ tục.

- Trong *lập trình hướng sự kiện (event-driven programming)*, mã lệnh được thực hiện vào lúc kích hoạt sự kiện.

Sự kiện

- Một *sự kiện* (*event*) có thể được định nghĩa là một loại tín hiệu báo cho chương trình có điều gì đó đã xảy ra..
- Sự kiện được sinh ra bởi các hành động của người sử dụng (ví dụ: di chuột, kích phím chuột, ấn phím) hoặc bởi HĐH (vd: timer).

Các lớp sự kiện



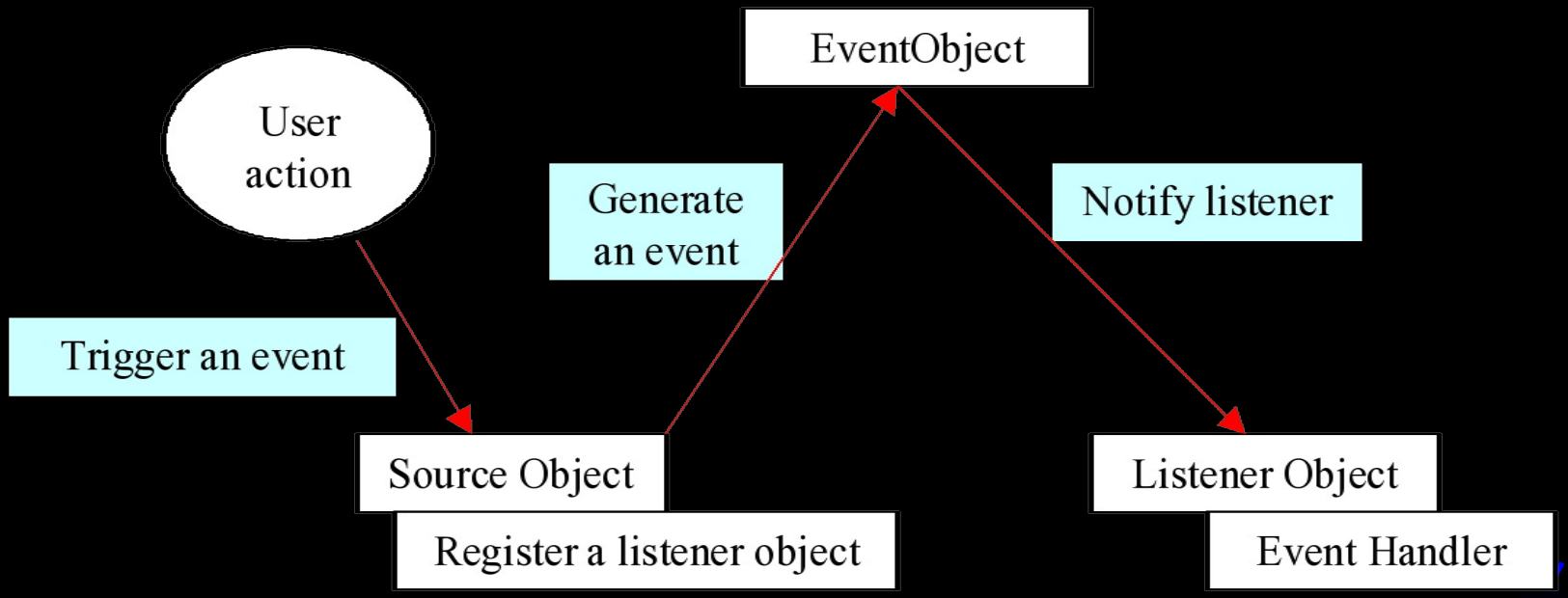
Các lớp sự kiện trên nằm trong gói `java.awt.event`

riêng ListSelectionEvent ở trong gói `javax.swing.event`

Selected User Actions

User Action	Source Object	Event Type Generated
Clicked on a button	`JButton`	`ActionEvent`
Changed text	`JTextComponent`	`TextEvent`
Double-clicked on a list item	`JList`	`ActionEvent`
Selected or deselected an item with a single click	`JList`	`ItemEvent`
Selected or deselected an item	`JComboBox`	`ItemEvent`

Mô hình ủy quyền



Selected Event Handlers

Event Class	Listener Interface	Listener Methods (Handlers)
`ActionEvent`	`ActionListener`	`actionPerformed(ActionEvent)`
`ItemEvent`	`ItemListener`	`itemStateChanged(ItemEvent)`
`WindowEvent`	`WindowListener`	`windowClosing(WindowEvent)`
		`windowOpened(WindowEvent)`
		`windowIconified(WindowEvent)`
		`windowDeiconified(WindowEvent)`
		`windowClosed(WindowEvent)`
		`windowActivated(WindowEvent)`
		`windowDeactivated(WindowEvent)`
`ContainerEvent`	`ContainerListener`	`componentAdded(ContainerEvent)`
		`componentRemoved(ContainerEvent)`

Ví dụ 9.7:
Xử lý sự kiện hành động đơn giản

- Mục tiêu: Hiển thị 2 nút bấm OK và Cancel trong cửa sổ. Khi kích chuột vào một nút, một message được hiển thị để chỉ ra nút nào đã được kích.

```
import java.util.*;import java.awt.*;import java.awt.event.*;import javax.swing.*;
class Gui {
   private static JFrame fr = null;   private static JButton btn = null;
   private static JButton btn1 = null;
   private static JTextField txt = null;   private static Label reg;
   public static void main(String args[]) {
     Label hi = new Label("Hi"); Label nhap = new Label("Nhap:");
     txt = new JTextField(); reg = new Label("");
     JPanel pane = new JPanel(new GridLayout(3,2));
     pane.add(hi);pane.add(reg);pane.add(nhap);pane.add(txt);
     btn=new JButton("OK");  btn.setActionCommand("btn");
     btn.setMnemonic(KeyEvent.VK_O);  //Su dung phim tat Alt+C
     btn.addActionListener(new ButtonListener()); // nut btn dang ky xu ly su kien
     btn1 = new JButton("Cancel"); // Tao nut  “Cancel”
     btn1.setActionCommand("btn1");
     btn1.setMnemonic(KeyEvent.VK_C);  //Su dung phim tat Alt+C
     btn1.addActionListener(new ButtonListener()); // nut btn1 dang ky xu ly su kien
     pane.add(btn);  pane.add(btn1);
     fr = new JFrame("Thu nghiem");  fr.setContentPane(pane);
     fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     fr.setSize(200,100);    fr.setLocation(200, 200); fr.setVisible(true);
   }
```

```
// Xu ly su kien khi nguoi dung bam chuot vao nut
  static class ButtonListener implements ActionListener {
     public void actionPerformed(ActionEvent e) {
       String whichButton = e.getActionCommand( ); // Get name

       if (whichButton.equals("btn")){
           reg.setText(txt.getText());
           txt.setText("");
       }
       else{
           fr.dispose();//Dong khung
           System.exit(0);//Ket thuc chuong trinh
       }
     }
  }
}
```

Ví dụ 9.8: Xử lý sự kiện cửa sổ

- Mục tiêu: Minh họa việc xử lý sự kiện cửa sổ. Bất kỳ lớp con nào của lớp Window có thể tạo ra các sự kiện cửa sổ sau: đã mở, đang đóng, đã đóng, đã được kích hoạt, đã mất kích hoạt, đã được thu nhỏ thành biểu tượng, phóng to trở lại cửa sổ. Chương trình này tạo một frame, lắng nghe các sự kiện cửa sổ, và hiển thị thông báo chỉ ra sự kiện đang xuất hiện.

```
import javax.swing.*;
class Gui1 {
public static void main(String args[ ]) {
    JFrame  myWindow  =  new JFrame();
    myWindow.setSize(400,300);
    JLabel myLabel = new JLabel("Hello my world!");
    myWindow.getContentPane().add(myLabel);
    WindowDestroyer myListener = new WindowDestroyer();
    myWindow.addWindowListener(myListener);
    myWindow.setVisible(true);
  }
}
import java.awt.*; import java.awt.event.*;
public class WindowDestroyer extends WindowAdapter {
  public void windowClosing(WindowEvent e) {
    System.exit(0);
  }
}
```

JButton

- *Button* là một thành phần gây ra một sự kiện hành động khi được kích chuột.
- Các constructor của JButton:

JButton()

JButton(String text)

JButton(String text, Icon icon)

JButton(Icon icon)

Các thuộc tính **JButton**

- `text`
- `icon`
- `mnemonic`
- `horizontalAlignment`
- `verticalAlignment`
- `horizontalTextPosition`
- `verticalTextPosition`

Đáp ứng các sự kiện **JButton**

```
JButton bt = new JButton(“OK”);
bt.setActionCommand(“Left”);
bt.addActionListener(new Nghenut());
static class Nghenut implements ActionListener{
public void actionPerformed(ActionEvent e)
{
  // Get the button label
  String actionCommand = e.getActionCommand();
  // Make sure the event source is Left button
  if (e.getSource() instanceof JButton)
    // Make sure it is the right button
    if ("Left".equals(actionCommand))
    System.out.println ("Button pressed!");
}
```

JCheckBox

- *Check box* là một thành phần cho phép người dùng bật hay tắt một lựa chọn, giống như 1 công tắc đèn.
- Các constructor:

```
JCheckBox()
JCheckBox(String text)
JCheckBox(String text, boolean selected)
JCheckBox(Icon icon)
JCheckBox(String text, Icon icon)
JCheckBox(String text,Icon icon,boolean selected)
```

Các thuộc tính **JCheckBox**

- JCheckBox có tất cả các thuộc tính trong JButton. Ngoài ta, JCheckBox có thuộc tính:

selected

- Using Check Box:

```
JCheckBox cb = new
JCheckBox("Chon");
if (cb.isSelected()) { . . . }
```

JRadioButton

- Các *Radio button* là sự biến đổi của các check box. Chúng thường được sử dụng trong một nhóm khi mà chỉ có 1 button được chọn tại một thời điểm.
- Các constructor:

```
JRadioButton()
JRadioButton(String text)
JRadioButton(String text, boolean selected)
JRadioButton(Icon icon)
JRadioButton(String text, Icon icon)
JRadioButton(String text, Icon icon, boolean selected)
```

Các thuộc tính **JRadioButton**

JRadioButton có tất cả các thuộc tính trong JButton. Ngoài ra, JRadioButton có thuộc tính:

selected

Gộp nhóm các Radio Button

```
ButtonGroup btg = new ButtonGroup();

btg.add(jrb1);

btg.add(jrb2);
```

Using Radio Buttons:

```
JRadioButton rb1 = new JRadioButton("Chon");

JRadioButton rb2 = new JRadioButton("Khong
  chon");

btg.add(rb1); btg.add(rb2);

if (rb1.isSelected()) { . . . }
```

JLabel

- *Label* dùng để hiển thị một chuỗi văn bản thông thường nhằm mô tả thêm thông tin cho các đối tượng khác.
- Các constructor của JLabel:

```
JLabel()

JLabel(String text)

JLabel(String text,int hAlignment)

JLabel(Icon icon)

JLabel(Icon icon, int hAlignment)

JLabel(String text,Icon icon,int hAlignment)
```

Các thuộc tính JLabel

- text
- icon
- horizontalAlignment
- verticalAlignment

Using Labels

JTextField

- ***Text field*** là ô nhập dữ liệu dạng văn bản trên 1 dòng.
- Các constructor của JTextField:

JTextField()

JTextField(int columns)

Tạo một text field trống có số cột xác định.

JTextField(String text)

Tạo một text field với văn bản có sẵn.

JTextField(String text, int columns)

Tạo một text field với văn bản có sẵn và số cột xác định.

Các thuộc tính **JTextField**

- text
- horizontalAlignment
- editable
- columns

Các phương thức **JTextField**

- getText()
  Trả về chuỗi ký tự trong text field.
- setText(String text)
  Đặt chuỗi ký tự trong text field.
- setEditable(boolean editable)
  Cho phép hoặc vô hiệu hóa soạn thảo trong text field. Mặc định, editable là true.
- setColumns(int)
  Thiết lập số cột trong text field. Chiều dài của text field có thể thay đổi.

JTextArea

- ***TextArea*** là khung cho phép người sử dụng nhập vào nhiều dòng văn bản.
- Các constructor của JTextArea:

JTextArea()

JTextArea(String s)

JTextArea(int rows, int columns)

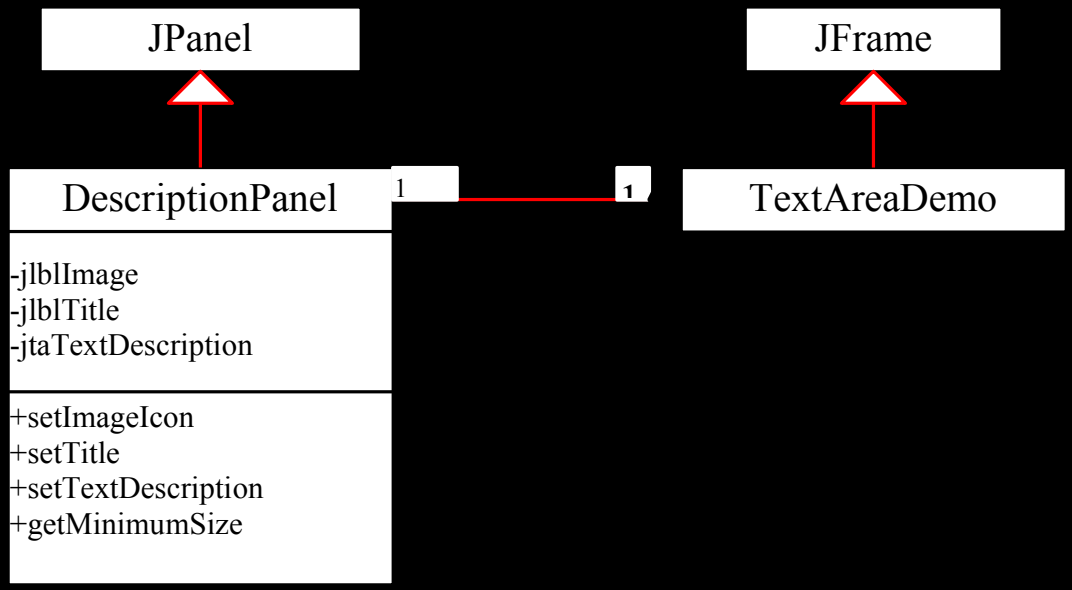
JTextArea(String s, int rows, int columns)

Các thuộc tính **JTextArea**

- text
- editable
- columns
- lineWrap
- wrapStyleWord
- rows
- lineCount
- tabSize

Sử dụng **Text Area**

Chương trình hiển thị 1 ảnh và 1 title trong 1 label, hiển thị văn bản trong text area.



JComboBox

- *Combo box* là danh sách đơn giản các mục chọn. Cơ bản nó thực hiện chức năng giống như 1 list, nhưng chỉ có thể lấy 1 giá trị.
- Các constructor:

JComboBox()

tạo 1 combo box rỗng

JComboBox(Object[] stringItems)

tạo 1 combo box chứa các phần tử trong dãy

Các phương thức **JComboBox**

```
jcbo.addItem(Object item)
  thêm 1 mục chọn vào JComboBox jcbo

jcbo.getItem()

jcbo.getItemAt(int index)
  lấy 1 mục chọn từ JComboBox jcbo

jcbo.removeItemAt(int index)
  loại 1 mục chọn khỏi JComboBox jcbo

Using Combo Box:
```

Sử dụng `itemStateChanged` Handler

Khi một lựa chọn được check hoặc uncheck, `itemStateChanged()` cho `ItemEvent` và `actionPerformed()` handler cho `ActionEvent` sẽ được gọi.

```
public void itemStateChanged(ItemEvent e){
  // Make sure the source is a combo box
  if (e.getSource() instanceof JComboBox)
    String s = (String)e.getItem();
}
```

JList

- *List* là một thành phần cơ bản thực hiện chức năng giống combo box, nhưng nó cho phép người sử dụng chọn một hoặc nhiều giá trị.
- Các constructor:

JList()

    tạo 1 list rỗng.

JList(Object[] stringItems)

    tạo 1 list chứa các phần tử trong dãy.

Các thuộc tính **JList**

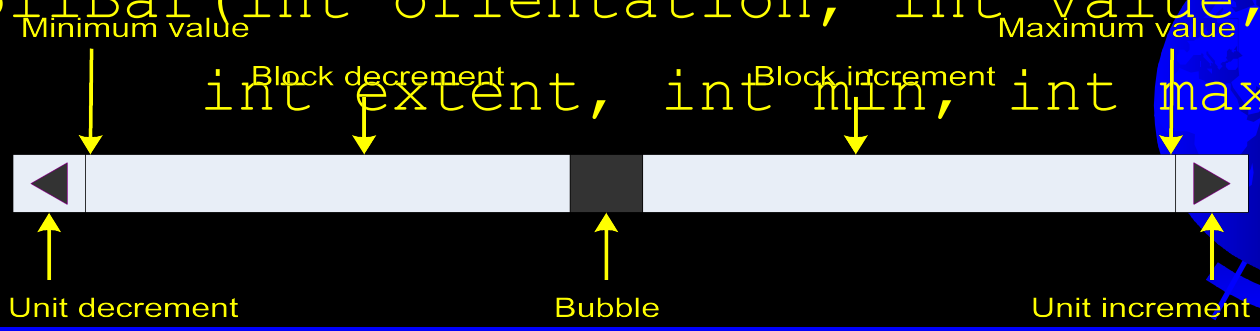
- selectedIndex
- selectedIndices
- selectedValue
- selectedValues
- selectionMode
- visibleRowCount

Using Lists:

JScrollBar

- *ScrollBar* là một điều khiển cho phép người sử dụng chọn từ một dải các giá trị.
- Các constructor:

```
JScrollBar()
JScrollBar(int orientation)
JScrollBar(int orientation, int value,
           int extent, int min, int max)
```



Các thuộc tính **JScrollBar**

- orientation: 1 - dọc, 0 - ngang
- maximum, minimum
- visibleAmount (extent): độ rộng của phần con chạy
- value: giá trị hiện thời của scroll bar
- blockIncrement: giá trị được cộng thêm khi kích hoạt vùng tăng.
- unitIncrement: giá trị được cộng thêm khi kích hoạt đầu tăng.

Các phương thức **JScrollBar**

- `setBlockIncrement(int increment)`
- `setMaximum(int maximum)`
- `setMinimum`
- `setOrientation(int orientation)`
- `setUnitIncrement(int increment)`
- `setValue(int value)`
- `setVisibleAmount (extent):`
- `getBlockIncrement()`
- `...`

Borders

- Bạn có thể thiết lập một border trên bất kỳ đối tượng nào của lớp JComponent, nhưng thường hữu ích khi thiết lập một titled border trên JPanel để nhóm một tập các thành phần giao diện người sử dụng có liên quan.
- Using border:

Các phương thức tĩnh để tạo Borders

- `createTitledBorder(String title)`
- `createLoweredBevelBorder()`
- `createRaisedBevelBorder()`
- `createLineBorder(Color color)`
- `createLineBorder(Color color, int thickness)`
- `createEtchedBorder()`
- `createEtchedBorder(Color highlight, Color shadow, boolean selected)`
- `createEmptyBorder()`
- `createMatteBorder(int top, int left, int bottom, int right, Icon tileIcon)`
- `createCompoundBorder(Border outsideBorder, Border insideBorder)`

Dialogs

- Có thể sử dụng lớp JOptionPane để tạo 4 loại dialog chuẩn:
  - ***Message Dialog*** hiển thị một message và đợi người sử dụng kích nút OK để đóng hộp thoại.
  - ***Confirmation Dialog*** hiển thị câu hỏi và đề nghị người sử dụng trả lời, vd: OK hay Cancel
  - ***Input Dialog*** hiển thị câu hỏi và nhận dữ liệu vào từ 1 text field, combo box hoặc list.
  - ***Option Dialog*** hiển thị câu hỏi và nhận câu trả lời từ một tập các lựa chọn.

Tạo các Message Dialog

Sử dụng phương thức tĩnh trong lớp JOptionPane

```
showMessageDialog(Component parentComponent,
     Object message)

showMessageDialog(Component parentComponent, Object
message,String title, int messageType)

showMessageDialog(Component parentComponent,
     Object message, String title,
     int messageType, Icon icon)
```

Tạo các Confirmation Dialog

Sử dụng phương thức tĩnh trong lớp JOptionPane

```
showConfirmDialog(Component parentComponent,
     Object message)

showConfirmDialog(Component parentComponent, Object
message, String title, int optionType)

showConfirmDialog(Component parentComponent, Object
message, String title, int optionType,
     [[int messageType], Icon icon])
```

Tạo các Option Dialog

Sử dụng phương thức tĩnh trong lớp JOptionPane

```
showOptionDialog(Component parentComponent, Object
message, String title, int optionType, int
messageType, Icon icon, Object[] options, Object
initialValue)
```

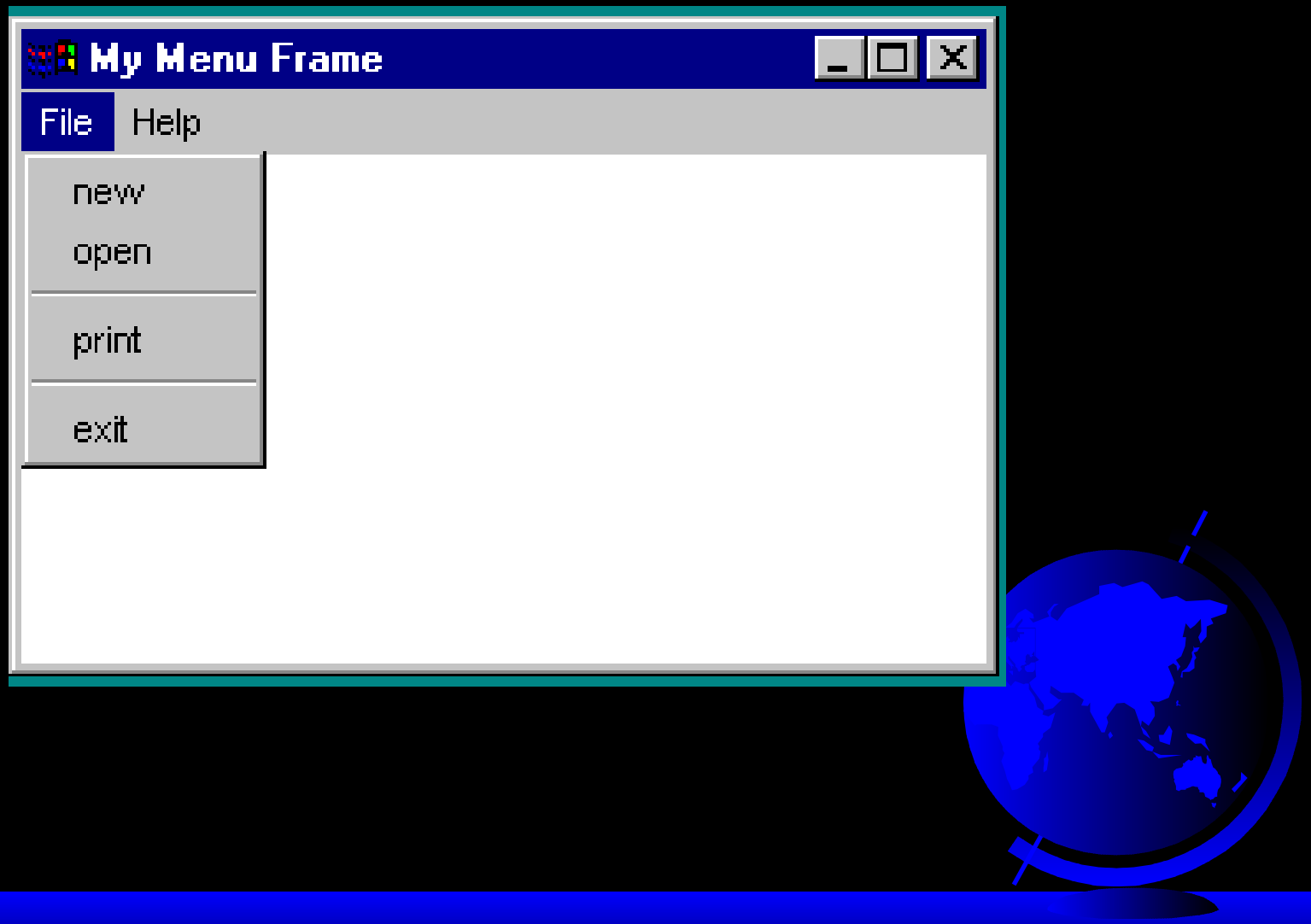
Sử dụng các dialog:

Menus

- Java cung cấp một số lớp - JMenuBar, JMenu, JMenuItem, JCheckBoxMenuItem, và JRadioButtonMenuItem - để thực thi menu trong một frame.

- Một JFrame hoặc JApplet có thể chứa một *menu bar* trên đó có gắn các *pull-down menu*. Các menu chứa các *menu item* để người dùng lựa chọn (hoặc bật/tắt). Menu bar có thể được xem như một cấu trúc để hỗ trợ các menu.

Menu Demo



Lớp **JMenuBar**

Menu bar chứa các menu; menu bar chỉ có thể được thêm vào 1 frame. Đoạn code sau tạo và thêm một JMenuBar vào 1 frame:

```
JFrame f = new JFrame();
f.setSize(300, 200);
f.setVisible(true);
JMenuBar mb = new JMenuBar();
f.setJMenuBar(mb);
```

Lớp **Menu**

Bạn gắn các menu vào một `JMenuBar`. Đoạn code sau tạo 2 menu `File` và `Help`, và thêm chúng vào `JMenuBar mb`:

```
JMenu fileMenu = new JMenu("File", false);
JMenu helpMenu = new JMenu("Help", true);
mb.add(fileMenu);
mb.add(helpMenu);
```

Lớp **JMenuItem**

Đoạn code sau thêm các mục chọn (menu item) và các separator trong menu `fileMenu`:

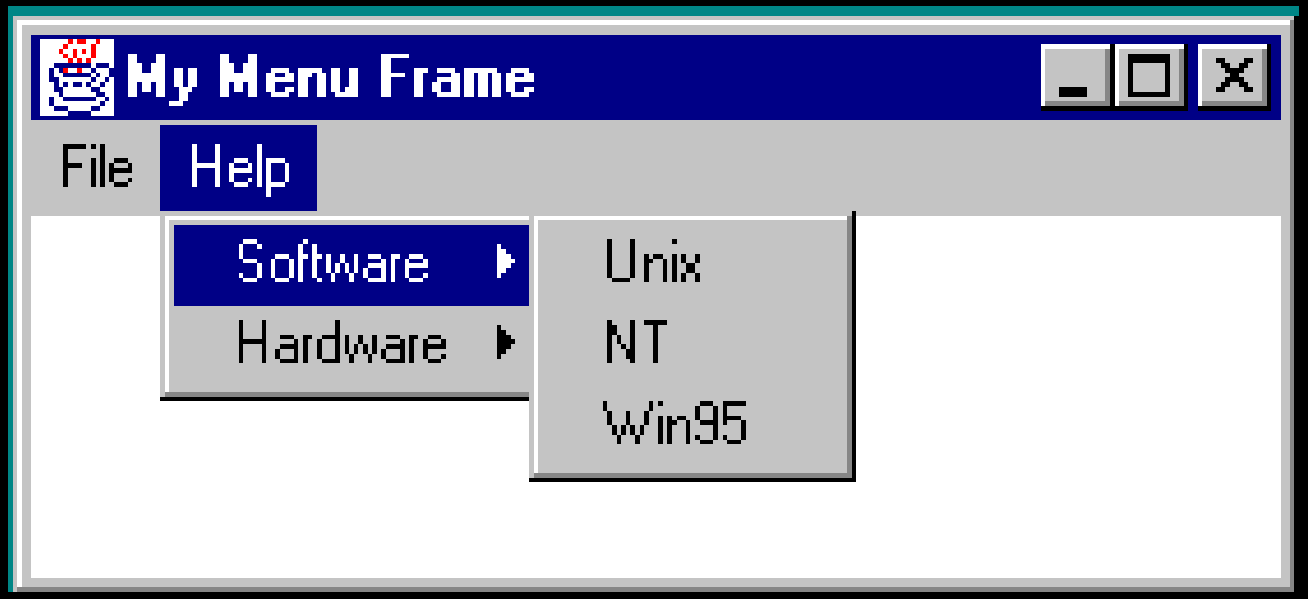
```
fileMenu.add(new JMenuItem("New"));
fileMenu.add(new JMenuItem("Open"));
fileMenu.addSeparator();
fileMenu.add(new JMenuItem("Print"));
fileMenu.addSeparator();
fileMenu.add(new JMenuItem("Exit"));
```

Submenus

Bạn có thể thêm các submenus vào các menu item. Đoạn code sau thêm các submenu “Unix”, “NT”, và “Win95” vào trong mục chọn “Software”.

```
JMenu softwareHelpSubMenu = new JMenu("Software");
JMenu hardwareHelpSubMenu = new JMenu("Hardware");
helpMenu.add(softwareHelpSubMenu);
helpMenu.add(hardwareHelpSubMenu);
softwareHelpSubMenu.add(new JMenuItem("Unix"));
softwareHelpSubMenu.add(new JMenuItem("NT"));
softwareHelpSubMenu.add(new JMenuItem("Win95"));
```

Submenu Demo



Sử dụng Menu

Tạo một giao diện thực hiện các phép toán số học giữa 2 số Number1 và Number2. Giao diện chứa các nhãn và text field cho Number 1, Number 2, và Result (sử dụng như đối với JButton).

Tạo thêm Window - bước 1

Bước 1: Tạo 1 subclass của lớp `JFrame` (được gọi là 1 `SubFrame`) để xác định cửa sổ mới làm việc gì. Ví dụ, tất cả các chương trình ứng dụng GUI mở rộng `JFrame` và là các subclass của `JFrame`.

Tạo thêm Window - bước 2

Bước 2: Tạo 1 instance của SubFrame trong ứng dụng hoặc trong applet.

Ví dụ:

```
SubFrame subFrame = new
   SubFrame("SubFrame Title");
```

Tạo thêm Window - bước 3

Bước 3: Tạo 1 JButton để kích hoạt subFrame.

```
add(new JButton("Activate SubFrame"));
```

Tạo thêm Window - bước 4

Bước 4: chồng phương thức actionPerformed() như sau:

```
public actionPerformed(ActionEvent e)
{
  String actionCommand = e.getActionCommand();
  if (e.target instanceof Button)
  {
    if ("Activate
     SubFrame".equals(actionCommand))
    {
      subFrame.setVisible(true);
    }
  }
}
```

Ví dụ: Tạo nhiều Window

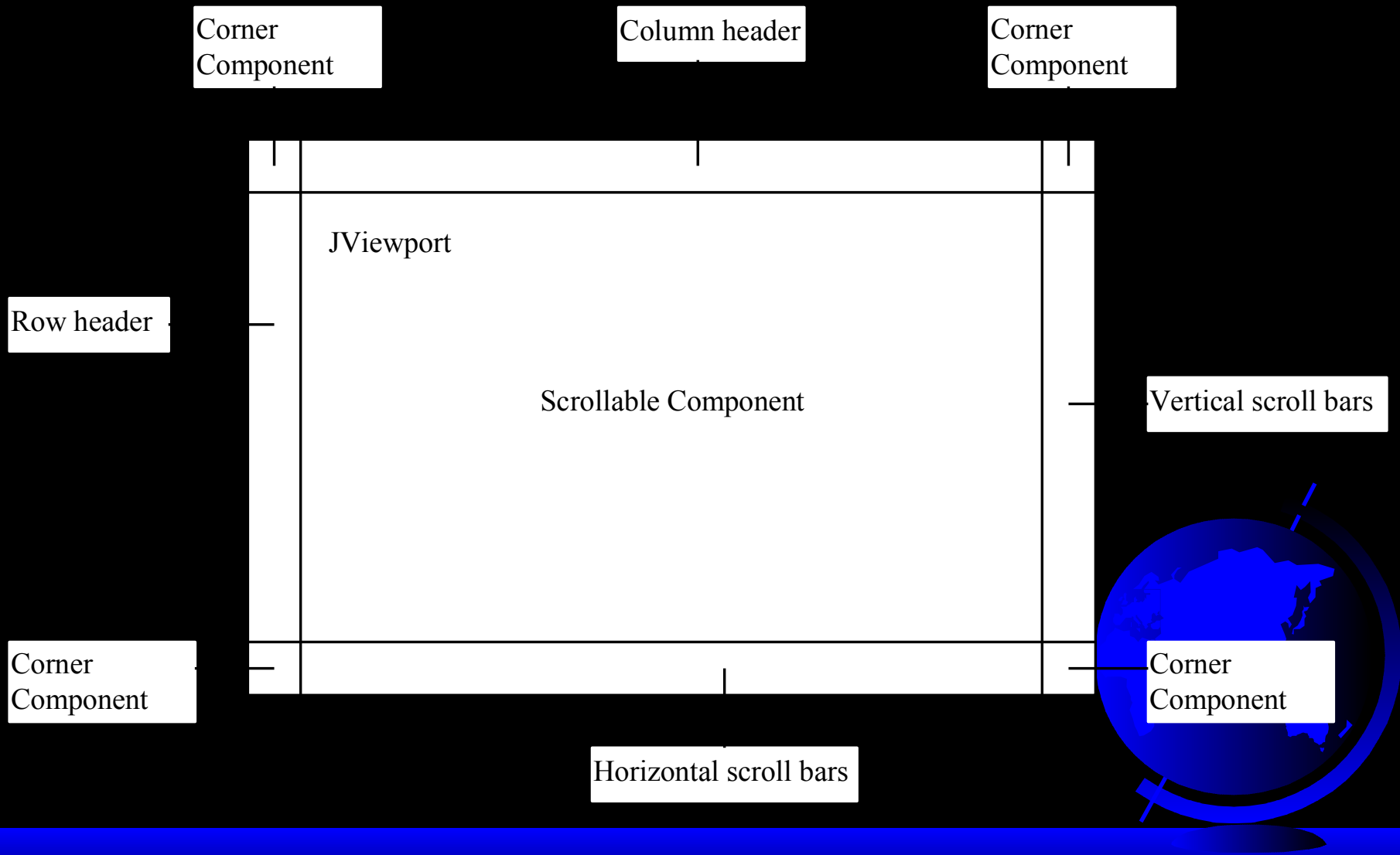
Ví dụ tạo 1 main window có 1 text area trong scroll pane, 1 button "Show Histogram". Khi người sử dụng kích vào button, 1 cửa sổ mới xuất hiện để hiển thị biểu đồ cho biểu diễn tần số xuất hiện của các ký tự trong text area.

JScrollPane

Scroll pane là một thành phần tự động hỗ trợ cuộn cửa sổ mà không cần lập trình.

Using Scroll Pane: add một TextArea vào trong một Scroll pane

Cấu trúc Scroll Pane



JTabbedPane

Tabbed pane cung cấp một tập các tab loại trừ lẫn nhau để truy nhập nhiều thành phần.

Using Tabbed Pane: