

LẬP TRÌNH JAVA

Đỗ Ngọc Như Loan - Nguyễn Thị Hồng Anh

Chương 3. Lập trình giao diện đồ họa

Nội dung



- Try ... catch
- Swing và AWT
- Frame, Panel
- Label, Textfield, Button
- JCheckBox
- JRadiobutton
- ArrayList
- Jcombobox
- JTable
- JMenu/ PopupMenu/ JTabPane
- BorderLayout/ GridLayout

Chương 3. Lập trình giao diện đồ họa

Try ... catch



- Một số dạng lỗi trong lập trình:
 - Lỗi cấu hình hệ thống
 - Lỗi biên dịch (compiler error)
 - Lỗi cú pháp
 - Lỗi truy cập trái phép (ví dụ: truy cập vào các thuộc tính private)
 - Lỗi thực thi:
 - Lỗi bất thường (Error)
 - Ngoại lệ
 - Sai kết quả đầu ra

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

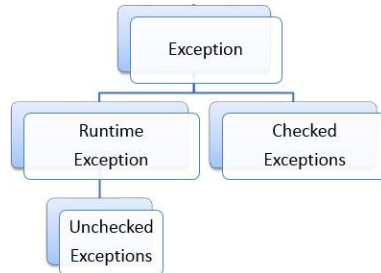
Try... catch



- Ngoại lệ: là các trường hợp gây lỗi chương trình như xử lý chậm, treo, dừng chương trình,... tuy nhiên có thể tiên đoán và phòng tránh được. Ta có thể sử dụng cấu trúc try... catch để bắt ngoại lệ giúp chương trình có thể thực thi theo ý muốn.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



Một số ngoại lệ 'checked':



- ClassNotFoundException
- IOException
- FileNotFoundException
- EOFException

Một số ngoại lệ 'unchecked':



- ArithmeticException
- IndexOutOfBoundsException
- NullPointerException

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



Try... catch – Một số dạng ngoại lệ

- **NullPointerException**: gặp phải khi tham số hoặc thuộc tính có KDL là kiểu có cấu trúc tuy nhiên tại thời điểm sử dụng chưa được khởi tạo.
- **NumberFormatException**: gặp khi thực hiện chuyển đổi định dạng một chuỗi sang dạng số, tuy nhiên chuỗi gồm các ký tự không phải dạng số.
- **IndexOutOfBoundsException**: gặp phải khi thực hiện truy cập tới các dữ liệu vượt quá chỉ số giới hạn của đối tượng.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Try... catch – Một số dạng ngoại lệ



- `ArrayIndexOutOfBoundsException`: gặp phải khi thực hiện truy cập tới các phần tử có chỉ số vượt quá chỉ số của mảng.
- `ArithmeticException`: gặp phải khi thực hiện các biểu thức số học vô nghĩa. (Ví dụ: chia cho 0)
- `FileNotFoundException`: khi mở 1 file không tồn tại.
- `IOException`: gặp phải khi đọc/ ghi dữ liệu vào file bị gián đoạn.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Try... catch – Cách khắc phục ngoại lệ



- Có thể sử dụng khối lệnh `try ... catch` để xử lý các ngoại lệ
- Cú pháp:

```
try{  
    // khối lệnh không an toàn  
} catch(Exception e) {  
    // Khối lệnh xử lý khi gặp ngoại lệ  
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



Không dùng Try catch

```
c = a/b;  
System.out.println("Sau phép chia !");(*)
```

Câu lệnh (*) sẽ không được thực hiện nếu mẫu số b=0, chương trình lập tức ngừng lại và xuất hiện thông báo lỗi của hệ thống

Sử dụng Try catch

```
try { c = a/b; } catch(Exception e) {  
System.out.println("Có lỗi "+e); }  
System.out.println("Sau phép chia !");  
(*)
```

Câu lệnh (*) sẽ luôn được thực hiện dù mẫu số b bằng 0 hay b khác 0.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



Try... catch – Cách khắc phục ngoại lệ

■ Ví dụ:

```
public boolean tinhChia(){  
    try{  
        thuong=a*1F/b;  
        return true;  
    } catch(ArithmeticException e) {  
        e.printStackTrace(); //in lỗi ra màn hình  
        return false;  
    }  
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Try...catch – xử lý đồng thời nhiều ngoại lệ



- Nếu trong 1 đoạn code có nhiều ngoại lệ, nhưng các xử lý cho từng ngoại lệ là giống nhau, chúng ta có thể áp dụng 1 trong 2 cách:
 - Liệt kê các ngoại lệ trong khối catch và xây dựng chung 1 đoạn code xử lý
 - Biểu diễn các ngoại lệ đó bằng 1 ngoại lệ chung có KDL là cha của các ngoại lệ đó và xây dựng 1 đoạn code để xử lý.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Try...catch – xử lý đồng thời nhiều ngoại lệ



- Ví dụ:

```
int a=5;
int b=0;
String text="hi";
try {
    int thuong=a/b;
    char kyTu=text.charAt(3);
} catch(ArithmeticException| IndexOutOfBoundsException e){
    e.printStackTrace();
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Try...catch – xử lý đồng thời nhiều ngoại lệ



- Ví dụ:

```
int a=5;
int b=0;
String text="hi";
try {
    int thuong=a/b;
    char kyTu=text.charAt(3);
} catch (Exception e){
    e.printStackTrace();
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Try ... catch



- Nếu trong 1 đoạn code có nhiều ngoại lệ, nhưng các xử lý cho từng ngoại lệ là khác nhau, chúng ta phải xây dựng từng khối catch riêng để xử lý cho từng ngoại lệ.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



■ Ví dụ:

```
int a=5;
int b=0;
String text="hi";
try {
    int thuong=a/b;
    char kyTu=text.charAt(3);
} catch( ArithmeticException e) {
    e.printStackTrace();
    System.err.println("khong thuc hien phep chia cho 0");
} catch( IndexOutOfBoundsException e) {
    e.printStackTrace();
    System.err.println("chi so bi tran");
}
```

Chương 3. Lập trình giao diện đồ họa

Lưu ý sử dụng khối try... catch để xử lý ngoại lệ



- Trong khối finally sẽ chứa một khối mã sẽ thực hiện sau khối try/catch. Khối finally sẽ được thực hiện dù ngoại lệ có xuất hiện hay không. Tuy nhiên, mỗi try sẽ yêu cầu có ít nhất 1 catch hoặc 1 finally

try ⇒ catch ⇒ finally

try ⇒ catch

try ⇒ finally

Chương 3. Lập trình giao diện đồ họa

Try ... catch – throw ngoại lệ



```
public class TestThrow {  
    static void valid(int age){  
        if(age <18)  
            throw new ArithmeticException("not enough age");  
        else  
            System.out.println("You are Welcome");  
    }  
    public static void main(String[] args) {  
        valid(13);  
    }  
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Xử lý ngoại lệ trong java – Sử dụng từ khóa throws



- Từ khóa throws được sử dụng trong phương thức dùng để đề xuất các ngoại lệ có thể xảy ra trong phương thức đó. Có những phương thức sử dụng một số lệnh mà các lệnh đó có thể xảy ra ngoại lệ 'checked' nên chúng ta bắt buộc phải xử lý ngoại lệ đó.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Ví dụ



```
public class SampleException4 {  
    public void writeFile() throws IOException {  
        FileWriter fw = new FileWriter("data.txt");  
        fw.write("Xu ly ngoai le trong java");  
        fw.close();  
    }  
    public static void main(String args[]) {  
        try {  
            SampleException5 se5 = new SampleException5();  
            se5.writeFile();  
        } catch (IOException ioe) {  
            System.out.println("Co loi ghi file: "+ ioe);  
        } } }
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Bài tập



1. Nhập 3 số nguyên a,b,c. Xuất kết quả $c/(a-b)$.
2. Cho mảng một chiều a chứa n số nguyên. Viết hàm
 - a) Xóa phần tử tại vị trí k
 - b) Thêm phần tử x tại vị trí k.
3. Viết chương trình thực hiện nhập thông tin nhân viên: họ tên, ngày sinh, giới tính. Hãy tính và xuất tháng/năm nghỉ hưu của nhân viên theo quy định:
 - * Nữ: 60 tuổi
 - * Nam: 62 tuổi

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Bài tập



4. Tạo package tên exception và tạo class tên SinhVien có các thuộc tính như mã sinh viên, họ tên, điểm, xếp loại. Tiêu chí xếp loại dựa vào điểm
- Nếu điểm ≥ 8 thì xếp loại giỏi
 - Điểm ≥ 7 thì xếp loại khá
 - Điểm ≥ 5 thì xếp loại trung bình
 - Điểm < 5 thì xếp loại kém
- Một số yêu cầu khác
- Viết các setter, getter, constructor, toString.
 - Nhập dữ liệu cho sinh viên từ bàn phím. Viết code xử lý nhập dữ liệu hợp lệ.
 - Nếu nhập sai kiểu dữ liệu thì thông báo lỗi và yêu cầu nhập lại. Ví dụ nhập điểm là 10a thì chương trình sẽ hiển thị thông báo “Bạn phải nhập dữ liệu là kiểu số”

Chương 3. Lập trình giao diện đồ họa

Khái niệm



- Lập trình giao diện GUI (Graphic User Interface) là việc sử dụng các đối tượng trong Java để thiết kế thành các giao diện trực quan giúp người dùng có thể tương tác để thực hiện các công việc.
- Trong một bài toán, có thể có 1 hoặc nhiều giao diện người dùng.

Chương 3. Lập trình giao diện đồ họa

Các thư viện lập trình giao diện



- AWT
- SWING

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Thư viện lập trình AWT (Abstract Window Toolkit)



- AWT là bộ thư viện được Java xây dựng từ phiên bản JDK 1.0 để hỗ trợ thiết kế các giao diện và đồ họa người dùng.
- Đây được xem là bộ thư viện cồng kềnh, gặp khó khăn và giao diện không ổn định trên các hệ điều hành khác nhau

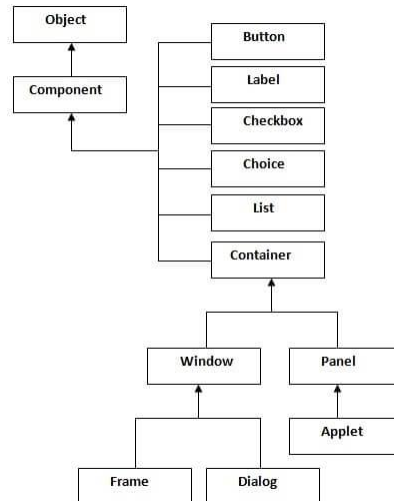
Lập trình Java

Chương 3. Lập trình giao diện đồ họa

AWT



- Kiến trúc một số đối tượng AWT



Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Thư viện lập trình Swing

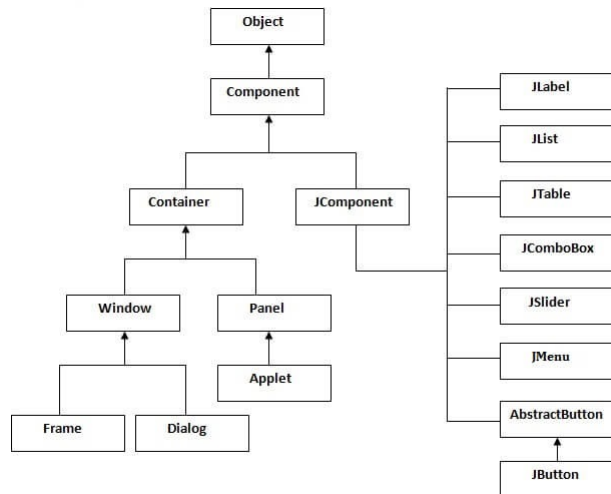


- Swing được xây dựng và tích hợp vào phiên bản JDK 1.1.2
- Được đánh giá là bộ thư viện tối ưu và thuận tiện cho việc thiết kế các giao diện người dùng, thân thiện và phù hợp cho nhiều hệ điều hành khác nhau.
- Java Swing là một phần của Java Foundation Classes (JFC) được sử dụng để tạo các ứng dụng Window-Based. Nó được xây dựng ở trên cùng của AWT API và được viết hoàn toàn bằng Java.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Kiến trúc một số đối tượng trong Swing



Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Điểm khác nhau giữa AWT và SWING



Java AWT	Java SWING
Các thành phần AWT là phụ thuộc nền tảng	Các thành phần Java Swing là độc lập nền tảng
Các thành phần AWT là nặng	Các thành phần Swing là gọn nhẹ
AWT không hỗ trợ pluggable L&F	Swing hỗ trợ pluggable L&F
AWT cung cấp ít thành phần hơn Swing	Swing cung cấp các thành phần mạnh mẽ hơn như table, list, scrollpanes, colorchooser, tabbedpane ...
AWT không theo sau MVC (Model View Controller), ở đây model biểu diễn dữ liệu, view biểu diễn sự trình bày và controller hoạt động như một Interface giữa model và view	Swing theo sau MVC

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Các thành phần GUI



- Các phần tử UI: : Đó là các phần tử nhìn thấy chủ yếu mà người dùng cuối cùng nhìn thấy và tương tác với. Swing cung cấp rất nhiều các phần tử đa dạng từ cơ bản tới nâng cao.
- Layout: Chúng định nghĩa cách các phần tử UI nên được tổ chức trên màn hình và cung cấp đối tượng L&F (là viết tắt của Look and Feel) cuối cùng tới GUI (Graphical User Interface).
- Hành vi: Đó là các sự kiện xảy ra khi người dùng tương tác với các phần tử UI.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Các phần tử Swing UI



- **Lớp JLabel** Một đối tượng JLabel là một thành phần để đặt text vào trong một Container
- **Lớp JButton** Lớp này tạo một button đã được gán nhãn
- **Lớp JTable** Lớp JTable được sử dụng để hiển thị dữ liệu trên các ô của bảng hai chiều
- **Lớp Graphics** Lớp này cung cấp nhiều phương thức để lập trình đồ họa
- **Lớp JColorChooser** Một JColorChooser cung cấp một pane gồm các control được thiết kế để cho phép một người dùng thao tác và lựa chọn màu

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Các phần tử Swing UI



- **Lớp JCheckBox** Một JCheckBox là một thành phần đồ họa mà có thể trong trạng thái on (true) hoặc off (false)
- **Lớp JRadioButton** Lớp JRadioButton là một thành phần đồ họa mà có thể trong trạng thái on (true) hoặc off (false) trong một nhóm
- **Lớp JList** Một thành phần JList biểu diễn cho người dùng một danh sách các item
- **Lớp JComboBox** Một thành phần JComboBox biểu diễn cho người dùng một menu các lựa chọn

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Các phần tử Swing UI



- **JTextField** Một đối tượng JTextField là một thành phần text cho phép chỉnh sửa một dòng text đơn
- **Lớp JTextArea** Một đối tượng JTextArea là một thành phần text cho phép sửa đổi một text có nhiều dòng
- **Lớp ImageIcon** Một ImageIcon control là một trình triển khai của Icon Interface mà tô màu các Icon từ Image
- **Lớp JScrollbar** Một Scrollbar control biểu diễn một thành phần scroll bar để cho người dùng khả năng lựa chọn từ trong một dãy các giá trị

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Các phần tử Swing UI



- **Lớp JOptionPane** JOptionPane cung cấp tập hợp các dialog box chuẩn mà gợi ý người dùng về một giá trị hoặc thông báo cho họ một cái gì đó nbnmbnbn
- **JFileChooser** Một JFileChooser control biểu diễn một dialog window từ đó người dùng có thể lựa chọn một file
- **Lớp JProgressBar** Thanh tiến trình hiển thị phần trăm hoàn thành tác vụ đang diễn ra
- **Lớp JSlider** Một JSlider cho phép người dùng lựa chọn một giá trị từ một dãy cụ thể
- **Lớp JSpinner** Một JSpinner là một trường input dòng đơn, cho phép người dùng lựa chọn một số hoặc một giá trị đối tượng từ dãy đã qua sắp xếp

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Các đối tượng Container trong Java



- **Container** là vùng để đặt các thành phần giao diện vào đó. Bất cứ vật gì mà kế thừa từ lớp Container sẽ là vật chứa.
- Một vật chứa có thể chứa nhiều phần tử, các phần tử này có thể được vẽ hay được tô màu tùy thích. Bạn hãy xem vật chứa như một cửa sổ.
- Một số container thường gặp:
 - Frame, JFrame
 - Panels
 - Dialogs
 - ...

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Một số phương thức thường dùng của Java Swing trong Component class



- `public void add(Component c)` : thêm 1 component trên 1 component khác
- `public void setSize(int width, int height)`: thiết lập kích thước Component
 - Ví dụ: `setSize(300,100);`
- `public void setLayout(layoutManager m)`: thiết lập layout chính cho Component
- `public void setVisible(Boolean b)`: thiết lập thuộc tính ẩn hay hiện cho Component (giá trị ngầm định là false).

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JFrame



- JFrame trong gói Swing kế thừa từ Frame của AWT. JFrame giống như cửa sổ chính có thể chứa trong nó các thành phần để tạo Gui như: labels, buttons, textfields, ...
- Ưu điểm của JFrame hơn Frame: nó có thêm tùy chọn hide và close cửa sổ thông qua phương thức *`setDefaultCloseOperation(int)`*.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Jframe - Một số constructor thường dùng



- **JFrame():** tạo 1 frame trống
- **JFrame(String title):** tạo 1 frame có tiêu đề title

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Jframe – Một số phương thức thường dùng



- **setTitle(String Title):** định nghĩa tiêu đề cho khung giao diện
 - Ví dụ: `setTitle("My first program");`
- **setSize(int width, int height):** định nghĩa kích thước chiều rộng và chiều cao của khung giao diện.
 - Ví dụ: `setSize(300,100);`
- **setBackground(color c);** định nghĩa màu sắc cho nền của JFrame
 - Ví dụ: `getContentPane().setBackground(Color.RED);`

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Jframe – Một số phương thức thường dùng



- **setLocation(int x, int y):** định nghĩa vị trí hiển thị của khung giao diện trong màn hình.
 - Ví dụ: setLocation(20,20);
- **setBounds(int x, int y, int width, int height):** định nghĩa vị trí và kích thước cho khung giao diện.
 - Ví dụ: setBounds(20,20,300,100);
- **setResizable(Boolean value);** value=true cho phép người dùng có thể kéo dãn kích thước khung giao diện, ngược lại thì kích thước khung giao diện bị cố định.
 - Ví dụ: setResizable(true);

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Jframe – Một số phương thức thường dùng



- **setDefaultCloseOperation(int mode);** định nghĩa hành động của khung giao diện khi người dùng click vào dấu X ở góc phải trên cùng.
 - **DISPOSE_ON_CLOSE:** đóng giao diện đang thao tác mà không ảnh hưởng tới các giao diện hiển thị khác. Nếu chương trình chỉ có 1 giao diện thì chương trình sẽ tự động kết thúc khi giao diện được đóng.
 - **HIDE_ON_CLOSE:** ẩn giao diện đang sử dụng xuống, giống như khi ta click vào – thu nhỏ màn hình. Nó không tắt giao diện và các trạng thái trên giao diện vẫn giữ nguyên.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



- **DO_NOTHING_ON_CLOSE:** không thực hiện hành động gì. Trong trường hợp này, thông thường chúng ta sẽ xử lý bắt thao tác của người dùng và tự định nghĩa hành vi cho khung giao diện.
- **EXIT_ON_CLOSE:** đóng giao diện đang thao tác, đồng thời tắt luôn chương trình. Thường được sử dụng cho giao diện chính của chương trình.
- Ví dụ:

```
setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



- **setLayout(LayoutManager layout);** định nghĩa việc hiển thị không gian hiển thị các đối tượng trong khung giao diện.
- **null:** cho phép các đối tượng trong khung giao diện được hiển thị ở các vị trí tùy ý, phụ thuộc vào việc định nghĩa x, y, width, height của các đối tượng.
- **CardLayout:** cho phép các đối tượng giao diện được hiển thị thành các tầng layer, mỗi đối tượng là một tầng. Đối tượng được thêm vào đầu tiên sẽ ở trên cùng.
- Ví dụ:

```
setLayout(null);  
setLayout(new CardLayout());
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



- **setIconImage(Image icon);** định nghĩa icon của khung giao diện.

- Ví dụ:

- Khởi tạo đối tượng Image:

```
Image icon=new ImageIcon
```

```
(Ten_doi_tuong.class.getResource(String path).getImage());
```

- Set Icon cho khung giao diện:

```
setIconImage(Image icon);
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



Jframe – Một số phương thức thường dùng

- **setVisible(Boolean true);** định nghĩa việc ẩn/hiện của khung giao diện. Nếu true: khung giao diện hiển thị ngược lại thì bị ẩn.
 - Ví dụ: setVisible(true); // hiển thị khung giao diện
- **setLocationRelativeTo(Component comp);** định nghĩa việc hiển thị của một đối tượng khung giao diện trong màn hình chứa nó, giúp khung giao diện luôn hiển thị chính giữa.
 - Ví dụ: setLocationRelativeTo(null); // hiển thị cửa sổ chính giữa màn hình
- **pack();** hiển thị Frame ôm sát nội dung trên form

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Jframe – Một số phương thức thường dùng



- **addWindowListener(WindowListener lis);** là bộ sự kiện giúp khung giao diện lắng nghe được khi trạng thái của nó bị thay đổi như: bị đóng lại, bị mở ra, bị ẩn, ...
- **addMouseListener(MouseListener m);** là bộ sự kiện giúp khung giao diện lắng nghe khi người dùng sử dụng chuột để tương tác: click, press, move, di chuột vào khung giao diện, ...
- **addKeyListener(KeyListener k);** là bộ sự kiện giúp khung giao diện người dùng lắng nghe được khi người dùng nhấn các phím cứng trên bàn phím để tương tác với nó.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Jframe – Một số phương thức thường dùng



- **WindowEvent:** là đối tượng chứa tất cả các thông số cần thiết của khung giao diện tương ứng với trạng thái lắng nghe được. Có thể sử dụng đối tượng này để lấy các dữ liệu liên quan của khung giao diện tại thời điểm lắng nghe được.
- **MouseEvent:** là đối tượng chứa các thông số cần thiết về chuột như (x, y, ...) tương ứng với trạng thái khung giao diện đang lắng nghe được.
- **KeyEvent:** đối tượng chứa các thông tin về phím mà người dùng vừa thao tác trong các trạng thái lắng nghe được, chúng ta có thể thông qua đối tượng này để lấy các thông tin như mã, nội dung phím vừa nhấn là gì.
- **Add(Component com);** phương thức dùng để add một đối tượng giao diện vào khung chứa.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



- Code tạo JFrame có thể được viết trong các constructor
- Chúng ta cũng có thể kế thừa lớp JFrame, nên khi đó không cần tạo các thể hiện của các JFrame

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Cách xây dựng đối tượng JFrame



Ví dụ 3.1 File đặt trong tệp: FirstSwingExample.java

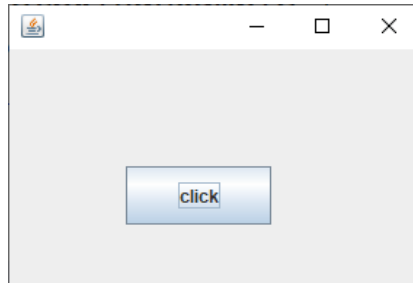
```
import javax.swing.*;
public class FirstSwingExample {
public static void main(String[] args) {
    JFrame f=new JFrame();//creating instance of JFrame
    JButton b=new JButton("click");//creating instance of JButton
    b.setBounds(80,80,100, 40);//x axis, y axis, width, height
    f.add(b);//adding button in JFrame
    f.setSize(300,200);//300 width and 200 height
    f.setLayout(null);//using no layout managers
    f.setVisible(true);//making the frame visible
}
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



■ Kết quả



Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Cách xây dựng đối tượng JFrame



■ Ví dụ 3.2:

```
public class GUI extends JFrame{  
    public static final int WIDTH =400;  
    public static final int HEIGHT =200;  
    public GUI(String title){  
        // gọi phương thức định nghĩa khung giao diện  
        initGUI(title);  
    }  
    private void initGUI(String title){  
        setTitle(title);  
        setSize(WIDTH,HEIGHT);  
        setLocationRelativeTo(null);  
    }  
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



```
getContentPane().setBackground(Colour.WHITE);
setDefaultCloseOperation(EXIT_ON_CLOSE);
setResizable(false);
setLayout(new CardLayout());
}
}

public class Main{
    public static void main(String[] args){
        GUI gui = new GUI("Vi du");
        gui.setVisible(true);
    }
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Xử lý các bộ sự kiện trên JFrame



WindowListener:

- Cú pháp:

```
object.addActionListener(new ActionListener(){
    @Override
    public void actionPerformed(ActionEvent e){
        // câu lệnh thực thi
    }
});
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
import static javax.swing.WindowConstants.EXIT_ON_CLOSE;

public class JFrame1 {
    public static void main(String[] args) {
        JFrame f=new JFrame();
        JButton b=new JButton("click");
        b.setBounds(80,80,100,40);
        f.add(b);
        //adding button in JFrame
    }
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



```
b.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        int a=JOptionPane.showConfirmDialog(f, "Bạn chắc chắn muốn thoát chứ?");
        if(a==JOptionPane.YES_OPTION)
            System.exit(0);
    }
});
f.setSize(300,200);//300 width and 200 height
f.setLayout(null);//using no layout managers
f.setVisible(true);//making the frame visible
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

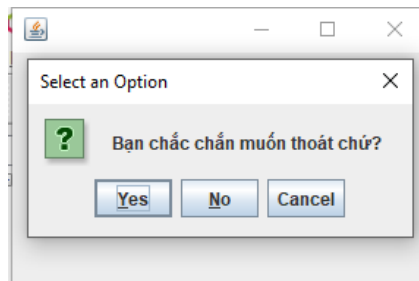
Event - WindowListener



```
f.addWindowListener(new WindowAdapter() {  
    @Override  
    public void windowClosing(WindowEvent e) {  
        int a=JOptionPane.showConfirmDialog(f, "Bạn chắc chắn muốn  
        thoát chứ?");  
        if(a==JOptionPane.YES_OPTION)  
        //      System.exit(0);  
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    else  
  
        f.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);  
    }  
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JPanel



- JPanel là một lớp container đơn giản nhất. Nó cung cấp không gian để các ứng dụng có thể thêm các component vào. Nó được kế thừa từ lớp JComponents.
- JPanel không có thanh tiêu đề (title bar).

Chương 3. Lập trình giao diện đồ họa

Jpanel – Một số constructor phổ biến



- `Jpanel()`: Tạo một JPanel mới với một double buffer và một Flow Layout.
- `JPanel(boolean isDoubleBuffered)`: Tạo một JPanel mới với Flow Layout và trình đếm đã xác định.
- `JPanel(LayoutManager layout)`: Tạo một JPanel mới với Layout Manager đã cho

Chương 3. Lập trình giao diện đồ họa

Jpanel – Ví dụ



Ví dụ 3.3

```
import java.awt.*;
import javax.swing.*;
public class PanelExample {
    PanelExample()
    {
        JFrame f= new JFrame("Panel Example");
        JPanel panel=new JPanel();
        panel.setBounds(40,80,200,200);
        panel.setBackground(Color.gray);
        JButton b1=new JButton("Button 1");
        b1.setBounds(50,100,80,30);
```

Lập trình Java

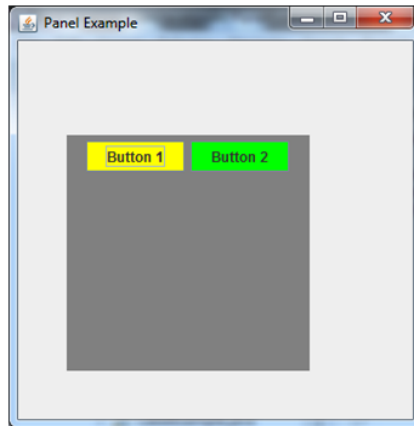
Chương 3. Lập trình giao diện đồ họa



```
        b1.setBackground(Color.yellow);
        JButton b2=new JButton("Button 2");
        b2.setBounds(100,100,80,30);
        b2.setBackground(Color.green);
        panel.add(b1); panel.add(b2);
        f.add(panel);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);    }
    public static void main(String args[]) {
        new PanelExample();
    } }
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



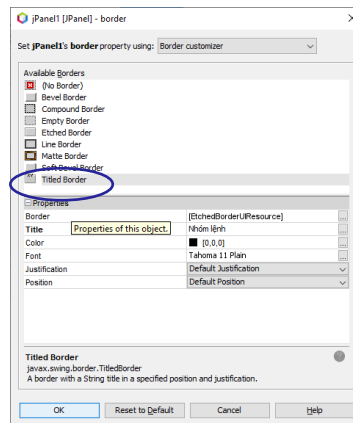
Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Jpanel – Tạo border Title



- Properties → Chọn border → Title Border



Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JLabel



- JLabel: là đối tượng giao diện hiển thị một tiêu đề có tác dụng chú thích hoặc thông báo trên giao diện
- JLabel: không sửa được văn bản hiển thị đối với người dùng.

Chương 3. Lập trình giao diện đồ họa

JLabel – Một số constructor phổ biến



- JLabel(): tạo một thể hiện JLabel không có hình ảnh và tiêu đề trống.
- JLabel(String s): tạo một thể hiện JLabel với text cụ thể.
- JLabel(Icon i): tạo một thể hiện JLabel với 1 ảnh xác định.
- JLabel(String s, Icon i, int horizontalAlignment): tạo một thể hiện JLabel với text xác định có ảnh và được căn chỉnh theo chiều ngang.

Chương 3. Lập trình giao diện đồ họa

JLabel – Một số phương thức thường dùng



- `String getText()`: trả về chuỗi mà label đang hiển thị.
- `void setText(String text)`: thiết lập 1 dòng đơn chuỗi text hiển thị trên component.
- `void setHorizontalAlignment(int alignment)`: thiết lập nội dung hiển thị của label theo hàng ngang trục X.
- `Icon getIcon()`: trả về hình ảnh mà label hiển thị.
- `int getHorizontalAlignment()`: trả về dạng căn chỉnh của nội dung label theo trục X.

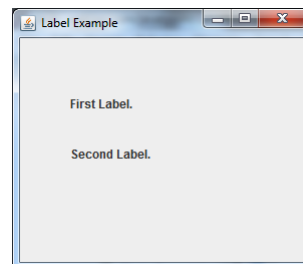
Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JLabel – Ví dụ 3.5



```
import javax.swing.*;
class LabelExample{
public static void main(String args[]) {
    JFrame f= new JFrame("Label Example");
    JLabel l1,l2;
    l1=new JLabel("First Label.");
    l1.setBounds(50,50, 100,30);
    l2=new JLabel("Second Label.");
    l2.setBounds(50,100, 100,30);
    f.add(l1); f.add(l2);
    f.setSize(300,300);
    f.setLayout(null);
    f.setVisible(true); } }
```



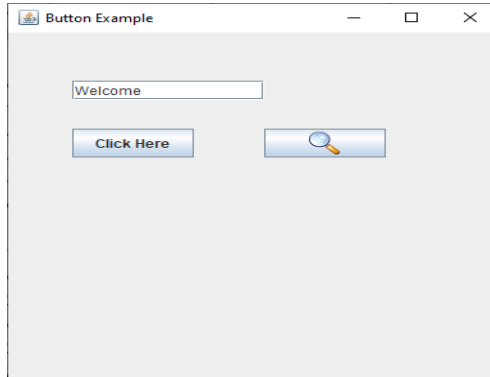
Lập trình Java

Chương 3. Lập trình giao diện đồ họa

TextField



- JTextField: đối tượng dùng để nhập dòng dữ liệu đơn.



Lập trình Java

Chương 3. Lập trình giao diện đồ họa

TextField – Một số constructor thường dùng



- JTextField(): tạo 1 textfield trống mới
- JTextField(String text): tạo 1 textfield có nội dung là text.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JTextField- Một số phương thức thường dùng



- `setText(String st)`: thiết lập chuỗi `st` cho đối tượng
- `requestFocus()`: thiết lập vị trí con trỏ soạn thảo.
- `getText()`: trả về chuỗi text của đối tượng.

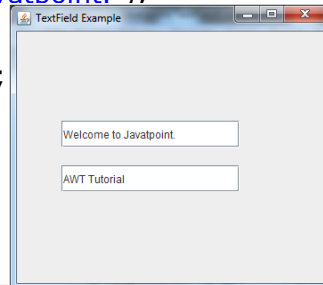
Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JTextField – Ví dụ



```
import javax.swing.*;  
class TextFieldExample {  
    public static void main(String args[]) {  
        JFrame f= new JFrame("TextField Example");  
        JTextField t1,t2;  
        t1=new JTextField("Welcome to Javatpoint.");  
        t1.setBounds(50,100, 200,30);  
        t2=new JTextField("AWT Tutorial");  
        t2.setBounds(50,150, 200,30);  
        f.add(t1); f.add(t2);  
        f.setSize(400,400);  
        f.setLayout(null);  
        f.setVisible(true); } }
```



Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JButton



- JButton: là một component được sử dụng để tạo ra một nút có nhãn được thực thi một cách độc lập. Thực thi 1 hành động nào đó khi người dùng nhấn nút.

Chương 3. Lập trình giao diện đồ họa

Jbutton – Một số constructor phổ biến



- JButton(): tạo ra 1 nút không có text cũng như ảnh.
- JButton(String s): tạo ra nút có text.
- JButton(Icon i): tạo ra nút có ảnh

Chương 3. Lập trình giao diện đồ họa

Jbutton – Một số method thường dùng



- `void setText(String s)`: dùng để thiết lập text cho nút .
- `String getText()`: trả về text của nút
- `void setEnabled(boolean b)`: để thiết lập nút được kích hoạt hay không.
- `void setIcon(Icon b)`: thiết lập ảnh cho nút.
- `Icon getIcon()`: trả về icon đang dung của nút.
- `void setMnemonic(int a)`: thiết lập mnemonic cho nút.
- `void addActionListener(ActionListener a)`: dùng để thêm action listener cho nút.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Jbutton – Một số method thường dùng



- Ẩn/ hiện nút: `setVisible(Boolean value)`: nếu value nhận false thì ẩn nút, ngược lại hiện nút.
- `setEditable(Boolean value)`: nếu bằng false thì không cho phép sửa nội dung của đối tượng. Mặc định là true
- `setToolTipText(String st)`: hiện chuỗi st khi ta thực hiện thao tác đưa chuột tới đối tượng.

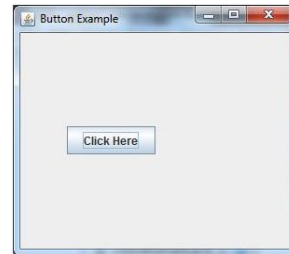
Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Jbutton – Ví dụ



```
import javax.swing.*;  
  
public class ButtonExample {  
    public static void main(String[] args) {  
        JFrame f=new JFrame("Button Example");  
        JButton b=new JButton("Click Here");  
        b.setBounds(50,100,95,30);  
        f.add(b);  
        f.setSize(400,400);  
        f.setLayout(null);  
        f.setVisible(true);  
    }  
}
```



Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Jbutton – Sự kiện Action Listener



```
import java.awt.event.*;  
import javax.swing.*;  
  
public class ButtonExample {  
    public static void main(String[] args) {  
        JFrame f=new JFrame("Button Example");  
        final JTextField tf=new JTextField();  
        tf.setBounds(50,50, 150,20);  
        JButton b=new JButton("Click Here");  
        b.setBounds(50,100,95,30);  
        JButton c= new JButton(new  
            ImageIcon("E:\\ImageIcon\\Find.png"));  
        c.setBounds(200,100 , 95, 30);  
    }  
}
```

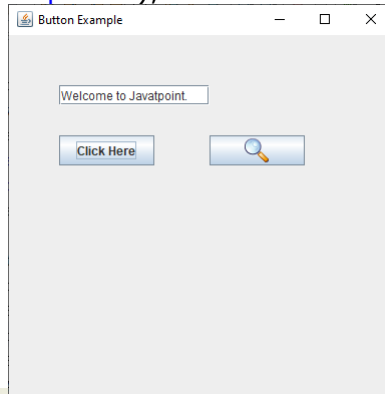
Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Jbutton – Sự kiện Action Listener



```
b.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
        tf.setText("Welcome to Javatpoint.");
    }
});
f.add(b);f.add(tf); f.add(c);
f.setSize(400,400);
f.setLayout(null);
f.setVisible(true);
}
```



Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Jbutton – Hiển thị Dialog



```
import javax.swing.*;
import java.awt.event.*;
public class JButtonAction {
    public static void main(String[] args) {
        JFrame f=new JFrame("Button Example");
        final JTextField tf=new JTextField();
        tf.setBounds(50,50, 150,20);
        JButton b=new JButton("Click Here");
        b.setBounds(50,100,95,30);
        b.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent e){
                tf.setText("Welcome to Javatpoint.");
            }
        });
    }
}
```

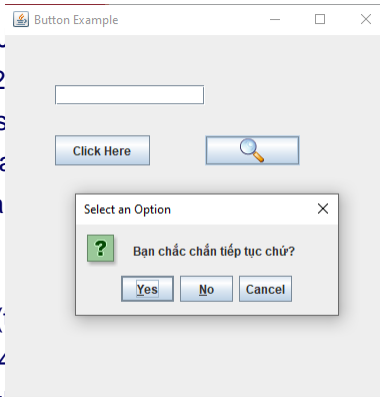
Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Jbutton – Hiển thị Dialog



```
JButton c= new JButton("Click Here");
c.setBounds(200, 100, 100, 30);
c.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        JOptionPane.showMessageDialog(f, "Bạn chắc chắn tiếp tục chứ?", "chắc chắn tiếp tục", JOptionPane.YES_NO_CANCEL_OPTION);
    }
});
f.add(b);f.add(c);
f.setSize(400, 400);
f.setLayout(null);
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
f.setVisible(true);
} }
```



chắc chắn tiếp tục

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Jbutton – Ví dụ (tiếp)



- Xử lý nhiều button

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Bài tập



1. Viết chương trình giải phương trình bậc nhất
2. Viết chương trình nhập vào hệ số a và b. Thực hiện tính tổng, thương, hiệu, tích.
3. Viết chương trình nhập vào 1 số nguyên n, sau đó in ra màn hình các số nguyên tố nhỏ hơn n.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JCheckBox



- Đối tượng cho phép chọn/ không chọn giá trị
- Cho phép lựa chọn đồng thời nhiều giá trị
- Một số constructor thường dùng:
 - JCheckBox(): tạo ra 1 nút check box chưa được lựa chọn không có text và icon.
 - JCheckBox(String s): tạo ra 1 checkbox chưa có lựa chọn có text.
 - JCheckBox(String text, boolean selected): tạo 1 checkbox có text, hoặc hoặc không được lựa chọn ngay từ đầu.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JCheckBox



- Các thuộc tính thường dùng:
 - Text: văn bản hiển thị trên đối tượng
 - Selected: đối tượng được lựa chọn hay không
- Một số phương thức thường dùng:
 - `protected String paramString():` trả về xâu đại diện của JCheckBox.
 - `boolean isSelected():` trả về true khi JCheckBox đang được select, false là ngược lại.
 - `setSelect(Boolean value):` dùng để set trạng thái cho JCheckBox là đang được chọn hoặc đang được bỏ chọn.

Lập trình Java

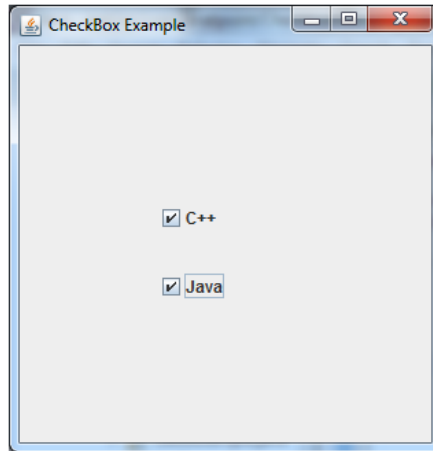
Chương 3. Lập trình giao diện đồ họa



```
import javax.swing.*;  
  
public class CheckBoxExample {  
    CheckBoxExample(){  
        JFrame f= new JFrame("CheckBox Example");  
        JCheckBox checkBox1 = new JCheckBox("C++");  
        checkBox1.setBounds(100,100, 50,50);  
        JCheckBox checkBox2 = new JCheckBox("Java", true);  
        checkBox2.setBounds(100,150, 50,50);  
        f.add(checkBox1);    f.add(checkBox2);  
        f.setSize(400,400); f.setLayout(null); f.setVisible(true);  
    }  
    public static void main(String args[]) {  
        new CheckBoxExample();    }  
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Java JCheckBox Example with ItemListener



```
import javax.swing.*;
import java.awt.event.*;
public class CheckBoxExample {
    CheckBoxExample(){
        JFrame f= new JFrame("CheckBox Example");
        final JLabel label = new JLabel();
        label.setHorizontalAlignment(JLabel.CENTER);
        label.setSize(400,100);
        JCheckBox checkbox1 = new JCheckBox("C++");
        checkbox1.setBounds(150,100, 50,50);
        JCheckBox checkbox2 = new JCheckBox("Java");
        checkbox2.setBounds(150,150, 50,50);
        f.add(checkboxbox1); f.add(checkboxbox2); f.add(label);
    }
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



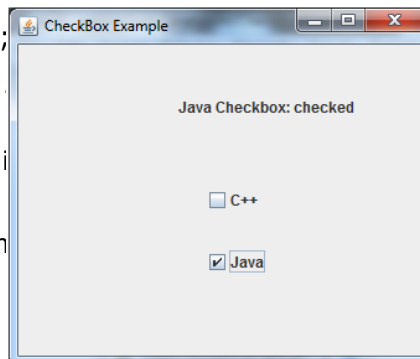
```
checkbox1.addItemListener(new ItemListener() {  
    public void itemStateChanged(ItemEvent e) {  
        label.setText("C++ Checkbox: "  
            + (e.getStateChange() == 1 ? "checked" : "unchecked"));  
    }  
});  
checkbox2.addItemListener(new ItemListener() {  
    public void itemStateChanged(ItemEvent e) {  
        label.setText("Java Checkbox: "  
            + (e.getStateChange() == 1 ? "checked" : "unchecked"));  
    }  
});
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



```
f.setSize(400,400);  
f.setLayout(null);  
f.setVisible(true)  
}  
public static void main()  
{  
    new CheckBoxExam  
}  
}
```



Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JRadioButton



- Đối tượng cho phép chọn/không chọn giá trị.
- Cho phép chọn một giá trị tại 1 thời điểm
- Một số constructor thường dùng:
 - `JRadioButton()`: tạo ra 1 radio button không có text và không được lựa chọn.
 - `JRadioButton(String s)`: tạo 1 radio button có text và không được lựa chọn.
 - `JRadioButton(String s, boolean selected)`: tạo radio button có text và trạng thái được chọn.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JRadioButton



- Một số phương thức thường dùng:
 - `isSelected()`: trả về true nếu đối tượng được check
- Lưu ý:
 - Các radio button nên được đặt trong 1 button group

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JRadioButton – Ví dụ 1



Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JRadioButton – Ví dụ

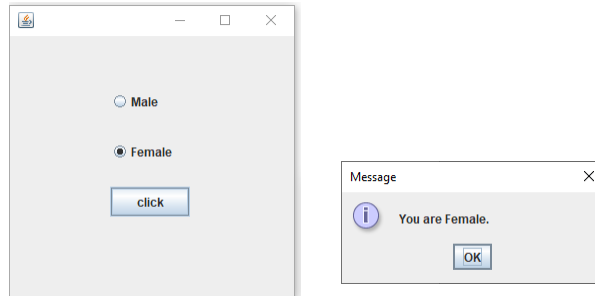


```
public class JRadioExample extends javax.swing.JFrame {  
    public JRadioExample() {  
        initComponents();  
        buttonGroup1.add(btnRed);  
        buttonGroup1.add(btnGreen);  
        buttonGroup1.add(btnBlue);    }  
    private void btnRedActionPerformed(java.awt.event.ActionEvent evt)  
    {  
        if(btnRed.isSelected())  
            jLabel1.setForeground(Color.red);  
    }  
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JRadioButton – Ví dụ 2



Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JRadioButton – Ví dụ 2



```
public class JRadioEx1 extends JFrame implements
ActionListener{
    JRadioButton rb1,rb2;
    JButton b;
    JRadioEx1 (){
        rb1=new JRadioButton("Male");
        rb1.setBounds(100,50,100,30);
        rb2=new JRadioButton("Female");
        rb2.setBounds(100,100,100,30);
        ButtonGroup bg=new ButtonGroup();
        bg.add(rb1);bg.add(rb2);
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JRadioButton – Ví dụ 2



```
b=new JButton("click");  
b.setBounds(100,150,80,30);  
b.addActionListener(this);  
add(rb1);add(rb2);add(b);  
setSize(300,300);  
setLayout(null);  
setVisible(true);  
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JRadioButton – Ví dụ 2



```
public void actionPerformed(ActionEvent e){  
    if(rb1.isSelected()){  
        rb1.JOptionPane.showMessageDialog(this,"You are Male.");  
    }  
    if(rb2.isSelected()){  
        JOptionPane.showMessageDialog(this,"You are Female.");  
    }  
}  
  
public static void main(String args[]){  
    new JRadioEx1();  
}}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

ArrayList



- Là một đối tượng được Java định nghĩa để biểu diễn một danh sách các phần tử mà số lượng phần tử có thể thay đổi được.
- ArrayList quản lý các phần tử của danh sách giống như mảng 1 chiều.
- ArrayList thường được sử dụng để biểu diễn danh sách các phần tử có kiểu dữ liệu cấu trúc.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



- Khai báo:
`ArrayList <KDL> ten_danh_sach=new ArrayList();`
- Ví dụ:
 - `ArrayList <Integer> listInt=new ArrayList();`
 - `ArrayList <Student> studentList=new ArrayList();`

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

ArrayList – Một số phương thức



- Lấy số lượng phần tử của danh sách:
 - Cú pháp: `size()`
 - Ví dụ: `int size=listInt.size();`
- Thêm phần tử vào danh sách:
 - Cú pháp: `add(giá trị);` hoặc `add(int index, Giá trị);`
 - Ví dụ:
`listInt.add(2);`
`listInt.add(0,3); // listInt gồm {3,2}`

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

ArrayList – Một số phương thức



- Thay thế phần tử trong danh sách:
 - Cú pháp: `set(index, giá_ trị);`
 - Ví dụ: `listInt.set(1,5); // listInt={3,5}`
- Xóa phần tử khỏi danh sách:
 - Cú pháp: `remove(int index)`
 - Ví dụ: `listInt.remove(0); // listInt={5}`
- □ Lấy giá trị phần tử trong danh sách:
 - Cú pháp: `get(int index)`
 - Ví dụ: `int so=listInt.get(0); // so=5;`

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

ArrayList – Một số phương thức



- Kiểm tra danh sách rỗng:
 - Cú pháp: isEmpty()
 - Ví dụ: boolean kt=listInt.isEmpty(); // kt=false;
- Xóa tất cả các phần tử khỏi danh sách:
 - Cú pháp: clear()
 - Ví dụ: listInt.clear();
- Tìm vị trí xuất hiện của phần tử trong danh sách:
 - Cú pháp: indexOf(KDL giatri)
 - Ví dụ:

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

ArrayList – Một số phương thức



- Kiểm tra phần tử có tồn tại trong danh sách
 - Cú pháp: contains(KDL giá trị)
 - Ví dụ:
boolean kt=studentList.getHoTen().contains("Hoa");
- Duyệt danh sách ArrayList
 - Sử dụng For:

```
for(int i=0; i < listInt.size(); i++) {  
    System.out.printf("%3d", listInt.get(i));  
}
```
 - Sử dụng Foreach:

```
for(KDL tên_biến: tên_danh_sách) {  
    <Lệnh>;  
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JComboBox



- Được dùng để tạo ra các menu trái xuống các lựa chọn. Mục đã chọn sẽ được hiển thị ngay trên đầu của menu.
- Các constructor thường dùng:
 - **JComboBox():** tạo 1 combobox với mô hình dữ liệu ngầm định.
 - **JComboBox(Object[] items):** tạo 1 combobox chứa các phần tử trong mảng chỉ định.
 - **JComboBox(Vector<?> items):** tạo 1 combobox chứa các phần tử trong vecto đã được chỉ định.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Jcombobox- Các phương thức thường dùng



- **getItemAt(int index):** trả về item ở vị trí index, phần tử đầu tiên của combo được hiểu ở vị trí 0, nếu index nằm ngoài phạm vi của combo sẽ trả về null.
- **void addItem(Object anObject):** dùng để thêm 1 item vào danh sách.
- **void removeItem(Object anObject):** dùng để xóa 1 item trong danh sách item.
- **void removeAllItems():** dùng để xóa toàn bộ item trong danh sách.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Jcombobox- Các phương thức thường dùng



- **getSelectedIndex():** trả về chỉ số của item được chọn
- **getSelectedItem():** trả về nội dung item đang được lựa chọn
- **void setEditable(boolean b):** dùng để thiết lập Combo box có sửa được hay không.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JComboBox – Ví dụ 1



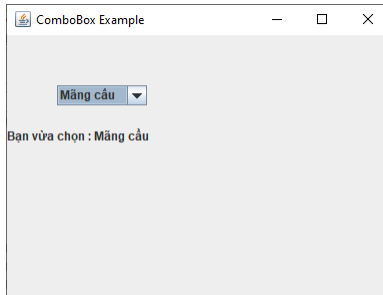
```
import javax.swing.*;

public class JComboExample {
    JFrame f;
    JComboExample(){
        f=new JFrame("ComboBox Example");
        String combo[]{"Cam","Quýt","Ổi","Mãng cầu","Lê"};
        JComboBox cb=new JComboBox(combo);
        cb.setBounds(50, 50,90,20);
        f.add(cb);          f.setLayout(null);
        f.setSize(400,300);  f.setVisible(true);    }
    public static void main(String[] args) {
        new JComboExample();    } }
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JComboBox – Ví dụ 2



Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JComboBox – Ví dụ 2



```
public class JComboExample1 {  
    JFrame f;  
    JComboExample1(){  
        f=new JFrame("ComboBox Example");  
        String combo[]={"Cam","Quýt","Ổi","Măng cầu","Lê"};  
        JComboBox cb=new JComboBox(combo);  
        cb.setBounds(50, 50,90,20);  
        f.add(cb);  
        f.setLayout(null);  
        f.setSize(400,300);  
        f.add(label1);  
    }  
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JComboBox – Ví dụ 1



```
JLabel label1 = new JLabel();
label1.setSize(300, 200);
cb.addActionListener(new ActionListener(){
    @Override
    public void actionPerformed(ActionEvent e) {
        label1.setText("Bạn vừa chọn : " + cb.getSelectedItem());
    }
});
f.setDefaultCloseOperation(EXIT_ON_CLOSE);
f.setVisible(true);
}
public static void main(String[] args) {
    new JComboExample();
} }
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JComboBox – Sử dụng Model thêm dữ liệu



- Sử dụng DefaultComboBoxModel
- Khai báo đối tượng model:
 - `DefaultComBoBoxModel model = new DefaultComBoBoxModel();`
 - Thêm đối tượng vào combobox:
 - `Model.addelement(element);`
 - `JComBoBoxx cmb= new JComboBox(model);`

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



```
import javax.swing.*;

public class JCombo1 {
    JFrame f;
    JCombo1(){
        f=new JFrame("ComboBox Example");
        DefaultComboBoxModel model =new DefaultComboBoxModel();
        model.addElement("");
        model.addElement("Ghế");
        model.addElement("Bàn");
        model.addElement("Sofa");
        model.addElement("Chiếu");
        JComboBox cb=new JComboBox(model);
        cb.setSelectedIndex(0);
        cb.setBounds(50, 50,90,20);
    }
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JTable



- Là đối tượng được dùng để hiển thị dữ liệu dạng bảng, bao gồm các hàng và các cột.
- Một số constructor:
 - **JTable():** tạo ra 1 bảng gồm các ô trống.
 - **JTable(Object[][], rows, Object[] columns):** tạo 1 bảng với dữ liệu ban đầu

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Jtable – Ví dụ



```
import javax.swing.*;

public class TableExample {
    JFrame f;
    TableExample(){
        f=new JFrame("Bảng điểm");
        Object data[][]={ {"Sv01","Nguyen Van An","7.5"},
                           {"Sv02","Le THI THuong","8.0"},
                           {"Sv03","Hoang Ha","6.5"} };
        String column[]={"ID","NAME","MARK"};
        JTable jt=new JTable(data,column);
        jt.setBounds(30,40,200,300);
    }
}
```

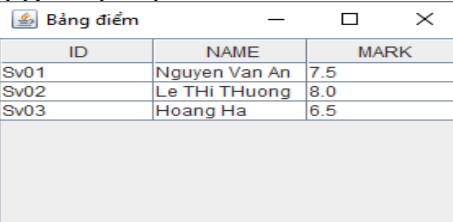
Lập trình Java

Chương 3. Lập trình giao diện đồ họa



```
JScrollPane sp=new JScrollPane(jt);
f.add(sp);    f.setSize(300,400);
f.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
f.setVisible(true);    }

public static void main(String[] args) {
    new TableExample();    }
}
```



ID	NAME	MARK
Sv01	Nguyen Van An	7.5
Sv02	Le THI THuong	8.0
Sv03	Hoang Ha	6.5

Chương 3. Lập trình giao diện đồ họa

Jtable – Một số thuộc tính thường dùng



- **setModel(Table Model<KDL> model):** định nghĩa danh sách dữ liệu hiển thị trong Jtable.
- **setColumnModel(TableColumnModel model):** định nghĩa tên hiển thị của các cột trong bảng.
- **isSelected(int r, int c):** trả về true khi ô dữ liệu ở vị trí (r,c) đang được chọn, ngược lại trả về false.
- **isColumnSelected(int c):** trả về true khi ô dữ liệu ở cột c đang được chọn, ngược lại trả về false.
- **isRowSelected(int r):** trả về true khi ô dữ liệu ở hàng r đang được chọn, ngược lại trả về false.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Jtable – Một số thuộc tính thường dùng



- **getValueAt(int r, int c):** trả về giá trị dữ liệu tại vị trí (r,c).
- **getRowCount():** trả về số dòng trong bảng.
- **getColumnCount():** trả về số cột trong bảng.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Jtable – Ví dụ



ĐANH SÁCH QUẢN LÝ ĐIỂM

Mã SV:

Tên SV:

Điểm:

STT	Mã Sinh viên	Họ và tên sinh v...	Điểm
1	s1	Trần Thị Linh	8.0
2	s2	Hoàng Văn Thái	6.0
3	s3	Trịnh Tuấn Kiệt	8.0
4	s4	Nguyễn Thị Văn...	5.0

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



```
private void btnThemActionPerformed(java.awt.event.ActionEvent  
    evt) {  
        studentList.add(new SinhVien(tfMaSV.getText(),  
            tfTenSV.getText(),  
            Double.parseDouble(tfDiem.getText())));  
        int i=studentList.size()-1;  
        SinhVien s=studentList.get(i);  
        model.addRow(new Object []{  
            ++i,s.getID(),s.getHoTen(),s.getDiem()});  
        jTable1.setModel(model);  
    }  
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



```
private void btnXoaActionPerformed
    (java.awt.event.ActionEvent evt)
{
    int i=jTable1.getSelectedRow();
    if(i>=0){
        model.removeRow(i);
        jTable1.setModel(model);
    }
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



```
private void btnSuaActionPerformed
    (java.awt.event.ActionEvent evt) {
    int i=jTable1.getSelectedRow();
    if (i>=0){
        model.setValueAt(tfMaSV.getText(), i, 1);
        model.setValueAt(tfTenSV.getText(), i, 2);
        model.setValueAt(tfDiem.getText(), i, 3);
    }
    jTable1.setModel(model); }
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



```
private void jTable1MouseClicked  
(java.awt.event.MouseEvent evt) {  
    int i=jTable1.getSelectedRow();  
    if (i>=0){  
        tfMaSV.setText(jTable1.getModel().getValueAt(i, 1).toString());  
        tfTenSV.setText(jTable1.getModel().getValueAt(i, 2).toString());  
        tfDiem.setText(jTable1.getModel().getValueAt(i, 3).toString());  
    } } }
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

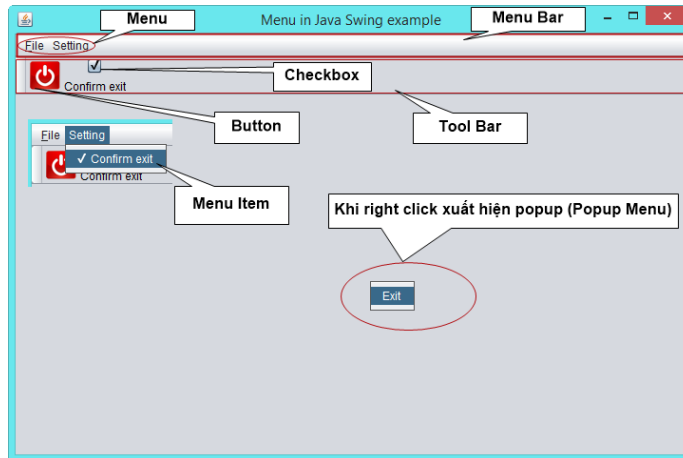


```
private void hienThiList(){  
    int i=1;  
    while (i<=studentList.size()-1){  
        SinhVien s=studentList.get(i);  
        model.addRow(new Object []  
            {i++,s.getID(),s.getHoTen(),s.getDiem()});  
    }  
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JMenu



Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JMenu



- **Các bước tạo menu:**
 - Tạo thanh **Menu Bar**
 - Tạo các menu chứa các menu con: sử dụng các **Jmenu**
 - Tạo các mục menu con: sử dụng các **Menu Item**
- **Viết event cho các menu Item:**
 - Chọn menu Item cần viết event
 - Right click, chọn Events → Actions → chọn event cần thực hiện

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JMenu



- **Tạo phím tắt:** thuộc tính accelerator
- **Tạo icon:** thuộc tính icon
- **Tạo kí tự gạch chân:** mnemonic

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Menu



```
import javax.swing.*;

public class Jmenu {
    JMenu menu, submenu;
    JMenuItem i1, i2, i3, i4, i5;

    Jmenu(){
        JFrame f= new JFrame("Menu and MenuItem Example");
        JMenuBar mb=new JMenuBar();   menu=new JMenu("Menu");
        submenu=new JMenu("Sub Menu");
        i1=new JMenuItem("Item 1");   i2=new JMenuItem("Item 2");
        i3=new JMenuItem("Item 3");   i4=new JMenuItem("Item 4");
        i5=new JMenuItem("Item 5");
        menu.add(i1); menu.add(i2); menu.add(i3);
        submenu.add(i4); submenu.add(i5);   menu.add(submenu);
        mb.add(menu);   f.setJMenuBar(mb);   f.setSize(400,400);
    }
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

PopUp Menu



- PopupMenu là cửa sổ động hiển thị ở một vị trí đặc biệt trên các component.
- Các constructor thường dùng:
 - JPopupMenu(): tạo PopupMenu ngầm định.
 - JPopupMenu(String label): tạo popupMenu có nội dung xác định.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Thực thi PopUp Menu



- Đưa PopUp Menu vào Frame
- Thêm các mục Menu: right click vào popup Menu trên cửa sổ Navigator, chọn Add From palette --> chọn mục tương ứng.
- Chọn đối tượng chứa JPopupMenu
 - Trong thuộc tính Component Popup menu: chọn tên PopupMenu cần hiển thị
- Viết event cho các mục menu: right click vào mục cần viết event, chọn Events → Actions → chọn sự kiện tương ứng.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



```
package com.mycompany.swing.JFrame;
import javax.swing.*;
import java.awt.event.*;
public class popUpMenu {
    popUpMenu(){
        final JFrame f= new JFrame("PopupMenu Example");
        final JPopupMenu popupmenu = new JPopupMenu("Edit");
        JMenuItem cut = new JMenuItem("Cut");
        JMenuItem copy = new JMenuItem("Copy");
        JMenuItem paste = new JMenuItem("Paste");
        popupmenu.add(cut);popupmenu.add(copy);
        popupmenu.add(paste);
    }
}
```

Lập trình Java

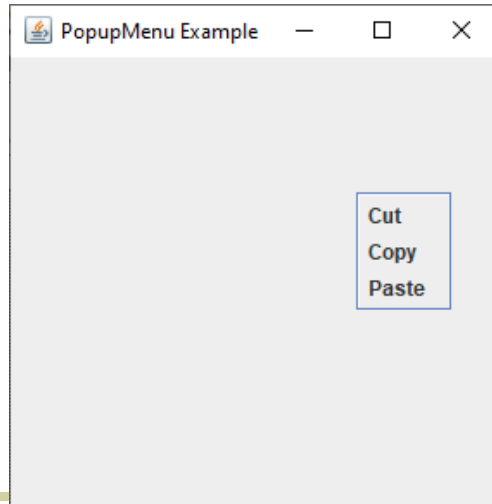
Chương 3. Lập trình giao diện đồ họa



```
f.addMouseListener(new MouseAdapter() {
    public void mouseClicked(MouseEvent e) {
        popupmenu.show(f , e.getX(), e.getY());
    }
});
f.add(popupmenu);
f.setSize(300,300);
f.setLayout(null);
f.setVisible(true);
}
public static void main(String args[]) {
    new popUpMenu();
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



Lập trình Java

Chương 3. Lập trình giao diện đồ họa

Layout Manager



Layout Manager dùng để sắp xếp các component theo 1 cách đặc biệt nào đó. Layout manager là một interface được thực thi bởi tất cả các lớp layout manager. Một số lớp thể hiện layout manager như sau:

- `java.awt.BorderLayout`
- `java.awt.FlowLayout`
- `java.awt.GridLayout`
- `java.awt.CardLayout`
- ...

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

BorderLayout



- BorderLayout được dùng để sắp xếp các component theo 5 vùng: north, south, east, west and center. Mỗi vùng có thể chứa 1 component duy nhất. BorderLayout cung cấp 5 hằng cho 5 vùng như sau:
 - **public static final int NORTH**
 - **public static final int SOUTH**
 - **public static final int EAST**
 - **public static final int WEST**
 - **public static final int CENTER**

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

BorderLayout – Các constructor thường dùng



- **BorderLayout():** creates a border layout but with no gaps between the components.
- **JBorderLayout(int hgap, int vgap):** creates a border layout with the given horizontal and vertical gaps between the components.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

BorderLayout - Example



```
import java.awt.*;  
import javax.swing.*;  
public class Border {  
    JFrame f;  
    Border(){  
        f=new JFrame();  
        JButton b1=new JButton("NORTH");  
        JButton b2=new JButton("SOUTH");  
        JButton b3=new JButton("EAST");  
        JButton b4=new JButton("WEST");  
        JButton b5=new JButton("CENTER");
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



```
        f.add(b1, BorderLayout.NORTH);  
        f.add(b2, BorderLayout.SOUTH);  
        f.add(b3, BorderLayout.EAST);  
        f.add(b4, BorderLayout.WEST);  
        f.add(b5, BorderLayout.CENTER);  
        f.setSize(300,300);  
        f.setVisible(true);  
    }  
    public static void main(String[] args) {  
        new Border();  
    }  
}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



Lập trình Java

Chương 3. Lập trình giao diện đồ họa

GridLayout



- GridLayout được dùng để sắp xếp các component trong 1 lưới ô hình chữ nhật. Mỗi component được hiển thị trong 1 hình chữ nhật.
- Các lớp Constructor của GridLayout
 - **GridLayout():** tạo 1 lưới hiển thị gồm 1 cột, 1 hàng có component.
 - **GridLayout(int rows, int columns):** tạo 1 lưới gồm row hàng và column cột nhưng không có đường kẻ giữa các component.
 - **GridLayout(int rows, int columns, int hgap, int vgap):** tạo 1 lưới gồm các hàng và cột đã chỉ định có đường kẻ giữa các hàng và cột.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



Lập trình Java

Chương 3. Lập trình giao diện đồ họa



```
import java.awt.*;  
import javax.swing.*;  
public class MyGridLayout{  
    JFrame f;  
    MyGridLayout(){  
        f=new JFrame();  
        JButton b1=new JButton("1");  
        JButton b2=new JButton("2");  
        JButton b3=new JButton("3");  
        JButton b4=new JButton("4");  
        JButton b5=new JButton("5");
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



```

JButton b6=new JButton("6");
JButton b7=new JButton("7");
JButton b8=new JButton("8");
JButton b9=new JButton("9");
f.add(b1);f.add(b2);f.add(b3);f.add(b4);f.add(b5);
f.add(b6);f.add(b7);f.add(b8);f.add(b9);
f.setLayout(new GridLayout(3,3));
//setting grid layout of 3 rows and 3 columns
f.setSize(300,300);
f.setVisible(true); }
public static void main(String[] args) {
    new MyGridLayout();
} }
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JTabPane



- N
- Các constructor thường dùng:
 - JTabbedPane(): tạo tabPane trống với vị trí ngầm định là JTabbedPane.Top.
 - JTabbedPane(int tabPlacement): tạo 1 Tabpane trống với vị trí tab dc chỉ định.

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JTabPane



- Các bước tạo JTabPanel:
 - Tạo JtabPanel
 - Chọn các Panel, kéo thả vào trên TabPane để tạo các tab
- Sự kiện thường dùng:
 - StateChange
 - GetSelectedIndex: Lấy chỉ số của tab đang dùng
 - Ví dụ: `int index=JTabpane1.getSelectedIndex();`
 - Các tab đánh chỉ số từ 0

Lập trình Java

Chương 3. Lập trình giao diện đồ họa

JTabPane - Example



```
import javax.swing.*;
public class TabbedPaneExample {
    JFrame f;
    TabbedPaneExample(){
        f=new JFrame();
        JTextArea ta=new JTextArea(200,200);
        JPanel p1=new JPanel();
        p1.add(ta);
        JPanel p2=new JPanel();
        JPanel p3=new JPanel();
        JTabbedPane tp=new JTabbedPane();
        tp.setBounds(50,50,200,200);
    }
}
```

Lập trình Java

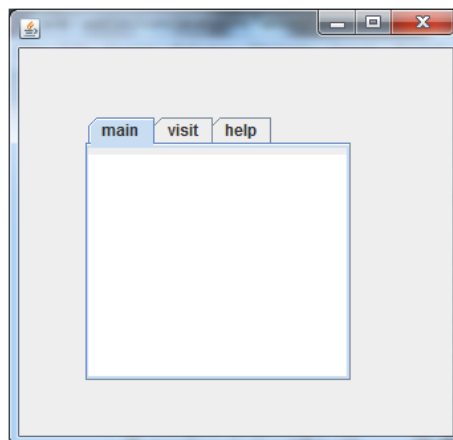
Chương 3. Lập trình giao diện đồ họa



```
tp.add("main",p1);
tp.add("visit",p2);
tp.add("help",p3);
f.add(tp);
f.setSize(400,400);
f.setLayout(null);
f.setVisible(true);
}
public static void main(String[] args) {
    new TabbedPaneExample();
}}
```

Lập trình Java

Chương 3. Lập trình giao diện đồ họa



Lập trình Java