MỤC LỤC

[MỤC LỤC 1](#_Toc75872452)

[TỔNG HỢP KIẾN THỨC 1](#_Toc75872453)

[Kiến Thức Chương 2 1](#_Toc75872454)

[Câu 1: Các phần tử cơ bản của Java 1](#_Toc75872455)

[Câu 2: Kiểu dữ liệu 1](#_Toc75872456)

[Câu 3: Kiểu dữ liệu cơ sở 1](#_Toc75872457)

[Câu 4: Ép kiểu 1](#_Toc75872458)

[Câu 5: Biến số 2](#_Toc75872459)

[Câu 6: Những từ khóa của Java 2](#_Toc75872460)

[Câu 7: Kiểu dữ liệu tham chiếu 2](#_Toc75872461)

[Câu 8: Khai báo mảng 2](#_Toc75872462)

[Câu 9: Lớp trong Java 3](#_Toc75872463)

[Câu 10: Phương thức 3](#_Toc75872464)

[Câu 11: Phương thức khởi tạo 3](#_Toc75872465)

[Câu 12: Những phương thức được nạp chồng 3](#_Toc75872466)

[Câu 13: Những phương thức được ghi đè 3](#_Toc75872467)

[Câu 14: Quyền truy xuất 3](#_Toc75872468)

[Câu 15: Từ khóa bổ sung phương thức 4](#_Toc75872469)

[Câu 16: Phương thức khởi tạo của lớp dẫn xuất 4](#_Toc75872470)

[Câu 17: Các toán tử 4](#_Toc75872471)

[Câu 18: Toán tử số học 4](#_Toc75872472)

[Câu 19: Toán tử Bit 5](#_Toc75872473)

[Câu 20: Toán tử so sánh 5](#_Toc75872474)

[Câu 21: Toán tử logic 5](#_Toc75872475)

[Câu 22: Toán tử điều kiện 5](#_Toc75872476)

[Câu 23: Toán tử gán 5](#_Toc75872477)

[Câu 24: Thứ tự ưu tiên các toán tử 6](#_Toc75872478)

[Câu 25: Các ký tự định dạng xuất dữ liệu 6](#_Toc75872479)

[Câu 26: Các lệnh điều khiển 6](#_Toc75872480)

[Kiến Thức Chương 3 7](#_Toc75872481)

[Câu 1: Các khái niệm cơ bản trong lập trình hướng đối tượng 7](#_Toc75872482)

[Câu 2: Phương thức và lớp Final 7](#_Toc75872483)

[Câu 3: Lớp trừu tượng (Abstract) 8](#_Toc75872484)

[Câu 4: Lớp cụ thể 8](#_Toc75872485)

[Câu 5: Đa hình trong Java 8](#_Toc75872486)

[Câu 6: Phương thức Abstract 8](#_Toc75872487)

[Câu 7: Giao diện trong Java (Interface) 8](#_Toc75872488)

[Kiến Thức Chương 5 9](#_Toc75872489)

[Câu 1: Giới thiệu về JFC 9](#_Toc75872490)

[Câu 2: Lấy kích thước màn hình 9](#_Toc75872491)

[Câu 3: JPanel 9](#_Toc75872492)

[Câu 4: Dialog 9](#_Toc75872493)

[Câu 5: Các thành phần giao diện (Component) 10](#_Toc75872494)

[Câu 6: JLabel 10](#_Toc75872495)

[Câu 7: JTextField 10](#_Toc75872496)

[Câu 8: JTextArea 11](#_Toc75872497)

[Câu 9: JButton 11](#_Toc75872498)

[Câu 10: JSlider 12](#_Toc75872499)

[Câu 11: JCheckbox 12](#_Toc75872500)

[Câu 12: JRadionButton 12](#_Toc75872501)

[Câu 13: ButtonGroup 12](#_Toc75872502)

[Câu 14: JComboBox 13](#_Toc75872503)

[Câu 15: Layout Manager 13](#_Toc75872504)

[Câu 16: FlowLayout 13](#_Toc75872505)

[Câu 17: BorderLayout 13](#_Toc75872506)

[Câu 18: CardLayout 14](#_Toc75872507)

[Câu 19: GridLayout 14](#_Toc75872508)

[Câu 20: Không dùng Layout 14](#_Toc75872509)

[Câu 21: Xử lý sự kiện 14](#_Toc75872510)

[Câu 22: Xử lý sự kiện Button 16](#_Toc75872511)

[Câu 23: Xử lý sự kiện Slider 16](#_Toc75872512)

[Câu 24: Giao diện ItemListener 16](#_Toc75872513)

[Câu 25: Menu 16](#_Toc75872514)

[Câu 26: Đồ họa trong Java (Graphics) 17](#_Toc75872515)

[Câu 27: Vẽ các ký tự 17](#_Toc75872516)

[Câu 28: Vẽ đoạn thẳng 17](#_Toc75872517)

[Câu 29: Vẽ hình oval 17](#_Toc75872518)

[Câu 30: Vẽ hình chữ nhật 18](#_Toc75872519)

[Câu 31: Điều khiển màu 18](#_Toc75872520)

[Câu 32: Vẽ đồng hồ 18](#_Toc75872521)

[Kiến Thức Chương 8 18](#_Toc75872522)

[Câu 1: Các bước khởi tạo 1 server đơn giản 19](#_Toc75872523)

[Câu 2: Server 19](#_Toc75872524)

[Câu 3: Client 19](#_Toc75872525)

TỔNG HỢP KIẾN THỨC

Kiến Thức Chương 2

1. Các phần tử cơ bản của Java

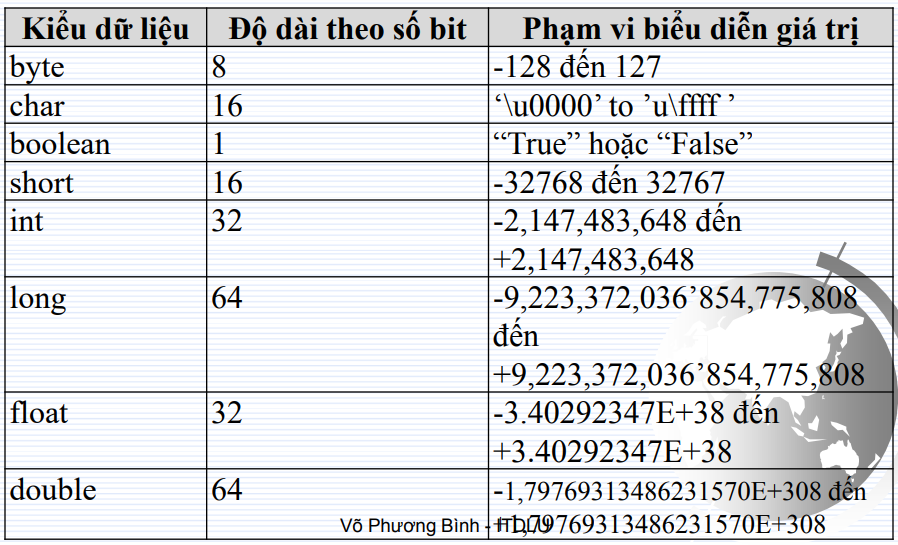
* Kiểu dữ liệu
* Lớp và phương thức (Classes & Methods)
* Biến số
* Toán tử
* Cấu trúc điều khiển

1. Kiểu dữ liệu

Gồm có:

* Kiểu dữ liệu cơ sở (Primitive Data Types)
* Kiểu dữ liệu tham chiếu (Reference data types)

1. Kiểu dữ liệu cơ sở



1. Ép kiểu

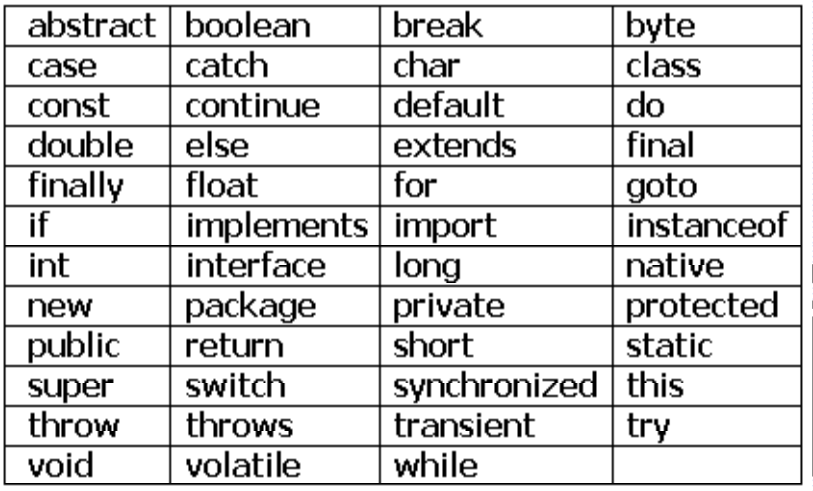
Kiểu dữ liệu này được chuyển đổi sang một kiểu dữ liệu khác

1. Biến số

Khai báo biến số gồm 3 thành phần:

* Kiểu dữ liệu của biến số
* Tên biến
* Giá trị ban đầu của biến (không bắt buộc)

1. Những từ khóa của Java



1. Kiểu dữ liệu tham chiếu

* Mảng (Array)
* Lớp (Class)
* Giao diện (Interface)

1. Khai báo mảng

Ba cách để khai báo mảng:

* datatype identifier [ ];
* datatype identifier [ ] = new datatype[size];
* datatype identifier [ ]= {value1,value2,….valueN};

1. Lớp trong Java

Cú pháp khai báo lớp (Class): Lớp = Thuộc Tính + Phương Thức

Có tính đa hình và tính đóng gói

class Classname {

var\_datatype variablename;

:

met\_datatype methodname(parameter\_list)

:

}

1. Phương thức

Phương thức được định nghĩa như là một hành động hoặc một tác vụ thật sự của đối tượng

Cú pháp:

access\_specifier modifier datatype method\_name(parameter\_list) { //body of method }

1. Phương thức khởi tạo

Là một phương thức đặc biệt dùng để khởi tạo giá trị cho các biến thành viên của lớp đối tượng, có cùng tên với tên lớp và không có giá trị trả về, được gọi khi đối tượng được tạo ra. Có 2 loại: Tường minh, Ngầm định

1. Những phương thức được nạp chồng

* Cùng ở trong một lớp
* Có cùng tên
* Khác nhau về danh sách tham số

Những phương thức được nạp chồng là một hình thức đa hình (polymorphism) trong quá trình biên dịch (compile time)

1. Những phương thức được ghi đè

* Có mặt trong lớp cha (superclass) cũng như lớp kế thừa (subclass)
* Được định nghĩa lại trong lớp kế thừa (subclass)

Những phương thức được ghi đè là một hình thức đa hình (polymorphism) trong quá trình thực thi (Runtime)

Phương thức con kế thừa các thuộc tính của phương thức cha bằng cách sử dụng super();

1. Quyền truy xuất

* **private** (chỉ sử dụng trong lớp đó)
* **public** (có thể được sử dụng từ các lớp bên ngoài)
* **protected** (chỉ được sử dụng từ các lớp kế thừa lớp đó)

1. Từ khóa bổ sung phương thức

* **static:** Dùng cho lớp (Không được sử dụng với bất kỳ biến nào khác)
* **abstract:** Trừu tượng, không cần định nghĩa
* **final:** Không được định nghĩa lại
* **synchronized**
* **volatile**

1. Phương thức khởi tạo của lớp dẫn xuất

* Có cùng tên với lớp dẫn xuất (subclass)
* Mệnh đề gọi constructor của lớp cha (superclass) phải là mệnh đề đầu tiên trong constructor của lớp dẫn xuất (subclass): super();

1. Các toán tử

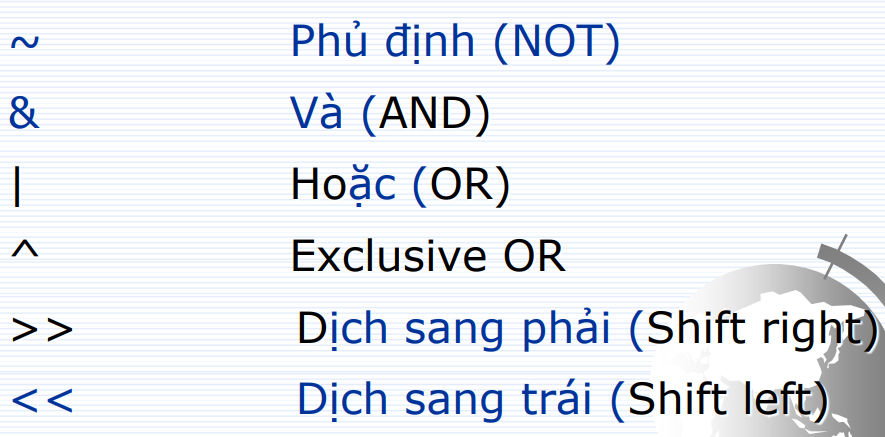
Các loại toán tử:

* Toán tử số học (Arithmetic operators)
* Toán tử dạng Bit (Bitwise operators)
* Toán tử so sánh (Relational operators)
* Toán tử logic (Logical operators)
* Toán tử điều kiện (Conditional operator)
* Toán tử gán (Assignment operator)

1. Toán tử số học

+, -, \*, /, %, +=, -=, \*=, /=. ++. –

1. Toán tử Bit



1. Toán tử so sánh

==, !=, <=, >=, <, >

1. Toán tử logic

&&, ||, !

1. Toán tử điều kiện

Cú pháp

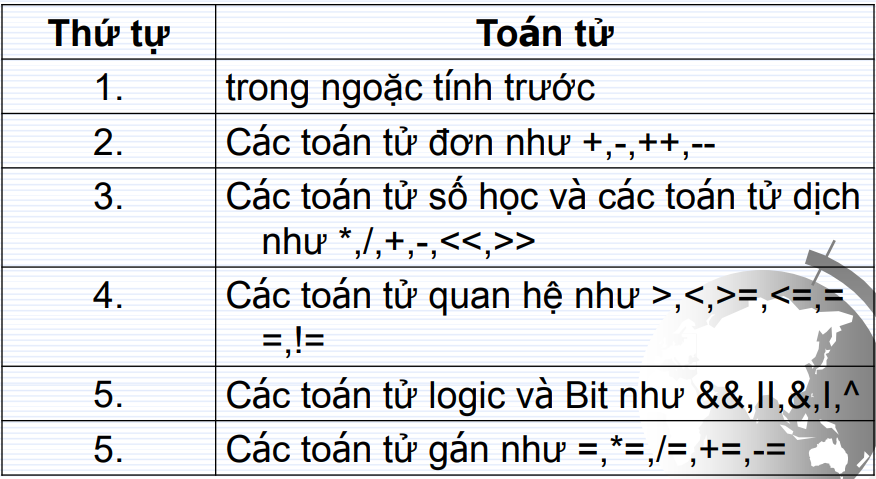
Biểu thức 1 ? Biểu thức 2 : Biểu thức 3;

* Biểu thức 1 Điều kiện kiểu Boolean trả về giá trị True hoặc False
* Biểu thức 2 Trả về giá trị nếu kết quả của mệnh đề 1 là True
* Biểu thức 3 Trả về giá trị nếu kết quả của mệnh đề 1 là False

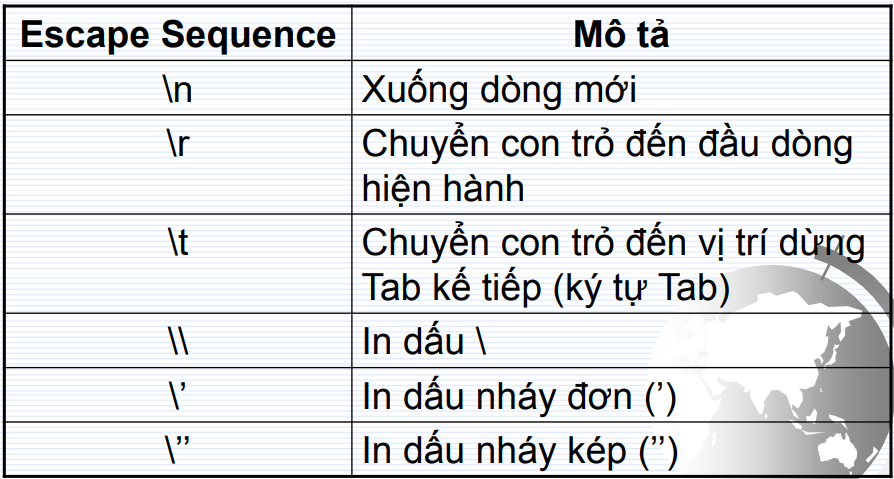
1. Toán tử gán

Giá trị có thể gán cho nhiều biến số

1. Thứ tự ưu tiên các toán tử



1. Các ký tự định dạng xuất dữ liệu



1. Các lệnh điều khiển

**Điều khiển rẻ nhánh:**

* Mệnh đề if-else
* Mệnh đề switch-case

**Vòng lặp (Loops):**

* Vòng lặp while
* Vòng lặp do-while
* Vòng lặp for

Kiến Thức Chương 3

1. Các khái niệm cơ bản trong lập trình hướng đối tượng

**Đóng gói hay Lớp (Class):** Gồm 2 thành phần cơ bản: thuộc tính và phương thức.

**Kế thừa (Inheritance):** lớp mới định nghĩa từ lớp có sẵn.

* Lớp con kế thừa thuộc tính và phương thức của lớp cha.
* Có 2 kiểu kế thừa: Đơn kế thừa và đa kế thừa

**Đa hình (Polymorphism)**

* Lớp trừu tượng - Abstraction
* Giao diện – Interface

**Dùng từ khóa extends:** Đơn kế thừa

class TwoDimensionalShape extends Shape{ ... }

Các thành phần **private** của lớp cha không được truy xuất trực tiếp từ lớp con.

Tất cả các thuộc tính khác giữ nguyên quyền truy xuất.

1. Phương thức và lớp Final

**Biến final:**

* Cho biết biến này không được sửa đổi sau khi khai báo
* Cần phải được gán giá trị khi khai báo

**Phương thức final**

* Không thể bị ghi đè bởi lớp con
* Các phương thức static và private được ngầm hiểu là final

**Lớp final**

* Không có lớp nào kế thừa được lớp final
* Tất cả các phương thức trong lớp final được ngầm hiều là final

1. Lớp trừu tượng (Abstract)

* Mục đích duy nhất là trở thành lớp cha, nghĩa là để cho các lớp khác kế thừa từ nó
* Không thể khởi tạo các đối tượng của một lớp trừu tượng nhưng vẫn có thể xác định các hàm tạo
* Quá chung chung để xác định các đối tượng thực
* Khai báo lớp với từ khóa **abstract**

1. Lớp cụ thể

* Có thể khởi tạo các đối tượng, cung cấp các thông tin chi tiết cụ thể

1. Đa hình trong Java

**Tính đa hình**

* Gọi một phương thức có thể sinh ra các hành động khác nhau, phụ thuộc vào đối tượng gọi.
* Lớp abstract + phương thức abstract 🡪 Tính đa hình

**Tham chiếu**

* Có thể tạo tham chiếu đến các lớp trừu tượng
* Không thể khởi tạo các đối tượng của các lớp trừu tượng

1. Phương thức Abstract

Phương thức abstract – Dùng từ khóa abstract

Bất kỳ lớp nào có phương thức trừu tượng đều phải là trừu tượng, các phương thức trừu tượng phải được ghi đè trong lớp con

1. Giao diện trong Java (Interface)

**Giao diện**

* Dùng từ khóa **interface**
* Có tập phương thức **public abstract**

**Sử dụng giao diện**

* Dùng từ khóa **implements**
* Nhiều interfaces thì sử dụng dấu phẩy để ngăn cách
* Phải định nghĩa tất cả phương thức abstract trong interface
* Cùng số lượng đối số, kiểu dữ liệu trả về.

Kiến Thức Chương 5

1. Giới thiệu về JFC

**JFC (Java Foundation Class, thuộc gói javax.swing)**

* Một bộ các lớp cho phép tạo GUI và chấp nhận các nhập liệu của người dùng thông qua bàn phím và chuột.
* Cung cấp các thành phần khác nhau để tạo GUI hiệu quả.

**Các thành phần JFC:**

* Vật chứa (Container)
* Thành phần (Component)
* Trình quản lý cách trình bày (Layout manager)
* Đồ họa (Graphic) và các tính năng vẽ (draw)

**Container:**

* Vùng chứa các thành phần giao diện.
* Kế thừa từ lớp Container.
* Là một cửa sổ.

Lớp Container nằm trong gói javax.swing

**Frame và Panel là các Container thường được sử dụng.**

* Frame là cửa sổ độc lập.
* Panel là vùng nằm trong cửa sổ khác

1. Lấy kích thước màn hình

**Lấy kích thước màn hình**

* Dimension d = Toolkit.getDefaultToolkit().getScreenSize();
* d.height; //Lấy chiều dài
* d.width; //Lấy chiều rộng

**Canh giữa màn hình**

* setLocation(x,y)?

1. JPanel

Nhóm các thành phần lại với nhau

1. Dialog

Lớp JDialog tương tự như lớp JFrame.

**Đối tượng Dialog được tạo như sau:**

* JDialog dialog = new JDialog();
* dialog.setSize(int width, int height);
* dialog.show();

**Dialog có tham số Frame:**

* JDialog dialog = new JDialog(JFrame f, String title, boolean modal);

1. Các thành phần giao diện (Component)

Các thành phần (Component) được chứa trong Container ( Frame, Panel).

Ví dụ: các thành phần Button, Label, TextField, …

1. JLabel

Lớp JLabel được sử dụng để trình bày một chuỗi.

. **Các constructor tạo một Label:**

* **JLabel():** Tạo một Label trống.
* **JLabel(String labeltext):** Tạo một Label với nội dung được cho.
* **JLabel(String labeltext, int alignment):** Tạo một Label với một chế độ canh lề (alignment): JLabel.LEFT, JLabel.RIGHT, JLabel.CENTER

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

* **setFont(Font f):** Thay đổi phông chữ của Label
* **setText(String s):** Thiết lập nhãn cho Label
* **getText():** Lấy nội dung hiện tại của nhãn

1. JTextField

TextField là vùng chứa một dòng văn bản, trong đó văn bản có thể được hiển thị hay được nhập vào bởi người dùng.

**Các constructor tạo một TextField:**

* **JTextField():** Tạo một textfield mới.
* **JTextField(int columns):** Tạo một textfield mới với số cột được cho trước.
* **JTextField(String s):** Tạo một textfield mới với chuỗi văn bản được cho trước.
* **JTextField(String s, int columns**): Tạo một textfield mới với nội dung và số cột được cho trước.

**Các phương thức thường được sử dụng của đối tượng TextField:**

* **setText(String s):** Gán nội dung cho TextField.
* **getText():** Lấy nội dung của TextField.
* **setEditable(boolean):** Xác định TextField có soạn thảo được hay không. Nó chỉ được soạn thảo khi giá trị tham số truyền vào là True.
* **isEditable():** Xác định xem có đang trong chế độ soạn thảo hay không. Giá trị trả về kiểu Boolean.

1. JTextArea

**TextArea** được sử dụng khi văn bản nhập vào có từ hai hay nhiều dòng. TextArea có một scrollbar.

**Các constructor tạo TextArea:**

* **JTextArea():** Tạo một TextArea mới.
* **JTextArea(int rows, int cols):** Tạo một TextArea mới với số lượng cột và dòng được cho trước.
* **JTextArea(String text):** Tạo một TextArea mới vớớcnoij dung được cho trước.
* **JTextArea(String text, int rows, int cols):** Tạo một TextArea mới với dung, số dòng và số cột được cho trước.

**Các phương thức thường được sử dụng của TextArea:**

* **setText(String):** Gán nội dung cho TextArea.
* **getText():** Trả về nội dung của TextArea.
* **setEditable(boolean):** Xác định xem TextAreacó thể được soạn thảo hay không. TextArea có thể được soạn thảo khi giá trị này là True.
* **isEditable():** Xác định xem TextArea có đang trong chế độ soạn thảo được không. Trả về giá trị là kiểu Boolean.
* **insert(String, int):** Chèn chuỗi vào vị trí được cho trước.
* **replaceRange(String s, int x, int y):** Thay thế văn bản nằm giữa vị trí x, y cho trước bởi s

1. JButton

Button hay còn gọi là nút lệnh là một phần không thể thiếu của bất kỳ giao diện nào.

Sử dụng button là cách dễ nhất để nhận các tác động của người dùng.

**Các constructor tạo button:**

* **JButton():** Tạo một Button mới
* **JButton(String text):** Thêm nhãn cho Button
* **JButton(Icon icon):** Thêm icon mới cho Button
* **JButton(String text, Icon icon):** Thêm nhãn và icon cho Button

Sử dụng setLabel() và getLabel() để thiết lập và lấy giá trị nhãn của button.

1. JSlider

Tạo một thanh kéo có thể thay đổi giá trị bất kỳ

* **JSlider():** Tạo một Slider mới
* **JSlider(int minValue, int maxValue, int valueDefault):** Thêm miền giá trị và giá trị mặc định
* **JSlider(Shape, int minValue, int maxValue, int valueDefault):** Thêm miền giá trị và giá trị mặc định và cách thay đổi giá trị theo hương ngang hay dọc

1. JCheckbox

Checkbox được sử dụng khi người dùng tiến hành chọn một hay nhiều tùy chọn.

Người dùng click trên các checkbox để chọn hay bỏ chọn chúng.

**Các constructor tạo checkbox:**

* **Checkbox():** Tạo một checkbox trống.
* **Checkbox(String text, boolean selected):** Tạo một checkbox với nhãn và giá trị được chọn hay không.
* **Checkbox(Icon icon, boolean selected)**: Tạo một checkbox với icon và giá trị được chọn hay không
* **Checkbox(String text, Icon icon):** Tạo một checkbox với nhãn và icon
* **Checkbox(String text, Icon icon, boolean selected):** Tạo một checkbox với nhãn, icon
* và giá trị được chọn hay không

1. JRadionButton

Tương tự như JCheckbox

1. ButtonGroup

Nhóm các radion button để có thể chỉ chọn 1 trong các radion button đó

1. JComboBox

JComboBox cho phép tạo danh sách chứa nhiều mục. Người dùng chỉ có thể click vào một mục từ danh sách.

**Các hàm Contructor tạo ComboBox:**

* **JComboBox():** Tạo Combobox rỗng
* **JComboBox(String[] stringArray):** Tạo Combobox với danh sách stringArray.

**Thêm một item bằng cách sử dụng phương thức addItem như sau:**

* addItem(“Chuỗi”);

**Các phương thức thường được sử dụng của JComboBox:**

* **setBackground(Color):** Xác định màu nền
* **setForeground(Color):** Xác định màu chữ
* **int getSelectedIndex():** Lấy giá trị chỉ số mục được chọn, trả về số nguyên.
* **Object getSelectedItem():** Lấy giá trị dữ liệu được chọn, trả về kiểu Object

1. Layout Manager

Layout manager điều khiển cách trình bày vật lý của các phần tử GUI như là button, textbox, label, …

Các kiểu Layout khác nhau:

* Flow layout
* Border layout
* Card layout
* Grid layout
* GridBag Layout

1. FlowLayout

Các thành phần được sắp xếp theo hàng, từ trái sang phải

Có thể canh lề bằng cách sử dụng FlowLayout.(RIGHT, CENTER, LEFT) 🡪 Mặc định là LEFT

1. BorderLayout

Mặc định cho Window, Frame và Dialog.

Sắp xếp tối đa 5 thành phần trong một container:

* **NORTH** – Đặt ở đỉnh của container.
* **EAST** – Đặt phía bên phải của container.
* **SOUTH** – Đặt ở phía dưới của container.
* **WEST** – Đặt phía bên trái của container.
* **CENTER** – Đặt ở giữa của container.

**Contructor tạo BorderLayout:**

* BorderLayout() //Khởi tạo Border Layout
* BorderLayout(int, int) //Khởi tạo Border Layout và gán khoảng cách giữa các phần trong container

1. CardLayout

CardLayout có thể lưu trữ một ngăn xếp các giao diện.

Ta có thể chuyển qua lại các giao diện:

1. GridLayout

Chia container thành các ô lưới đều nhau dạng ma trận.

Các thành phần được đặt trong các ô giao theo dòng và cột theo thứ tự từ trái sang phải và từ trên xuống dưới..

**GridLayout(int rows, int columns):** rows là số dòng và columns là số cột.

**GridLayout(int rows, int columns, int, int):** rows là số dòng và columns là số cột và gán khoảng cách giữa các dòng và các cột

1. Không dùng Layout

Sắp xếp các thành phần trên Container mà không cần dùng Layout:

* setLayout(null);

**Xác định phạm vi cần hiển thị thành phần:**

setBound(int x, int y, int width, int height);

* Tọa độ góc top-left x, y
* Kích thước đối tượng width, height

1. Xử lý sự kiện

**Xử lý sự kiện thông qua các giao diện lắng nghe sự kiện (Event Listener):**

* Mỗi giao diện lắng nghe một loại sự kiện.
* Mỗi giao diện cung cấp các phương thức xử lý những sự kiện này.
* Lớp thi hành giao diện cần phải định nghĩa những phương thức này.

**ActionEvent:** Phát sinh khi một button được nhấn, một item trong danh sách chọn lựa được nhấn đúp (double-click) hay một menu được chọn.

**AdjustmentEvent:** Phát sinh khi một thanh scrollbar được sử dụng.

**ComponentEvent:** Phát sinh khi một thành phần được thay đổi kích thước, được di chuyển, bị ẩn hay làm cho hoạt động được.

**FocusEvent:** Phát sinh khi một thành phần nhận focus từ bàn phím.

**ItemEvent:** Phát sinh khi một mục menu được chọn hay bỏ chọn; hay khi một checkbox hay một item trong danh sách được click.

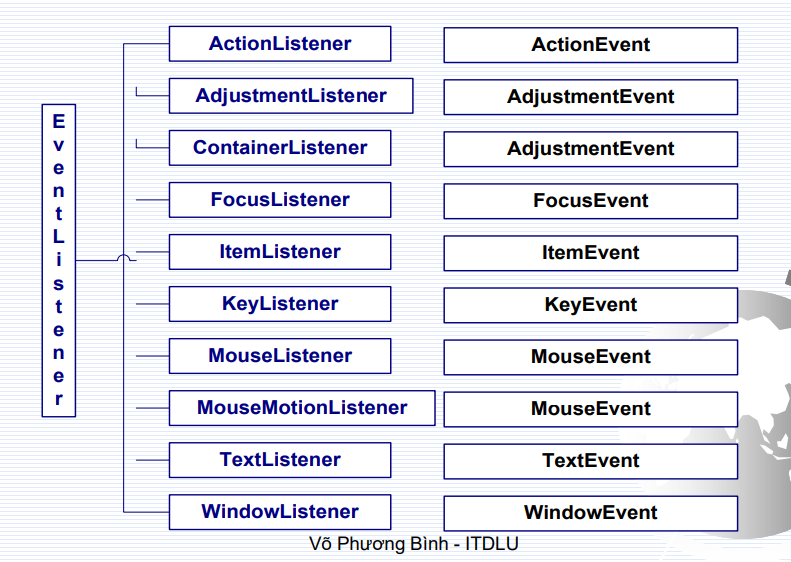
**WindowEvent:** Phát sinh khi một cửa sổ được kích hoạt, được đóng, được mở hay thoát.

**TextEvent:** Phát sinh khi giá trị trong thành phần textfield hay textarea bị thay đổi.

**MouseEvent:** Phát sinh khi chuột di chuyển, được click, được kéo hay thả ra.

**KeyEvent:** Phát sinh khi bàn phím ấn, nhả.

**Các giao diện Event Listener và Event tương ứng**



1. Xử lý sự kiện Button

**Cài đặt giao diện Event Listener thích hợp.**

implement ActionListener

**Đăng ký một thành phần với listener, ta có thể sử dụng:**

addActionListener(this);

**Xử lý sự kiện:**

public void actionPerformed(ActionEvent ae) { //code xử lý }

1. Xử lý sự kiện Slider

**Cài đặt giao diện Event Listener thích hợp.**

implement ChangeListener

**Đăng ký một thành phần với listener, ta có thể sử dụng:**

addChangeListener(this)

**Xử lý sự kiện**

public void stateChanged(ChangeEvent evt) {//code xử lý}

1. Xử lý sự kiện MouseListener

Cần định nghĩa các phương thức sau:

* public void mousePressed(MouseEvent evt); //Xử lý khi click chuột
* public void mouseReleased(MouseEvent evt); //Xử lý khi thả chuột ra sau khi click
* public void mouseClicked(MouseEvent evt); //Xử lý khi người dùng nhấp giữ chuột
* public void mouseEntered(MouseEvent evt); //Xử lý sau khi di chuyển con trỏ tới vùng được lắng nghe
* public void mouseExited(MouseEvent evt); //Xử lý sau khi di chuyển con trỏ ra khỏi vùng được lắng nghe

1. Xử lý sự kiện MouseMotionListener

* public abstract void mouseDragged(MouseEvent e);  //Xử lý khi click chuột và di chuyển trên vùng được lắng nghe
* public abstract void mouseMoved(MouseEvent e);  //Xử lý khi di chuyển chuột vào vùng lắng nghe và không nhấn một nút nào

1. Giao diện ItemListener

Giao diện ItemListener xử lý sự kiện ItemEvent trên các thành phần như ComboBox, CheckBox, List.

**Xử lý sự kiện**

public void itemStateChanged(ItemEvent evt) {//code xử lý}

1. Menu

Có 2 loại:

* Pull – down: Menu có sẵn ở trên và chỉ click chuột trái vào thực hiện thao tác
* Pop – up:: Menu không xuất hiện sẵn, chỉ khi người dùng click chuột phải mới hiện lên để thực hiện thao tác

Chỉ có duy nhất 1 Menubar đặt trong 1 Frame.

**Các thành phần của Menu:**

* **Menubar:** là một thanh nằm ngang được đặt tại đỉnh của frame.
* **Menu:** liệt kê các mục chọn khác nhau (File, Edit, …)
* **MenuItem:** các mục chọn con (Open, Close, …)

**Xử lý MenuItem:**

* .getActionCommand().equals(“Tên Menu Item”)
* .getSource == Tên biến MenuItem;

**Đặt phím tắt cho các JMenu bằng cách dùng phương thức setMnemonic**

Ví dụ: menuFile.setMnemonic('F');

**Đặt phím tắt cho các JMenuItem bằng phương thức setAccelerator**

Ví dụ: itemNew.setAccelerator( KeyStroke.getKeyStroke("ctrl N") );

1. Đồ họa trong Java (Graphics)

Java cung cấp gói AWT cho phép ta vẽ các hình đồ hoạ trong một Panel.

Để vẽ trong một panel:

* Tạo lớp kế thừa lớp JPanel
* Định nghĩa lại phương thước paintComponent để chỉ cho panel vẽ các hình cần vẽ.

**Lớp Graphics bao gồm tập hợp rất nhiều phương thức, được sử dụng để vẽ trong các hình sau:**

* Text
* Oval
* Rectangle
* Square
* Circle
* Lines

1. Vẽ các ký tự

**drawChars (char array[], int offset, int length, int xCoor, int yCoor);**

* Mảng các ký tự.
* Vị trí bắt đầu, nơi các ký tự được vẽ.
* Số các ký tự cần được vẽ.
* Toạ độ xCoor, yCoor

**drawBytes (byte array[], int offset, int length, int xCoor, int yCoor);**

1. Vẽ đoạn thẳng

drawLine(x1, y1, x2, y2); //Tọa độ điểm đầu và điểm cuối

1. Vẽ hình oval

drawOval(x, y, w, h); Tọa độ điểm bắt đầu, chiều rộng và chiều cao của oval

fillOval(x, y, w, h);

1. Vẽ hình chữ nhật

drawRect(x, y, w, h); //Tọa độ điểm bắt đầu, chiều rộng và chiều cao của hình chữ nhật

fillRect(x, y, w, h);

drawRoundRect(x, y, w, h, aw,ah); // Tọa độ điểm bắt đầu, chiều rộng và chiều cao của hình chữ nhật, chiều rộng và chiều dài bo góc

fillRoundRect(x, y, w, h, aw,ah);

1. Điều khiển màu

Java sử dụng mô hình màu RGB – **color (int red, int green, int blue);**

1. Vẽ đồng hồ

Kế thừa lớp JPanel và sử dụng phương thức paintComponent() để vẽ ra hình tròn sử dụng hàm drawOval(x, y, width, height)

Lấy thời gian hiện tại bằng cách: LocalTime time = LocalTime.now();

**Kim giây: 1 giây đi được góc C = second \* (2π/60)**

* xEnd = xCenter + handLength x sin(C)
* yEnd = yCenter - handLength x cos(C)

**Kim phút: 1 phút đi được góc C = (minute + second/60) \* (2π/60)**

* xEnd = xCenter + handLength x sin(C)
* yEnd = yCenter - handLength x cos(C)

**Kim giờ: 1 phút đi được góc C = (hour + minute/60 + second/3600) \* (2π/12)**

* xEnd = xCenter + handLength x sin(C)
* yEnd = yCenter - handLength x cos(C)

Kiến Thức Chương 8

Java hỗ trợ truyền tín hiệu TCP và UDP đảm bào nối không mất thông tin và kết nối mất thông tin

**Các class hỗ trợ cho chương trình Networking:**

* Socket - a TCP connection socket
* ServerSocket - a TCP server socket
* DatagramPacket - a UDP packet
* DatagramSocket - a UDP socket
* InetAddress - an internet address

**Thường dùng:**

* **ServerSocket:** Tạo server
* **Socket:** Tạo client
* **DataInputStream:** Nhận dữ liệu từ nơi gửi dến nơi nhận
* **DataOutputStream:** Chuyển dữ liệu từ nơi gửi đến nơi nhận
* **readUTF();** //đọc dữ liệu lưu vào biến chuỗi
* **writeUTF();** //ghi dữ liệu từ biến chuỗi vào dữ liệu chuyển đi

1. Các bước khởi tạo 1 server đơn giản

* Tạo một đối tượng ServerSocket
* Lắng nghe client
* Nhận tín hiệu đầu vào và đầu ra
* Giao tiếp với client
* Đóng kết nối

1. Server

**Giả sử:** ServerSocket server = null;

Socket cilent = null;

Luôn đặt trong try {} catch {}

**Tạo một server lắng nghe ở cổng 8000**: server = new ServerSocket(8000);

**Lắng nghe client:** client = server.accept();

**Ngắt kết nối client:** client.close();

1. Client

**Giả sử:** ServerSocket server = null;

Socket cilent = null;

**Tạo một client:** client = new Socket(InetAddress.getLocalHost(), 8000); //Được tạo từ localHost kết nối với cổng lắng nghe của server