Hướng đối tượng

## Khái niệm về lập trình hướng đối tượng trong java

**Lập trình hướng đối tượng (OOP-Object-Oriented Programming)** là một phương pháp hay mô hình giúp tăng năng suất, đơn giản hóa việc bảo trì, dễ dàng mở rộng trong thiết kế phần mềm bởi việc cung cấp một vài khái niệm như:

* Đối tượng (Object)
* Lớp (Class)
* Kế thừa (Inheritance)
* Đa hình (Polymorphism)
* Trừu tượng (Abstraction)
* Đóng gói (Encapsulation)

**Đối tượng (Object)**

Tất cả những thực thể có trạng thái và hành vi được biết đến như là một đối tượng. Ví dụ: bàn, ghế, bút chì, xe dạp, ô tô...

**Lớp (Class):** Tập hợp các đối tượng được gọi là lớp.

**Lập trình hướng đối tượng có 4 tính chất:**

1. **Kế thừa (Inheritance)**

Khi một đối tượng được truyền lại tất cả các thuộc tính và phương thức của đối tượng cha được gọi là kế thừa. Kế thừa giúp tái sử dụng lại mã nguồn. Nó được sử dụng cho đa hình lúc runtime.

**Kế thừa trong java** là sự liên quan giữa hai class với nhau, trong đó có class cha (superclass) và class con (subclass). Khi kế thừa class con được hưởng tất cả các phương thức và thuộc tính của class cha. Tuy nhiên, nó chỉ được truy cập các thành viên public và protected của class cha. Nó không được phép truy cập đến thành viên private của class cha.

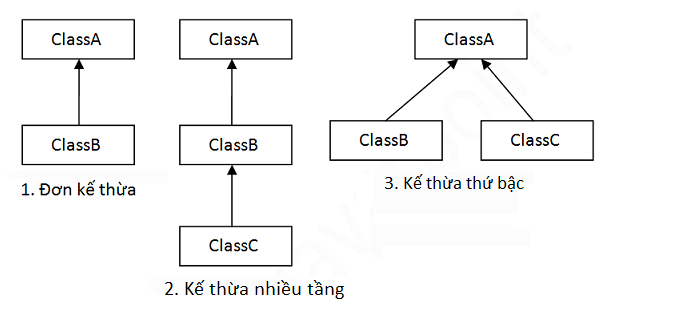
Ví dụ:

**class** Subclass-name **extends** Superclass-name {

   //methods and fields

}

Các kiểu kế thừa trong java



1. **Đa hình (Polymorphism)**

Khi một nhiệm vụ được thực hiện bởi nhiều cách khác nhau, tính chất này được gọi là đa hình. Ví dụ có nhiều các để thuyết phục các khách hàng khác nhau, để vẽ một cái gì đó như hình tròn, hình chữ nhật, ...

Trong java, để áp dụng tính đa hình chúng ta sử dụng phương thức orverloading hoặc overriding.

**Đa hình trong java (Polymorphism)** là một khái niệm mà chúng ta có thể thực hiện một hành động bằng nhiều cách khác nhau. Polymorphism được cấu tạo từ 2 từ Hy Lạp: poly và morphs. Trong đó "poly" có nghĩa là nhiều và "morphs" có nghĩa là hình thể. Vậy polymorphism có nghĩa là nhiều hình thể.

1. **Trừu tượng (Abstraction)**

Trừu tượng là sự ẩn đi những chi tiết bên trong và hiển thị ra các chức năng, tính chất này gọi là trừu tượng. Ví dụ: khi gọi điện thoại chúng ta không biết xử lý nội bộ thế nào, khi đi xe máy cũng vậy, mà chúng ta chỉ biết đến các chức năng thông qua giao tiếp bên ngoài.

Trong java, chúng ta áp dụng tính chất trừu tượng bằng cách sử dụng abstract class và interface.

Một lớp được khai báo với từ khóa abstract là lớp abstract trong Java. Lớp abstract có nghĩa là lớp trừu tượng, nó có thể có các phương thức abstract hoặc non-abtract.

**Phương thức trừu tượng trong Java**

Một phương thức được khai báo là abstract và không có trình triển khai thì đó là phương thức trừu tượng.

Nếu bạn muốn một lớp chứa một phương thức cụ thể nhưng bạn muốn triển khai thực sự phương thức đó để được quyết định bởi các lớp con, thì bạn có thể khai báo phương thức đó trong lớp cha ở dạng abstract.

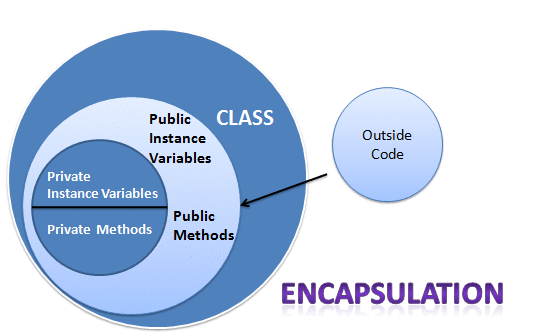
Từ khóa abstract được sử dụng để khai báo một phương thức dạng abstract. Một phương thức abstract không có thân phương thức.

1. **Đóng gói (Encapsulation)**

Việc ràng buộc giữa code và data với nhau tạo thành một khối duy nhất được biết đến là đóng gói. Ví dụ: viên thuốc con nhộng được đóng gói với nhiều loại thuốc bên trong.

Một class trong java là một ví dụ về đóng gói. Java bean là một lớp được đóng gói hoàn toàn vì tất cả các dữ liệu thành viên là private.

**Tính đóng gói trong java** là kỹ thuật ẩn giấu thông tin không liên quan và hiện thị ra thông liên quan. Mục đích chính của đóng gói trong java là giảm thiểu mức độ phức tạp phát triển phần mềm.



**Lợi ích của đóng gói trong java**

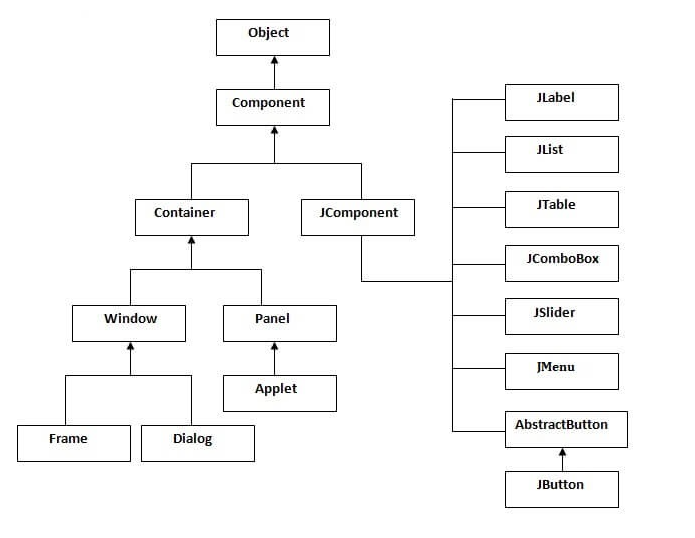
Bạn có thể tạo lớp **read-only** hoặc **write-only** bằng việc cài đặt phương thức setter hoặc getter.

Bạn có thể kiểm soát đối với dữ liệu. Giả sử bạn muốn đặt giá trị của id chỉ lớn hơn 100 bạn có thể viết logic bên trong lớp setter.

## Java Swing

Phân cấp các lớp Java Swing

Hệ thống phân cấp của API java swing được đưa ra dưới đây.



**Các phương thức thường dùng của lớp Component**

Các phương thức của lớp Component được sử dụng rộng rãi trong java swing được đưa ra dưới đây.

|  |  |
| --- | --- |
| **Phương thức** | **Mô tả** |
| public void add(Component c) | thêm một thành phần vào thành phần khác. |
| public void setSize(int width, int height) | thiết lập kích thước của thành phần. |
| public void setLayout(LayoutManager m) | thiết lập trình quản lý bố cục (layout) cho thành phần. |
| public void setVisible(boolean b) | thiết lập khả năng hiển thị của thành phần. Nó theo mặc định là false (ẩn) |
|  |  |

1. Lớp JFrame trong Java Swing

Container là thành phần chủ chốt trong các thành phần của SWING GUI. Một Container cung cấp một không gian, là nơi đặt một thành phần. Một Container trong AWT chính là một Component và nó có thêm khả năng để thêm các thành phần khác vào chính nó.

**Lớp này kế thừa các phương thức từ các lớp sau:**

* java.awt.Frame
* java.awt.Window
* java.awt.Container
* java.awt.Component
* java.lang.Object

**Lớp JFrame này có các constructor sau:**

1. **JFrame()**: Xây dựng một Frame mới, ban đầu là không nhìn thấy (invisible).
2. **JFrame(GraphicsConfiguration gc)**: Tạo một Frame trong GraphicsConfiguration đã cho của một thiết bị màn hình và một title trống.
3. **JFrame(String title)**: Tạo một Frame mới, ban đầu là không nhìn thấy (invisible) với title đã cho.
4. **JFrame(String title, GraphicsConfiguration gc)**: Tạo một Frame với title đã cho và GraphicsConfiguration đã cho của một thiết bị màn hình.
5. **Lớp JPanel trong Java Swing**

JPanel trong Java Swing được sử dụng để tạo ra các container nơi chứa các phần tử khác. Container là thành phần chủ chốt trong các thành phần của SWING GUI. Một Container cung cấp một không gian, là nơi đặt một thành phần. Một Container trong AWT chính là một Component và nó có thêm khả năng để thêm các thành phần khác vào chính nó.

**Lớp này kế thừa các phương thức từ các lớp sau:**

* javax.swing.JComponent
* java.awt.Container
* java.awt.Component
* java.lang.Object

**Lớp này bao gồm các constructor sau:**

* **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 đệm đã xác định.
* **JPanel(LayoutManager layout):** Tạo một JPanel mới với Layout Manager đã cho
* **JPanel(LayoutManager layout, boolean isDoubleBuffered):** Tạo một JPanel mới với Layout Manager đã cho và trình đệm đã xác định.

**Các phương thức của lớp Jpanel:**

* **AccessibleContext getAccessibleContext():** Lấy AccessibleContext được liên kết với JPanel này.
* **PanelUI getUI():** Trả về đối tượng L&F mà truyền thành phần này
* **String getUIClassID():** Trả về một chuỗi xác định tên của lớp L&F mà truyền thành phần này
* **protected String paramString():** Trả về một biểu diễn chuỗi của JPanel
* **void setUI(PanelUI ui):** Thiết lập đối tượng L&F mà truyền thành phần
* **void updateUI():**Phục hồi thuộc tính UI về một giá trị Look và Feel hiện tại.

1. Xử lý sự kiện trong Swing

**Event** là một sự thay đổi trong trạng thái của đối tượng, chẳng hạn như sự kiện mô tả sự thay đổi trong trạng thái của source. Các sự kiện được tạo ra là do tương tác của người dùng với các thành phần UI. Ví dụ như việc nhấn vào một nút button, di chuyển chuột, nhập ký tự thông qua bàn phím, … Các sự kiện có thể được phân chia thành hai loại sau:

* Foreground Event: Những sự kiện này yêu cầu tương tác trực tiếp của người dùng. Chúng được tạo ra do tương tác của người dùng với các thành phần đồ họa trong Graphical User Interface. Ví dụ như nhấn vào một nút, di chuyển chuật, chọn một item từ list, …
* Background Event: Các sự kiện này yêu cầu tương tác của người dùng cuối cùng, ví dụ như tín hiệu ngắt hệ điều hành, hardware hoặc software failure

**Xử lý sự kiện (Event Handling)** là một kỹ thuật kiểm soát sự kiện và quyết định những gì cần thực hiện nếu một sự kiện xảy ra. Kỹ thuật này có code, mà được biết như là Event Handler, được thực thi khi một sự kiện xảy ra. Java sử dụng Delegation Event Model để xử lý các sự kiện. Model này định nghĩa kỹ thuật chuẩn để tạo và xử lý các sự kiện. Model này bao gồm hai thành phần quan trọng sau:

* Source: đây là một đối tượng mà trên đó sự kiện xuất hiện. Source chịu trách nhiệm cung cấp thông tin về sự kiện đã xảy ra tới bộ xử lý Handler của nó.
* Listener: Còn được biết như là Event Handler. Listener chịu trách nhiệm tạo phản hồi tới một sự kiện. Theo quan điểm của Java, Listener cũng là một đối tượng. Listener đợi tới khi nó nhận một sự kiện. Khi một sự kiện đã được nhận, Listener xử lý sự kiện đó và sau đó trả về kết quả.

1. **Lớp EventObject trong Java Swing**

Đây là lớp gốc (root class) từ đó tất cả đối tượng về trạng thái sự kiện sẽ được kế thừa. Tất cả sự kiện được xây dựng với một tham chiếu tới đối tượng đó, là source. Lớp này được định nghĩa trong java.util package. Cú pháp khai báo của lớp java.util.EventObject như sau:

public **class** EventObject

   extends Object

      implements Serializable

Lớp EventObject có một trường là protected Object source. Đây là đối tượng mà trên đó sự kiện được xảy ra. Lớp EventObject có một constructor là **EventObject(Object source)** dùng để xây dụng một nguyên mẫu sự kiện.

**Các phương thức của lớp EventObject:**

* Object getSource(): Đối tượng mà trên đó sự kiện xảy ra.
* String toString(): Trả về một biểu diễn chuỗi của EventObject này.

1. **Lớp AWTEvent**

Đây là lớp sự kiện gốc (root class) cho tất cả sự kiện AWTEvent. Lớp này và các lớp con của nó thay thế lớp ban đầu java.awt.Event. Lớp này được định nghĩa trong java.awt package. Lớp AWTEvent có phương thức getID() được sử dụng để xác định kiểu của sự kiện. Cú pháp khai báo của lớp java.awt.AWTEvent là:

public **class** AWTEvent

   extends EventObject

1. **Lớp ActionEvent**

Lớp ActionEvent được định nghĩa trong java.awt.event package. ActionEvent được tạo ra khi một nút được nhấn hoặc một item của một danh sách được nhấn đúp. Cú pháp khai báo cho lớp ActionEvent như sau:

public **class** ActionEvent

   extends AWTEvent

1. **Lớp InputEvent**

Lớp InputEvent là lớp sự kiện gốc (root class) cho tất cả sự kiện lien quan tới đầu vào (cấp độ thành phần). Các sự kiện liên quan tới đầu vào (input event) được phân phối bởi Listener trước khi chúng được xử lý một cách thông thường bởi source, nơi chúng sinh ra. Điều này cho phép các Listener và các lớp thành phần con có thể "consume" sự kiện để mà source sẽ không xử lý chúng theo phương thức mặc định. Cú pháp khai báo của lớp java.awt.event.InputEvent là:

public abstract **class** InputEvent

   extends ComponentEvent

1. **Lớp KeyEvent**

Sự kiện liên quan tới phím (Key Event) được tạo ra khi bạn nhập ký tự. Có ba kiểu key event, mà được biểu diễn bởi các hằng nguyên, chúng là:

* KEY\_PRESSED
* KEY\_RELASED
* KEY\_TYPED

Cú pháp khai báo của lớp java.awt.evetn.KeyEvent như sau:

public **class** KeyEvent

   extends InputEvent

1. **Lớp MouseEvent**

Sự kiện này chỉ một hoạt động liên quan tới chuột xảy ra trong một thành phần. Sự kiện tầm thấp này được tạo bởi một đối tượng Component cho các sự kiện liên quan tới chuột và di chuyển chuột, chẳng hạn như một nút chuột được nhấn, được nhả ra, được click (nhấn và nhả ra), di chuyển chuột, kéo chuột, …

Cú pháp khai báo cho lớp java.awt.event.MouseEvent:

public **class** MouseEvent

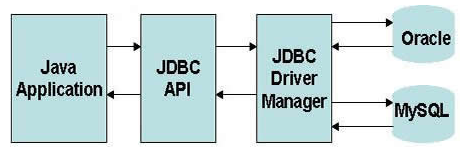
   extends InputEvent

## Lý thuyết JDBC với MySQL

**Khái niệm: JDBC** (Java Database Connectivity) là một API tiêu chuẩn dùng để tương tác với các loại cơ sở dữ liệu quan hệ. **JDBC** có một tập hợp các class và các Interface dùng cho ứng dụng **Java** có thể nói chuyện với các cơ sở dữ liệu.

Các thành phần của JDBC Api về cơ bản bao gồm:

1. **DriverManager:**
   * Là một class, nó dùng để quản lý danh sách các Driver (database drivers).
2. ***Driver*:**
   * Là một Interface, nó dùng để liên kết các liên lạc với cơ sở dữ liệu, điều khiển các liên lạc với database. Một khi Driver được tải lên, lập trình viên không cần phải gọi nó một cách cụ thể.
3. ***Connection*:**
   * Là một Interface với tất cả các method cho việc liên lạc với database. Nó mô tả nội dung liên lạc. tất cả các thông tin liên lạc với cơ sở dữ liệu là thông qua chỉ có đối tượng *Connection*.
4. ***Statement*:**
   * Là một Interface, gói gọn một câu lệnh SQL gửi tới cơ sở dữ liệu được phân tích, tổng hợp, lập kế hoạch và thực hiện.
5. ***ResultSet*:**
   * *ResultSet* đại diện cho tập hợp các bản ghi lấy do thực hiện truy vấn.



Các class sử dụng để kết nối

Class DBConnection

Các lệnh thao tác thêm, sửa xoá

Danh sách chức năng

Tài liệu tham khảo

<https://viettuts.vn/java/tong-quan-java-oops>

<https://openplanning.net/10167/java-jdbc>