JAVA CORE

# 1 Khái niệm về OOP

## Đóng gói

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.

Đóng gói cũng được sử dụng để bảo vệ trạng thái bên trong của một đối tượng. Bởi việc ẩn giấu các biến biểu diễn trạng thái của đối tượng. Việc chỉnh sửa đối tượng được thực hiện, xác nhận thông qua các phương thức. Hơn nữa, việc ẩn giấu các biến thì các lớp sẽ không chia sẻ thông tin với nhau được.

Ta tạo lớp read-only hoặc write-only bằng việc cài đặt phương thức setter hoặc getter.

Bảo vệ tính toàn vẹn của phần mềm thông qua cú pháp private và geter seter

## Kế thừa

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.

Tư tưởng của kế thừa trong java là có thể tạo ra một class mới được xây dựng trên các lớp đang tồn tại. Khi kế thừa từ một lớp đang tồn tại bạn có sử dụng lại các phương thức và thuộc tính của lớp cha, đồng thời có thể khai báo thêm các phương thức và thuộc tính khác.

Cái này thể hiện lại sự tái sử dụng code làm cho code không bị trùng lặp

## Đa hình

**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.

Có hai kiểu của đa hình trong java, đó là đa hình lúc biên dịch **(compile)** và đa hình lúc thực thi **(runtime).** Chúng ta có thể thực hiện đa hình trong java bằng cách nạp chồng phương thức và ghi đè phương thức.

Nếu bạn nạp chồng phương thức static trong java, đó là một ví dụ về đa hình lúc biên dịch. Đa hình lúc runtime là quá trình gọi phương thức đã được ghi đè trong thời gian thực thi chương trình. Trong quá trình này, một phương thức được ghi đè được gọi thông qua biến tham chiếu của một lớp cha.

**Upcasting.**

Khi biến tham chiếu của lớp cha tham chiếu tới đối tượng của lớp con, thì đó là Upcasting.

|  |  |
| --- | --- |
| 1  2 | class A{}  class B extends A{} |

|  |  |
| --- | --- |
| 1 | A a=new B();//upcasting |

## Trừu tượng

### 1.4.1 Khái niệm

* Tính trừu tượng (abstraction) là một trong bốn tính chất cơ bản của lập trình hướng đối tượng trong Java.
* Tính trừu tượng là một tiến trình ẩn các chi tiết trình triển khai và chỉ hiển thị tính năng tới người dùng. Tính trừu tượng cho phép bạn loại bỏ tính chất phức tạp của đối tượng bằng cách chỉ đưa ra các thuộc tính và phương thức cần thiết của đối tượng trong lập trình.
* Tính trừu tượng giúp bạn tập trung vào những cốt lõi cần thiết của đối tượng thay vì quan tâm đến cách nó thực hiện.
* Trong Java, chúng là sử dụng abstract class và abstract interface để có tính trừu tượng.

1.4.1 Abstract

* Một lớp được khai báo với từ khóa abstract là lớp trừu tượng (abstract class).
* Lớp trừu tượng có thể có các phương thức abstract hoặc non-abtract.
* Lớp trừu tượng có thể khai báo 0, 1 hoặc nhiều method trừu tượng bên trong.
* Không thể khởi tạo 1 đối tượng trực tiếp từ một class trừu tượng.
* Một lớp kế thừa từ lớp trừu tượng (subclass – lớp con) không cần phải implement non-abstract methods, nhưng những method nào có abstract thì bắt buộc phải override. Trừ khi subclass cũng là abstract.

### 1.4.2 Phương thức khai báo abstract

* Một phương thức được khai báo là abstract và không có trình triển khai thì đó là phương thức trừu tượng (abstract method).
* 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. Phương thức abstract sẽ không có định nghĩa, được theo sau bởi dấu chấm phẩy, không có dấu ngoặc nhọn theo sau

### 1.4.3 Interface

* Các phương thức trong interface đều là các phương thức trừu tượng.
* Interface là một kỹ thuật để thu được tính trừu tượng hoàn toàn và đa kế thừa trong Java.
* Interface luôn luôn có modifier là: public interface, cho dù bạn có khai báo rõ hay không.
* Nếu có các trường (field) thì chúng đều là: public static final, cho dù bạn có khai báo rõ hay không.
* Các method của nó đều là method trừu tượng, nghĩa là không có thân hàm, và đều có modifier là: public abstract, cho dù bạn có khai báo hay không.
* Interface không có hàm khởi tạo (constructor).
* Một interface không phải là một lớp. Viết một interface giống như viết một lớp, nhưng chúng có 2 định nghĩa khác nhau. Một lớp mô tả các thuộc tính và hành vi của một đối tượng. Một interface chứa các hành vi mà một class triển khai.
* Trừ khi một lớp triển khai interface là lớp trừu tượng abstract, còn lại tất cả các phương thức của interface cần được định nghĩa trong class.
* Hỗ trợ đa kế thừa

### 1.4.4 Interface vs class

* Bạn không thể khởi tạo một interface.
* Một interface không chứa bất cứ hàm Contructor nào.
* Tất cả các phương thức của interface đều là abstract.
* Một interface không thể chứa một trường nào trừ các trường vừa static và final.
* Một interface không thể kế thừa từ lớp, nó được triển khai bởi một lớp.
* Một interface có thể kế thừa từ nhiều interface khác.

## This

Từ khóa this trong java là một biến tham chiếu được sử dụng để tham chiếu tới đối tượng của lớp hiện tại.

* Từ khóa this có thể được dùng để tham chiếu tới biến instance của lớp hiện tại.
* this() có thể được dùng để gọi Constructor của lớp hiện tại.
* Từ khóa this có thể được dùng để gọi phương thức của lớp hiện tại.
* Từ khóa this có thể được truyền như một tham số trong phương thức.
* Từ khóa this có thể được truyền như một tham số trong phương Constructor.
* Từ khóa this có thể được dùng để trả về instance của lớp hiện tại.

## Super

Từ khóa super trong java là một biến tham chiếu được sử dụng trong lớp con để tham chiếu trực tiếp đến đối tượng của lớp cha gần nhất.

Bất cứ khi nào bạn tạo ra instance(thể hiển) của lớp con, một instance của lớp cha được tạo ra ngầm định, nghĩa là được tham chiếu bởi biến super.

* Từ khóa super được sử dụng để tham chiếu trực tiếp đến biến instance của lớp cha gần nhất.
* super() được sử dụng để gọi trực tiếp Constructor của lớp cha.
* Từ khóa super được sử dụng để gọi trực tiếp phương thức của lớp cha.

## Final

Từ khóa final trong Java được sử dụng để hạn chế người dùng

* Biến final: bạn không thể thay đổi giá trị của biến final (nó sẽ là hằng số).
* Phương thức final: bạn không thể ghi đè phương thức final.
* Lớp final: bạn không thể kế thừa lớp final.
* Biến static final trống: Một biến final mà không được khởi tạo tại thời điểm khai báo được gọi là biến final trống.
* **static final : kết hợp cả 2 vừa đc khởi tạo vừa k thể thay đổi giá trị // đặt tiên biến viết hoa hết tất cả các ký tự ấy lên ...................**
* **final biến : hằng số**
* **final method : khoogn thể override**
* **final class : khong thể có sub class // không thế có thằng class con**

## Static

Từ khóa static trong Java được sử dụng chính để quản lý bộ nhớ. Chúng ta có thể áp dụng từ khóa static với các biến, các phương thức, các khối, các lớp lồng nhau(nested class). Từ khóa static thuộc về lớp chứ không thuộc về instance(thể hiện) của lớp.

Trong java, Static có thể là:

* Biến static: Khi bạn khai báo một biến là static, thì biến đó được gọi là biến tĩnh, hay biến static.
* Phương thức static: Khi bạn khai báo một phương thức là static, thì phương thức đó gọi là phương thức static.
* Khối static: Được sử dụng để khởi tạo thành viên dữ liệu static.

Khi bạn khai báo một biến là static, thì biến đó được gọi là biến tĩnh, hay biến static.

Biến static có thể được sử dụng để tham chiếu thuộc tính chung của tất cả đối tượng (mà không là duy nhất cho mỗi đối tượng), ví dụ như tên công ty của nhân viên, tên trường học của các sinh viên, ...

Biến static lấy bộ nhớ chỉ một lần trong Class Area tại thời gian tại lớp đó.

Lợi thế của biến static

Sử dụng biến static giúp chương trình của bạn sử dụng bộ nhớ hiệu quả hơn (tiết kiệm bộ nhớ).

**static method : giống hàm main ưu tiên chạy trước .. runtime compile nó đc khởi tạo nó sẽ sdc ưu tiên**

**static class : khởi tạo thành viên dữ liệu // tùy mục đích sử dụng khởi tạo 1 db cố dịnh không thể thay đổi ///**

**static { }**

## Exception

Excaption xuất ra ngoại lệ : CheckedExx và Runtime

CheckedExx check luôn khi mình đang code vd như thiếu dấu chấm hay hàm sai

Runtime : khi chạy nó Ex ra lỗi cho mình ví dụ như chia hết cho 0 hay là lấy ra danh sách ngoài mảng vd mảng [10] lấy phần tử [12]

## Phân biệt for, while, do-while trong Java?

FOR: biết được số vòng lặp, kiểm soát được.

WHILE: có thể không được thực hiện lần nào.

DO-WHILE: chắc chắn sẽ thực hiện 1 lần.

## Phân biệt SCase – với If else

Swith case nó sẽ đi hết các case rồi mới đưa ra kq

If else thì nó sẽ cái nào đúng nó dừng đưa ra kq vd if đúng thì ra không vào else-if đúng thì ra ko thì mới đến else

## Accessmodifier

* Default( nếu không khai báo sẽ là mặc định ): truy cập bên trong class và trong cùng package.
* Protected: trong class, trong cùng package và ngoài package với các class có quan hệ kế thừa.
* Private: truy cập bên trong class.
* Public: truy cập ở mọi nơi.

## Contructer

* Constructor giống như là 1 phương thức được gọi khi một đối tượng mới được khởi tạo.
* Mỗi lớp có 1 constructor mặc định nếu như không viết constructor 1 cách tường mình.

## Over load ride

* Overloading là khi 1 class tồn tại 1 method cùng tên nhưng khác tham số truyền vào.
* Overriding xuất hiện ở class con trong 2 class có quan hệ kế thừa, phương thức ghi đề phải cùng tên và cùng tham số truyền vào.

## Enum

* Enum là kiểu dữ liệu đặc biệt trong Java được sử dụng để định nghĩa tập hợp các hằng số.
* Không thể khởi tạo bằng từ khóa new vì chỉ có constructor private.

## Collection

Collections trong java là một khuôn khổ cung cấp một kiến trúc để lưu trữ và thao tác tới nhóm các đối tượng. Tất cả các hoạt động mà bạn thực hiện trên một dữ liệu như tìm kiếm, phân loại, chèn, xóa,... có thể được thực hiện bởi Java Collections.

**Set: không chứa 2 giá trị trùng lặp. Sử dụng để biểu diễn các bộ.**

* **TreeSet**
* **HashSet**
* **EnumSet**

**List: 1 collection có thứ tự hay được hiểu là 1 danh sách, có thể có các phần tử trùng lặp.**

* **ArrayList**
* **LinkedList**
* **Vector**
* **Stack:LIFO**

Queue:  là một collection được sử dụng để chứa nhiều phần tử trước khi xử lý. Bên cạnh các thao tác cơ bản của collection, Queue cung cấp các thao tác bổ sung như chèn, lấy ra và kiểm tra. Queue có thể được sử dụng như là FIFO (first-in, first-out - vào trước, ra trước).

* PriorotyQueue
* ArrayDeque

Map:  là một đối tượng ánh xạ mỗi key tương ứng với một giá trị. Map không thể chứa giá trị trùng lặp. Mỗi key có thể ánh xạ đến nhiều nhất một giá trị.

* HashMap
* TreeMap
* EnumMap
* WeakHashMap

## Phân biệt bộ nhớ Heap và Stack

* Bộ nhớ heap sử dụng để lưu các đối tượng được khởi tạo bằng từ khóa new, biến global và static. public
* Bộ nhớ stack sử dụng để lưu các tham số và biến local (biến cụ bộ )của mỗi phương thức khi phương thức được gọi.
* Bộ nhớ stack sẽ ít hơn so với bộ nhớ heap.
* Heap thì chia sẻ giữa các luồng còn stack thì không.

## JVM,JRE,JDK

JVM là 1 máy ảo Java hay được hiểu là trình thông dịch Java. Cung cấp môi trường để code Java có thể thực thi.JVM đảm nhận các nhiệm vụ chính sau

* Tải code
* Kiểm tra code
* Thực thi code
* Cung cấp môi trường runtime

JRE là trình triển khai JVM đảm nhận khởi tạo JVM và đảm bảo các phụ thuộc có sẵn trong chương trình của bạn.

JDK bao gồm JRE và các công cụ để lập trình viên có thể phát triển các chương tình Java.