1. Những thứ mà muốn học một ngôn ngữ lập trình ai cũng bắt buộc phải biết

* Kiểu dữ liệu (nếu có)
* Biến
* Ép kiểu (nếu có)
* Cấu trúc rẽ nhánh
* Mảng, Chuỗi
* Vòng lặp

1. OOP – Hướng đối tượng
   1. Đối tượng (Object)

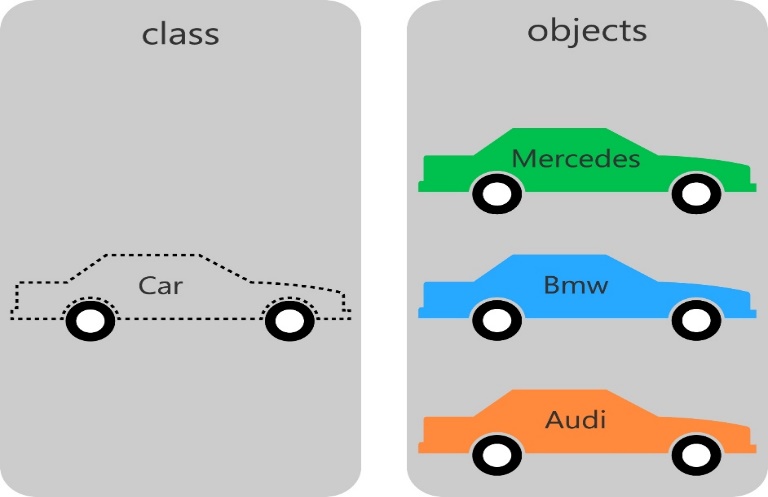
* Đối tượng có thể hiểu là những khái niệm bên ngoài như: Con người, Xe máy, Nhà cửa,…
* Một đối tượng sẽ bao gồm thuộc tính và phương thức.
* Thuộc tính: Là những thông tin riêng của đối tượng. VD: Đối tượng con người có họ tên, giới tính, quê quán,…
* Phương thức: Là những thao tác, hành động mà đối tượng có thể thực hiện được. Phương thức giống hàm trong lập trình hướng cấu trúc nhưng phương thức chỉ được khai báo trong class.

VD: Đối tượng con người có thể chạy, nhảy, ăn, uống,…

* Kiểu dữ liệu nguyên thủy (boolean, char, byte, short,..) khi ta thay đổi giá trị của một biến thì các biến còn lại không thay đổi giá trị.
* Kiểu dữ liệu tham chiếu (array, list, Object,…hoặc class) khi ta thay đổi giá trị của một biến thì các biến còn lại sẽ thay đổi giá trị theo

1.2 Lớp (Class)

* Lớp là người dùng định nghĩa cho của đối tượng, ta sẽ xây dựng lớp để tạo ra những đối tượng khác nhau.



* Nói cách khác class là bản thiết kế để tạo ra các đối tượng. Mà các đối tượng đều phải có những thuộc tính của class
* Cú pháp khai báo class

<Pham\_vi\_truy\_cap> **class** <ten\_lop>

{

<Pham\_vi\_truy\_cap> <Cac\_thanh\_phan\_cua\_lop>

}

Trong đó :

<Pham\_vi\_truy\_cap> : Là các mức độ truy cập: Public, Protected, Private,…

<ten\_lop> : Là tên của class

<Cac\_thanh\_phan\_cua\_lop> : Là biến hoặc phương thức trong lớp

1.2.1 Phương thức

- Là những thao tác, hành động mà đối tượng có thể thực hiện được. Phương thức giống hàm trong lập trình hướng cấu trúc nhưng phương thức chỉ được khai báo trong class.

- Cú pháp:

<Pham\_vi\_truy\_cap> <tu\_khoa> <du\_lieu\_tra\_ve> <name> (Tham\_so){

<Chuong\_trinh>

}

Trong đó:

+ <Pham\_vi\_truy\_cap> : public, protected, private,.

+ <tu\_khoa> : Gồm các từ khóa final, static,…

+ <du\_lieu\_tra\_ve> : Trả về kiểu int, long, double, hoặc tên 1 class nào đó. Nếu không có trả về ta dùng kiểu void. Để trả về dữ liệu ta dùng từ khóa return trong phương thức

+ <name> : Tên của phương thức

+ (Tham\_so) : Là những tham số muốn truyền vào phương thức

+ <Chuong\_trinh> : Các chương trình code trong phương thức

1.2.2 Phương thức khởi tạo

- Khi ta tạo 1 đối tượng Persion a = **new** Persion(); thì phương thức khởi tạo của đối tượng Persion(); sẽ cùng tên với class.

- Ta thường phải chỉnh sửa phương thức khởi tạo cho hợp với yêu cầu.

- Sau khi khởi tạo ta cần phải gán giá trị tương ứng cho các đối tượng a

- Tuy nhiên trong quá trình phát triển phần mềm người lập trình chưa chắc có thể biết hết toàn bộ code bên trong class. Vì vậy khi khởi tạo một đối tượng ta cần cung cấp thông tin về đối tượng đó ngay.

VD:

//Phương thức khởi tạo

**public** Persion(String name, **int** age, **float** height) {

**this**.name = name; //this.name là để gọi đến biến name của class dùng để phân biệt biến name của phương thức

**this**.age = age;

**this**.height = height;

}

//Khai báo đối tượng

Persion a = **new** Persion("Quang",26,1.68f);

* 1. Các phạm vi truy cập trong lập trình hướng đối tượng
* Có 4 loại phạm vi truy cập: Private, (Default), Protected, Public
  + 1. Private
* Chỉ cho phép truy cập nội bộ trong 1 class
  + 1. Default
* Đây là phạm vi truy cập mặc định khi ta không đặt mức truy cập nào cho thuộc tính. Ở dạng mặc định thì phạm vi truy cập chỉ ở trong nội bộ của package mà thôi.
  + 1. Protected
* Là phạm vi truy cập có thể truy cập từ trong hoặc ngoài package, nhưng thông qua tính kế thừa. Protected chỉ có thể áp dụng bên trong class như thuộc tính, phương thức hay lớp con. Không thể áp dụng cho lớp ngoài hay interface.
  + 1. Public
* Là mức truy cập mà nó có thể truy cập ở mọi mức độ
  1. Từ khóa Static
* Dùng để khai báo một biến tĩnh. Biến đó có thể lưu thông tin chung cho tất cả các đối tượng.
* Thuộc tính static là thuộc tính mà chúng ta có thể gọi mà không cần khai báo đối tượng cho nó. Và được gọi ra bằng cú pháp <ten\_class>.<ten\_thuoc\_tinh>
* Phương thức static là phương thức mà ta cũng không cần phải tạo đối tượng mới mà vẫn có thể gọi nó. Nó rất phù hợp cho những class thư viện viết sẵn
* Khối Static được sử dụng cho mục đích khởi tạo giá trị các biến static. Khối sẽ được thực hiện khi lớp chứa nó được load vào trong bộ nhớ. Trong một lớp có thể có nhiều khối tùy ý. Các khối này sẽ chạy cùng nhau và chạy trước cả chương trình main của lớp đó
  1. Từ khóa This
* Từ khóa this dùng để ánh xạ đối tượng hiện tại.
  + Ánh xạ đối tượng khi cần sử dụng.
  + Gọi phương thức từ lớp hiện tại.
  + Gọi lại phương thức khởi tạo. Một phương thức cùng một tên nhưng có nhiều biến truyền vào gọi là overloading.
  + Trả về đối tượng (instance) của lớp hiện tại

1. Kế thừa

II.1 Định nghĩa

* Kế thừa có nghĩa là một lớp sẽ thừa hưởng lại thuộc tính và phương thức của lớp trước đó. Mục đích của kế thừa là nhằm tái sử dụng code đã viết trước đó. Thuật tiện cho việc bảo trì và nâng cấp cấu trúc
* Cấu trúc:

Class A extends B{

<phuong\_thuoc\_ke\_thua\_tu\_lop\_cha>{

Super(gia\_tri\_truyen\_vao);

<code>;

}

}

* Cha có thì con có, con có chưa chắc cha đã có.
* Từ khóa super mục đích chính là truy cập các phương thức của lớp cha.
  1. Setter và Getter
* Định nghĩa: Là hai phương thức để cập nhật hoặc lấy giá trị của thuộc tính. Đặc biệt là những thuộc tính private. Settet và getter cần thiết cho việc truy cập những luồng thông tin quan trọng thường cần giá trị chính xác.
* Cấu trúc:

Setter

Public void set<ten\_thuoc\_tinh>(<tham\_so\_gia\_tri\_moi>){

This.<ten\_thuoc\_tinh> = <tham\_so\_gia\_tri\_moi>;

}

Getter

<kieu\_du\_lieu\_cua\_thuoc\_tinh>get<ten\_thuoc\_tinh>(){

Return this.<ten\_thuoc\_tinh>;

}

* Khi đã dùng getter và setter thì thuộc tính nên để private. Vì setter và getter nhằm quản lý truy cập của thuộc tính, thì ta không nên để thuộc tính truy cập ở mức dễ dàng.
* Hãy cẩn thận với kiểu dữ liệu tham chiếu. Ta dùng phương thức clone(). Đây là phuonwg thức hỗ trợ của Java, nó sẽ tạo một bản sao rồi trả bản sao đó cho đối tượng được gán.

VD: Trong phương thức getter hoặc setter thay vì this.array = array thì ta dùng this.array = array.clone()

* Thuộc tính có kiểu dữ liệu String thì ta dùng setter với getter bình thường. Vì bản chất khi tạo 1 kiểu dữ liệu tham chiếu cho string thì thực chất nó vẫn tạo đối tượng string mới.
* Tạo phương thức clone:

Public <Ten\_class> clone(){

//Khai báo đối tượng;

//return đối\_tượng;

}

* 1. Overriding và Overloading
     1. Overriding (Ghi đè)
* Có nghĩa là hai phương thức giống nhau về tên và tham số truyền vào. Một phương thức ở lớp cha, còn phương thức còn lại ở lớp con. Overriding cho phép lớp con có thể thực hiện riêng biệt cho phương thức mà lớp cha đã cung cấp.
* Nếu không muốn lớp con overriding lại phương thức nào đó thì ta sử dụng từ khóa final trước kiểu trả về của phương thức ở lớp cha.
  + 1. Overloading (Nạp chồng)
* Có nghĩa là nhiều phương thức trong một lớp có chung tên nhưng khác tham số truyền vào

1. Tính trừu tượng trong Java
   1. Tính trừu tượng

* (Tài liệu)
  + 1. Tính trừu tượng trong lập trình hướng đối tượng
* Tính trừu tượng trong hướng đối tượng là che giấu thông tin thực hiện từ người dùng. Người dùng chỉ biết tính năng đang được cung cấp và thông tin đối tượng.

VD: Nếu bạn lập trình ra 1 phần mềm rất lớn. Bạn không thể tưởng tượng hết tất cả các khả năng nó có thể xảy ra trong đầu. Thay vì phải nạp hết vào đầu thì ta trừu tượng hóa nó để tạo ra 1 khung cho biết tính năng mình đang muốn thực hiện thì nó sẽ có kết quả như thế nào. Còn làm như thế nào để ra kết quả như vậy ta tính sau.

* + 1. Tính trừu tượng trong Java
  + Lớp trừu tượng (Asbtract Class)
    - Lớp trừu tượng là lớp được khai báo mà không thể tạo ra đối tượng từ lớp đó. Ta sẽ tạo những lớp con kế thừa lớp trừu tượng.
    - Mục đích của lớp trừu tượng là tạo ra lớp chung cho những lớp có liên quan với nhau kế thừa.

VD: Khi xây dựng phần mềm quản lý nhà trường. Thì những lớp Sinh viên, giảng viên, cán bộ,... có những thuộc tính và phương thức chung như: Họ tên, giới tính, chức vụ,.. Thì ta sẽ tạo lớp trừu tượng là lớp Persion để khai báo những thuộc tính chung đó. Khi phát triển chương trình ta có thể tạo ra các lớp đối tượng từ lớp con kế thừa từ lớp con người đó. Không thể cho tạo đối tượng từ lớp con người được.

* + Phương thức trừu tượng
    - Các phương thức trừu tượng là các phương thức chỉ định nghĩa mà không có chương trình bên trong, lớp con kế thừa phải bắt buộc override nó để sử dụng.
    - Phương thức trừu tượng có ý nghĩa định nghĩa phương thức bắt buộc phải có trong lớp con kế thừa
    - Nên cho các thuộc tính ở lớp trừu tượng ở mức protected để lớp con có thể kế thừa dễ dàng hơn thay vì private
  1. Interface
     1. Định nghĩa
     + Interface là một kiểu dữ liệu tham chiếu trong java. Nó là tập hợp các phương thức abstract.
     + Khi 1 class kế thừa interface thì nó sẽ kế thừa những phương thức abstract của interface đó.
     1. Đặc điểm
     + Không thể khởi tạo nên không có phương thức khởi tạo
     + Tất cả các phương thức trong interface luôn ở dạng public abstract mà không cần khai báo.
     + Các thuộc tính trong luôn ở dạng public static final mà không cần khai báo, yêu cầu phải có giá trị
     1. Mục đích
     + Là để thay thế đa kế thừa của lớp trong các ngôn ngữ lập trình khác.
     + Interface sẽ giúp đồng bộ hóa và thống nhất trong việc đồng bộ hóa và trao đổi thông tin.
     + Khi đặt tên của interface nên cho thêm chữ I vào trước tên đó.
     + Chỉ có thể extends 1 class cha nhưng có thể implement bao nhiêu interface cũng được.
     + Nếu một lớp abstract mà implement một interface nào đó thì class con của lớp abstract đó mặc dù không implement trực tiếp nhưng vẫn phải override phương thức của interface đó

1. Try-catch