1. ***Encapsulation***: (đóng gói) -> Bảo mật => dùng Access Modifier (private)

***Để thk khác có thể sài thì ‘ public hàm get,set’***

1. ***Inheritance***: (kế thừa) -> Tái sử dụng

***1 cấp: 1 class kế thừa 1 class***

***Đa cấp: nối đuôi ( bắc cầu )***

1. ***Polymorphism*** (đa hình)

***Có vài cái hàm, phương thức có tên giống hệt nhau Nhưng:***

***Overloading : khác parameter truyền vào***

***Ví dụ có thể chọn ăn trưa với cơm và canh. Eatlunch ( rice, canh)***

***Với nước mắm . Eatlunch ( souce )***

***Overiding : viết lại hàm của thằng cha***

***Khi khởi tạo hàm mà nó có thể override đc ở class con thì ta sd từ khóa -> virtual khi khởi tạo hàm***

* Chúng ta **không thể ghi đè (overriding)** những phương thức **final** và **static**.
* Lớp con **phải luôn luôn ghi đè (overriding)** phương thức **abstract** của lớp cha.
* Phương thức được ghi đè ở lớp con có thể có **access modifier** khác với lớp cha. Nhưng **access modifier** này phải có phạm truy truy cập lớn hơn của lớp cha

1. ***Abstraction:*** (trừu tượng)

Dùng ***Abstract class*** hoặc ***Interface*** khi không muốn class tạo nên một ***instance***(thực thể) của cái class đó.

**Abstaction** đc sử dụng để ngta phân biệt đc class thuộc chủng loại riêng biệt. vd class Animal ko thể là class Door

Tính chất của trừu tượng gồm: Abstract class và Interface :

|  |  |
| --- | --- |
| * abstract class (Abstract chưa hoàn toàn) * Để tạo cái khung của 1 object nào đấy mà ko tạo **instance** * Nếu sd **abstract class** phải có class nào đó extends abstract class đấy * Abstract class ~ Normal class . Nhưng có thêm **abstract function()** * Thk nào extends * 1 AbClass nên có ít nhất 1 abstract method | * Interface (đếch biết làm gì hết) * Nếu sử dụng **interface** thì phải có class nào đấy implements cái interface đó * Interface chỉ chứa mỗi **abstract function() , *nếu có viết* Properties *thì phải là* static, final**   Implements thì bắt buộc @Override abstract function |

Cấp phát vùng nhớ -> new

Int [] array = new

Kích thước dữ liệu mảng dễ xài nhưng bị fix kích thước

**Data type:**

1. Primitive: ( int, char, float, boolean… )
2. Reference: ( class: Dog, Animal, Student )

*Instance* là một biến đã được cấp phát vùng nhớ

Class ~~ datatype do chúng ta tự định nghĩa, khuôn do cta đúc ra

**Encapsulation** ( public ) : tính đóng gói

1. Details of a student include code, name, year of birth, address. Write a Java program that will allow input a student, output his/her,

|  |
| --- |
| Class Classname <- Main Noun |
| * Property 1 * \_\_\_\_\_\_ 2 <- Noun * \_\_\_\_\_\_ 3 |
| + method 1  + method 2 <- verb |

|  |
| --- |
| Class Student |
| * private String code; * private String name; * private int year\_of\_birth; * private String address; |
| + Public void input Student()  + Public void output Student() |

**Modifier in Encapsulation concepts :**

* Private: trong cùng một Class có thể truy cập được
* Default (Ex: int size ) Trong cùng một package có thể truy cập được
* Protected: trong cùng một Class con (Subclass – khi được extends) có thể truy cập được
* Public: bất cứ đau cũng có thể truy cập đc

Propeties: sở hữu, tài sản : -> bắt buộc phải là **Private**

|  |
| --- |
| Class Array |
| * private int size; * int[] element; |
| + public void enterSize()  + public void enterElement()  + public void display()  + public void findSmallest()  + public void calculateAverage() |

2) Write a program to peform operations on integer arrays with the following function:

- User enter the size of the array

- The user enters each element of the array

- Display the value of each element in the array.

- Find the smallest in the array and display it on the screen

- Calculate the average value of the array

2. Tính kế thừa ( INHERITANCE use key word: “ EXTENDS ” )

- khai báo ‘protected’ . Class ( A extends B) : A kế thừa thk B. A có thể sử dụng các thuộc tính của thk B

3.. Tính đa hình ( POLYMORPHISM )

* Overloading: Cùng 1 class có *nhiều hàm cùng tên* nhưng **khác nhau** về mặt tham số truyền vào
* Overriding: *Lớp con viết lại hàm của thằng cha*. Tại sao phải viết lại? *Vì thk con có cách xử lí khác thk cha*

REVIEW: **1 CLASS GỒM ( PROPERTIES , METHOD )**

CLASS : danh từ chính

* Properties là những danh từ biểu diễn cho danh từ chính
* Method là hành động của cái Class

**Modifier** in Encapsulation :

* Private: used for Propeties ( thuộc tính) : cùng class
* No(default),: cùng class + package
* Protected : cùng class +package + **subclass**
* Public

‘ **Final** ’ đặt trước :

* Class: ko cho kế thừa ; *Final class*: Class don’t have sub-class: ( triệt sản thk cha )
* Method: cho kế thừa nhưng không cho Overiding
* Properties: hằng số ( final public int MAXN = 5)
* Empty properties:

**Static:**

1. Mot hang so chưa khoi tao gia tri - Khoi tao gia tri trong ham khoi tao

=> Khai báo trước **Method** dùng để truy cập mà ko cần khởi tạo

2. Khoi tao gia tri trong block static { } - block static dc thuc thi truoc khi Class được gọi đến

Từ “ this” là một từ khóa: truy cập đến chính nó, để tham chiếu đến thuộc tính hoặc phương thức = > this dùng để phân biệt giữa Attribute( thuộc tính ) và Parameters ( tham số ) .

Một hàm cấp phát và một hàm khởi tạo thằng nào chạy trước ?

MyClass obj = new MyClass();

Hàm khởi tạo có tên = Class\_name. [ MyClass(): là hàm khởi tạo ]

Hàm cấp phát:

**Class Relationship:**   
“Is a/ a kind of” : mqh Cha Con ( A is B: A-Cha, B-Con )

“Has a” : biểu diễn properties

“association” : quan hẹ 1vs1, 1vsnhieu, nhieu vs nhieu

Trong Java là đơn kế thừa ( nhưng vẫn có cách để đa kế thừa )

**Inheritant:**

Keyword: Extends

Object class:

Super:

Empty properties:

**Overriding:**

Sub class change modifier? Phạm vi truy cập lớn hơn lớp Cha

Override methods: viết lại hàm của thằng cha. Để ko viết được thì dùng từ khóa **Final**

Hidding methods: thk con viết lại một hàm (static method) của thk cha

~~Con = new Cha()~~ : ko đc . Cha mang nợ thì con ko trả đc

**Cha a = new Con();** được vì Con vùng nhớ lớn hơn Cha

Được truy cập vô phương thức của **CHA**

Còn muốn truy cập vo thk **CON** phải ép kiểu => ( (Con) a). method();

Chạy vô thk cha trước : nếu có static thì dùng của thk CHA , nếu ko có static thì chạy xuống thk con xem có ko

Chỉ có Static mới chơi với Static

**Interface : ( toàn bộ là những Abstract method ko có body )**

Dùng từ khóa implements( giống như extends)

Interface chỉ chứa các Abstract fuction() không có body

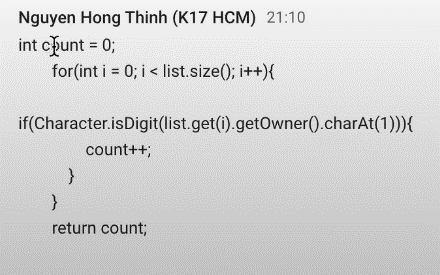
**Abstact Classes :**

Không tạo ra **instances**

Gồm cả Abstract method() và Non-abstract Method()

Dùng từ khóa extends .

Abstract method: thk con bắt buộc phải Override lại Abstract method



Chỉ được thừa kế( Extends) một cha ruột

Được thừa kế( Implements) nhiều cha nuôi ( Interface )

* Nói đó là đa kế thừa là vì ( kế thừa cả cha ruột lẫn nhiều cha nuôi )

Pet System – Giáo làng

Class MyToys: Utils ( getInterger, getDouble, getID,getString )

Class Cabinet: Hàm ( add,search,remove,print List)