**Chương 2: OOP - Object-Oriented Programming**

1. **Khái niệm**

**OOP (Object-Oriented Programming)** hay còn gọi là Lập trình hướng đối tượng là phương pháp viết chương trình dựa vào đối tượng thay vì viết chương trình tuần tự if else,.....

*Vậy thế nào là Object (Đối tượng) ?*  
Đối tượng là những sự vật,việc hay vật thể mang những đặc điểm và tính chất hoặc hành động chung.

Ví dụ:

Con người có chiều cao, cân nặng, Đẹp, xấu / ăn, uống, đi lại

| | |

Vật thể Đặc điểm, tính chất Hành động

Vậy ta có thể nhìn thấy Con người có các đặc điểm, tính chất và hành động chung của các con người nói chung vậy nên ta có thể xem nó là 1 Object

**Note: đôi khi có những Object có thuộc tính riêng**

1. **Các nguyên lý của OOP**

*Nhưng lý thuyết trên chỉ dành cho thực tế, muốn máy chúng ta hiểu thì làm sao ??*

Trong lập trình Java, chúng ta có 4 khái niệm để hiện thực hóa điều này :

* **Class**: bạn cũng có thể xem 1 class là template của Object, class sẽ làm nhiệm vụ chứa tất cả các thuộc tính và hành động của Object này vào trong class, hãy tưởng tượng rằng class giống như việc nhà bạn còn các đồ như ti-vi,ghế, bàn là thuộc tính nằm trong nhà vậy (Note: chú ý class nếu chưa đc new thì vẫn chưa được xemlà 1 Object thật sự )
* **Attribute**: vì trong class chứa các đặc điểm tính chất của Object nên ta quy chung về 1 khái niệm gọi là Attribute (Thuộc Tính)
* **Method**: Ngoài đặc điểm tính chất ta còn có Hành Động của Object đó nữa và ta cũng ngữ gọi Method (Phương thức)
* **Constructor**: đây là 1 phương thức đặc biệt trong 1 class,vì đây là nơi quyết định Object có được khởi tạo hay không

public class Human{ ---> Class đây nè

String name;

int age;

double height;

double weight; ---> đây gọi là Attribute

public Human(){

} -> đây là constructor

public Human(String name, int age, double height,double weight){

this.name = name;

this.age = age;

this.height = height;

this.weight= weight;

} --> đây cũng là constructor

public String getName(){

return name;

} --> Đây là method nà dùng để lấy tên của Object Human

public String setName(String name)}

this.name = name;

} --> đây là method luôn nà dùng để đặt name của

}

Ohh ta thấy trong class này có tận 2 Constructor sao vậy ta ??

* Như ta đã thấy Constructor là cần có để có thể tạo Object,khi tạo Object làm sao chúng ta thêm thông tin Attribute cho Object đó đây nên ta mới có 2 dạng Constructor để thực hiện luôn việc này
* Với Constructor public Human() như chúng ta gọi chúng là Constructor mặc định vì là mặc định nên bắt buộc có . Chúng ta sao khi tạo Object thì Constructor này sẽ cho phép chúng ta gọi các method của Object Huamn này, chúng ta có thể gọi phương thức Set để thực hiện thêm thông tin Attribute cho Object.
* Còn public Human(String name, int age, double height,double weight) Constructor này người ta gọi là Constructor khi chúng ta new Object thì có thể thực hiện thêm thông tin trực tiếp vào mà không cần gọi phương thức Set để thêm thông tin

Ví dụ:

Human human = new Human(); ---> Đây là cách tạo 1 Object và Object đó đang sử dụng Constructor mặc định

Human.setName(“Phát”); --> Đây là lúc thêm thông tin Object thông qua việc gọi method Set

Human human = new Human(“Phát”, 20, 1.73, 72.5); --> Đây là chúng ta thực hiện thêm thông tin trực tiếp bằng cách sử dụng Constructor có tham số

Note: Khi chúng ta tạo Constructor có tham số thì nên thêm thông tin đúng chỗ và đúng kiểu dữ liệu nếu bỏ trống thì dữ liệu sẽ bị null

**OOP có 4 nguyên lý cơ bản :**

* Encapsulation (Tính Đóng Gói)
* Inheritance (Tính Kế Thừa)
* Polymorphism (Tính Đa Hình)
* Abstraction (Tính Trừu Tượng)

1. **Encapsulation - Tính Đóng Gói**

Trong thực tế, trang thông tin Facebook của bạn, bạn có thể tùy chỉnh cho người lạ hoặc bạn bè thấy được thông tin của bạn như họ tên, giới tính, số điện thoại (public) ,..... hoặc hiệu chỉnh không được nhìn thấy như danh sách bạn bè, bài viết (private), ..... Nhằm mục đích kiểm soát và bảo vệ thông tin của bạn và đó cũng chính là ý nghĩa của Encapsulation.

Trong Java tính Đóng Gói (Encapsulation) cũng có ý nghĩa như vậy,các Attribute, Method của Class này sẽ được che giấu hoặc công khai thông qua 1 khái niệm là Access Modifiers, 1 class đúng nghĩa có tính đóng gói nên có đầy đủ các Attribute của Object, Constructor mặc định , Các method liên quan đến Object (nên có đầy đủ Get và Set method).

Ví dụ:

public class Human {  
 public String name; // bạn đang công khai tên của mình cho cả thế giới biết  
 public int age; // bạn đang công khai tuổi của mình cho cả thế giới biết  
 private String gender; // bạn đang giấu giới tính của mình với thiên hạ  
 private Date dob; // bạn đang che giấu đi sinh nhật của mình  
  
 public Human() {  
 }  
  
 public Human(String name, int age) {  
 this.name = name;  
 this.age = age;  
 }  
   
 private Date calculateNextBirthday(){  
 Date today = new Date();  
 Date nextBirthday = new Date(dob.getYear(), dob.getMonth(), dob.getDate());  
 if (nextBirthday.before(today)) {  
 nextBirthday.setYear(nextBirthday.getYear() + 1);  
 }  
 return nextBirthday;  
 } // bạn đang che đi cách tính sinh nhật tiếp theo của mình  
  
 public String getName() {  
 return name;  
 }  
  
 public int getAge() {  
 return age;  
 }  
  
 public void setAge(int age) {  
 this.age = age;  
 }  
  
 public String getGender() {  
 return gender;  
 }  
  
 public void setGender(String gender) {  
 this.gender = gender;  
 }  
  
 public Date getDob() {  
 return dob;  
 }  
  
 public void setDob(Date dob) {  
 this.dob = dob;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
  
  
 @Override  
 public String toString() {  
 return "Human{" +  
 "name='" + name + '\'' +  
 ", age=" + age +  
 '}';  
 }  
}

1. **Inheritance (Tính kế thừa)**

Ở thực tế các bạn hãy nhìn lại chiếc xe bạn đi hằng ngày thời còn đại học có phải của ba mẹ bạn không, hoặc đôi lúc bạn cũng được ba mẹ nhượng lại những chiếc điện thoại loanh toanh hay những bộ đồ. Hay là có một cô dì họ nào của bạn nói bạn rằng “ Ý chà thằng này cao y như cha nó nhỉ “ thì có nghĩa là bạn được thừa hưởng chiều cao từ cha của bạn.

Trong Java cũng có sự thừa hưởng hay kế thừa như vậy, những class cho phép kế thừa hay còn gọi là class cha có thuật ngữ là Superclass, còn các class đi kế thừa hay còn lớp con có thuật ngữ là Subclass và các class con được kế thừa thì có thể sử dụng các method ở class cha

ví dụ:

1. **Polymorphism (Tính Đa Hình)**

Có bao giờ bạn nghĩ rằng cùng 1 ly cà phê những mỗi quán ra 1 vị khác nhau chưa, và cùng 1 loại cà phê đấy nhưng lại nhiều giá thành khác nhau luôn. Đó là do các quán sử dụng nhiều phương thức pha chế và nguyên vật liệu khác nhau. Đây có thể xem 1 dạng đa hình.

Vậy đa hình là gì ?

Đa hình, Đa tức là đa dạng, hình là hình thức vậy đa hình là sự đa dạng thức của 1 việc gì đó. Trong code cũng vậy đa hình giúp chúng ta đa dạng các phương thức với nhau để tăng tính phong phú của code.

Trong tính đa hình OOP có 2 loại :

* Overloading: trong 1 class có nhiều method trùng nhau nhưng lại khác tham số truyền vào. Cũng như việc trong 1 bếp ăn cùng 1 món cơm chiên nhưng đầu bếp A chiên cơm cho topping sau cùng, đầu bếp B chiên cơm cho topping vào trước

Ví dụ:

public class Laptop extends Technology{  
 private String brand;  
 private String color;  
 private double price;

// hãy nhìn ở đây ta có tận 3 method total dùng để tính giá cả nhưng các tính giá mỗi phương thức khác nhau ta gọi đây là overloading  
public double total(){  
 return price;  
}  
  
public double total(int quantity){  
 return price \* quantity;  
}  
  
public double total(int quantity, double discount){  
 return price \* quantity - discount;  
}

}

**Note: Nhớ rằng Overloading nằm là thuật ngữ đúng khi các method nằm trong 1 class**

* Overriding: Hãy tưởng tượng rằng ba được ba giao lại chiếc xe nhưng bạn lại thấy chiếc xe khá xấu không ưng thế nên bạn định sang sửa chiếc xe. Trong code đây có nghĩa là bạn sử dụng lại phương thức lớp cha tại lớp con nhưng tùy chỉnh theo ý muốn và ghi đè lại phương thức lớp cha

Ví dụ:

public class Technology {

public String getName() {  
 return name;  
}

}

public class Laptop extends Technology{

@Override  
public String getName() {  
 return brand + " " + super.getName();  
}

}

Như chúng ta đã thấy class con Laptop sử dụng lại phương thức getName của class cha Technology nhưng thay vì trả về name thì class con lại trả về thêm cả brand nữa.

**Note: Khi sử dụng lại phương thức của lớp cha cần có anotation @Override để có thể hiểu và tùy chỉnh phương thức được kế thừa**

1. **Abstraction (Tính trừu tượng)**

Có bao giờ remote TV và bạn tò mò rằng nó hoạt động thế nào chưa, giả rằng bạn tháo vỏ hộp của remote ra liệu bạn có biết nút là nút on/off không, vì thế chúng ta mới cần 1 vỏ case và các phím để bạn dễ dàng phân biệt . Đó là ý nghĩa của việc Abstraction.

Ví dụ:

public abstract class Car { // đây là abstract class  
 private String name;  
  
 public abstract void makeSound(); // đây là abstract method\  
 // abstract method không có body  
 // đây là phương thức trừu tượng làm ẩn đi chi tiết sound của xe  
}

public class Mercedes extends Car{

@Override  
 public void makeSound() {  
 System.*out*.println("Brummmm brummmm");  
 }  
}

public class Bwm extends Car{  
  
 @Override  
 public void makeSound() {  
 System.*out*.println("Vroom vroom");  
 }  
}

Note: Khi class extends (Kế thừa) abtract class thì bắt buộc override lại tất cả các phương thức trừu tượng (abstract method) có trong absstract class