**https://kipalog.com/posts/4-tinh-chat-dac-thu-cua-lap-trinh-huong-doi-tuong**

**What Is a Class?**

Classes are the basic units of programming in the object-oriented paradigm. They are used as templates to create objects.

A class in Java may consist of five components:

* Fields
* Methods
* Constructors
* Static initializers
* Instance initializers

Fields and methods are also known as members of the class.

Classes and interfaces can also be members of a class.

A class can have zero or more class members.

Constructors are used to create objects of a class. We must have at least one constructor for a class.

Initializers are used to initialize fields of a class. We can have zero or more initializers of static or instance types.

## Declaring a Class

The general syntax for declaring a class in Java is

<modifiers> class <class name> {

// Body of the class

}

<modifiers> are keywords with special meanings to the class declaration.

A class declaration may have zero or more modifiers.

The keyword class is used to declare a class.

<class name> is a user-defined name for the class, which should be a valid identifier.

Each class has a body, which is specified inside a pair of braces ({}).

The body of a class contains its different components, for example, fields, methods, etc.

The following code defines a class named Dog with an empty body. Dog class does not use any modifiers.

class Dog {

}

## Dot Notation to Access Fields of a Class

Dot notation is used to refer to instance variables.

The general form of the dot notation syntax is

<Reference Variable Name>.<Instance Variable Name>

obj.name to refer to the name instance variable of the instance to which the obj reference variable is referring.

To assign a value to the name instance variable, use

obj.name = "Rectangle";

The following statement assigns the value of the name instance variable to a String variable aName:

String aName = obj.name;

To reference class variables, use the name of the class.

ClassName.ClassVariableName

For example, we can use Dog.count to refer to the count class variable of the Dog class.

To assign a new value to the count class variable

Dog.count = 1;

To read the value of the count class variable into a variable

long count = Dog.count;

The following code shows how to use class fields

class Dog {// www.ja va2 s . co m

static int count = 0;

String name;

String gender;

}

public class Main {

public static void main(String[] args) {

Dog obj = new Dog();

// Increase count by one

Dog.count++;

obj.name = "Java";

obj.gender = "Male";

obj.name = "XML";

String changedName = obj.name;

}

}

# **4 tính chất đặc thù của lập trình hướng đối tượng**

Với [lập trình hướng đối tượng](http://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_h%C6%B0%E1%BB%9Bng_%C4%91%E1%BB%91i_t%C6%B0%E1%BB%A3ng) (OOP), có 2 câu hỏi thực dụng:

* Nó có những tính chất đặc thù gì? Để ta có thể dễ nhớ, dễ hiểu, dễ hình dung về nó.
* Khi học một ngôn ngữ OOP nào đó, làm sao để kiểm tra xem ta đã nắm bắt được ngôn ngữ này hay chưa, bằng cách viết thử code để diễn tả các đặc thù này bằng cú pháp của ngôn ngữ này?

## Trả lời câu hỏi 1

[OOP](http://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_h%C6%B0%E1%BB%9Bng_%C4%91%E1%BB%91i_t%C6%B0%E1%BB%A3ng) có 4 tính chất đặc thù chính, các ngôn ngữ OOP nói chung đều có cách để diễn tả:

* Tính đóng gói: Có thể gói dữ liệu (data, ~ biến, trạng thái) và mã chương trình (code, ~ phương thức) thành một cục gọi là lớp (class) để dễ quản lí. Trong cục này thường data rất rối rắm, không tiện cho người không có trách nhiệm truy cập trực tiếp, nên thường ta sẽ che dấu data đi, chỉ để lòi phương thức ra ngoài. Ví dụ hàng xóm sang mượn búa, thay vì bảo hàng xóm cứ tự nhiên vào lục lọi, ta sẽ bảo: "Ấy bác ngồi chơi để tôi bảo cháu lấy cho". Ngôn ngữ Ruby "phát xít" đến nỗi dấu tiệt data, cấm không cho truy cập từ bên ngoài. Ngoài ra, các lớp liên quan đến nhau có thể được gom chung lại thành package (tùy ngôn ngữ mà còn gọi là module, namespace v.v.).
* Tính trừu tượng: Có câu "program to interfaces, not to concrete implementations". Nghĩa là khi viết chương trình theo phong cách hướng đối tượng, khi thiết kế các đối tượng, ta cần rút tỉa ra những đặc trưng của chúng, rồi trừu tượng hóa thành các interface, và thiết kế xem chúng sẽ tương tác với nhau như thế nào. Nói cách khác, chúng ta định ra các interface và các contract mà chúng cần thỏa mãn.
* Tính thừa kế: Lớp cha có thể chia sẻ dữ liệu và phương thức cho các lớp con, các lớp con khỏi phải định nghĩa lại những logic chung, giúp chương trình ngắn gọn. Nếu lớp cha là interface, thì lớp con sẽ di truyền những contract trừu tượng từ lớp cha.
* Tính đa hình: Đối tượng có thể thay đổi kiểu (biến hình). (1) Với các ngôn ngữ OOP có kiểu, có thể mượn phát biểu của C++ "con trỏ kiểu lớp cha có thể dùng để trỏ đến đối tượng kiểu lớp con". Như vậy khi khai báo chỉ cần khai báo p có kiểu lớp cha, còn sau đó nó trỏ đến đâu thì kệ cha con nó: nếu cha và con cùng có phương thức m, thì từ p cứ lôi m ra gọi thì chắc chắn gọi được, không cần biết hiện tại p đang trỏ đến cha hay con. Khi lớp B thừa kế từ lớp A, thì đối tượng của lớp B có thể coi là đối tượng của lớp A, vì B chứa nhiều thứ thừa kế từ A. (2) Với ngôn ngữ OOP không có kiểu như Ruby, có thể mượn phát biểu của [phương pháp xác định kiểu kiểu con vịt](http://en.wikipedia.org/wiki/Duck_typing): "nếu p đi như vịt nói như vịt, thì cứ coi nó là vịt". Như vậy nếu lớp C có phương thức m, mà có thể gọi phương thức m từ đối tượng p bất kì nào đó, thì cứ coi p có kiểu là C.

Để dễ nhớ, có thể chia 4 đặc thù làm 2 nhóm:

1. Nhóm 1: tính chất 1. Tính đóng gói là tính dễ nhận thấy nhất nếu bạn bắt đầu học OOP sau khi đã học qua những ngôn ngữ thủ tục như C và Pascal (thường trường phổ thông ở Việt Nam đều dạy).
2. Nhóm 2: tính chất 2, 3, và 4 đi một dây với nhau.

## Trả lời câu hỏi 2

Nên làm bài tập nhỏ về các con vật sau, gọi tắt là bài 4OOP:

* Tạo interface Animal có phương thức say\_hello. <- Thể hiện tính trừu tượng, có nghĩa ta định ra contract là rằng dù là con vật gì đi nữa thì nó cũng có phương thức say\_hello để chào hỏi gì đấy.
* Tạo 2 lớp Cat và Dog kế thừa từ Animal. Khi khởi tạo chúng sẽ có tên. Chúng override lại phương thức say\_hello để chào hỏi theo cách riêng của chúng. <- Thể hiện tính đóng gói (đóng gói biến tên và phương thức say\_hello với nhau) và tính thừa kế (Cat và Dog mang đặc điểm chung là có say\_hello từ Animal).
* Tạo lớp Zoo để quản lí nhiều Animal, có (1) phương thức add, remove để thêm, bớt các Animal (các đối tượng của các lớp thừa kế từ Animal), (2) phương thức say\_hello\_all để gọi say\_hello của tất cả đối tượng nó quản lí. <- Thể hiện tính đa hình, Zoo gọi chỉ gọi một phương thức say\_hello, nhưng tùy con vật mà lời chào hỏi sẽ khác nhau.

Đây là bài tập gối đầu giường, khi được yêu cầu viết phải viết được ngay không ngắc ngứ.