**4 tính chất cơ bản của OOP là :**

* Tính kế thừa (Inheritance)
* Tính đóng gói(Encapsulation)
* Tính đa hình (Polymorphism)
* Tính trừu tượng (Abstraction)

1. Tính kế thừa
   * Đặc tính này cho phép một đối tượng có thể có sẵn các đặc tính mà đối tượng khác đã có thông qua kế thừa. Điều này cho phép các đối tượng chia sẻ hay mở rộng các đặc tính sẵn có mà không phải tiến hành định nghĩa lại
   * Cú pháp kế thừa

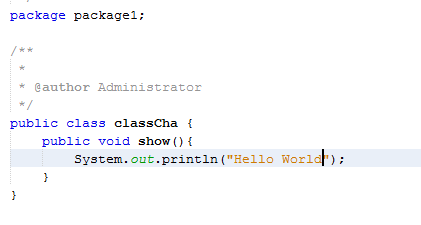
class lop\_cha extends lop\_con{

//cac phuong thuc

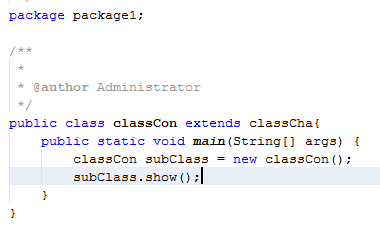
}

* + Ví dụ

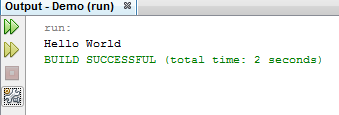
Class Cha



Class Con



Kết quả

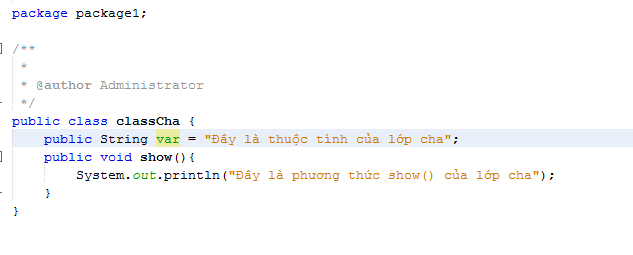
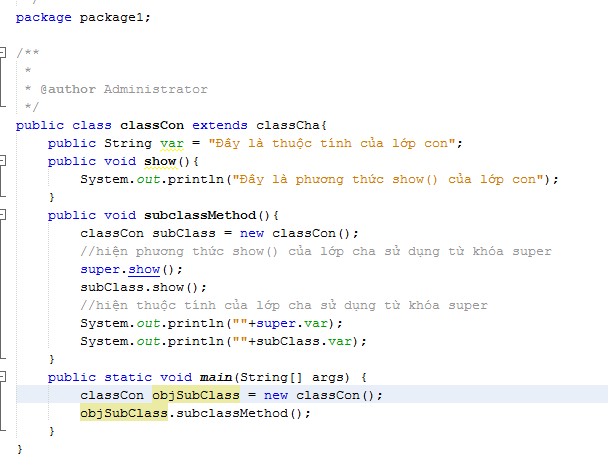


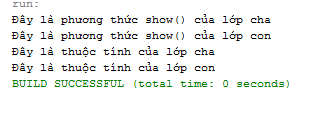
* + Trên cơ sở các lớp thì có 3 loại kế thừa trong Java, đó là *single (đơn), multilevel (nhiều tầng) và hierarchical (có cấu trúc)*. Trong lập trình Java, *đa kế thừa (multiple) và kế thừa lai (hybrid)* chỉ được hỗ trợ thông qua Interface.
  + Từ khóa super :
    - Được sử dụng trong các trường hợp sau :
      * Nó được sử dụng để phân biệt các thành phần có cùng tên giữa lớp cha và lớp con.
      * Nó được sử dụng để gọi hàm tạo của lớp cha từ lớp con.
    - **Phân biệt các thành phần cùng tên giữa lớp cha với lớp con**
      * Nếu lớp con kế thừa lớp cha và các thành phần của lớp cha có cùng tên với các thành phần của lớp con, chúng ta sẽ dùng từ khóa *super* như sau :

super.ten\_thuoc\_tinh;

super.ten\_phuong\_thuc();

* + - * Ví dụ

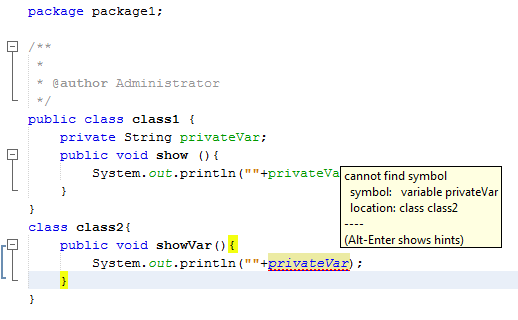
Kết quả



1. Tính bao đóng
   * Tính chất này không cho phép người sử dụng các đối tượng thay đổi trạng thái nội tại của một đối tượng. Chỉ có các phương thức nội tại của đối tượng cho phép thay đổi trạng thái của nó. Việc cho phép môi trường bên ngoài tác động lên các dữ liệu nội tại của một đối tượng theo cách nào là hoàn toàn tùy thuộc vào người viết mã. Đây là tính chất đảm bảo sự toàn vẹn của đối tượng.
   * Tính chất này thể hiện qua các access modifier : private, default ,protected, public
   * Tổng quan về phạm vi của các modifier

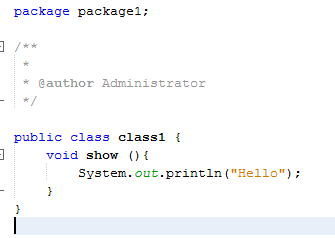
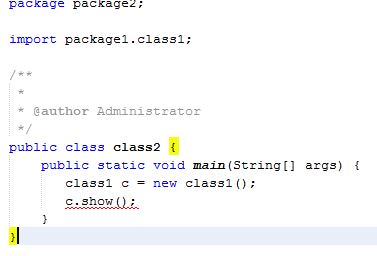
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Access Modifier** | **Truy cập bên trong class?** | **Truy cập bên trong package?** | **Truy cập bên ngoài package bởi class con?** | **Truy cập bên ngoài class và không thuộc class con?** |
| **private** | Y |  |  |  |
| **Default** | Y | Y |  |  |
| **protected** | Y | Y | Y |  |
| **public** | Y | Y | Y | Y |

1. Private access modifier
   * Private access modifier chỉ cho phép truy cập trong nội bộ 1 class
   * Ví dụ :

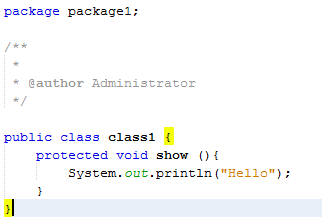


1. Default access modifier

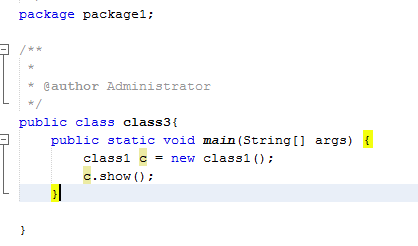
* Default access modifier cho phép truy cập trong cùng 1 package
* Ví dụ :

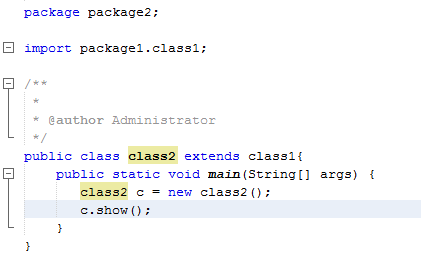
1. Protected access modifier
   * Protected access modifier có thể truy cập cung 1 package hay bên ngoài package (nhưng phải thông qua kế thừa)



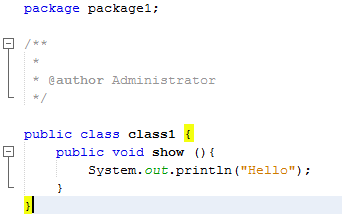
Cùng một package

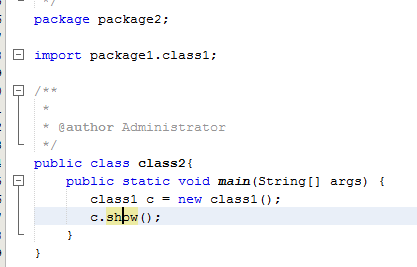


Khác package



1. Public access modifier
   * Public access modifier cho phép truy cập ở bất cứ đâu
   * Ví dụ





1. Tính đa hình
   * Tính đa hình cho phép  gia tăng khả năng tái sử dụng những đoạn mã nguồn được viết một cách tổng quát và có thể thay đổi cách ứng xử một cách linh hoạt tùy theo loại đối tượng.
   * Tính đa hình thể hiện qua ghi đè (Overriding) và nạp chồng (Overloading)
2. Tính trừu tượng
   * Đây là khả năng của chương trình bỏ qua hay không chú ý đến một số khía cạnh của thông tin mà nó đang trực tiếp làm việc lên, nghĩa là nó có khả năng tập trung vào những cốt lõi cần thiết
   * Ưu điểm
     + Tính trừu tượng cho phép các lập trình viên 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, cải thiện khả năng bảo trì của hệ thống.
     + Tính trừu tượng giúp chúng ta tập trung vào những cốt lõi cần thiết của đối tượng thay vì quan tâm đến cách nó thực hiện.
     + Tính trừu tượng cung cấp nhiều tính năng mở rộng khi sử dụng kết hợp với tính đa hình và kế thừa trong lập trình hướng đối tượng.

* Phương thức trừu tượng (Abstract method)

Cú pháp khai báo

[access\_modifier] abstract [kiểu\_trả\_về] [tên\_phương\_thức\_trừu\_tượng] [<đối\_số\_truyền\_vào>];

Trong đó

* **[access\_modifier]** là phạm vi truy cập của phương thức trừu tượng. Phạm vi truy cập của phương thức trừu tượng tương tự như của các phương thức bình thường nhưng không được khai báo phạm vi truy cập là private, nếu để là private thì trình biên dịch sẽ báo lỗi.
* **[kiểu\_trả\_về]** là kiểu dữ liệu của phương thức.
* **[tên\_phương\_thức\_trừu\_tượng]** phải tuân theo quy tắc đặt tên phương thức (*hàm*) của Java.
* Phương thức này có thể có hoặc không  có **<đối\_số\_truyền\_vào>**

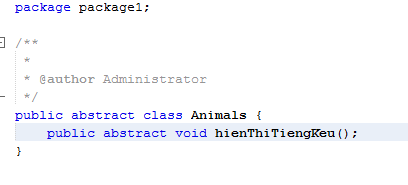
***Lưu ý:*** Để sử dụng phương thức trừu tượng này, chúng ta cần phải ghi đè (override) nó trong lớp con kế thừa trực tiếp lớp khai báo phương thức này.

* **Lớp trừu tượng (Abstract class)**
  + Lớp trừu tượng là lớp được khai báo với từ khóa abstract đứng trước tên của lớp.
  + Nếu 1 lớp được khai báo là 1 lớp trừu tượng thì chúng ta không thể dùng trực tiếp nó để tạo ra đối tượng mà phải viết một lớp kế thừa của lớp trừu tượng đó.
  + Lớp trừu tượng có thể có hoặc không có phương thức trừu tượng. Nếu một lớp có ít nhất 1 phương thức trừu tượng thì lớp đó phải được khai báo là lớp trừu tượng.
  + Những lớp là lớp trừu tượng cũng không cần có phương thức khởi tạo.
  + Một khi có một lớp nào đó kế thừa lớp trừu tượng thì lớp con đó bắt buộc phải override lại nội dung tất cả các phương thức trừu tượng có trong lớp đó.

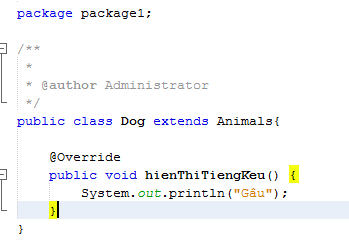
Tóm lại, lớp trừu tượng là 1 lớp không thể khởi tạo đối tượng từ nó, nhưng nó lại ràng buộc các lớp con kế thừa trực tiếp nó phải có các phương thức trừu tượng của nó thông qua sự ghi đè (override) phương thức.

Ví dụ

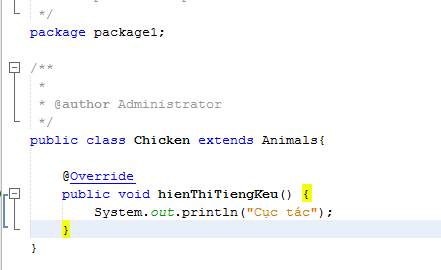
**Animal.java**



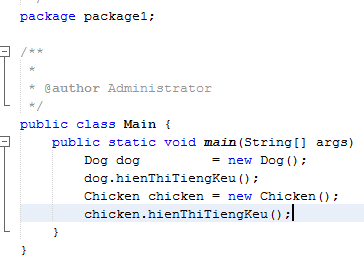
**Dog.java**



**Chicken.java**



**Main.java**



Kết quả

