**Abstract Class**

- Một lớp có tính trừu tượng tới mức không thể tạo được đối tượng của lớp đó gọi là lớp trừu tượng (abstract class).

- Class abstract không thể được định nghĩa (hiện thực hoá) bằng contructor.

- Lớp abstract có thể tạo ra các phương thức abstract (phương thức không có phần thân), các class con của abstract class phải hiện thực lại cụ thể các phương thức abstract từ lớp cha của nó.

==> Abstract method (phương thức trừu tượng) là những phương thức được khai báo (declare) nhưng không có phần thân (không được implement).

Phương thức trừu tượng được bổ sung (implement) ở các lớp con.

**UML**

Tên lớp abstract và phương thức abstract được in nghiêng

# protected

+ public

- private

**Các tính chất của abstract class**

- Không thể tạo đối tượng của lớp abstract

- Lớp abstract có thể có các thuộc tính và phương thức bình thường

- Lớp chứa phương thức abstract thì lớp đó là abstract

- Lớp con không phải abstract kế thừa lớp abstract phải implement tất cả phương thức abstract của lớp cha

- Lớp abstract không thể là final

- Phương thức abstract không thể là final

**Object Oriented Design**

- **Cohesion (tính gắn kết):**

+ Mỗi lớp chỉ nên đại diện cho 1 thực thể nhất định.

+ Tất cả phương thức của lớp cần phối hợp để hỗ trợ cho tính chất cohesion.

**- Consistency (tính đồng nhất):**

+ Tuân th các tiêu chuẩn của Java và các qui ước đặt tên.

+ Chọn tên phù hợp cho lớp, thuộc tính phù hợp cho phương thức.

+ Cấu trúc 1 lớp: các trường DL, các contructor, các phương thức.

+ Nên định nghĩa 1 contructor không có tham số cho lớp.

**- Encapsulation (tính đóng gói):** dùng private cho trường DL, setter/ getter.

-  **Clarity (tính rõ ràng):**

+ Có nghĩa là nhiệm vụ của các lớp,của các phương thức cần phải dễ hiểu,dễ giải thích

+ Các lớp, các phương thức có thể được sử dụng kết hợp với nhau theo nhiều cách khác nhau, do đó sự rõ ràng là cần thiết

+ Các thuộc tính trong một lớp nên độc lập với nhau, tránh thừa dữ liệu.

**- Interface (kế thừa) or Aggregation (tập hợp).**

[is-a] [has-a]

**- Interface or Abstract class**

[is-a] [can-do]

**Interface**

- Là 1 cấu trúc tương tự lớp, nhưng **CHỈ CHỨA CÁC HẰNG SỐ VÀ ABSTRACT METHOD**.

- Interface quy định các hành vi chung cho các lớp triển khai nó.

**- Cú pháp:**

modifier interface interfaceName{

Khai báo hằng số;

Abstract medthod;

}

**- Tính chất:**

+ Định nghĩa 1 interface là tạo ra 1 kiểu DL mới.

+ Không thể tạo đối tượng interface ==> vì interface là trừu tượng

+ Interface KHÔNG THỂ CHỨA các phương thức KHÔNG ABSTRACT (k chứa các phương thứ có phần thân).

+ Khi 1 lớp triển khai interface thì cần triển khai tất cả phương thức được khai báo trong interface đó.

+ Interface có thể thiết kế để khai báo các phương thức chung cho các lớp KHÔNG LIÊN QUAN với nhau KHÁC VỚI abstract class, được kế thừa bởi các lớp có liên quan với nhau).

+ Interface bổ sung cho việc Java không hỗ trợ đa kế thừa.

class ClassName implement interfaceName{

}

Ví dụ:

public interface Flyable{

String fly();

}

pubclic class Bird implement Flyable{

{

@Override

public String fly(){

return "Flying with wings";

}

**- Kế thừa interface**

+ Một interface có thể kế thừa interface khác.

=> thừa hưởng các phương thức và hằng số từ interface cha

+ Interface con có thể khai báo thêm thành phần mới

+ Từ khoá extends để kế thừa interface

public interface AnimalFlyable extends Flyable{

}

public interface EngineFlyable extends Flyable{

}

**- Access modifier:**

+ Các phương thức abstract của interface có AM mặc định là public

+ Không thể sử dụng private/ protected cho các phương thức của interface

+ Không cần thiết phải chỉ rõ AM là public cho phương thức của interface

public interface Flyable{

public String fly(); //public ==> không cần thiết

}

**- Khai báo hằng số trong interface**

+ Không cần thiết phải chỉ rõ AM cho hằng số, mặc định là public

+ Không cần thiết phải ghi từ khoá final

public interface Flyable{

int ORIENTATION\_LEFT = 1;

int ORIENTATION\_RIGHT = 2;

int ORIENTATION\_UP =3;

int ORIENTATION\_DOWN = 4;

}

**- Anonymous class:**

+ Là lớp đặc biệt được khai báo và khởi tạo đối tượng tại cùng 1 thời điểm

+ Anonymous class ần kế thừa 1 lớp hoặc triển khải 1 interface

public interface Flyable{

String fly();

}

public static void main(String[] args) {

Flyable flyableObj = new Flyable();

@Override

public String fly(){

return "Flying...";

}

System.out.println(flyableObj.fly());

}