OOP

# Chương 1.

Từ những thập kỷ 1970, 1980 phương pháp phát triển phần mềm chủ yếu là lập trình có cấu trúc. Cách tiếp cận cấu trúc đối với việc thiết kế chương trình dựa trên chiến lược chia để trị ( chia bài toán lớn thành những bài toán nhỏ để giải quyết ) khi đó có các vấn đề xảy ra :

1. Không quan tâm đủ đến dữ liệu mà chương trình cần xử lý
2. Khó có thể tái sử dụng các phần của chương trình này cho các chương trình khác
3. Môi trường hoạt động trong thực tế của các ứng dụng luôn thay đổi.

Khi đó dẫn đến việc thiết kế và lập trình từ trên xuống(top-down programming) luôn được kết hợp với lập trình từ dưới lên (bottom-up) hướng đến giải pháp cho 1 bài toán tổng

Các thành phần tái sử dụng được nên có tính modul-hóa cao nhất có thể. Mỗi modul là một thành phần của hệ thống lớn hơn, nó giao tiếp với hệ thống theo 1 cách đơn giản và quy ước chặt chẽ. Những gì modul làm không được xét đến đối với hệ thống miễn là modul làm đúng vai trò của nó. Đến đây được gọi là che giấu thông tin (information hiding).

Những modul hỗ trợ dạng information hiding này dần trở nên phổ biến những năm 1980 nên dần dần cách tiếp cận đó sinh ra khái niệm gọi là lập trình hướng đối tượng (object-oriented programming) gọi tắt là OOP

Đối tượng và lớp : Mỗi lớp là đặc tả các đặc điểm của đối tượng thuộc lớp đó.

Các nguyên tứac trụ cột : có 3 nguyên tắc trụ cột là đóng gói, kế thừa và đa hình còn trừu tượng hóa là khái niệm nền tảng

* Trừu tượng hóa (abstraction) : là một cơ chế cho phép biểu diễn một tình huống phức tạp trong thế giới thực bằng một mô hình đơn giản hóa
* Đóng gói (encapsulation): các trừu tượng hóa của những gì có liên quan đến nhau được đóng gói vào trong một đơn vị duy nhất. Các trạng thái và hành vi của các trừu tượng hóa đươc bọc lại trong một khối gọi là lớp
* Thuộc tính: dùng để lưu trữ thông tin trạng thái của một đối tượng, một thuộc tính có thể đơn giản chỉ là một biến Boolean hay phức tạp hơn khi chính nó lại là một đối tượng khác.
* Các thuộc tính được khai báo trong định nghĩa lớp được goi là biến của thực thể (instance variable). Chúng còn được gọi là các thành viên dữ liệu (data member) hay trường (filer)
* Trạng thái (state) : phản ánh là các giá trị hiện tại của các thuộc tính của một đối tượng và là kết quả theo thời gian của đối tượng đó
* Hành vi (behavior) : là hoạt động của một đối tượng mà có thể được nhìn thấy từ bên ngoài. Trong đó
* Phương thức (method): là một thao tác hay dịch vụ được thực hiện đối với đối tượng khi nó nhận thông điệp tương ứng . Phương thức (Method) còn được gọi là bằng các tên như sau : hàm thành viên (function) – gọi tắt là hàm , thao tác (operation), dịch vụ (service)
* Thừa kế (inheritance) : là một quan hệ phân cấp mà trong đó các thành viên của một lớp được kế thừa bởi các lớp được dẫn xuất trực tiếp hoặc gián tiếp từ lớp đó
* Đa hình (polymorphism) : là khả năng tồn tại trong nhiều hình thức đa hình đi kèm với quan hệ thừa kế và nó có nghhĩa là cùng một cái tên có thể hiểu theo nhiều cách khác nhau tùy từng tình huống

# Chương 2.

Ngôn ngữ java:

* Java được thiết kế là ngôn ngữ hướng đối tượng
* Lỗi trong java không gây sự cố hệ thống (system crash). Một số đặc tính còn phát hiện lỗi tiềm tang trước khi chương trình chạy.
* Java có tính độc lập nền tảng (platform independent)
* Java : Write once, run anywhere (Viết 1 lần, chạy bất cứ đâu) có nghĩa là chương trình java có thể chạy trên nền tảng khác mà không cần dịch lại. Một số ngôn ngữ bậc cao khác không có đặc tính này cho nên java rất phù hợp cho các ứng dụng web
* Java là ngôn ngữ phân tán, an toàn và được thiết kế để dùng cho mạng máy tính
* Java có những đặc tính tự bảo vệ
* Máy ảo java – java Virtual Machine
* Ngôn ngữ máy bao gồm những chỉ thị đơn giản mà CPU máy tính có thể thực hiện trực tiếp. Một chương trình bậc cao càn phải dịch sang ngôn ngữ máy trước khi chạy trên máy tính. Thay vì dùng một trình biên dịch để dịch thẳng toàn bộ chương trình ta có thể dùng một trình thông dịch , nó dịch từng chỉ thị và chỉ đích khi cần đến. Một trình thông dịch là một chương trình hoạt động gần như một CPU với một dạng chu trình nạp-và-thực-thi (fetch-and-execute) tức là lặp đi lặp lại một chuỗi công việc
* Quá trình chạy 1 đoạn code java khi trước : source java ---(biên dịch)-> bytecode –(thông dịch)-> máy
* Quá trình chạy 1 đoạn code java bây giờ : source java –(biên dịch)-> bytecode ---(biên dịch)-> máy

Có 4 nền tảng java :

* Java card dành cho thẻ thông minh(smartcard) và các thiết bị nhớ nhỏ tương tự. thẻ SIM và thẻ Atm có sử dụng nền tảng này
* Java platform, micro edition (Java ME) : dành cho các môi trường hệ thống nhúng, chẳng hạn như điện thoại di động
* Java platform, standard edition (Java SE): là nền tảng tiêu chuẩn thường dùng để dành cho môi trường máy trạm, thường được dùng phát triển java application và java applet. Java SE gồm một máy ảo, một bộ các thư viện cần thiết cho việc sử dụng hệ thống file, mạng, giao diện đồ họa, v.v
* Java platform, enterprise edition (Java EE): dành cho môi trường lớn như doanh nghiệp hoặc internet, thường dùng để phán triển các server. Khác với Java SE ở chỗ có them chức năng và thư viện phần mềm phân tán đa tầng có khả năng chịu lỗi.
* Dịch : trình biên dịch java (javac) lấy file mã nguồn và biên dịch sang dạng bytecode kết quả là file có đuôi .class
* Nạp và chạy : trình nạp java(java) sẽ dùng máy ảo java để chạy chương trình đã dịch sang bytecode

# Chương 3.

Chương trình java là tập hợp tất cả các đối tượng . một đối tượng được tạo ra từ một lớp thì được gọi là một thực thể (instance) của lớp đó(thực thể = đối tượng) có 2 thông tin quan trọng về đối tượng :

Những thông tin mà đối tượng đó biết

Những thông tin mà đối tượng đó làm

Dấu . dùng để truy nhập vào các biến thực thể (instance variable) và gọi phương thức (method) của đối tượng (object)

Phương thức main phục vụ 2 mục tiêu : (1) để thử nghiệm các lớp đã cài (2) để khởi động ứng dụng Java

Khi ở phương thức main ta không thực sự ở trong môi trường hướng đội tương, main chỉ tạo và chạy thử các đối tượng. Trong khi đó, ở một ứng dụng hướng đối tượng thực thụ, các đối tượng phải nói chuyện với nhau .

Một ứng dụng hướng đối tượng nói chung và một ứng dụng Java nói riêng thực chất là các đối tượng đang nói chuyện với nhau. “ Nói chuyện “ ở đây thực chất là các đối tượng (Object) gọi các phương thức của nhau

Tất cả mã Java đều nằm trong 1 lớp nào đó

Một lớp giống như 1 bản thiết kế

Một đối tượng có thể tự lo cho bản than , ta không cần biết đối tượng đó làm việc thế nào

Một đối tượng có thể biết 1 số thứ và làm một số việc

Những gì mà đối tượng đó biết thì gọi là biến thực thể (instance variable) của đối tượng. chúng đại diện cho trạng thái của đối tượng đó

Những gì mà đối tượng đó làm thì gọi là phương thức (method). Chúng đại diện cho hành vi của đối tượng đó

Khi viết một lớp ta có thể muốn viết một lớp khac để test. Tại đó ta tạo các đối tượng của lớp kia để chạy và thử nghiệm

Tại thời gian chạy. một chương trình Java là một nhóm đối tượng đang nói chuyện với nhau

## Chương 4.

Biến thực thể (instance variable ) là trạng thái (state) của đối tượng (object) và biến địa phương là biến được khai báo bên trong một phương thức (method)

Java là ngôn ngữ định kiểu mạnh (strongly-typed language). Nghĩa là, biến nào cũng có kiểu dữ liệu xác định và phải được khai báo trước khi sử dụng.

Các kiểu dữ liệu của java được chia làm 2 loại : Dữ liệu cơ bản (primitive) và tham chiếu đối tượng (object refenrence)

Các dữ liệu cơ bản là như int , float , …

Các tham chiếu đối tượng là các tham chiếu tới đối tượng

Định danh (identifier) là thuật ngữ chỉ tên. Java quy định định danh là một chuỗi kí tự liền nhau

Một biến tham chiếu đối tượng không phải là một đối tượng mà nó chỉ đóng vai trò như 1 con trỏ trỏ tới đối tượng nào đó. Thực chất các đối tượng không có tên và chúng cũng không nằm trong biến nào. Trong Java các đối tượng được tạo ra đều nằm trong bộ nhớ Heap

Phép gán chỉ sao chép chuỗi bit của biến tham chiếu này tới biến tham chiếu kia

# Chương 5.

Nếu một phương thức yêu cầu 1 tham số ta cần truyền cho nó một giá trị nào đó, và giá trị đó phải cùng kiểu khai báo của tham số

Lớp định nghĩa những gì mà một đối tượng biết và nó có thể làm

Những gì mà một đối tượng biết là các biến thực thể của nó (trạng thái của đối tượng)

Những gì mà một đối tượng làm là cá phương thức của nó (hành vi của đối tượng)

Các phương thức có thể sử dụng biến thực thể của đối tượng, nhờ đó mà các đối tượng cùng một lớp có thể có các hành sử không giống nhau

Một phương thức có thể có các tham số , ta có thể truyền vào cac giá trị của phương thsc qua tham số của phương thức

# Chương 6.

Java cũng có dạng vector là ArrayList và khi sử dụng ta cần import java.util.ArrayList (java.ulti là gọi đến gói packet còn ArrayList là lớp) hoặc cũng có thể dùng java.ulti.ArrayList<Cow> = new java.util.ArrayList<Cow> và ta cũng dùng đến các lớp như System, String và Math và nó nằm trong gói java.lang nhưng gói java.lang thuộc dạng đã nạp sẵn nên không cần import

ArrayList có các tiện ích như sau :

1. add(Object item), add(int i, object item) : chèn item vào vị trí I, nếu không có i mặc định là cuối
2. get(int i) trả về đối tượng tại vị trí i
3. remove (int i) xóa đối tượng tại vị trí có chỉ số i
4. remove(object item) xóa đối tượng item nếu nó nằm trong danh sách
5. contains(object item) trả về true nếu item có trong danh sách
6. isEmpty() trả về true nếu danh sách rỗng
7. size() trả về kích thước của danh sách

Có 3 lí do cho việc tổ chức các lớp vào các gói :

1. Gói giúp ích cho việc tổ chức project hay thư viện. Thay cho 1 lô các lớp đặt cùng 1 chỗ, các lớp đươc đặt vào các gói khác nhau tùy vào chức năng
2. Cấu trúc cho ta 1 không gian tên, tránh trùng tên
3. Tổ chức gói cho ta một lớp bảo mật, ta cũng có thể hạn chế các mã ta viết trong 1 gói để chỉ có các lớp nằm trong gói đó mới có thể truy cập

Sử dụng API bằng cách nào ?

1. Trong thư viện có những lớp nào
2. Khi tìm thấy 1 lớp làm thế nào để ta biết nó làm gì -> tra sách hoặc tài liệu về java và Api

Có các lớp biểu diễn xâu kí tự : String, StringBuffer/StringBuilder

String.format(“n = %, d”, 1234567) -> 1,234,567

String.format(“n = %, .1f”, 12345.67) -? 12,345.7

String.format(Locale.GERMAN, “n = %, .1f”, 12345.67) -> 12.345,7

1. charAt(int index): trả về giá trị tại vị trí index
2. compareTo() so sánh giá trị với một đối tượng cùng loại
3. indexOf() tìm vị trí của xâu từ trái -> phải
4. lastIndexOf() tìm vị trí của xâu từ phải -> trái
5. lenth() trả về độ dài của xâu
6. substring(int start, int end) trả về đối tượng String là xâu con
7. concat() cho String và append() cho StringBuffer/StringBuilder để nối xâu
8. trim() trả về xâu đã xóa kí tự trắng ở đầu và cuối
9. int n = 3232; String s1 = “” + n; String s2 = Interger.toString(n);