coggle
made for free at coggle.it

Advanced Programming with Java 2.1

Tổng quan Java

Khái niệm ngôn ngữ lập trình Java:
- Là ngôn ngữ hướng đối tượng
- Có khả năng thực thi ở nhiều loại thiết bị
- Được sử dụng rộng rãi
  "Write one, run any where"

Khái niệm máy ảo Java(JVM), JDK, JRE và byte code

Byte code
- Là ngôn ngữ được thực thi trên một máy tính ảo JVM

JDK(Java development)::
- Là bộ công cụ để hỗ trợ lập trinhd viên phát triển ứng dụng Java(java.exe, javac.exe)
- Giúp phát triển và biên dịch mã Java thành chương trình Java

JRE(Java Runtime Envionment)
- Là môi trường thực thi ứng dụng
- Giúp thực thi các ứng dụng trong JVM
- Cũng được kèm theo trong JDK

JVM(Java Virtual Machine)
- Máy ảo Java thực thi mã Java byte code(.class) ra ngôn ngữ máy

Luồng thực thi chương trình

Thành thạo cú pháp của ngôn ngữ Java

- Sử dụng được câu lệnh break trong Java
- Sử dụng được biến trong Java
- Sửa dụng được kiểu dữ liệu trong Java
- Sử dụng được toán tử trong Java
- Sử dụng được câu lệnh if - else trong Java
- Sử dụng được câu lệnh switch-case trong Java
- Sử dụng được câu lệnh for trong Java
- Sử dụng được câu lệnh while trong Java
- 8 Sử dụng được câu lệnh do-while trong Java

Khai niệm máy ảo Java(JVM), JDK, JRE và byte code

- Có 8 kiểu dữ liệu nguyên thủy(primitive datatype
- Kiểu đối lượng : String (or any object)

Khi khai báo một biến của đối tượng (không phải biến địa phương) mà không gán giá trị của nó thì nó sẽ có giá trị mặc định

Kiểu đối tượng
- String
- Object

null

Khi biến có giá tị Null thì không thể truy xuất bất cứ gì

- 0 — byte : Kiểu số nguyên có kích thước 1 byte. Các giá trị nằm trong khoảng -128 đến 127
- 0 — short: Kiểu số nguyên có kích thước 2 byte. Các giá trị nằm trong khoảng -32768 đến 32767.
- 0 — int : Kiểu số nguyên có kích thước 4 byte Các giá trị nằm trong khoảng -2^31 đến 2^31-1.
- 0L — long : Kiểu số nguyên có kích thước 8 byte. Các giá trị nằm trong khoảng -2^63 đến 2^63-1
- 0.0f — float : Kiểu số thực có kích thước 4 byte.
- 0.0d — double : Kiểu số thực có kích thước 8 byte.
- false — boolean: Bao gồm 2 giá trị là true và false.
- '\u0000' — char : Kiểu kí tự Unicode có kích thước 2 byte. Có giá trị nhỏ nhất là '\u0000'(tương đương với 0) và giá trị lớn nhất là '\uffff'(tương đương với 65535)

Mảng

Khái niệm:
- Là tập hợp các phần tử có cùng kiểu dữ liệu được lưu giữ gần nhau trong ô nhớ

Cách khai báo
- element Type [ ] arrayName
- elementType arrayName[ ]

Cách khởi tạo

Khởi tạo dùng từ khóa 'new'
- elementType[ ] arrayName = new elementType[array size]

Khởi tạo nhanh
- elementType[ ] arrayName = {element1, element 2, ...}

OOP (Object Oriented Programing)

Khái niệm:
Là kĩ thuật lập trình có nhiệm vụ ánh xja các đối tượng có trong thực giưới thực vào trong các đối tượng lập trình

Cleancode
- Dễ đọc
- Đơn giản: giải quyết vấn đề ngắn gọn không phức tạp hóa
- Dễ cải tiến mở rộng: có thể cải tiến mở rộng mà không gây nhiều lộn xộn cho mã nguồn
- Tên class, method, biến thể hiện rõ ý nghĩa
  _ Ít phụ thuộc
- Không trùng lặp
- Đúng ý tưởng thiết kế

Enscapsulation (Tính bao đóng)

Là kĩ thuật dùng để ẩn dấu thông tin của đối tượng, chỉ hiển thị nhữ thông tin cần thiết

- Mục đích:
- Bảo vệ trạng thái bên trong đối tượng
- Giảm độ phức tạp của chương trình.

-Để đạt được tính bao đóng thì dùng :
- Access Modifier,
- Getter/Setter

Stactic:

Khái niệm:
- Là từ khóa dùng để khai báo thuộc tính và phương thức, class. Khi đó thuộc tính và phương thức được khai báo sẽ chịu sự quản lý của Class- không phải của đối tượn.
- Truy cập thành phần static(thuộc tính, phương thức) thông qua tên Class(nên dùng) hoặc có thể thông qua đối tượng.
  -> Có thể truy xuất các thành phần static mà không cần khởi tạo đối tượng.

Mục đích sử dụng
- Đinh nghĩ thuộc tính, phương thức dùng chung cho toàn bộ đối tượng.
- Tạo ra các lớp tiện ích

Cácràng buộc
- Phương thước static chỉ có thể gọi được phương thức Static khác
- Phương thức static chỉ có thể truy cập được biến static
- Dùng khối **Static { }** để khởi tạo giá trị cho biến static

Inheritance (TÍnh kế thừa)

Là cơ chế cho phép lướp con sử dụng lại các đặc điểm và hành vi đã định nghĩa ở lớp cha

Mục đích:
- Tái sử dụng được source code

- Mối quan hệ của lớp cha con là **is - a**
- Lớp con chỉ được phép sở hữu tài sản (thuộc tính method) public, default(nếu lớp cha và con cùng package),protected.
- Lớp con không thể truy cập vào tài sản private của lớp cha
- Lớp con không thể kế thừa constructor của lớp cha
- Java không hỗ trợ đã kế thừa.

Polymorphism (Đa hình)

Là khả năng một đối tượng có thể thể hiện hành vi theo nhiều cách khác nhau tùy vào ngữ cảnh

Overiding (Ghi đè)
- Là cơ chế cho phép lớp con định nghĩa lại nội dung các phương thức đã có trước đó của lớp cha
- Phương thức overiding phải cùng tên , kiểu dữ liệu trả về, danh sách tham số
- Access Modifier của lớp con phải cùng kiểu hoặc rộng hơn so với ở lớp cha
- Xảy ra mối quan hệ kế thừa **(is -a)**

Over load(Nạp chồng)
- Là cơ chế cho phép một đinh nghĩa ra nhiều phương thức cùng tên nhưng khác tham số truyền vào

Abstruction(Trừu tượng)

Là khả năng ẩn đi chi tiết triển khai chỉ hiển thị tính năng / kết quả cho người dùng -> Chỉ quan tâm đến kết quả đạt được, không quan tâm các thực hiện **(implement)**
- Trong java tính trừu tượng được thể hiện qua **abstract** và **interface**

Abstract class
Là 1 class có tính trừu tượng cao đến mức không thể tạo mới đối tượng

- Không thể tạo mới đối tượng
- Một lớp chứa abstract method thì lớp đó phải là abstract(ngược lại thì không)
- Không được dùng từ khóa final cho abstract class
- Lớp abstract có thể kế thừa từ lớp abstract khác(không cần phải overidng) của lớp cha
- AccessModifier không hạn chế

Inter face
Là một bản thiết kế cung cấp các hành vi bổ sung cho lớp triển khai nó(dùng từ khóa **Implement**)

- Không khởi tạo đối tượng
  - Method toàn bộ là absstract
- Không có constructor
- Đại diện cho mối quan hệ has - a
- Có thể extends từ nhiều interface khác nhau
- Thuộc tính mặc định là final, phương thức phần thân không chứa code
- AccessModifier chỉ dùng public

Object:
- Là thể hiện cụ thể của class
- Dùng từ khóa **new** để khởi tạo đối tượng
- Đối tượng có thể khởi tạo nhiều lần

Class :
- Là khuôn mẫu định nghĩa ra các thuộc tính và hành vi chung cho tất cả các đối tượng cùng loại
- Dùng từ khóa Class để khai báo
- Class chỉ khai báo 1 lần (không trùng) trong cùng pham vi Package

Access Modifier
- private : trong class
- default: trong class, trong package
- protected; trong class trong package, ngoài package nếu class có mối quan hệ cha con
- public: trong class, trong ngoài package, t(toàn bộ package)

Constructor:
*Là phương thức dùng để khởi tạo đối tượng
- Tên Constructor : phải trùng tên với clas
- Không có kiểu trả về
- Trong 1 class có thể có nhiều Constructor
- Mặc định nếu không khai báo Consstructor thì Java sẽ cung cấp 1 Constructor mặc định không có tham số.
- Có thể gọi Constructor khác thông qua từ khóa **this** và **lưu ý** : this(.)phải gọi đầu tiên trong nội dung phương thức khởi tạo

Method:
- Có thể có kiểu trả về hoặc không
- Có thể đặt tên giống hoặc khác tên Clas
- Java không cung cấp Method mặc định
- Có thể được gọi nhiều lần

Các 3 loại biến:

Biến Local::
- Biến khai báo bên tổng Method hoặc block
- Biến Local được tạo ra khi mà phương thức, constructor , block được gọi và bị hủy khi Method, constructor thực hiện xong
- Không dùng được Access modifier
- Cần khởi tạo giá trị mặc định khi sử dụng.

Biến toàn cục:
- Biến khai báo trong lớp (bên ngoài constructor, method, block
- Biến được khởi tạo khi khởi tạo đối tượng
  và bị hủy khi đối tượng bị hủy
- Có thể sử dụng Access Modifier

Biến static:
- Khai báo bằng từ khóa Static
- Biến static được gọi khi chương trình bắt đầu chạy và bị hủy khi chương trình dừng