coggle
made for free at coggle.it

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

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

- null
- **Kiểu dữ liệu**
  - String
  - Object

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 trình 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 Environment)
- 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

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

Luồng thực thi chương trình

- 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. Các giá trị nằm trong khoảng '\u0000'(tương đương với 0) đến '\uffff'(tương đương với 65535)

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

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"**

IO text File

Stream
- Khái niệm:
  +Là các hoạt động nhập/xuất dữ liệu được gọi là stream (dòng/luồng)

Có hai dạng stream

Character stream được thiết kế để hỗ trợ nhập xuất dữ liệu theo dạng kí tự code
- Gồm các lớp triển khai sau:

Byte stream hỗ trợ nhập xuất dữ liệu theo dạng byte

Absstract class
- Reader
- writer

Class triển khai hay dùng
- File reader
- File writer

Class tạo bộ nhớ đệm
- Buffered Reader
- Buffered Writer

Absstract class
- Inputstream
- Outpustream

Class hay dùng
- File Inputstream
- File Outputstream

Class đọc ghi Object
- Object Inputstream
- Object Output Stream

Class tạo bộ nhớ đệm
- Buffered Inputstream
- Buffered outputstream

Exception
(Ngoại lệ)
Là các lỗi thường gặp trong quá trình thực thi, nó phá vỡ luồng xử lý bình thường của chương trình thậm chí gây chết chương trình

Error
- Là những lỗi nghiêm trọng liên qua đến môi trường thực thi JVM của ứng dụng hệ thống mà lập trình viên không thể xử lý nó, có thể gây chết chương trình

Checked Exception
- Là loại Exception xảy ra trong quá trình Compile time không thể bỏ qua trong lúc Compile. Lập trình viên phải handle nó
- Ví dụ :
  +IO Exception
  +FileNotFound Exception
  +Database Exception

UnCheck Exception
- Xảy ra tại thời điểm thực thi chương trình lỗi này không chắc chắn xảy ra có thể bỏ qua trong quá trình Compile lập trình viên không bắt buộc phải **handdle**
- Ví dụ
  +Những lớp con của Runtime Exception, NullPointerExcception, Number Format Exception.

Cách xử lý ngoại lệ

Lưu ý
- Khối try -có thể không có khối catch hoặc có nhiều khối catch nhưng duy chỉ có 1 khối finnaly
- Tại mỗi thời điểm thì chỉ có duy nhất một khối catch được thực thi
- Khối catch phải được sắp xếp từ cụ thể 1 loại Exception đến loại chung nhất(tức exception con đến exception cha)
- Khối finally luôn được thực thi cho dùng chương trình có xảy ra ngoại lệ hay không(Một vài trường hợp khối Finally không được thực thi do người dùng có tình can thiệp làm dừng chương trình trước khi bước finally chạy)
- Nếu trong 1 hàm có dùng từ khóa return thì khối finally ưu tiên chạy trước sau đó mới chạy dòng lệnh return

Try catch (finally
- Xử lý ngoại lệ ngay tại đoạn mã bị lỗi

Dùng throw/throws để ném ra ngoại lệ cho người khác xử lý

Advanced Programming with Java 2.1

Collection Framework Hierarchy in Java

Collection

Map
- **Hash Map**
- **LinkedHashMap**
- **TreeMap**

Set
- **HashSet**
- **LinkedHashSet**
- **TreeSet**

List
- **ArrayList**
- **LinkedList**

Linked List
- là 1 dạng List hoạt động dựa trên cơ sở cấu trúc dữ liệu danh sách liên kết

- Sử dụng danh sách liên kết để lưu trữ phần tử
- Truy xuất chậm
- Thêm xóa nhanh

Thao tác cơ bản :
- add()
- get()
-remove()
- ...

Stack

Khái niệm
- Stack là một cấu trúc dữ liệu dạng ngăn xếp
- Các phần tử của stack được hoạt động theo nguyên lý phần tử nào vào trước thì được xếp dưới cùng lấy phần tử trên cùng ra trước

Các phương thức

- Một ngăn xếp là một cấu trúc dữ liệu dạng thùng chứa (container) của các phần tử (thường được gọi là các Node). Có hai thao tác cơ bản là push và pop.
  - **Push** bổ sung một phần tử vào đỉnh (top) của ngăn xếp, nghĩa là nó sẽ được thêm vào sau các phần tử đã có sẵn trong ngăn xếp.
- **Pop** giải phóng và trả về phần tử đang đứng ở đỉnh của ngăn xếp. Phần tử sau khi được lấy sẽ bị xóa khỏi ngăn xếp.
- **isEmpty()**: Kiểm tra xem stack có rỗng hay không.
- **Top()**: Trả về giá trị của phần tử nằm ở đầu Stack mà không hủy nó khỏi Stack. Nếu Stack rỗng thì thông báo và không thực hiện được thao tác này.

PriorityQueue

Queue

Deque

ArrayDeque

Khái niệm:
- Là 1 dạng list của List có kích thước mảng có thể thay đổi được
  Đặc điểm:
- Có kiểu lưu trữ dữ liệu là đối tượn, các tham số kiểu nguyên thủy sẽ được tự động chuyển thành Object nhờ cơ chế auto boxing
- Có thể lưu 2 phần tử giống nhau, dữ liệu có thể là null
- Có nhiều phương thức hỗ trợ cho việc xử lý dữ liệu
- Tốc độ xử lý chậm hơn array.

Các phương thức hỗ trợ cho Array List

- arraylist.add(int index, E element)

arraylist.addAll(int index, Collection c)
- arraylist - một đối tượng của lớp ArrayList.
- index (option) - chỉ mục mà tại đó tất cả các phần tử của một tập hợp được chèn vào.
- Collection - tập có chứa các phần tử sẽ được chèn vào.

arraylist.clear()

arraylist.clone()
- Ví dụ: chúng ta có một mảng arr = [1, 3, 5, 6], bây giờ copy một mảng mới arrNew với các phần tử giống hệt của mảng arr.:
  arrNew = arr.clone();

arraylist.contains(Object obj)
- arraylist - một đối tượng của lớp ArrayList.
- obj - phần tử được kiểm tra.

arraylist.get(int index)

arraylist.indexOf(Object obj)
-Nếu cùng một phần tử obj xuất hiện ở nhiều vị trí, thì vị trí của phần tử xuất hiện đầu tiên trong danh sách mảng được trả về.

arraylist.removeAll(Collection c)
- arraylist là một đối tượng của lớp ArrayList.
- collection - tất cả các phần tử có trong tập sẽ bị xóa khỏi danh sách mảng.
- Ví dụ: chúng ta có mảng arr = [1, 4, 5, 7, 8], bây giờ xóa tập các phần tử arr2 = [4, 5] khỏi mảng arr.

arraylist.size()

arraylist.isEmpty()
- Trả về true nếu danh sách mảng không chứa bất kỳ phần tử nào.
- Trả về false nếu danh sách mảng chứa một số phần tử.

arraylist.subList(int fromIndex, int toIndex)
- Kết quả trả về phần tử từ fromIndex đến toIndex -1

set()
Về cú pháp thì phương thức add() và phương thức set() khá giống nhau. Tuy nhiên phương thức set() **thêm mới** phần tử và **thay thế** phần tử mới đó vào vị trí Index thêm vào

arraylist.sort(Comparator c)
- arrlist là đối tượng của class ArratList.
- Comparator - phương thức interface chỉ định tiêu chí để sắp xếp

So sánh Array và Array List
- Kích thước lưu trữ mảng
- Kiểu lưu trữ
- Phương thức, thuộc tính
- Tốc độ lưu trữ

Array
- Kích thước lưu trữ mảng: Cố định
- Kiểu lưu trữ: Nguyên thủy, đối tượng
- Phương thức, thuộc tính: Chỉ có thuộc tính length
- Tốc độ lưu trữ: Nhanh hơn Array List

Array List
- Thay đổi được
- Đối tượng, kiểu nguyên thủy tự động chuyển đổi thành dạng Object nhờ cơ chế auto boxing
- Phương thức, thuộc tính: Có hỗ trợ nhiều phương thức
- Tốc độ lưu trữ: Chậm hơn Array

Thuật toán sắp xếp(Sorting Algorithms):

Bubble sort(Sắp xếp nổi bọt

Ý tưởng:
- So sánh liên kề và trao đổi thứ tự nếu chúng không đảm bảo được sắp xếp tăng dần
- Độ phức tạp:
- Tối ưu: O(n)
  +Tệ nhất: O(n^2)

Selection Sort(Sắp xếp lựa chọn)

Ý tưởng
- Chọn phần tử lớn nhất/ nhỏ nhất đưa lên đầu dãy lựa sắp xếp và đồng thời loại đối tượng đó ra khỏi dãy sắp xếp tiếp theo
- Sau khi đã loại bỏ phần tử lựa chọn trước đó ra khỏi danh sách tiếp tục danh sách sắp xếp mới là k-1. Tiếp tục thực hiện so sánh sắp xếp đưa lên đầu như bước trước.
- Quá trình này lặp lại cho đến khi dãy sắp xếp còn 1 phần tử
- Độ phức tạp là:
- Tối ưu: O(n^2)
- Tệ nhất: O(n^2)

Insertion Sort (sắp xếp chèn)

Ý tưởng
- Một danh sách con luôn luôn được duy trì dưới dạng đã qua sắp xếp .
- Chèn phần tử vào đầy đủ được sắp xếp. Phần tử được chèn sao cho vẫn đảm bảo theo trình tự của dãy con đã sắp xếp. Bằng cách so sánh phần tử chèn với các phần tử đã chèn vào trước đó

Thuật toán tìm kiếm

Linear Search(Thuật toán tìm kiếm tuyến tính)

Ý tưởng:
- Một phần tử đều được duyệt để kiểm tra, nếu tìm thấy phần tử cần tìm thì trả về vị trí index của phần tử đó
- Độ phức tạp:
- Tối ưu O(1)
- Tệ nhất O(n)
- Áp dụng để duyệt các bài toán chưa được sắp xếp, số lượng phần tử nhỏ.

Tìm kiếm nhị phân(Binary Search):

Ý tưởng:
1 So sánh phần tử cần tìm với phần tử giữa
2 Nếu tìm thấy thì trả về Index
3 Nếu phần tử lớn hơn phần tử giữa thì phần tử cần tìm sẽ nằm ở mảng con phải của phần tử giữa
4 Nếu phần tử cần tìm nhỏ hơn với phần tử giữa thì phần tử cần tìm nằm ở mảng con trái của phần tử giữa.
5 Lặp lại bước 1
- Độ phức tạp:
- Tối ưu O(1)
- Tệ nhất O(log n)

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

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

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(nếu 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
- Định 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 cânh

Overding (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 overding phải cùng tên , kiểu dữ liệu trả về, tham số truyền vào
- 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 định nghĩa ra nhiều phương thức cùng tên nhưng khác tham số truyền vào

Abstraction(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 overding) 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 nội(dùng từ khóa **Implement**)

- Không khởi tạo đối tượng
  - Method toàn bộ là abstract
- 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

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, tị toàn bộ package)

Constructor:
*Là phương thức dùng để khởi tạo đối tượng
- Tên Constructor : phải trùng tên với class
- Không có kiểu trả về
- Trong 1 class có thể có nhiều Constructor
- Mặc định nếu không khai báo Constructor 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

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 phạm vi Package

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 tông 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