**Write once, run anywhere**

Là khi viết mã bằng java, mã nguồn được biên dịch thành bytecode (ngôn ngữ trung gian dễ dàng được chuyển đổi), sẽ được chuyển đổi thành mã máy bởi JVM.

**JDK**

Là viết tắt của Java Development Kit. Đây là một công cụ phát triển java, bao gồm JRE và các công cụ phát triển khác (như trình biên dịch Java, các công cụ để tạo và quản lý file, và các công cụ khác để phát triển và thử nghiệm ứng dụng Java).

**JRE**

Là viết tắt của Java Runtime Environment. Đây là một môi trường thực thi cho các ứng dụng Java, bao gồm JVM (Java Virtual Machine) và các thư viện cần thiết để chạy các ứng dụng Java.

**Compile**

Là quá trình chuyển đổi mã nguồn Java có đuôi .java sang mã bytecode có đuôi .class bởi trình biên dịch Java

**Runtime**

Là một lớp trong Java cung cấp một số phương thức để tương tác với môi trường thực thi của hệ thống. Được sử dụng để lấy thông tin về hệ thống và thực hiện các hoạt động cơ bản như tạo tiến trình, tắt hệ thống, lấy thông tin tài nguyên hệ thống,…

Một số phương thức quan trọng trong lớp Runtime là:

* **availableProcessors()**: trả về số lượng bộ xử lý có sẵn trên hệ thống.
* **freeMemory()**: trả về số lượng bộ nhớ không sử dụng.
* **totalMemory()**: trả về tổng số lượng bộ nhớ có sẵn trên hệ thống.
* **maxMemory()**: trả về số lượng bộ nhớ tối đa có thể được sử dụng bởi JVM.
* **exec(String command)**: thực thi một lệnh được chỉ định trên hệ thống.
* **halt(int status)**: tắt hệ thống với mã trạng thái được chỉ định.

**Giá trị mặc đinh của kiểu dữ liệu nguyên thủy.**Boolean: false

Byte: 0

Short: 0

Int: 0

Long: 0L

Float: 0.0f

Double: 0.0d

Char: ‘\u0000’

**Các vòng lặp(for, while, do while)(đặc điểm)**

Vòng lặp for sẽ biết trước số lần lặp.

Vòng lặp while sẽ không biết trước số lần lặp.

Vòng lặp do-while sẽ lặp ít nhất 1 lần, rồi mới kiểm tra điều kiện.

**passByReference & passByValue**

Trong tham trị, giá trị của biến được truyền vào phương thức được sao chép và truyền vào phương thức. Bất kì thay đổi nào trong phương thức thì không ảnh hưởng đến giá trị ban đầu.

Trong tham chiếu, tham chiếu đến vùng nhớ của biến được truyền vào phương thức được truyền vào thay vì giá trị thực tế của biến đó. Bất kì thay đổi nào được thực hiện trong phương thức đối với biến truyền vào đều ảnh hưởng đến giá trị của biến ban đầu.

# Khác nhau Biến tham trị và Biến tham chiếu?

* Biến tham trị là biến chứa giá trị trong vùng nhớ được cấp.

Biến tham chiếu là biến chứa tham chiếu đến đối tượng trong vùng nhớ được cấp.

**Array (reference || value ? ) chứng minh?**

Mảng là một kiểu dữ liệu tham chiếu, không phải kiểu dữ liệu giá trị

Khi khai báo một mảng trong Java, một biến tham chiếu được tạo ra để trỏ đến vùng nhớ lưu trữ các phần tử của mảng. Biến tham chiếu này chứa địa chỉ bộ nhớ của vùng nhớ chứa các phần tử của mảng. Khi truy cập đến phần tử của mảng, là đang sử dụng biến tham chiếu đấy để truy cập đến phần tử của mảng tại địa chỉ bộ nhớ tương ứng.

**OOP là gì**

OOP là viết tắt của Object-Oriented Programming (Lập trình hướng đối tượng). là một phương pháp lập trình sử dụng đối tượng để mô hình hóa thế giới thực.

Trong OOP có 4 khái niệm:

*Tính kế thừa: cho phép một lớp con kế thừa các thuộc tính và phương thức của một lớp cha và mở rộng chúng hoặc ghi đè lên chúng.*

*Tính đa hình: cho phép các đối tượng của các lớp khác nhau có thể được xử lý bằng cách sử dụng cùng 1 phương thức với cùng 1 tên, nhưng có thể thực hiện khác nhau tùy thuộc vào lớp của đối tượng đó.*

*Tính đóng gói: cho phép ẩn giấu thông tin bên trong các đối tượng và chỉ cho phép truy cập thông qua các asscess modifer.*

*Tính trừu tượng: cho phép ta tập trung vào các khía cạnh quan trọng của đối tượng mà không cần quan tâm đến các chi tiết bên trong.*

**Class & quy tắc đặt tên**

Trong Java, một lớp là một bản thiết kế mô tả về các đối tượng sẽ được tạo ra

Quy tắc đặt tên cho lớp:

Viết hoa chữ cái đầu và không bắt đầu bằng số

Quy tắc đặt tên cho các thành phần của lớp:

Đặt tên theo quy tắc Camelcase

# Sự khác nhau giữa bộ nhớ heap và stack trong java?

Heap và Stack đều là vùng nhớ được tạo ra và lưu trữ trong RAM khi chương trình được thực thi.

* Stack: dung lượng nhỏ hơn, dùng để lưu các tham số và các biến local của phương thức mỗi khi một phương thức được gọi ra.
* Heap: dùng để lưu các đối tượng khi từ khóa new được gọi, các biến static và các biến toàn cục (biến instance).
* **Biến instance** được khai báo trong một lớp(class), bên ngoài các phương thức, constructor và các block.
* **Biến static** được khai báo trong một class với từ khóa "static", phía bên ngoài các phương thức, constructor và block.

# Chuyện gì xảy ra nếu khai báo static public void thay vì public static void?

Vị trí của modifier như public hay static không quan trọng đối với hàm main hay bất cứ hàm nào khác trong Java, nên static public void main cũng vẫn được.

Tuy vậy thì returnType (kiểu dữ liệu trả về) bắt buộc phải đi sau các modifier.

# Giá trị mặc định của các biến local là gì?

Các biến local không được khởi tạo với bất kỳ giá trị mặc định nào, bất kể là nguyên thủy hay tham chiếu đối tượng.

# Định nghĩa Lớp (class)?

* Là đơn vị thực thi cơ bản trong Java.
* Quy định thuộc tính và hành vi của các đối tượng => là khuôn mẫu để tạo nên các đối tượng.
* Khai báo lớp ==> sẽ khai báo 1 KIỂU DỮ LIỆU MỚI.

# Kiểu biến nào một lớp có thể chứa?

Một lớp có thể gồm biến local, biến instance (biến của đối tượng), và biến static (biến lớp).

# Biến local hay biến cục bộ là gì?

- Biến local được khai báo trong các phương thức, hàm contructor hoặc trong các block.

- Sẽ bị phá hủy khi kết thúc các phương thức, contructor và block.

- Không được sử dụng "access modifier" khi khai báo biến local.

- Các biến local được lưu trên vùng nhớ stack của bộ nhớ.

- Cần khởi tạo giá trị mặc định cho biến local trước khi có thể sử dụng.

# Biến instance là gì?

- Biến instance được khai báo trong một lớp (class), bên ngoài các phương thức, constructor và các block.

- Biến instance được tạo khi một đối tượng được tạo bằng việc sử dụng từ khóa “new” và sẽ bị phá hủy khi đối tượng bị phá hủy.

- Biến instance có thể được sử dụng bởi các phương thức, constructor, block, ... Nhưng nó phải được sử dụng thông qua một đối tượng cụ thể

- Bạn được phép sử dụng "access modifier" khi khai báo biến instance, mặc định là "default".

- Biến instance có giá trị mặc định phụ thuộc vào kiểu dữ liệu của nó. Ví dụ nếu là kiểu int, short, byte thì giá trị mặc định là 0, kiểu double thì là 0.0d, ...

# Constructor là gì?

* Là phương thức giúp khởi tạo đối tượng.
* Có TÊN TRÙNG VỚI TÊN LỚP.
* Không có kiểu dữ liệu trả về.
* Một lớp có thể có nhiều Constructor, trùng tên nhưng khác tham số.
* Nếu không khai báo thì mặc định lớp có 1 Constructor không có tham số.

# Mục đích của constructor là gì?

* Phương thức khởi tạo ==> để khởi tạo các đối tượng của lớp, cung cấp các giá trị mặc định cho các đối tượng.

# Tại sao phương thức main là static?

Bởi vì không cần thiết phải tạo đối tượng để gọi phương thức static. Nếu nó là phương thức non-static, JVM đầu tiên tạo đối tượng và sau đó gọi phương thức main() mà có thể gây ra vấn đề về cấp phát bộ nhớ bộ nhớ phụ.

# Sự khác nhau giữa phương thức static và phương thức instance?

|  |  |
| --- | --- |
| **Phương thức static** | **Phương thức instance** |
| Khai báo với từ khóa static. | Không được khai báo với từ khóa static. |
| Không cần tạo đối tượng cũng gọi được phương thức static thông qua class. | Phải tạo đối tượng để gọi phương thức instance. |
| Biến non-static không được truy cập trực tiếp trong phương thức static (hoặc khối static). | Biến static và non-static được truy cập trực tiếp trong phương thức instance. |