**Bài 1: Môi trường Java (JVM, JRE, JDK)**

**JVM (Java Virtual Machine) là 1 phần giúp quản lý mã nguồn và quản lý tài nguyên**

* Quy trình JVM:

+ Load (Biên dịch) mã nguồn Java (.java) sẽ được biên dịch bởi javac thành các mã nguồn .class và sau đó đc mã hóa thành mã

+ Loading Class khi chương trình đc execute thì class này sẽ đi tìm các mã nguồn cần thiết được biên dịch tải vào bộ nhớ

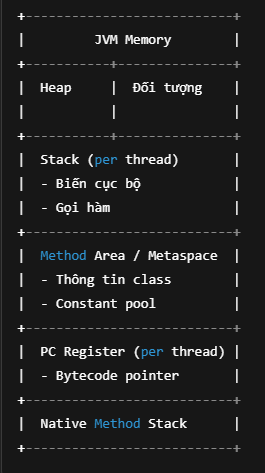
* **Các loại ClassLoader**:
  + **Bootstrap ClassLoader**: Nạp các lớp cốt lõi của Java (như **java.lang.\***).
  + **Extension ClassLoader**: Nạp các lớp từ thư mục mở rộng (extension directories).
  + **Application ClassLoader**: Nạp các lớp từ đường dẫn lớp (classpath) của ứng dụng, bao gồm cả các thư viện và phụ thuộc mà ứng dụng sử dụng.

+ Linking sau khi các mã byte đã được tải về bộ nhớ JVM sẽ kiểm trả và liên kết các mã byte đó

* + 1. **Kiểm tra (Verification)**: Đảm bảo rằng mã byte tuân thủ các quy tắc của Java và không chứa mã độc hại.
    2. **Liên kết (Preparation)**: Cấp phát bộ nhớ cho các biến tĩnh và khởi tạo chúng với giá trị mặc định.
    3. **Giải quyết (Resolution)**: Thay thế các tham chiếu đến các lớp, phương thức và trường bằng địa chỉ thực tế trong bộ nhớ.

1. **Khởi tạo (Initialization)**:
   1. Sau khi liên kết, JVM sẽ khởi tạo các lớp. Quá trình này bao gồm việc thực thi các khối khởi tạo tĩnh (static initializer) và khởi tạo các biến tĩnh.
2. **Thực thi mã byte (Execution)**:
   1. JVM bắt đầu thực thi mã byte. Có hai cách chính để thực hiện điều này:
      1. **Thông dịch (Interpretation)**: JVM đọc và thực thi từng lệnh mã byte một cách tuần tự.
      2. **Biên dịch Just-In-Time (JIT)**: JVM biên dịch một phần mã byte thành mã máy (native code) trong thời gian thực thi, giúp cải thiện hiệu suất. Mã máy sau đó có thể được thực thi trực tiếp bởi CPU.
3. **Quản lý bộ nhớ**:
   1. Trong suốt quá trình thực thi, JVM quản lý bộ nhớ cho các đối tượng được tạo ra. Nó sử dụng Garbage Collection để tự động giải phóng bộ nhớ cho các đối tượng không còn được tham chiếu.

Sơ đồ JVM sơ khai:



Heap: là bộ nhớ để lưu trữ đối tượng khi được new thông tin đối tượng, có một biến tham chiếu nắm giữ địa chỉ thông tin đối tượng trong heap

Ví dụ:

Student stu = new Student(“Phát”, 2004, 7.1);

| | |

V V V

Data param Object được new

Type

Thông tin mỗi lần được new thì sẽ được lưu trong Heap và có địa chỉ cho chỗ chứa thông tin đó

param sẽ được lưu trong stack cùng với địa chỉ của thông tin Object được lưu trong bộ nhớ heap đóng vai trò biến sẽ đi tham chiếu đến thông tin Object qua địa chỉ

Stack: thường sẽ được dùng để lưu các biến reference hoặc lưu biến lẫn giá trị thuộc premitive type

Ví dụ:

int a = 10;

float b = 2.5;

các biến này sẽ được lưu trong stack cùng với giá trị của nó

LƯU Ý : Các wrapper class cần tìm hiểu Caching, Boxing và AutoBoxing

Còn các method thì sẽ có một nơi Method Area để chứa

1. **Kết thúc (Termination)**:
   1. Khi chương trình hoàn thành, JVM sẽ giải phóng tất cả các tài nguyên đã sử dụng và kết thúc quá trình thực thi.

**JRE (Java Runtime Enviroment)**

là môi trường để chạy Java

JRE cung cấp :

+ Java Core Library: Các bộ thư viện tiêu chuẩn +3000 bộ như java.lang, java.utils, java.io,......

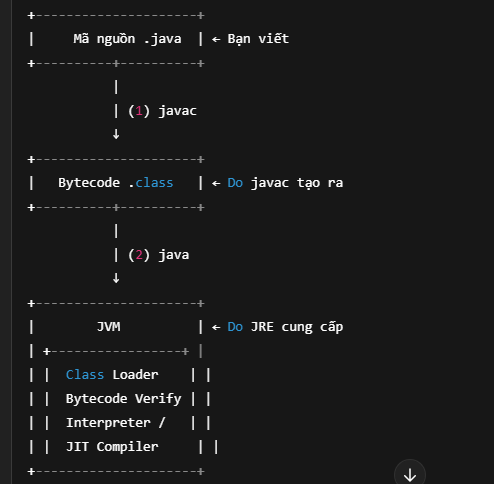
+ Class Loader System: nạp các file .class đã được javac biên dịch để load vào JVM

+ Bytecode Verifier: Nơi kiểm các định dạng bytecode

-> JRE như một bầu trời, một sân chơi cung cấp các công cụ đồ chơi để đảm bảo JVM có thể load .class và execute nó.

**JDK (Java Development Kit)**

là công cụ để code và phát triển Java, chứa mọi thứ từ để viết code, biên dịch, Debug và thực thi code.



Cơ chế:

1. Dùng IDE để tạo file .java và viết code
2. Sau đó dùng chức năng Build của IDE để javac biên dịch .java thành .class
3. JVM sẽ tiến hành load file .class vào để init và execute