**Biến primitive**: biến nguyên thủy (byte, int,..): dữ liệu được lưu ngay biến

**Biến preference**: biến tham chiếu (new): trỏ đến vị trí lưu của đối tượng lưu trong Heap

**code ---javac---> bytecode ----jvm----> machinecode**

Từ .java -> biên dịch (compile = javac) mã nguồn ra mã máy ra .class(bytecode)

Để chạy .class cần có JVM để tạo ra machine code, từ đó chạy trên HĐH

JVM chạy được trên window, linux, macOS,.., nó cung cấp môi trường runtime để chạy bytecode

Cấu trúc JVM

* **Classloader:** Là một hệ thống con của JVM được sử dụng để tải class file.
* **Just-In-Time (JIT) Compiler:** được sử dụng để cải thiện hiệu suất. **JIT biên dịch các phần của Bytecode mà có cùng tính năng tại cùng một thời điểm, và vì thế giảm lượng thời gian cần thiết để biên dịch.** Ở đây khái niệm Compiler là một bộ biên dịch tập chỉ thị của JVM thành tập chỉ thị của một CPU cụ thể. **JIT Compiler nó sẽ biên dịch các bytecode đã được thực thi trong JVM và cache (đọc gần giống catch) lại. Mỗi khi gọi đến các lệnh đã được thực thi trước đó thì sẽ gọi vào cache trong JIT Compiler**

HinhVuong hv = new HinhVuong();

* + h là biến tham chiếu, lưu trong stack, trỏ đến đối tượng HinhVuong trong bộ nhớ Heap

Tương tự: String s = new String();

* **Stack (nhỏ hơn)** là một vùng nhớ được sử dụng để lưu trữ **biến local của phương thức, tham số của phương thức, lời gọi hàm(main,..**. Các tham số và các biến local của một phương thức tạo thành **một bản ghi** kích hoạt, còn được gọi là một **stack frame**. Các bản ghi kích hoạt được đẩy vào một stack khi phương thức được gọi và đẩy ra khỏi stack khi phương thức trả về. Mỗi Thread có một Stack riêng, được tạo tại cùng thời điểm với Thread

Vd: biến args, biến tham chiếu khi tạo 1 đối tượng chưa new (Customer cus)

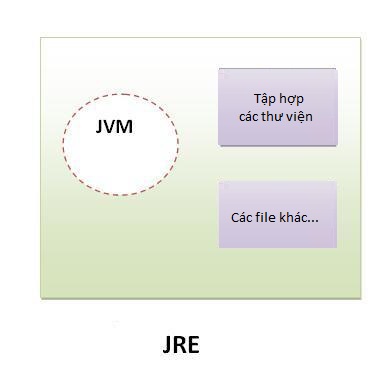
* **Heap** là một vùng nhớ trong bộ nhớ được sử dụng để **lưu trữ các đối tượng khi từ khóa new được gọi ra**, các **biến static và các biến toàn cục** (biến instance).

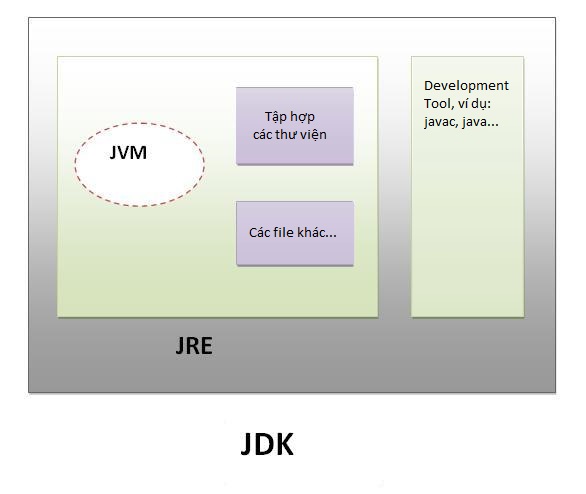
**Garbage collection (Quá trình thu gom rác)** trong máy ảo Java (JVM) là quá trình xác định và loại bỏ các Object không được sử dụng (unreferenced) khỏi bộ nhớ Heap

Quá trình thu gom rác cơ bản thông qua 3 bước sau:

1. **Marking**: Là bước đánh dấu những Object còn sử dụng và những Object không còn sử dụng.
2. **Normal deleting**: Trình Garbage Collector sẽ xóa cá Object không còn sử dụng.
3. **Deletion with Compacting**: Sau khi những Object không còn được sử dụng bị xóa, những Object còn được sử dụng sẽ được "gom" lại gần nhau. Điều đó làm tăng hiệu xuất sử dụng bộ nhớ trống để cấp phát cho những Object mới.

JRE (là viết tắt của Java Runtime Environment) được sử dụng để cung cấp môi trường runtime





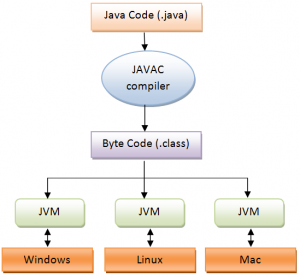
Trong quá trình JVM chạy, các object được sinh ra liên tục và thực thi nhiệm vụ. Sau một khoảng thời gian sẽ phải được kill đi nếu không thì JVM không còn chỗ chứa. Đây là quá trình Garbage Collector hay nôm na gọi là "dọn rác".

Trong Java, **hiện tượng rò rỉ bộ nhớ hầu như không xảy ra do bộ nhớ được quản lý bởi Java Virtual Machine (JVM) bằng cách tự động "dọn dẹp rác**". Người lập trình không phải quan tâm đến việc cấp phát và xóa bộ nhớ như C, C++. Tuy nhiên khi sử dụng những tài nguyên mạng, file IO, database (nằm ngoài kiểm soát của JVM) mà người lập trình không đóng (close) các streams thì rò rỉ dữ liệu vẫn có thể xảy ra.

Java là ngôn ngữ vừa biên dịch vừa thông dịch. Đầu tiên mã nguồn được biên dịch bằng công cụ JAVAC để chuyển thành dạng ByteCode. Sau đó được thực thi trên từng loại máy cụ thể nhờ chương trình thông dịch. Mục tiêu của các nhà thiết kế Java là cho phép người lập trình viết chương trình một lần nhưng có thể chạy trên bất cứ phần cứng cụ thể.

Môi trường phát triển của Java được chia làm hai phần: Trình biên dịch và trình thông dịch. Không như C hay C++, trình biên dịch của Java chuyển mã nguồn thành dạng bytecode độc lập với phần cứng mà có thể chạy trên bất kỳ CPU nào.

Nhưng để thực thi chương trình dưới dạng bytecode, tại mỗi máy cần phải có trình thông dịch của Java hay còn gọi là máy ảo Java. Máy ảo Java chuyển bytecode thành mã lệnh mà CPU thực thi được.



* Thông dịch (Interpreter) : Nó dịch từng lệnh rồi chạy từng lệnh, lần sau muốn chạy lại thì phải dịch lại.
* Biên dịch (Compiler): Code sau khi được biên dịch sẽ tạo ra 1 file thường là .exe, và file .exe này có thể đem sử dụng lại không cần biên dịch nữa

Compile once and run everywhere

javac.exe (javac: java compiler) có nhiệm vụ **biên dịch** code java sang byteccode  
Sau đó java.exe có nhiệm vụ **thông dịch** bytecode trong class file sang mã máy tùy vào chip/hệ điều hành.

# Các đặc trưng của Java

* Đơn giản: bỏ contro, goto, struct,. trong c/c++
* Hướng đối tượng
* Độc lập phần cứng và hệ điều hành: ko giống như c++ là đổi mã nguồn thành mã máy, phụ thuộc vào CPU
* Mạnh
* Bảo mật và an toàn
* Ngôn ngữ lập trình Java **yêu cầu chặt chẽ về kiểu dữ liệu.**
* Dữ liệu phải được khai báo tường minh.
* **Không sử dụng con trỏ và các phép toán với con trỏ.**
* **Java kiểm soát chặt chẽ việc truy nhập đến mảng, chuỗi. Không cho phép sử dụng các kỹ thuật tràn**. Do đó các truy nhập sẽ không vượt quá kích thước của mảng hoặc chuỗi.
* **Quá trình cấp phát và giải phóng bộ nhớ được thực hiện tự động.**
* Cơ chế xử lý lỗi giúp việc xử lý và phục hồi lỗi dễ dàng hơn
* Phân tán
* Đa luồng
* Động