Trần Trung Hiếu – Báo cáo thực tập

26 – 30/12/2022

1. **JDK:**

JDK viết tắt của Java Development Kit là một bộ công cụ phần mềm cung cấp môi trường phát triển ứng dụng viết bằng ngôn ngữ Java. JDK bao gồm cả Java Runtime Environment giúp lập trình viên có thể chạy thử để kiểm tra ứng dụng trong quá trình phát triển ứng dụng.

JDK bao gồm compiler, interpreter, debugger,…

* Trình thông dịch: Dịch từng hướng dẫn tại một thời điểm
* chậm vì CPU phải đợi cho đến khi mỗi câu lệnh được dịch.
* Cho phép bạn nhìn thấy ngay lỗi của mình và thực hiện các thay đổi cho phù hợp
* Trình thông dịch luôn nằm giữa chương trình (program) và máy tính (computer)
* Trình biên dịch: Dịch tất cả code của bạn sang ngôn ngữ mong muốn
* Mất nhiều thời gian hơn để biên dịch tuy nhiên ứng dụng được thực thi nhanh hơn nhiều so với trình thông dịch

JDK = JRE + Các công cụ hỗ trợ phát triển ứng dụng Java compiler là một trong số các công cụ hỗ trợ phát triển ứng dụng của JDK. Java compiler được sử dụng để biên soạn các tập tin Java (đuôi .java) để tạo ra tập tin .class tương ứng.

1. **JVM**

JVM (Java Virtual Machine) là 1 máy ảo java. Nó cung cấp môi trường để code java có thể được thực thi, chương trình Java khi biên dịch sẽ tạo ra các file \*.class chứa byte code. Các file \*.class sẽ được JVM thực hiện chuyển byte code thành mã máy tương ứng với từng hệ điều hành và phần cứng khác nhau thực để thi.

JVM thực hiện các công việc chính sau đây:

* Nạp code (các class, resource)
* Kiểm tra code (kiểm tra code có đúng cú pháp không, có bị lỗi không, tất nhiên nếu code có lỗi thì sẽ không chạy được chương trình rồi)
* Thực thi code
* Cung cấp môi trường runtime

JVM có 3 thành phần chính:

* Class Loader: Tìm kiếm và load các file \*.class vào vùng nhớ của java dưới dạng bytecode
* Data Area : vùng nhớ hệ thống cấp phát cho Java Virtual Machine
* Execution Engine: chuyển các lệnh của JVM trong file \*.class thành các lệnh của máy, hệ điều hành tương ứng và thực thi chúng.

Cơ chế thực hiện/cấu trúc JVM:

Sau khi Classloader tìm và load các file .class, các file này sẽ được máy ảo JVM cung cấp bộ nhớ tương ứng với chúng.

* Class (Method) area: là vùng nhớ cấp phát cho Class (Method) trong đó lại phân chia thành heap, stack, PC register, native method stack.
* Heap: Đây là khu vực dữ liệu thời gian chạy (runtime) trong đó các object được phân vùng nhớ.
* Stack: Các phương thức và tham chiếu tới đối tượng địa phương được lưu trữ trong Stack. Mỗi Thread quản lý một stack. Khi phương thức được gọi, nó được đưa vào đỉnh của Stack. Stack lưu trữ trạng thái của phương thức bao gồm: dòng code thực thi, tham chiếu tới đối tượng địa phương. Khi phương thức chạy xong, vùng nhớ (dòng code thực thi, tham chiếu tới đối tượng địa phương) được đẩy ra khỏi stack và tự động giải phóng.
* PC register (Program Counter Register – Instruction pointer): Nó chứa địa chỉ của các instruction (chỉ lệnh) của máy ảo Java hiện đang được thực hiện.
* Native Method Stack: Nơi chứa tất cả các method native trong chương trình
* Execution Engine bao gồm:
  + Một bộ vi xử lý ảo
  + Một trình thông dịch để đọc dòng bytecode sau đó thực hiện các chỉ lệnh
  + Trình biên dịch Just-In-Time (JIT). JIT biên dịch các phần của mã byte có chức năng tương tự như cùng một lúc, và do đó làm giảm thời gian cần thiết cho việc biên dịch từ những chỉ lệnh của một máy ảo Java (JVM) sang các chỉ lệnh cho một CPU cụ thể.
* Java Native Interface: JNI là một framework cung cấp giao diện để giao tiếp với một ứng dụng khác được viết bằng ngôn ngữ khác như C, C++, Assembly, v.v. Java sử dụng JNI để gửi đầu ra tới Bảng điều khiển (console) hoặc tương tác với các thư viện của hệ điều hành.
* Java Garbage Collection: Mỗi khi một đối tượng được tạo trong Java, nó sẽ đi vào một vùng bộ nhớ được gọi là Heap. Tất cả các đối tượng, bất kể chúng được tạo khi nào, ở đâu hoặc như thế nào, đều tồn tại trên heap. Trong Java Heap còn được gọi là Garbage – collectible heap. Khi bạn tạo một đối tượng, Java sẽ phân bổ không gian bộ nhớ trên heap tùy theo đối tượng cụ thể đó cần bao nhiêu. Ví dụ, một đối tượng có 15 biến thể hiện sẽ cần nhiều không gian hơn đối tượng chỉ có hai biến thể hiện. Khi JVM có thể thấy rằng một đối tượng không bao giờ có thể được sử dụng lại, thì đối tượng đó sẽ đủ điều kiện để thu gom rác. Và nếu bạn sắp hết bộ nhớ, Trình thu gom rác sẽ chạy, loại bỏ các đối tượng không còn truy cập và giải phóng dung lượng để không gian có thể được sử dụng lại.