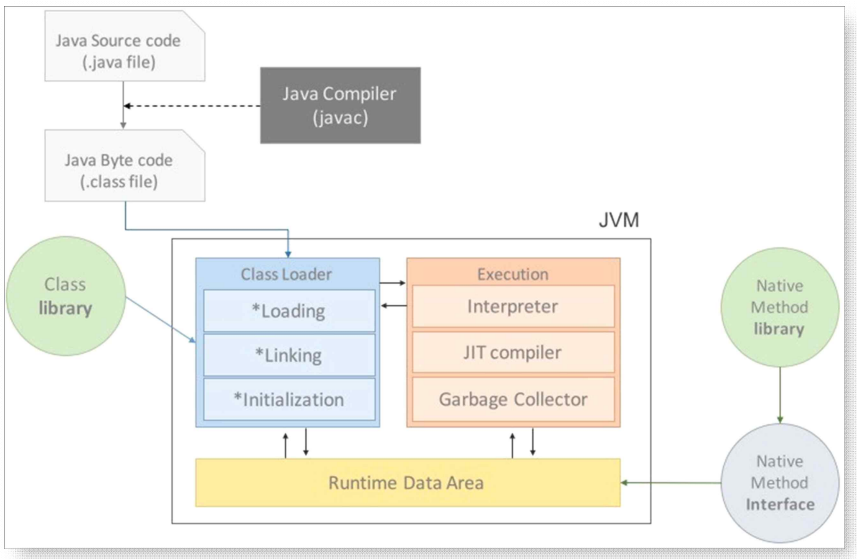
[JAVA]
- 썬 마이크로시스템즈의 제임스 고슬링이라는 사람과 다른 연구원들이 개발한 프로그래밍 언어.
- 어떠한 하드웨어, 운영체제이던 상관없이 컴파일된 코드(바이트 코드)가 플랫폼에 상관없이 독립적으로 구동된다.

[JVM - JAVA Virtual Machine]
- 자바는 플랫폼에 구애받지 않고 자바 소스코드를 실행시키기 위해 JVM을 사용한다.
- JVM은 자바 소스코드를 해석하여 실행하는 역할, 메모리 관리 등을 수행한다.
- JVM은 스택기반의 가상머신이다.

[자바 프로그램 실행과정]
- 프로그램이 실행되면 JVM은 OS로부터 프로그램이 필요로 하는 메모리를 할당받는다.
- JVM은 할당 받은 메모리를 용도에 따라 여러 영역으로 나누어 관리한다.
- 자바 컴파일러(javac)가 자바 소스코드(.java)를 읽으들여 자바 바이트코드(.class)로 변환시킨다.
- Class Loader를 통해 class 파일들을 JVM으로 로딩한다.
- 로딩된 class 파일들은 Execution engine을 통해 해석된다.
- 해석된 바이트코드는 Runtime Data Areas에 배치되어 실질적인 수행이 이루어지게 된다.
- 위의 실행 과정 속에서 JVM은 필요에 따라 Thread Synchroniztion과 GC같은 관리작업을 수행한다.



[JVM 구성]
- Class Loader
- Execution Engine
- Interpreter
- JIT (Just - In - Time)
- Garbage Collector
- Runtime Data Area

[Class Loader]
- JVM 내로 클래스(.class)파일을 로드한다.
- 사용하지 않는 클래스는 메모리에서 삭제한다.

[Execution Engine]
- Class Loader가 Load한 클래스 파일을 실행한다.
- 클래스 파일은 기계어 보다는 비교적 사람이 보기 편한 형태로 기술되어 있어 이를 완전한 기계어로 번역하는 과정이 필요하다.

[Interpreter]
- 클래스 파일을 명령어 단위로 읽어 기계어로 번역한다.
- 한 줄 씩 수행하기 때문에 상대적으로 속도가 느리다.

[JIT]
- 인터프린터 방식으로 수행하다 적절한 시점에 클래스 파일을 컴파일하여 기계어로 번역한 후 더 이상 인터프리팅 하지 않고 번역된 기계어를 직접 실행하는 방식.
- 기계어로 번역된 코드는 캐시에 보관되어 빠르게 수행된다.
- 컴파일을하여 기계어로 전체 번역하는 시간이 필요하기에 많이 사용되지 않는 코드는 인터프리팅하는 것이 유리하다.
- JVM은 내부적으로 해당 메서드가 얼마나 자주 수행되는지 확인하여 컴파일이 필요한지 파악한다.

[Garbage Collector]
- 사용하지 않는 메모리 공간을 확보하는 역할을 한다.

[Runtime Data Area]
- 프로그램을 수행하기 위해 OS에서 할당받은 메모리 공간
- PC, Stack, Native Method Stack, Method Area, Heap 영역 등으로 구분된다.

[PC]
- Program Counter

[Stack]
- 매개변수, 지역변수, 리턴 값, 스레드, 메소드 저장

[Native Method Stack]
- 번역된 기계어를 실행시키는 영역.
- JAVA가 아닌 다른 언어로 작성된 코드를 위한 공간. (C언어 코드를 실행시켜 커널에 접근할 수 있다.)

[Method]
- Class Area, Static Area라고도 불린다.
- 클래스의 필드 메소드 타입정보를 저장하는 장소.
- 상수 자료형을 저장하는 Runtime Constant Pool이 있다.

[Heap]
- 객체를 저장하는 공간.
- Method 영역에 올라온 클래스만 객체로 생성할 수 있다.
- Permanent Generation, New/Young 영역, Old 영역으로 구분된다.