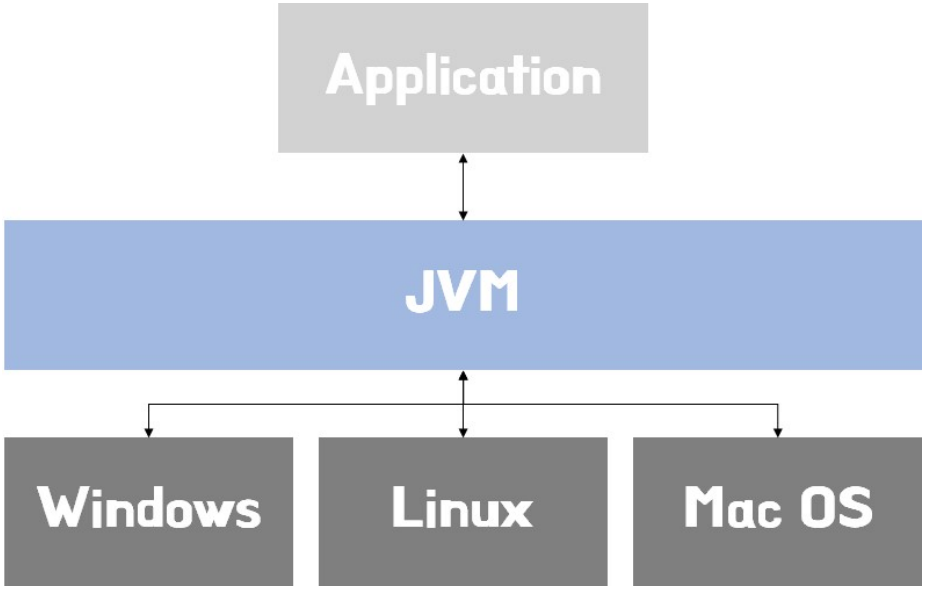
JVM

JVM

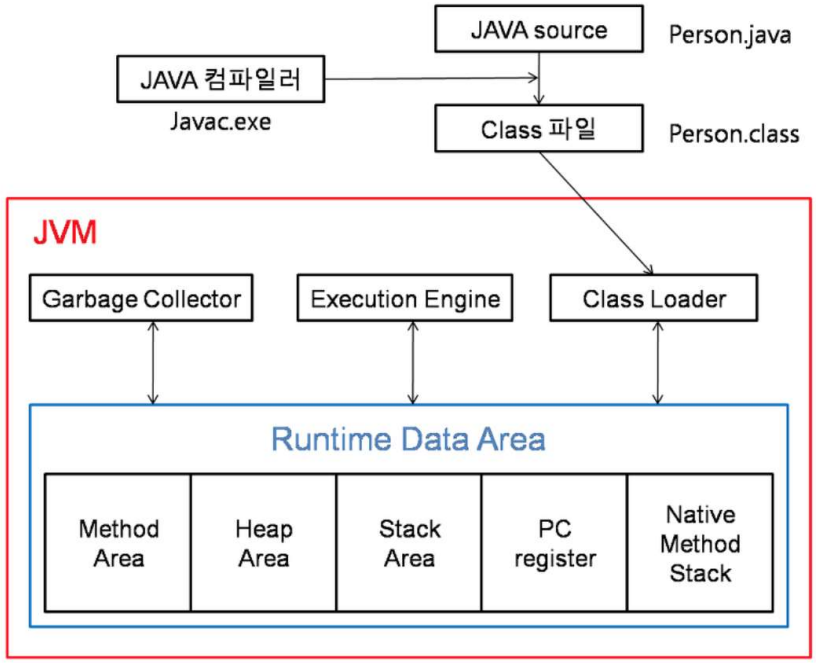
JVM 역할

1. 자바의 바이트 코드(Class파일)를 실행하는 실행기
   Class파일을 기계어로 변환하여 실행하기 때문에 소스코드가 OS에 종속되지 않는다.
   * WORA(Write Once Run Anywhere)

2. 프로그램을 실행 하는데 필요한 데이터들을 저장하고 관리한다.



JVM - 자바 코드 실행과정



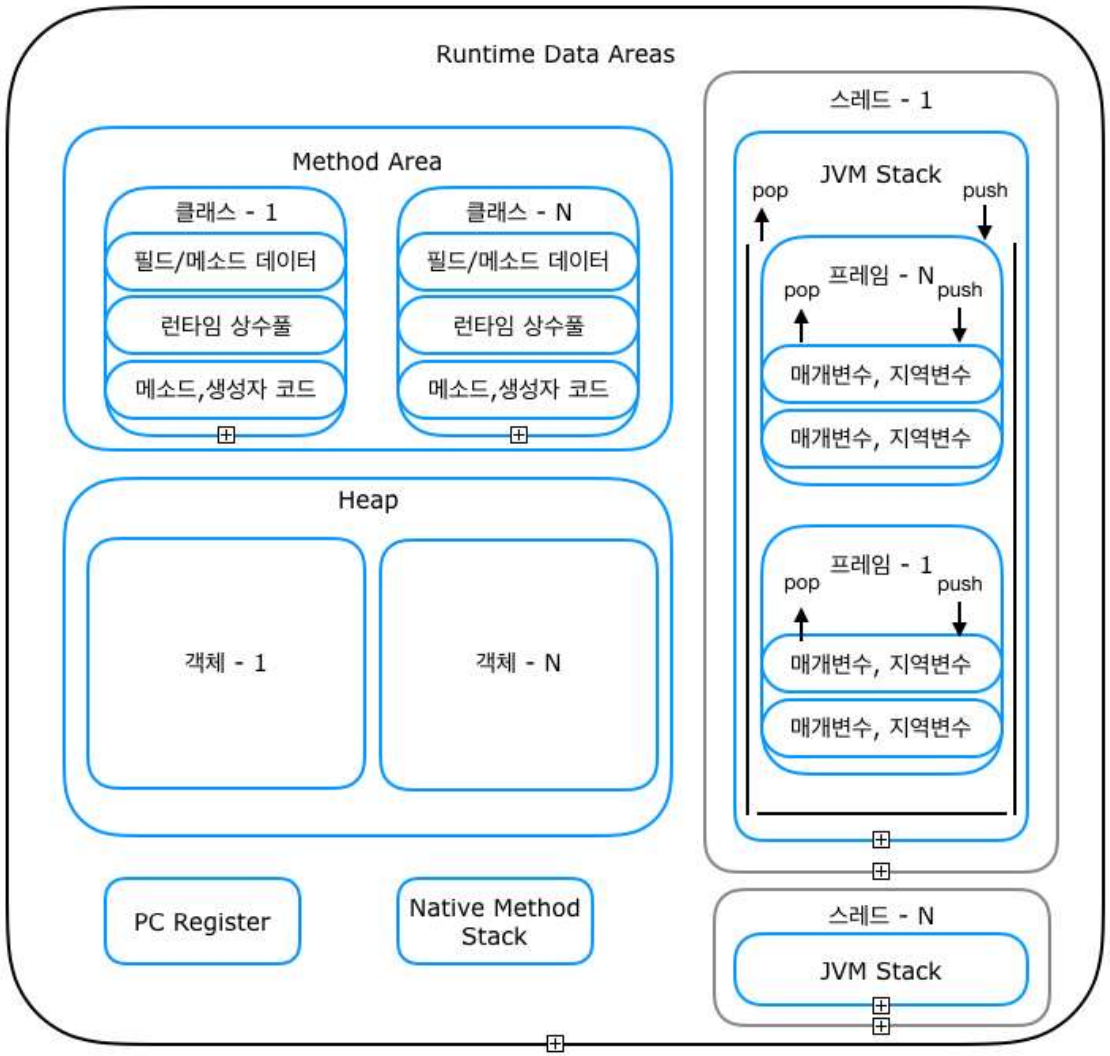
1. Java 컴파일러가 Java 파일을 Class파일로 컴파일

2. Class Loader가 Class파일을 JVM의 메모리에 적제

3. Execution Engine이 Class파일의 Byte Code를 한 줄 씩 읽어 실행

4. 빈번하게 사용되는 코드는 JIT(just-in-time compilation) 가 미리 기계어로 번역하여 저장해둔다.

JVM – Runtime Data Area



JVM – Runtime Data Area

1. **Method Area == Static Area == Code Area == Class Area:**
   a. Class 수준의 정보가 저장되는 영역 (static변수, instance변수의 선언부, 메서드(선언부+코드), Class Name, Type정보)
   b. Static Area 영역에 올라간 데이터는 프로그램이 종료될 때 까지 메모리에서 내려오지 않는다.
   c. 모든 thread에 공유하는 메모리 영역이다.
   - Runtime Constant Pool : 모든 클래스와 인터페이스의 상수, 메서드와 필드에 대한 주소를 가지고 있는 영역
     실행엔진은 Runtime Constant Pool 을 사용해 메서드나 필드의 실제 주소를 찾아서 참조

2. **Heap Area** : Object가 저장되는 영역. Garbage Collector에 의해 관리되는 영역이다.
   모든 thread에 공유하는 메모리 영역이다.

JVM – Runtime Data Area

3. Stack Area : 원시타입의 데이터와 Heap 영역에 저장된 데이터에 접근할 수 있는 주소가 저장된다.
지역변수들은 block scope를 지니며 scope가 끝난 변수는 메모리에서 삭제 된다.
thread 별로 독자적인 메모리 공간을 지닌다.

4. PC register : 현재 실행되는 메서드 스택 프레임의 주소값

5. Native Method Stack : 자바 외의 언어로 작성된 네이티브코드를 저장. 주로 OS에 접근하기 위한 코드가 저장되어 있다.