JVM 의

(Java Virtual Machine)

구조와 Stack & Heap

2020.08.29
김연정

JDK? JRE?? JVM???

JDK : Java Development Kit

Development Tools

JRE : Java Runtime Environment

Libraries
Classes
Other files

JVM : Java Virtual Machine

Java Virtual Machine : 자바 가상 머신

- 자바 Byte code를 실행하는 주체
- OS와 Java Program 사이를 연결해주는 중계자 역할
- Linux, Window, Mac 등 OS에 맞게 구현
  -> CPU나 OS에 상관없이 JAVA실행이 가능
- 가비지 컬렉션 (Garbage Collection)

참고) JVM Language : JVM에서 실행가능한 언어
Kotlin, Scala, Groovy, Clojure, JRuby, Jython

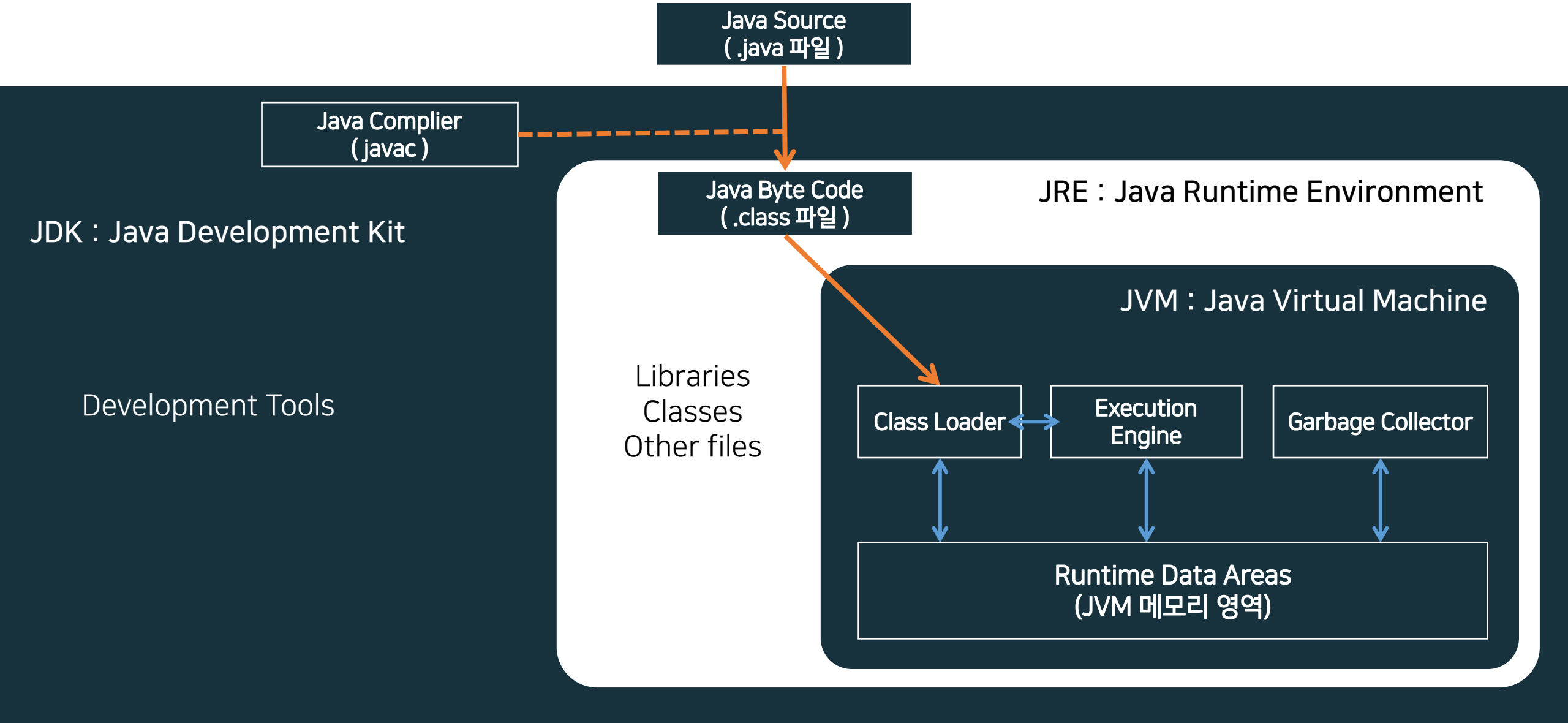
JVM

(Java Virtual Machine)

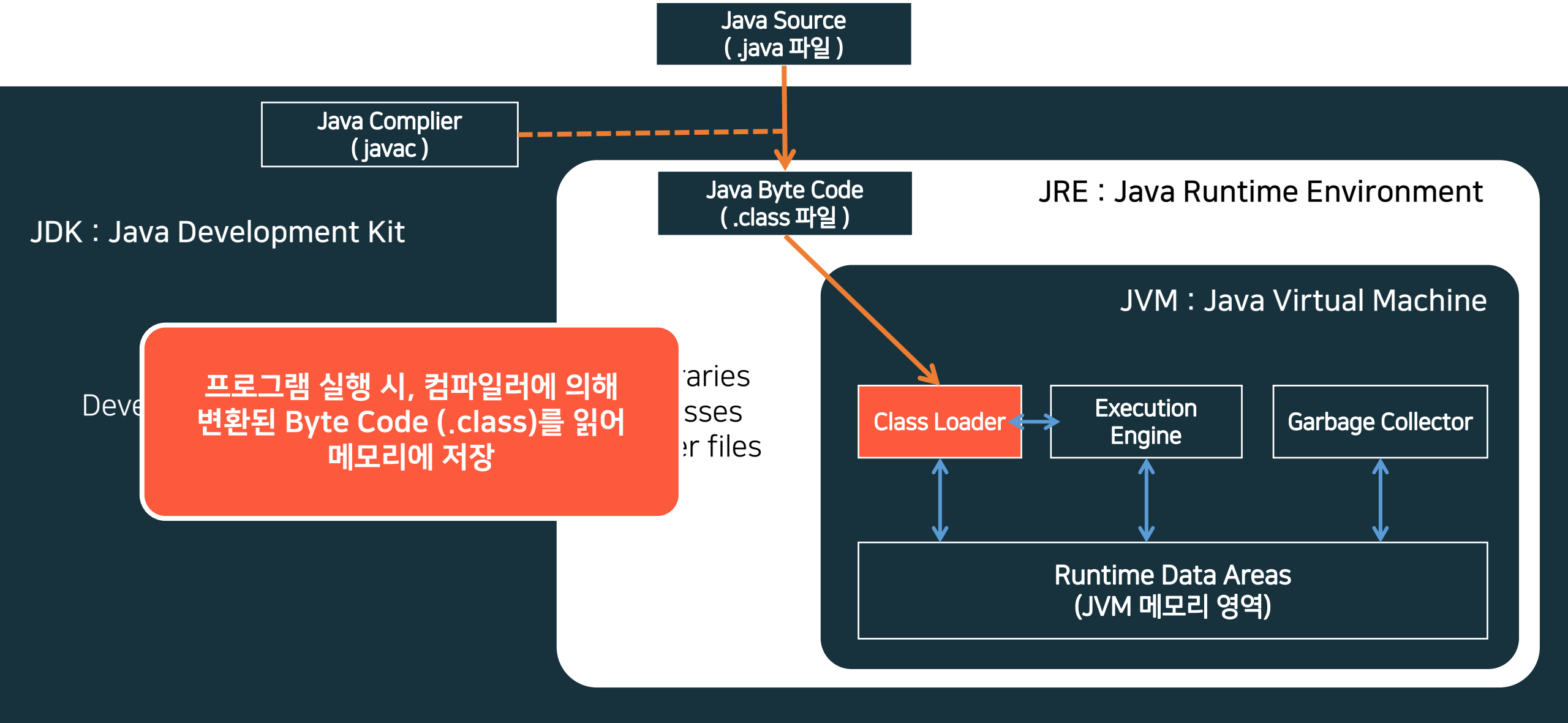
일반 프로그램 실행구조

자바 프로그램 실행구조

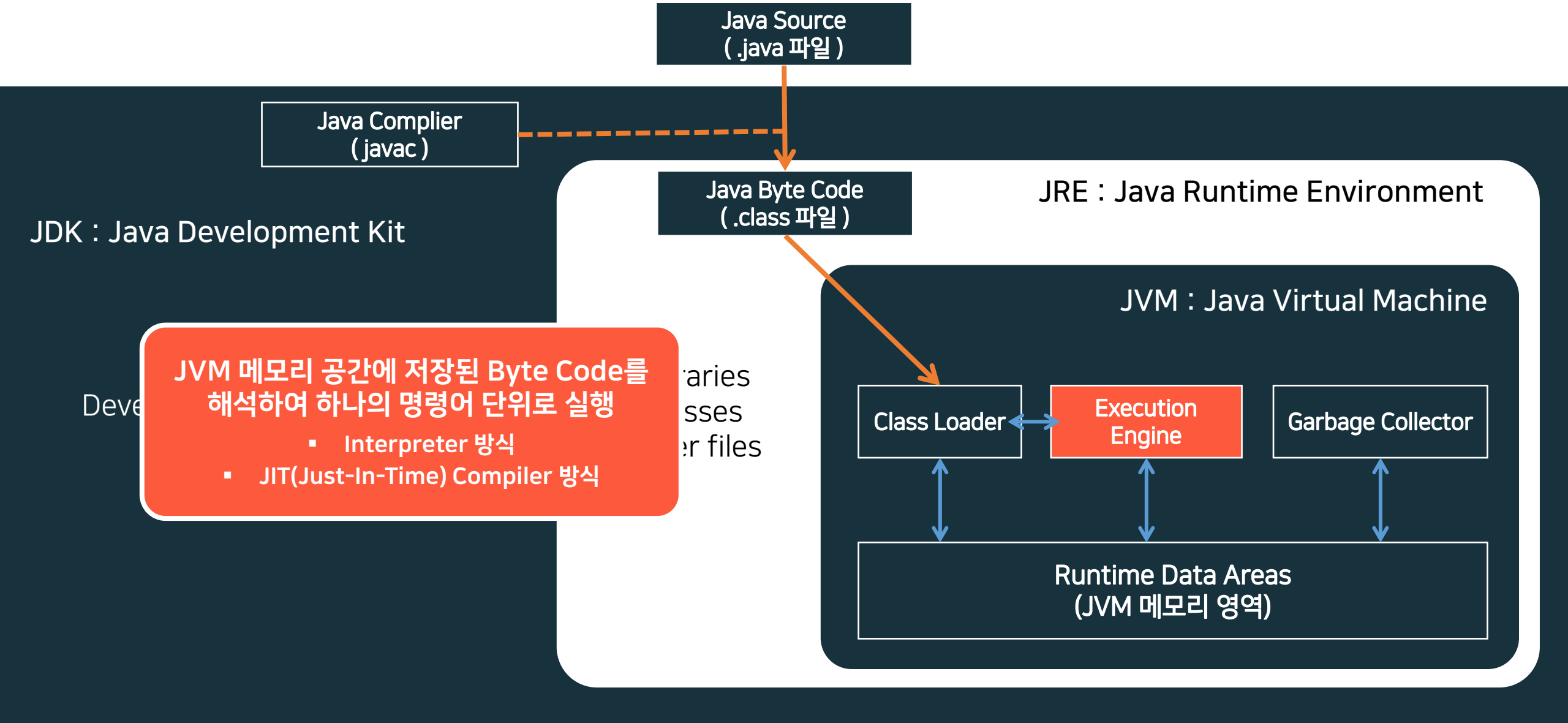
Java 실행 과정


Java Source
(.java 파일)
Java Complier
(javac)
Java Byte Code
(.class 파일)
JRE : Java Runtime Environment
JDK : Java Development Kit
Development Tools
Libraries
Classes
Other files
JVM : Java Virtual Machine
Class Loader
Execution Engine
Garbage Collector
Runtime Data Areas
(JVM 메모리 영역)

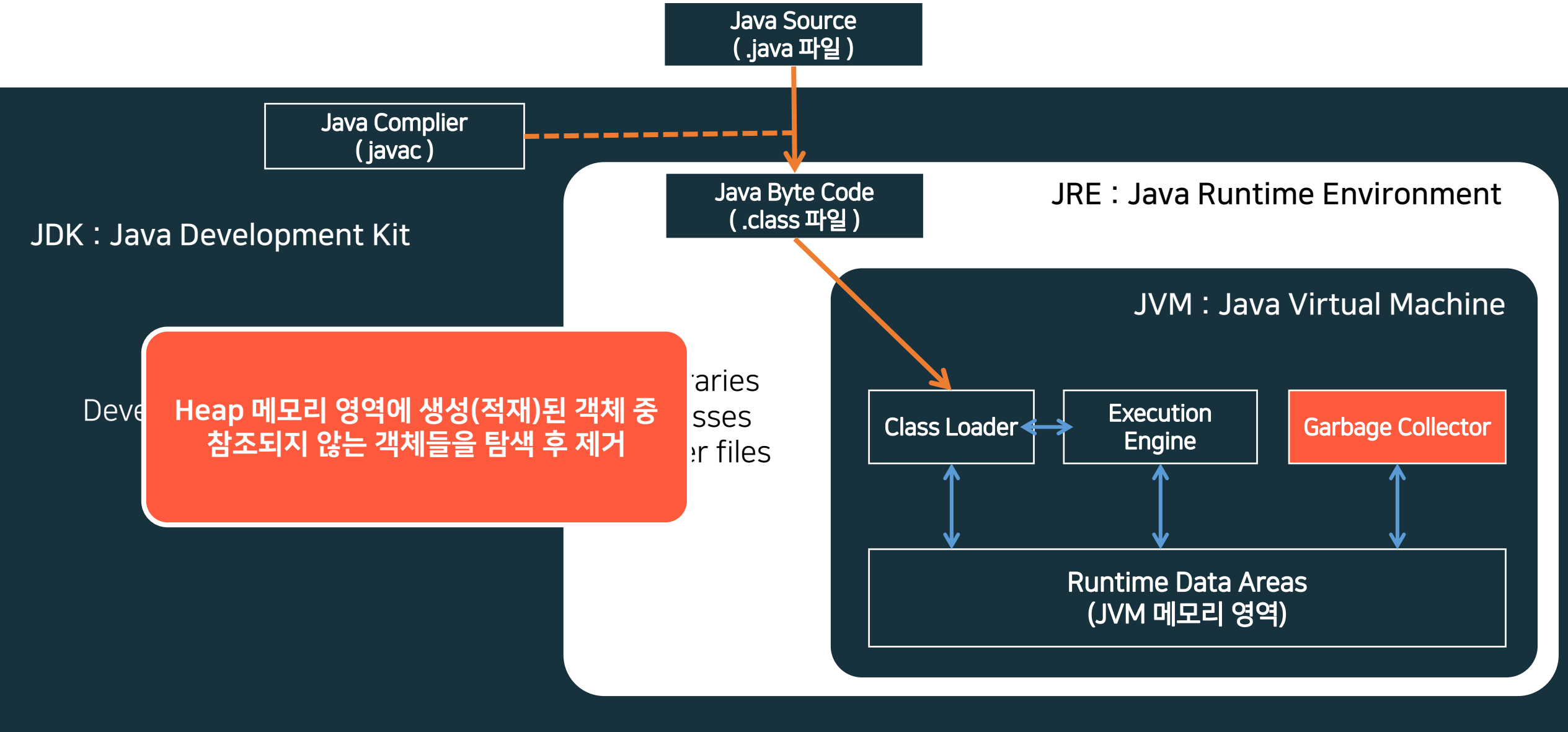
Java 실행 과정


Java Source
(.java 파일)
Java Complier
(javac)
Java Byte Code
(.class 파일)
JDK : Java Development Kit
JRE : Java Runtime Environment
JVM : Java Virtual Machine
프로그램 실행 시, 컴파일러에 의해
변환된 Byte Code (.class)를 읽어
메모리에 저장
Class Loader
Execution Engine
Garbage Collector
Runtime Data Areas
(JVM 메모리 영역)

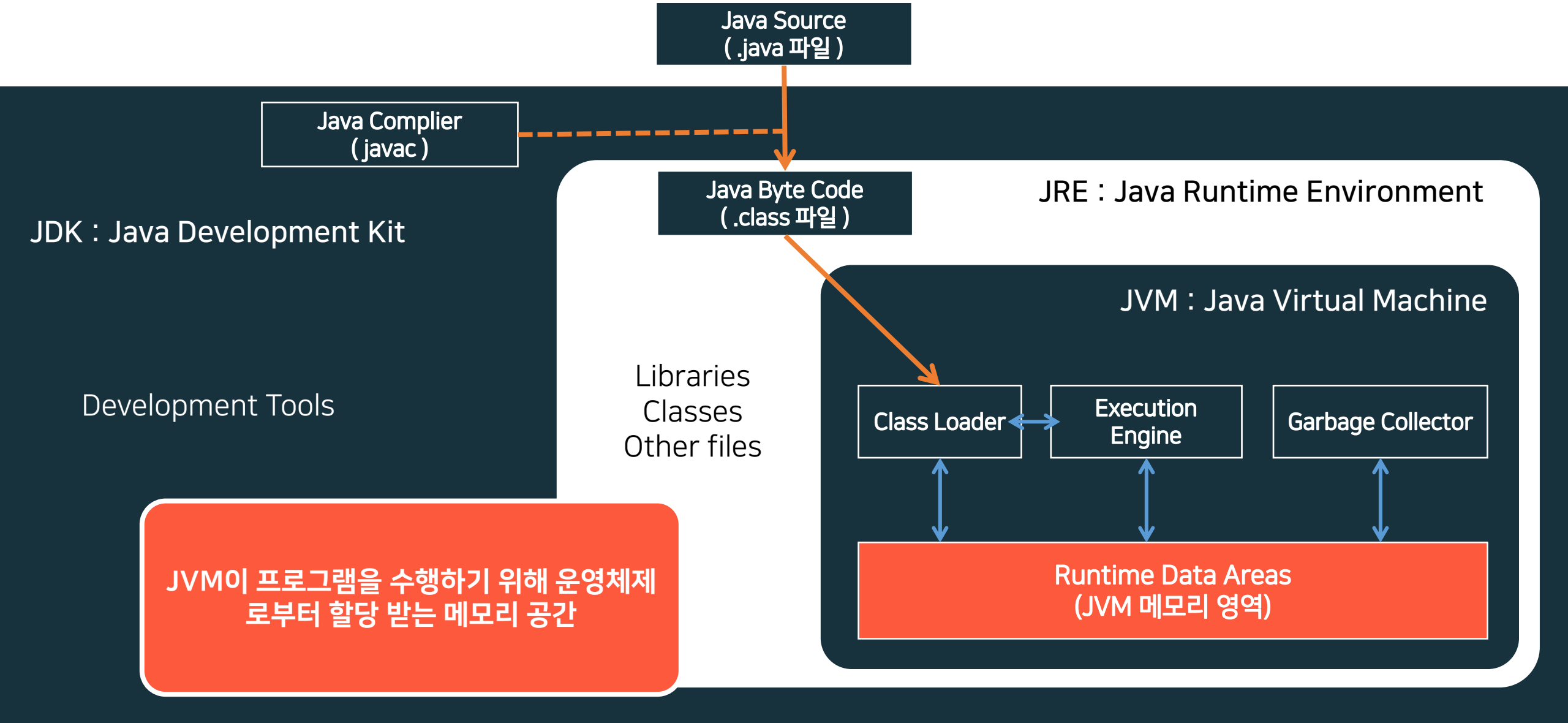
Java 실행 과정


Java Source
(.java 파일)
Java Complier
(javac)
Java Byte Code
(.class 파일)
JDK : Java Development Kit
JRE : Java Runtime Environment
JVM : Java Virtual Machine
JVM 메모리 공간에 저장된 Byte Code를
해석하여 하나의 명령어 단위로 실행
Interpreter 방식
JIT(Just-In-Time) Compiler 방식
Class Loader
Execution Engine
Garbage Collector
Runtime Data Areas
(JVM 메모리 영역)

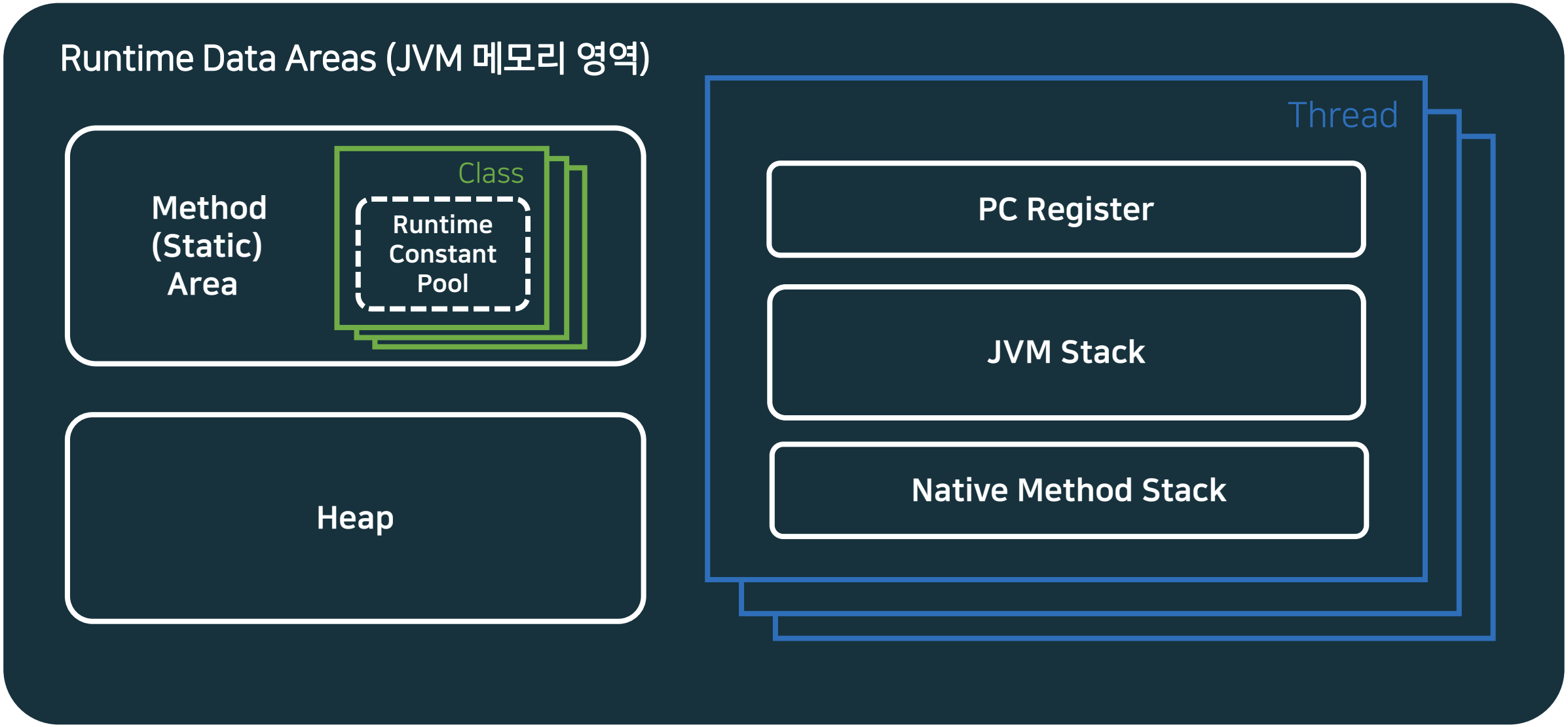
Java 실행 과정


Java Source
(.java 파일)
Java Complier
(javac)
Java Byte Code
(.class 파일)
JDK : Java Development Kit
JRE : Java Runtime Environment
JVM : Java Virtual Machine
Heap 메모리 영역에 생성(적재)된 객체 중
참조되지 않는 객체들을 탐색 후 제거
Class Loader
Execution Engine
Garbage Collector
Runtime Data Areas
(JVM 메모리 영역)

Java 실행 과정


Java Source
(.java 파일)
Java Complier
(javac)
Java Byte Code
(.class 파일)
JDK : Java Development Kit
JRE : Java Runtime Environment
JVM : Java Virtual Machine
Development Tools
Libraries
Classes
Other files
Class Loader
Execution Engine
Garbage Collector
Runtime Data Areas
(JVM 메모리 영역)
JVM이 프로그램을 수행하기 위해 운영체제 로부터 할당 받는 메모리 공간

JVM 메모리 구조


Runtime Data Areas (JVM 메모리 영역)
Method (Static) Area
Class
Runtime Constant Pool
Heap
Thread
PC Register
JVM Stack
Native Method Stack

JVM 메모리 구조

Method(Static) Area

JVM이 읽어들인 각각의 클래스, 인터페이스에 대한 Runtime Constant Pool, 멤버 변수(필드), 멤버 함수(Method), 클래스 변수(Static 변수), 생성자, Method의 바이트 코드 등을 저장하는 공간

Runtime Constant Pool

각 클래스와 인터페이스의 상수 뿐만 아니라 메서드와 필드에 대한 모든 Reference까지 담고 있는 테이블
어떤 메서드나 필드를 참조할 때, JVM은 런타임 상수 풀을 통해 메서드나 필드의 실제 메모리 상 주소를 참조하여 중복을 막는 역할

JVM 메모리 구조

Heap

런타임 시 동적으로 할당하여 사용하는 영역, 가비지 컬렉션 대상
New 연산자로 생성된 인스턴스 또는 객체와 배열을 저장하는 공간

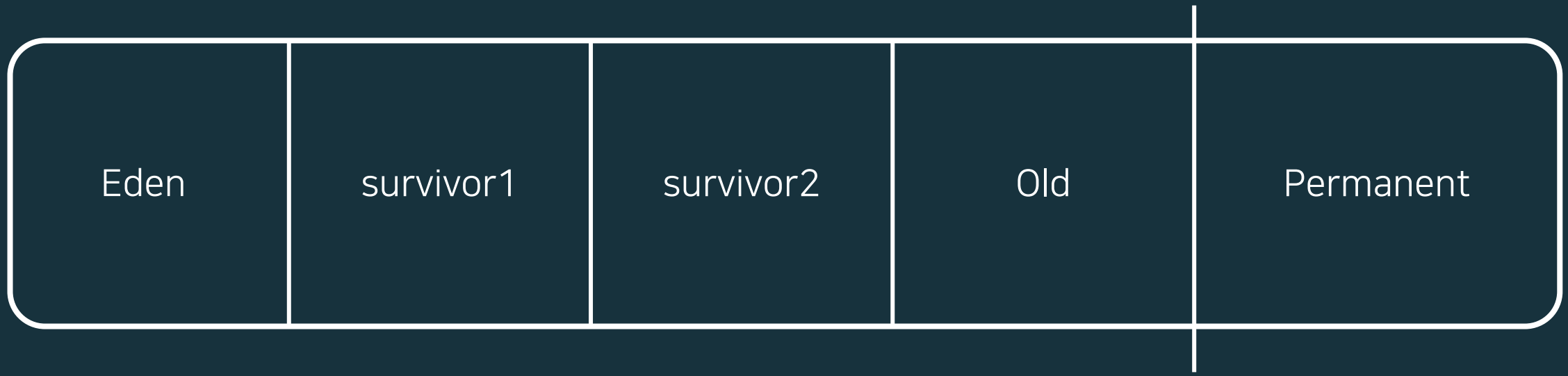
힙 구성 방식이나 가비지 컬렉션 방법 등은 JVM 벤더의 재량

힙 영역에서 생성된 객체와 배열은
스택 영역의 변수나 다른 객체의 필드에서 참조

참조하는 변수나 필드가 없다면 의미 없는 객체가 되어 GC 대상이 됨

JVM 메모리 구조

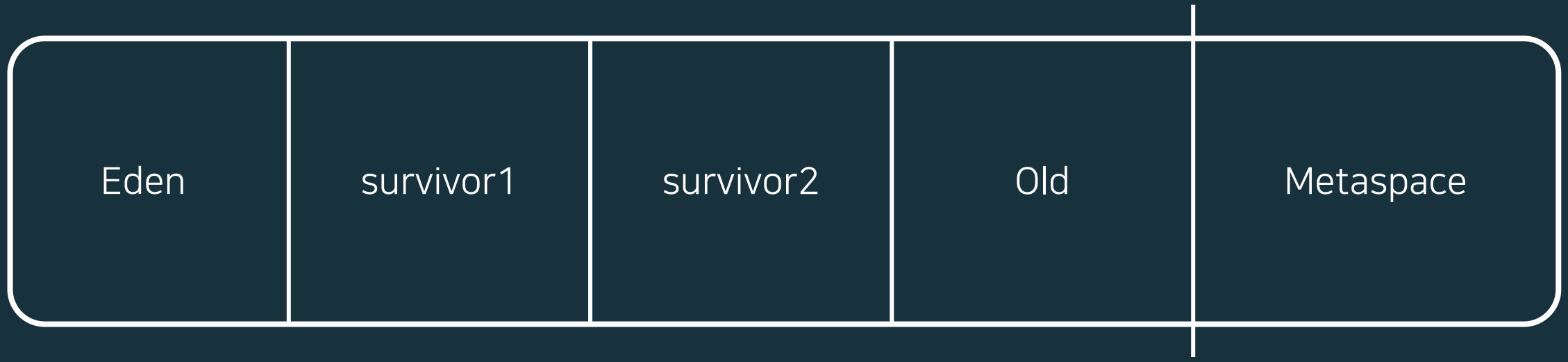
Heap



Heap (JDK7)

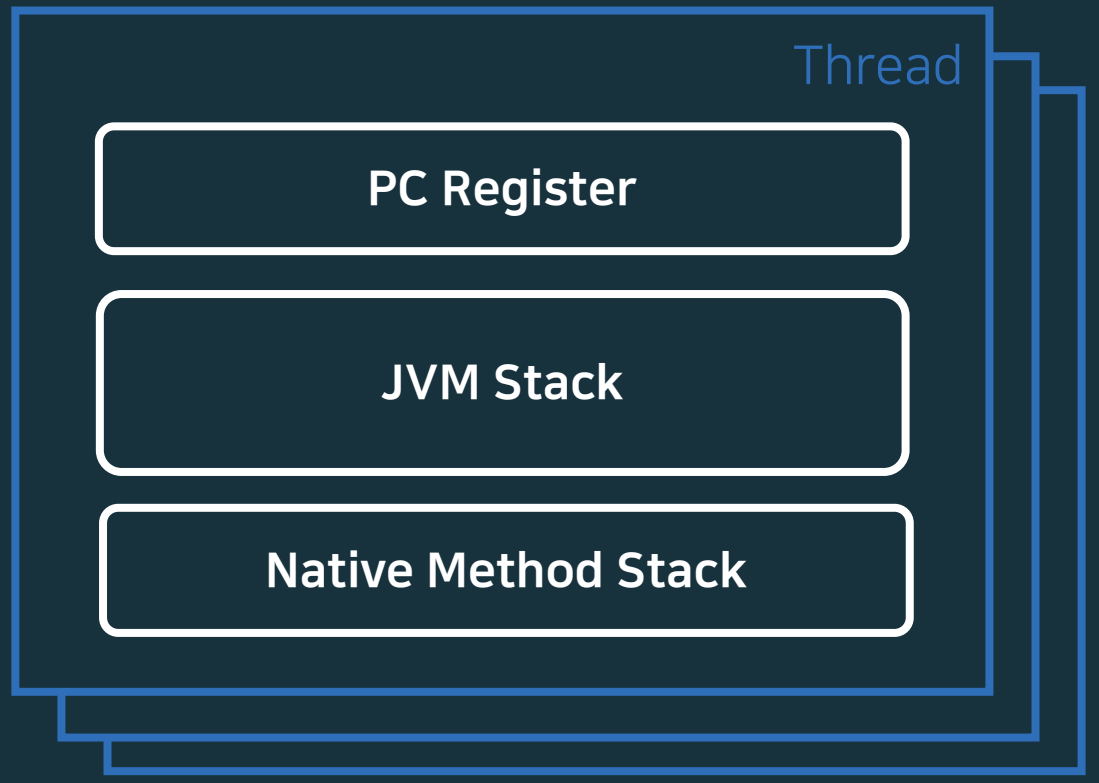
JVM 메모리 구조

Heap



Heap (JDK8)

JVM 메모리 구조



각 스레드마다 하나씩 존재하며 스레드가 시작될 때 생성

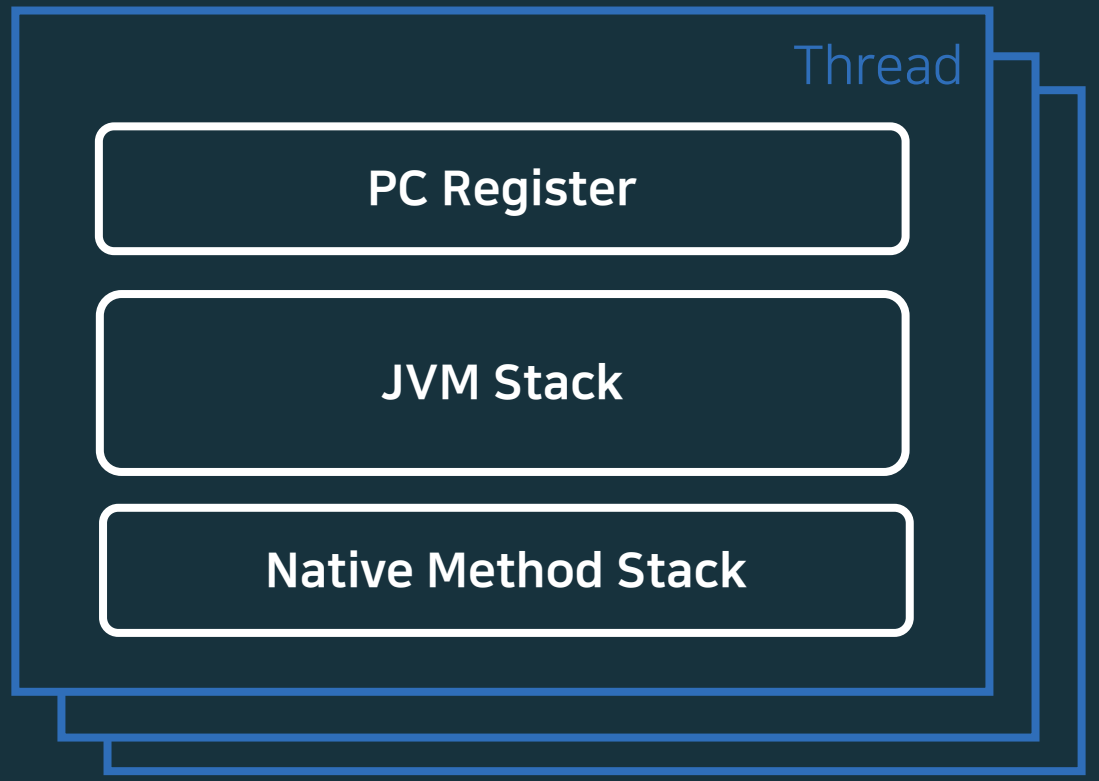
PC Register

스레드가 어떤 명령어로 실행되어야 할 지에 대한 기록을 하는 부분
현재 수행중인 명령의 주소(Address)를 갖음
현재 실행중인 명령이 종료되면 카운트 값을 증가시켜 다음 명령 실행

Native Method Stack

자바 외 언어로 작성된 네이티브 코드를 위한 Stack 메모리 영역
바이트 코드가 아닌, 기계어로 작성된 코드를 실행하는 공간
Java Native Interface(JNI)를 통해 바이트 코드로 변환됨
JNI를 통해 호출하는 C/C++ 등의 코드를 수행하기 위한 스택으로
언어에 맞게 C스택이나 C++스택이 생성

JVM 메모리 구조



각 스레드마다 하나씩 존재하며 스레드가 시작될 때 생성

JVM Stack

Method들이 호출되게 되면 각 Method를 위한 메모리 할당
컴파일 시 결정되는 기본형(Primitive) 데이터타입이 저장되는 공간
호출된 Method의 지역변수, 매개변수, return 값, 참조변수 등이 저장

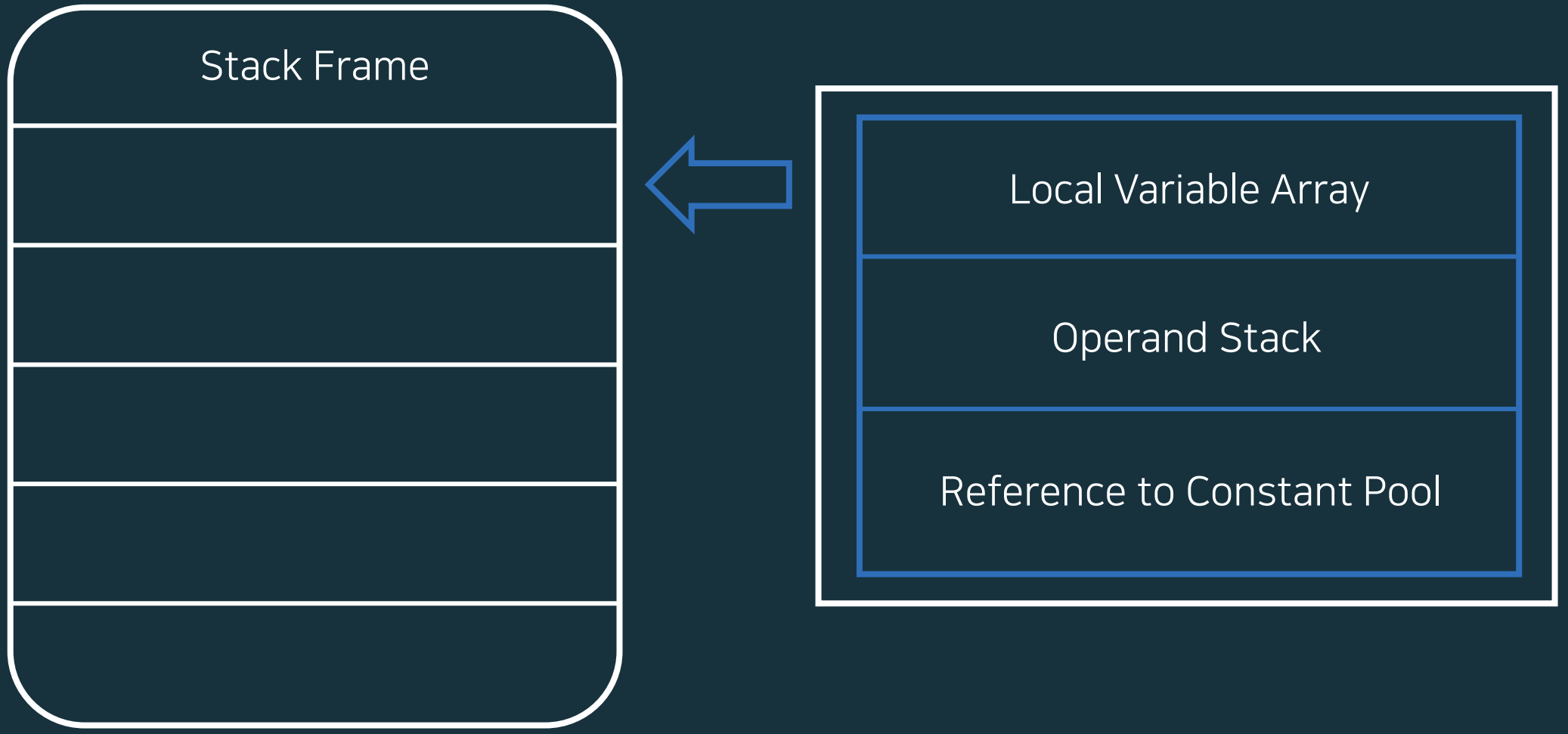
'값'이 아닌 '참조' 타입 변수는 Heap 영역이나 Method 영역에
Reference 값을 가리키게 됨

스택 프레임(Stack Frame)이라는 구조체를 저장하는 스택
JVM은 오직 JVM 스택에 스택 프레임을 추가(Push) 제거(Pop)하는
동작만 수행

예외 발생시 printStackTrace() 등의 Method를 보여주는
Stack Trace의 각 라인은 하나의 스택 프레임을 표현

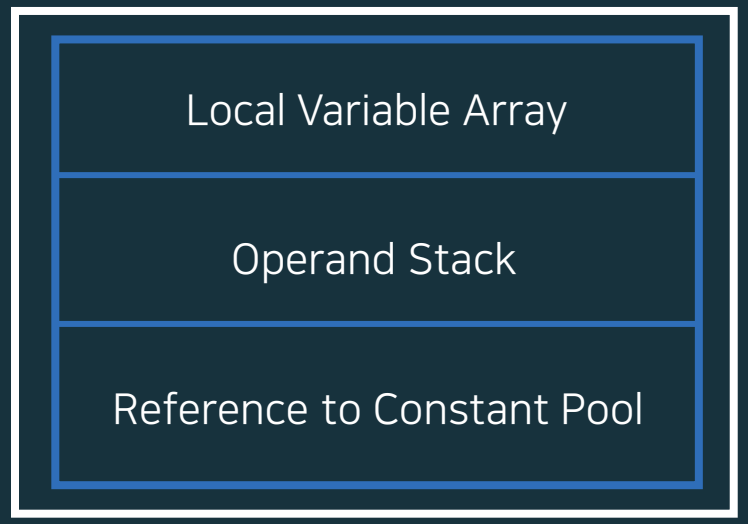
JVM 메모리 구조

JVM Stack



JVM 메모리 구조

JVM Stack



Stack Frame

Stack Frame

JVM 내에서 Method가 수행될 때마다 하나의 스택 프레임이 생성되어 해당 Thread의 JVM 스택에 추가되고 Method가 종료되면 스택 프레임이 제거됨

지역 변수 배열, 피연산자 스택의 크기는 컴파일 시 결정되기 때문에 스택 프레임의 크기도 메서드에 따라 크기가 고정됨

Local Variable Array

0부터 시작하는 인덱스를 가진 배열
0은 Method가 속한 클래스 인스턴스의 this 레퍼런스
1부터는 Method에 전달된 파라미터들이 저장
Method 파라미터 이후에는 메서드의 지역 변수들이 저장

Operand Stack

Method의 실제 작업 공간
각 Method는 피연산자 스택과 지역 변수 배열 사이에서 데이터를 교환
다른 Method 호출 결과를 추가(Push)하거나 제거(Pop)함
피연산자 스택의 크기도 컴파일 시에 결정

JVM 실행 과정

아주 좋은 참고 자료^^

https://homoefficio.github.io/2019/01/31/Back-to-the-Essence-Java-%EC%BB%B4%ED%8C%8C%EC%9D%BC%EC%97%90%EC%84%9C-%EC%8B%A4%ED%96%89%EA%B9%8C%EC%A7%80-2/

The End

2020.08.29
김연정

Reference

- https://homoefficio.github.io/2019/01/31/Back-to-the-Essence-Java-%EC%BB%B4%ED%8C%8C%EC%9D%BC%EC%97%90%EC%84%9C-%EC%8B%A4%ED%96%89%EA%B9%8C%EC%A7%80-2/
- https://d2.naver.com/helloworld/1230
- https://gbsb.tistory.com/2
- https://re-build.tistory.com/2
- https://jeong-pro.tistory.com/148
- https://advenoh.tistory.com/14