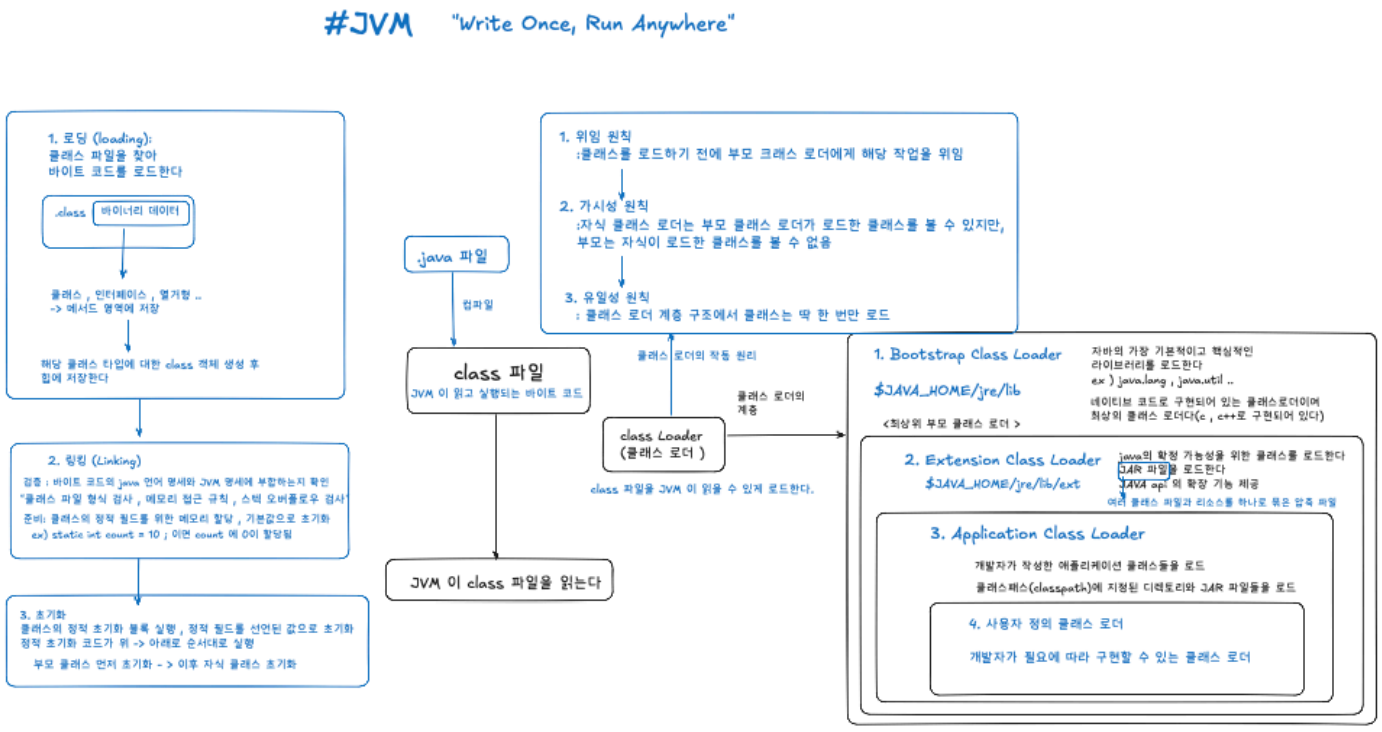
About JVM "Write Once, Run Anywhere"



▶️JVM 은 > ?

"자바가 어디서든 실행될 수 있게 해주는 가상 환경 이자 바이트코드(.class)를 해석하고 실행하는 가상 머신"

- JVM 이 class 파일을 읽고 스레드가 생성되고 , 어플리케이션이 실행되며 내부적으로 추가적인 컴파일 과정(JIT 컴파일)을 수행한다 .

▶️JVM 이 class 파일을 읽는 프로세스 흐름

class 파일은 ? JVM 이 읽는 '바이트 코드

JVM 이 바이트 코드를 읽으려면 "**클래스 로더"** 가 있어야 한다

🔍그럼 **클래스 로더**란 뭘까??

:JVM의 일부분으로 동작하는 서브시스템이며 , .class 파일(바이트코드)을 찾아서 JVM의 메모리로 로드하는 역할을 담당한다 .

class loader의 계층

1. Bootstrap class loader : 자바의 가장 기본적인 라이브러리를 로드한다 (최상위 계층 클래스 로더) ex) java.util , java.lang
2. extension class loader : java 의 jar 파일("Java Archive"의 약자로, 여러 클래스 파일과 리소스를 하나로 묶은 압축 파일 형식) 을 로드 한다
3. Application Class Loader : 개발자 들이 작성한 클래스들을 로드

🔍그럼 **클래스 로더**는 어떻게 작동 할까?

클래스 로더의 작동 원리

1. 위임 원칙 : 부모 클래스 로더가 해당 클래스를 찾지 못할 경우에만 자식 클래스 로더가 로드를 시도한다.
2. 가시성 원칙 : 부모 클래스로더는 자식 클래스로더가 로드한 클래스를 볼 수 없다
3. 유일성 원칙 : 계층 구조에서의 클래스는 딱 한번만 로드됨

클래스 로드 과정

1. 로딩 : 해당 클래스 파일의 바이트 코드를 로드 한다(클래스 , 인터페이스 등등 을 메서드 영역에 저장하고 해당 class 객체를 생성 후 *힙(hip)* 에 저장한다)
2. 링킹 : 바이크 코드의 java 언어명세와 JVM 명세에 부합하는지 확인 하며 , 메모리 접근 규칙 확인, 스택 오버플로우 방지를 위한 검사를 수행한다.
3. 초기화 : 클래스의 정적(static) 초기화 블록을 실행시킨다 , 즉 정적 필드를 선언된 값으로 초기화를 시킨다 이 과정 또한 부모 클래스부터 먼저 진행된다 .

▶️실질적인 java 코드로 알아보는 클래스로딩의 과정

```
public class ClassLoadingExample {
// 정적 변수 - 초기화 단계에서 값이 설정됨
static int staticCounter = 100;

// 정적 초기화 블록 - 초기화 단계에서 실행됩니다
static {
    System.out.println("1. 초기화 단계: 정적 초기화 블록 실행");
    System.out.println("   staticCounter = " + staticCounter);
    staticCounter = 200;
}

// 인스턴스 변수
private String name;

// 생성자 - 객체 생성 시 호출
public ClassLoadingExample(String name) {
    System.out.println("4. 객체 생성: 생성자 실행");
    this.name = name;
}


// 메인 메서드 - 클래스 로딩 완료 후 실행
public static void main(String[] args) {
    System.out.println("2. 클래스 로딩 완료!");
    System.out.println("3. 메인 메서드 실행 시작");
    System.out.println("   staticCounter = " + staticCounter);

    // 객체 생성
    ClassLoadingExample example = new ClassLoadingExample("테스트 객체");
    example.printName();
}

public void printName() {
    System.out.println("5. 인스턴스 메서드 호출: " + this.name);
}
}
```

<클래스 로딩 단계>

1. 로딩 단계: JVM이 ClassLoadingExample 클래스를 메모리에 로드
2. 링킹 단계: 바이트코드 검증, 메모리 준비 등이 수행

3. 초기화 단계: 정적 변수(staticCounter)가 초기화되고 정적 초기화 블록이 실행

<실행>

1. 메인 메서드 실행: 클래스 로딩이 완료된 후 main 메서드가 실행
2. 객체 생성: main 메서드 내에서 객체가 생성되고 생성자가 호출됨
3. 메서드 호출: 생성된 객체의 메서드가 호출