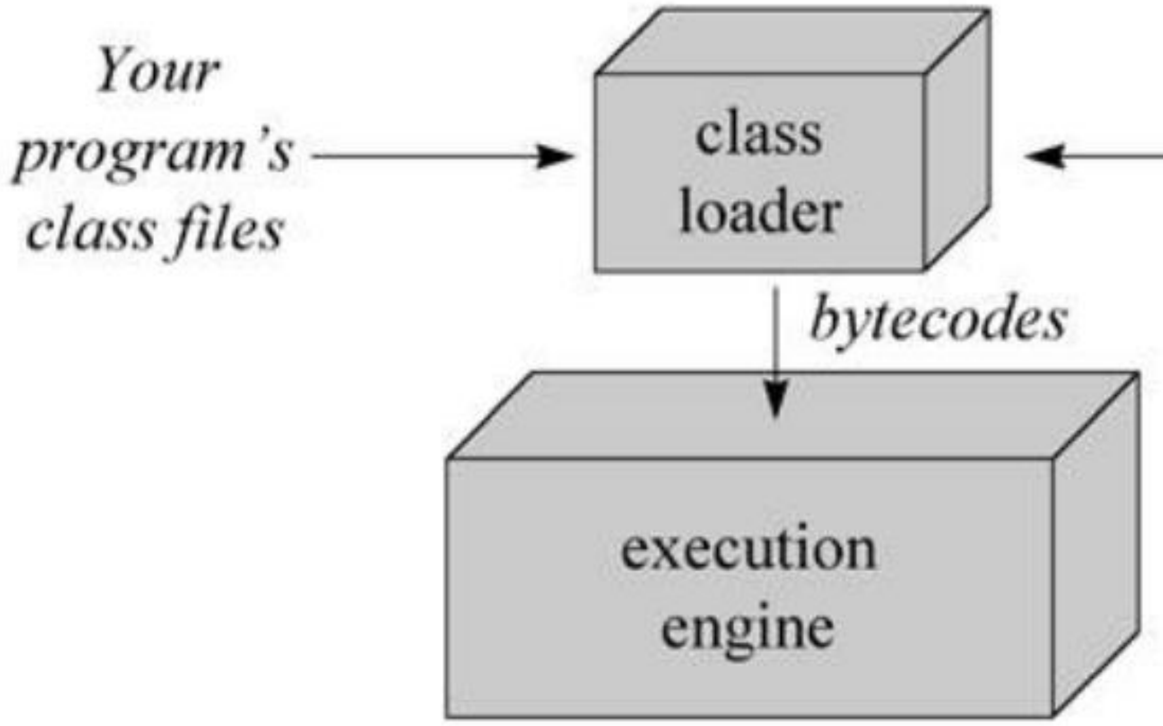
JVM

A Java Virtual Machineís main job is to load class files and execute the bytecodes they contain.



a **class loader**,
which loads class files from both the program and the Java API

The bytecodes are executed in an execution engine,
which is one part of the virtual machine that can vary in different implementations

The Class Loader Architecture

two types of class loaders

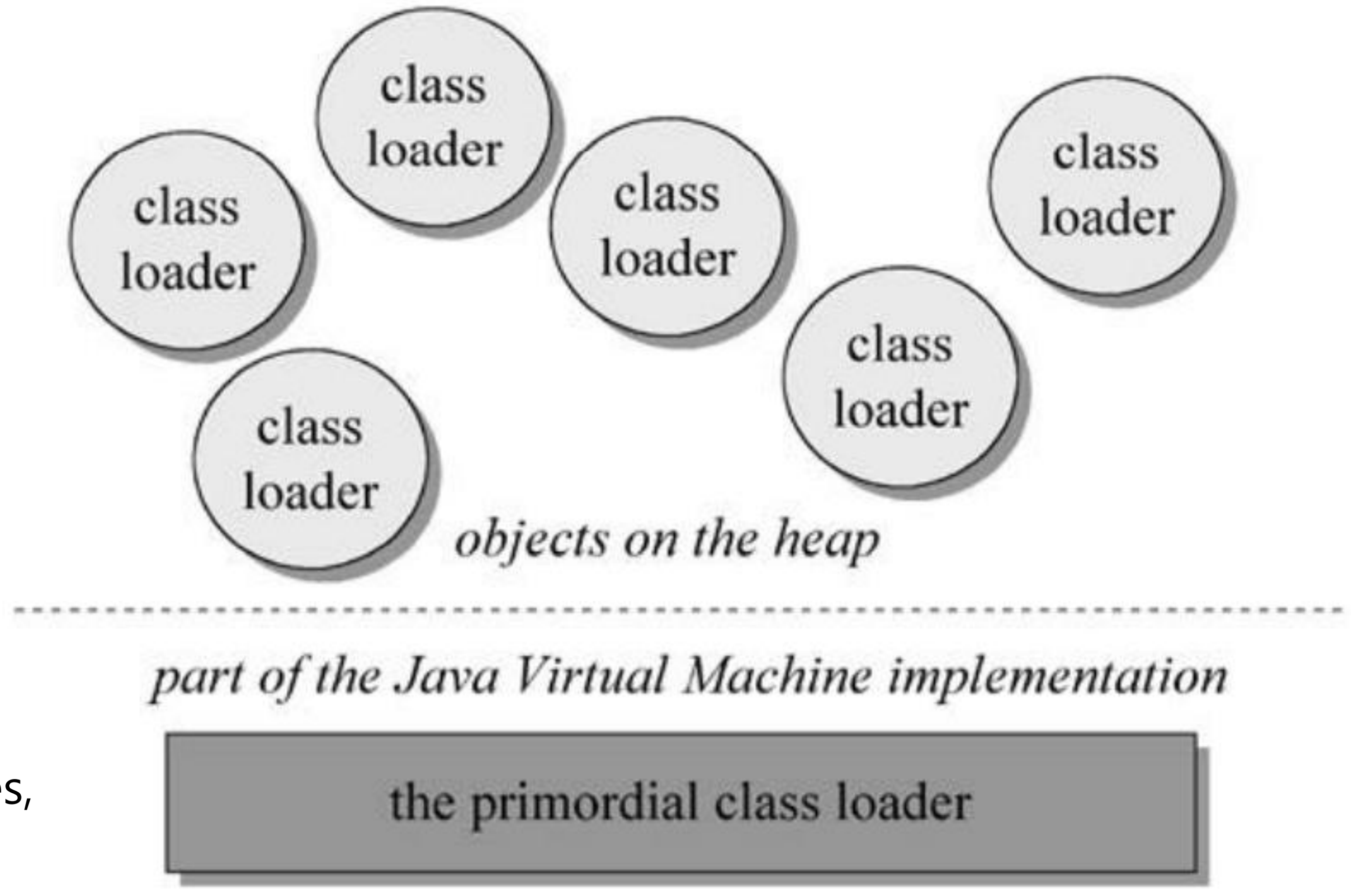
1 class loader objects.

- written in Java,
- compiled to class files,
- loaded into the virtual machine,
- instantiated just like any other object.

2 a "primordial" class loader

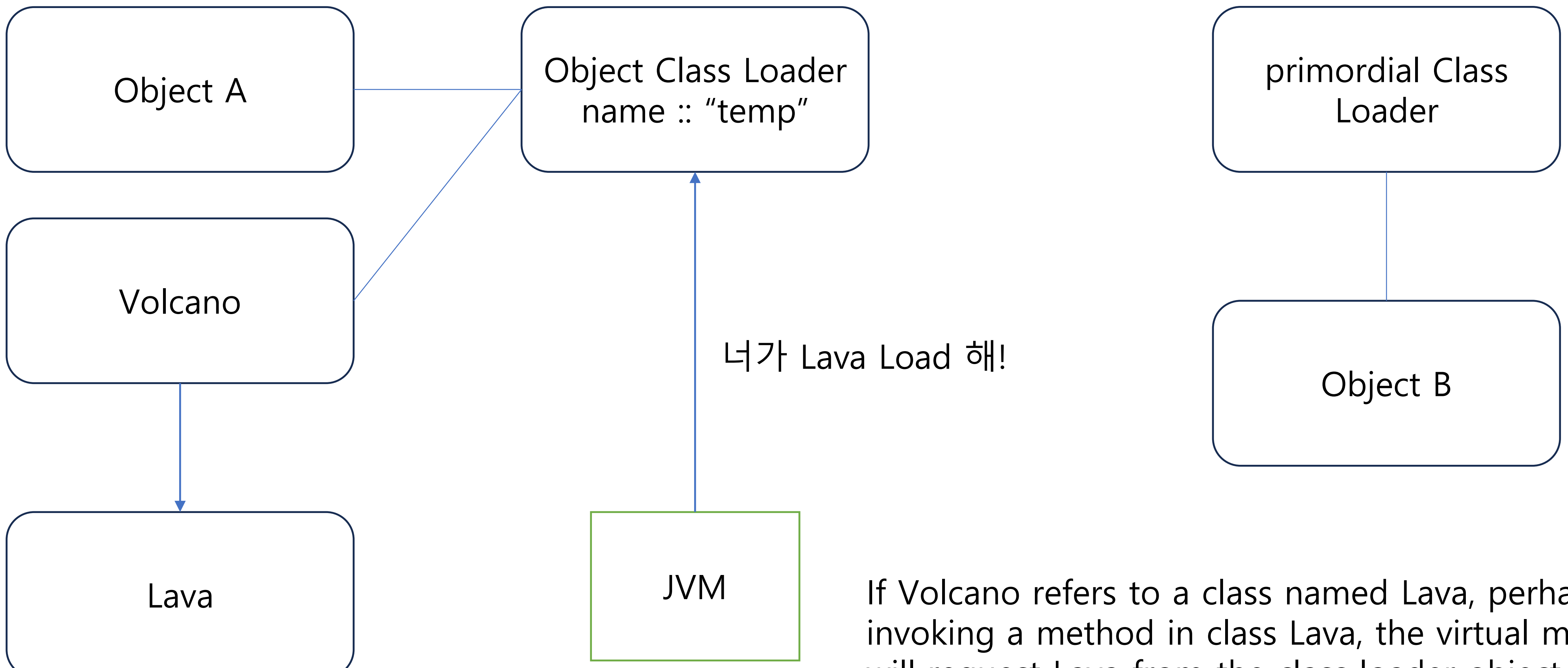
- primordial class loader loads trusted classes, including the classes of the Java API

They enable you to dynamically extend a Java application at run-time.



The Class Loader Architecture

For each class it loads, the Java Virtual Machine keeps track of which class loader--whether primordial or object--loaded the class.

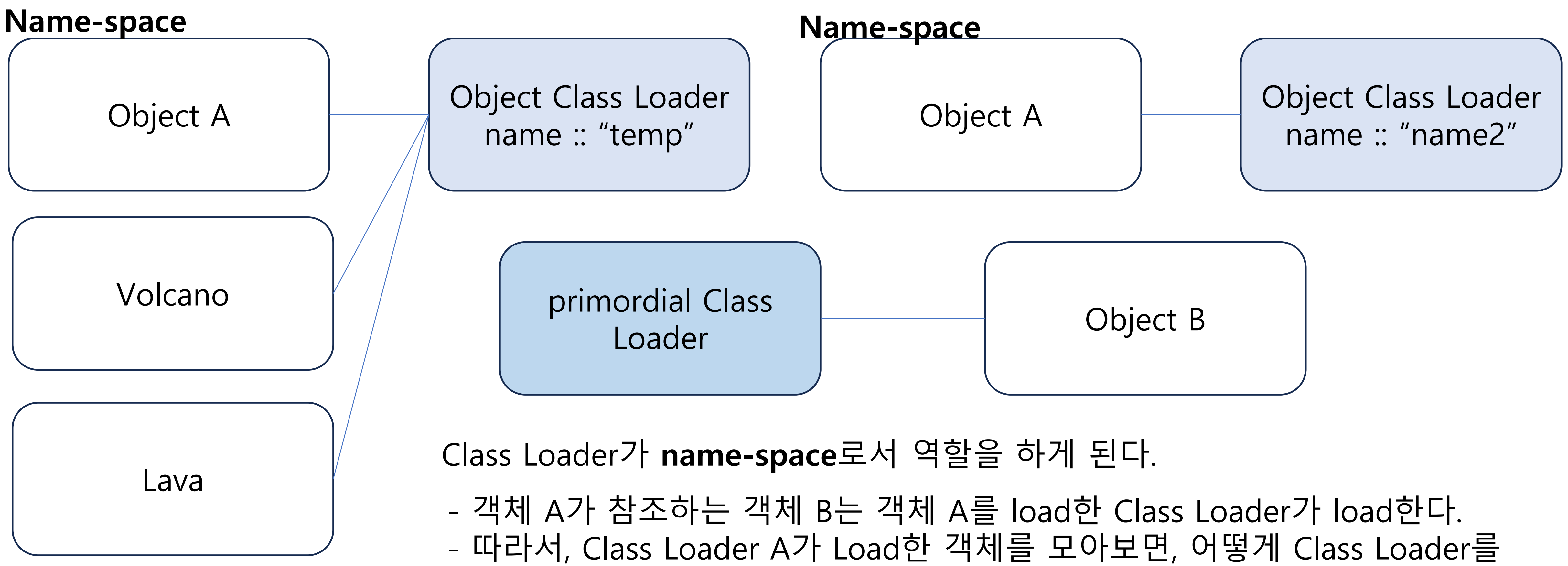


If Volcano refers to a class named Lava, perhaps by invoking a method in class Lava, the virtual machine will request Lava from the class loader object that loaded Volcano.

The Class Loader Architecture

Because the Java Virtual Machine takes this approach to loading classes, classes can by default only see other classes that were loaded by the same class loader

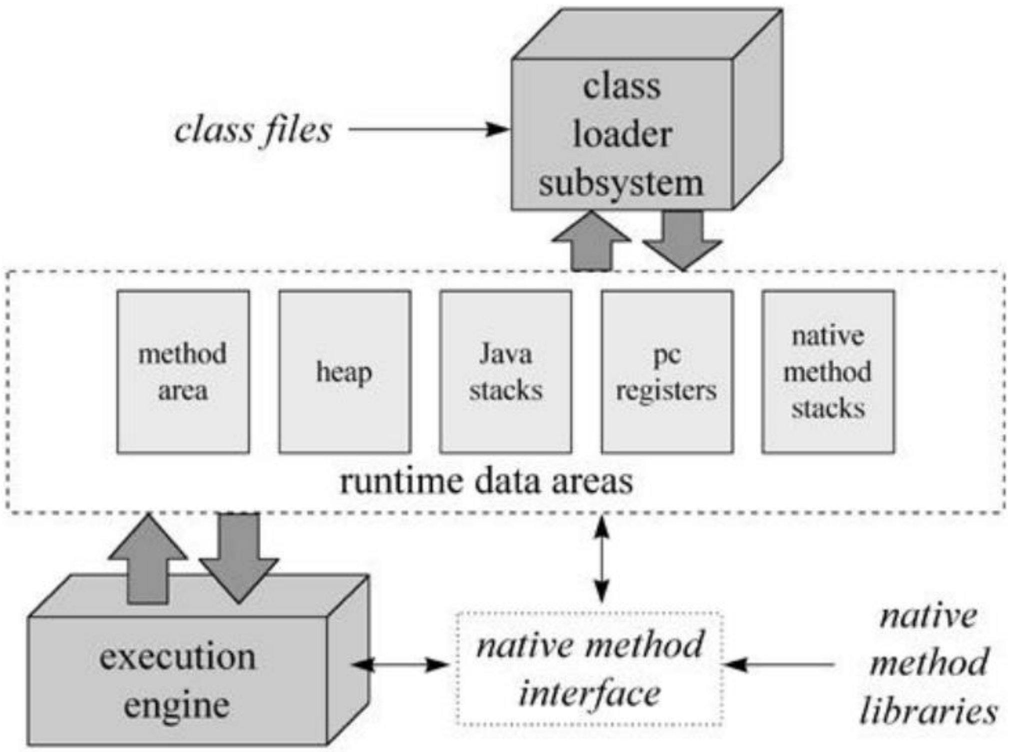
This is how Javaís architecture enables you to create **multiple name-spaces** inside a single Java application

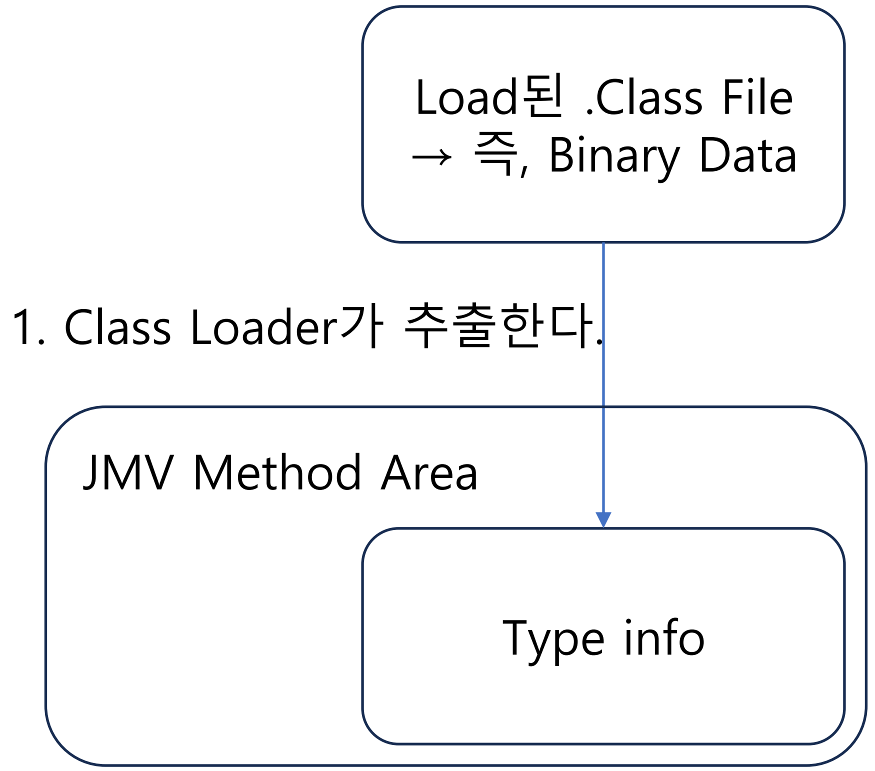


Class Loader가 **name-space**로서 역할을 하게 된다.

- 객체 A가 참조하는 객체 B는 객체 A를 load한 Class Loader가 load한다.
- 따라서, Class Loader A가 Load한 객체를 모아보면, 어떻게 Class Loader를 통해 name-space를 형성하는지 이해할 수 있다.

The Architecture of the Java Virtual Machine

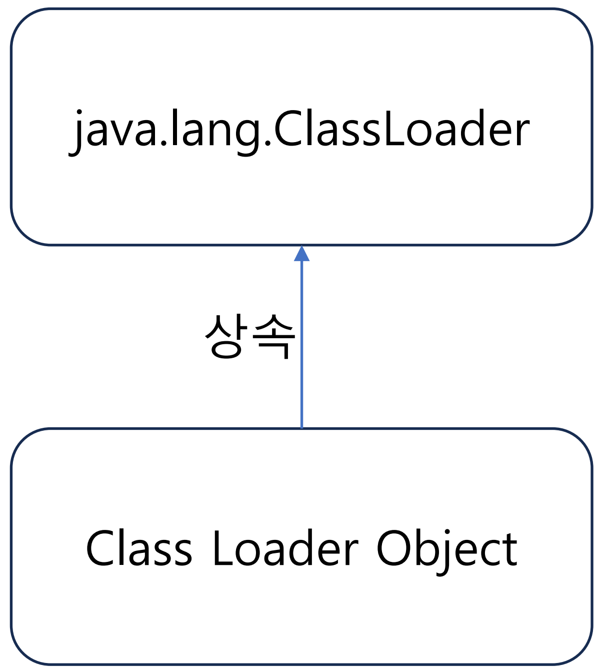




Step1.
When the virtual machine loads a class file, it parses information about a type from the binary data contained in the class file

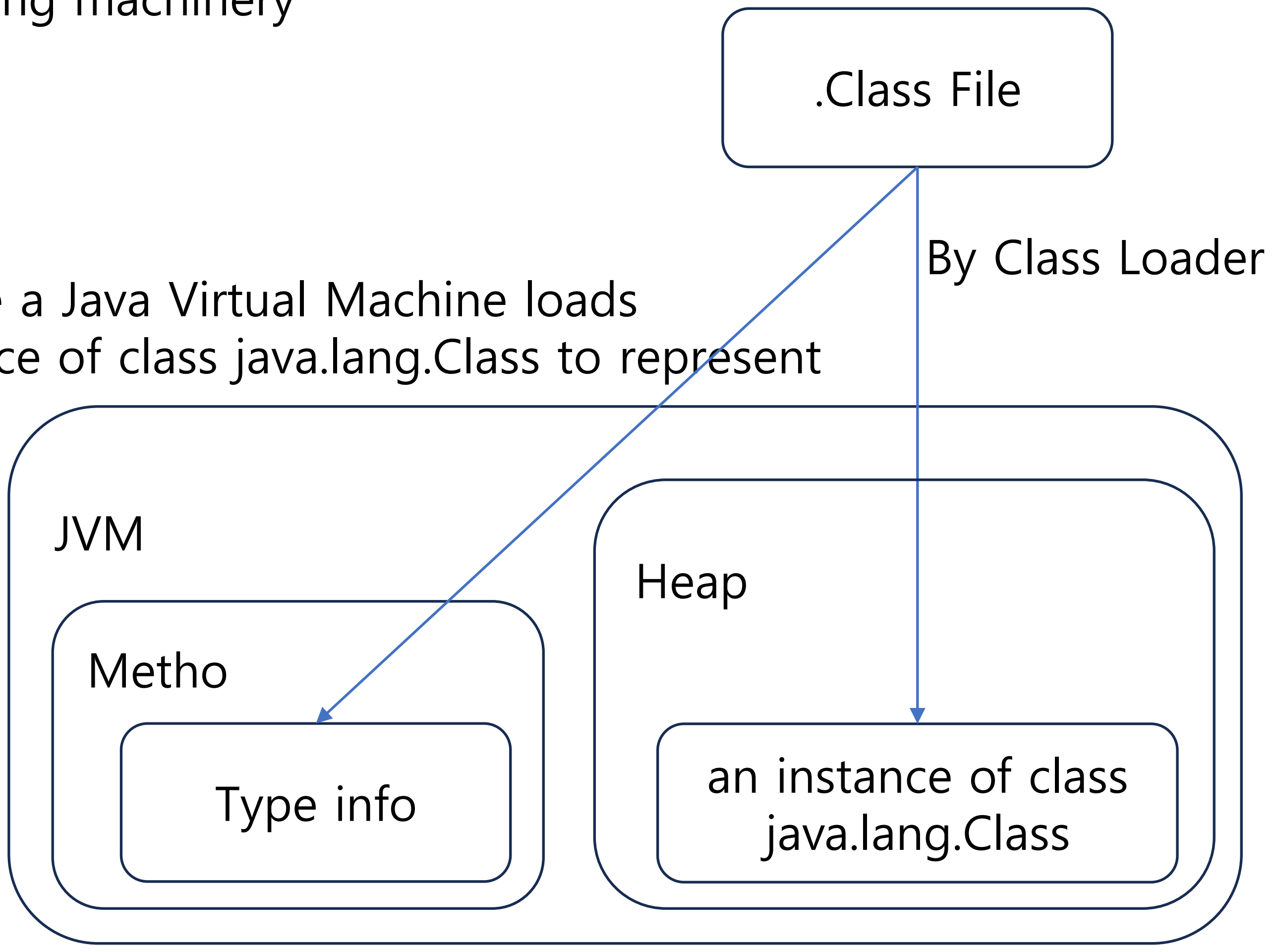
Step2.
It places this type information into the method area

The Class Loader



The methods of class ClassLoader allow Java applications to access the virtual machineís class loading machinery

Also, for every type a Java Virtual Machine loads it creates an instance of class java.lang.Class to represent that type.



Like all objects, class loader objects and instances of class Class reside on the heap. Data for loaded types resides in the method area.

Responsibility of the Class Loader

- locating and importing the binary data for classes
- verify the correctness of imported classes
- allocate and initialize memory for class variables
- assist in the resolution of symbolic references

Class Loader 동작 과정

Loading

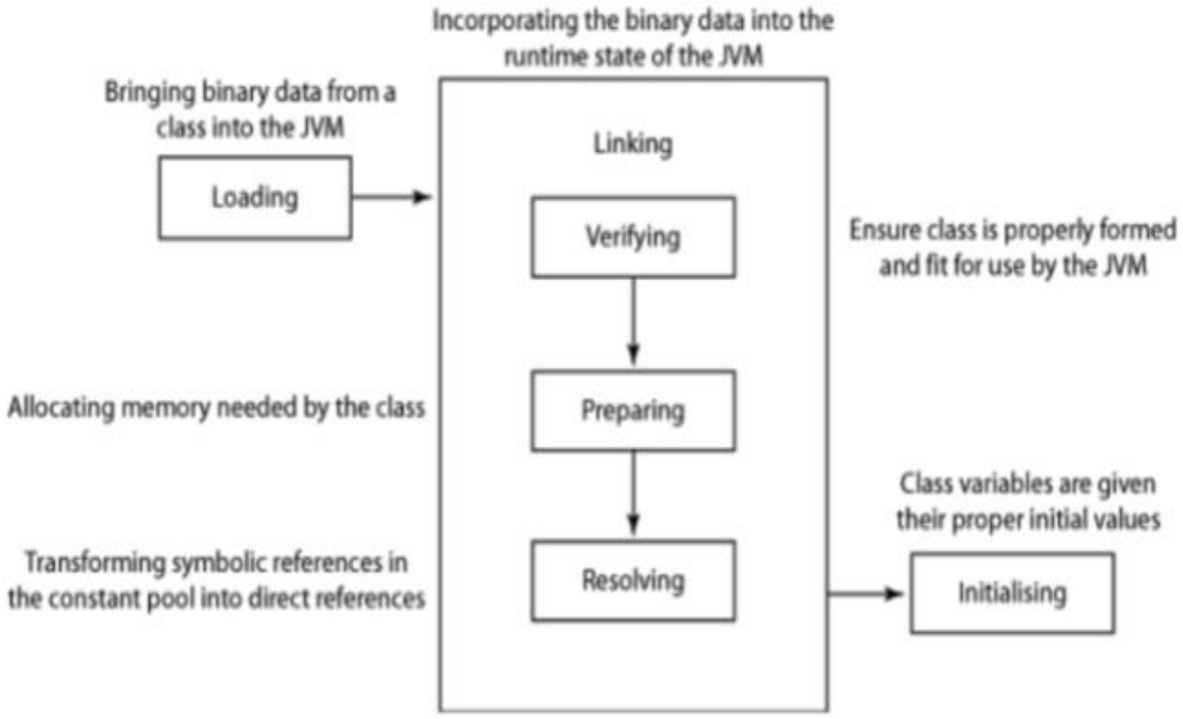
- finding and importing the binary data for a type

Linking

- Verification → ensuring the correctness of the imported type
- Preparation → allocating memory for class variables and initializing the memory to default values
- Resolution → transforming symbolic references from the type into direct references.

Initialization

- invoking Java code that initializes class variables to their proper starting values


Incorporating the binary data into the runtime state of the JVM
Bringing binary data from a class into the JVM
Loading
Linking
Verifying
Ensure class is properly formed and fit for use by the JVM
Allocating memory needed by the class
Preparing
Class variables are given their proper initial values
Transforming symbolic references in the constant pool into direct references
Resolving
Initialising

The Primordial Class Loader