→ CLASSES METADATA, LIKE:
- SIGNATURE
- CONSTANT POOL
- FIELD DESCRIPTOR

METHOD AREA SETUP

↳ PREPARE → DEFAULT INITIALIZATION

MEMORY ALLOCATION

→ STATIC VARIABLES ARE INITIALIZED WITH THEIR DEFAULT VALUE

→ PERFORMANCE RESOLUTION, JVM STRATEGY DEFINITION

CHECKS THE GENERATED BYTECODE IN:

↳ THE CHECKING IS MORE SAFETY FOCUSED, USING DIFFERENT TYPES OF ANLYSIS

- STRUCTURE
- JVM STANDARDS ← VERIFY   LINKING LAYERS   RESOLVE → SYMBOLIC REFERENCE RESOLUTION

↳ JAVA BYTECODE HAVE SYMBOLIC REFERENCES TO OTHER CLASSES, METHODS AND FIELDS.

- STRUCTURAL
- TYPE SAFETY
- CONTROL FLOW VERIFICATION

↳ ENSURE THE JUMPS AND BRANCHES ARE VALID

- SYMBOLIC REFERENCES ARE CONVERTED TO DIRECT REFERENCES
- DIRECT REFERENCES POINTS TO A MEMORY LOCATION, OR A METHOD IMPLEMENTATION

-JVM-
Class-Loader-
Subsystem

LOADING LAYERS

→ IT'S IMPLEMENTED IN C/C++

1: BOOTSTRAP CLASS LOADER → ROOT OF JVM'S CLASS LOADING

2: EXTENSION CLASS LOADER → RUNS FUNDAMENTAL CLASSES, LIKE:

3: APPLICATION CLASS LOADER

- JAVA.LANG
- JAVA.UTIL
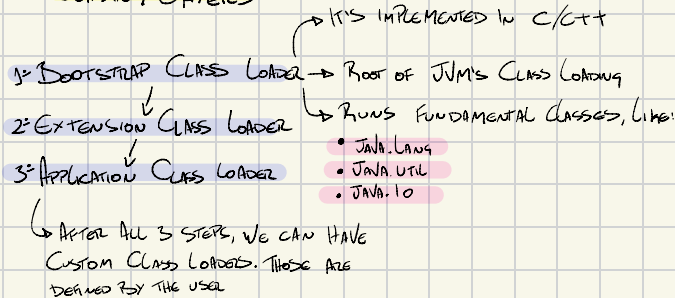- JAVA.IO

↳ AFTER ALL 3 STEPS, WE CAN HAVE CUSTOM CLASS LOADERS. THOSE ARE DEFINED BY THE USER

THE LAST LAYER IS THE INITIALIZATION