contents -

Core language feature,

Design Principles,

Project

Java is

Object oriented programming (real world)

Platform independent (write once, run anywhere)

Concurrent (multi-threading)

Safer

Automatic memory management (garbage collection)

Security

Compilation

Computer – fix set of instructions

Program – set of instructions containing 0,1

Called machine language

Assembly language is low level, programmer readable

Eg add#01, #02

Machine lang 🡨 assembler 🡨 assembly lang

Java (high level)

, source code 🡪 compiler 🡪 target lang (can be machine lang)

Compiler steps

Verify syntax, semantics

Code optimizations

Generate machine code

Machine code generated on platform1 cannot be executed on platform2

Compiler machine code for different platform are different

Machine code🡪 CPU 🡪result

Source code 🡪 interpreter 🡪 result

Interpreter, virtual machine

Platform independence

For all source code,

like add – there’s complied steps to execute in it library

advantages

platform independence

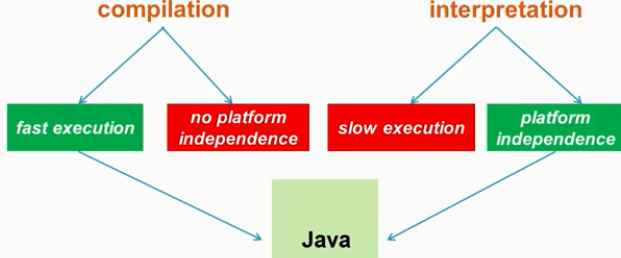
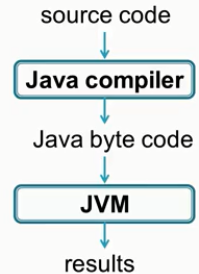
easier to update

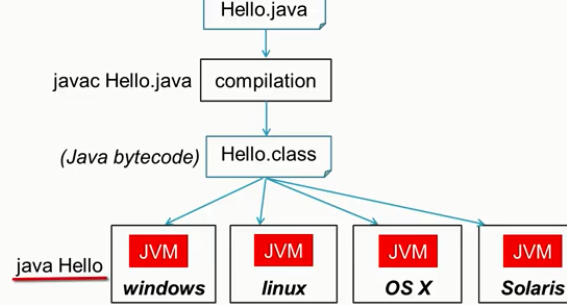
dis advantages

slow, costly access

source code, reinterpreted every time

interpreter is loaded into memory

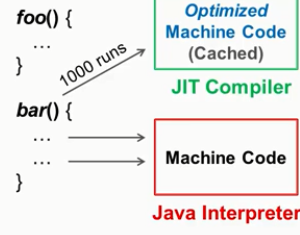
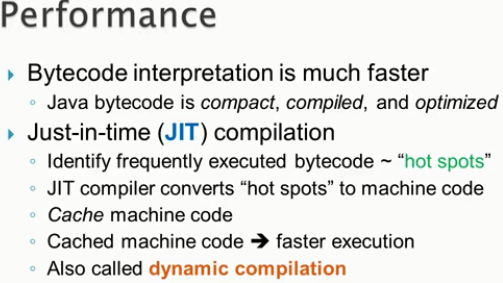
java takes best of both

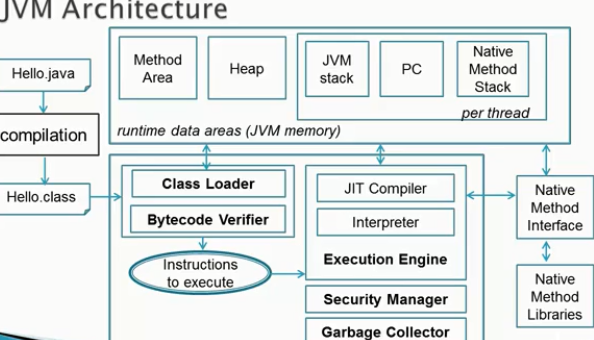


JAVA BYTE CODE IS optimized, compiled, compact

So interpretation by jvm subcomponent is faster

Also JIT just in time compilation

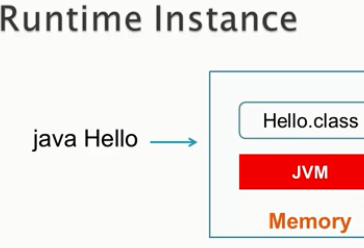




Jvm – ABSTRACT COMPUTING MACHINE

Java bytecode is i/p

Auto memory management

 a jvm instance is created and loaded into memory for execution of only one class file at a time.

Java SE Java Software Edition, standalone applications

Java EE Java Enterprise Edition, servers

Java ME, resources constrained devices

JAVA system variables

JAVA\_HOME =…path to jdk or jre

PATH = %JAVA\_HOME%/bin

JAVA classpath variables

Classpath = .; //ensures jvm search and execute java files in the current folder