Java Virtual Machine 101

•••

Sarath Soman
https://sarathsoman.wordpress.com/
@sarathsom4

About Me

- Software Engineer @ Hifx
- Ported SlangForDotNet to JVM
- Added Lambda expression support and local variable type inference to SLANG
- Writing a Matrix and Stat library and trying to implement machine learning algorithms on top of that
- Started writing LKMs and interfacing C/C++ with Java, C# and Python
- At work I write Scala & Java code

Why you should learn VM internals ?

3 types of programmers

Application programmer – Is one who has knowledge in commonly used frameworks, libraries and common design patterns

Library designer – Is one who has knowledge in DSA, design patterns / principles and API design

System programmer – Is one who has knowledge in architecture of the VM / cpu, native code programming and cross platform programming

Where do you belong now ?

If you want to remain as a average application developer you don’t need to know anything other than the commonly used frameworks and libraries in the industry.

Specialization is for insects.

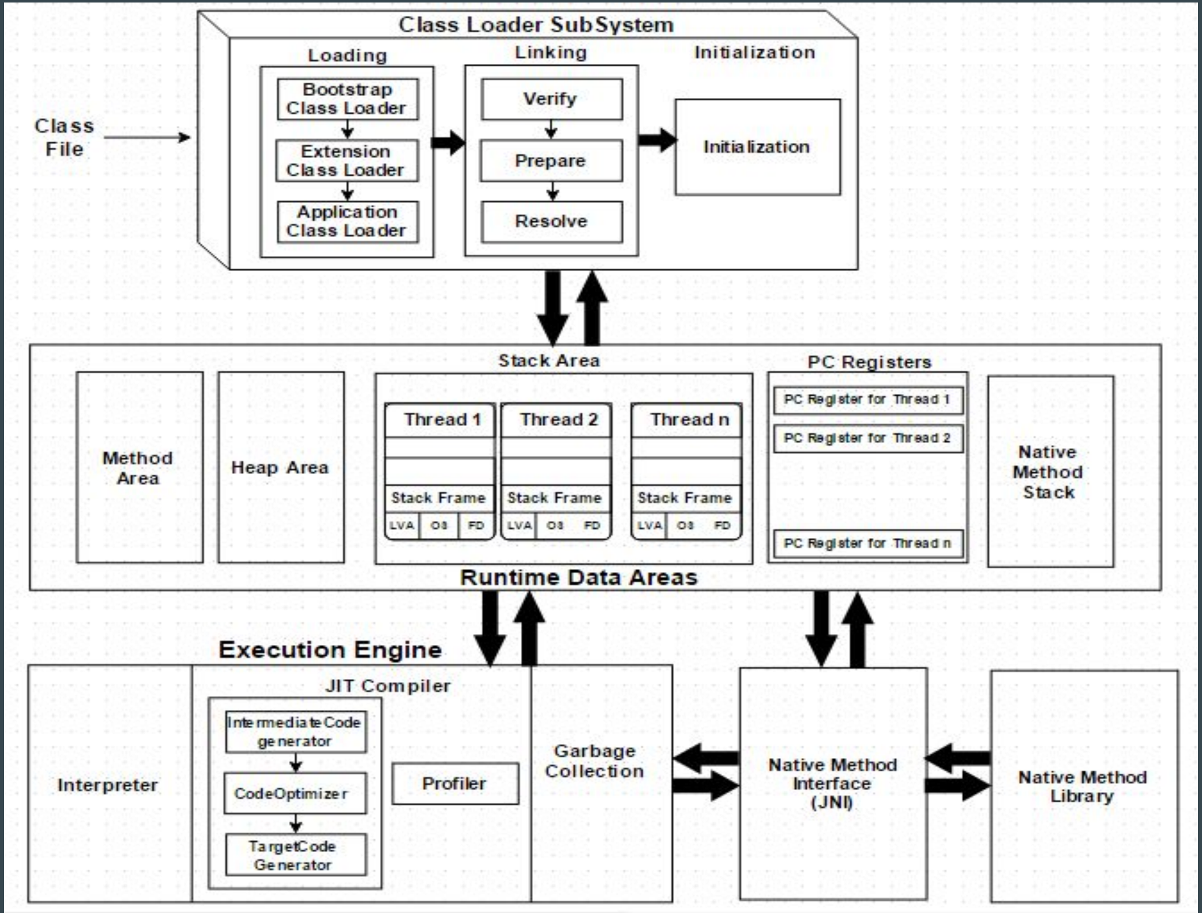
A human being should be able to change a diaper, plan an invasion, butcher a hog, conn a ship, design a building, write a sonnet, balance accounts, build a wall, set a bone, comfort the dying, take orders, give orders, cooperate, act alone, solve equations, analyze a new problem, pitch manure, program a computer, cook a tasty meal, fight efficiently, die gallantly. Specialization is for insects.

-Robert A. Heinlein

0xcafebabe

- Minor & Major version
- Access Flags
- Constant Pool
- Field Instructions
- Method Instructions

JVM Architecture



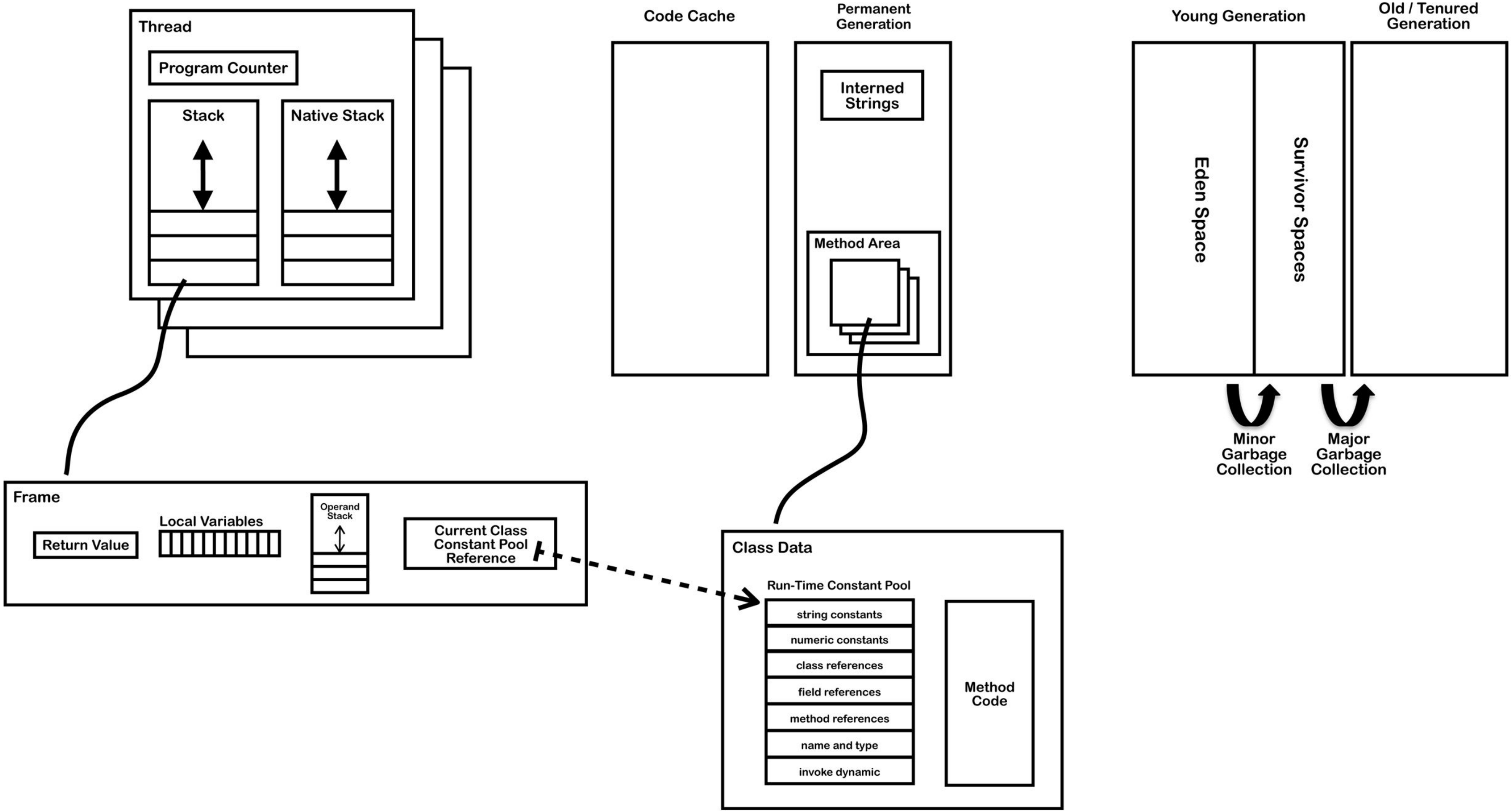
ClassLoading Procedure

Execution Engine

Stack

Non Heap

Heap



Thread

Program Counter

Stack

Native Stack

Frame

Return Value

Local Variables

Operand Stack

Current Class Constant Pool Reference

Code Cache

Permanent Generation

Interned Strings

Method Area

Class Data

Run-Time Constant Pool

- string constants
- numeric constants
- class references
- field references
- method references
- name and type
- invoke dynamic

Method Code

Young Generation

Eden Space

Survivor Spaces

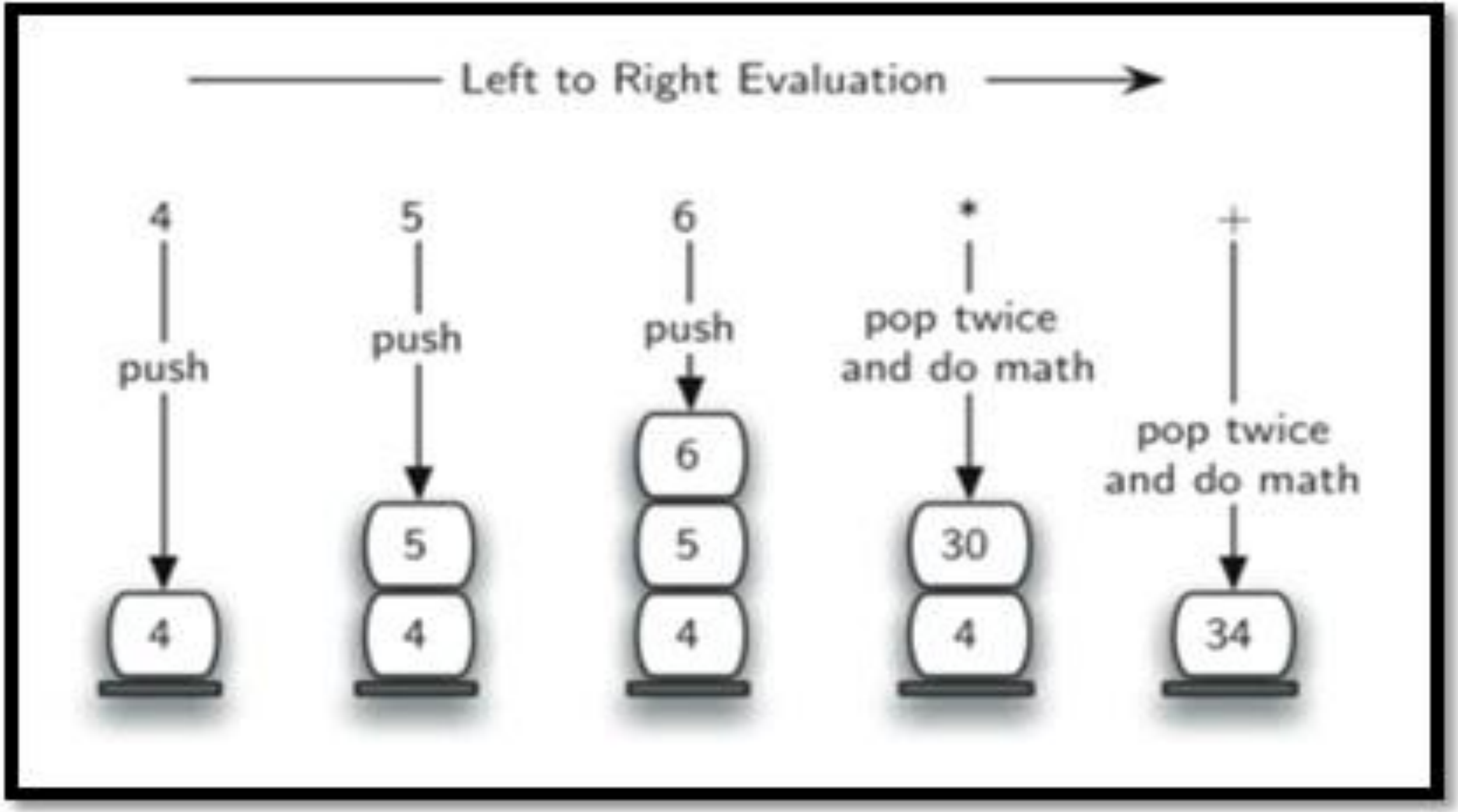
Old / Tenured Generation

Minor Garbage Collection

Major Garbage Collection

Postfix arithmetic expression evaluation

Ex: 4 5 6 * +


Left to Right Evaluation
4
push
4
5
push
5
4
6
push
6
5
4
*
pop twice
and do math
30
4
+
pop twice
and do math
34

JVM Instruction Set

Load & Store

- load
- store
- ldc

Method invoke & ret

- invokevirtual
- invokestatic
- invokeinterface
- invokespecial
- invokedynamic
- return

Others

- putstatic
- getstatic
- putfield
- getfield
- div
- add
- mul
- ....

JVM, CLR, Dalvik comparison

- Register based vs Stack based

What happens when Method A->x() calls Method B->y()

Language Constructs, translation and execution

Bytecode engineering

Modifying existing bytecode - Proxy Obj, AOP

Creating bytecode from scratch - Scala, Spring Data JPA, Slang4JVM

Examples - Generating bytecode

Method Handles & Invokedynamic

- JSR 292
- Dynamic languages on JVM
- Lambda expressions

Thank You

Links

- https://archive.codeplex.com/?p=slangfordotnet
- https://github.com/nsarathsoman/SlangCompiler
- https://github.com/nsarathsoman/slango
- https://github.com/nsarathsoman/FAYA-JVM-101