AJCSE
ASIAN JOURNAL OF COMPUTER SCIENCE ENGINEERING

www.ajcse.info

Asian Journal of Computer Science Engineering 2016;1(1):1-3

REVIEW ARTICLE

A Brief study on JVM

*Aayush Mehta[1],Akriti Saxena[2],Tanmay Bhawsar[3]

CSE Department Mandsaur Institute of Technology Mandsaur [M.P.]

Received on: 04/06/2016, Revised on: 08/09/2016, Accepted on: 15/09/2016

ABSTRACT

This paper describes the working and architecture of Java Virtual Machine (JVM). Our research on JVM architectures that how a program gets space in the JVM. The JVM provides platform independent property to any java program. That means; JVM provides "Compile once and Execute anywhere" technique We describe the basic architecture of JVM here.

Keywords- Oak, byte code, optop, CORBA, LDAP, WORA.

INTRODUCTION

Objective:-

The objective of this research work is to describe the basic architecture of JVM. Its memory management and the flow of execution of a program.

Java:-

Java is an object oriented programming language developed by James Goslings, Patrick Naughton, Chris warth, Ed fronk and Mike Sherfidan in Sun Microsystems inc. in 1991. It took 18 months to develop the first working version. This language was initially called "OAK", but was renamed "java" in 1995. The primary motivation was the need for a platform independent language that can be used to create embedded softwares in a cost effective way.

Features of Java:-

- Compiled and Interpreted.
- Portable.
- Simple Language.
- Platform independent.
- Distributed.
- Robust language.
- Dynamic and Extensible.

JVM (Java Virtual Machine)

A **Java virtual machine** (**JVM**) is an abstract computing machine that enables a computer to run a Java program. There are three notions of the JVM specification, implementation and instance. The specification is a document that formally describes what is required of a JVM implementation. Having a single specification ensures all implementations are interoperable. A JVM implementation is a computer program that meets the requirements of the JVM specification. An instance of a JVM is an implementation running in a process that executes a computer program compiled into Java bytecode.

Java Runtime Environment (**JRE**) is a software package that contains what is required to run a Java program. It includes a Java Virtual Machine implementation together with an implementation of the Java Class Library. The Oracle Corporation, which owns the Java trademark, distributes a Java Runtime environment with their Java Virtual Machine called HotSpot.

Java Development Kit (**JDK**) is a superset of a JRE and contains tools for Java programmers, e.g. a javaccompiler. The Java Development Kit is provided free of charge either by Oracle Corporation directly, or by the OpenJDK open source project, which is governed by Oracle.

JVM specification

The Java virtual machine is an abstract (virtual) computer defined by a specification. This specification omits implementation details that are not essential to ensure interoperability: the memory layout of run-time data areas, the garbage-collection algorithm used, and any internal optimization of the Java virtual machine instructions (their translation into machine code). The main reason for this omission is to not unnecessarily constrain implementers. Any Java application can be run only inside some concrete implementation of the abstract specification of the Java virtual machine.

Starting with Java Platform, Standard Edition (J2SE) 5.0, changes to the JVM specification have

***Corresponding Author:** Aayush Mehta, **Email:** aayushmehta95@gmail.com

been developed under the Java Community ++++++-Process as JSR 924. As of 2006, changes to specification to support changes proposed to the class file format (JSR 202) are being done as a maintenance release of JSR 924. The specification for the JVM was published as the *blue book*, The preface states: We intend that this specification should sufficiently document the Java Virtual Machine to make possible compatible clean-room implementations. Oracle provides tests that verify the proper operation of implementations of the Java Virtual Machine.

One of Oracle's JVMs is named HotSpot, the other, inherited from BEA Systems is JRockit. Clean-room Java implementations include Kaffe and IBM J9. Oracle owns the Java trademark and may allow its use to certify implementation suites as fully compatible with Oracle's specification.

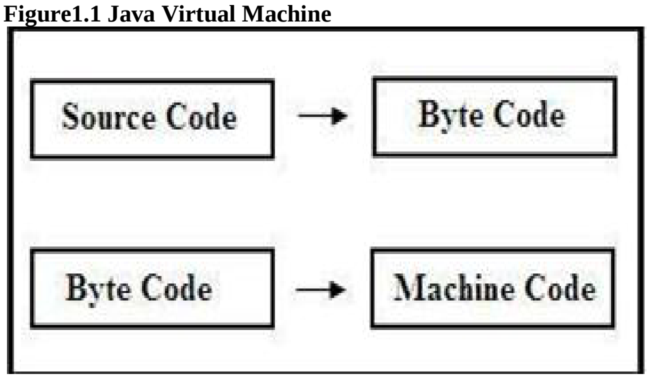
Memory Areas for JVM:

- Method area.
- Class description.
- Constant pool.
- Heap.
- Garbage collection.
- Stack.

JVM Architecture:

The JVM is basically a stack based machine, with a 32 bit word size, using 2‟s complement arithmetic. JVM is an efficient way of getting memory protection on simple architectures that lack an MMU (Memory Management Unit). This is analogous to managed code in Microsoft„‚s .NET Common Language Runtime.

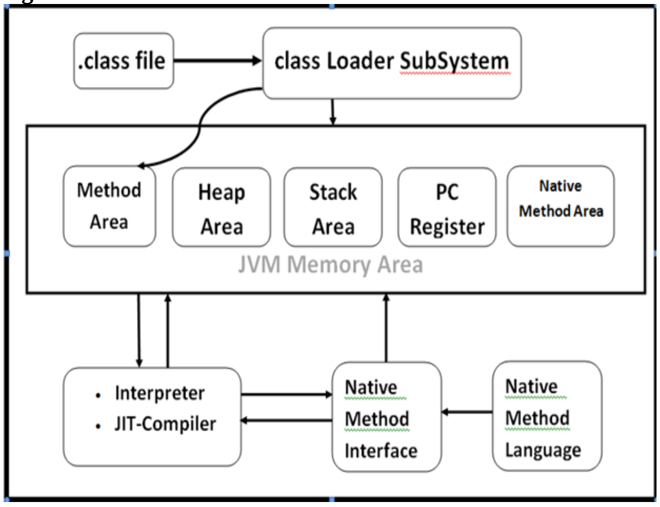
Figure1.1 Java Virtual Machine



- Optop: Pointer to the top of the operand stack for the currently active method.
- Frame: Pointer to the stack frame of the currently active method.
- Vars: Pointer to the beginning of the local variables of the currently active method.

The JVM Architecture is very much stack oriented. Most instruction access the stack in some way. The stack is also used for method calls, as with many classical machine architectures. A method call produces a new stack frame, which is pushed onto the stack and return pops the frame from the program‟s stack. Almost all JVM instructions are stack based. In the JVM specification, the behavior of a virtual machine instance is described in terms of subsystem, memory areas, data types, and instruction. These components describe an abstract inner architecture for the abstract java virtual machine. It is more to provide a way to strictly define the required behavior of implementations. The Specification defines the required behavior of any java virtual machine implementation in term of these abstract components and there interaction.

Figure 1.2 Internal Architecture of JVM



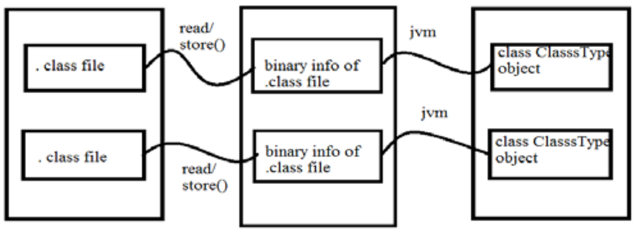
1) Class Loader Subsystem

- Special Java runtime objects that are used for loading Java classes into the Java Virtual Machine.
- They provide JVM with functionality similar to the one of a dynamic linker.
- Each class Loader defines a unique namespace.
- For every class loaded into JVM a reference to its class Loader object is maintained.

a) Class Loader Types:

- Bootstrap Class Loader.
- Extension Class Loader.
- Application Class Loader.

Figure 1.2 Internal working of class loader subsysstem



AJCSE, Sep-Oct, 2016, Vol. 1, Issue 1

2) Method Area:

- Area where all the classes loads.
- Classes loads in the class context area of the method area.

3) Stack Area:

- Area used to store the local variables.
- Used to store the values of the intermediate variables.
- Here only the reference variable of the object is created.
- When any new variables encounter then it gets PUSH into the Stack.
- When the scope of the variable gets over then the memory gets destroyed and it directly POP from the Stack.

4) Heap Area:

- Area where the Dynamic memory allocation takes place.
- Here the object gets created , and it gets memory.
- As the new object created then it gets memory into the Heap.
- Size of Heap is also not static. It get increased as according to the memory allocation.
- As the work of object is over then with the help of the garbage collector all the memory occupied by that object is released.

5) PC Register(Program Counter):

- It is a register used to store the address.
- It stores the address of the next Instruction to be executed..

6) Native Method Area:

- This memory area is reserved for the native methods.
- Methods that we used by inheriting from its parent technology is stored here.

7) Execution Engine:
A) Interpreter:

- Used for step by step execution.
- Saves memory.
- Time consuming.
- So in java JIT compiler added which overcomes the problem of fast execution.

B) JIT Compiler:

- Just in Time Compiler is a special feature of java.
- also known as **dynamic translation.**
- It first Compile the program into the related Intermediate code.
- Then after interpretation takes place.
- Its uses the properties of the compiler as well as interpreter.

CONCLUSION

According to the study of JVM, JVM provide us the facility of portable code, which support the mechanism of WORA (Write Once, Run Anywhere).JVM is Stack based Architecture, so the operation like push and pop takes place.Here heap is also used so that runtime memory allocation can be implemented. Methods from native language is also inherited here so we can used the properties of native language.

ACKNOWLEDGMENT

We would like to show our gratitude to our CSE lecturer miss. Priyanka Mangal mam for her support and efforts.

REFERENCE

1. T. Suganuma, T. Ogasawara, M. Takeuchi, T. Yasue, M. Kawahito, K. Ishizaki, H. Ko-matsu, and T. Nakatani,
2. The Last Stage of Delirium Research Group ,Poland, "Java and Java Virtual Machine security vulnerabilities and their exploitation techniques.", October 3rd - 4th 2002.
3. J. Meyer and T. Downing., "Java Virtual Machine." O"Reilly, 1997.
4. James Gosling, Bill Joy, Guy Steele, and GiladBracha,"TheJavaLanguageSpecification",Addison-Wesley,2000,JavaSpec.
5. E. Balaguruswami, "Programming with java", Tata McGraw-Hill
6. http://www.java.sun.com/docs/books/vmspec/.
7. http://www.bitpipe.com/tlist/Java-Virtual-Machine.html.
8. http://www.artima.com/insidejvm/ed2/jvm2.html.
9. http://www.javacoffeebreak.com/articles/inside_java/insidejava-jan99.html.
10. http://java.sun.com/javase/technologies/core/jndi/index.jsp.
11. http://en.wikipedia.org/wiki/Java_Virtual_Machine.
12. http://java.sun.com/javase/technologies/core/jndi/index.jsp.
13. http://en.wikipedia.org/wiki/Java_Virtual_Machine.