

Objects and Method Calling in Java Virtual Machine

Sumit Gupta
Department of CSE
MANIT BHOPAL
MP, India

Nargish Gupta
Department of CSE
ASET, Amity University
Sector-125, Noida, UP, India

Rishabh Gupta
Department of CSE
ASET, Amity University
Sector-125, Noida, UP, India

ABSTRACT

Java virtual machine is interpreters for byte code in this paper we have discuss the internal architecture of JVM which shows that how objects go into the java heap and how method calling goes into the java stack. In this paper we also discuss the java security and process of how java programs run step by step inside JVM.

Keywords

JVM, Java, Java Heap, Java Stack, Java Virtual Machine.

1. INTRODUCTION

JVM stands for java virtual machine, JVMs available for many hardware and software platforms (i.e. JVM is platform dependent). JVM is nothing it is only on paper written instruction. Java is very secure because of its sandbox security model. The local code is directly interacting to JVM but remote code first interacts to the sandbox security. Sandbox security check whether the code is trusted or not, if code is trusted then sandbox security allows interacting with JVM, otherwise restricted the code [6].

The JVM perform four main tasks:

- Loading of codes
- Verification of codes
- Execution of codes
- Provides runtime environment.

1.1 Hierarchy of Java Program

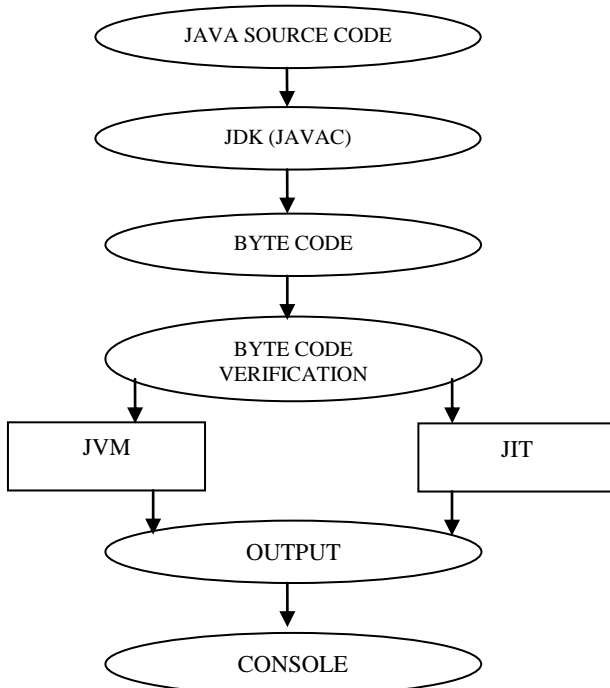


Fig 1: Hierarchy of java program

2. PROCESS

In the java programming language, all source code is first written in plain text files ending with .java extension. Those source files are then compiled in to .class files by the javac compiler. A .class file does not contain code that is native to your processor; it instead contains byte code the machine language of java virtual machine because of the java virtual machine available on many different operating systems. The same .class file is capable of running on Microsoft windows, the Solaris operating system, Linux, or Mac operating system. After the successful compilation of java program the byte code is generated, and it is platform independent. The byte code is executed through interpreter line by line which is very slow.

To overcome this problem sun micro system introduces a new compiler which is known as JIT (just in time) compiler which executes whole instruction in one goes [1]. Java Virtual Machine contains a byte code verifier to check the code for type errors before it is run, after successful verification of byte code by byte code verifier [2], JVM run program through java runtime environment (JRE) and JRE is the implementation of JVM [8].

2.1 Process Diagram of JDK and JRE

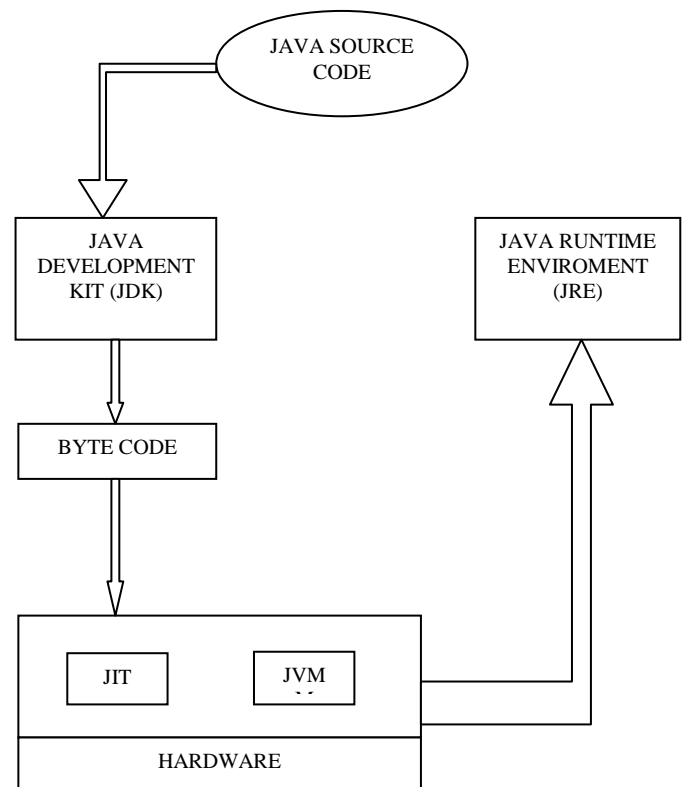


Fig 2: Process diagram of JDK and JRE

2.2 Inside JVM Process

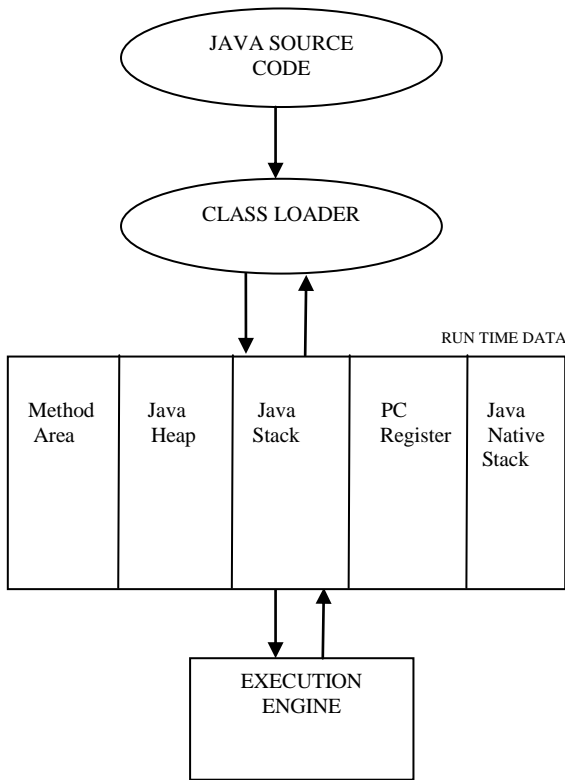


Fig 3: Inside JVM Process

2.2.1 Class loader

Class loader is subsystem of JVM that is used to load class files. There are basically three types of class loaders,[7]

- **Bootstrap Class Loader**
Bootstrap class loader loads java's core classes like java.lang, java.util etc.
- **Extensions Class Loader**
Extensions class loader loads classes from this ext folder.
- **System Class Loader**
Java classes that are available in the java class path are loaded using System class loader.

2.2.2 Method Area

Method area contains all the java expressions.

2.2.3 Java heap

All objects which are created through new keyword go into java heap [4].

2.2.4 Java stack

The calling of method goes into java stack [3].

2.2.5 Pc register

Pc register hold what is the next instruction to be executed.

2.2.6 Java native stack

In java native stack, those codes which are not written inside the java are executed.

2.2.7 Execution Engine

Execution Engine Processor contains a virtual processor.

3. EXPERIMENT

3.1 Understanding the Exception in thread Main

```

class HeapFull{
    public static void main(String args[])
    {
        int[] a=new int[300000000];
    }
}
  
```

If the heap size is less as compared to given array i.e. 300000000 than the Exception in main "Out-Of-Memory-error" occurred which shows the java heap space is full. Heap size should be half of the RAM.

3.2 Understanding stack-over-flow Exception

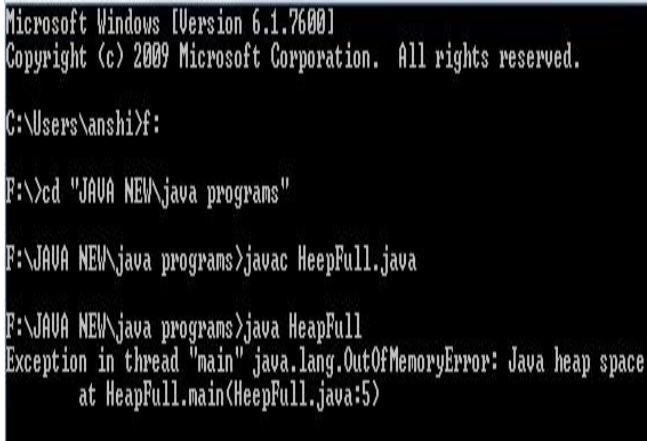
```

class Stackoverflow
{
    public void fun()
    {
        System.out.println("i am from fun====>>>>");
        fun1();
    }
    public void fun1()
    {
        System.out.println("i am from fun1====>>>>");
        fun();
    }
    public static void main(String args[])
    {
        Stackoverflow o=new Stackoverflow();
        o.fun();
    }
}
  
```

In the output of the program java.lang.stackoverflow Exception is occurred which shows the method calling goes into the java stack.

4. RESULTS

4.1 Heap Size is full



```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\anshi>f:

F:\>cd "JAVA NEW\java programs"

F:\JAVA NEW\java programs>javac HeapFull.java

F:\JAVA NEW\java programs>java HeapFull
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
at HeapFull.main(HeapFull.java:5)
```

Fig 4: Heap size is Full

4.2 Stack overflow



```
at Stackoverflow.fun1(Stackoverflow.java:13)
at Stackoverflow.fun1(Stackoverflow.java:6)
at Stackoverflow.fun1(Stackoverflow.java:13)
at Stackoverflow.fun1(Stackoverflow.java:6)
at Stackoverflow.fun1(Stackoverflow.java:13)
at Stackoverflow.fun1(Stackoverflow.java:6)
at Stackoverflow.fun1(Stackoverflow.java:13)
at Stackoverflow.fun1(Stackoverflow.java:6)
at Stackoverflow.fun1(Stackoverflow.java:13)
at Stackoverflow.fun1(Stackoverflow.java:6)
at Stackoverflow.fun1(Stackoverflow.java:13)
at Stackoverflow.fun1(Stackoverflow.java:6)
at Stackoverflow.fun1(Stackoverflow.java:13)
at Stackoverflow.fun1(Stackoverflow.java:6)
at Stackoverflow.fun1(Stackoverflow.java:13)
```

Fig 5: Stack overflow

5. CONCLUSION

With the help of this paper we understand the step by step process for execution of the program inside the java virtual machine. In this paper we also understand the java security through sandbox security model. With the help of both the experiments, we also understand the internal architecture of java

virtual machine which shows that all objects goes into the java heap and calling of methods goes into the java stack. When the size of memory is less than the given value into the array then OutOfMemory Exception occurs which shows that heap size is full and if the java stack is full then exception stack-over-flow occur which shows that the calling of methods goes into the java stack.

6. REFERENCES

- [1] N. Shaylor. A “Just-in-Time Compiler for Memory-Constrained Low-Power Devices”, In Proceedings of the 2nd Java Virtual Machine Research and Technology Symposium, Aug. 2002.
- [2] STEPHEN N. FREUND and JOHN C. MITCHELL “A Type System for the Java Byte code Language AND VERIFIER” JOURNAL OF AUTOMATED REASONING 30: 271–321, 2003.
- [3] Zi-Gang Lin ; Han-Wen Kuo ; Zi-Jing Guo ; “Stack memory design for a low-cost instruction folding Java processor” Circuits and Systems (ISCAS), 2012 IEEE International Symposium.
- [4] Gibbs,C. ; Coady,Y.”Aspects of Memory Management” System Sciences, 2005. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference.
- [5] Guangyu.Chen ; Kandemir.M,”Improving Java virtual machine reliability for memory-constrained embedded systems” Design Automation Conference, 2005.
- [6] Besson, F. ; Blanc, T. ; Fournet, C. ; Gordon,” From stack inspection to access control: a security analysis for libraries” A.D. Computer Security Foundations Workshop, 2004.
- [7] Drossopoulou, S.: “An abstract model of Java dynamic linking, loading and verification”, in R. Harper (ed.), Workshop on Types in Compilation, Lecture Notes in Comput. Sci. 2071, 2001,pp. 53–84.
- [8] Hevarbert Schildt “the complete Reference”, java 2 Fifth edition.