Projeto Logico JVM

Estruturas e observações

Lukas Ferreira Machado - 12/0127377
Raphael Luís Souza de Queiroz - 13/0154989

Outubro 11, 2017

Outline

Outline

Estrutura & Análise

subdivisão de um JVM real

As especificações de uma JVM são:

Características

- Stack-based virtual machine ;
- Referência simbólica ;
- Garbage collector ;
- Tipo primitivos bem definidos ;
- Network byte order em Big Endian .

Estrutura & Análise

subdivisão de um JVM real

Para a JVM que implementaremos, vamos a subdivídir em:

Estrutura

- Class-loader ;
- Run-time date area ;
- Execution engine .

A JVM garante que o código Java que será executado tenha uma independência de plataforma, possibilitando a portabilidade do código Java. Também é responsável pela execução de pilhas, gerenciamento de memória, threads, entre outros.

Estrutura & Análise

subdivisão de um JVM real

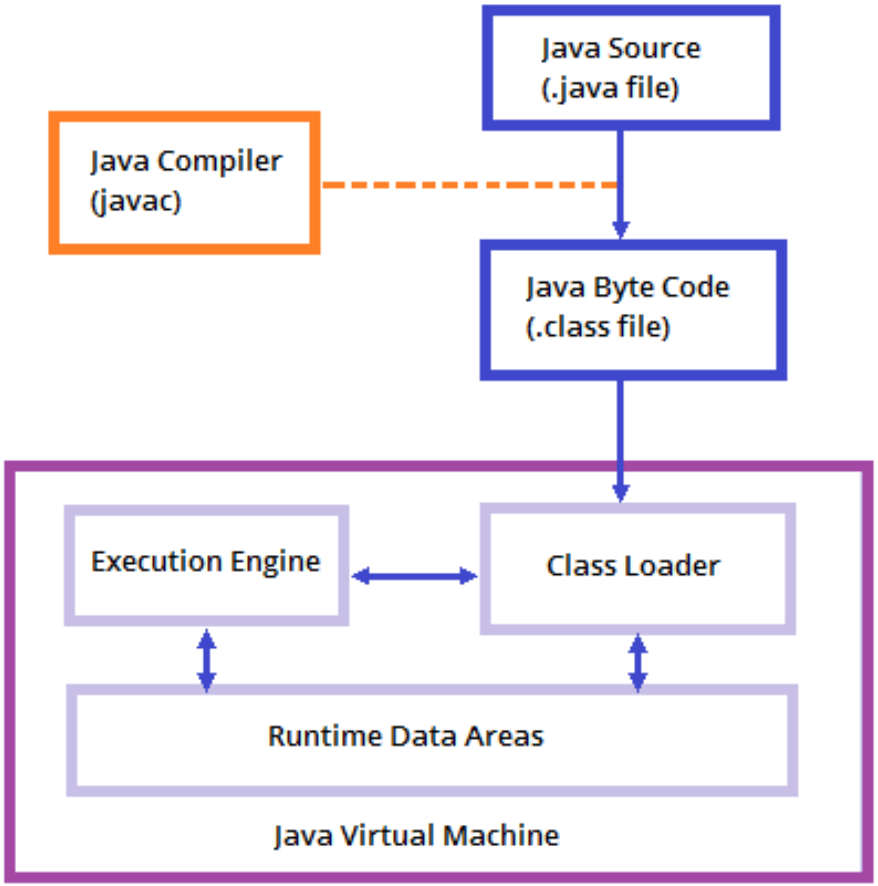

Figure: Processo de execução de um código java

Outline

Estrutura & Análise

Características do Class-loader

Carrega e linka o java bytecode compilado, java .class, em tempo de execução para a áreas de memória.

Estágios do Class-loader

- Loading : Class obtida de um arquivo e carregada na memória da JVM;
- Verifying : Checa se a classe obtida é compatível com as definições em Java;
- Preparing : Prepeara a região de memória para receber as interfaces, campos, métodos, atributos definidos no class;
- Resolving : Mudança de referência simbólica do constant pool para referências diretas;
- Initializing : Inicialização da class com seus valores e execução.

Estrutura & Análise

Características do Class-loader

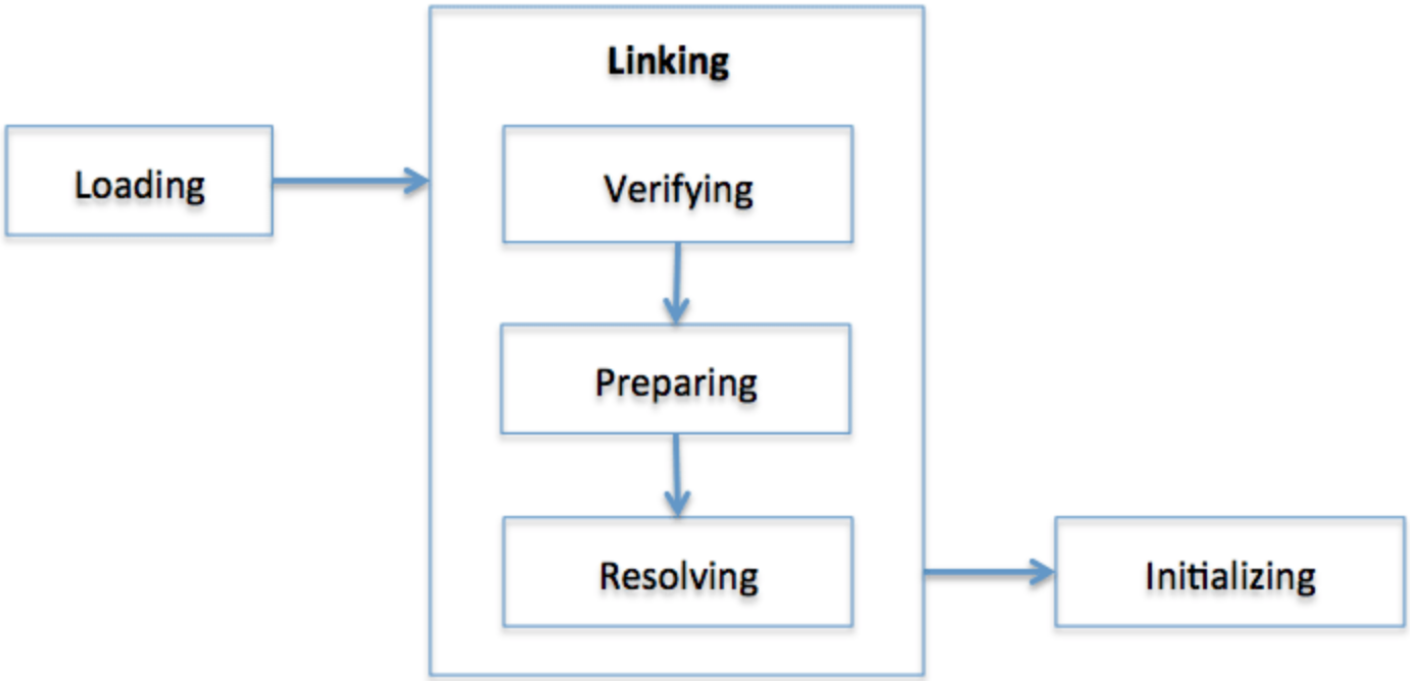

Figure: Estrutura do Class-loader

Estrutura & Análise

Características do Class-loader

Sobre a Estrutura

A estrutura do Class-loader será na forma de uma lista encadeada simples, onde o class será carregado em memória a partir dos dados adquiridos por meio do leitor de .class java.

Outline

Estrutura & Análise

Características da Runtime Data Areas

Análise

A JVM possui uma área de memória, porém, essa área é dividida para armazenamento de dados temporários relacionados a execução de um código Java. A Runtime Data Area pode ser divida em 6 áreas: Heap , Method Area , Runtime Constant Pool , PC Register , JVM Stack e Native Method Stack .

Vale a pena ressaltar

Sendo as 3 primeiras globais à JVM, compiladas por todas as thread, e as 3 últimas são subáreas criadas para cada thread.

Estrutura & Análise

Características da Runtime Data Areas

Características

1. PC register : Program counter para a instrução atual;
2. JVM stack : Pilha de execução que possue os frames de execução(Stack Frames);
3. Native method stack : Para código nativo sem ser Java;
4. Method area : Possue informações sobre a constant pool, campos, métodos para cada classe e interface;
5. Runtime constant pool : Área referente a constant pool do .class;
6. Heap : Armazena instancias de objetos valendo-se de realocação dinâmica.

Estrutura & Análise

Características da Runtime Data Areas

Características

1. PC register : Program counter para a instrução atual;
2. JVM stack : Pilha de execução que possue os frames de execução(Stack Frames);
3. Native method stack : Para código nativo sem ser Java;
4. Method area : Possue informações sobre a constant pool, campos, métodos para cada classe e interface;
5. Runtime constant pool : Área referente a constant pool do .class;
6. Heap : Armazena instancias de objetos valendo-se de realocação dinâmica.

Estrutura & Análise

Características da Runtime Data Areas

Características

1. PC register : Program counter para a instrução atual;
2. JVM stack : Pilha de execução que possue os frames de execução(Stack Frames);
3. Native method stack : Para código nativo sem ser Java;
4. Method area : Possue informações sobre a constant pool, campos, métodos para cada classe e interface;
5. Runtime constant pool : Área referente a constant pool do .class;
6. Heap : Armazena instancias de objetos valendo-se de realocação dinâmica.

Estrutura & Análise

Características da Runtime Data Areas

Características

1. PC register : Program counter para a instrução atual;
2. JVM stack : Pilha de execução que possue os frames de execução(Stack Frames);
3. Native method stack : Para código nativo sem ser Java;
4. Method area : Possue informações sobre a constant pool, campos, métodos para cada classe e interface;
5. Runtime constant pool : Área referente a constant pool do .class;
6. Heap : Armazena instancias de objetos valendo-se de realocação dinâmica.

Estrutura & Análise

Características da Runtime Data Areas

Características

1. PC register : Program counter para a instrução atual;
2. JVM stack : Pilha de execução que possue os frames de execução(Stack Frames);
3. Native method stack : Para código nativo sem ser Java;
4. Method area : Possue informações sobre a constant pool, campos, métodos para cada classe e interface;
5. Runtime constant pool : Área referente a constant pool do .class;
6. Heap : Armazena instancias de objetos valendo-se de realocação dinâmica.

Estrutura & Análise

Características da Runtime Data Areas

Características

1. PC register : Program counter para a instrução atual;
2. JVM stack : Pilha de execução que possue os frames de execução(Stack Frames);
3. Native method stack : Para código nativo sem ser Java;
4. Method area : Possue informações sobre a constant pool, campos, métodos para cada classe e interface;
5. Runtime constant pool : Área referente a constant pool do .class;
6. Heap : Armazena instancias de objetos valendo-se de realocação dinâmica.

Estrutura & Análise

Características da Runtime Data Areas

Características

1. PC register : Program counter para a instrução atual;
2. JVM stack : Pilha de execução que possue os frames de execução(Stack Frames);
3. Native method stack : Para código nativo sem ser Java;
4. Method area : Possue informações sobre a constant pool, campos, métodos para cada classe e interface;
5. Runtime constant pool : Área referente a constant pool do .class;
6. Heap : Armazena instancias de objetos valendo-se de realocação dinâmica.

Estrutura & Análise

Características da Runtime Data Areas

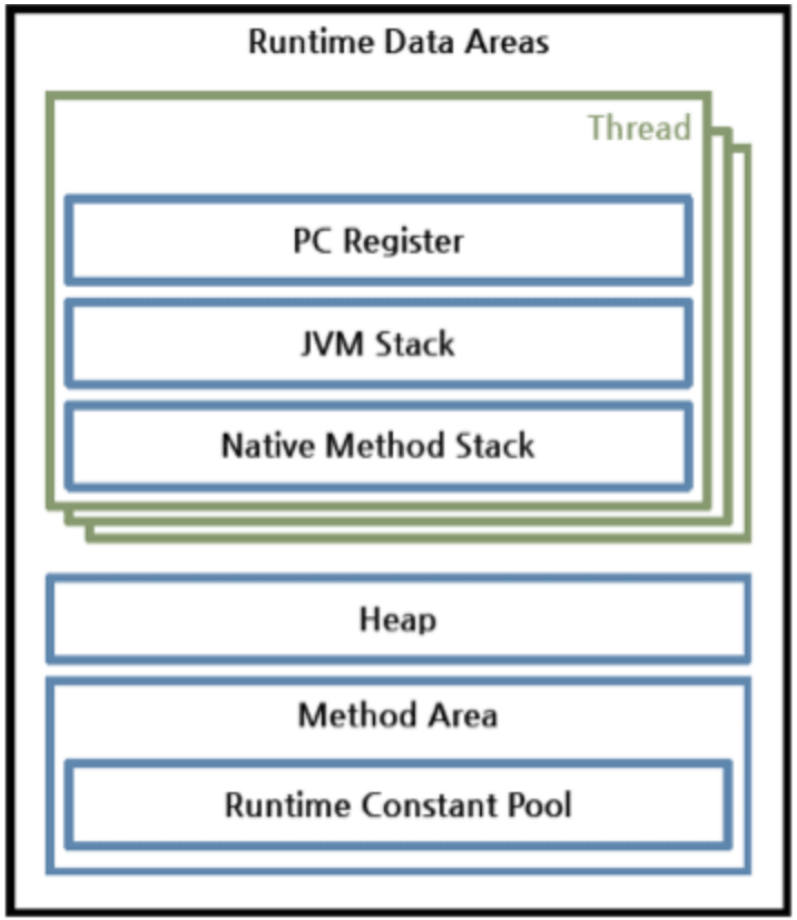

Figure: Estrutura da Runtime Data Areas

Estrutura & Análise

Características da Runtime Data Areas

Análise da Stack Frame

Cada Stack Frame possue referência ao array de variáveis locais, pilha de operandos e ao constant pool. O tamanho da pilha de operandos e do constant poolé determinado em compilação, logo o tamanho do frame é fixo de acordo com o método. Neste trabalho, utilizando um gerenciador de memória, todas as variáveis alocadas em memória são desalocadas pelo mesmo que possue referência ao elemento atual em memória onde first in, last out na desalocação.

Estrutura & Análise

Características da Runtime Data Areas

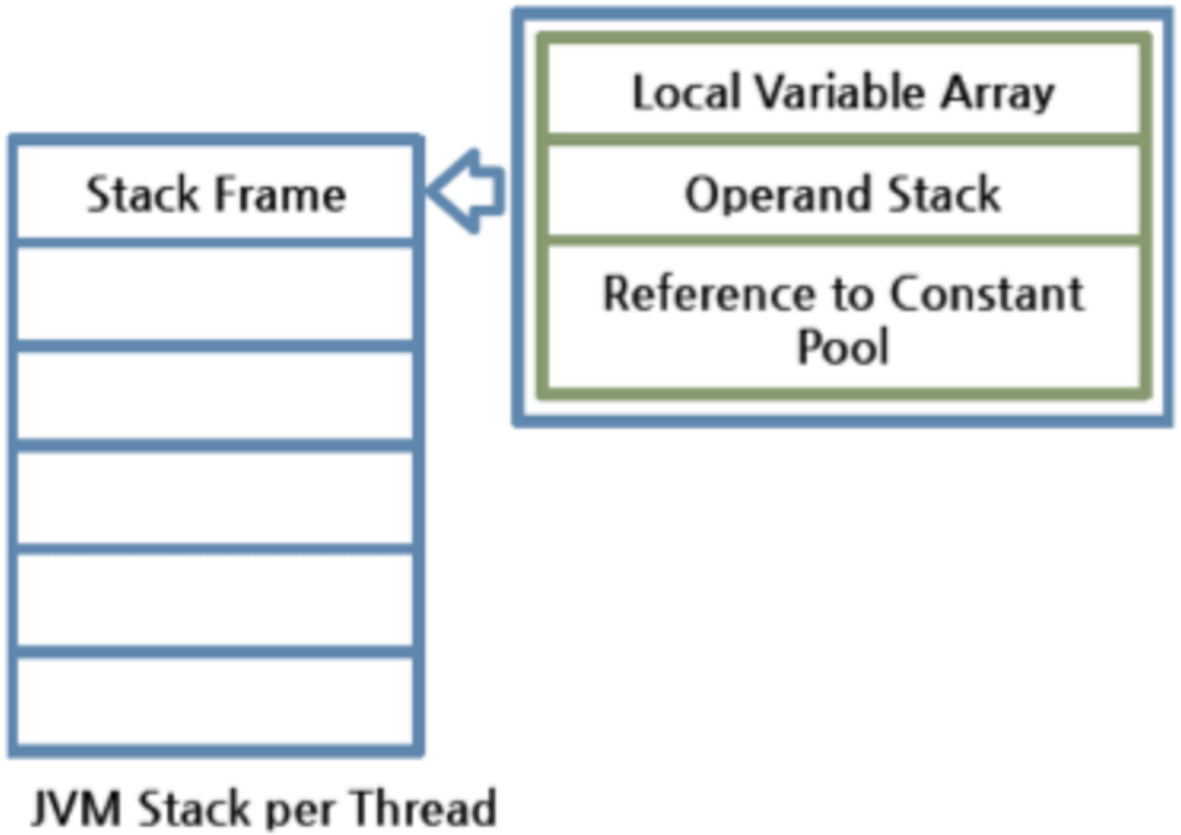

Figure: Estrutura de um Stack Frame

Estrutura & Análise

Características da Runtime Data Areas

Análise da Área de Métodos

Área de Métodos ,como sabemos, possue informações sobre a constant pool, campos, métodos para cada classe e interface. Procurando no array de metódso do class, dando matching pelo nome do método, podemos achar o método a ser executado e, com isso, pegar as informações relavantes do códigos e executá-lo utilizandos-e da pilha de frames.

Estrutura & Análise

Características da Runtime Data Areas

Análise do Heap

Heap , como sabemos ,armazena instâncias de objetos valendo-se de realocação dinâmica. O mesmo faz referência as informações da classe do objeto, os valores de suas variáveis e sua constant pool.

Vale a pena ressaltar

Todas as estruturas até aki citadas serão implementadas como listas encadeadas simples para melhor análise e manipulação, onde sua gerência de memória ficará a cargo de um memory manager.

Outline

Estrutura & Análise

Características da Execution Engine

Estágios do Class-loader

Para que seja executado o código Java na máquina, a JVM utiliza do subsistema Execution Engine que pega o bytecode gerado na Runtime Data Area e da Class Loader. Cada comando do bytecode consiste em 1 byte de opCode mais Operand e executa o comando um a um, semelhante a uma CPU (virtual machine).

Possue duas maneiras de leitura:

- Interpreter : lê, interpreta e executa o bytecode intrução por instrução;
- JIT (Just-In-Time): compila o bytecode inteiro, converte para código nativo e executa o código nativo gerado.

Estrutura & Análise

Características da Execution Engine

Sobre a Estrutura

A estrutura da Execution Engine será subdivida, isso na etapa e execução dos métodos, será decodificadas em cada instrução por meio de matching em uma look up table de acordo com o tipo de instrução. Com isso, a pilha de operandos será utilizada para gerenciar a execução divida em fetch , interpretação ,execute e próxima instrução . Metódos nativos não sendo java não serão feitos neste trabalho .

Bibliografia

Referências

LaTeX Beamer

http://latex-beamer.sourceforge.net/

JVM Internals

https://www.cubrid.org/blog/

Obrigado!

Lukas Ferreira Machado - 12/0127377
Raphael Luís Souza de Queiroz - 13/0154989