Leitor de cabeçalho de arquivos .class em Python.

Giovanna da Costa Gomes Pacheco[1]

[1]Faculdade de Computação – Universidade Federal do Mato Grosso do Sul (UFMS)

giovanna.pacheco@ufms.br

Abstract. *.class Files are necessary for the execution of Java programs. This type of file is the result of the Java compiler, and they carry valuable information about the defined class as headers. This work aims the creation of a Python script to decode this header. As a result, we compared the results using the javap command with the results of the class_header_reader. It was observed that the script correctly decoded the values.*

Resumo. *Arquivos .class são arquivos necessários para execuções de programas na linguagem Java. Esse arquivo resultado do compilador Java possui uma cabeçalho com informações sobre a classe definida. O objetivo desse trabalho e criar um script em python para decodificação desse cabeçalho. Como resultado tem-se a comparação entre o valores encontrados pelo comando javap e os valores encontrados pelo script class_header_reader. Foi observada a correta decodificação dos valores.*

1. Introdução

Arquivos .class são arquivos que contém *bytecodes*, com instruções que podem ser executadas a partir de uma Máquina Virtual Java (JVM). Um arquivo .class Java é produzido por um compilador Java com base em um arquivo-fonte (.java) e possui um cabeçalho contendo informações sobre a estrutura da classe criada no arquivo-fonte. As seções desse cabeçalho são [Antonioli and Pilz 1998]:

- Magic Number (do português, Número Mágico): São os primeiros quatro bytes, que identificam um arquivo .class. Necessariamente este valor deve ser **0xCAFEBABE**.
- Version of Class File Format (do português, Versão do Formato do Arquivo da Classe): Os próximos quatro bytes trazem a informação da menor e maior versão da JVM que a classe suporta.
- Constant Pool (do português, Piscina de Constantes): É uma lista de elementos das quais cada elemento ocupa uma posição da lista e é identificado por um byte para identificação.
- Access Flags (do português, Bandeiras de Acesso): São os próximos dois bytes após a *Constant Pool*, seu valor é como uma lista de permissões de acesso e propriedades da classe ou interface.
- This Class (do português, Essa Classe): São dois bytes que especificam o índice na tabela *Constant Pool*. O valor presenta na tabela é o nome da classe ou da interface do arquivo.
- Super Class (do português, Super Classe): Analogamente ao *This Class*, traz o nome da super classe da qual a classe herda.

- Interfaces: Informa quantas interfaces a classe (ou interface) definida no arquivo implementado. Em seguida, há uma matriz que contém um índice na tabela *Constant Pool* para cada interface implementada pela classe.
- Fields (do português, Campos): Representa uma tabela com todos os campos, tanto variáveis de classe quanto variáveis de instância, declarados pela classe ou interface.
- Methods (do português, Métodos): Começa com um valor de dois bytes que indica a quantidade de métodos na classe ou interface. Cada valor na tabela "methods"descreve um método na classe ou interface.
- Attributes (do português, Atributos): Inicia também com um contador de dois bytes que indica o número de atributos. Fornece informações adicionais sobre a classe ou interface, como seu nome, versão, informações de depuração e outras informações relevantes.

Com a finalidade de apresentar essas informações de forma amigável aos usuários, esse trabalho propõe um leitor de cabeçalho para arquivos .class na linguagem Python utilizando como apoio a biblioteca jawa. A biblioteca nos ajuda a decodificar a informação em *bytecode* transformando as informações em objetos manipuláveis.

O código é apresentado e explicado na seção Metodologia 2. E na seção de resultados 3 são mostrados utilizando o script desenvolvido nesse trabalho.

2. Metodologia

Para cumprir a tarefa proposta nesse presente trabalho, foram pesquisadas bibliotecas de apoio para decodificar arquivos .class. Pela simplicidade de uso foi escolhida a biblioteca jawa em python.

Para utilizar o script class_header_reader, se faz necessário seguir os seguintes passos:

1. Certificar-se que em sua máquina há uma versão python-3.x instalada.
2. Baixar o código do repositório existente nesse link: class_header_reader.
3. Adicionar na mesma pasta do repositório os arquivos .class a serem analisados.
4. Abrir o terminal no caminho em que o repositório foi baixado.
5. Executar o comando: setup.bat. Esperar a finalização da execução.
6. E então executar o script com o comando: python main.py ¡Nome do arquivo .class a ser analisado¿

O arquivo *main.py* (Figura 1) é responsável por verificar a existência do arquivo passado como parâmetro do script. Caso o arquivo exista é instanciado um objeto do tipo *HeaderReader*.

A classe *HeaderReader* possui um construtor (Figura 1) responsável por executar todos os métodos necessários para abrir e decodificar o arquivo .class. Primeiro, o arquivo é aberto (Figura 2) em modo de leitura em binário. Assim que o arquivo é aberto passamos-o para a classe *ClassFile* da biblioteca jawa.

Essa classe é encarregada de instanciar todos os atributos que compõem o cabeçalho do arquivo. Assim que instanciado a classe verifica se o **Magic Number** condiz com arquivos .class, caso contrário é levantado um erro impedindo o resto da execução do script.

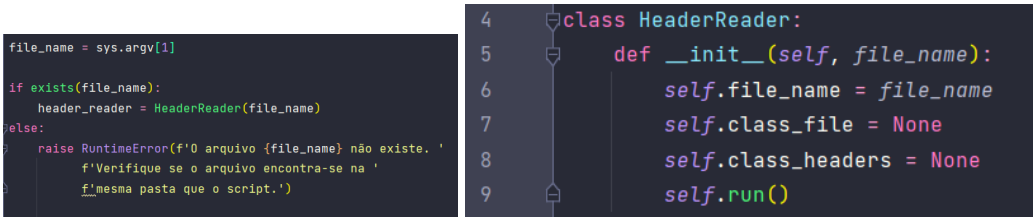

Figura 1. Á esquerda arquivo main.py e à direita construtor da classe *HeaderReader*

Se instanciada corretamente, ou seja o *Magic Number* for **0xCAFEBABE**, todos os atributos, se existirem, já estarão disponíveis para serem acessados. Com isso então no método **run** chamamos a função encarregada por criar o dicionário das informações encontradas pela classe *ClassFile*. Na criação do dicionários acessamos os atributos da classe *ClassFile* a fim de finalmente imprimir as informações decodificas ao usuário (Figura 2).

Figura 2. Á esquerda método que abre o arquivo e à direita método que acessa e salva as informações

Por fim o dicionário criado é impresso ao usuário. Na seção de Resultados 3 são mostrados testes utilizando o sript descrito acima.

3. Resultados

Com o fim de avaliar a correta sintaxe e lógica do script foram executados dois testes e comparados com o resultado da execução do comando *javap -verbose (Arquivo .class)*. Os testes e comparações são mostrados a seguir.

3.1. Teste 1

O primeiro teste trata-se de uma classe de exemplo que possui atributos e métodos. Para esse caso o retorno do comando *javap* foi o mostrado na figura 3 à esquerda. Na figura é possível identificar pelas setas vermelhas que foram encontrados três campos (do inglês, *fields*), três atributos e zero interfaces. Assim como a menor e maior versão como sendo respectivamente 0 e 60, foram encontradas 24 constantes, e nome da classe em questão é ”Teste”e sua super classe ”java/lang/Object”. O resultado da execução do script demonstra ter encontrado mesmos valores (Figura 3).

3.2. Teste 2

Para o segundo teste temos uma classe mais simples que possui apena o seu construtor. Na figura 4, à esquerda temos o resultado a partir do comando *javap* para o arquivo Teste2.class. Nele foram identificados como cabeçalho zero interfaces, zero campos (do inglês, *fields*), um método, um atributo e 12 constantes. Os valores de maior e menor

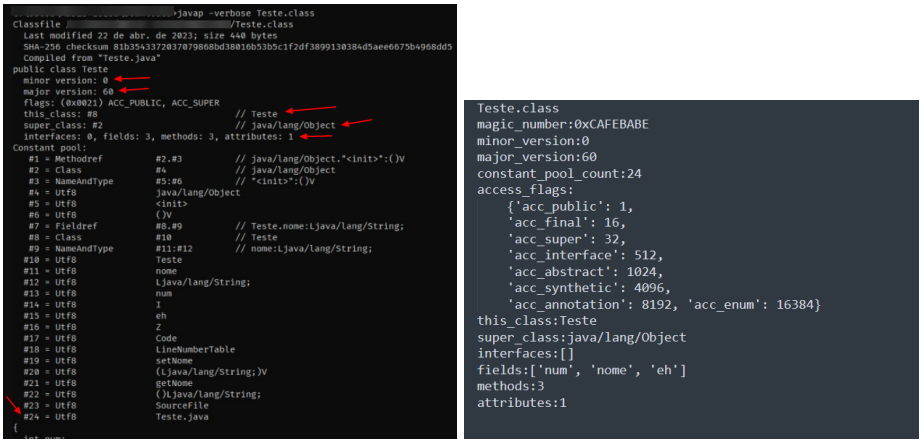

Figura 3. Resultado do arquivo Teste.class

versão são semelhantes ao Teste 1 3.1, sendo respectivamente 60 e 0. Para os valores do script encontramos os mesmos resultados (Figura 4, à direita)

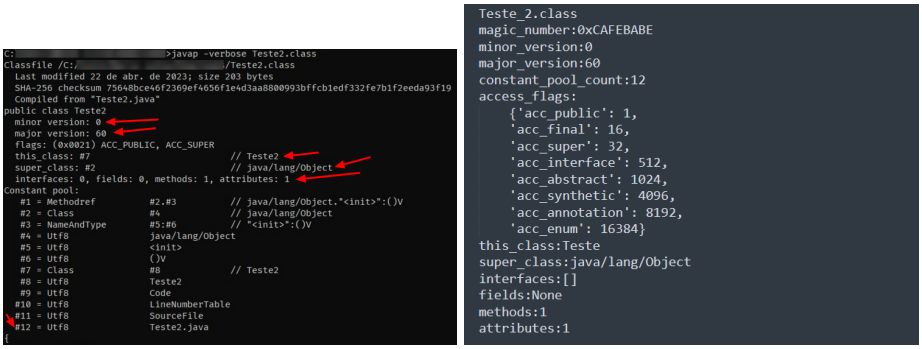

Figura 4. Resultado do arquivo Teste2.class

4. Conclusão

Ler e decodificar arquivos .class possui uma alta complexidade, visto que é necessário lidar com informações de baixo nível e traduzi-las para uma linguagem mais amigável ao usuário. Entretanto, as informações são de extrema importância para que o programa possa ser executado corretamente pela JVM. O entendimento da estruturação do arquivo também é uma tarefa complexa mas que pode ser entendida e colocada em prática a partir deste trabalho.

Referências

[Antonioli and Pilz 1998] Antonioli, D. N. and Pilz, M. (1998). *Analysis of the Java class file format*. Citeseer.