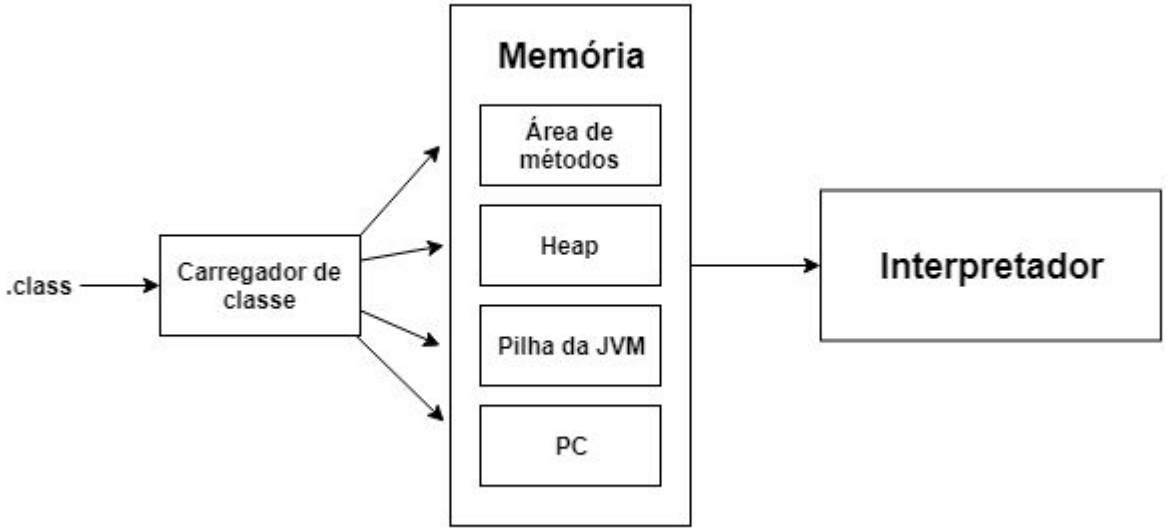
Projeto Lógico JVM

Alec Ryo Emura - 15/0115326
João Victor Poletti - 15/0132425
Rafael Chianca - 15/0145608
Rodrigo Guimarães - 14/0170740
Thiago Araújo da Silva - 15/0149832

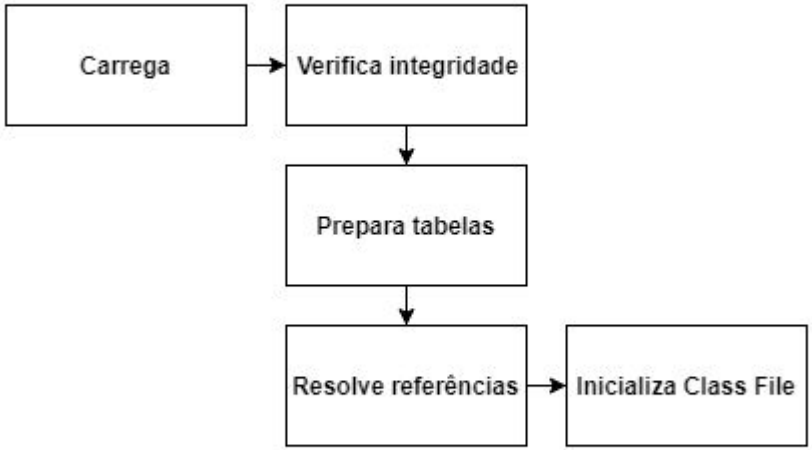
Arquitetura da JVM



1. Class Loader

- Obtém os *bytecodes* de um *.class;*
- É responsável por inicializar dinamicamente as classes e interfaces:
  - Verificar a corretude do arquivo *.class* e se a classe/interface já foi carregada;
  - Preparar e inicializar as estruturas de dados (atributos, métodos e interfaces);
  - Resolve referências simbólicas (tabela de Constant Pool) para referências diretas;
  - Aloca os blocos estáticos de código e os atributos estáticos com seus valores pré-definidos.
- É responsável por carregar as classes (*.class*) e interfaces em tempo de execução da JVM.
  - Instancia um vetor representando a classe/interface como uma estrutura *ClassFile*;

Esquemático do Class Loader



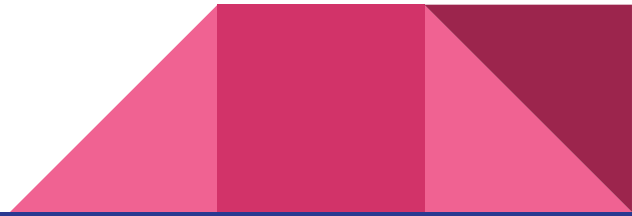
2. Memória

- Armazenamento e operações da memória da JVM
- Trabalharemos em:

1. Área de métodos
2. Pilha JVM
3. Registrador PC

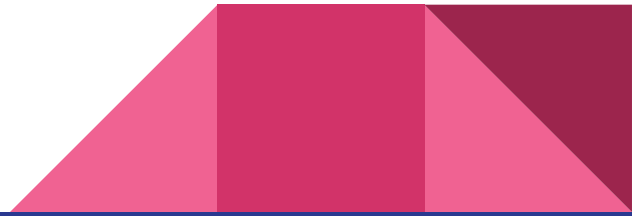
2.1 Área de métodos

- Armazena todas as classes carregadas em um vetor de interfaces genéricas.
- Cada elemento armazenará os atributos, campos, código do método, códigos do construtor, entre outros, de uma determinada classe.



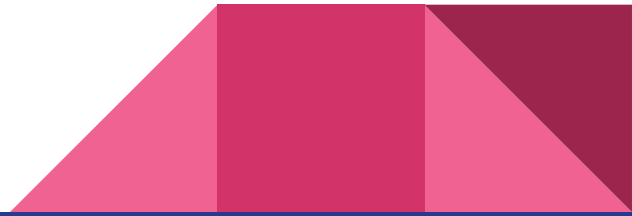
2.2 Pilha JVM

- Utiliza-se a estrutura de pilha;
- Trabalha-se com frames de tamanho fixo:
  - Relacionado com um método;
  - Contém array de variáveis locais, pilha de operandos, referência para tabela de símbolos.
- Quando um método é chamado um frame é criado;
- Quando o processo de um método for finalizado o frame é retirado da pilha;



2.3 Registrador PC

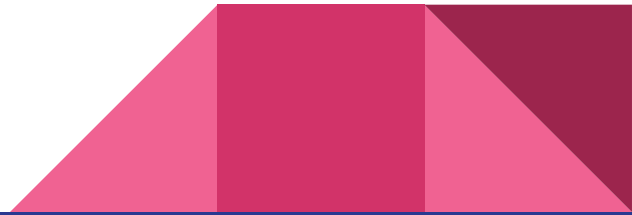
- Armazena o endereço da próxima instrução de execução;
  - Índice válido para o ponteiro de bytes na classe CodeAttribute.
- Inicializado colocando-o na primeira instrução do programa a ser executado pela JVM.



2.3 Registrador PC

Como calcular o PC:

- Se a instrução está no mesmo método e é imediatamente seguinte;
  - Baseada no número de operandos da instrução atual;
- Se a instrução está no mesmo frame e não é imediatamente seguinte;
  - Endereço atual + desvio relativo;
- Se a instrução atual invoca um novo método;
  - Endereço da primeira instrução desse método;
- Se a instrução é a última de um método e deve-se retornar para outro.
  - Empilha o PC e depois chama o método.



3. Interpretador

- Será implementado com a lógica de um interpretador;
  - FETCH - Lê a instrução;
  - DECODE - Interpreta a instrução;
  - EXECUTE - Executa a instrução;
  - PC + 1 - Vai para a próxima instrução.
- Fluxo de execução;
  - Carrega o arquivo .class;
  - Cria frames correspondentes ao método da classe carregada que será usado e o adiciona na pilha da JVM;
  - Executa o método <init> seguido do método main da classe (acessa o método na área de métodos, e inicializa o pc com a primeira instrução).
- Percorre a pilha da JVM executando os métodos e suas Instruções.
- Para resolução de opcodes será utilizado um vetor de ponteiro para funções.