UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS SOBRAL

ENGENHARIA DE COMPUTAÇÃO

AMANDA PAULA FONTENELE - 537536

CLARA RICARDO DE MELO - 509606

GABRIEL CAVALCANTE ALVES MESQUITA - 519389

TRABALHO FINAL DE PARADIGMAS DE LINGUAGEM DE PROGRAMAÇÃO

SOBRAL

2023

1. INTRODUÇÃO

O propósito deste trabalho é falar sobre a linguagem de programação Java e implementar um analisador léxico/sintático para esta. A justificativa deste trabalho está relacionada com a necessidade de compreender e colocar em prática tudo o que foi visto nas aulas da disciplina Paradigmas de Linguagem de Programação em conjunto com conhecimentos prévios e adquiridos ao longo da realização do projeto por meio de diversas fontes.

Para a primeira parte, foi feita uma análise histórica, apresentando algumas curiosidades e inovações da linguagem com base no livro texto da disciplina e nas apresentações feitas em sala de aula. Já na segunda parte, foi implementado um analisador léxico/sintático para a linguagem Java, dado um arquivo .txt, na qual ler informações do teclado, imprime informações no terminal, por exemplo. E, na terceira parte desse trabalho em questão foi feito uma análise mais profunda das características da linguagem e a identificação de elementos que são essenciais na construção da linguagem Java, como nomes, variáveis, endereços. Tal análise está ligada intrinsecamente ao Capítulo 5 e partes do Capítulo 6 do livro base da disciplina.

2. ANÁLISE HISTÓRICA

A linguagem de programação Java foi criada por James Gosling na década de 1990. Inicialmente, foi projetada como uma linguagem para controlar dispositivos eletrônicos embarcados para o consumidor, como sistemas interativos de TV. Confiabilidade e simplicidade era um dos objetivos primários para tal linguagem.

Java é baseado em C++, mas foi projetada para ser menor, mais simples e mais confiável. Uma das principais características e inovações de Java é a sua portabilidade, o que significa que, através do uso da Máquina Virtual Java (JVM), os programas escritos em Java podem ser executados em diferentes sistemas operacionais sem a necessidade de modificar o código-fonte, permitindo os desenvolvedores escrevessem um único código e usassem em várias plataformas.

Outra grande característica inovadora é o mecanismo de gerenciamento automático de memória conhecido como coleta de lixo (garbage collection), isso ajudou em problemas de vazamento de memória, por exemplo. É importante salientar também que embora não tenha sido a primeira linguagem a adotar a programação orientada a objetos, o Java popularizou esse paradigma e tornou amplamente acessível. Essa linguagem também oferece um suporte integrado a threads e sincronização, permitindo que os programas Java executem várias tarefas simultaneamente e compartilhem recursos de forma mais segura, algo essencial para economizar tempo e esforço. Java também se difere de outras linguagens por ter um tipo booleano chamado boolean, usado principalmente para as expressões de controle de suas sentenças de controle (como if e while). Diferentemente de C e C++, expressões aritméticas não podem ser usadas para expressões de controle.

Ademais, sabe-se que Java é uma das linguagens mais populares do mundo. Além de ser uma linguagem para desenvolvimento de software, ela também é usada em outros dispositivos como microcontroladores e consoles de videogame. Além disso, Java é a principal linguagem para o desenvolvimento de aplicativos Android, inclusive o popular jogo Minecraft é escrito em Java. Desse modo, percebe-se que essa linguagem é uma das mais influentes da história e se tornou uma escolha comum para uma ampla gama de aplicações em modo geral, que continua evoluindo e a ser uma força motriz na indústria de software, demonstrando sua duradoura relevância e importância para o mundo da programação.

3. ANALISADOR LÉXICO/SINTÁTICO

Um analisador léxico é uma ferramenta que permite apontar os lexemas e tokens de determinada linguagem. Um analisador sintático verifica se a sequência desses tokens segue o formato da linguagem de programação utilizada.

A implementação do analisador léxico/sintático de java desenvolvido pela equipe pode ser encontrada no seguinte repositório do github:

https://github.com/Amandapaulaf/Trabalho-Paradigmas-de-Programa-o.git

4. DEBULHANDO A LINGUAGEM

- **Formato de nomes**

Essa seção trata do formato de nomes que podem ser utilizados na programação em Java, podendo ser dividida em cinco partes:

1. Quanto a utilização de letras maiúsculas e minúsculas: ao criarmos os nomes de variáveis, podemos usar qualquer letra, seja maiúscula ou minúscula.

2. Evitar nomes de classes ou pacotes: faz-se mister não usar nomes de classes ou pacotes para nomear variáveis, pois esses são nomes de tipos primitivos da linguagem Java, portanto não podem ser utilizados para variáveis. Por exemplo, não devemos dar o nome "int" ou "String" a uma variável.

3. Composição de referências: uma referência pode ser formada por letras e números, desde que não comece com um número (0-9). Portanto, podemos usar números na formação do nome da variável, desde que não estejam no início. Por exemplo, "5int" ou "1num" não são permitidos. Isso ocorre porque o interpretador do Java pode confundir um nome que começa com números com um valor numérico, como um número complexo, em vez de uma variável.

4. Caracteres especiais: a maioria dos caracteres especiais, como ç, /, =, !, @, #, $, %, &, ( ), [ ], ^, ~ e ´, não podem ser utilizados na composição de nomes de variáveis, porém temos uma exceção que é o caractere de sublinhado (_), ele pode ser usado. Portanto, é recomendado evitar caracteres especiais nos nomes de variáveis.

5. Palavras reservadas: o Java possui muitas palavras reservadas que têm um uso específico na linguagem. Logo, essas palavras não podem ser usadas como nomes de variáveis. Um exemplo são as palavras reservadas IF e ELSE, palavras reservadas que foram utilizadas no código proposto pela equipe, por exemplo.

- **Variáveis**

A linguagem de programação Java define os seguintes tipos de variáveis:

Variáveis de instância (campos não estáticos): nesse tipo de variável os objetos armazenam seus estados individuais em "campos não estáticos", ou seja, campos declarados sem a palavra-chave static. Precisa-se ter em mente que os campos não estáticos também são conhecidos como variáveis de instância porque seus valores são exclusivos para cada instância de uma classe, que é um objeto.

Variáveis de classe (campos estáticos): uma variável de classe é qualquer campo declarado com static; isso informa ao compilador que existe exatamente uma cópia dessa variável, independentemente de quantas vezes a classe foi instanciada.

Variáveis locais: aqui nesse tipo de variáveis um método geralmente armazena seu estado temporário em variáveis locais. A sintaxe para declarar uma variável local é semelhante à declaração de um campo (por exemplo, int count = 0;). Não há palavra-chave especial que designe uma variável como local; é importante ressaltar que essa determinação vem inteiramente do local em que a variável é declarada — que está entre as chaves de abertura e fechamento de um método. Desse modo, sabe-se que as variáveis locais são visíveis apenas para os métodos nos quais são declaradas e não são acessíveis pelo resto da classe.

Parâmetros: lembre-se de que a assinatura do main método é public static void main(String[] args). Aqui, a args variável é o parâmetro para este método. O importante a lembrar é que os parâmetros são sempre classificados como "variáveis" e não "campos". Isso também se aplica a outras construções que aceitam parâmetros (como construtores e manipuladores de exceção).

- **Endereço**

Esse assunto trata dos diferentes tipos de dados em Java e como eles são armazenados na memória. Por exemplo, na linguagem Java, existem dois tipos de dados principais: primitivos e referência. Os dados primitivos são aqueles que representam valores reais, como números inteiros (int), números com ponto flutuante (double) e também os caracteres (char). Por outro lado, os dados de referência são usados para armazenar endereços na memória que apontam para objetos.

Além disso, a memória é dividida em duas seções: stack (pilha) e heap (monte). Os dados primitivos são alocados na pilha, enquanto os dados de referência são alocados no monte. As variáveis de tipo primitivo armazenam o valor real desses dados, enquanto as variáveis de referência armazenam endereços na memória que apontam para objetos armazenados no monte.

Em relação ao monte, é a área da memória onde os objetos da sua aplicação são armazenados, esses objetos são alocados no monte quando são referenciados em algum lugar do programa, por isso são chamados de "objetos por referência".

Existem também os objetos chamados de curtos e de curta duração, eles são armazenados na pilha. A pilha funciona como uma pilha real, uma área de memória com tamanho fixo, mas nem sempre contínua. Os dados são empilhados na pilha enquanto são alocados e desempilhados quando não são desnecessários.

Ao criar um objeto do tipo Animal, por exemplo, usando o comando "new Carrol();", o objeto é criado nesse momento, para isso faz-se necessário a escrita do "new" para criar o objeto em Java. No entanto, para manipular as informações desse objeto, precisa de uma variável de referência que armazena o endereço de memória do objeto criado. É por essa razão que escrevemos "Carro carro = new Carrol();", onde "carro" é a variável de referência que guarda o endereço de memória do objeto. É comum chamar essa variável de referência de "objeto".

- **Tipos de dados primitivos**

A linguagem de programação Java é tipada estaticamente, o que significa que todas as variáveis devem ser declaradas antes de serem usadas. O tipo de dados de uma variável nos informa qual o tipo de valor ela pode conter, e como podemos trabalhar com eles. A linguagem Java oferece suporte a oito tipos de dados primitivos . Um tipo primitivo é predefinido pelo idioma e é nomeado por uma palavra-chave reservada.

Byte: o byte tipo de dados é um inteiro de complemento de dois com sinal de 8 bits. Tem um valor mínimo de -128 e um valor máximo de 127 (inclusive). Pode ser utilizado para economia de memória em arrays.

Short: o short tipo de dados é um inteiro de complemento de dois com sinal de 16 bits. Tem um valor mínimo de -32.768 e um valor máximo de 32.767.

Int: por padrão, o int tipo de dados é um inteiro de complemento de dois com sinal de 32 bits, que tem um valor mínimo de -2 31 e um valor máximo de 2 31 -1.

Long: o long tipo de dados é um inteiro de complemento de dois de 64 bits. O longo assinado tem um valor mínimo de -2 63 e um valor máximo de 2 63 -1. Usado para intervalos mais longos.

Float: o float tipo de dados é um ponto flutuante IEEE 754 de 32 bits de precisão simples.

Double: o double tipo de dados é um ponto flutuante IEEE 754 de 64 bits e precisão dupla.

Boolean: o boolean recebe apenas dois valores: true ou false, usado para condições.

Char: O char tipo de dados é um único caractere Unicode de 16 bits.

- **Valores padrão**

Campos declarados mas não inicializados recebem um valor padrão pelo compilador. De modo geral esse padrão assume dois valores: zero ou null. Confiar em tais valores padrão, no entanto, é considerado um desvio das boas normas de programação.

Tipo de dados	Valor padrão (para campos)
byte	0
curto	0
int	0
longo	oL
flutuador	0.0f
dobro	0,0d
caracteres	'\u0000'
String (ou qualquer objeto)	null
boleano	false

É importante salientar que o compilador nunca atribui um valor padrão a uma variável local não inicializada.

- **Escopo**

O escopo em Java refere-se à parte do programa onde uma variável é acessível. É válido ressaltar que existem diferentes tipos de escopo, e cada um determina onde uma variável pode ser usada, ou seja, visível e acessada. Esse escopo vai depender da linguagem, mas neste trabalho estamos nos referindo à linguagem Java. Segue as principais categorias de escopo em Java:

1. Variáveis de membros (escopo de nível de classe): Essas variáveis são declaradas dentro de uma classe, porém fora de função ou método. Dessa forma, elas podem ser acessadas em qualquer lugar dentro da classe, portanto outros métodos da mesma classe podem usar essas variáveis.

2. Variáveis locais (escopo de nível de método): Variáveis locais são declaradas dentro de um método e têm escopo restrito apenas a esse método ou função. Elas não podem ser acessadas fora do método em que foram declaradas, ou seja, visíveis apenas dentro do método onde foram criadas.

3. Variáveis de loop (escopo de bloco): Quando uma variável é declarada dentro de chaves em um método, ela tem um escopo limitado a essas chaves. Essas variáveis são usadas em loops, como o "for" e "while" para controlar o fluxo do loop.

Em geral, podemos acessar uma variável desde que ela tenha sido definida dentro do mesmo conjunto de chaves em que estamos escrevendo o código ou dentro de chaves aninhadas dentro desse conjunto, ou seja, os subprogramas que são visíveis a partir desse escopo local.

- **Constantes**

Quanto às constantes, elas são declaradas usando uma palavra chave chamada"final" e seu valor não pode ser alterado durante a execução do programa. Elas podem armazenar valores fixos, como números ou strings, que não devem ser modificados. As constantes podem ser acessadas em toda a classe onde foram declaradas.

- **Tipo de dados por referência**

Quando vamos escrever programas, utilizamos diferentes tipos de dados para armazenar informações e um deles é o tipo por referência, que nos permite armazenar a localização de objetos na memória(do computador). Dentro desses objetos referenciados podem conter várias variáveis e métodos.

Por exemplo, dada uma classe chamada "Token" que possui métodos e atributos, para criar um objeto dessa classe e poder acessar seus métodos e atributos, precisamos ter uma referência a esse objeto da seguinte forma:

Token acao = new Token();

No exemplo, foi criado o objeto da classe "Token", e a variável "ação" contém uma referência a esse objeto. Dessa forma, podemos usar a variável "ação" para invocar métodos e acessar atributos da classe "Token", deixando um código mais organizado e limpo. É válido ressaltar que as variáveis de referência são inicializadas com o valor "null" (nulo) caso não sejam atribuídas a um objeto específico.

Diferente das variáveis de referência, as variáveis de tipos por valor não fazem referência a objetos, não podendo assim serem usadas para chamar métodos ou acessar atributos. Temos também que as variáveis locais não são inicializadas automaticamente, dessa forma é necessário atribuir um valor a elas antes de usá-las para não gerar erros.

- **Matrizes**

Matriz é uma coleção de variáveis do mesmo tipo e são referenciadas por um nome comum. Os Arrays funcionam de maneira um pouco diferente a depender da linguagem. Segue abaixo algumas informações sobre os arrays em Java:

Temos a alocação dinâmica, onde todos os arrays são alocados dinamicamente, permitindo que seu tamanho possa ser definido durante tempo de execução. Temos também a "Propriedade de Comprimento", onde podemos determinar o tamanho do array usando a palavra "length" do objeto array. Por exemplo, em um array chamado "coisas", ficaria da seguinte forma: coisas.length.

Em relação a variável de uma matriz, ela pode ser declarada especificando seu tipo de dados seguido por colchetes. Exemplo: uma variável de matriz de inteiros chamada "números" declarada como "int[] numeros".

Quanto aos índices de arrays, seus elementos são ordenados e para cada um deles há um índice associado que começa a partir de 0(primeiro elemento está nesse índice), logo o segundo elemento estará no próximo índice que é 1, e assim sucessivamente. Esses arrays podem ser utilizados como campos estáticos, variáveis locais ou parâmetros de métodos.

Todo array tem um tamanho e ele deve ser especificado como um valor inteiro "int" ou um valor curto "short". Não é possível usar um valor longo "long" para definir o

tamanho de um array, pois os arrays em Java só aceitam tipos de referência, não tipos primitivos.

Em relação a esses tipos de dados em arrays, eles podem ter referência a tipos de dados primitivos como char, int, etc, quanto a objetos de classe.

- **Strings**

Strings são sequências de caracteres em Java, são utilizadas para manipulação de textos e representação. Uma string é criada de forma semelhante a um tipo primitivo, basta que o conteúdo esteja entre aspas e ele pode ser atribuído de forma fácil.

String texto = "Qualquer texto entre aspas é uma String";

O java utiliza um mecanismo facilitador chamado pool de strings, ao encontrar strings no código ele retorna uma mesma instância de String, que aponta para uma entrada no pool interno da JVM.

Quando utilizamos o operador new na declaração de uma string forçamos a criação de um novo objeto dessa classe, o que causa a utilização de mais recursos.

5. REFERÊNCIAS

FILHO, Cláudio Rogério. NOMENCLATURA DAS VARIÁVEIS EM JAVA: CURSO DE JAVA. eXcript, 2016. Disponível em: http://excript.com/java/caracteristica-variavel-java.html. Acesso em: 05 jul. 2023.

REIS, Fábio. Como declarar e usar Constantes em Java: Constantes em Java. Boson treinamentos, 2018. Disponível em: http://www.bosontreinamentos.com.br/java/como-declarar-e-usar-constantes-em-java/. Acesso em: 07 jul. 2023.

SEBESTA, Robert W.. Conceitos de Linguagens de Programação. 9. ed. Porto Alegre: Bookman Companhia Editora Ltda, 2011. 795 p. v. 1.

Louden, Kenneth C. Compiladores: Princípios e Práticas. São Paulo: Pearson, 2021.

Linguagens formais e autômatos, recuperado de: https://youtube.com/playlist?list=PLqlIQgAFrQ14oDPZliY1-tyupYs0prBmW

Compiladores, recuperado de: https://youtube.com/playlist?list=PLqlIQgAFrQ14VmHe8VbIVUkBv5Hziv86-