



**INSTITUTO SUPERIOR POLITÉCNICO DE TECNOLOGIAS E CIÊNCIAS
DET – DEPARTAMENTO DE ENGENHARIAS E TECNOLOGIAS
ENGENHARIA INFORMÁTICA
COMPILADORES**

MANUAL DO UTILIZADOR DO MINI-COMPILADOR

Cândido Ucuahamba

**LUANDA
2023**

MANUAL DO UTILIZADOR DO MINI-COMPILADOR

Trabalho de Compiladores apresentado ao
Instituto Superior Politécnico de
Tecnologias e Ciências, como exame final

Docente: Eng. André Filemon

Cândido Ucuahamba

LUANDA

2023

RESUMO

Este manual foi elaborado com o intuito de auxiliar qualquer utilizador a trabalhar com o mini compilador criado por Cândido Ucuahamba.

INTRODUÇÃO

Este mini-compilador foi desenvolvido com a linguagem de programação java, com o intuito de reconhecer código fonte da mesma linguagem acima referida.

PRÉ-REQUISITOS

Para a ter acesso a este mini compilador, é imperativo ter em posse alguns requisitos de forma prévia que passarei a citar abaixo:

- Ter uma IDE instalada em sua máquina de nome Netbeans (versão 17 de preferência);
- Ter o Java Development Kit instalado em sua máquina, que possui o compilador do código fonte da linguagem de programação java;
- Ter conhecimentos básicos de como manejar a IDE Netbeans;
- Ter conhecimentos de programação e da linguagem java em específico.

I. TUTORIAL

Após a IDE Netbeans e o JDK estarem instalados, prime do Windows e pesquise por “Netbeans” e obterá o seguinte resultado:

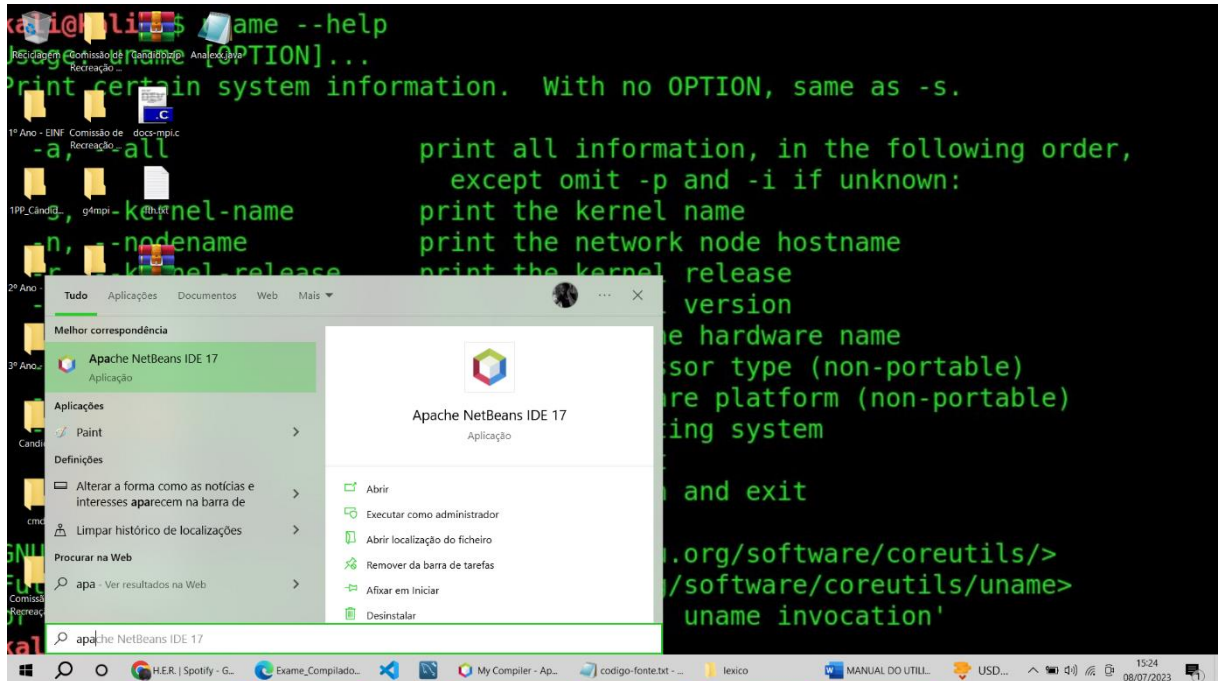


Figura 1. Resultado da pesquisa Netbeans.

Após pressionar a tecla enter, terá o seguinte resultado:

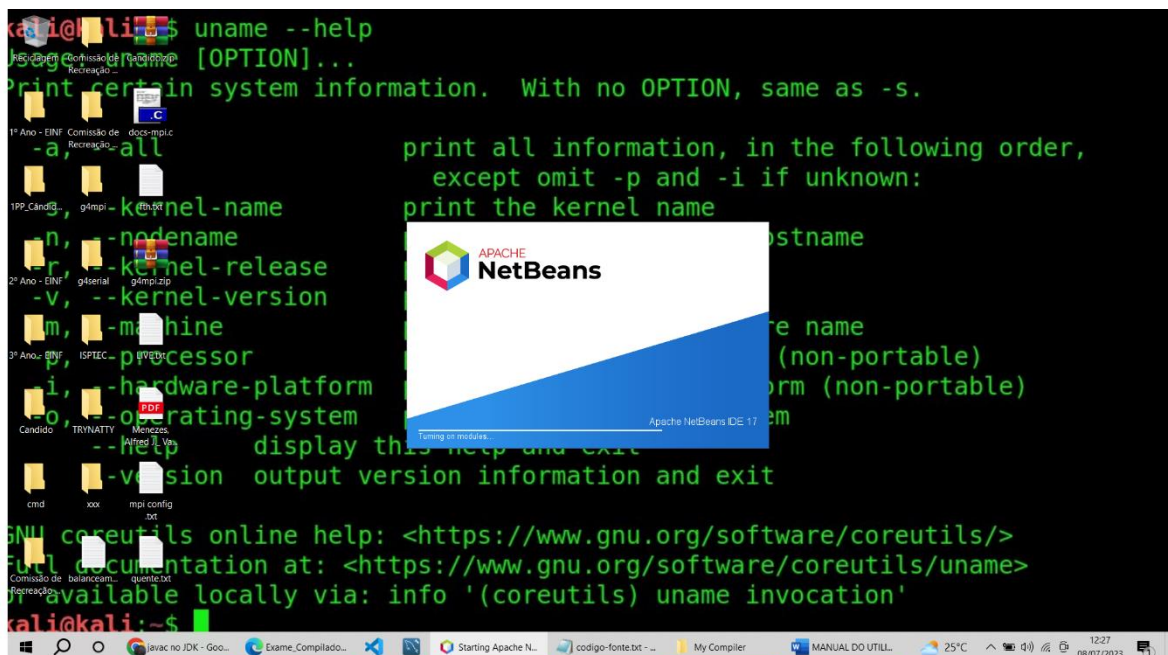


Figura 2. Abertura do Netbeans

Supondo que tenha o projeto em sua posse, após abrir o netbeans deverá abrir o projeto no diretório em que o mesmo se encontrar (pode usar o atalho Ctrl+Shift+O para abrir a janela de abertura de projetos).

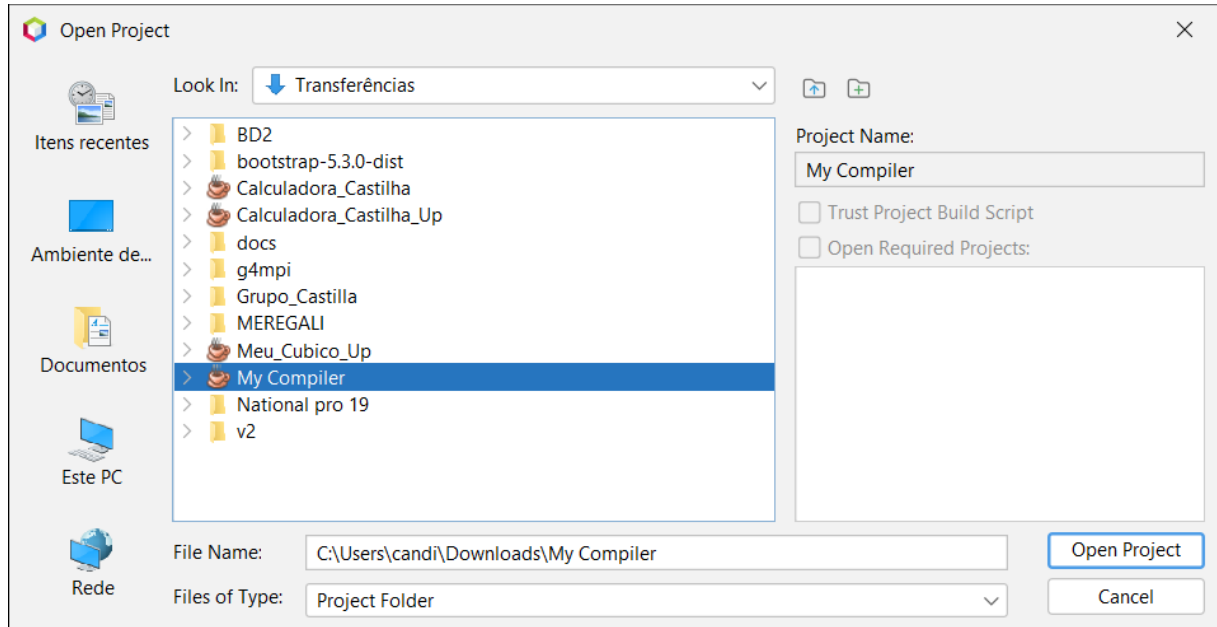


Figura 3. Abertura da janela dos projetos

Após selecionar o projeto, pressiona a tecla enter ou clicar com a seta do cursor no botão “Open Project” e terá o seguinte resultado:

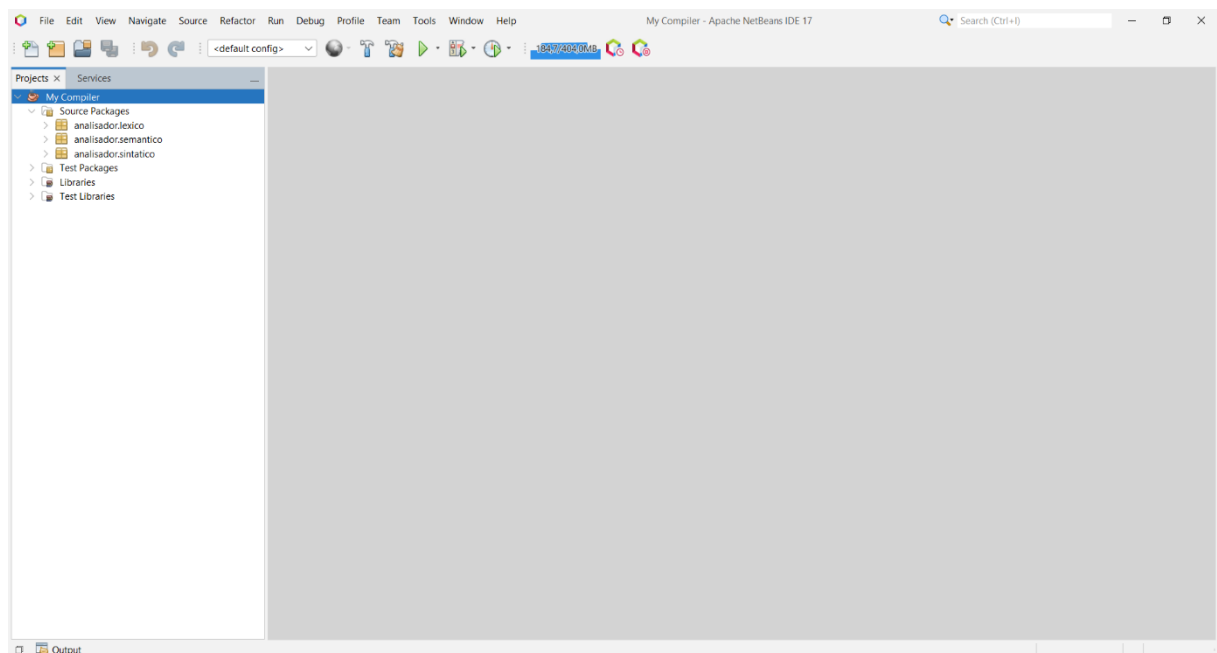


Figura 4. Projeto aberto

II. FUNCIONAMENTO

O mini compilador está composto por três packages, onde temos os analisadores léxico, sintático e semântico com as suas respectivas funcionalidades, todavia uma complementando a outra.

Para utilizador cada um dos analisadores deve ir até o diretório em que se encontra o projeto, e abrir o bloco de notas de nome “código-fonte.txt”, onde estará todo o código fonte que será analisado pelo mini compilador.

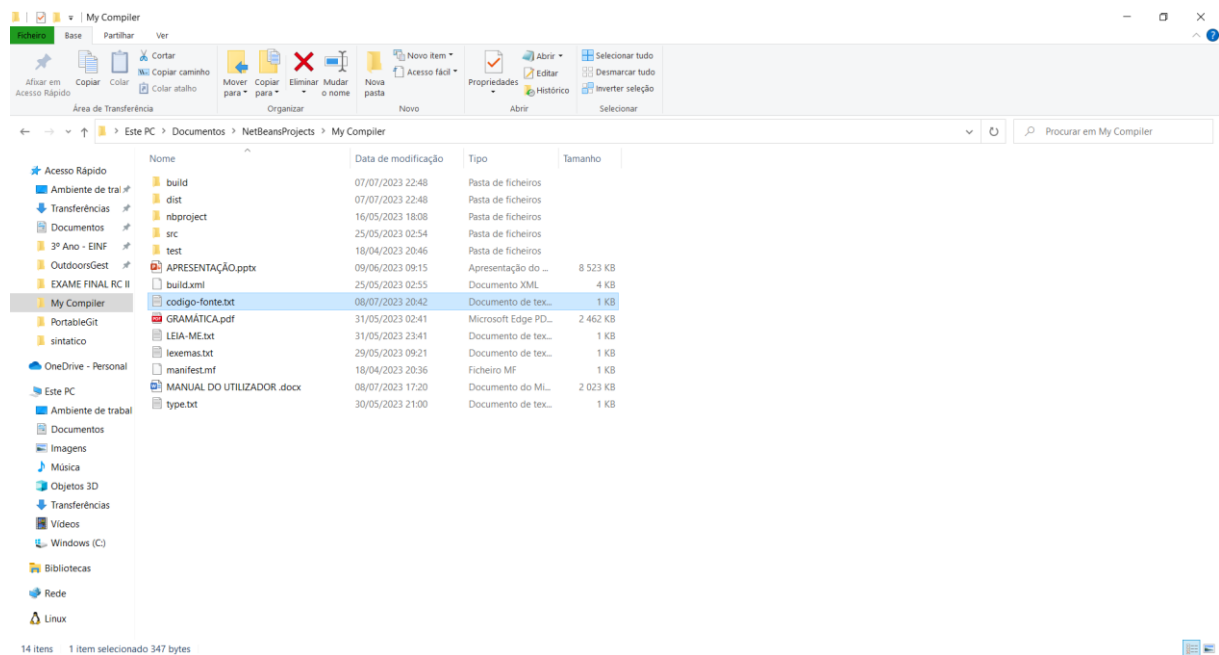


Figura 5. Ficheiro de texto que contém o código fonte

Após abrir o ficheiro, o documento estará ao seu dispor para edição e poderá colocar código fonte da linguagem java.

```
codigo-fonte.txt - Bloco de notas
Ficheiro Editar Formatar Ver Ajuda

package analisador.lexico;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.Reader;
import java.util.ArrayList;
import java.util.Scanner;
import java.util.StringTokenizer;

public class Analex {
    public static void main (String [] args) {

    }
}
```

Figura 6. Código fonte

Para trabalhar com o analisador léxico, com o código fonte ao seu dispor, vá até à package “analisador.lexico” e prime na java main class “ExecutaAnalex.java” utilize o atalho F6 ou marque o cursor no código fonte do “ExecutaAnalex.java” e prime o botão direito do mouse e selecione a opção “Run File” ou utilize o atalho Shift+F6.

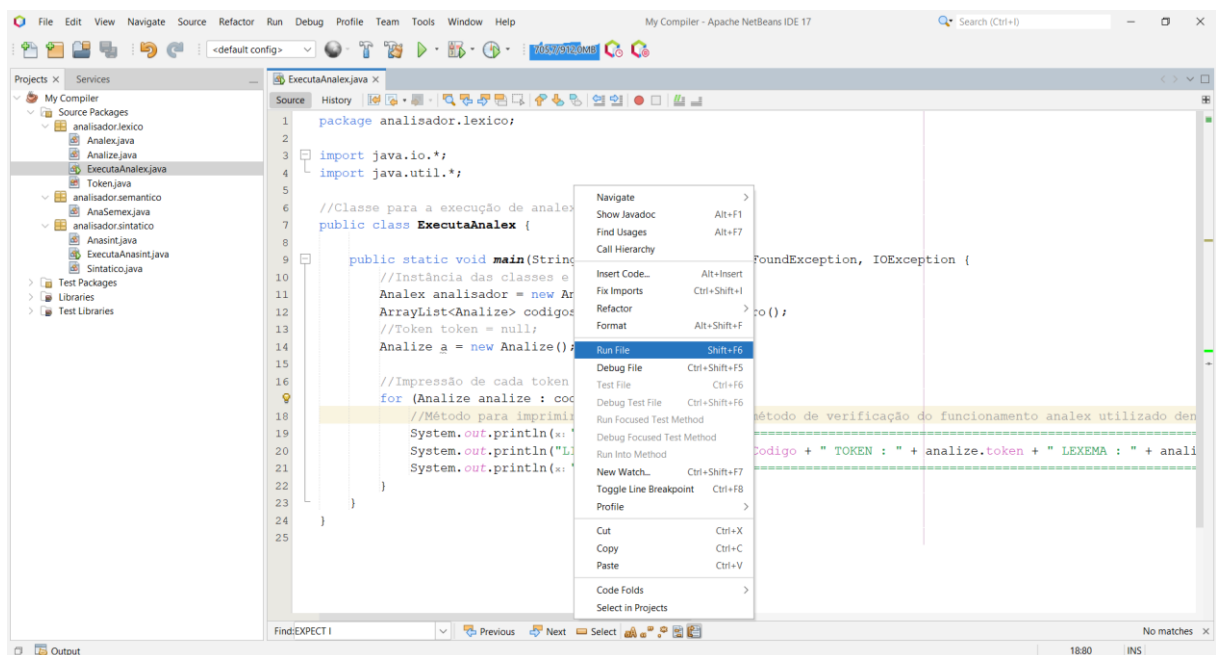
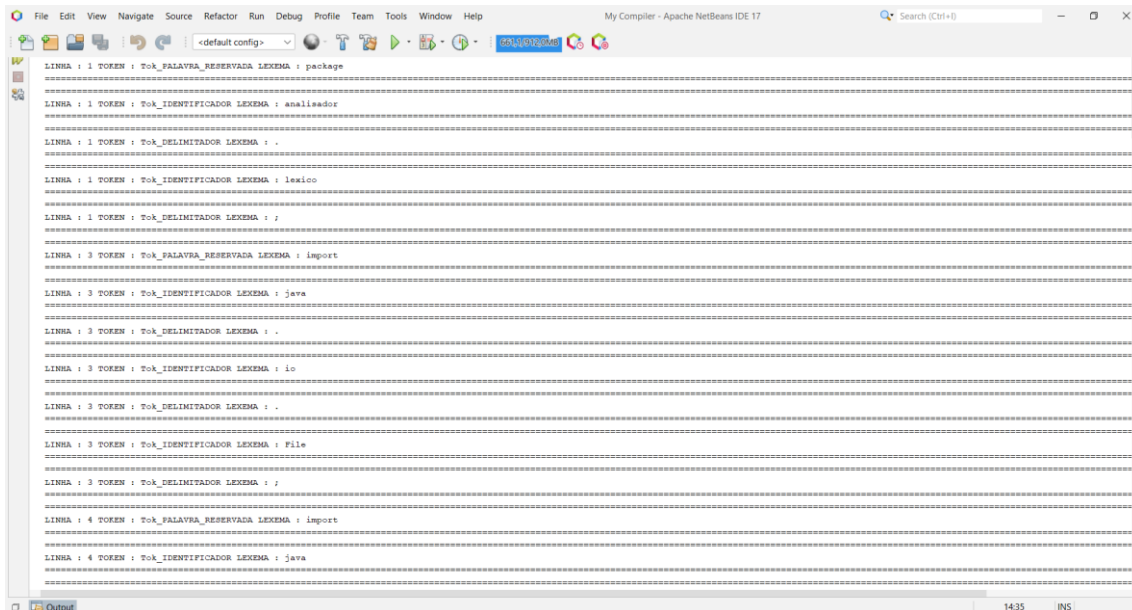


Figura 7. Java main class “ExecutaAnalex”



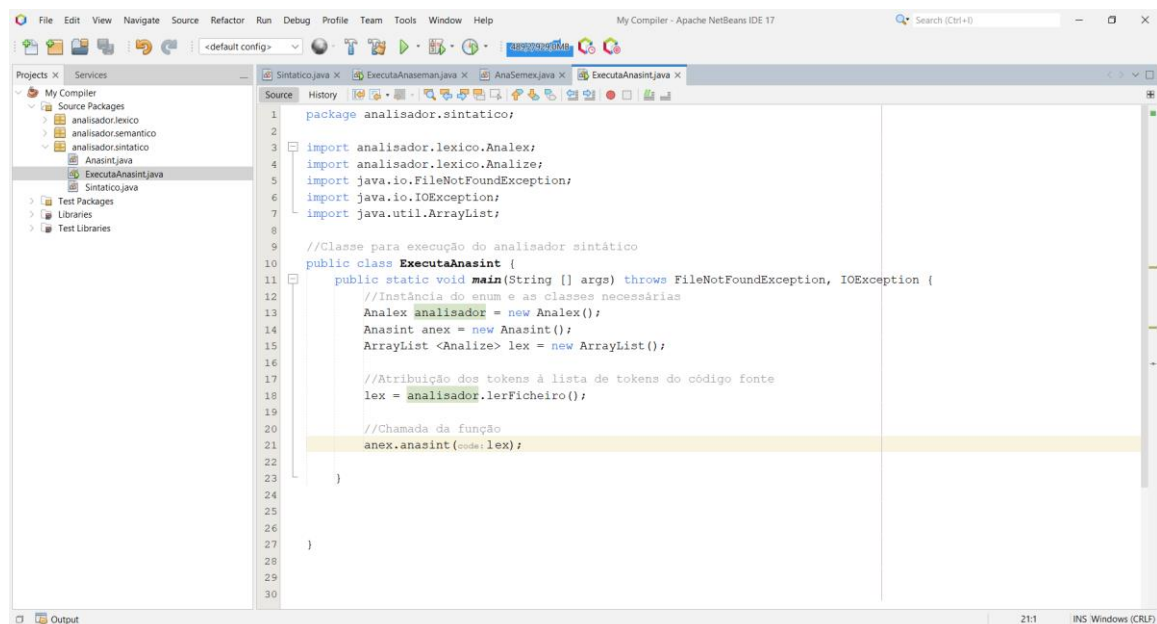
```

LINHA : 1 TOKEN : Tok_PALAVRA_RESERVADA LEXEMA : package
=====
LINHA : 1 TOKEN : Tok_IDENTIFICADOR LEXEMA : analisador
=====
LINHA : 1 TOKEN : Tok_DELIMITADOR LEXEMA : .
=====
LINHA : 1 TOKEN : Tok_IDENTIFICADOR LEXEMA : lexico
=====
LINHA : 1 TOKEN : Tok_DELIMITADOR LEXEMA : ;
=====
LINHA : 3 TOKEN : Tok_PALAVRA_RESERVADA LEXEMA : import
=====
LINHA : 3 TOKEN : Tok_IDENTIFICADOR LEXEMA : java
=====
LINHA : 3 TOKEN : Tok_DELIMITADOR LEXEMA : .
=====
LINHA : 3 TOKEN : Tok_IDENTIFICADOR LEXEMA : io
=====
LINHA : 3 TOKEN : Tok_DELIMITADOR LEXEMA : .
=====
LINHA : 3 TOKEN : Tok_IDENTIFICADOR LEXEMA : File
=====
LINHA : 3 TOKEN : Tok_DELIMITADOR LEXEMA : ;
=====
LINHA : 4 TOKEN : Tok_PALAVRA_RESERVADA LEXEMA : import
=====
LINHA : 4 TOKEN : Tok_IDENTIFICADOR LEXEMA : java
=====

```

Figura 8. Resultado da execução.

Para trabalhar com o analisar sintático, não há diferença nos passos, pois a estrutura da package é a mesma. Portanto irá para o package “analisador.sintatico”, e abrirá o ficheiro “ExecutaAnasint.java” e aplicará o mesmo processo para executar que usou na package anterior.



```

1 package analisador.sintatico;
2
3 import analisador.lexico.Analex;
4 import analisador.lexico.Analize;
5 import java.io.FileNotFoundException;
6 import java.io.IOException;
7 import java.util.ArrayList;
8
9 //Classe para execução do analisador sintático
10 public class ExecutaAnasint {
11     public static void main(String [] args) throws FileNotFoundException, IOException {
12         //Instância do enum e as classes necessárias
13         Analex analisador = new Analex();
14         Anasint anex = new Anasint();
15         ArrayList <Analyze> lex = new ArrayList();
16
17         //Atribuição dos tokens à lista de tokens do código fonte
18         lex = analisador.lerFicheiro();
19
20         //Chamada da função
21         anex.anasint (code:lex);
22     }
23
24 }
25
26
27
28
29
30

```

Figura 9. Java main class “ExecutaAnasint”

Após a abertura do ficheiro executável ,estará em condições de ver o analisador sintático a funcionar. A forma de execução é a mesma que do analisador anterior e as

formas de testes são as mesmas, escrevendo o código fonte que pretende analisar no bloco de notas “código-fonte.txt”.

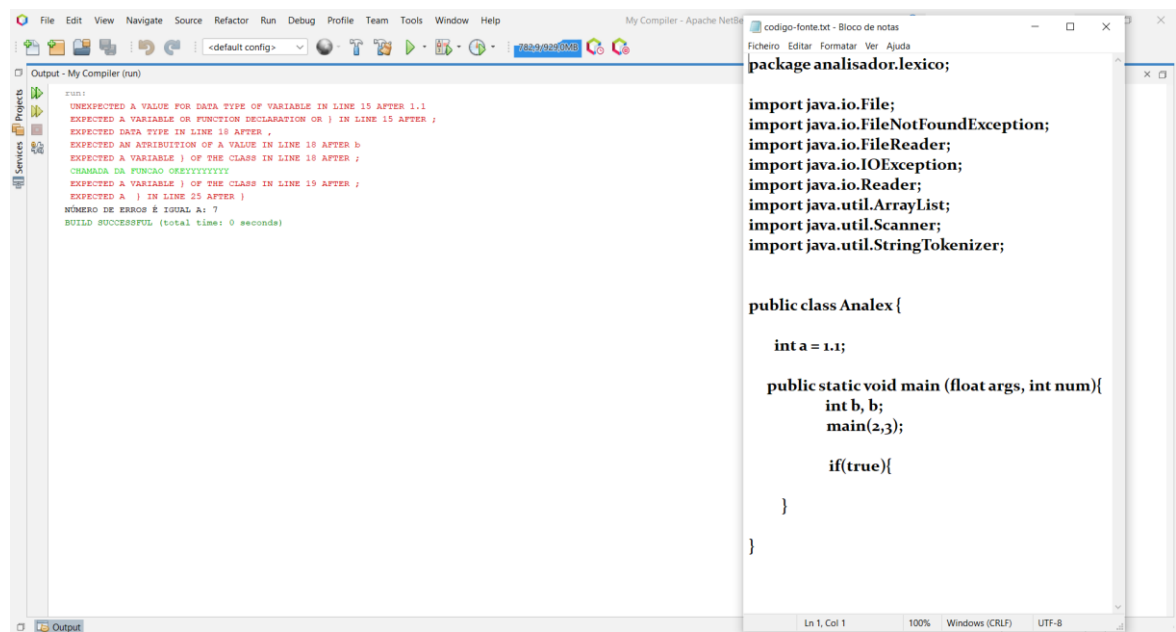


Figura 10. Resultado da execução do analisador sintático

A terceira package de nome “analisador.semantico” temos o ficheiro executável de nome “ExecutaAnaSeman” do analisador semântico e o processo para execução é exatamente o mesmo que nas packages anteriores e teremos como resultado o seguinte:

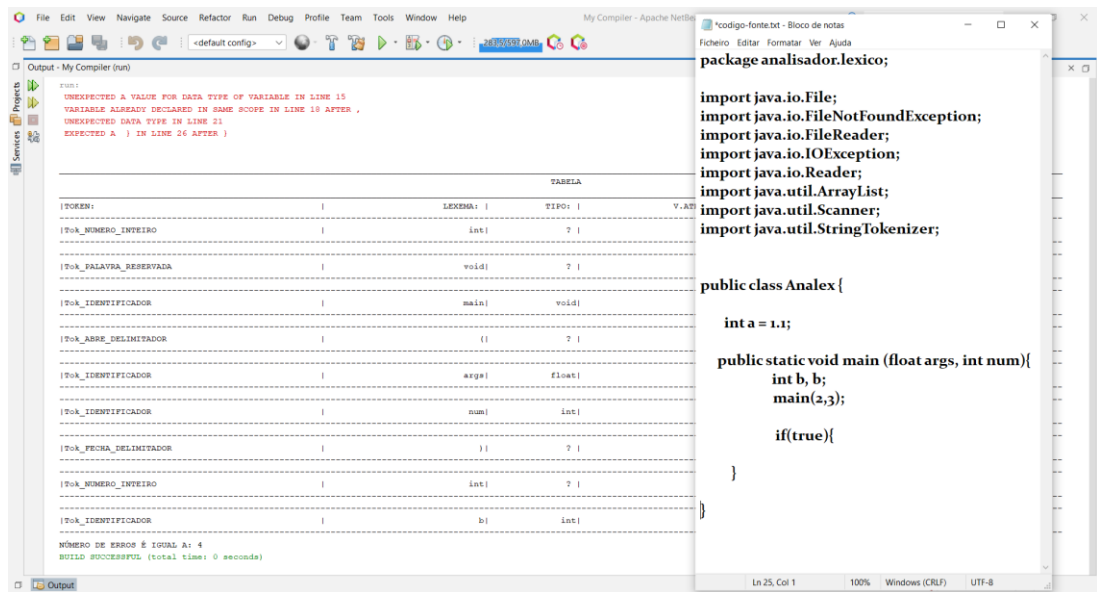


figura 11. Resultado da execução do