**Explicação Etapa 1**

**Analisador Léxico (Lexer)**

O analisador léxico é a primeira etapa da compilação. Sua função é ler o código-fonte como uma sequência de caracteres e transformá-lo em tokens, que são unidades léxicas significativas como palavras-chave (int, float), identificadores (idade, nome), operadores (=, +) e símbolos (;, ,).

Além disso, o léxico também:

* Ignora espaços, tabulações e comentários.
* Detecta erros léxicos, como caracteres inválidos ou identificadores malformados (ex: 2variavel).
* Classifica cada token com seu tipo e posição (linha e coluna), o que ajuda na análise sintática.

**Exemplo de entrada:**

int idade = 30;

**Tokens gerados:**

* INT, IDENTIFICADOR(idade), IGUAL, NUM\_INT(30), PVIRG

**Analisador Sintático (Parser)**

O analisador sintático é a **segunda etapa da compilação**. Ele recebe os tokens do léxico e **verifica se a sequência está de acordo com a gramática da linguagem** — ou seja, se a estrutura do código está correta.

Ele trabalha como um verificador de **ordem e combinação**: não basta que os tokens estejam certos, eles precisam **estar na ordem certa**.

**Funções do parser:**

* Identifica **declarações válidas** (como int idade = 30;).
* Detecta **erros de sintaxe**, como int = 30;, que é inválido porque não tem um identificador antes do =.
* Garante que regras como **fim com ponto e vírgula**, **lista separada por vírgulas** e **tipos permitidos** estejam sendo respeitadas.

**Estrutura dos Arquivos**

* **AnalisadorLexico.flex**  
  Arquivo que define o analisador léxico. Nele estão especificados os padrões das expressões regulares que reconhecem palavras-chave (int, float, char), identificadores, números, literais e operadores. Também trata erros léxicos, como identificadores inválidos que começam com números.
* **AnalisadorSintatico.cup**  
  Contém a gramática usada pelo analisador sintático gerado com CUP. Define os tipos, estruturas de declaração, lista de identificadores, atribuições e valores. É responsável por validar a estrutura das declarações conforme as regras da linguagem.
* **AnalisadorLexico.java e AnalisadorSintatico.java**  
  Arquivos gerados automaticamente pelas ferramentas JFlex e CUP, a partir dos arquivos .flex e .cup. Esses arquivos implementam os analisadores propriamente ditos.
* **sym.java**  
  Classe que contém os símbolos terminais utilizados pela gramática. Também é gerada automaticamente pelo CUP.
* **TesteLexico.java**  
  Classe de teste utilizada para verificar apenas o analisador léxico. Lê o arquivo teste.txt e imprime os tokens identificados com seus valores, linha e coluna.
* **teste.txt**  
  Arquivo de entrada com exemplos de declarações de variáveis. É usado para testar o funcionamento dos analisadores.
* **Arquivos .class**  
  Arquivos compilados das classes Java (.java) para execução no interpretador da JVM.
* **jflex-full-1.8.1.jar e java-cup-11b.jar**  
  Bibliotecas necessárias para a geração e execução dos analisadores. São usadas para processar os arquivos .flex e .cup.

**Como Compilar e Executar o Projeto**

**Pré-requisitos**

* Java JDK instalado (recomendado: versão 8 ou superior)
* Terminal ou prompt de comando (cmd, PowerShell, etc.)
* Arquivos java-cup-11b.jar e jflex-full-1.8.1.jar disponíveis na mesma pasta do projeto

**Etapa 1 – Gerar o Analisador Léxico com JFlex**

java -jar jflex-full-1.8.1.jar AnalisadorLexico.flex

**Etapa 2 – Gerar o Analisador Sintático com CUP**

java -jar java-cup-11b.jar -parser AnalisadorSintatico -symbols sym AnalisadorSintatico.cup

**Etapa 3 – Compilar os Arquivos Java**

javac -cp ".;java-cup-11b.jar" AnalisadorLexico.java AnalisadorSintatico.java sym.java TesteLexico.java

**Etapa 4 – Rodar o Analisador Léxico**

java -cp ".;java-cup-11b.jar" TesteLexico

**Etapa 5 – Rodar o Analisador Sintático Completo**

java -cp ".;java-cup-11b.jar" AnalisadorSintatico

**Exemplos de Entradas Válidas**

int x;

float y = 2.5;

char c = 'A';

char nome = "Ana";

int a, b, c;

**Exemplos de Entradas Inválidas**

double d; // tipo não suportado

int = 30; // faltando identificador

123abc = 10; // identificador inválido

**Explicação Etapa 2**

Esta etapa do trabalho tem como objetivo desenvolver os analisadores léxico e sintático para o reconhecimento dos comandos de seleção `if` e `switch-case`, no estilo da linguagem C. A análise será realizada com o uso das ferramentas JFlex (analisador léxico) e JavaCUP (analisador sintático).

**Requisitos de Sintaxe**

Os comandos que devem ser reconhecidos nesta etapa são:

* Comando if:
* - Sintaxe: if(condição) { <comandos> } else { <comandos> }
* - A condição utiliza operadores relacionais (==, !=, <, >, <=, >=).
* - Os comandos são expressões matemáticas simples terminadas por ponto e vírgula.
* - O bloco else é opcional.
* Comando switch-case:
* - Sintaxe: switch(variável) { case id: { <comandos> break; } default: { <comandos> } }
* - Cases podem ser múltiplos ou únicos.
* - O bloco default é opcional.

**Analisador Léxico (.flex)**

O analisador léxico foi construído com base na especificação das expressões regulares e estruturas da linguagem C. Ele reconhece identificadores, palavras-chave, operadores relacionais e aritméticos, símbolos, literais e ignora espaços em branco e comentários.

**Analisador Sintático (.cup)**

O analisador sintático define a gramática para reconhecer os comandos `if` e `switch-case`, seguindo a sintaxe e semântica da linguagem C. Foram utilizados não-terminais como `comando\_if`, `comando\_switch`, `condicao`, `comando`, entre outros. Também foi implementado suporte à precedência dos operadores matemáticos.

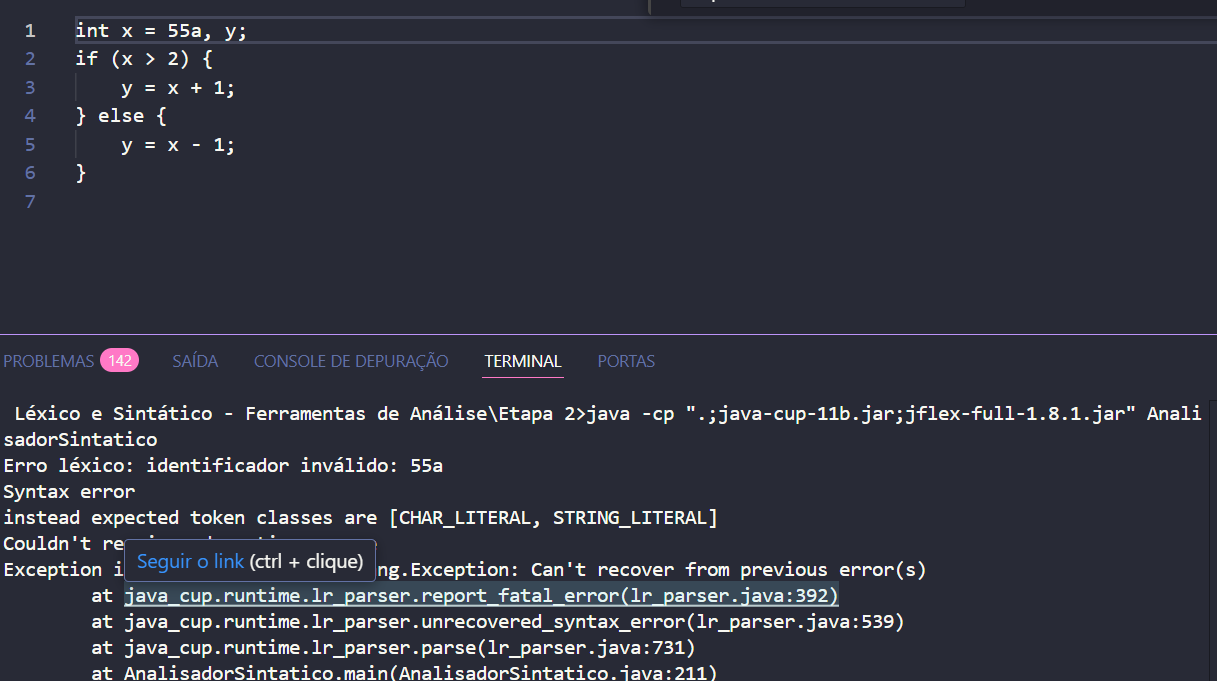
**Testes Realizados**

Foram realizados testes com arquivos de entrada válidos e inválidos para garantir a robustez dos analisadores.

**Exemplo de código válido:**



**Exemplo de código inválido:**



**Como Compilar e Executar o Projeto**

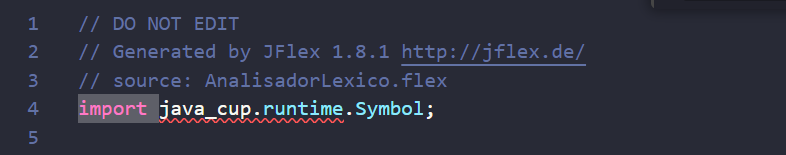
**Pré-requisitos**

* Java JDK instalado (recomendado: versão 8 ou superior)
* Terminal ou prompt de comando (cmd, PowerShell, etc.)
* Arquivos java-cup-11b.jar e jflex-full-1.8.1.jar disponíveis na mesma pasta do projeto

**Etapa 1 – Gerar o Analisador Léxico com JFlex**

java -jar jflex-full-1.8.1.jar AnalisadorLexico.flex

OBS: após essa etapa, é necessário inserir no AnalisadorLexico.java o import java\_cup.runtime.Symbol; na primeira linha, conforme imagem abaixo.



**Etapa 2 – Gerar o Analisador Sintático com CUP**

java -jar java-cup-11b.jar -parser AnalisadorSintatico -symbols sym AnalisadorSintatico.cup

Isso vai gerar os seguintes arquivos:

* AnalisadorSintatico.java
* sym.java

**Etapa 3 – Criar o arquivo de teste**

Crie um arquivo teste.txt com código a ser testado, por exemplo:

int x = 5, y;

if (x > 2) {

y = x + 1;

} else {

y = x - 1;}

**Etapa 4 – Compilar os Arquivos Java**

javac -cp ".;java-cup-11b-runtime.jar" \*.java

**Etapa 5 – Rodar o Analisador Léxico**

java -cp ".;java-cup-11b-runtime.jar" AnalisadorSintatico

Com isso o retorno deve ser esse

**? Código válido.**