Relatório de Implementação do Analisador Léxico

Este relatório descreve o processo de implementação de um analisador léxico para a leitura e processamento de um código-fonte. A implementação foi dividida em diversas etapas, abrangendo a leitura do arquivo fonte, remoção de elementos desnecessários e separação dos lexemas, até a formação dos tokens e sua categorização. As classes principais envolvidas no processo são `leituraLinha.java` e a classe responsável pelo analisador léxico.

1. Estrutura Geral

1.1 Classes e Aplicação
A implementação foi organizada em três partes:
- Classe leituraLinha.java: Responsável pela leitura e processamento do arquivo `codigo-fonte.txt`.
- Classe Analisador Léxico: Responsável pela categorização e formação dos tokens.
- Aplicação Principal: Onde essas classes são integradas e o processo é executado.

2. Processamento do Código-Fonte

2.1 Leitura do Arquivo
A classe `leituraLinha.java` tem como principal objetivo ler o conteúdo do arquivo `codigo-fonte.txt`. A leitura é feita linha por linha, e o tratamento subsequente separa os lexemas encontrados no código.

2.2 Primeira Etapa: Remoção de Quebras de Linha
Na primeira etapa do processo, todas as quebras de linha são removidas. A estratégia utilizada para isso foi converter o código-fonte em um **ArrayList** e, em seguida, utilizar o método:

```
removeIf(lex -> lex.isEmpty() || lex.isBlank());
```

Essa abordagem remove todos os elementos vazios (`lex.isEmpty()`) e as linhas em branco (`lex.isBlank()`).

2.3 Segunda Etapa: Remoção de Tabulações
Na segunda etapa, as tabulações do código são removidas. A implementação foi feita com um loop `foreach` que percorre cada linha do ArrayList. O primeiro caractere de cada linha é verificado para identificar tabulações (`\t`). Utilizando um StringBuffer, uma nova linha sem a tabulação é criada e substitui a linha original na lista.

2.4 Terceira e Quarta Etapa: Remoção de Comentários
- Comentários de Linha: Para remover comentários em linha (`//`), novamente foi utilizado o método `removeIf`, onde as condições verificam se o primeiro e segundo caractere de uma linha são barras (`/`).

- Comentários de Bloco: A implementação para remover comentários de bloco ainda não está funcional, necessitando melhorias.

2.5 Separação dos Lexemas
Após a remoção de quebras de linha, tabulações e comentários, o processo de leitura prossegue com a divisão do código em lexemas. Os critérios para essa divisão são:
- Espaços em branco
- Símbolos especiais

Há também um tratamento específico para textos entre aspas, que não passam pelo processo de divisão e são lidos exatamente como estão.

2.6 Retorno da Lista de Lexemas
A classe `leituraLinha.java` retorna uma lista com todos os lexemas encontrados. Essa lista será passada como argumento para o construtor do analisador léxico, onde os tokens são formados e armazenados na lista de tokens, e os identificadores são inseridos na tabela de símbolos.

3. Analisador Léxico

3.1 Definição de Atributos
Na classe do Analisador Léxico, são definidos diversos atributos que armazenam:
- Palavras reservadas
- Símbolos especiais
- Operadores aritméticos, relacionais e lógicos

3.2 Categorização dos Tokens
A categorização de cada token é feita utilizando **expressões regulares** que definem padrões para cada tipo de token. Esses padrões são aplicados em uma sequência de condicionais que verificam os lexemas. Se algum lexema não puder ser categorizado, uma exceção é lançada exibindo o token desconhecido.

3.3 Armazenamento de Tokens e Identificadores
Todos os tokens e identificadores são armazenados nas listas de tokens e na tabela de símbolos, podendo ser acessados através dos métodos `get` de cada atributo.

Conclusão
A implementação do analisador léxico foi desenvolvida em etapas claras e bem estruturadas, com a classe `leituraLinha.java` desempenhando um papel crucial na leitura e processamento do código-fonte. O processo de categorização e identificação dos tokens ocorre de maneira eficiente, com o uso de expressões regulares. O sistema ainda pode ser melhorado, especialmente na implementação da remoção de comentários de bloco, mas o fluxo geral de análise léxica está funcional.