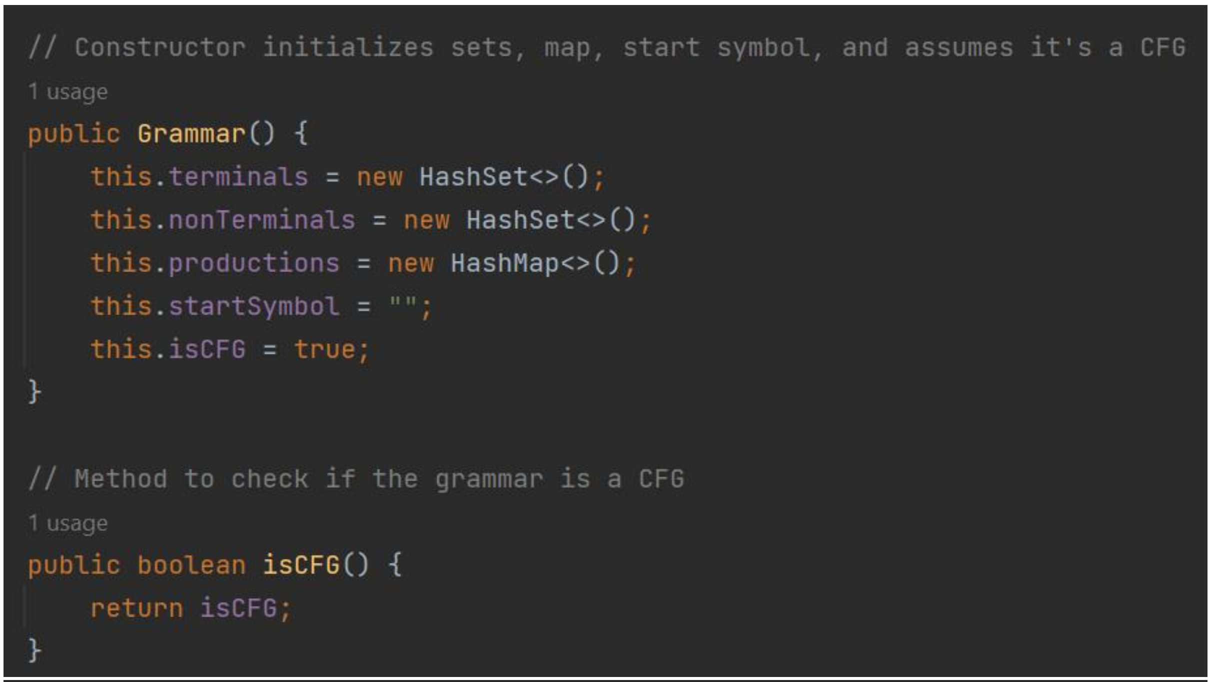
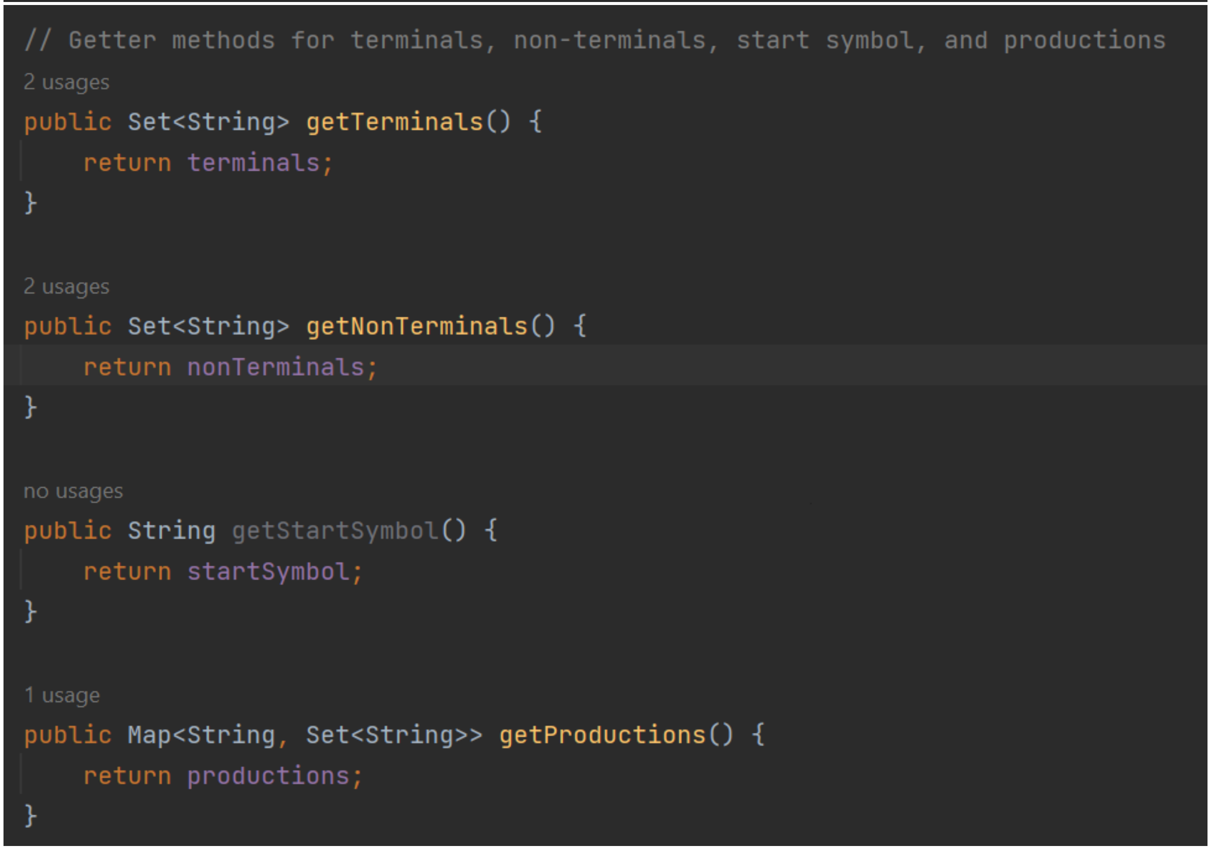
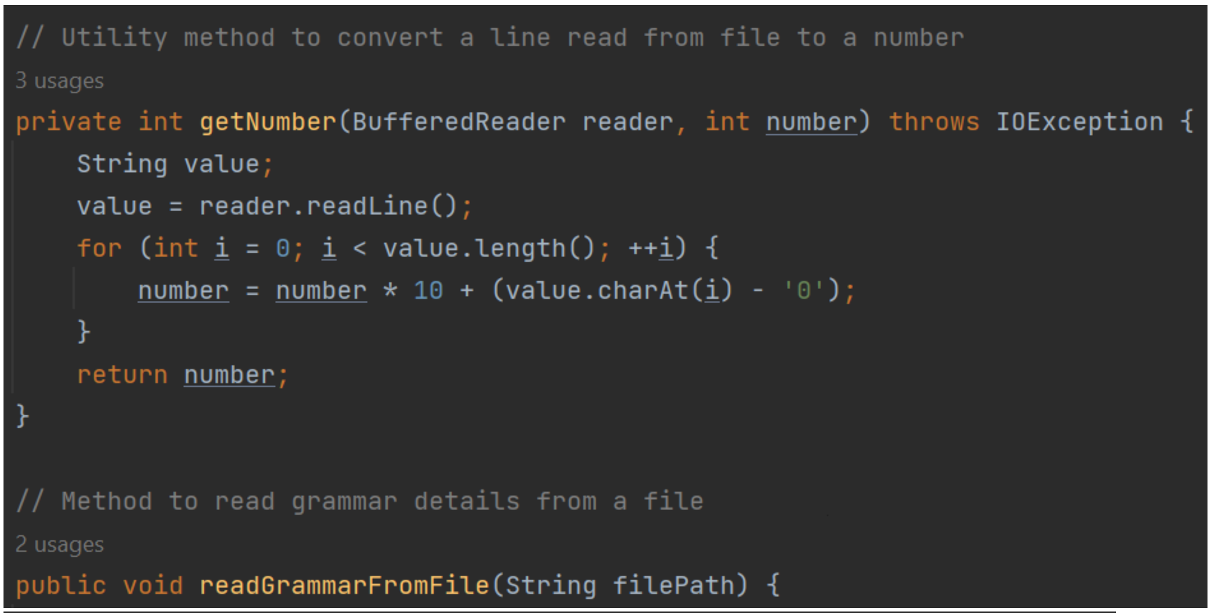
https://github.com/RazvanAndreiMoga/LFTC



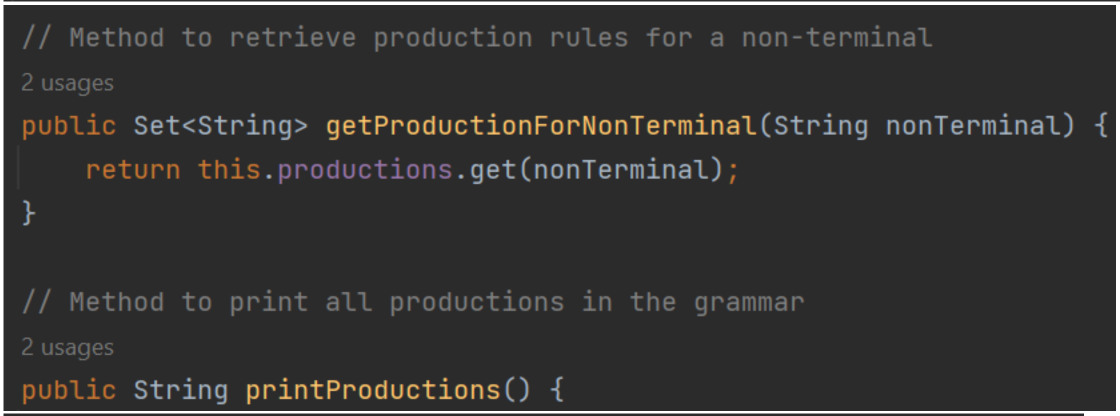
```
// Constructor initializes sets, map, start symbol, and assumes it's a CFG
1 usage
public Grammar() {
    this.terminals = new HashSet<>();
    this.nonTerminals = new HashSet<>();
    this.productions = new HashMap<>();
    this.startSymbol = "";
    this.isCFG = true;
}

// Method to check if the grammar is a CFG
1 usage
public boolean isCFG() {
    return isCFG;
}
```



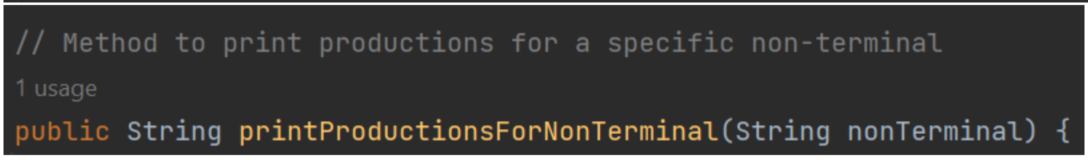
```
// Getter methods for terminals, non-terminals, start symbol, and productions
2 usages
public Set<String> getTerminals() {
    return terminals;
}

2 usages
public Set<String> getNonTerminals() {
    return nonTerminals;
}

no usages
public String getStartSymbol() {
    return startSymbol;
}

1 usage
public Map<String, Set<String>> getProductions() {
    return productions;
}
```



```
// Utility method to convert a line read from file to a number
3 usages
private int getNumber(BufferedReader reader, int number) throws IOException {
    String value;
    value = reader.readLine();
    for (int i = 0; i < value.length(); ++i) {
        number = number * 10 + (value.charAt(i) - '0');
    }
    return number;
}

// Method to read grammar details from a file
2 usages
public void readGrammarFromFile(String filePath) {
```



```
// Method to retrieve production rules for a non-terminal
2 usages
public Set<String> getProductionForNonTerminal(String nonTerminal) {
    return this.productions.get(nonTerminal);
}

// Method to print all productions in the grammar
2 usages
public String printProductions() {
```



```
// Method to print productions for a specific non-terminal
1 usage
public String printProductionsForNonTerminal(String nonTerminal) {
```

The EBNF of the input files (g1.txt, g2.txt):

nz_digit := “1” | ”2” | .. | ”9”

digit := “0” | ”1” | ”2” | .. | ”9”

number := nz_digit {digit}

letter := “a” | “b” | .. | “z” | “A” | “B” | .. | “Z”

character := letter | digit

string := character {character}

first_line := number (* it represents the number of nonterminals *)

second_line := {string} (* it represents the nonterminals *)

third_line := number (* it represents the number of terminals *) fourth_line := {string} (* it represents the terminals *)

fifth_line := number (* it represents the number of productions *)

sixth_line := {string “->” string} (* it represents the productions *)

seventh_line := string (* it represents the start symbol *)

inputFile := first_line second_line third_line fourth_line fifth_line sixth_line seventh_line