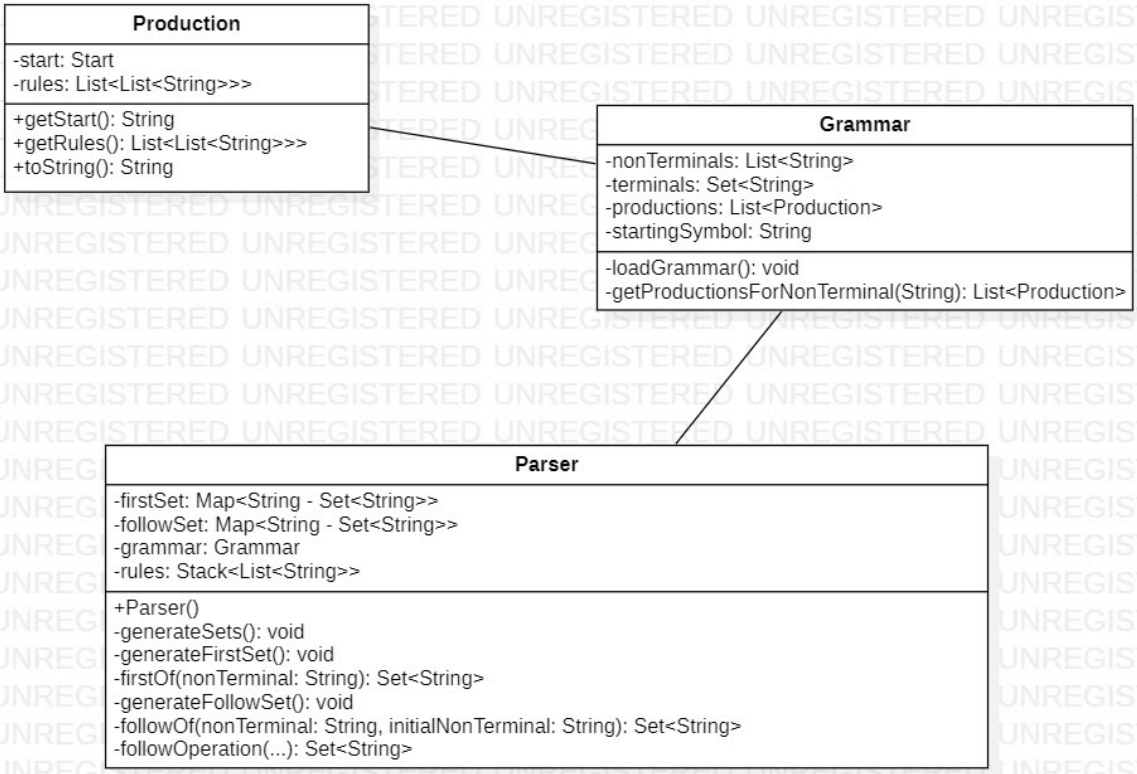
https://github.com/mocadavid/LFTC/tree/Lab5

```java
/**
* The grammar class.
*/
public class Grammar {

   private List<String> nonTerminals;
   private Set<String> terminals;
   private List<Production> productions;
   private String startingSymbol;

/**
constructor
*/
   public Grammar();

   public List<String> getNonTerminals();
   public Set<String> getTerminals();
   public List<Production> getProductions();
   public String getStartingSymbol();

   /**
    * Reads a grammar definition from a file.
    */
   private void loadGrammar();

   /**
    * Finds all productions for a given nonterminal.
    * @param nonTerminal: String
    * @return The productions found: List<Production>
    */
   public List<Production> getProductionsForNonTerminal(String nonTerminal);
}

/**
* Defines a set of productions for a nonterminal.
*/
public class Production {
   //nonterminal
   private String start;
   //list of productions
   private List<List<String>> rules;
```

```
    Production(String start, List<List<String>> rules);
    String getStart();
    List<List<String>> getRules();

    /**
     * Build all productions for a nonterminal as a string.
     * @return the result of the built: String
     */
    public String toString();
}

/**
 * Class which can generate the first and follow sets.
 */
public class Parser {
    private Map<String, Set<String>> firstSet;
    private Map<String, Set<String>> followSet;
    private Grammar grammar;
    private static Stack<List<String>> rules = new Stack<>();

    /**
     *Constructor
     */
    public Parser();

    /**
     * Initializing the first and follow sets
     */
    private void generateSets();

    /**
     * Generating first for every nonTerminal.
     */
    private void generateFirstSet();

    /**
     *Generates the first set for the given nonTerminal.
     * @param nonTerminal: nonTerminal: String
     * @return The set of terminals for the given nonTerminal.
     */
    private Set<String> firstOf(String nonTerminal);

    /**
     *Generating the follow set for all the nonTerminals.
     */
    private void generateFollowSet();
```

```
    /*
     * Analyses the productions in which the given nonTerminal is present and calls accordingly the follow operations with the needed values.
     * @param nonTerminal the given nonTerminal which we examine: String
     * @param initialNonTerminal the starting point for which we search for the follow set: String
     * @return the finalResult of the follow set: Set<String>
     */
    private Set<String> followOf(String nonTerminal, String initialNonTerminal);

    /**
     * Decides upon the case of the follow in which we are.
     * @param nonTerminal the nonTerminal for which we search follow: String
     * @param intermediaryResult the list in we save the found elements so far: Set<String>
     * @param terminals the terminals from the grammar: Set<String>
     * @param productionStart the starting nonTerminal of the production
     * @param rule the current production we analyse: String
     * @param indexNonTerminal the index of the nonTerminal: int
     * @param initialNonTerminal the given nonTerminal: String
     * @return the result of the follow operation for the given nonTerminal staring from the initialNonTerminal: Set<String>
     */
    private Set<String> followOperation(String nonTerminal, Set<String> intermediaryResult, Set<String> terminals, String productionStart, List<String> rule, int indexNonTerminal, String initialNonTerminal);
```


Production
-start: Start
-rules: List<List<String>>>
+getStart(): String
+getRules(): List<List<String>>>
+toString(): String
Grammar
-nonTerminals: List<String>
-terminals: Set<String>
-productions: List<Production>
-startingSymbol: String
-loadGrammar(): void
-getProductionsForNonTerminal(String): List<Production>
Parser
-firstSet: Map<String - Set<String>>
-followSet: Map<String - Set<String>>
-grammar: Grammar
-rules: Stack<List<String>>
+Parser()
-generateSets(): void
-generateFirstSet(): void
-firstOf(nonTerminal: String): Set<String>
-generateFollowSet(): void
-followOf(nonTerminal: String, initialNonTerminal: String): Set<String>
-followOperation(...): Set<String>