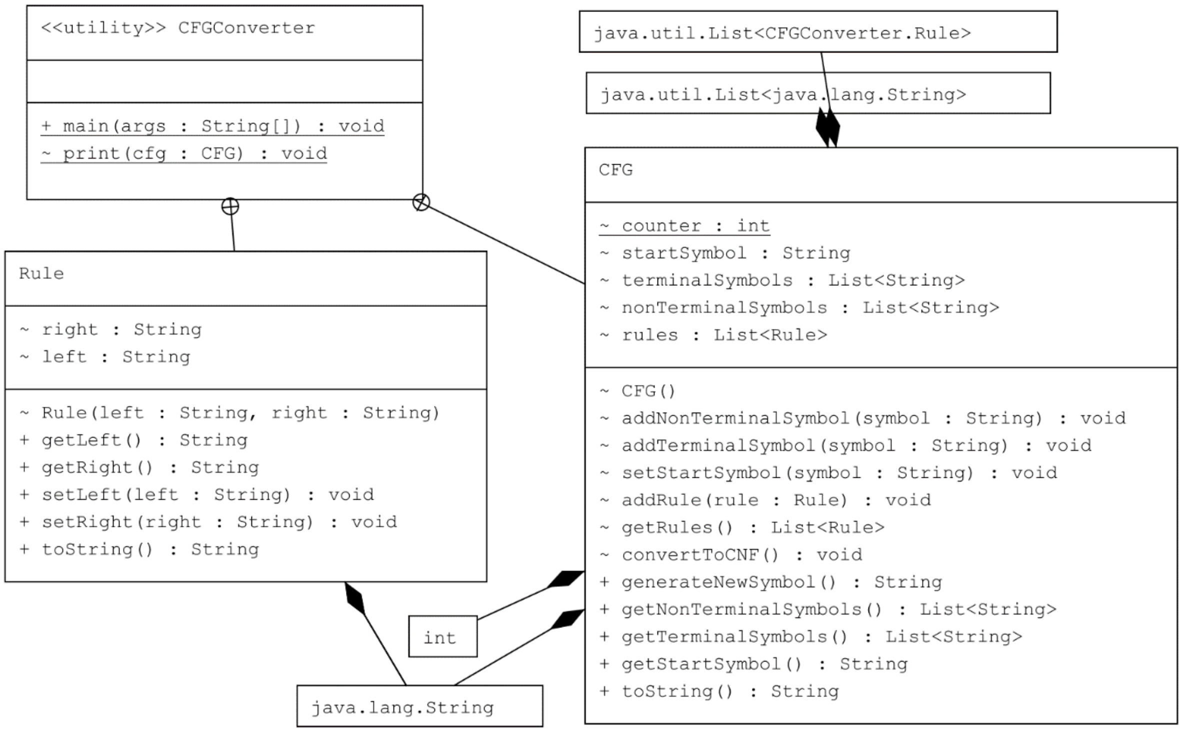
# CS410 Project 2

**Berkecan Koçyiğit S0210632**



<<utility>> CFGConverter
+ main(args : String[]) : void
~ print(cfg : CFG) : void
java.util.List<CFGConverter.Rule>
java.util.List<java.lang.String>
CFG
~ counter : int
~ startSymbol : String
~ terminalSymbols : List<String>
~ nonTerminalSymbols : List<String>
~ rules : List<Rule>
~ CFG()
~ addNonTerminalSymbol(symbol : String) : void
~ addTerminalSymbol(symbol : String) : void
~ setStartSymbol(symbol : String) : void
~ addRule(rule : Rule) : void
~ getRules() : List<Rule>
~ convertToCNF() : void
+ generateNewSymbol() : String
+ getNonTerminalSymbols() : List<String>
+ getTerminalSymbols() : List<String>
+ getStartSymbol() : String
+ toString() : String
Rule
~ right : String
~ left : String
~ Rule(left : String, right : String)
+ getLeft() : String
+ getRight() : String
+ setLeft(left : String) : void
+ setRight(right : String) : void
+ toString() : String
int
java.lang.String

I represented the CFG as a class. You can see the whole project's UML diagram above.

## Input file to class

I took the provided txt file and implemented it with the code below:

```
BufferedReader reader = new BufferedReader(new FileReader("src/G1.txt"));
// Create a new CFG object
CFG cfg = new CFG();

// Read the NON-TERMINAL header
String line = reader.readLine();
if (line == null || !line.equals("NON-TERMINAL")) {
    throw new IOException("Invalid input file.");
}

// Read the non-terminal symbols in the grammar
while ((line = reader.readLine()) != null) {
    // Check if we have reached the TERMINAL header
    if (line.equals("TERMINAL")) {
        break;
    }
    // Add the non-terminal symbol to the CFG object
    cfg.addNonTerminalSymbol(line);
}
```

As you can see I used Buffer Reader in order to parse the data. Then I created cfg object and added remaining information until next header. This process continues until all the information parsed.

## Rule Class

I have simple Rule Class in order to represent Rules. It has right and left Strings and their getter/setters.

## CFG Class

CFG class has startSymbol (String), terminalSymbols (String List), nonTerminalSymbols (String List), rules (Rule List) attributes for representing a CFG.

CFG Class also has conversion method which takes CFG and converts it to CFN in same object.

CFG also has helper methods other than getter/setter methods. generateNewSymbol method can generate unique Symbols for new states. It uses static counter variable in order to keep order.

### Conversion Method

My conversion method several While Loops to accomplish conversion.

My logic is:

-First eliminate the null products. To do that, inside a while loop check for “e” as righthand side for every loop for each while loop cycle and if it has “e” remove that rule from rules list and add it to the nullable list which created for keeping track of which rules are nullable. Inside that for loop check righthand side contains any nullable characters add a rule which has the version of that rule without that nullable state. Do all of these until nothing is changed in rules list.

-Second eliminate unit products. Inside an identical while loop check for righthand side’s length is 1 and if it is 1, remove it and add a new version for lefthand side is same but righthand side is same with what previous righthand side state is showing. Do all of these until nothing is changed in rules list.

-Third eliminate the more than 2 states for each rule. Create new rule with newly generated symbol for representing rest of rule. Do all of these until nothing is changed in rules list.

-Forth for every terminal-nonterminal pair. Create new state if there is no state for that terminal and combine that new state with previous non-Terminal state as new rule. And add new rule that takes new state as left side and previous terminal state as right side. Do all of these until nothing is changed in rules list.

*After calling conversion main function prints the cfg with same format.*