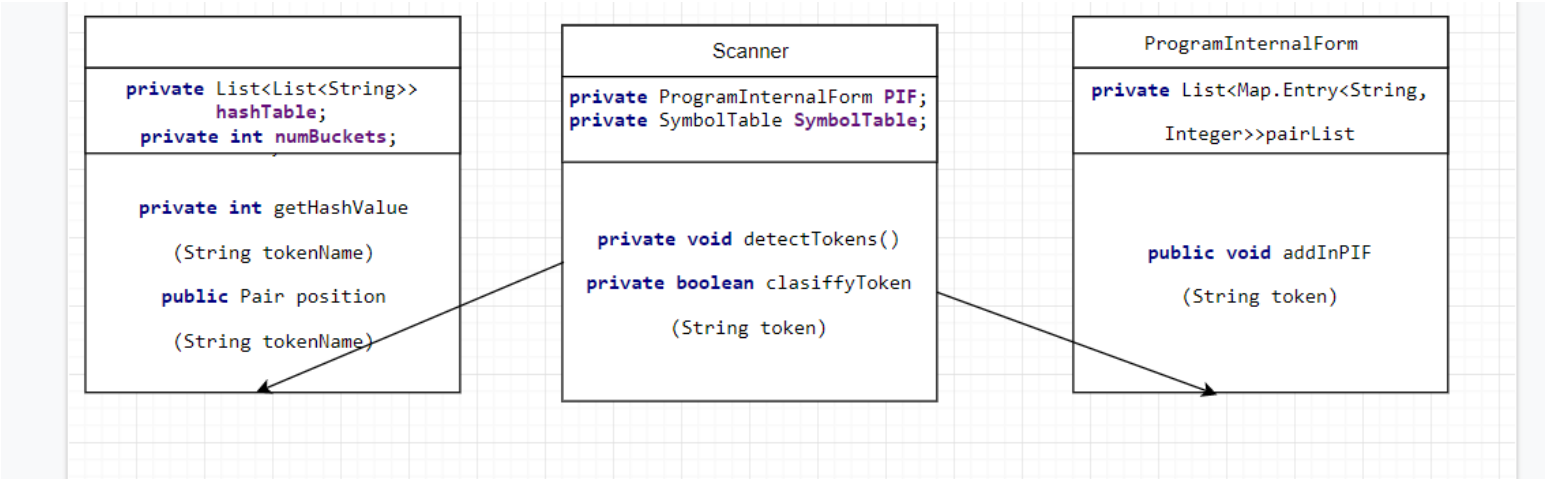
Tatar Flavia-Andreea

Git: https://github.com/tatarflavia/Formal-Languages-Compiler-Design-LAB/tree/main/lab3




```java
//checks for only letters
private boolean isIdentifier(String token){
    Pattern pattern = Pattern.compile("^[a-zA-Z]+$");
    Matcher m = pattern.matcher(token);
    return m.matches();
}


//checks to be integer, which means 0 or optional sign followed by an
integer(means a number made by several digits with no zeros in plus or at start)
private boolean isInteger(String token){
    Pattern pattern = Pattern.compile("0|^(\\+|\\-)?[1-9][0-9]+$");
    Matcher m = pattern.matcher(token);
    return m.matches();
}

//a character can be a digit or a letter
private boolean isChar(String token){
    Pattern pattern = Pattern.compile("^[a-zA-Z0-9]+$");
    Matcher m = pattern.matcher(token);
    return m.matches();
}

//a boolean can be True or False
private boolean isBoolean(String token){
    Pattern pattern = Pattern.compile("^(True|False)$");
    Matcher m = pattern.matcher(token);
    return m.matches();
}

//a string can be several characters, which can be a digit or a letter
private boolean isString(String token){
    Pattern pattern = Pattern.compile("^([a-zA-Z0-9]+)+$");
    Matcher m = pattern.matcher(token);
    return m.matches();
}
```

```
//is constant if it’s an integer, boolean, string or char
private boolean isConstant(String token){
    return isInteger(token) || isBoolean(token) || isString(token) ||
isChar(token);
}

//is reserved word if it’s list,int, bool, Initialise, char, string, Print, Read,
Do, If, Else,While,For, Return
private boolean isReservedWord(String token){
    Pattern pattern =
Pattern.compile("^(list|int|bool|Initialise|Read|Do|While|If|char|string|Print|Els
e|For|Return)$");
    Matcher m = pattern.matcher(token);
    return m.matches();
}
//is operator if it’s  "+" "-" "*" "/" "<" ">"  "<="  "=="  ">="  "sqrt" "&&"
"not"
private boolean isOperator(String token){
    Pattern pattern = Pattern.compile("^(\\+|\\-
|\\*|\\/|\\<|\\<\\=|\\=\\=|\\>\\=|sqrt|\\&\\&|not)$");
    Matcher m = pattern.matcher(token);
    return m.matches();
}
//is separator if it’s  ( ) ; { }
private boolean isSeparator(String token){
    Pattern pattern = Pattern.compile("^(\\(|\\)|\\{|\\}|\\;)$");
    Matcher m = pattern.matcher(token);
    return m.matches();
}
```