北京航空航天大學
BEIHANG UNIVERSITY

Unit1研讨课分享

计算机学院

分享人 袁子轩

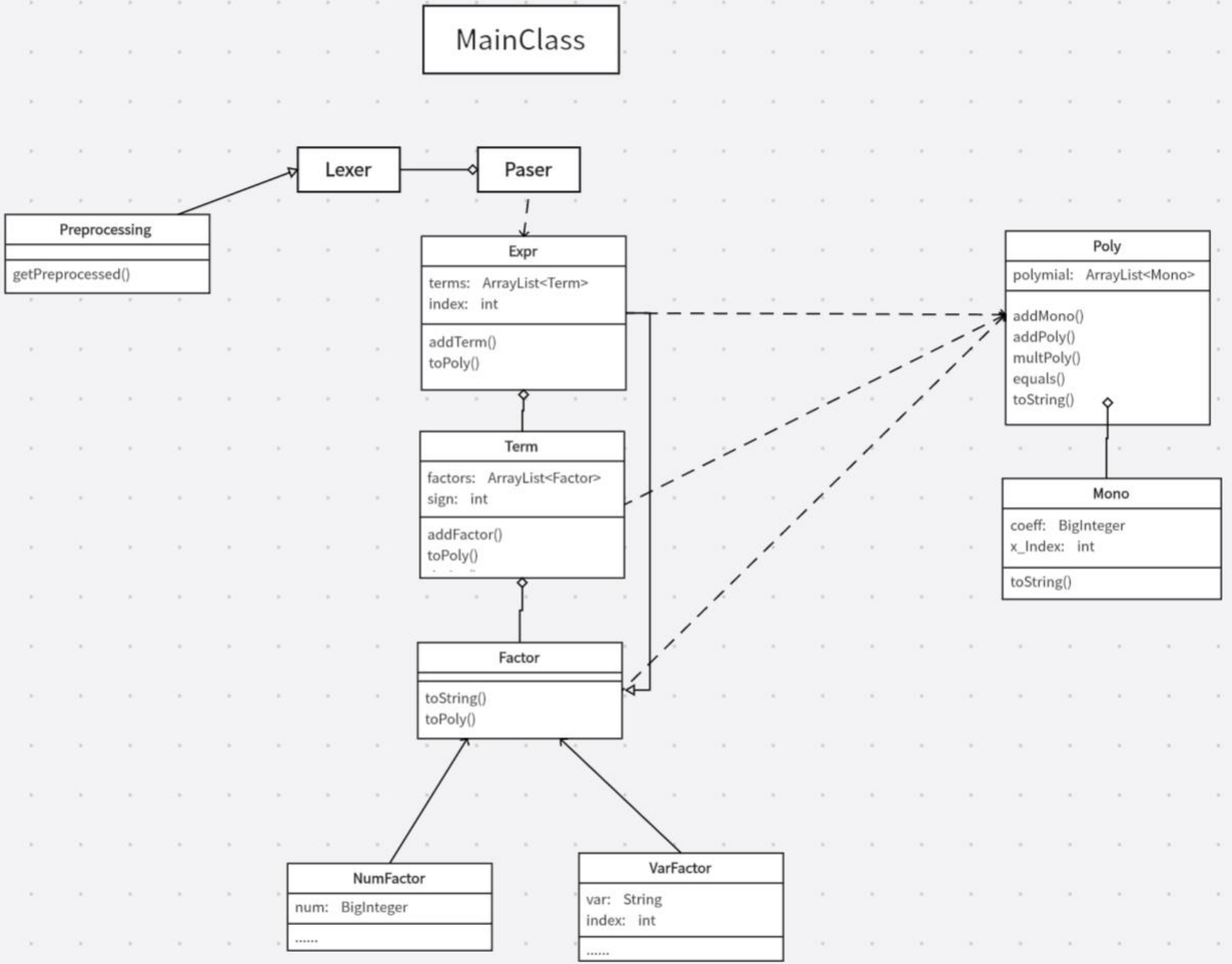
目 录

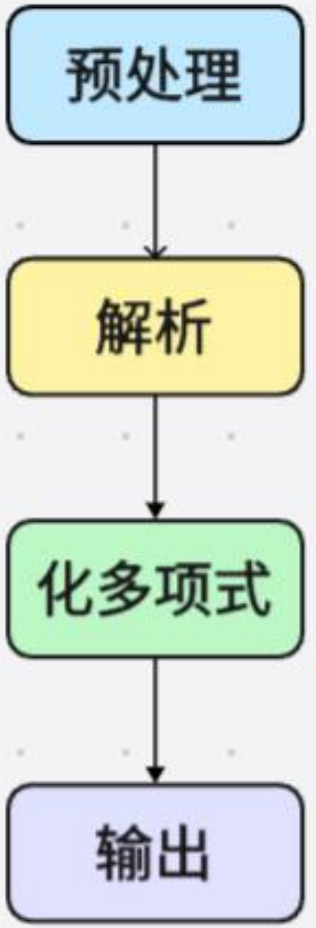
Contents

北京航空航天大學
BEIHANG UNIVERSITY

Part 1

第一次迭代


MainClass
Lexer
Paser
Preprocessing
getPreprocessed()
Expr
terms: ArrayList<Term>
index: int
addTerm()
toPoly()
Poly
polymial: ArrayList<Mono>
addMono()
addPoly()
multPoly()
equals()
toString()
Term
factors: ArrayList<Factor>
sign: int
addFactor()
toPoly()
Mono
coeff: BigInteger
x_Index: int
toString()
Factor
toString()
toPoly()
NumFactor
num: BigInteger
.....
VarFactor
var: String
index: int
.....


预处理
解析
化多项式
输出

Part 2

第二次迭代

- 三角函数
- 自定义递推函数

2.1 迭代分析

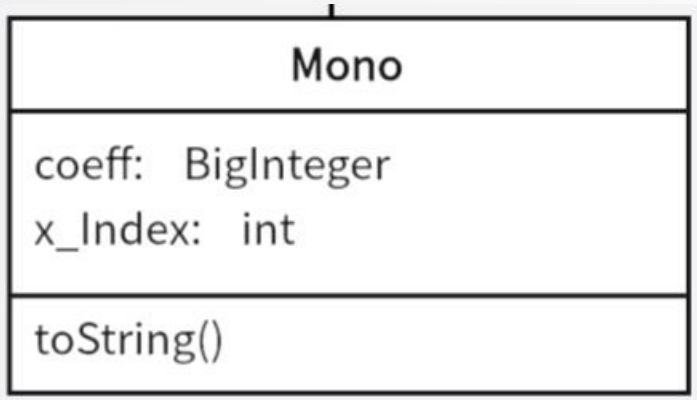
三角函数

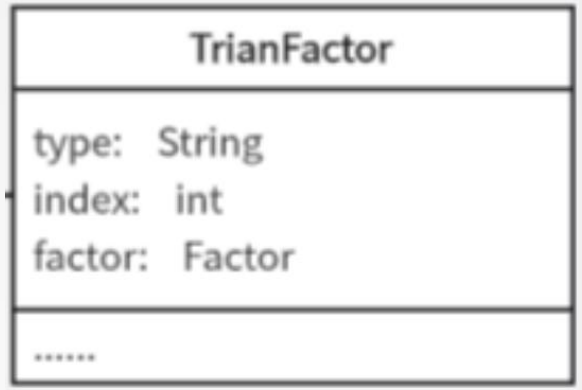
新增三角函数因子类 TrianFactor

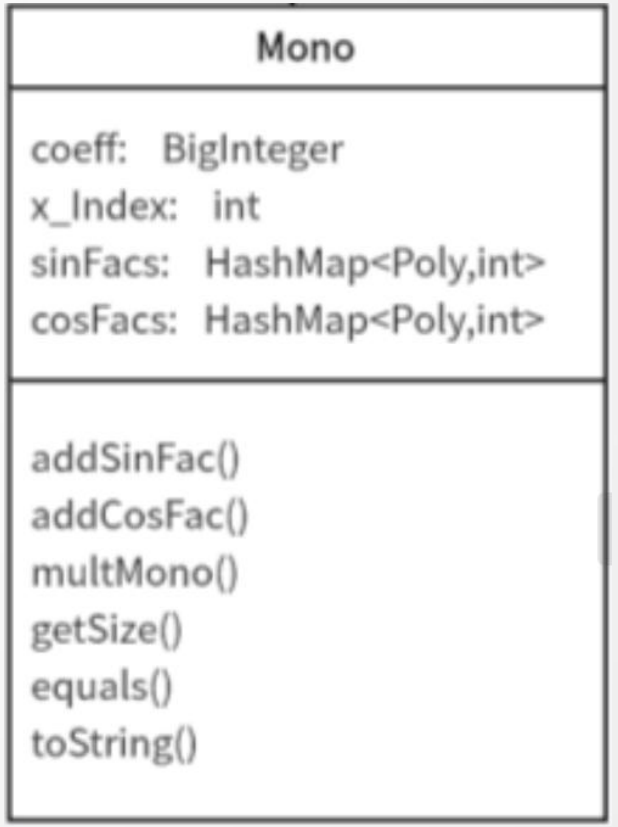
由于新的因子加入，单项式 Mono 不再像第一次作业那样形式简单统一

$$ax^n \prod_i sin(Factor_i) \prod_i cos(Factor_i)$$

为了适应新的统一形式，需要对 Mono 进行修改。为其添加两个 HashMap类型的成员变量，储存正弦因子和余弦因子；相应地，为了合并同类项，需要新增一些方法：







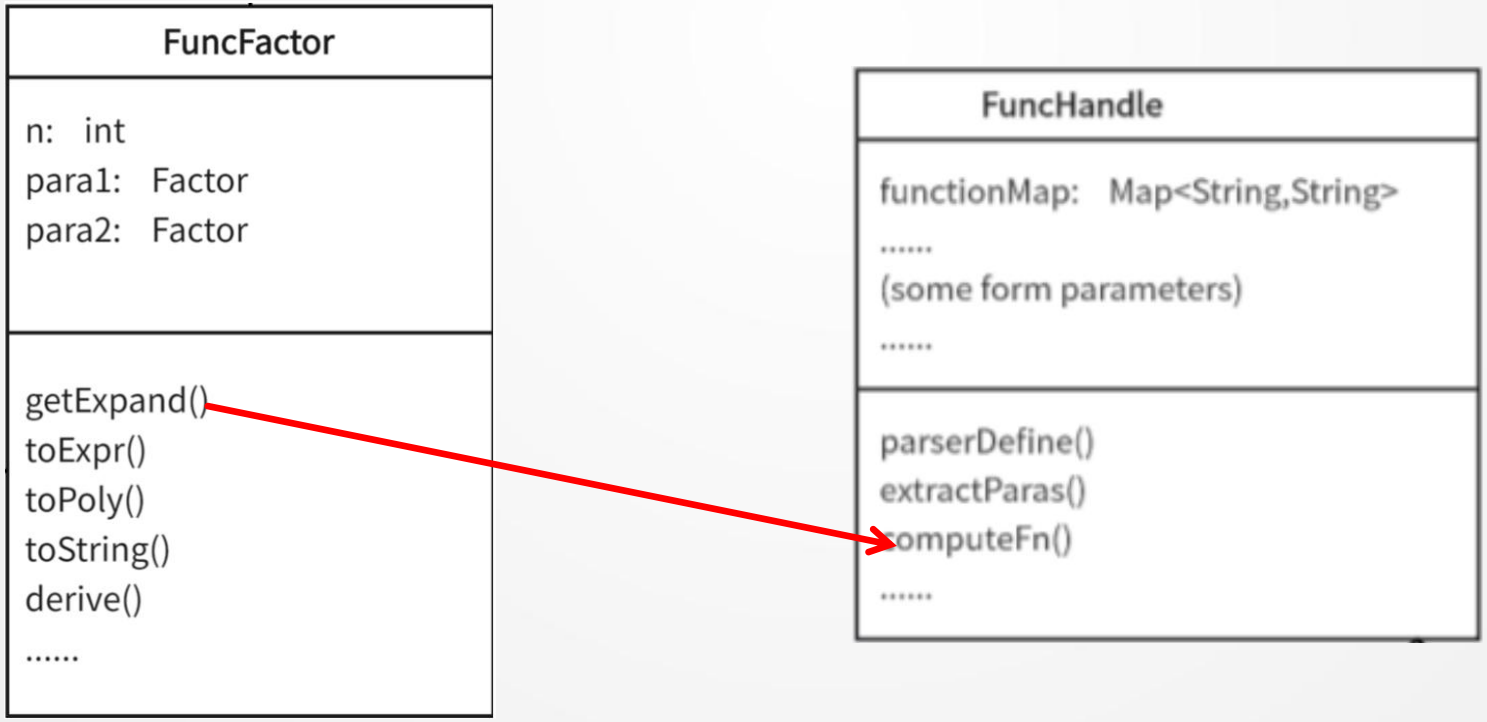
2.1 迭代分析

自定义递推函数

新增函数因子类 FuncFactor、工具类FuncHandle

FunFactor 的 getExpand() 方法，调用工具类，得到展开后的表达式（字符串类型），然后可通过新的 Lexer 和 Parser，解析成表达式Expr，再变成多项式Poly。

FuncHandle 则是用来实现 获取定义、解析参数、替换参数、计算并储存函数表达式 等操作。



2.1 迭代分析

自定义递推函数

工具类中的核心方法 computeFn(int n , String arg1 , String arg2) ， 功能是得到 f{n}(arg1 , arg2)的展开式。

思路是记忆化递归。

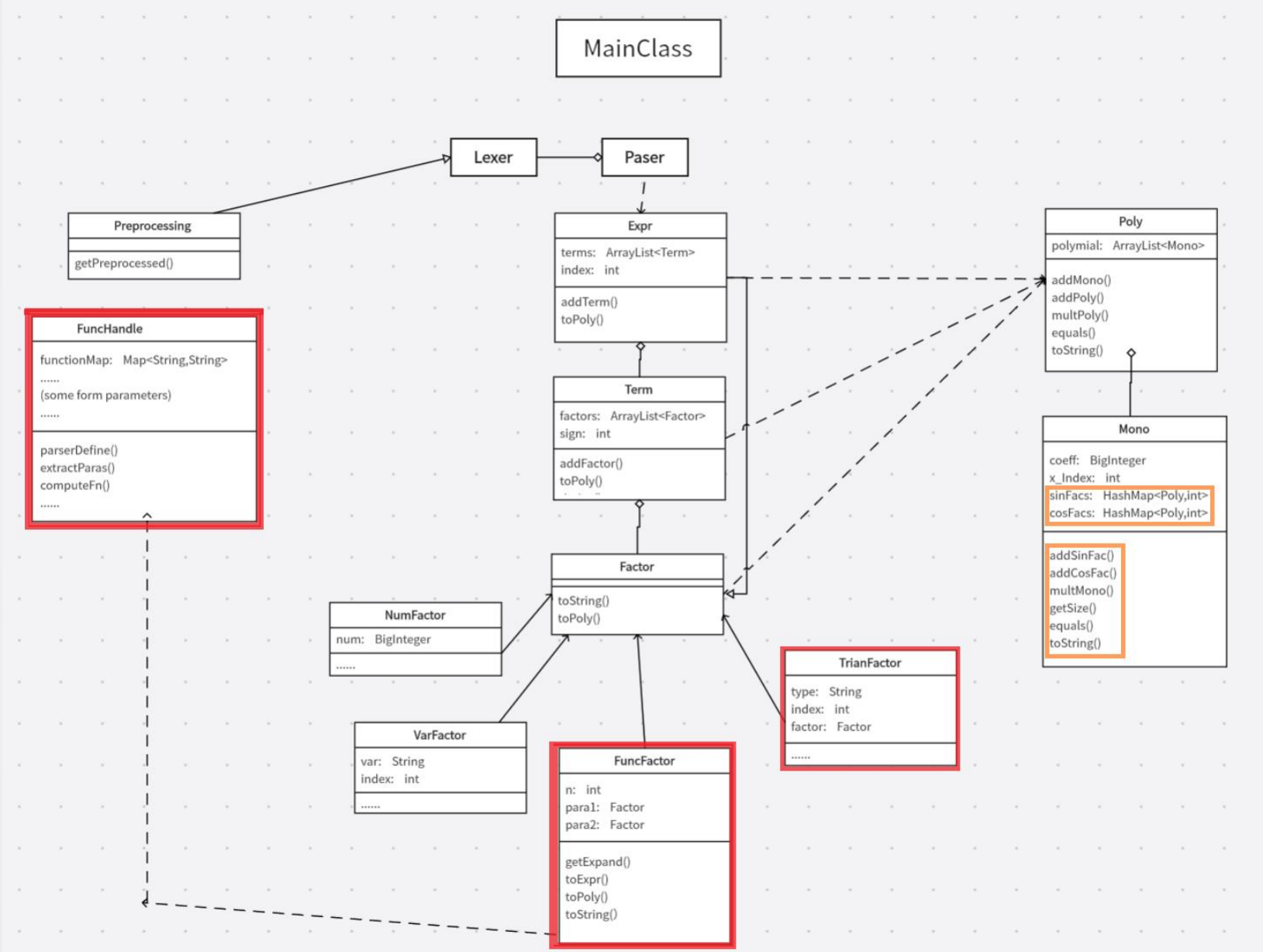
```
public static String computeFn(int n, String arg1, String arg2) {  0 个用法
    if (functionMap.containsKey(n)) {
        // 从 Map 中获取 f{n}(x,y) 表达式
        // x 替换为 arg1 , y 替换为 arg2
        // 返回替换结果
    } else {
        //根据定义式f{n}(x,y) = a*f{n-1}(.. , ..) + b*f{n-2}(.. , ..) + c
        //递归调用 computeFn(n-1 , .. , ..) 和 computeFn(n-2 , .. , ..)
        // 组合出 f{n}(x,y) 表达式，并存入 Map 中
        // x 替换为 arg1 , y 替换为 arg2
        // 返回结果
    }
}
```

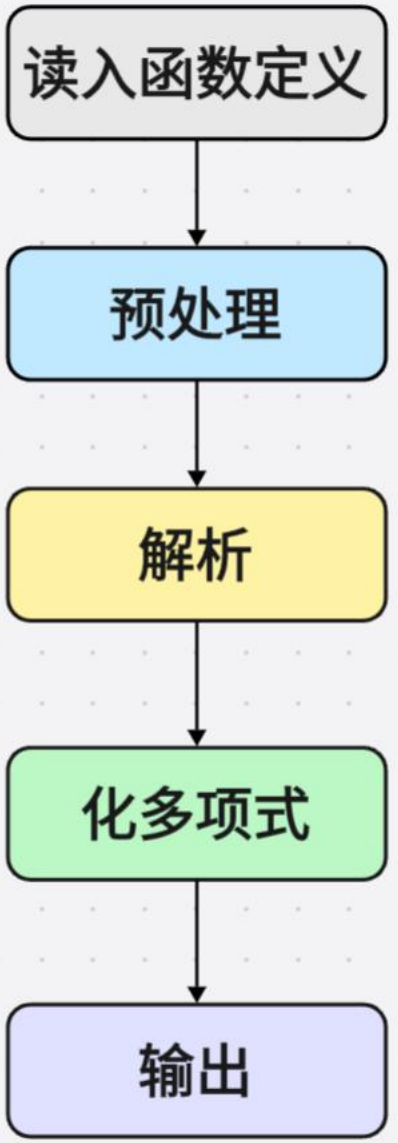
优点：

即用即取

即用即算即存

2.2 架构变化


MainClass
Lexer
Paser
Preprocessing
getPreprocessed()
Expr
terms: ArrayList<Term>
index: int
addTerm()
toPoly()
Poly
polymial: ArrayList<Mono>
addMono()
addPoly()
multPoly()
equals()
toString()
FuncHandle
functionMap: Map<String,String>
.....
(some form parameters)
.....
parserDefine()
extractParas()
computeFn()
.....
Term
factors: ArrayList<Factor>
sign: int
addFactor()
toPoly()
Mono
coeff: BigInteger
x_Index: int
sinFacs: HashMap<Poly,int>
cosFacs: HashMap<Poly,int>
addSinFac()
addCosFac()
multMono()
getSize()
equals()
toString()
Factor
toString()
toPoly()
NumFactor
num: BigInteger
.....
TrianFactor
type: String
index: int
factor: Factor
.....
VarFactor
var: String
index: int
.....
FuncFactor
n: int
para1: Factor
para2: Factor
getExpand()
toExpr()
toPoly()
toString()


读入函数定义
预处理
解析
化多项式
输出

Part 3

第三次迭代

- 求导算子
- 自定义普通函数

3.1 迭代分析

求导算子

根据其形式化表述，不难发现求导算子其实就是 ‘**dx 表达式因子**’，与第二次迭代时的三角函数因子、自定义递推函数因子相似。

故可视为一种新的因子类型 DeriveFactor，内部结构与递推函数如出一辙。

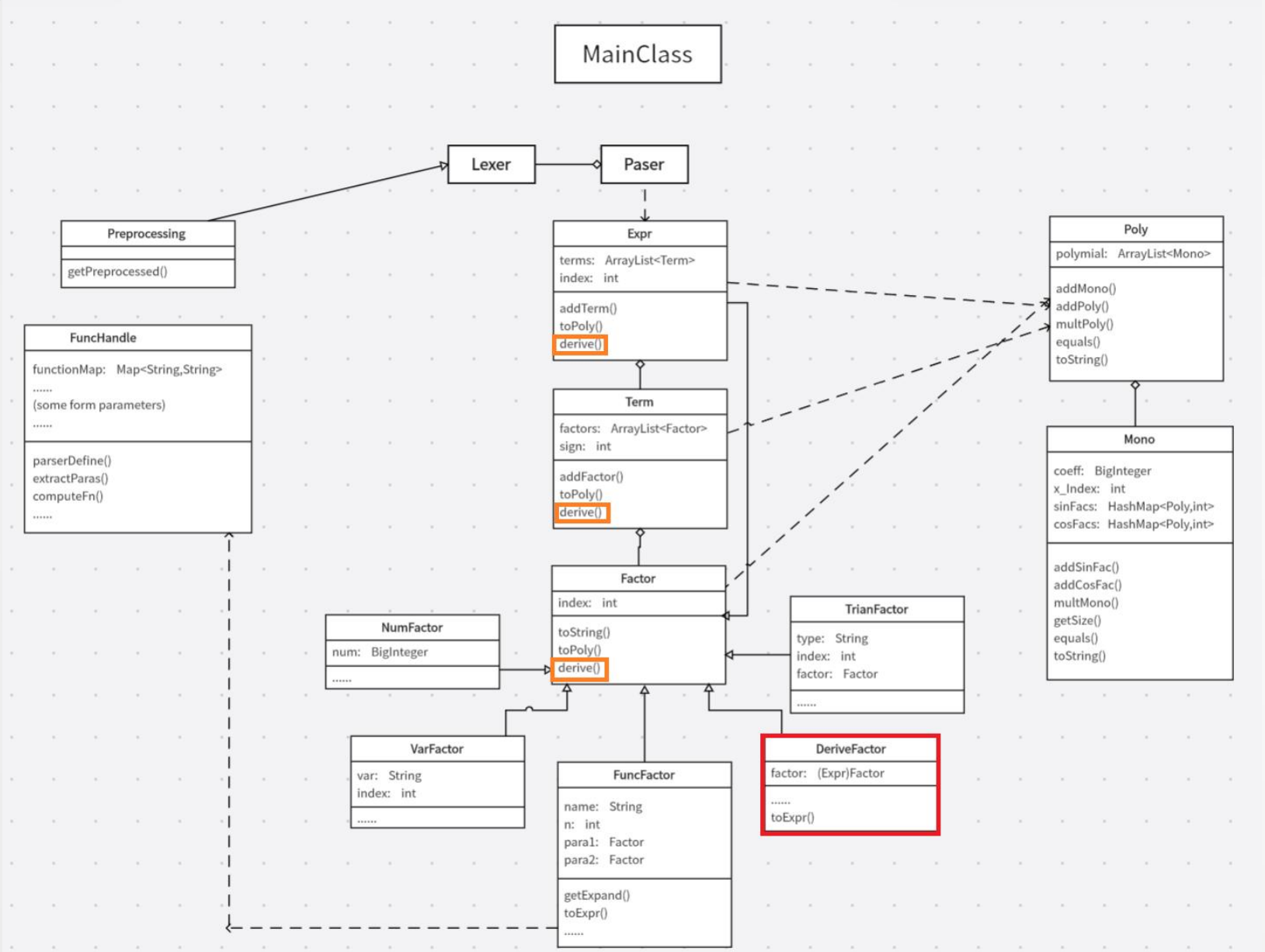
然后自上而下地，在 Expr 、 Term、各个 Factor 类里新增求导方法 derive() 即可

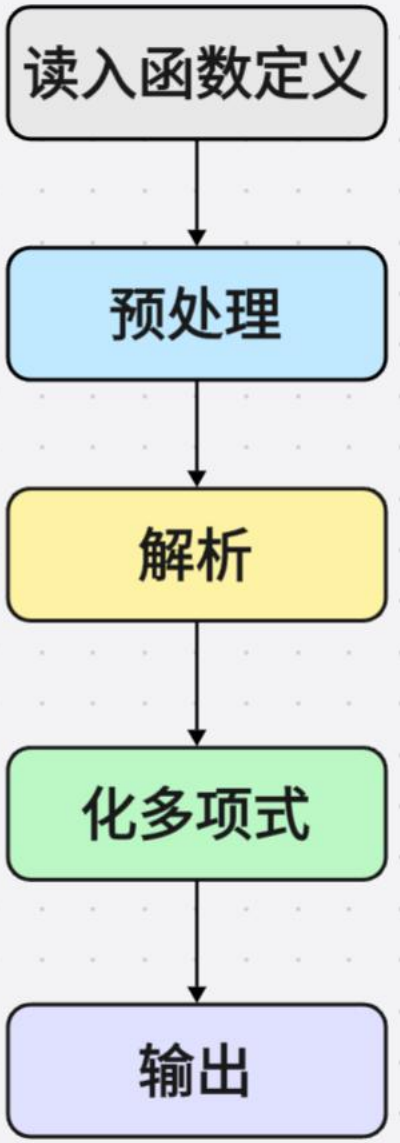
DeriveFactor
factor: (Expr)Factor
toExpr() .....

自定义普通函数

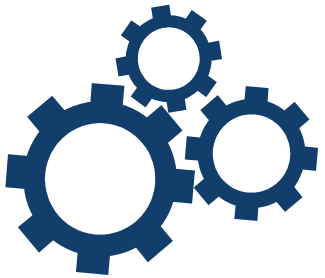
架构上无需任何调整，其地位与 递推函数的 f{0} 、f{1} 相同，仅格式略有差异。将其归入FuncFactor类，根据格式稍作修改即可

3.2 架构变化


MainClass
Lexer
Paser
Preprocessing
getPreprocessed()
Expr
terms: ArrayList<Term>
index: int
addTerm()
toPoly()
derive()
Poly
polymial: ArrayList<Mono>
addMono()
addPoly()
multPoly()
equals()
toString()
FuncHandle
functionMap: Map<String,String>
.....
(some form parameters)
.....
parserDefine()
extractParas()
computeFn()
.....
Term
factors: ArrayList<Factor>
sign: int
addFactor()
toPoly()
derive()
Mono
coeff: BigInteger
x_Index: int
sinFacs: HashMap<Poly,int>
cosFacs: HashMap<Poly,int>
addSinFac()
addCosFac()
multMono()
getSize()
equals()
toString()
Factor
index: int
toString()
toPoly()
derive()
NumFactor
num: BigInteger
.....
TrianFactor
type: String
index: int
factor: Factor
.....
VarFactor
var: String
index: int
.....
FuncFactor
name: String
n: int
para1: Factor
para2: Factor
getExpand()
toExpr()
.....
DeriveFactor
factor: (Expr)Factor
.....
toExpr()


读入函数定义
预处理
解析
化多项式
输出

北京航空航天大學
BEIHANG UNIVERSITY



Part 4

可扩展性

- 研究最终目标
- 应用前景
- 成果转化形式

4.1 可扩展性

•当新增其他种类的因子，如 e指数、ln对数时，在最底层即他们自身的类里完成toPoly()、toString()方法即可。而 Poly 的基本计算单元是 Mono ， 只需为 Mono 增添新的成员变量，使其适应统一形式，然后修改equal() 方法即可

•当新增其他像函数、求导、简单积分等计算时，可以选择将其视为一种因子，与第二、三次迭代的处理方式相同。与函数展开类似的字符串替换等需求，可交给工具类处理；与求导、积分类似的，从 Expr 到 Factor 自上而下地添加相关方法，向下调用即可

北京航空航天大學
BEIHANG UNIVERSITY

谢谢大家