Assingment 2 - External DSL Interpreter

Marc Bertelsen
berte20@student.sdu.dk

March 25, 2024

1 Desing

design

1.1 Metamodel

metamodel

redo

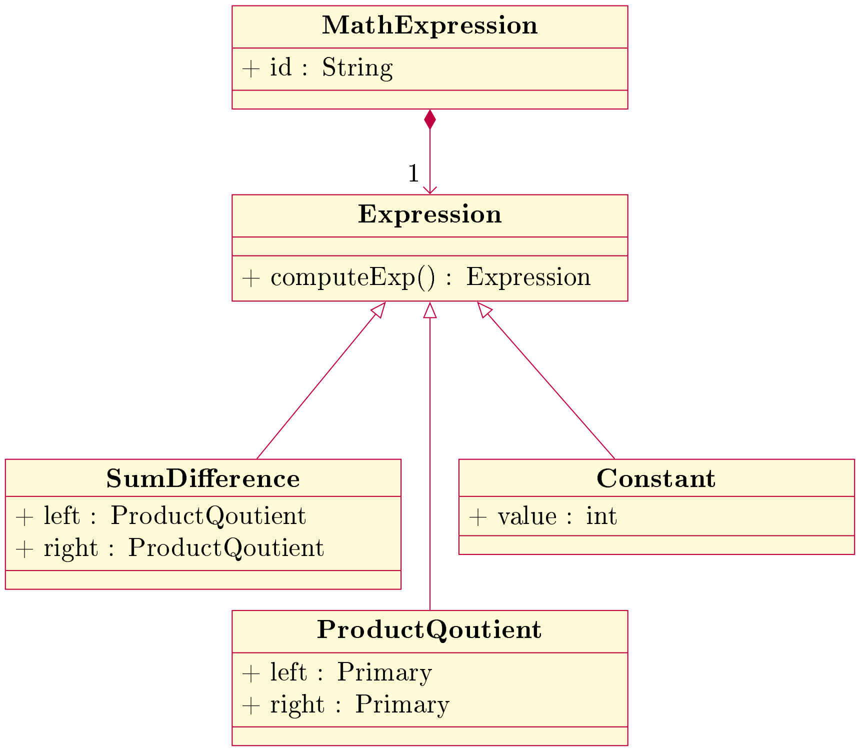


Figure 1: Mathematical Expression Metamodel

1.2 Syntax

```
var a = 1
var b = 1 + 1
var c = a + b
var d = let x = 1 + 1 in a end
```

Listing 1: Examples of *.math syntax

2 Implmentation

2.1 XText syntax

```
grammar dk.sdu.mmmi.mdsd.Math with org.eclipse.xtext.common.Terminals

generate math "http://www.sdu.dk/mmmi/mdsd/Math"

MathExp:
	exps+=Exp*
;

Exp:
	'var' name=ID '=' exp=SumDiff
;

SumDiff returns Expression:
	ProdQuot (('+'{Add.left=current} | '-'{Sub.left=current}) right=ProdQuot)*
;

ProdQuot returns Expression:
	Primary (('*'{Mul.left=current} | '/'{Div.left=current}) right=Primary)*
;

Primary returns Expression:
	Constant | Parenthesis | VariableUse | VariableBinding
;

Parenthesis returns Expression:
	{Parenthesis} '(' exp=SumDiff ')'
;

Constant returns Expression:
	{Constant} value=INT
;

VariableUse returns Expression:
	{VariableUse} ref=ID
;

VariableBinding returns Expression:
	{VariableBinding} 'let' id=ID '=' binding=SumDiff 'in' body=SumDiff 'end'
;
```

Listing 2: XText syntax

2.2 XTend Generator

```
override void doGenerate(
    Resource resource, IFileSystemAccess2 fsa, IGeneratorContext context) {
    val variables = resource.allContents.filter(MathExp).next.compute

    // You can replace with hovering, see Bettini Chapter 8
    variables.displayPanel
}

def static Map<String, Integer> compute(MathExp math) {
    val variables = new HashMap<String, Integer>()

    val tmp = new HashMap<String, Expression>()
    math.exps.forEach[exp | tmp.put(exp.name, exp.exp)]

    math.exps.forEach[exp | {
        val res = exp.exp.computeExp(variables, tmp)
        variables.put(exp.name, res)
    }]
    return variables
}

def static int computeExp(
    Expression exp, Map<String, Integer> vars, Map<String, Expression> tmp) {
    switch exp {
        Add: exp.left.computeExp(vars, tmp)+exp.right.computeExp(vars, tmp)
        Sub: exp.left.computeExp(vars, tmp)-exp.right.computeExp(vars, tmp)
        Mul: exp.left.computeExp(vars, tmp)*exp.right.computeExp(vars, tmp)
        Div: exp.left.computeExp(vars, tmp)/exp.right.computeExp(vars, tmp)
        Constant: exp.value
        Parenthesis: exp.exp.computeExp(vars, tmp)
        VariableUse:
        {
            if (!vars.keySet.contains(exp.ref)) {
                val res = tmp.get(exp.ref).computeExp(vars, tmp)
                vars.put(exp.ref, res)
            }
            vars.get(exp.ref)
        }
        VariableBinding: exp.body.computeExp(
            vars.bind(exp.id, exp.binding.computeExp(vars, tmp)), tmp)
        default: throw new Error("Could␣not␣compute␣expression")
    }
}

def static Map<String, Integer> bind(
    Map<String, Integer> vars, String key, Integer value) {
    val binding = new HashMap<String, Integer>(vars)
    binding.put(key, value)
    binding
}
```

Listing 3: XTend generator

2.3 XTend Validator

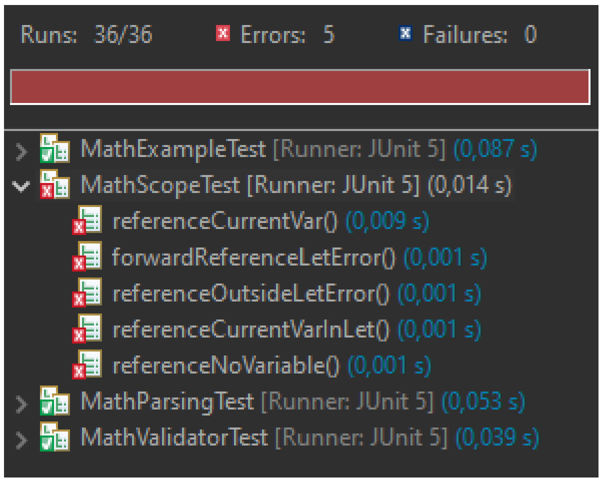
```
public static val DUBLICATE_VAR = 'dublicateVar'

val set = new HashSet<String>()

@Check
def clearSet(MathExp m) {
    set.clear
}

@Check
def checkNoDublicateVar(Exp exp) {
    if (set.contains(exp.name)) {
        warning("var␣" + exp.name + "␣has␣already␣been␣declared",
            MathPackage.Literals.EXP__NAME,
            DUBLICATE_VAR
        )
        return
    }
    set.add(exp.name)
}
```

Listing 4: XTend validator

3 Test

test results

Current implmentation passes **MathExampleTest**, **MathParsingTest**, and **MathValidatorTest**, but failes **MathScopeTest**.



```
Runs:  36/36        Errors:  5        Failures:  0

> MathExampleTest [Runner: JUnit 5] (0,087 s)
v MathScopeTest [Runner: JUnit 5] (0,014 s)
      referenceCurrentVar() (0,009 s)
      forwardReferenceLetError() (0,001 s)
      referenceOutsideLetError() (0,001 s)
      referenceCurrentVarInLet() (0,001 s)
      referenceNoVariable() (0,001 s)
> MathParsingTest [Runner: JUnit 5] (0,053 s)
> MathValidatorTest [Runner: JUnit 5] (0,039 s)
```

4 Conclusion

conclusion