# Assignment 3: External DSL code generation

## Description

> This assignment is about generating Java code, and integrating calls from the DSL to external Java code, that integrates into the generated Java code.
> The following is the summary of the tasks in the assignment:
>
> - Generate Java code that computes the value of the expressions
> - For each class, the method `compute` should fill the values of the variables defined with `var`.
> - The variables should be defined in the class as public instance variables.
> - The generated class should be in the package `math_expression`
> - When there are external functions defined:
>   - Create an interface inside the generated class
>   - A new method should be generated in the interface representing the external function for each external function.
>   - A constructor should be added that receives this generated interface.
> - For testing:
>   - Use `math` extension for your programs
>   - Implement the methods `pi()`, `sqrt(int n)` and `pow(n, m)` of the `ExternalImpl` class.
>   - Add the src-gen in the build path.

## Solution

The assignment have been solved by extending the code from assignment 2, and replacing the math generation code, with Java code generation.

The code generates a single file, for each `.math` file, named after the `program` name.

All unit tests provided for the assignment are passing.

In addition to the Java code generation, a new validator is implemented, that checks that the number of parameters for the external functions are correct, in accordance with the defined external method, defined when using the `external` keyword.

### Example

The provided test 34:

```
// Full example 3

program Test34
external pow(int,int)
external sqrt(int)
external pi()

var sideA = 3
var sideB = 4
```

```
var sideC =
    let powA = pow(sideA, 2) in
        let powB = pow(sideB, 2) in
            sqrt(powA + powB)
        end
    end

var perimeterTriangle = sideA + sideB + sideC

var radius = 5
var perimeterCircle =
    let diameter = 2 * radius in
        diameter * pi()
    end
```

Results in the following Java code:

```
package math_expression;
public class Test34 {
    public int sideA;
    public int sideB;
    public int sideC;
    public int perimeterTriangle;
    public int radius;
    public int perimeterCircle;

    private External external;

    public Test34(External external) {
        this.external = external;
    }

    public void compute() {
        sideA = 3;
        sideB = 4;
        sideC = this.external.sqrt((this.external.pow(sideA, 2) +
this.external.pow(sideB, 2)));
        perimeterTriangle = ((sideA + sideB) + sideC);
        radius = 5;
        perimeterCircle = ((2 * radius) * this.external.pi());
    }
    public interface External {
        public int pow(int n0, int n1);
        public int sqrt(int n0);
        public int pi();
    }
}
```

## External implementation

In addition to the DSL implementation, the assignment requires the implementation of the external functions `pi()`, `sqrt(int n)` and `pow(n, m)` in `ExternalImpl`.

This has been done, by using the `Java.lang.Math` class.

The implementation is as follows:

```
public int pi() {
    return (int) Math.PI;
}

public int sqrt(int n) {
    return (int) Math.sqrt(n);
}

public int pow(int n, int m) {
    return (int) Math.pow(n, m);
}
```

### Tests

The tests, including `ExternalImpl` and generated java files, are available at runtime-EclipseApplication/MathDemo.

## Source code

The source code for assignment 3 is available at github.com/The0mikkel/sdu-mbsd.
A pull request is available at github.com/The0mikkel/sdu-mbsd/pull/3, where changes from the Math (assignment 2) code can be seen.

Xtext file is available at `Math.xtext`.
Generator file is available at `generator/MathGenerator.xtend`.

*This document is available at github.com/The0mikkel/sdu-mbsd/assignment3/README.md.*