02332 Compiler Construction  
Assignment 2  
-  
s160107 - Alfred Röttger Rydahl  
s185097 - Rasmus Sander Larsen

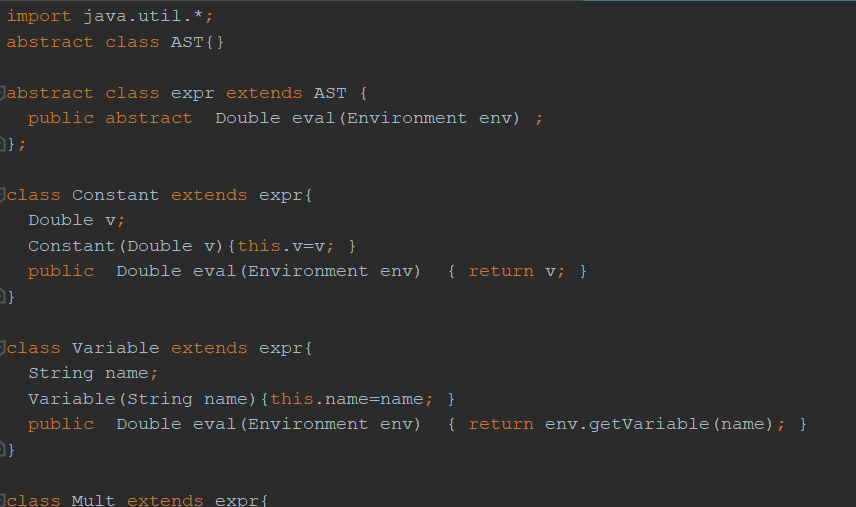
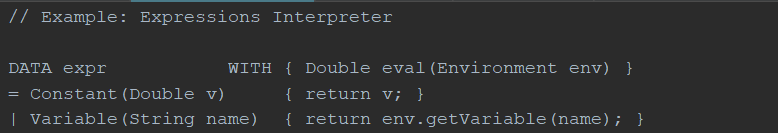
**Task 1**

The idea behind the coco is that it can generate its own code from an input and that the outputted code can be used for the compiling of an input matching the language syntax and schematics that the generated code is made from.

The AST.java file is our implementation of the needed methods and code for the generation of the needed output.  
We have added the small method “*String tab(int noOfTabs)”* to the *abstract class AST* which is used for the “cosmetics” of the outputted text so that the indentations fits the used standard.

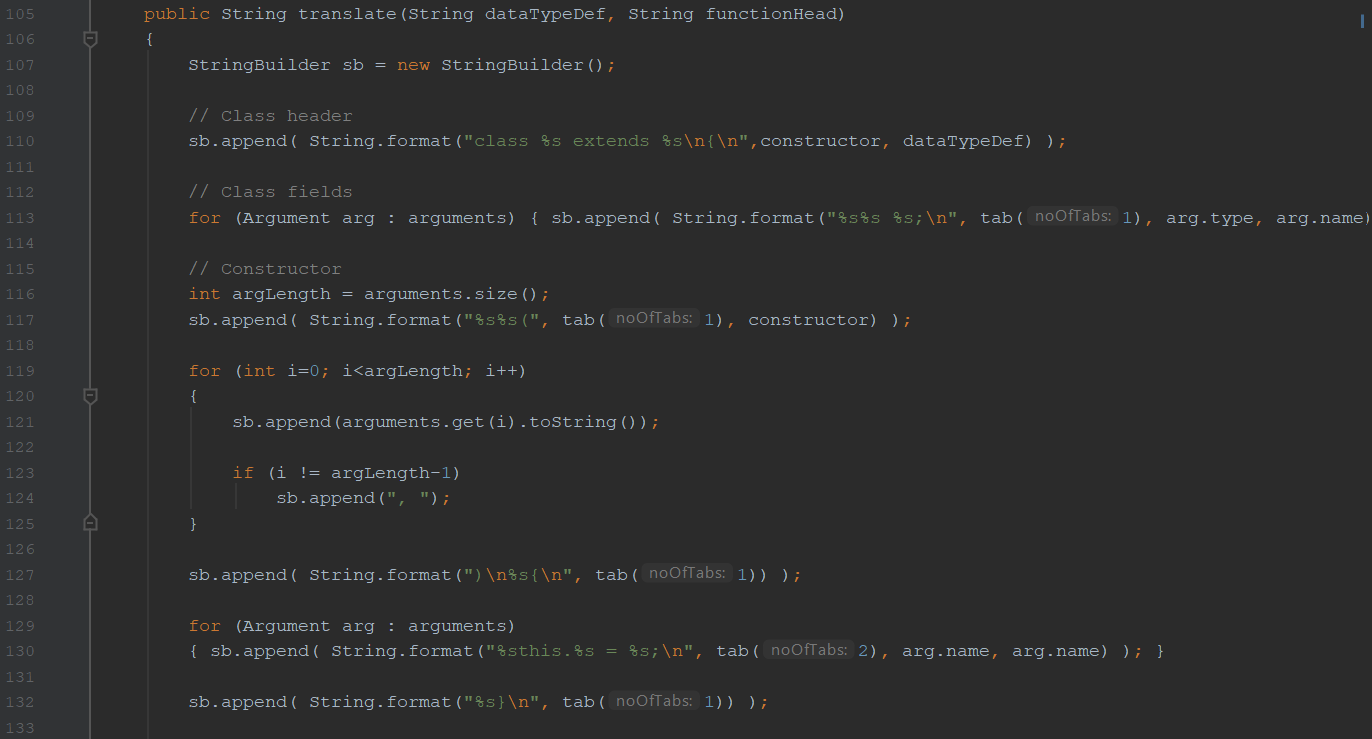
The following classes extends *the abstract class AST; Start, DataTypeDef, Alternative* and *Argument*. And each of them contains a set of variables matching the variables found in the coco.g4 file.

The code inside the methods *translate()* is what we created. The purpose of this method is the return a String that would correspond to the java code of a class instance matching the descripted class inside the coco\_input.txt file.



The picture on the left side a snippet of the content of the inputted file and the picture on the right side is a snippet of the file that is generate from the *translate()* methods.

Below is shown a snippet of the *translate()* method that generates the classes, variables and methods.



First of is the declarations of the class which is done in line 110 with the use of the information from the input file. Next of is the declarations of the class variables which is done in line 113 by calling inserting the variables for each of the “Argument” object, that the given class have, from the argument *type* and *name*. For the outputted *class Constant,* it would be the *Double v* and in *class Variable,* it would be the *String name,* that is shown on the above picture to the right*. Double* and *String* being the Argument *type* and *v* and *name* beings the Argument *name*.

Afterwards the constructor is created so that it takes all of the Arguments from the list as a parameter and sets it to the class variables via this.variableName = variableName. This is done from line 115 to line 132.  
This allows for the code to generate constructors with no parameters and x number of parameters.

Outside of the above code snippet that creation of the needed class methods is done. This is done in a similar fashion, by building the lines with information from the inputted files in a correct and flexible way, so a method can have no or x number of parameters.

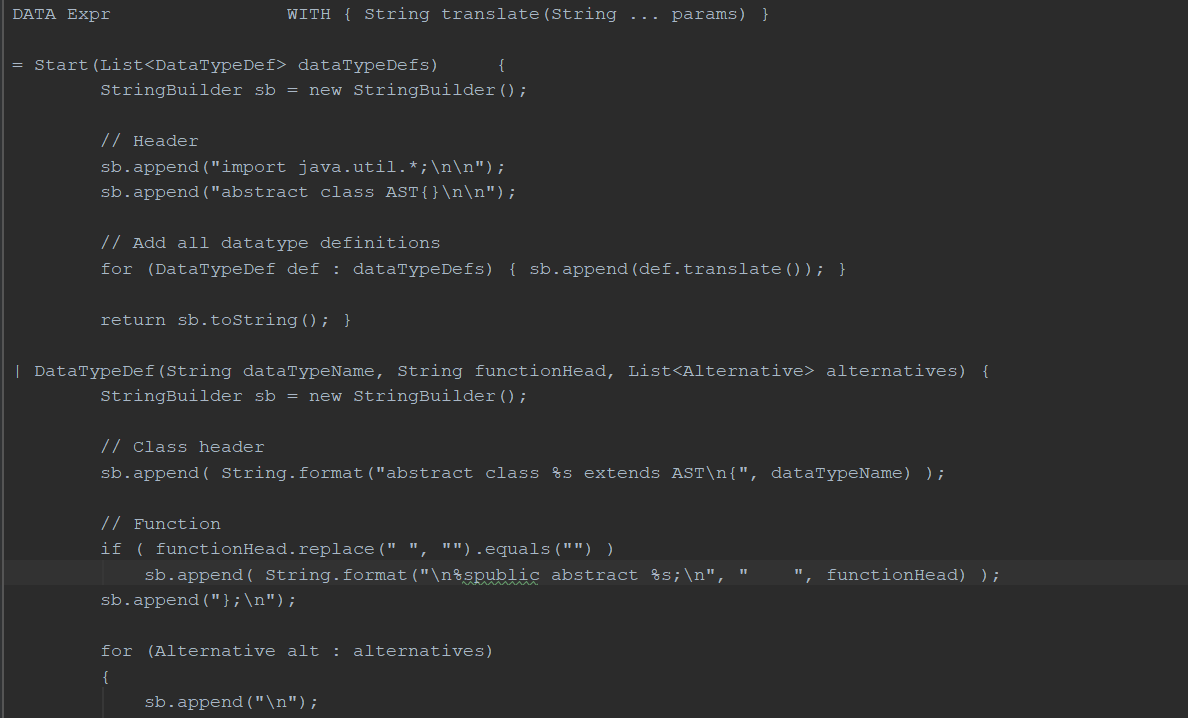
The *translate()* methods is implemented in all the necessary classes.

Then all implementation was done the *“make bigtest”* was called and it outputted the value 120 which is the correct result.

**Task 2**

Where our AST.java is used to generate a java file corresponding to the inputted functions and their returning values. The idea behind this task is to create a file named *coco.coco* which purpose is to generate the AST.java file that we just implemented above. So *coco.coco* will generate a java file, that is used to generate another java file which in the end can be used for the compiling.

The *coco.coco* is writing with the coco syntax and schematics.



This is a code snippet from the *coco.coco.*  
We start of by creating the *Expr class* that extends the *abstract class AST* and give it the method *translate()* that is used in all methods in the AST.java. Then the classes; *Start, DataTypeDef, Alternative* and *Argument* (as mentioned in task 1 above) is defined and their returns is defined. Each translate return is designed to insert the code needed for each class to return the correct result from its implementation of the *translate()* method.

This way the *coco.coco* file can generate a AST.java file (the AST file generated from our coco.coco is named AST\_generated.java). This AST.java can later on generate a java file with the correct classes, variabels and methods to match the inputted coco.g4 and input file.