You should use Meta-AspectJ to generate code as detailed below. Please attempt all of
the questions before the exercise class and bring your solutions along.

1. Write a Meta-AspectJ method `void genLogger(String cname, Stm logger)` that takes
   as input a class name and the syntactic representation of an arbitrary logging statement,
   and produces as output a new AspectJ aspect that counts all calls to methods in the
given class, and executes the given logging statement when each method is called.

2. Write a Meta-AspectJ program that accepts the string name of a class and generates a
   modified version of the class that has public setter/getter methods for all of its declared
   fields.
   [Hint: you should generate an aspect and use inter-type declarations. You may want to
   read about `unquote splicing` in the Meta-AspectJ documentation also.]

3. Use Meta-AspectJ to generate a Java class that, for a given input int \( N \), provides a
   number of overloaded methods named `pairsum`, that accept any number of ints up to
   \( N \), and returns the value given by adding all distinct pairs of multiplications of the
   inputted numbers. This can be calculated as

   \[
   a_1 \times (a_2 + a_3 + \ldots + a_{N-1} + a_N) \\
   + a_2 \times (a_3 + a_4 + \ldots + a_{N-1} + a_N) \\
   \ldots \\
   + a_{N-1} \times a_N
   \]

   This could be done in Java directly using `varargs` but at runtime that would involve
   autoboxing the parameters as an array. A generated solution provides a more direct
   implementation of the same.
   [Hint: think about how you might write this in Java using recursive methods first.]