Overview of evaluation

Evaluation occurs multiple times, until there are no changes anymore.

During each evaluation, unless a constant value has been established already, the *current value* of a variable starts with

- `NO_VALUE` for fields and local variables
- `VariableValue` for parameters

Local variables immediately graduate to real values, or a `VariableValue` if an initialiser is present. Given that the Java compiler will complain about missing initialisers, we do not need to worry about this.

Fields either:

- have an initialiser, and are final, in which case there is a concrete current value set as soon as the field has been analysed
- are assigned a value, exactly once, somewhere reachable from each of the constructors, which means they are effectively final
- are not effectively final, so there is nothing we can say about them; the value becomes `VariableValue`.

Until a field is declared effectively final, or not, its current value starts with `NO_VALUE`.

Method calls can potentially return constant values. Until it has been established that there is not a single return statement, `NO_VALUE` is returned.

Overview of analysis

The analyser produces and validates annotations. At the same time, it emits errors for some common programming errors.

In this section we first detail how each we compute each of the annotations.

@Constant for methods

At the end of the method analysis, we count the number of return statements in the method. If there is only one, and it has a definite value (i.e., `numberedStatement.returnValue` is different from `NO_VALUE`), we set `methodAnalysis.singleReturnValue` to this value. In the case of multiple return statements, we set an `Instance` value. The method analysis `check` method validates if the annotation corresponds to the value. Note that evaluation of the `MethodCall` and `MethodReference` expressions makes use of `methodAnalysis.singleReturnValue`.

Now we explain how we determine the errors:

Unused local variable

Finally, based on `methodAnalysis.unusedLocalVariables`, we emit errors during checking.

Unused assignment

It makes no sense to assign a value to a variable, and then assign another value before reading the former.