# Specification Language Syntax and Semantics

The purpose of this language is to specify API properties.

We define the language on top of Java as it is focused on Java API properties.

A property is specified as follows:

--- Syntax ---

(<@Before|@After>(<FunctionNames>)) +

void <PropertyName>(<ListOfArguments>) { <Body> }

The <@Before|@After> annotations indicate when an event handler will be triggered during program execution: at the entry of a method call (@Before) or at the exit of a method call (@After).

<FunctionNames> is a “.”-separated sequence of strings. Most often this string will be a fully-qualified name of a function. Alternatively, one can use the name of the class to denote a constructor or an ”\*” after a fully-qualified class name to denote all methods of a given class. For simplicity, we chose not to support arbitrary regexes to express patterns of events, but we do allow multiple annotations for the same handler function as long as @Before and @After annotations are not mixed.

<PropertyName> is the name of a function whose calls are to be monitored.

<List of args> is a comma-separated list of arguments. Cases:

* If <Function Name> denotes a selector of an instance method or class constructor, then it should include the target object and then the list of parameters.
* If <Function Name> denotes a selector of a class method then it should include the list of parameters of the corresponding class.
* If <Function Name> denotes a list of methods -- specified with a string of the form ``<class-selector>.\*'' -- then the list of arguments should be “String methodName, boolean isStatic, Object[] args”

<Body> is the body of the corresponding function which is responsible for checking and reporting whether a violation occurred or not. The body is a list of Java statements.

-- Semantics ---

A handler with the annotation @Before(fn) is always called at the entry of the functions denoted by fn. Likewise, a handler with the @After annotation is always executed at the exit of the referred functions.

The method call o.history() yields a list of strings corresponding to the names of methods called on object o since its creation. The order of the strings in the list represents the order in which the methods are called on the respective object.

A monitor can report property violations during execution with the call to the Log.violation()method, which outputs a string—the violation report—on the standard output.

To sum up, our specification language extends Java with two features: (1) the @Before and @After annotations, used in the definition of handler functions and (2) a special method, Object.history(), that enables the monitor to access the history of events on any given object.