

Contents

	Page
Editorial	3

COLUMNS

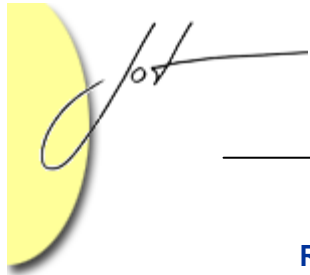
Why we should not add readonly to Java (yet) <i>By John Boyland</i>	5
---	---

In this paper, I examine some of reasons that “read-only” style qualifiers have been proposed for Java, and also the principles behind the rules for these new qualifiers. I find that there is a mismatch between some of the motivating problems and the pro-posed solutions. In particular, most have an overly restrictive “transitivity” rule, and all encourage “observational exposure” as a way to prevent representation exposure.

Thus I urge Java designers to proceed with caution when adopting a solution to these sets of problems.

A Static Analysis for Synthesizing Parametric Specifications of Dynamic Memory Consumption <i>By Victor Braberman, Diego Garbervetsky and Sergio Yovine</i>	31
---	----

This paper presents a static analysis for computing a parametric upper-bound of the amount of memory dynamically allocated by (Java-like) imperative object-oriented programs. It proposes a general procedure for synthesizing non-linear formulas which conservatively estimate the quantity of memory explicitly allocated by a method as a function of its parameters. The procedure has been implemented and evaluated on several benchmarks. Experimental results produced exact estimations for most test cases, and quite precise approximations for many of the others. The technique was used to compute usage in the context of scoped memory and discuss some open issues.

**Reasoning About Method Calls in Interface Specifications***By Adam Darvas and Peter Müller*

59

Interface specifications in languages such as Eiffel, the Java Modeling Language (JML), and Spec# are based on side-effect free expressions of the programming language.

In particular, such specifications contain calls to side-effect free methods to describe the program behavior without exposing implementation details. Reasoning about such specifications requires an encoding of programming language expressions in a program logic. This encoding is non-trivial for method calls.

This paper illustrates the subtle problems any encoding of method calls in specifications has to address. The paper gives sound encoding that allows side-effect free methods to create and initialize objects by explicitly modeling such modifications of the heap.

OUTLOOK

A brief outlook to the next issue87
