

Annotated Bibliography:
Integration of Java Generics Into The `jmlc`
Tool Within The Eclipse IDE

by
Adrian Kostrubiak

Submitted in partial fulfillment of Honors Requirements
For the Computer Science Major,
Dickinson College, 2008-09.

Professor Tim Wahls, Supervisor.

October 7, 2008.

1. B. Krause and T. Wahls. `jmlc`: A tool for executing JML specifications via constraint programming. In L. Brim, editor, Formal Methods for Industrial Critical Systems (FMICS '06), volume 4346 of Lecture Notes in Computer Science, pages 293 – 296. Springer-Verlag, August 2006.

The `jmlc` tool is a Java based tool created by Professor Tim Wahls and Ben Krause. This tool is an adaptation of the earlier `jmlc` tool (a JML tool to create code that performs runtime assertion checking) in order to compile JML specifications to constraint programs. These constraint programs are then executable via the Java Constraint Kit (JCK) (see annotation 4 for further information). This tool is a powerful one, alas lacks some implementation details. Amongst these details are the lack of Java 1.5 features, including, but not limited to generics and enumeration types. To this end, these will be the features that I will be adding in to this tool during the course of my honors project.

2. G. T. Leavens, K. R. M. Leino, E. Poll, C. Ruby, and B. Jacobs. JML: notations and tools supporting detailed design in Java. In OOPSLA 2000 Companion, Minneapolis, Minnesota, pages 105–106. ACM, Oct. 2000.

The Extended Static Checker (ESC) tool is used in one of the many tools that can be used along with the JML specification language. The ESC tool is used to automatically check for common errors in java code, such as null references, or array index out of bounds errors. This is all done statically, without actually running the program in question.

The notation of JML is one of its strong points: it is done in a very similar style to that of Java such that normal Java programmers can easily write these specifications. As well, JML allows for some more complex mathematical expressions.

3. L. Burdy, Y. Cheon, D. Cok, M. Ernst, J. Kiniry, G. T. Leavens, K. Rustan, M. Leino, and E. Poll. An overview of JML tools and applications. *International Journal on Software Tools for Technology Transfer*, 7(3):212-232, June 2005.

This paper is an overview paper on the entire JML technology. This goes over the JML notation and many of the commonly used and available tools that leverage the power behind JML. Among these tools mentioned in this paper are the `jmlc` JML compiler, `jmlunit`, a tool that combines JML with the unit-testing framework JUnit, the ESC/Java tool, which is used for static checking and verification, amongst other tools. As well, for documentation purposes, there is the `jmldoc` tool, a tool that extends to functionality of the normal `javadoc` tool in order to deal with JML specifications.

These are all tools with which it will be helpful to be familiar with during the course of this project, as the `jmlc` tool is built on these common JML tools.

4. S. Abdennadher, E. Krämer, M. Saft and M. Schmauss. JACK: A Java Constraint Kit. In *Electronic Notes For Theoretical Computer Science*, volume 64, pages 1-17. September 2002.

Constraint programming has come a long way in recent years. It is now at a point where many of the essential features are available as libraries, or these features are even embedded in some languages. The need for a constraint library in the Java language is underscored by the ever-growing popularity of the language. To this end, these researchers created the JAva Constraint Kit (JACK). Jack is made up of three main parts: the constraint handling rules, JCHR (Java Constraint Handling Rules), a tool to aid in the visualization of these rules, VisualCHR, and an abstract search engine to solve the constraint rules, JASE (Java Abstract Search Engine).

The JACK toolkit is an indispensable portion of the `jmlc` tool as the `jmlc` tool relies on this toolkit in order to not only turn formal JML specifications into programs, but to then have the ability to run these constraint programs.

5. N. Cantano, T. Wahls. Executing JML Specifications of Java Card Applications: A Case Study.

This article is about the use of the `jmlc` tool in order to execute JML specifications in a JavaCard application. The goals for this study were to determine the actual efficacy of the `jmlc` tool on a real moderately large and rather complex program. Overall, the project was a success, leading to changes both the implementation of the `jmlc` tool and the specifications in the electronic purse application. There were some issues encountered, one of these being the significant amount of time needed to diagnose problems. This was partly due to lacking error messages from the `jmlc` tool, wherein the only message given on failure was that the failure was in evaluating either the `requires`, `modifies`, or `ensures` clauses.

This article will be helpful in determining what changes can be made to the `jmlc` tool to make it a more powerful and fully useable tool, especially as it relates to actual use in industry and with larger and more complex applications.