Generics-related Refactorings in Eclipse

Adam Kieżun
MIT
Cambridge, MA, USA
akiezun@mit.edu

Robert Fuhrer, Frank Tip
IBM Research
Yorktown Heights, NY, USA
{rfuhrer,ftip}@us.ibm.com

Markus Keller
IBM Research
Zürich, Switzerland
markus_keller@ch.ibm.com

Categories and Subject Descriptors

D.1.5 [**Programming Techniques**]: Object-oriented Programming; D.2.2 [**Software Engineering**]: Design Tools and Techniques—*modules and interfaces*; D.3.3 [**Programming Languages**]: Language Constructs and Features—*data types and structures*

General Terms

languages, theory, experimentation

Keywords

refactoring, generics, generic types, Java, type constraints, type inference, parameterization, instantiation, Eclipse

ABSTRACT

We present refactorings that automate the process of migrating pre-generics Java programs to use generics. The task is divided in two parts: introduction of formal type parameters (*parameterization*) and inference of actual type parameters (*instantiation*). We developed efficient techniques and tools to assist developers in both parts. We will present them during the demonstration.

We present two generics-related refactorings for Java, one of which has been recently added to Eclipse 3.1 [2] and presented at OOPSLA'04 [1] and ECOOP'05 [3]. These refactorings facilitate migration of pre-generic libraries and applications to take advantage of the improved reuse and clarity provided by parametric polymorphism in OO programs.

The task of migrating pre-generic code to use generics can be understood as two related, yet distinct technical problems [1]:

- *Parameterization* — adding type parameters to existing classes so as to enhance reuse without the loss of type information.
- *Instantiation* — once classes have been parameterized, this analysis determines how occurrences of generic classes can be instantiated in client code.

We have addressed both problems by developing efficient algorithms embodied in practical tools. This demonstration presents refactorings that automate both tasks.

The automation of the refactorings requires advanced static analyses. Our approach uses the technique of polymorphic type inference based on type constraints [4, 3]. The technique is very scalable — it allows migration of large codebases to use generics in a matter of seconds or minutes.

Our demonstration includes the following refactorings:

Infer Type Arguments: addresses the instantiation problem. It helps in migrating Java application code to generics by determining what concrete types are used for each actual type parameter in the original program, modifying declarations and allocation sites where necessary, and removing casts that have been rendered redundant. This refactoring is based on the research prototype described in [3] and is included in the standard distribution of Eclipse 3.1.

Introduce Type Parameter: addresses the parameterization problem. It facilitates conveniently converting non-generic Java classes to generics, in a manner that is safe for existing clients. It adds a formal type parameter to a class to be used in place of the presently-declared type of a declaration, thus making the class generic. All other necessary changes are performed, which may include adding new type parameters to related classes. The refactoring also infers wildcards, which increases the flexibility of the parameterization.

1. REFERENCES

[1] A. Donovan, A. Kieżun, M. Tschantz, and M. Ernst. Converting Java programs to use generic libraries. In *Proc. of OOPSLA*, pages 15–34, Vancouver, BC, Canada, 2004.
[2] Eclipse. `http://www.eclipse.org`.
[3] R. Fuhrer, F. Tip, A. Kieżun, J. Dolby, and M. Keller. Efficiently refactoring Java applications to use generic libraries. In *Proc. of ECOOP*, Glasgow, Scotland, July 25–29, 2005.
[4] J. Palsberg and M. Schwartzbach. *Object-Oriented Type Systems.* John Wiley & Sons, 1993.