Family Polymorphism in JAVA

István Zólyomi and Zoltán Porkoláb

Family polymorphism – strongly investigated by Erik Ernst and others [1] – takes traditional polymorphism to the multi-object level. The object-oriented paradigm provides safe and flexible use of objects of classes arranged to inheritance hierarchies. Late binding ensures that we use the appropriate function body when we call a method on an actual object via polymorphic reference. In the same time we have compile-time guarantees to use only valid calls.

The problem arises when we use two or more independent hierarchies of classes together. In this case the collaborating ”families” may consist of similar but not interchangeable classes. Because there can be subtype relationship between classes in the different groups, it is not obvious to implement a constraint ensuring that only classes of the same family are used together. Traditional object-oriented languages are not able to handle this situation. Proposed solutions vary from run-time assertions to extensions of existing programming languages (like gbeta [2]).

In an earlier article [3] we discussed the solution in declarative way based on generative programming facilities of C++. C++ is rich in generative language tools, like templates. In Java however only version 1.5 introduces *generic* facilities and its expressive power is significantly differs from C++. In this article therefore we provide a different solution in Java for the family polymorphism problem.

References

[1] Ernst, E.: Family Polymorphism.
In Proceedings ECOOP 2001, LNCS 2072, pages 303–326, Budapest, Hungary 2001.

[2] gbeta homepage
http://www.daimi.au.dk/ eernst/gbeta

[3] Zolyomi, I., Porkoláb, Z.: A declarative approach to solve family polymorphism problem in C++
ICAI 6., Eger, Hungary 2004. Conference proceedings, under appearance

[4] Kim B. Bruce: Foundations of Object-Oriented Languages
2002, MIT Press