Generic instances in Java Byte Code
– Abstract –

Martin Plümicke

Baden-Wuerttemberg Cooperative State University Stuttgart
Department of Computer Science
Florianstraße 15, D–72160 Horb
pl@dhbw.de

April 13, 2014

Today it is not possible to use different instances of one generic Java type in the same class. This leds to a restriction in using the operation `instanceof` and generic exceptions. Additionally, generic overloading is not allowed [Ull12].
The reason for this restriction is given in the translation of generic instances to Java byte code. The compiler erases the parameters and translates all instances to the corresponding raw type. This means that all instances of one generic type coincides to one type in Java byte code.
The main reasons for this design decision are on the one hand the compatibility to older versions and on the other hand the reduction of the number of class file loadings during runtime.
We will show that there is a possibility to use generic instances in Java byte code without changing the JVM. For this it is necessary define a unique syntax to describe instatiated types in the JVM.
Unfortunately the problem of numerous loadings of class files could not be solved without changing the JVM, as the JVM has a restrictive type checker for the class files. We will show that the class files of different instances of one generic type differ only in the class name and the name of the direct superclass. The effective byte code is the same. This leds to our suppose, that it is possible to change the JVM, such that the class loader loads a class only if no other generic instance is already loaded.

References

[Ull12] Christian Ullenboom. *Java ist auch eine Insel*, volume 10. Galileo Computing, 2012. in german.