JvmInJava

- Java-in-Java VMs deal with low-level memory management through a small set of type safe extensions, supported by the compiler, which allow typed access to memory in a variety of ways. See JikesRVM docs for more information.
- Having written an extensive high performance memory management toolkit, which includes a wide range of GC algorithms, and which can outperform the standard glibc malloc(), I can assure you (as I and others have stated in previous posts), that a Java-in-Java VM is not encumbered by any such limitations.
- The Java-in-Java VM has some performance *advantages* through the lack of impedence mismatch between the supported language and the implementation language (this is one of the reasons for 2. above).
- There are a number of successful instances of this technology, including OVM, Jikes RVM, and Bartok (a C# in C# VM at MSR which can match the MS product VM on some benchmarks, despite having vastly less resoures applied to it).
- Having harmony Java code maximized will will also optimize the maintenability of the code. Mixed code (Java + native) is complex to debug and required multiple expert skills that are quite rare.
- Having Java-in-Java will also be profitable for the project management opportunities, as "Java interrested" developpers are also Java developpers or Java experts. But the change that they are good potential C or C++ resources are much less.
- Having Java-in-Java will help port of the VM on any new platform (from embeded to mainframe and portable devices)

See also The TechnicalFAQ