The Java Memory Model

This web page is a starting point for discussions of and information concerning the Java Memory Model (Chapter 17 of the Java Language Specification). The Java Memory Model defines how threads interact through memory. It used to be somewhat unclear and unnecessarily limiting, and so was revised. This is a reference page for that revision. The official site for JSR 133 - The Java(tm) Memory Model and Thread Specification Revision - is here.

This page is divded up into several sections:

- Primary reference material on the memory model.
- Pointers to the mailing list and archives.
- Additional material on the memory model, including information on double-checked locking.
- Older material on the memory model, now obsolete.
- Pointers to further reading material from other sources.

Reference Material

These reference materials are a good starting point for anyone trying to understand the memory model. Between them, they cover most of the major issues involved.

- **For First-Time Visitors**
  - A JSR-133 FAQ resource for programmers is available. This is a good place to start for those just becoming aware of the issues. (February 11, 2004)
- **For JVM and compiler implementors**
  - Doug Lea's JSR-133 cookbook, which is a guide for compiler writers who wish to implement the Java memory model.
  - Sarita Adve and Kourosh Gharachorloo wrote a tutorial on memory models in 1995 that remains an excellent reference and primer. Compaq Research Report 95/7, September 1995, 95.7 -- Shared Memory Consistency Models: A Tutorial.
- **For those wishing to understand the memory model in full**
  - A journal submission about the memory model that combines Jeremy Manson's dissertation, the POPL paper and the CSJP paper. For those interested in a thorough discussion of the memory model issues, this is the best bet. (October 7, 2005).
  - The JSR-133 specification, as sent to Final Approval Ballot. This is the "official specification" (August 9, 2004). It doesn't contain much in the way of explanation.
  - New presentation/description of the semantics of final fields. This is a brief description of the semantics of final fields. (May 12, 2004)

Mailing list

- To join the Java memory model mailing list, visit this page.
- To post to the list, email javamemorymodel-discussion@mimsy.cs.umd.edu (only subscribers may post to the list).
- The list has migrated. To access the archive of the old list, visit this page. To access the archive of the new list, visit this page.

Additional Information

Double-Checked Locking is Broken

Double-checked locking (also known as the multithreaded singleton pattern) is a widely employed idiom for publishing a singleton object to multiple threads.

- **The "Double-Checked Locking is Broken" Declaration**

  This document describes why the double-checked locking pattern is broken unless you use explicit memory barriers (or make assumptions about your processor and compiler).

- Descriptions of double-check idiom
  - Reality Check, Douglas C. Schmidt, C++ Report, SIGS, Vol. 8, No. 3, March 1996.
  - Double-Checked Locking: An Optimization Pattern for Efficiently Initializing and Accessing Thread-safe Objects, Douglas Schmidt and Tim Harrison. 3rd annual Pattern Languages of Program Design conference, 1996
  - Lazy instantiation, Philip Bishop and Nigel Warren, JavaWorld Magazine
  - Programming Java threads in the real world, Part 7, Allen Holub, Javaworld Magazine, April 1999.

Other information

- Causality test cases 1-20
- Informal notes we made about legal and illegal multithreaded patterns are available in this document .
- Volatile spec compliance tests
  - AtomicLong.java - Tests for atomic read/writes of volatile Longs
  - CoherenceVolatile.java - Tests for illegal compiler optimizations involving volatiles
  - ReadAfterWrite.java - Tests for sequential consistency of volatiles
  - Here is a tar file containing the tests and more detail about their output.
- Table of number of synchronization operations in Spec benchmarks

---

Older Material (for the memory model historians among you)

This is a list of many of the revisions that the memory model underwent over the course of its three years in flight. They are mostly out of date. If you are interested in the model as it stands, your best bet is the reference material above.

Draft Proposals for the Memory Model

- Earlier draft spec, incorporates some minor fixes from Draft Final Spec. (April 23, 2004)
- Proposed Final Draft for JSR-133 (April 12, 2004)
- Documents about the Unified Memory Model Proposal for Java, including additional test cases (March 16, 2004)
- Update of JSR-133 Public Review Document, includes clarifications and minor fixes, does not incorporate a new formalism. (March 16, 2004)
- Writeup of alternative formalism for the JSR-133 memory model (March 16, 2004)
- JSR-133 is in public review (March 16, 2004)
- Experimental version of MP without forbidden executions (February 24, 2004).
- JSR-133 Public Review Document (February 2, 2004).
- The competing models of February 6, 2004.
  - SC-
  - Manson/Pugh model:
    - Short Informal Description of the M/P model
    - Proof that M/P model has certain desirable properties
    - Long, formal description of the M/P model
- October 17, 2003
  - October 23th description of the Manson/Pugh core memory model (minor tweaks from previous version)
  - Proof that correctly synchronized programs have sequentially consistent semantics, and that standard reordering transformations are valid
  - JSR-133 Community Review Document
- August 29, 2003: NEW one page description of the Manson/Pugh core memory model (with a one and a half page appendix)
- August 8, 2003: JSR-133 Community Review Document
- August 4, 2003: Proof that reordering is legal under Manson/Pugh
- July 31, 2003: One page description of the Manson/Pugh core memory model (with a one page appendix)
- The full semantics of normal fields are in A New Approach to the Semantics of Multithreaded Java, by Jeremy Manson and William Pugh, Revised Jan 13, 2003.
- The full semantics of final fields are in Final Field Semantics, by Jeremy Manson and William Pugh, Revised April 7, 2003.
- Multithreaded semantics for Java, a previous version of the semantics. (2001)
- Weak Memory Orders and Object Oriented Programming, draft of the abstract for an OOPSLA poster session submission (PDF) (PS)
- The Java Memory Model is Broken by William Pugh, Journal version of the following paper; cleans up the paper somewhat and removes the naive fixes suggested in that paper.
- Fixing the Java Memory Model by William Pugh, 1999 ACM Java Grande

Talks

- Presentation given at Dagstuhl (October 24th, 2003)
- Multithreaded semantics for Java, (edited version of presentation given at MIT Sept 10th, 2000)
  Slides (PDF or PS) and handout (PDF or PS)
- JavaOne BOF on revising the Java Thread Spec, with Doug Lea (2000)
- JavaOne Talk, with Doug Lea (2000)
- OOPSLA 2000 workshop page

- Page for JavaOne 2000 BOF on revising the Java Thread Spec
- Slides from 1999 Java Grande Talk

Additional Background Reading

By Doug Lea

- The Java Memory Model, Section 2.2.7 of Concurrent Programming in Java, 2nd edition, Doug Lea, Addison Wesley, 1999
- Proposed revision to Section 17.4, Wait Sets and notification, Doug Lea

By Cenciarelli et al

- An Event-Based Structual Operational Semantics of Multi-Threaded Java, P. Cenciarelli, A. Knapp, B. Reus, M. Wirsing, In Jim Alves-Foss (Ed.) Formal Syntax and Semantics Of Java, LNCS 1523, pp. 157--200, Springer, Berlin, 1999.
- From Sequential To Multi-Threaded Java: An Event-Based Operational Semantics, P. Cenciarelli, A. Knapp, B. Reus, M. Wirsing, In Proc. 6^th Int. Conf. Algebraic Methodology and Software Technology, LNCS 1376, pp. 402--417. Springer Verlag. Berlin 1998.
- Verifying a Compiler Optimization for Multi-Threaded Java, B. Reus, A. Knapp, P. Cenciarelli, M. Wirsing, WADT Workshop 97, LNCS.

By Schuster et al.

- Java Memory Model: Precise Characterizations by Assaf Schuster, Workshop on Java for High-Performance Computing, June 1999, Rhodes.
- Java consistency: nonoperational characterizations for Java memory behavior by Alex Gontmakher and Assaf Schuster, ACM Transactions on Computer Systems Volume 18 , No. 4 (Nov. 2000) Pages 333 - 386.

On other memory models

- CAPSL Technical Memo 16: (148K gzipped Postscript), **"Location Consistency -- a new Memory Model and Cache Consistency Protocol,"** Guang R. Gao, Vivek Sarkar, February 16, 1998.
- TLA and TLA+, Lamport et al.

By Arvind et al.

- Improving the Java Memory Model Using CRF, Jan-Willem Maessen, Arvind, and Xiaowei Shen, OOPSLA 2000
- Commit-Reconcile and Fences (CRF): A New Memory Model for Architects and Compiler Writers, Xiaowei Shen, Arvind and Larry Rudolph, December 1998, To appear in proceedings of the 26th International Symposium on Computer Architecture, May 1999, Atlanta, Georgia., (14 pages).
- Improving the Java Memory Model Using CRF, Jan-Willem Maessen, Arvind, and Xiaowei Shen, OOPSLA 2000

By others

- Preliminary examination of the impact of "reads kill" by Dan Scales, Digital Western Research Laboratories
- Javasoft Bug # 4242244: JLS requires Coherence, Sun's JIT doesn't provide it.
- Investigating Java concurrency using Abstract State Machines, Yuri Gurevich, Wolfram Schulte and Charles Wallace

---

This page maintained by William Pugh. This material is based upon work supported by the National Science Foundation under Grant No. 0098162. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.