再谈volatile、synchronized

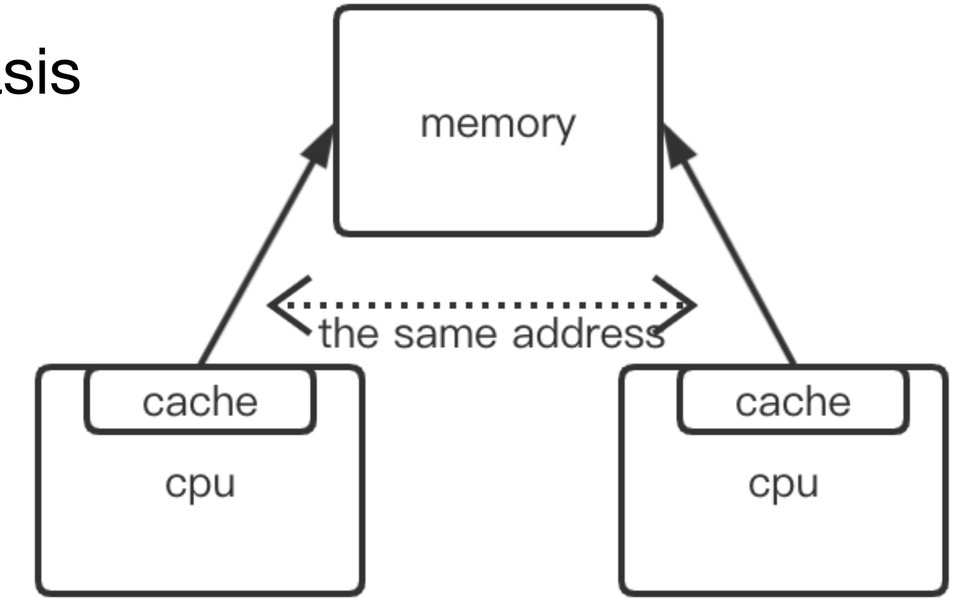
么广忠 20200608

volatile

- consistent value
  - Java Memory Model ensures that all threads see a **consistent value** for the variable
- synchronization order
  - A synchronization order is a **total order** over all of the synchronization actions of an execution. For each thread t, the synchronization order of the synchronization actions in t is **consistent** with the **program order** of t
  - A **write** to a volatile variable synchronizes-with all subsequent **reads** of v **by any thread** (where "subsequent" is defined according to the synchronization order).
- happens-before
  - Two actions can be ordered by a happens-before relationship. If one action happens-before another, then the first is **visible** to and **ordered before** the second
  - A **write** to a volatile field happens-before every subsequent **read** of that field.

cache coherency

- 缓存同一/一致性cache **coherency**
  - only needed for systems with caches
  - specified on a per-memory location basis



volatile & cache

- 线程可见性&缓存同一/一致性（cache **coherency**）
  - 正交（无关）
  - cache should be considered as transparent
  - 示例：单线程也存在问题

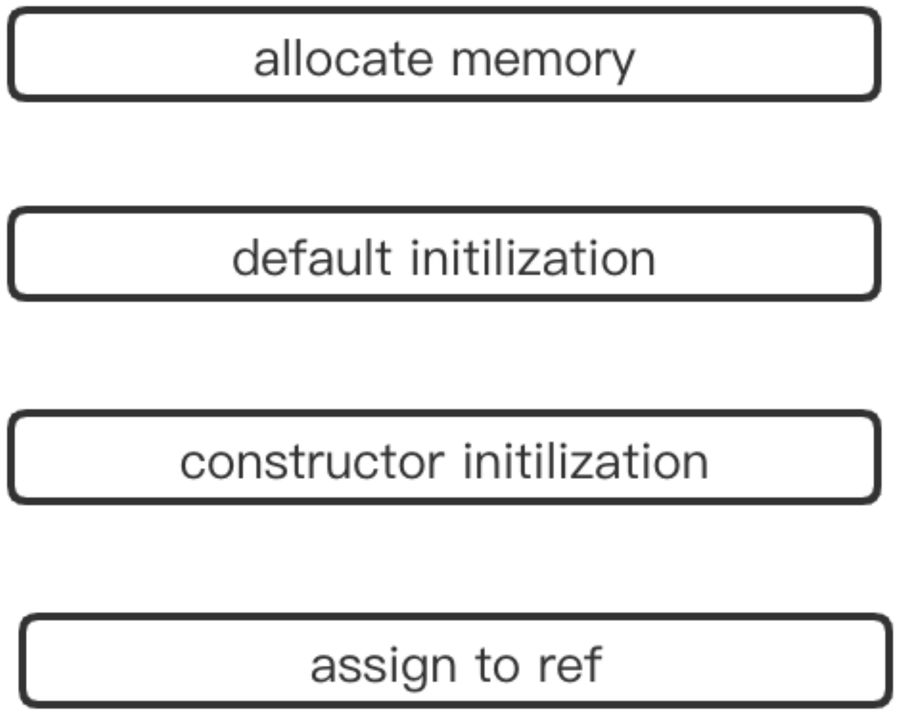
volatile note

- volatile用于可变对象或数组
  - 只能保证引用地址满足happens-before关系
  - 对象内的成员不保证happens-before关系
- 期望可变对象内成员更新满足顺序关系
  - 使用不可变对象
  - AtomicArray
  - 同步(如synchronized)

volatile应用

- DCL（double check lock singlelon）
  - volatile 防止指令重排导致引用未完全初始化的对象
  - 示例：复现困难
- Atomic*
  - volatile 修饰state

Creation of New Class Instances



synchronized for lock static/class

- answer
  - the thread acquires the intrinsic lock for the `Class` object associated with the class

Class

- Class
  - 记录类型信息,当classloader装载类时，由jvm自动创建Class类型的对象
  - class interface array primitive（boolean，int。。。）void
- 获取Class instance方式
  - ClassName.class
  - instance.getClass()
- 用处
  - 反射
  - class level fields（static field）
  - class level lock

class load(first use) process

initialization is thread-safe

class load process

12.2 Loading of Classes and Interfaces

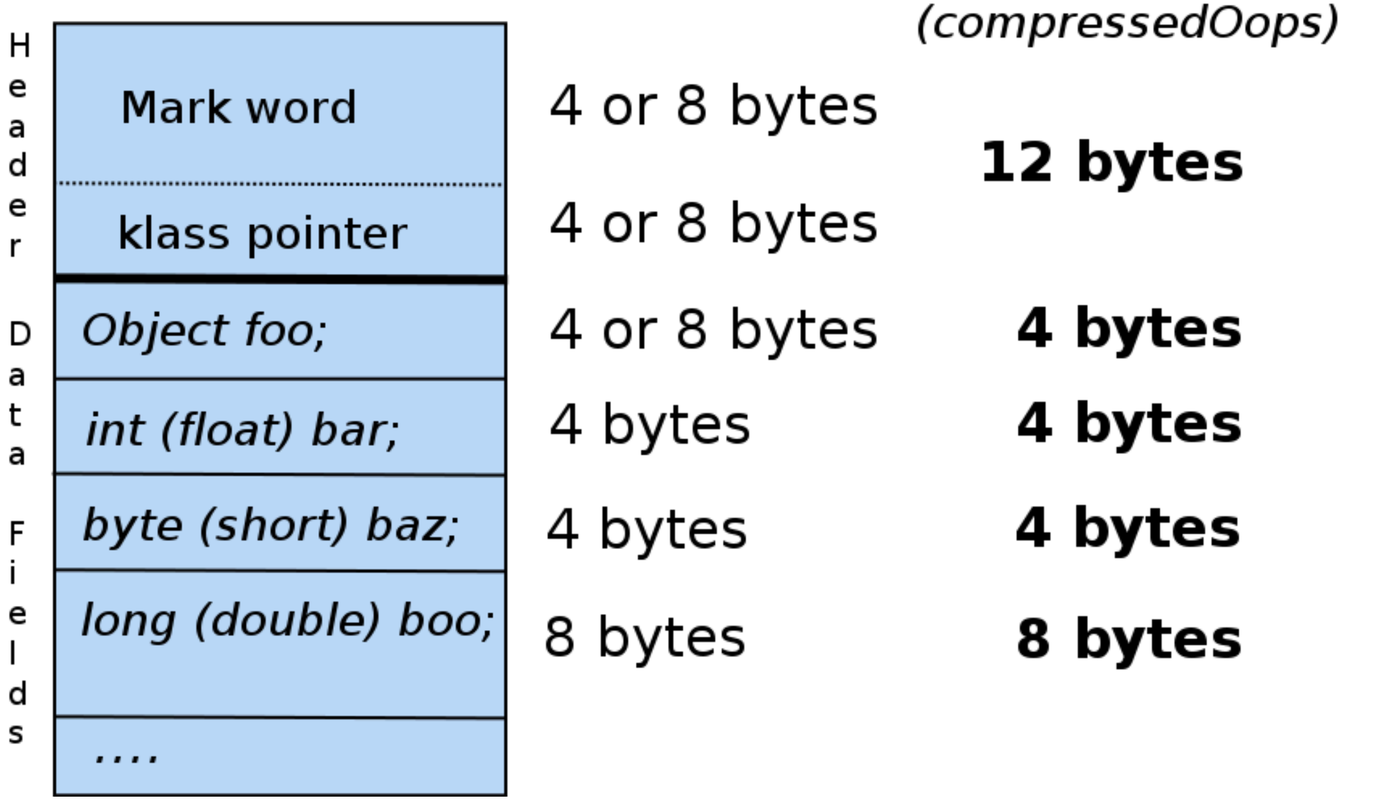
Loading refers to the process of finding the binary form of a class or interface type with a particular name, perhaps by computing it on the fly, but more typically by retrieving a binary representation previously computed from source code by a Java compiler, and constructing, from that binary form, a `Class` object to represent the class or interface.

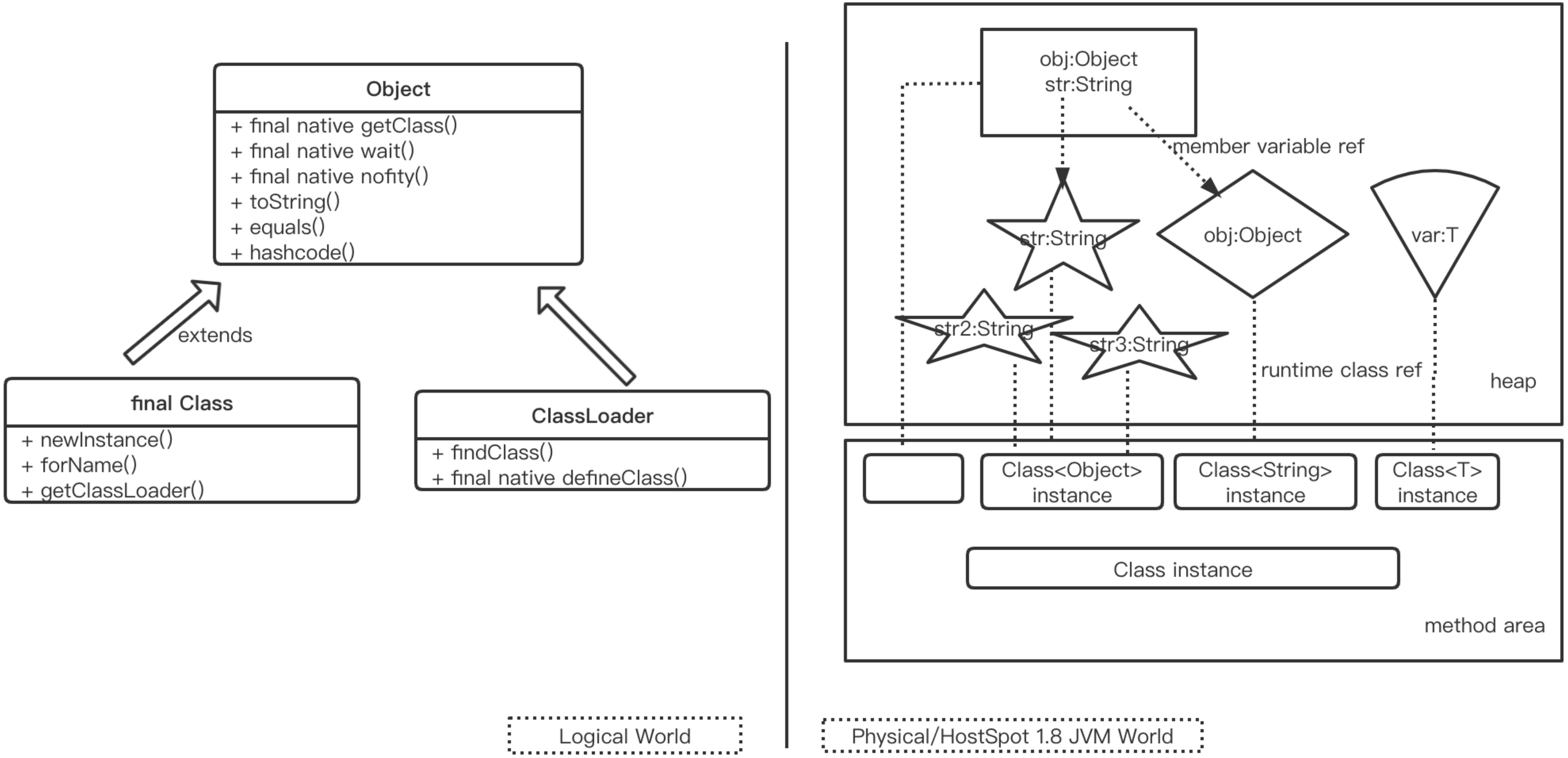
The precise semantics of loading are given in Chapter 5 of *The Java Virtual Machine Specification, Java SE 8 Edition*. Here we present an overview of the process from the viewpoint of the Java programming language.

The binary format of a class or interface is normally the `class` file format described in *The Java Virtual Machine Specification, Java SE 8 Edition* cited above, but other formats are possible, provided they meet the requirements specified in §13.1. The method `defineClass` of class `ClassLoader` may be used to construct `Class` objects from binary representations in the `class` file format.

Well-behaved class loaders maintain these properties:

Instance layout in hotspot




Object
+ final native getClass()
+ final native wait()
+ final native nofity()
+ toString()
+ equals()
+ hashcode()
extends
final Class
+ newInstance()
+ forName()
+ getClassLoader()
ClassLoader
+ findClass()
+ final native defineClass()
obj:Object
str:String
member variable ref
str:String
obj:Object
var:T
str2:String
str3:String
runtime class ref
heap
Class<Object>
instance
Class<String>
instance
Class<T>
instance
Class instance
method area
Logical World
Physical/HostSpot 1.8 JVM World

object design in java

- wait notify
  - thread communication
- object
  - lock intrinsic
- keyword
  - synchronized
- 优劣
  - 线程通信更简单
  - object空间占用多，头部markword

总结

- volatile
- synchronized

资料

- Volatile (http://gee.cs.oswego.edu/dl/cpj/jmm.html)
- Sonar rule related volatile（https://rules.sonarsource.com/java/tag/multi-threading/RSPEC-3077）
- Note for volatile（https://wiki.sei.cmu.edu/confluence/display/java/CON50-J.+Do+not+assume+that+declaring+a+reference+volatile+guarantees+safe+publication+of+the+members+of+the+referenced+object）
- memory consistency & cache coherency（https://os.inf.tu-dresden.de/Studium/DOS/SS2008/05_Coherency.pdf）
- class level lock（https://docs.oracle.com/javase/tutorial/essential/concurrency/locksync.html）
- Class object（https://docs.oracle.com/javase/tutorial/reflect/class/classNew.html）
- static fields location（https://stackoverflow.com/questions/2142192/java-where-do-static-fields-live-within-the-memory）
- java spec（https://docs.oracle.com/javase/specs/index.html）
- Hotspot terms (http://openjdk.java.net/groups/hotspot/docs/HotSpotGlossary.html)
- Hotspot storage management(http://openjdk.java.net/groups/hotspot/docs/StorageManagement.html)
- Hotspot java monitor（https://www.jianshu.com/p/e47ad923dee5）