If the object headers could talk

Rémi Forax

ToulouseJUG - May 2025

Rémi Forax ?

Associate Prof at Gustave Eiffel University

Expert for Java spec

- invokedynamic, lambda, module, record, sealed class, pattern-matching, etc

OpenSource developer

- ASM, github.com/forax

DUKE TURNS
30
foojay.io

Plan

- What is the header an object ?
- Header in Java 6 (2006)
- Header in Java 8 (2014)
- Header in Java 25 (2025)

What is the header of an object ?

Demo

https://github.com/forax/header-of-objects

Header of an object

2 values

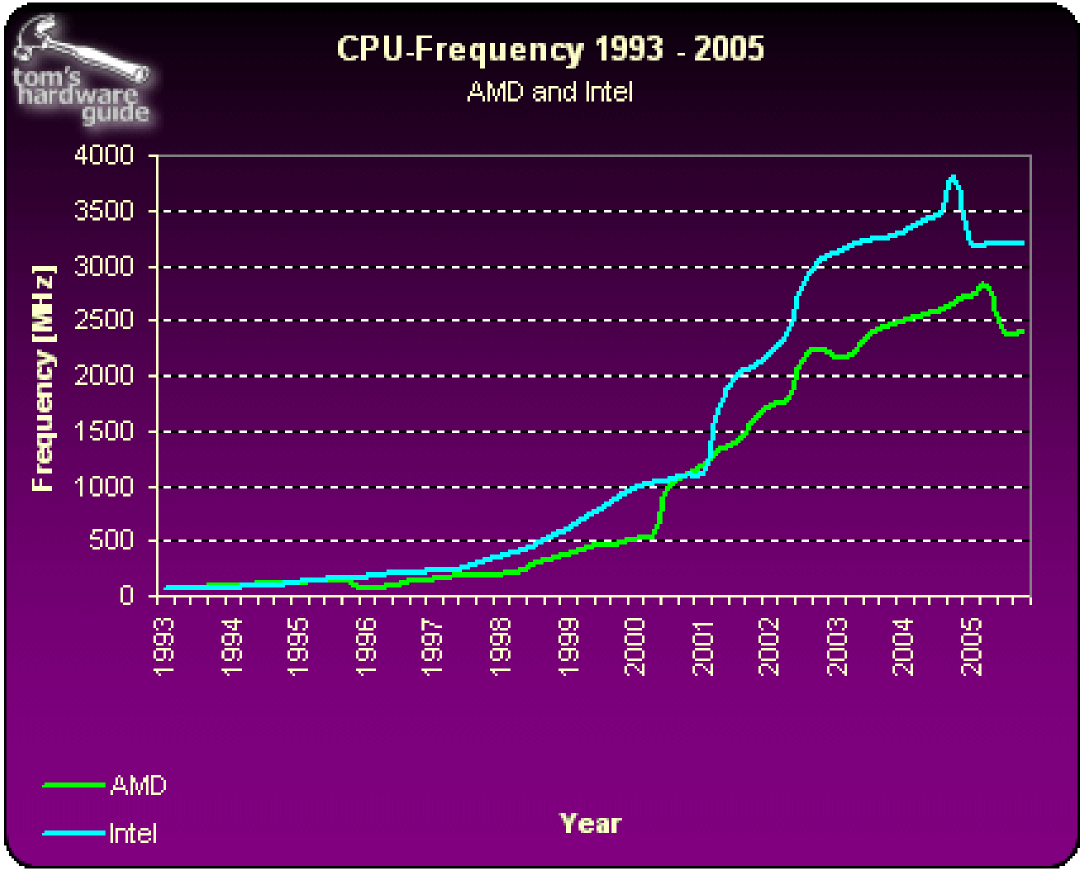
- Class word
  - Pointer to the Klass C++ object (!= java.lang.Class)
- Mark word
  - Identity hashCode
  - Locking bits (lock + bias lock)
  - GC marking bits
  - GC age

Header in 2006

Hardware in 2006

First dual core: Intel Pentium EE 800 series

End of free lunch



32 bits Header in 2006

- Class word
  - 32 bits pointer to PermGen
- Mark word
  - 32bits
    - HHHHHHHHHHHHHHHHHHHHHHHHHAAAABLL

hash 25 bits

X=unused H=hash A=GC age B=bias locked L=locked

Header in 2006

Class word

- 32/64 bits pointer to PermGen

Mark word

- 32bits

  HHHHHHHHHHHHHHHHHHHHHHHHHAAAABLL   hash 25 bits

- 64 bits

  XXXXXXXXXXXXXXXXXXXXXXXXXXHHHHHH   hash 31 bits
  HHHHHHHHHHHHHHHHHHHHHHHHHAAAABLL

X=unused H=hash A=GC age B=bias locked L=locked

Header Size in 2006

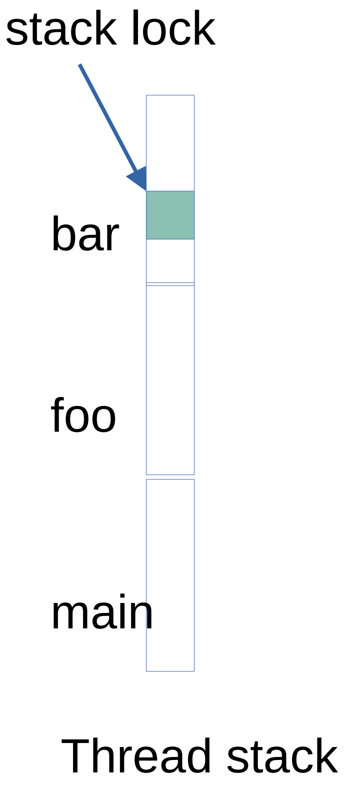
128 bits on 64 bits hardware

Median size of a Java objects

- between 256 to 512 bits *
- **~25 %** are the header

* see https://openjdk.org/jeps/450

How synchronized works ?



Java

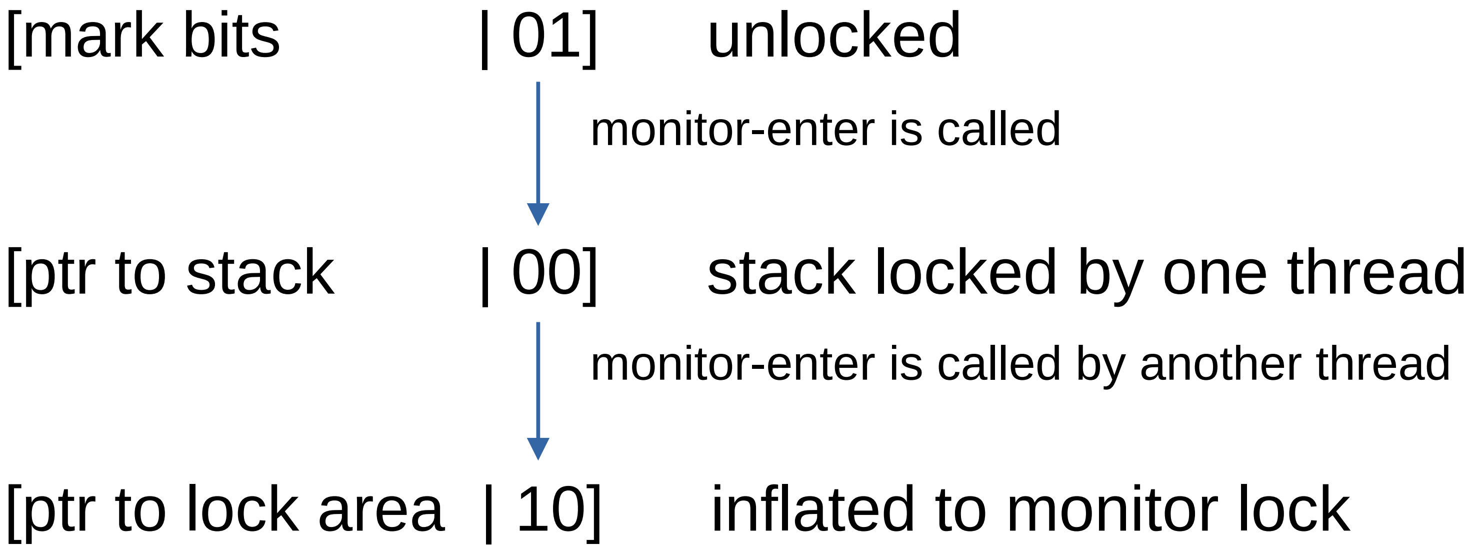
```
void bar() {
  synchronized(object) {
          …
  }
}
```

Bytecode

```
bar ()V
  aload 0
  getfield “object”
  monitorenter
     …
  monitorexit
```

Locking bits

Locking bits are used by synchronized



[mark bits | 01] unlocked

monitor-enter is called

[ptr to stack | 00] stack locked by one thread

monitor-enter is called by another thread

[ptr to lock area | 10] inflated to monitor lock

Monitor locks are freed during GC safepoint

https://github.com/openjdk/jdk/blob/8153779ad32d1e8ddd37ced826c76c7aafc61894/hotspot/src/share/vm/oops/markOop.hpp

Header displacement

```
struct BasicLock {
  volatile markOop _mark;              // displaced object header word - mark
}

struct ObjectMonitor {
  volatile markOop _mark;              // displaced object header word - mark

  void* volatile _object;              // backward object pointer - strong root

  void * volatile _owner;              // pointer to owning thread

  volatile intptr_t  _recursions;      // recursion count, 0 for first entry

  …  // + wait list + statistics
}
```

https://github.com/openjdk/jdk/blob/8153779ad32d1e8ddd37ced826c76c7aafc61894/hotspot/src/share/vm/runtime/objectMonitor.hpp

Locking bits in 2006

Locking bits are also used by CMS GC

[mark bits | 01] unlocked

[ptr to stack | 00] locked by one thread

[ptr to lock area | 10] inflated monitor lock

[ptr to heap | 11] forwarded for CMS GC

Biased Locking

Multicores CAS cost a lot => No CAS if biased

[0 | epoch | AAAA | 1 | 01] unbias (biased enable)

[Thread* | epoch | AAAA | 1 | 01] unlocked (biased enable)

- If the interpreter has seen only one thread
- If identity hashCode was never computed
- revoking bias locks are done in bulk at GC safepoint

https://github.com/openjdk/jdk/blob/8153779ad32d1e8ddd37ced826c76c7aafc61894/hotspot/src/share/vm/runtime/biasedLocking.hpp#L25

Header in 2014

Java & Death of Sun Microsystem

History (as I recall)

- 2001 Dotcoms burst but Sun has a huge war chest
- 2007 Java 6 then Java is OpenSourced
- April 20, 2009 Oracle acquire Sun
- 2011 Java 7
- 2014 Java 8 (lambda)

Hardware in 2014

Haswell 4xxx Core i7, 4 cores, 4Gz, 22 mn

with ultrabook version (15 W)

Compress oops

Store/read reference on 32 bits

- Address of objects are aligned on 64 bits
  - => address ends with xxxxx000
- Store
  - compressed = (oops – base) >>> 3
- Load
  - oops = base + compressed << 3

https://wiki.openjdk.org/display/HotSpot/CompressedOops

Compress oops (2)

Store/read reference on 32 bits

Encode/Decode 32 bits <-> 64 bits

Hardware support (Intel lea) or those ops are cheap

Constraints

- Heap size <= 32 G
- “base” must be stored in a register

Header in 2014

Class word (compress class pointer)

- 32 bits pointer to Metaspace

Mark word

- 64 bits

```
XXXXXXXXXXXXXXXXXXXXXXXXXXHHHHHH    hash 31 bits
HHHHHHHHHHHHHHHHHHHHHHHHHAAAABLL
```

X=unused H=hash A=GC age B=bias locked L=locked

https://github.com/openjdk/jdk/blob/jdk8-b40/hotspot/src/share/vm/oops/markOop.hpp

Header Size in 2014

96 bits on 64 bits hardware

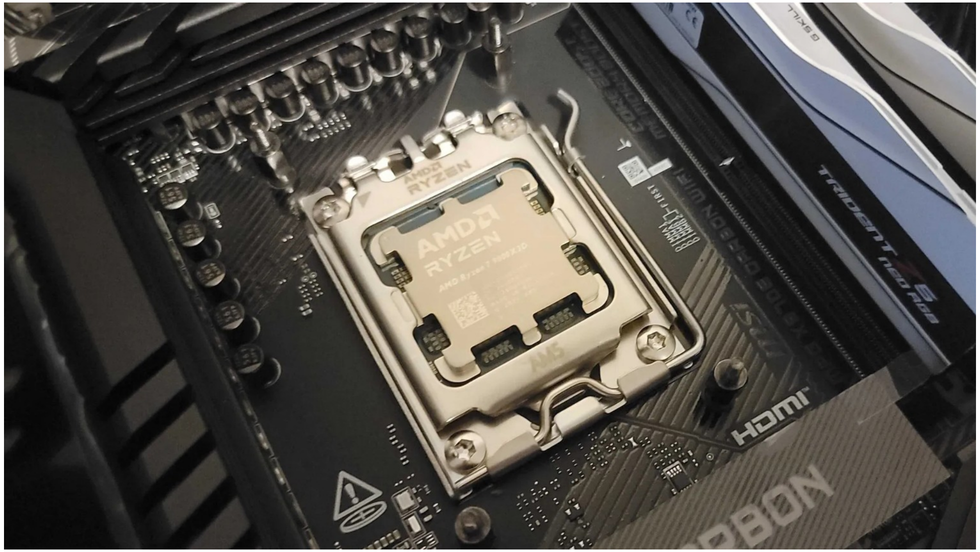
Median size of a Java objects

- between 256 to 512 bits
- **~18 %** are the header

Header in 2025

Hardware in 2025

AMD Ryzen 90XX, 12 cores, 5.6 Gz, 7nm



Java

Recent versions

- 2017 Java 11 (ZGC)
- 2020 Java 17 (record)
- 2023 Java 21 (virtual threads + pattern matching + gen ZGC)
- 2025 Java 25 (compact header + gen shenandoah)

ZGC / Shenandoah

Low latency GCs

- Pause < 1ms
- Marking and evacuation while program is running
- Requires more memory

minimal pause during GC safepoint

- Freeing of ObjectMonitor ?
- Revocation of bias locked object ?

ZGC / Shenandoah

Low latency GCs

- Pause < 1ms
- Marking and evacuation while program is running
- Requires more memory

minimal pause during GC safepoint

- Freeing of ObjectMonitor ?
  - Async monitor deflation
    https://wiki.openjdk.org/display/HotSpot/Async+Monitor+Deflation)
- Revocation of bias locked object ?
  - Remove biased locking
    - https://openjdk.org/jeps/374

Loom - Virtual Threads

Several virtual threads on one OS threads

- For IO calls
  - copy the stack to the heap
  - Schedule another virtual thread (copied back from the heap)

Can we always copy the stack ?

Loom - Virtual Threads

Several virtual threads on one OS threads

- For IO calls
  - copy the stack to the heap
  - Schedule another virtual thread (copied back from the heap)

Synchronized BasicLock are on stack !

- synchronized prevents de-scheduling of virtual threads

  => virtual thread pinning

Lilliput – Compact Header

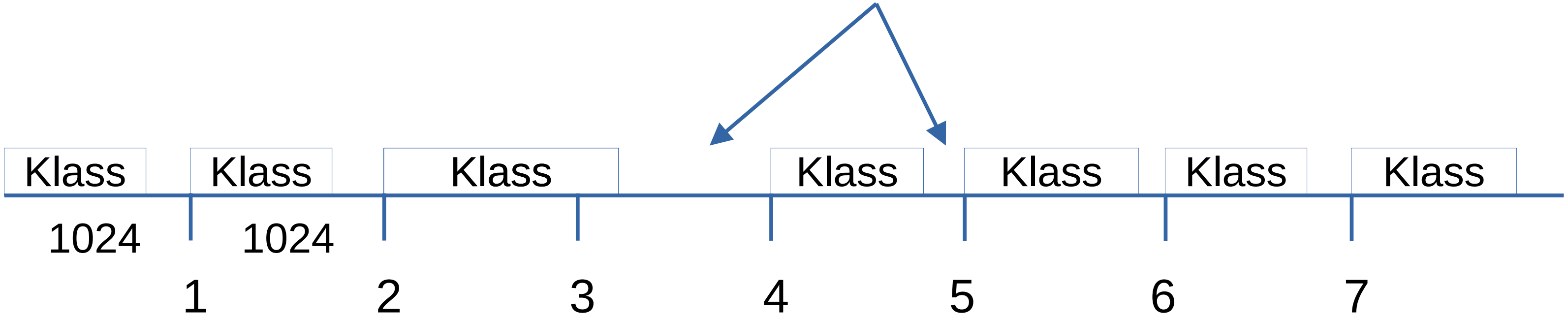
Reduce the header to 64 bits

- Merge the class word and the mark word
- Classes uses only 22 bits (~4 000 000 classes)
  - Use Tiny Klass

But …

Lilliput – Tiny Klass

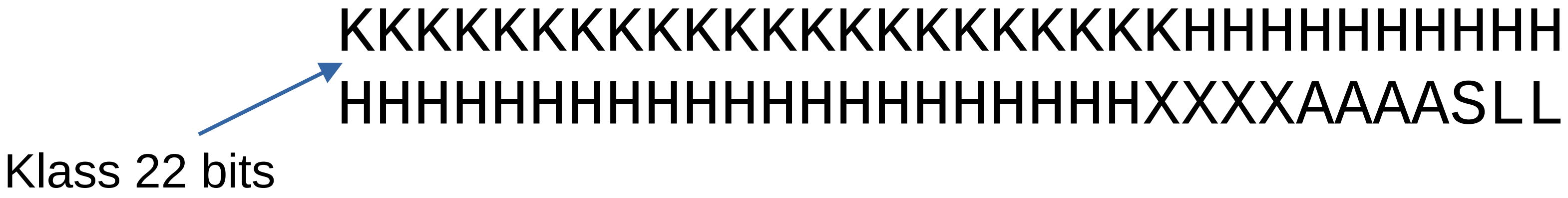
- Increase alignment to 1024 bytes
  - Most struct Klass takes between 512 and 1K
  - If size bigger than 1K, an index is lost
    - Put class metadata in the holes (oopmap, stubs, ...)



Header in 2025

Mark word (class word + mark word)

64 bits



KKKKKKKKKKKKKKKKKKKKKKHHHHHHHHHH
HHHHHHHHHHHHHHHHHHHHHXXXXAAAASLL

Klass 22 bits

K=klass X=unused H=hash A=GC age S=self forward L=locked

https://github.com/openjdk/jdk/blob/07c9f7138affdf0d42ecdc30adcb854515569985/src/hotspot/share/oops/markWord.hpp

Header Size in 2025

64 bits on 64 bits hardware

Median size of a Java objects

- between 256 to 512 bits *
- **~12 %** are the header

Lilliput – Change locking algo

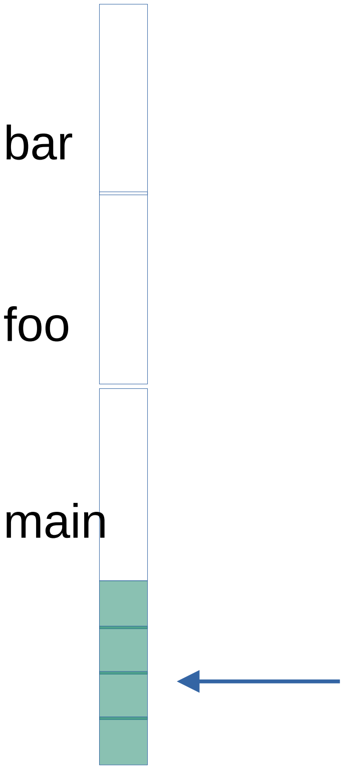
Reduce the header to 64 bits

- Merge the class word and the mark word
- Classes uses only 22 bits (~4 000 000 classes)
  - Use Tiny Klass

But problems if overridden by pointers (stack, monitor, GC)

- New locking algorithm (fast locking + MonitorLockTable)
- Change GCs to use a forward table

New fast locking



lock stack at the bottom of the stack

Thread stack

Java

```
void bar() {
  synchronized(object) {
          …
  }
}
```

Bytecode

```
bar ()V
  aload 0
  getfield “object”
  monitorenter
     …
  monitorexit
```

Locking bits in 2025

Locking bits (fast locking + ObjectMonitorTable)

[mark bits | 01] unlocked

[mark bits | 00] locked by one thread

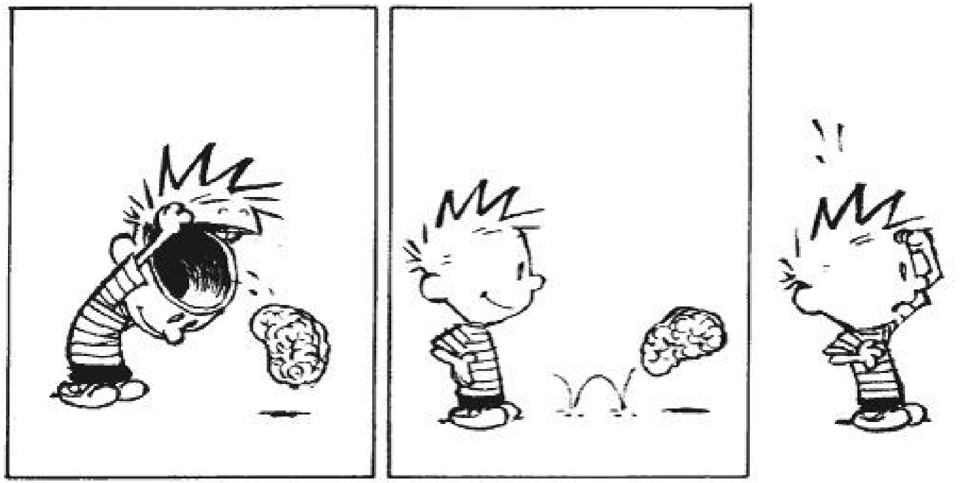
- Each thread as a stack of 8 BasicLock at the bottom

[mark bits | 10] inflated monitor lock

- Concurrent HashMap Object -> ObjectMonitor

No displacement headers anymore

Header in the Future ?



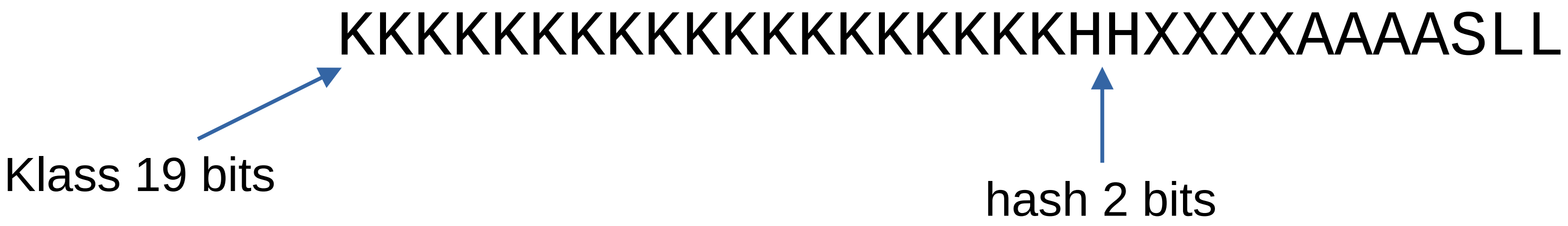
CALVIN & HOBBES © BIL WATTERSON

Don’t believe what I’m saying !

Header – Lilliput 2

Mark word

32 bits



KKKKKKKKKKKKKKKKKKKHHXXXXAAAASLL

Klass 19 bits

hash 2 bits

K=klass X=unused H=hash A=GC age S=self forward L=locked

https://github.com/openjdk/lilliput/blob/lilliput-2/src/hotspot/share/oops/markWord.hpp

Compact Identity HashCode

Allocate a field for the identity hashCode on demand

- Use the real address when first asked
- When the GC move the object, create a field

https://wiki.openjdk.org/display/lilliput/Compact+Identity+Hashcode

Lilliput 2 – Tiny Klass v2

Encode Klass with no instance (interface, abstract class, etc) in another space

Near/Far Klass, if more than ~500 000 then the class is encoded into the first field

Header Size in the Future

32 bits on 64 bits hardware

Median size of a Java objects

- between 256 to 512 bits *
- **~6 %** are the header

Roman Kennke

Questions ?