Neil Masson – Java L3 Service Core Teamt
24th September 2013

New JVM Features – Multitenancy and Packed Objects

Important Disclaimers

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY.

WHILST EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED.

ALL PERFORMANCE DATA INCLUDED IN THIS PRESENTATION HAVE BEEN GATHERED IN A CONTROLLED ENVIRONMENT. YOUR OWN TEST RESULTS MAY VARY BASED ON HARDWARE, SOFTWARE OR INFRASTRUCTURE DIFFERENCES.

ALL DATA INCLUDED IN THIS PRESENTATION ARE MEANT TO BE USED ONLY AS A GUIDE.

IN ADDITION, THE INFORMATION CONTAINED IN THIS PRESENTATION IS BASED ON IBM’S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM, WITHOUT NOTICE.

IBM AND ITS AFFILIATED COMPANIES SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION.

NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, OR SHALL HAVE THE EFFECT OF:

- CREATING ANY WARRANT OR REPRESENTATION FROM IBM, ITS AFFILIATED COMPANIES OR ITS OR THEIR SUPPLIERS AND/OR LICENSORS

Introduction to the speaker

- Neil Masson

- Many (many) years experience in Java
- IBM Service – Core L3 Team

What should you get from this talk?

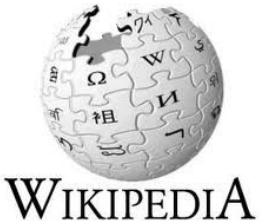
- JVM proving to be a fertile ecosystem for languages
- Plenty of opportunity to innovate in other spaces
- Runtime is the gateway to this innovation
- Largely ignored the last few years, but this is where the core inventions can occur

The runtime **isn’t** boring!

Multitenancy

Just what do you mean by “multitenancy”?

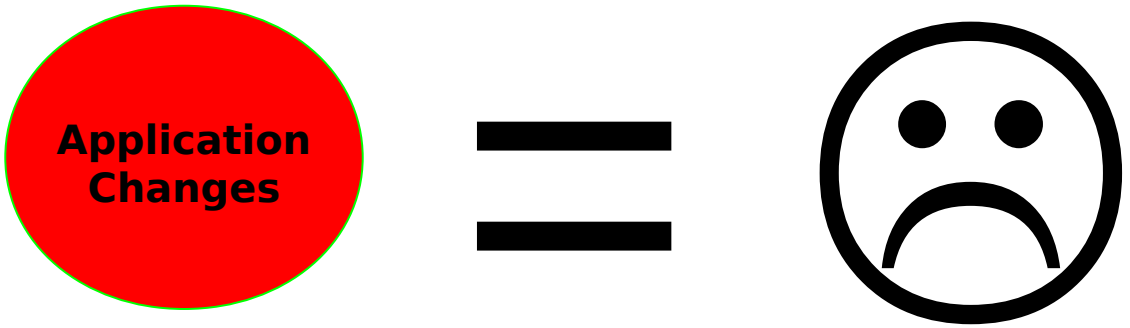
With a multitenant architecture, a software application is designed to virtually partition its data and configuration, and each client organization works with a customized virtual application instance.



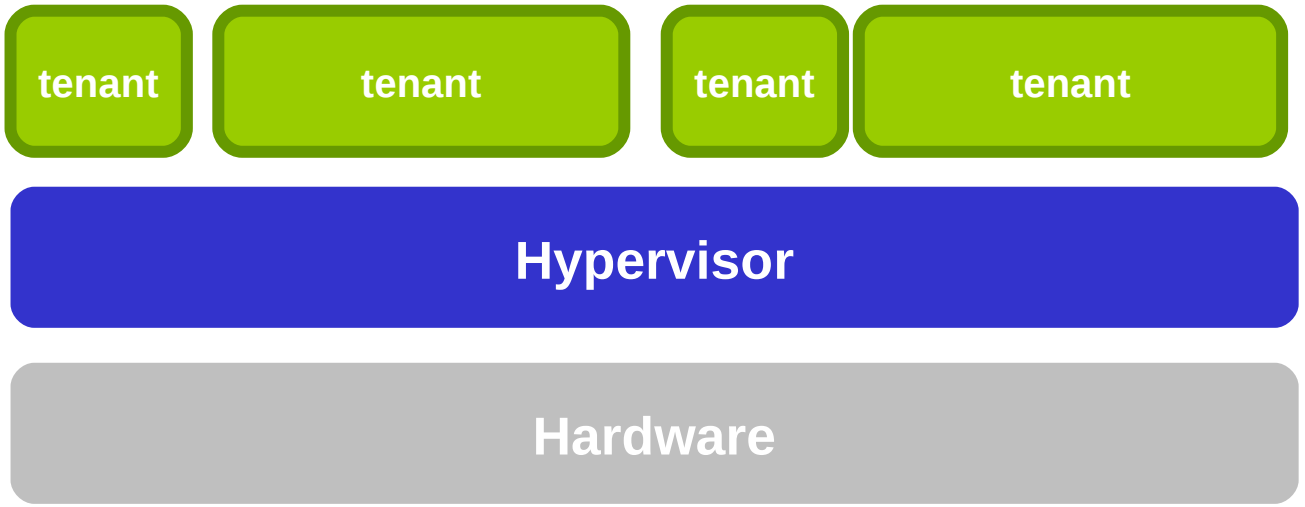
- **Working Definition**
  - A single instance of a software application that serves multiple customers
    → Each customer is a tenant.
  - Tenants can customize some parts of the application (look and feel) but not the code.
  - Infrastructure usually opaque
    → opportunity for provider

Why? Cost Savings: As compared to single-tenant deployment model

JDK Support for Spectrum of Sharing / Multitenancy (Level 1-5)

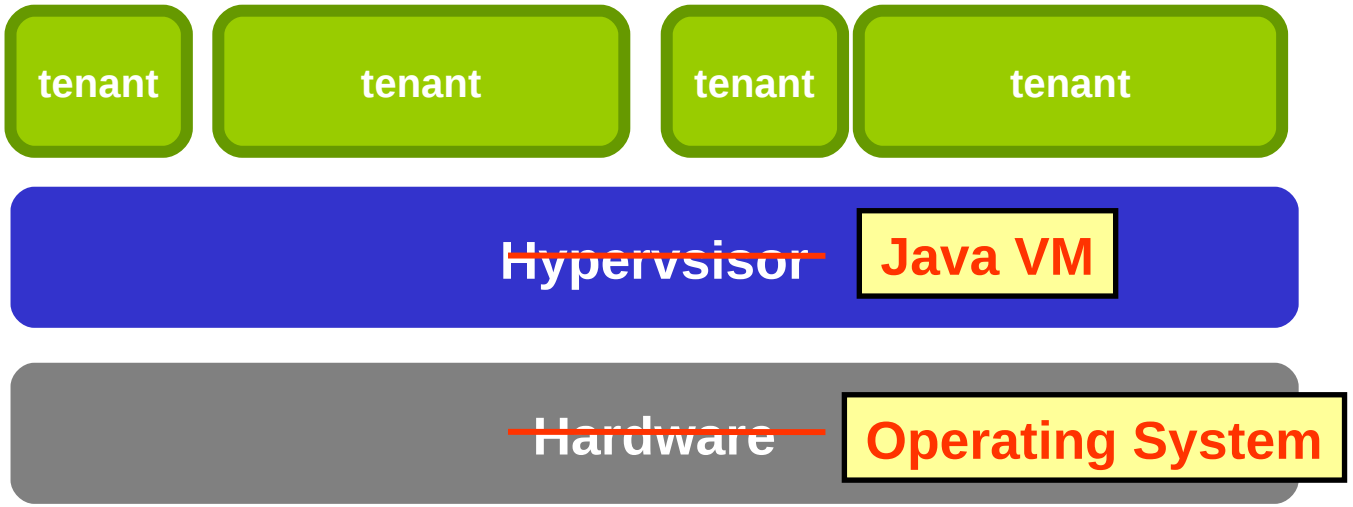


Hardware Virtualization



- Hypervisors run multiple applications side-by-side safely
- Advantages
  - Capture idle CPU cycles
  - Automatic de-duplication (RAM)
  - Ability to meter and shift resource toward demand
  - No need to change tenant applications

Hardware Virtualization

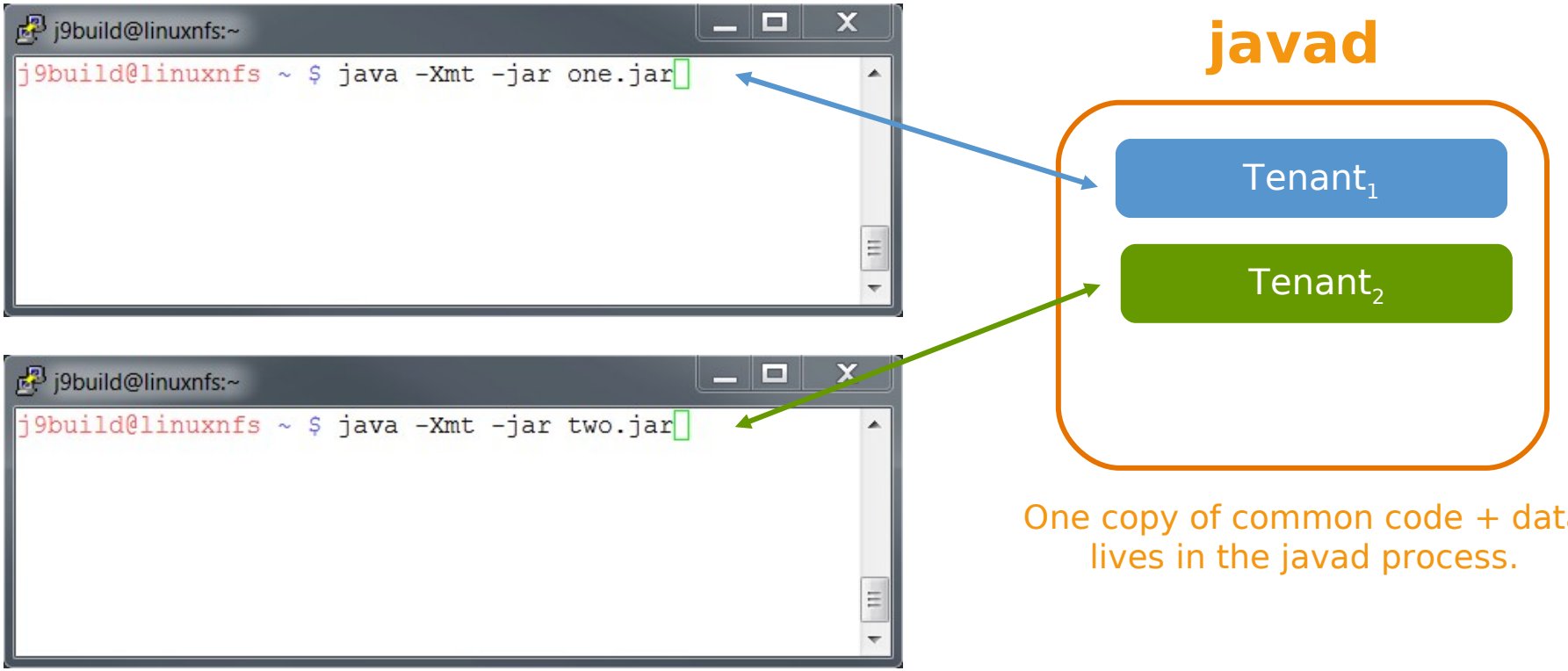


- ~~Hypervisors~~ JVMs can run multiple applications side-by-side safely
- Advantages
  - Capture idle CPU cycles
  - Automatic de-duplication (ability to share Java artifacts)
  - Ability to meter and shift resource toward demand
  - No need to change tenant applications

Multitenancy: Low (or no) barrier to entry

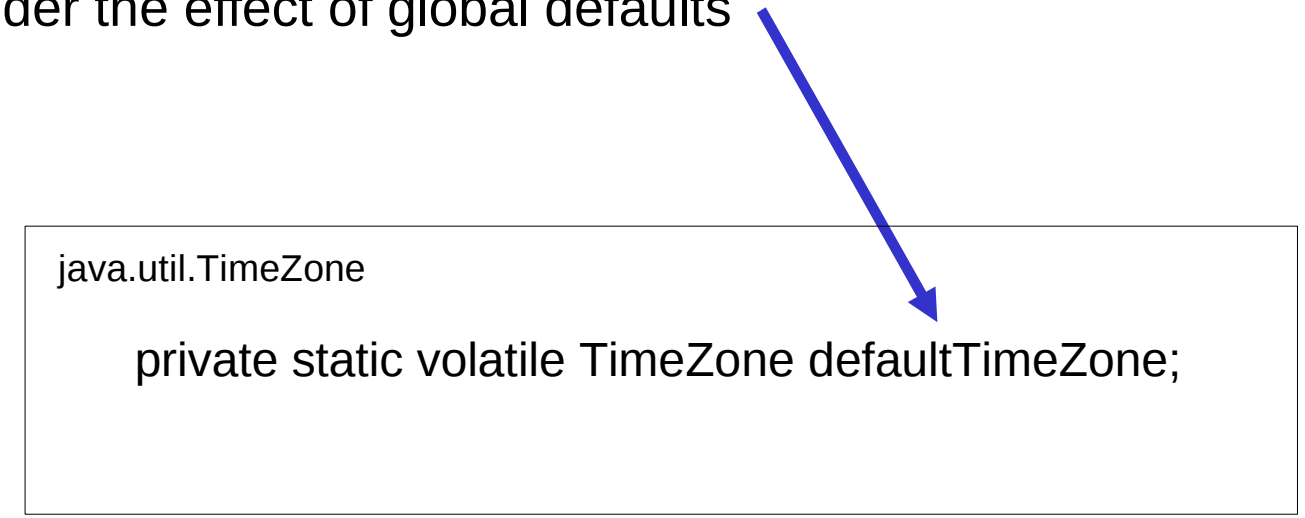
- Multitenancy is all about reducing duplication by transparently sharing a JVM
  - 1 GC, 1 JIT, shared heap objects
  - plus: JVM-enforced resource constraints to meter and limit consumption

- Ergonomics: Opt-in to multitenancy with a single flag: -Xmt (multitenancy)
  - no application changes required



JVM: Separating State

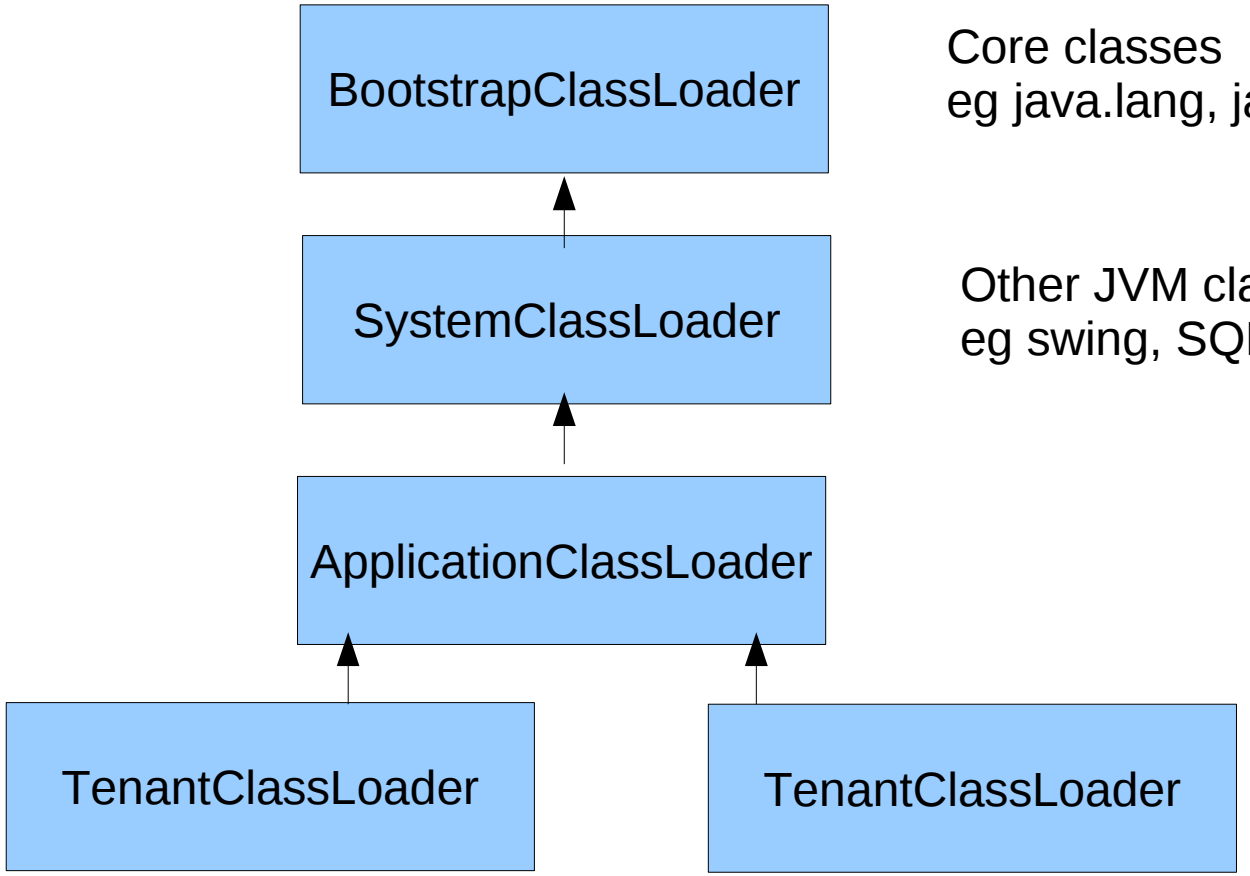
- Static variables are a problem for sharing
- Consider the effect of global defaults



JVM: Separating State

- Use @TenantScope annotation
- Each tenant has its own version of defaultTimeZone
- All JVM classes annotated

Tenants: Separating State

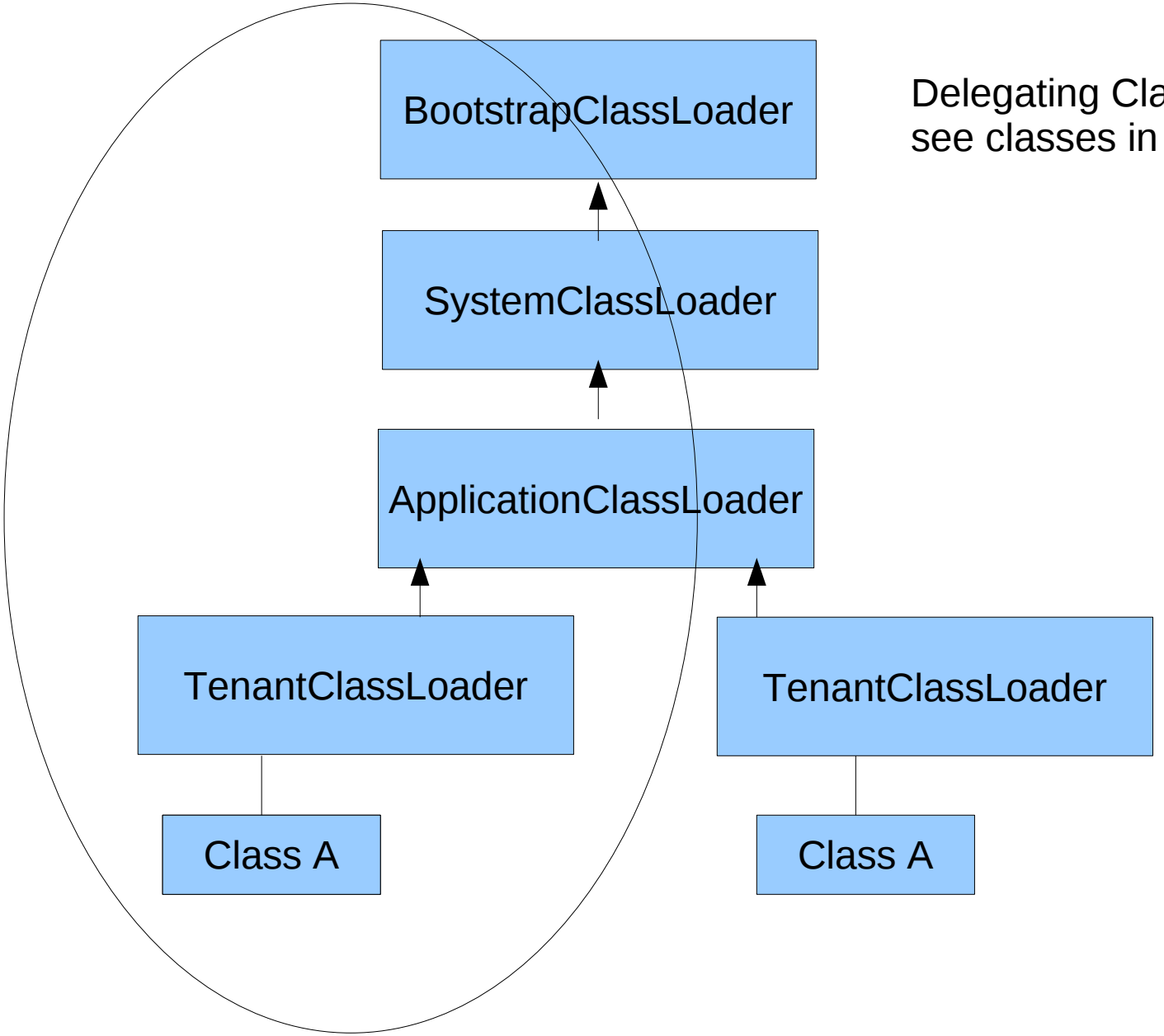


Core classes
eg java.lang, java.util, ...

Other JVM classes
eg swing, SQL, ...

ClassPath

Tenants: Separating State



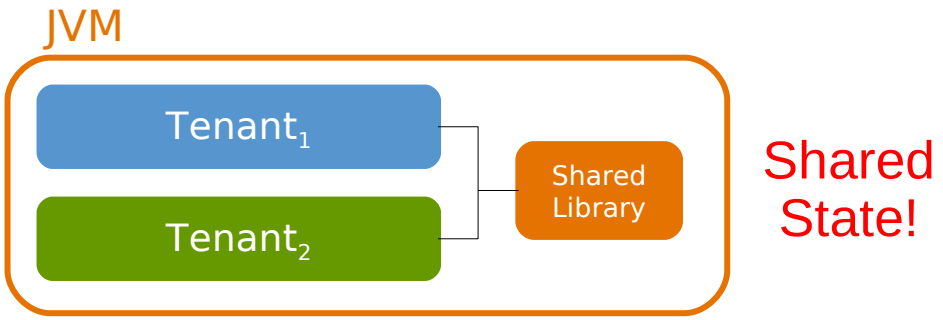
Delegating ClassLoaders only
see classes in their parent Classloaders

Class A != Class A

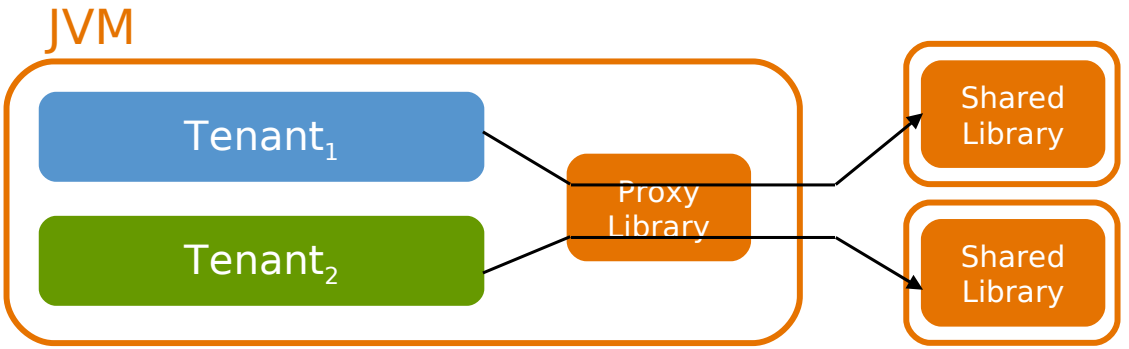
More that just JVM State…

- Throttling of resources
  - Threads, GC, sockets, files (IO in general), native memory
- Past and existing examples do exist!
  - Commercial / In house custom solutions
  - JSR 181 Isolates / 284 Resource Management
- Security is of course huge

Other Thoughts – Native Libraries and shared state



- Native libraries contain state that may not be shareable across tenants



- Use separate processes to manage different state
- Each process now holds the context
- Challenges: Latency

Questions?

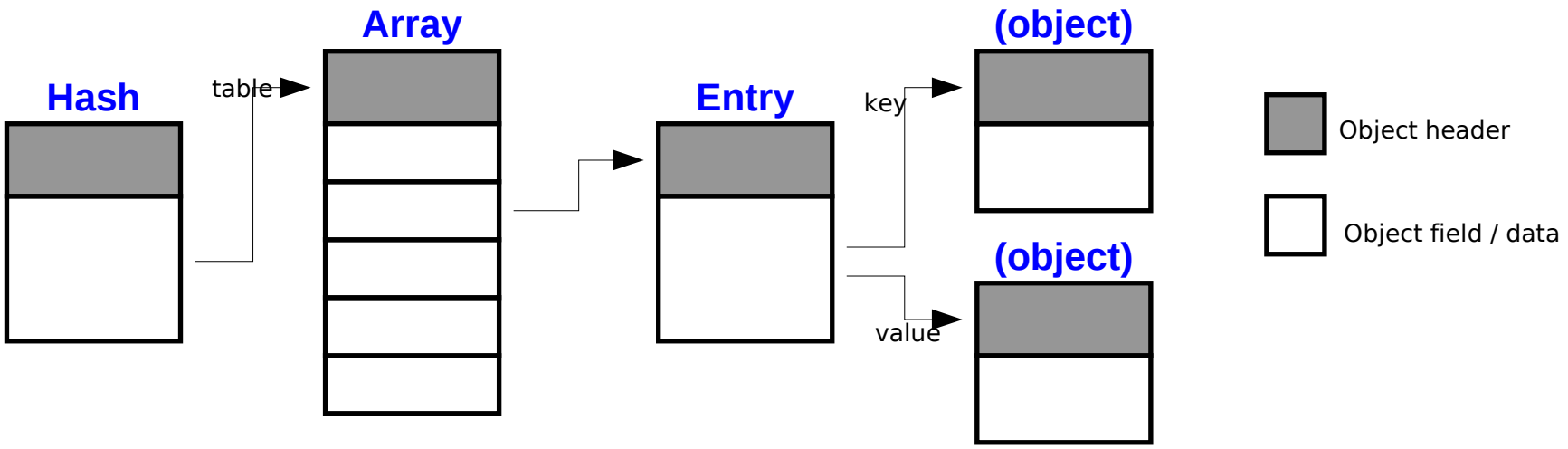
Packed Objects

Problem? What problem?

- JNI just isn't a great way to marshal data
- Locality in Java can matter
- Existing native and data placement stories aren't very good
- In many cases, legacy systems exist – the interop is just terrible

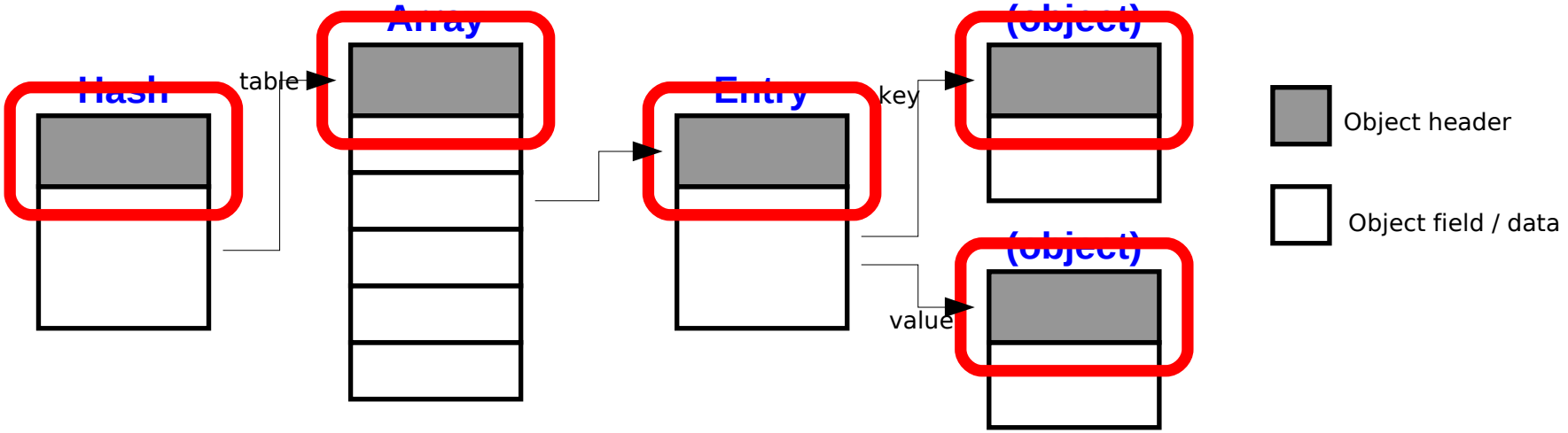
- So we want something that integrates well with the Java language and helps us…

What are we trying to solve?



Simple enough…

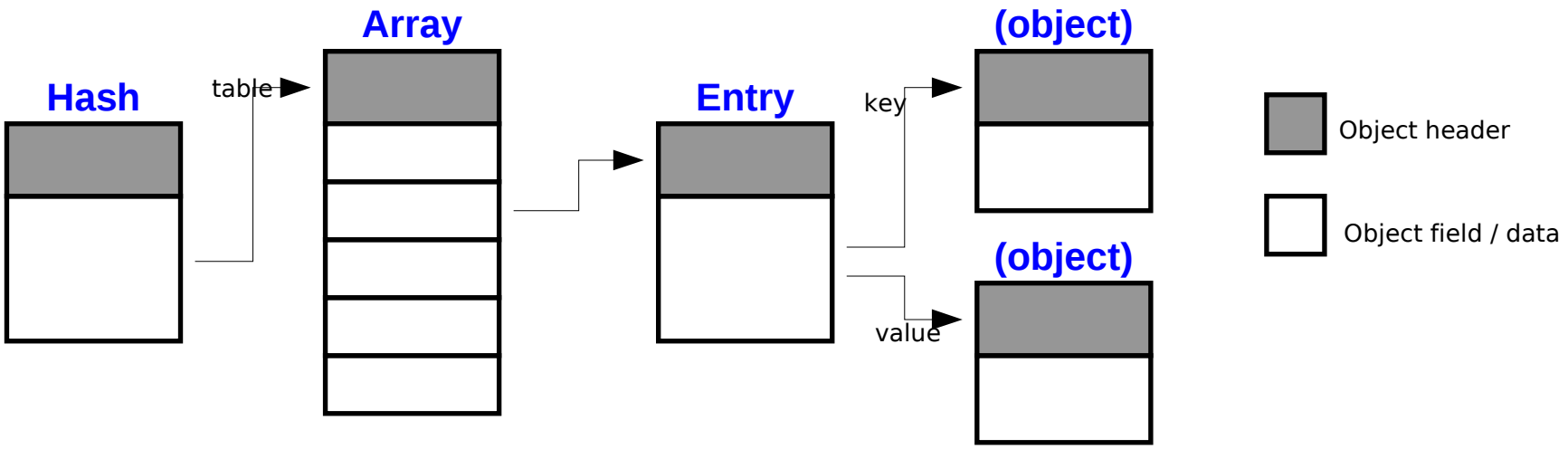
What are we trying to solve?



Simple enough…

- Header overhead

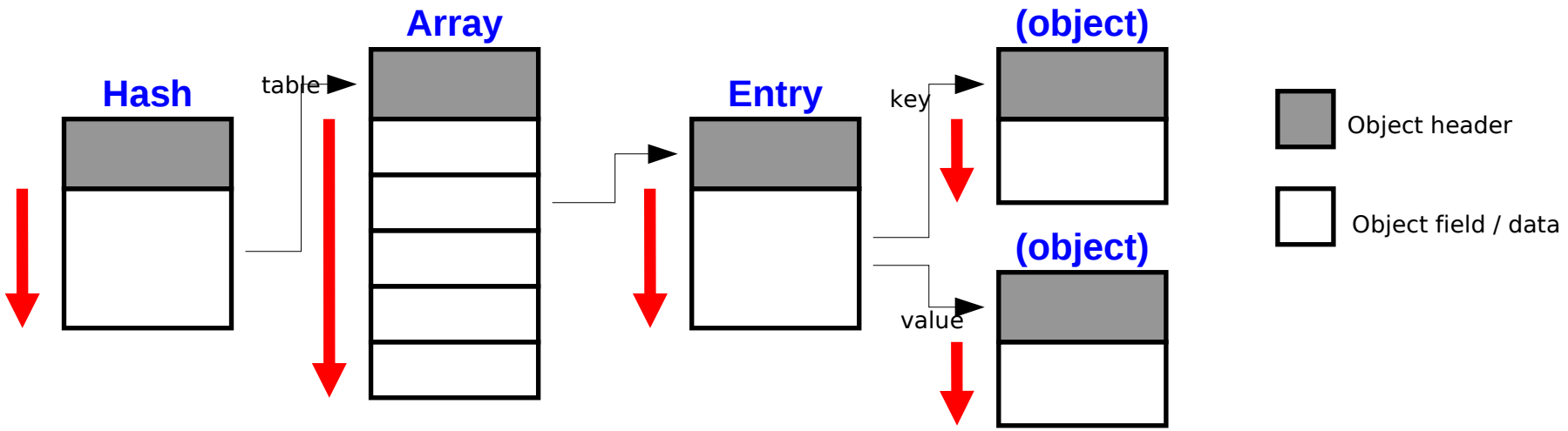
What are we trying to solve?



Simple enough…

- Header overhead
- Pointer chasing

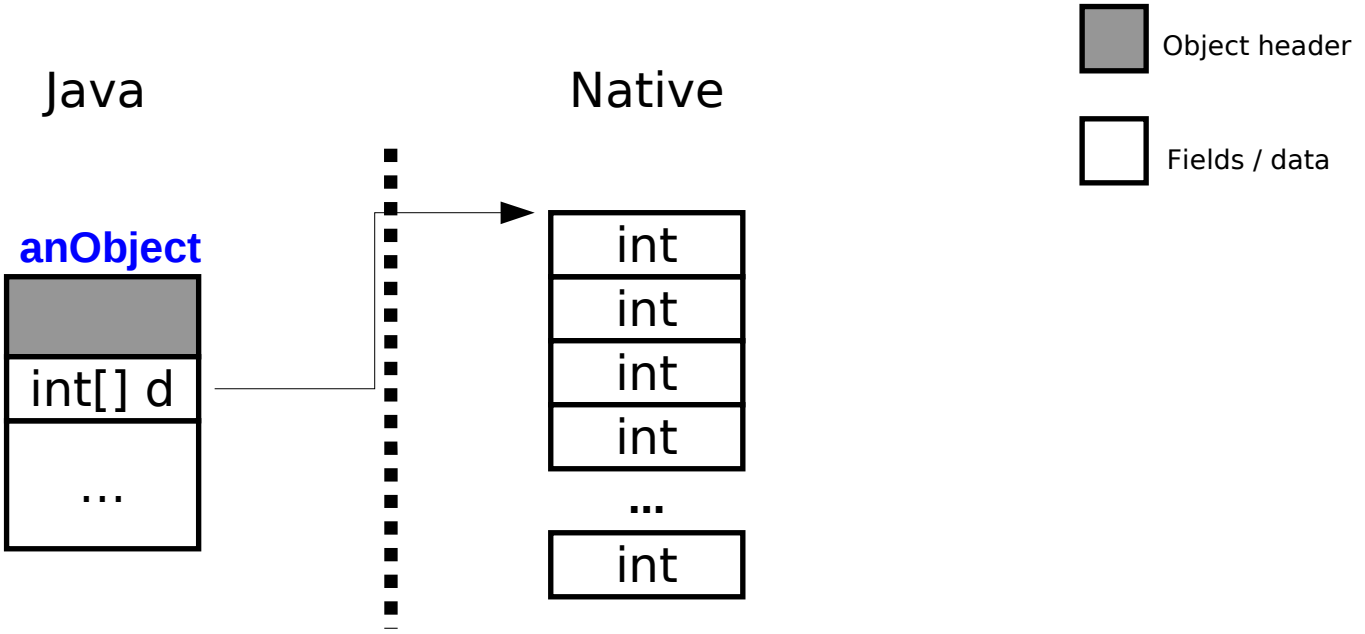
What are we trying to solve?



Simple enough…

- Header overhead
- Pointer chasing
- Locality

What are we trying to solve?



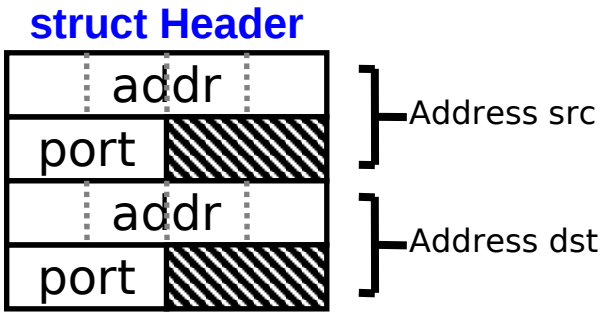
Fighting the Java/Native interface

Ok so we have some criteria…

- Ability to do away with headers
- Ability to bring multiple objects close together
- On heap / off heap seamless referencing of data

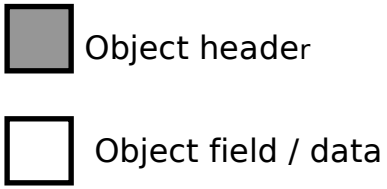
- This actually sounds a lot like C structure types

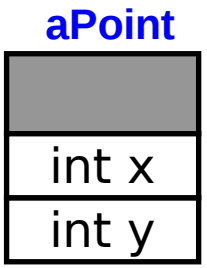
```
struct Address {
  char[4] addr;
  short port;
}
struct Header {
  struct Address src;
  struct Address dst;
}
```



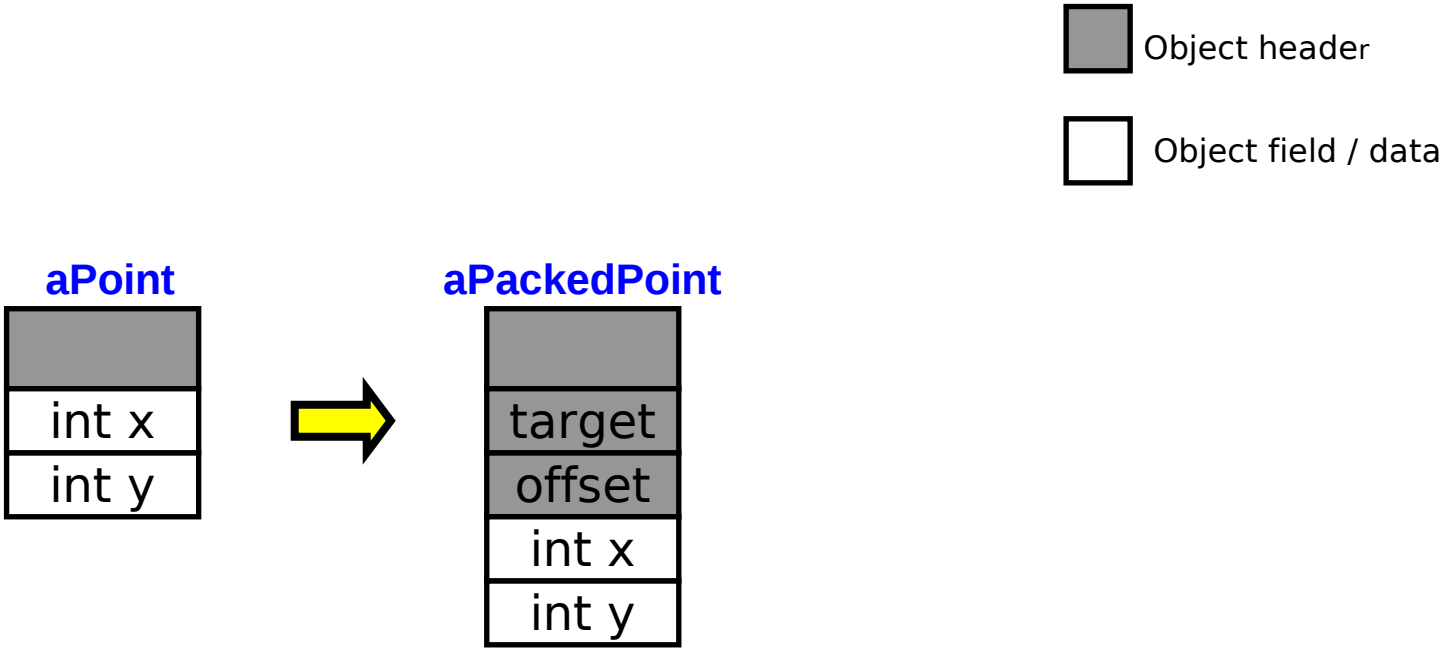
- Packed Objects!

Packed Objects: Under the covers

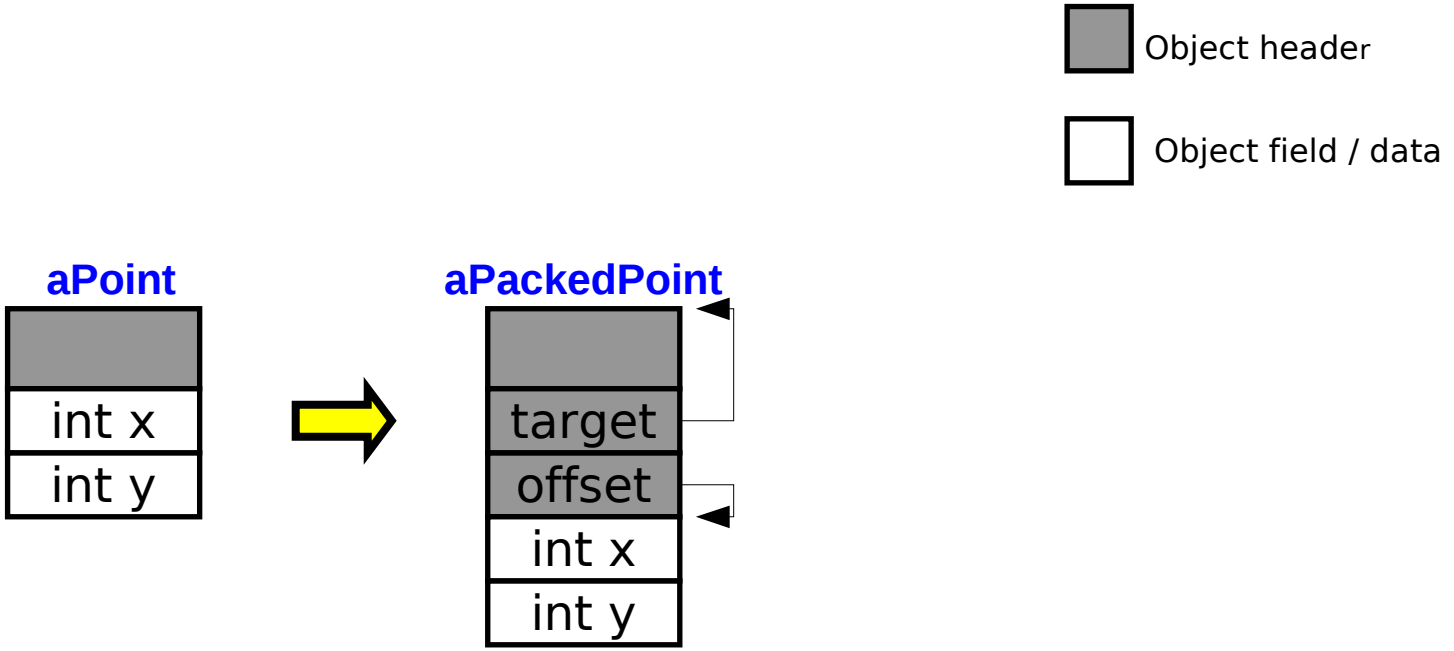




Packed Objects: Under the covers

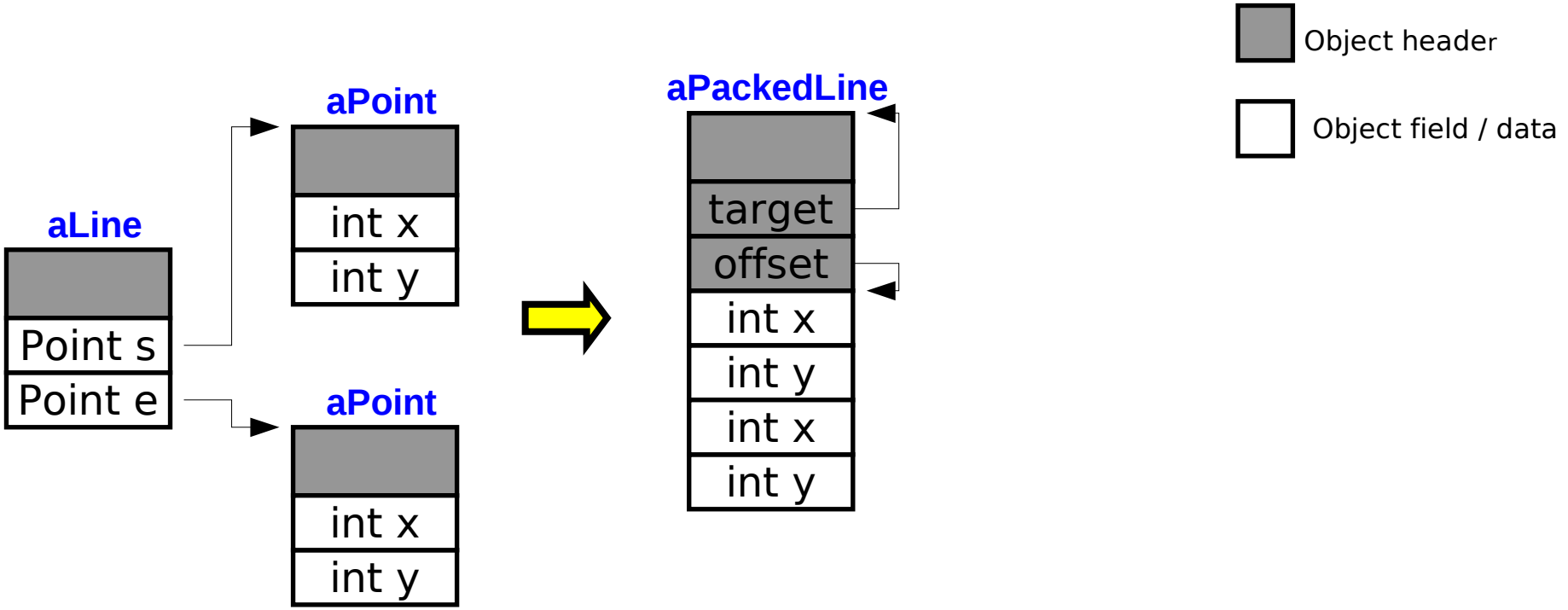


Packed Objects: Under the covers

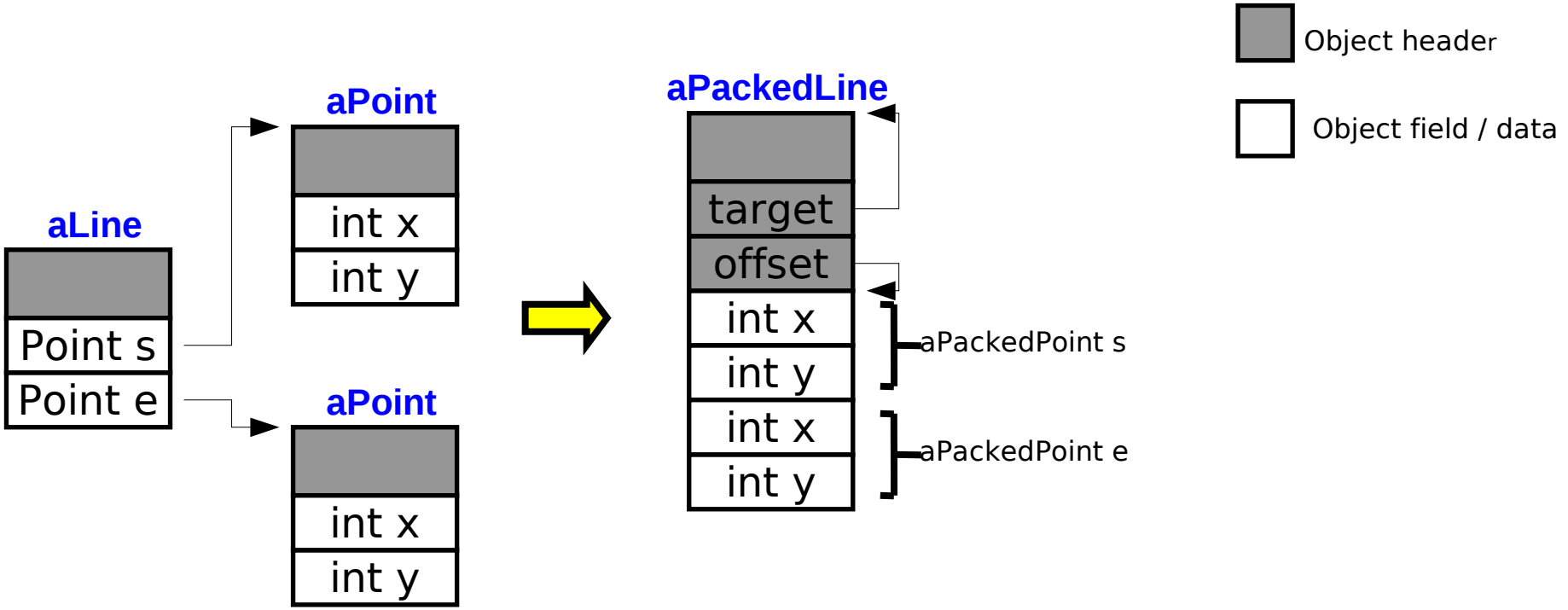


Packed Objects: In Practice

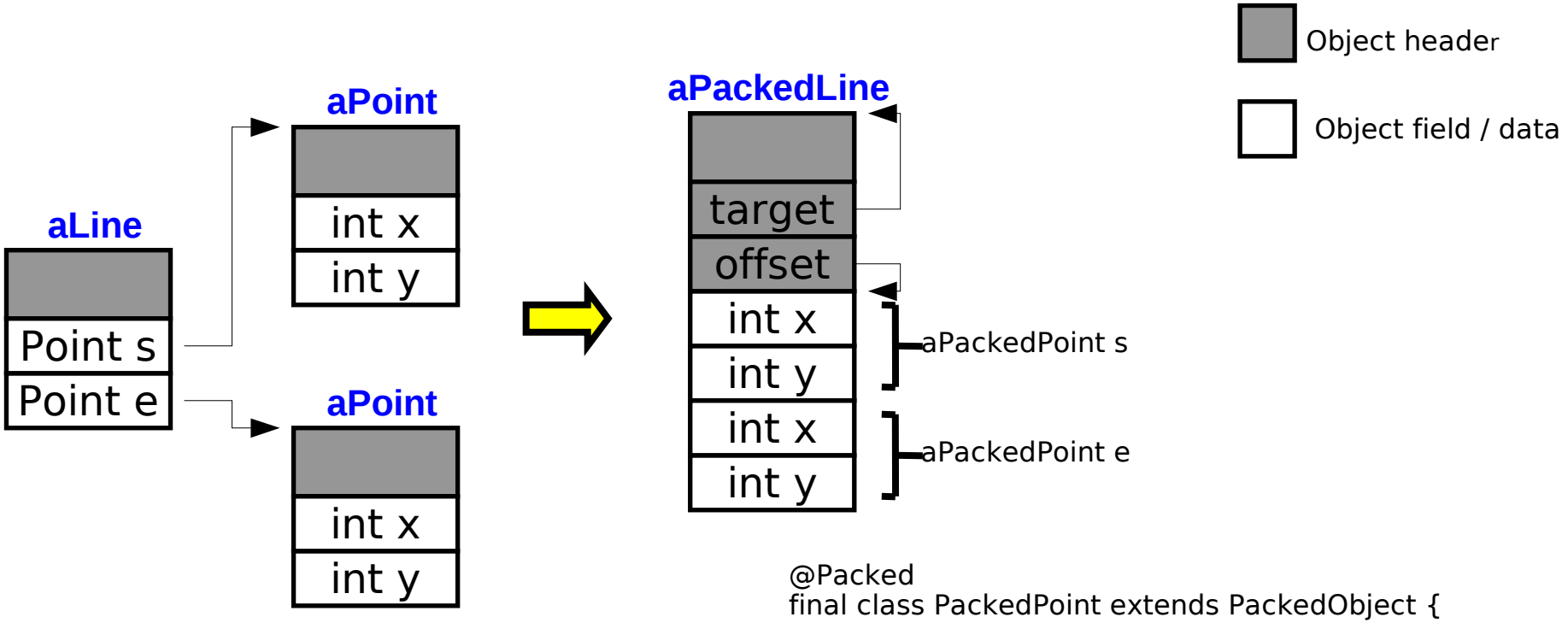
Packed Objects: In Practice



Packed Objects: In Practice

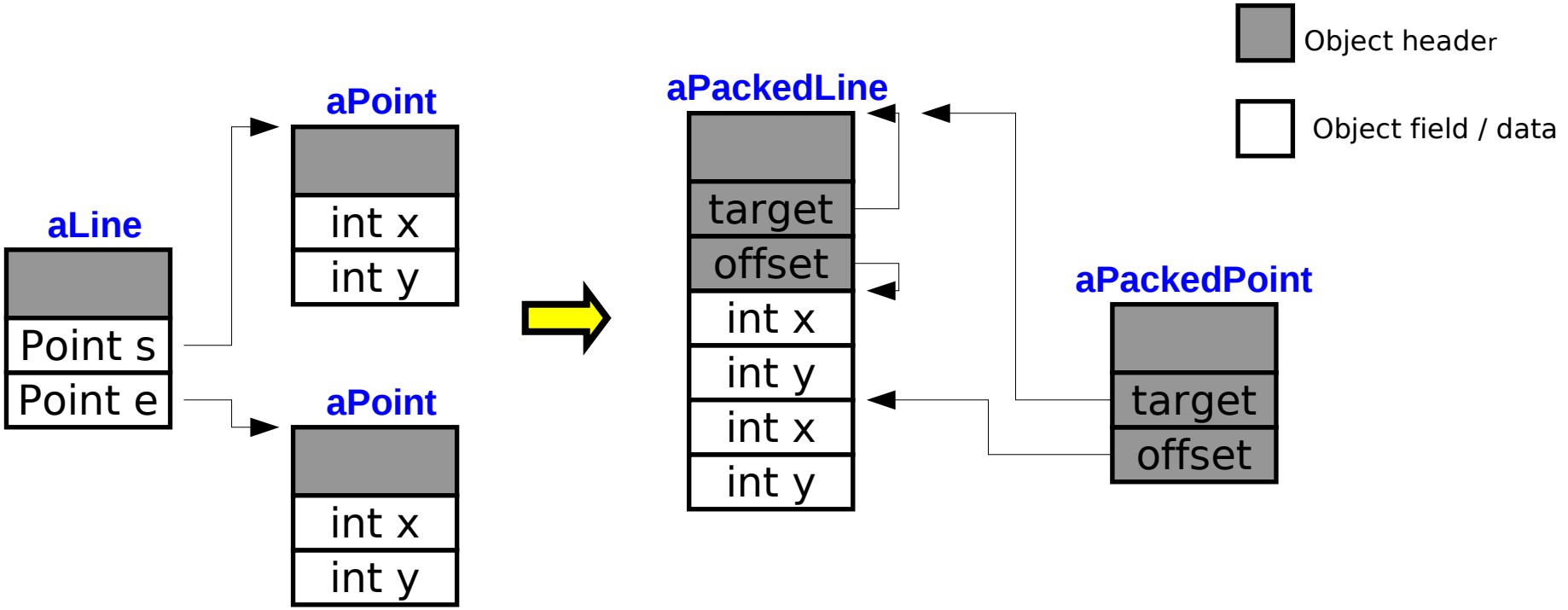


Packed Objects: In Practice



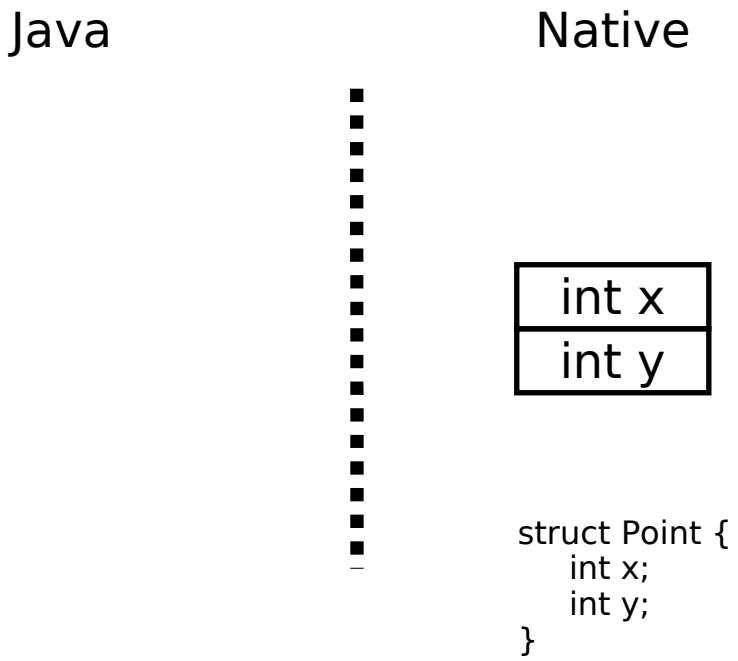

```
@Packed
final class PackedPoint extends PackedObject {
    int x;
    int y;
}

@Packed
final class PackedLine extends PackedObject {
    PackedPoint s;
    PackedPoint e;
}
```

Packed Objects: In Practice

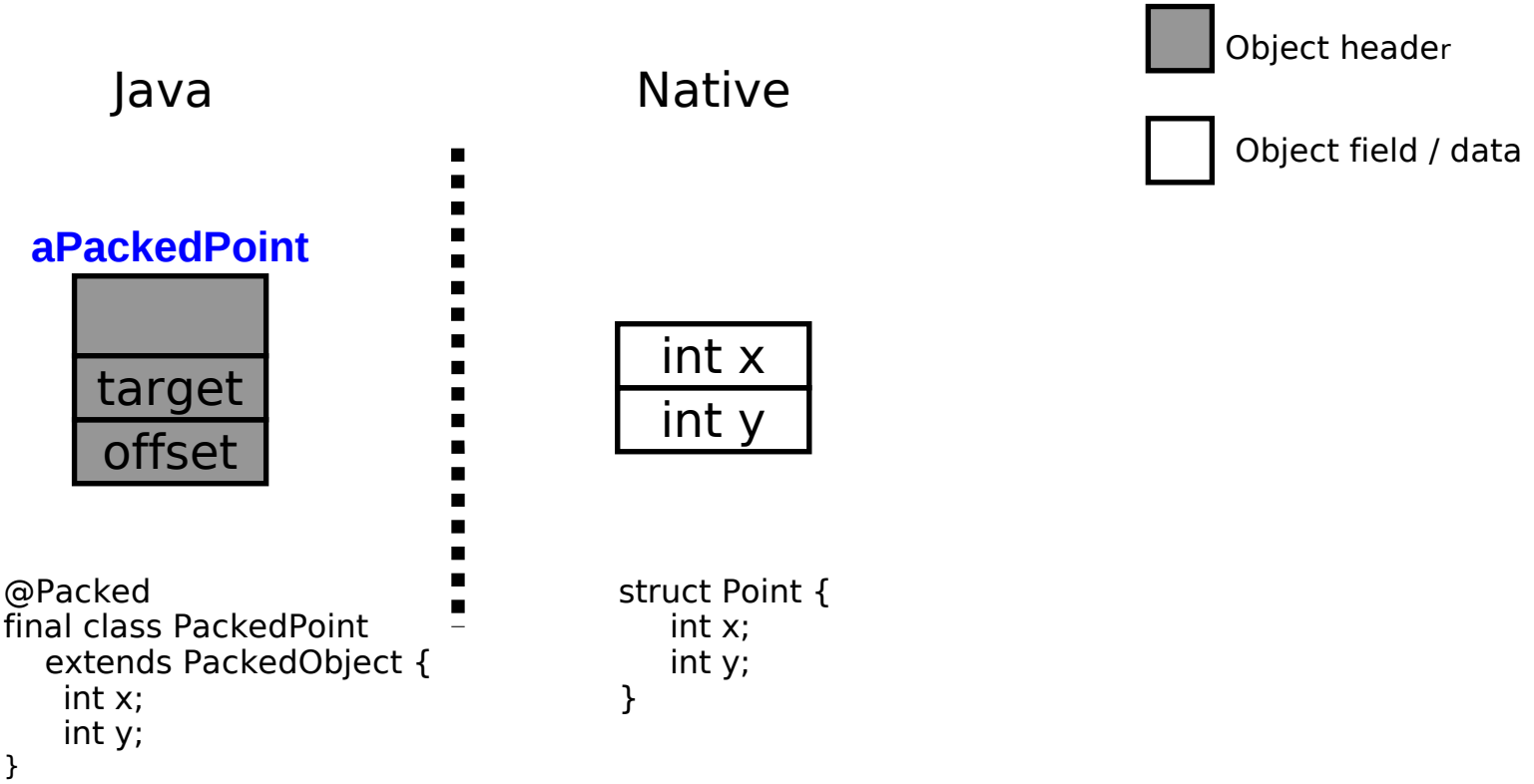


PackedPoint p = aPackedLine.e

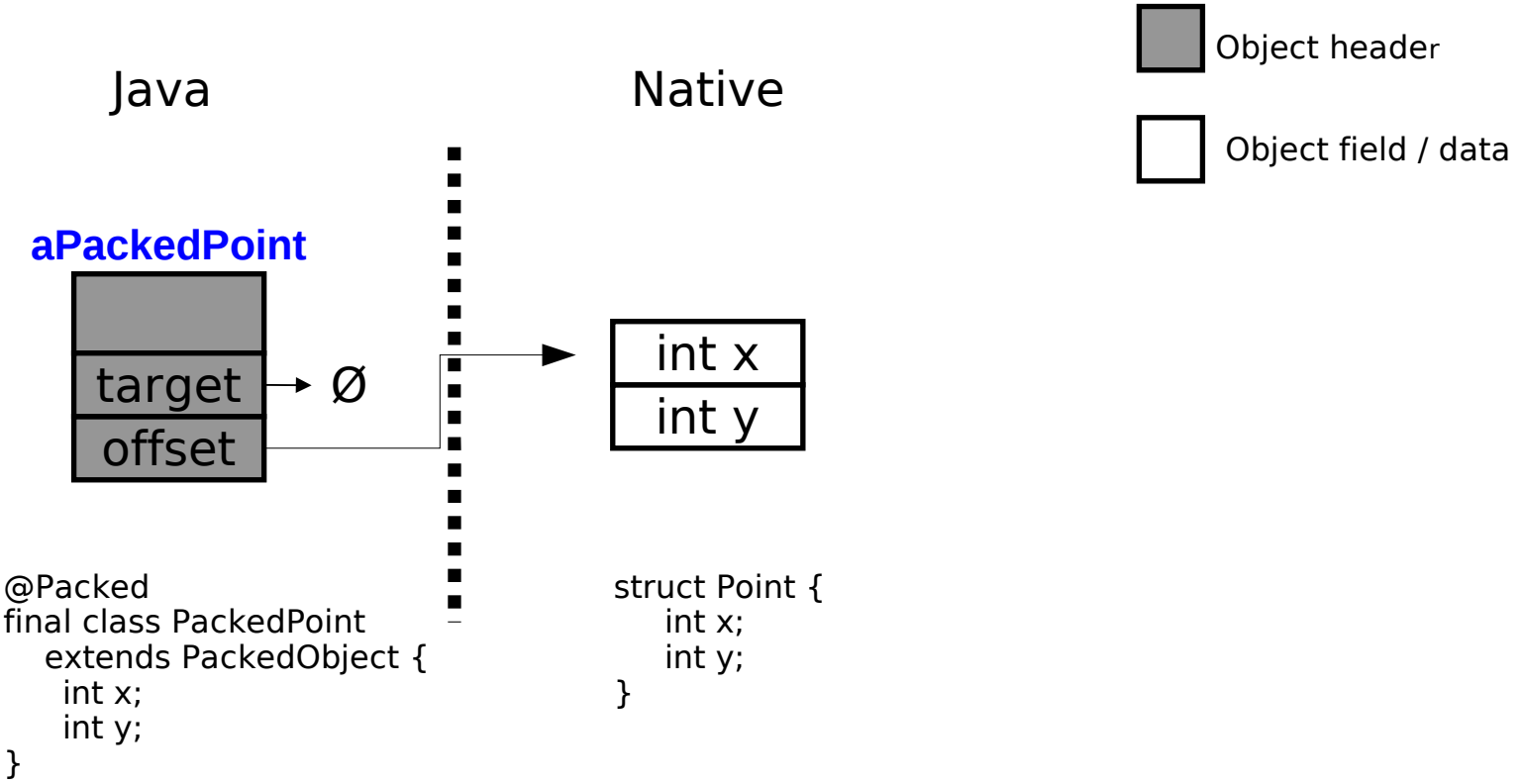
Packed Objects: In Practice with Native Access



Packed Objects: In Practice with Native Access

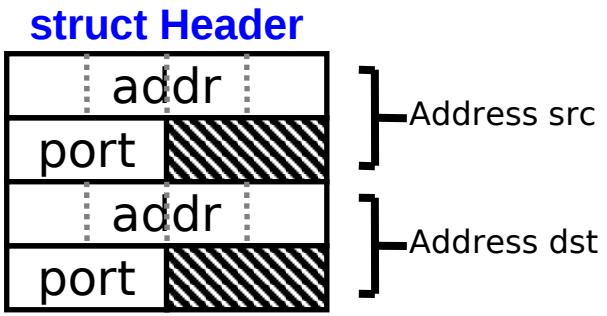


Packed Objects: In Practice with Native Access



Lets Build Something in C!

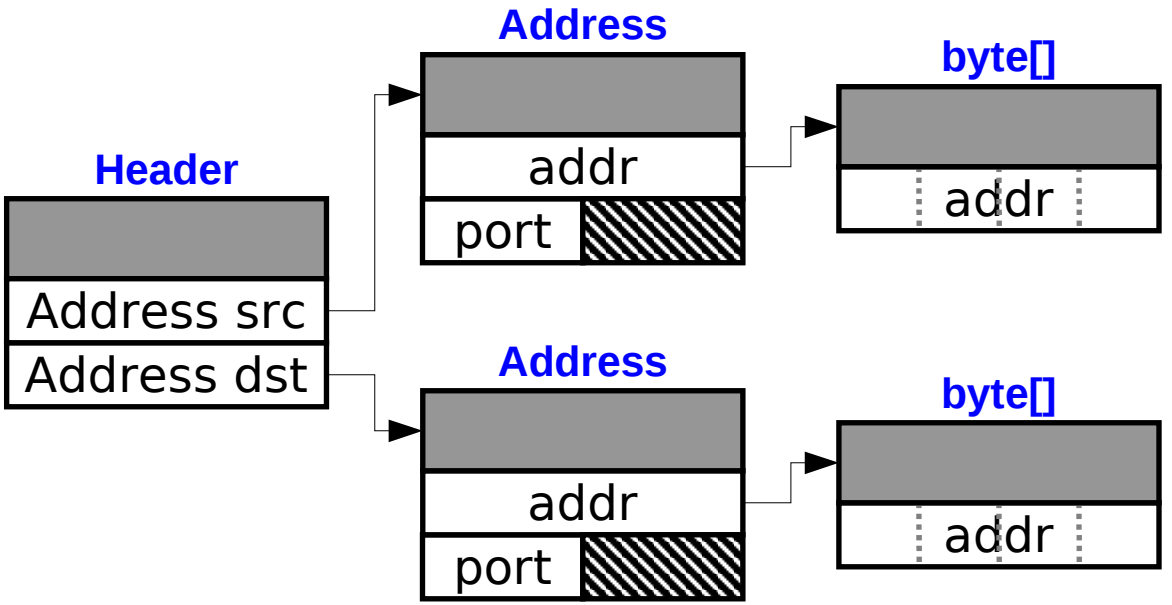
```
struct Address {
   char[4] addr;
   short port;
}
struct Header {
   struct Address src;
   struct Address dst;
}
```



- Nested substructures
- Compact representation
- Alignment aspects

Let's Build the Same "Something" in Java!

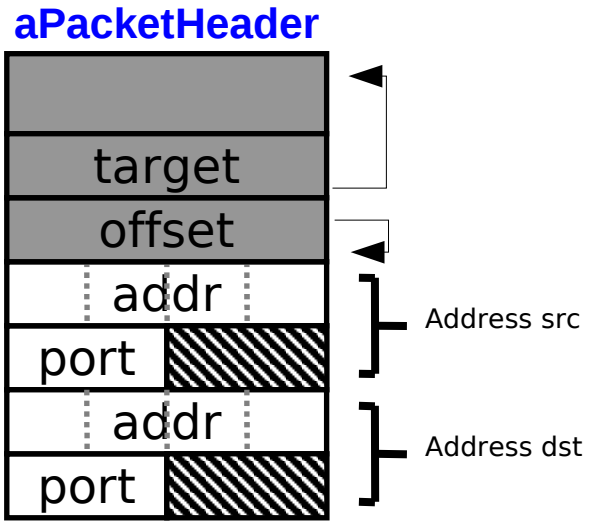
```
class Address {
        byte[] addr;
        short port;
}

class Header {
        Address src;
        Address dst;
}
```



- Headers
- No locality
- Alignment

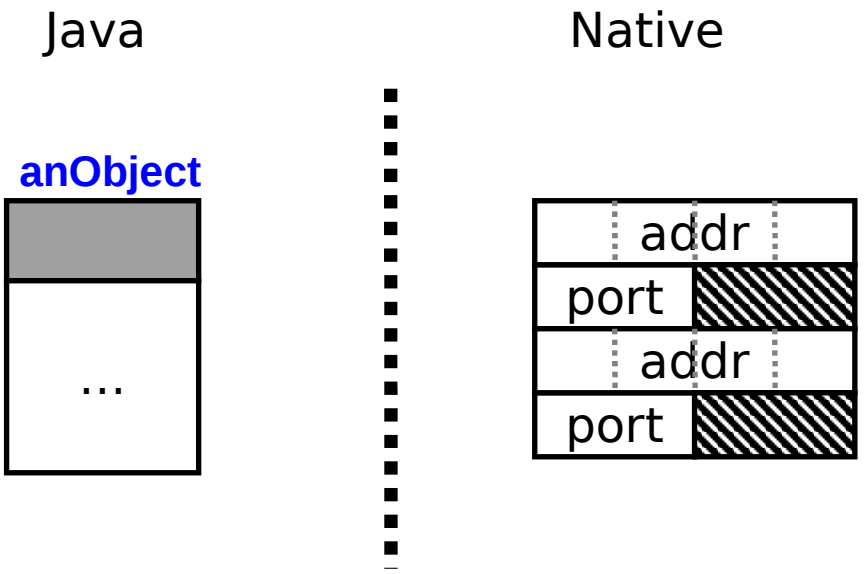
What if we did this with Packed Objects?

```
@Packed
final class Address extends PackedObject {
        PackedByte[[4]] addr;
        short port;
}

@Packed
final class PacketHeader extends PackedObject {
        Address src;
        Address dest;
}
```



- The Java code is pretty clean… and a pretty good result!

What about native access?



How do we implement this normally?

JNI implementation

```
public class PackedHeader {
    private long pointer;

    public byte[] getSourceAddress() { return getSourceAddressImpl(pointer); }
    public short getSourcePort() { return getSourcePortImpl(pointer); }
}
```

```
JNICALL jshort Java_pkg_PackedHeader_getSourcePort(JNIEnv* env, jobject recv, jlong pointer) {
   struct PacketHeader* header = (struct PacketHeader*)pointer;
   return (jshort)header->src.port;
}

JNICALL jbyteArray Java_pkg_PackedHeader_getSourceAddress(JNIEnv* env, jobject recv, jlong pointer) {
   struct PacketHeader* header = (struct PacketHeader*)pointer;
   jbyteArray result = (*env)->NewByteArray(env, 4);
   (*env)->SetByteArrayRegion(env, result, 0, 4, &(header->src.addr));
   return result;
}
```

- Usual “stash pointers in long types” tricks
- JNI costs tend to be high

DirectByteBuffer implementation

```
class PackedHeader {
    private ByteBuffer buffer;
    private static final int SRC_ADDR_OFFSET = 0;
    private static final int SRC_PORT_OFFSET = 4;
    private static final int DEST_ADDR_OFFSET = 8;
    private static final int DEST_PORT_OFFSET = 12;

    public short getSourcePort() { return buffer.getShort(SRC_PORT_OFFSET); }
    public byte[] getSourceAddress() {
        byte[] result = new byte[4];
        buffer.get(result, SRC_ADDR_OFFSET, 4);
        return result;
    }
}
```

- No extra JNI to write (this is good)
- Keeping your indices straight is never fun

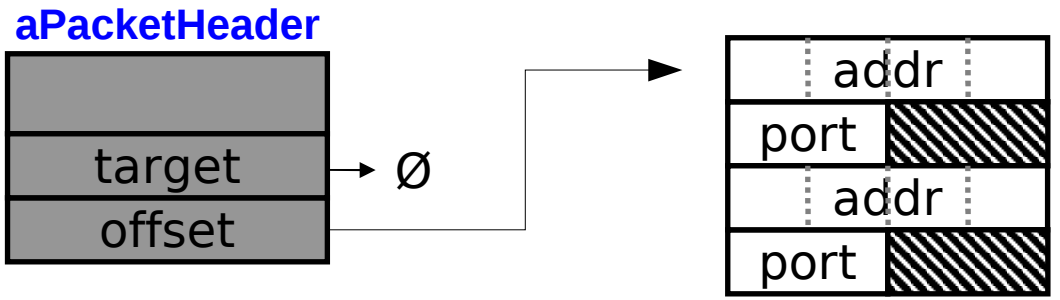
Unsafe implementation

```
class PackedHeader {
    private Unsafe unsafe;
    private long pointer;
    private static final int SRC_ADDR_OFFSET = 0;
    private static final int SRC_PORT_OFFSET = 4;
    private static final int DEST_ADDR_OFFSET = 8;
    private static final int DEST_PORT_OFFSET = 12;

    public short getSourcePort() { return unsafe.getShort(pointer + SRC_PORT_OFFSET); }
    public byte[] getSourceAddress() {
        byte[] result = new byte[4];
        unsafe.copyMemory(null, pointer + SRC_ADDR_OFFSET, result, 0, 4);
        return result;
    }
}
```

- You shouldn’t be here
- Still playing the indices game

PackedObject answer

```
final class PacketHeader extends PackedObject {
   Address src;
   Address dest;

   public short getSourcePort() { return src.port; }
   public PackedByte[] getSourceAddress() { return src.addr; }
}
```



- Looks like natural Java code
- Foregoes JNI
- Same type capable of on-heap representation

Active work and next steps

- Experimenting with this now
- Yes, there are security aspects to be aware of here
- This is potentially part of a larger look at Java / Platform interop
- Not specifically viewed as a cure to GC problems
- This forms the basis for many other solutions to existing problems…

Questions?

References

- **Get Products and Technologies:**
  - IBM Java Runtimes and SDKs:
    - https://www.ibm.com/developerworks/java/jdk/
  - IBM Monitoring and Diagnostic Tools for Java:
    - https://www.ibm.com/developerworks/java/jdk/tools/
- **Learn:**
  - IBM Java InfoCenter:
    - http://publib.boulder.ibm.com/infocenter/java7sdk/v7r0/index.jsp
- **Discuss:**
  - IBM Java Runtimes and SDKs Forum:
    - http://www.ibm.com/developerworks/forums/forum.jspa?forumID=367&start=0

Copyright and Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., and registered in many jurisdictions worldwide.

Other product and service names might be trademarks of IBM or other companies.

A current list of IBM trademarks is available on the Web – see the IBM “Copyright and trademark information” page at URL: www.ibm.com/legal/copytrade.shtml