Inspiration from the Other Side

Some CLR features that might benefit the JVM

Jeroen Frijters

Introduction

Who Am I?

- Jeroen Frijters (pronounced / Yeroon Frighters /)
- Author of IKVM.NET, a JVM implementation + OpenJDK port for the .NET Framework and Mono
- Working with Java since 1997
- Working with .NET since 2000

Disclaimer / Biases

- I don’t work for Oracle or Microsoft
- My biggest bias is towards statically typed languages
- I’m a plumber. I like writing low-level code, but not in C/C++

Terminology

- CLI = Common Language Infrastructure
  - Virtual machine + lowest level class libraries
  - ECMA standard
- CLR = Common Language Runtime
  - Microsoft’s implementation of CLI virtual machine
- IL = CIL = MSIL = (Common | Microsoft) Intermediate Language
- Managed Code vs Unmanaged Code
- Managed Data vs Unmanaged Data

Type System

Primitives

Types	Description
sbyte,bool,char,short,int,long,float,double	Same as the Java primitives
byte,ushort,uint,ulong	Unsigned integral types
IntPtr, UIntPtr	Pointer sized integers
TypedReference	Type safe reference

The CLI supports object references, unmanaged pointers, limited managed pointers, vectors (Java like arrays) and multi-dimensional arrays.

User Defined Value Types

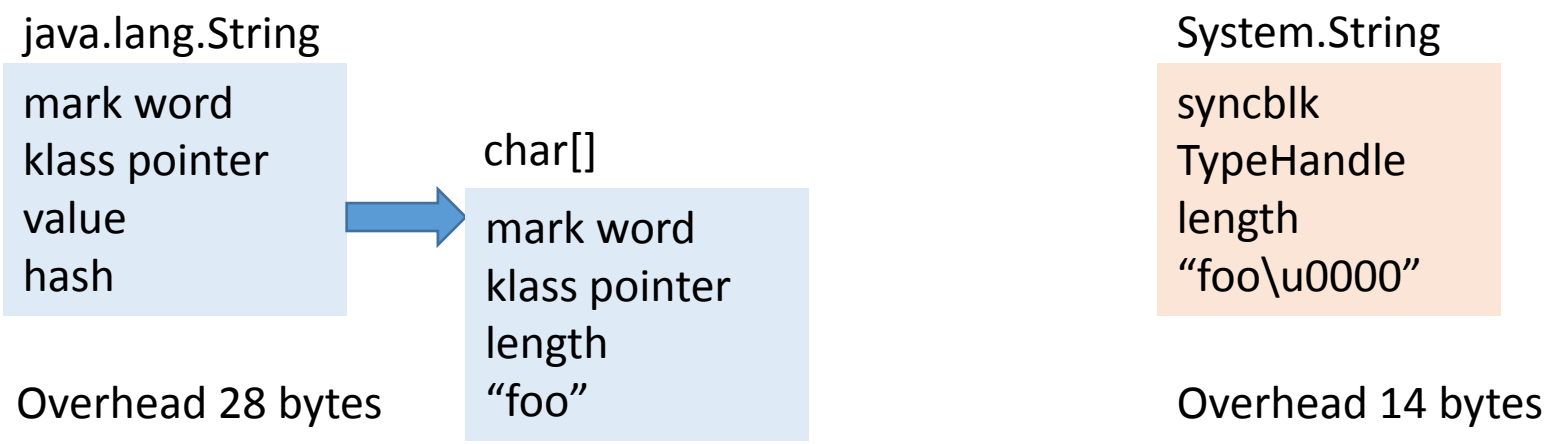
- Originally conceived as a way to make primitive value types less “primitive” and more integrated in the type system.
- Necessary for performance.
- Work very well in conjuction with reified generics.
- Not very suitable for (secure) encapsulation, due to tearing.

Arrays

- Common base class System.Array
  - Including boxing GetValue() and SetValue()
- Arrays implement standard collection interfaces
- RuntimeHelpers.InitializeArray()
- Buffer.BlockCopy()

java.lang.String vs System.String

- On the CLR System.String is a variable length type
- This saves memory and an indirection level
- JNI allows two-step string creation, but IKVM.NET does not support this and so far I've never encountered any code that does this. On HotSpot this can be implemented with a GC hack.



Reified Generics

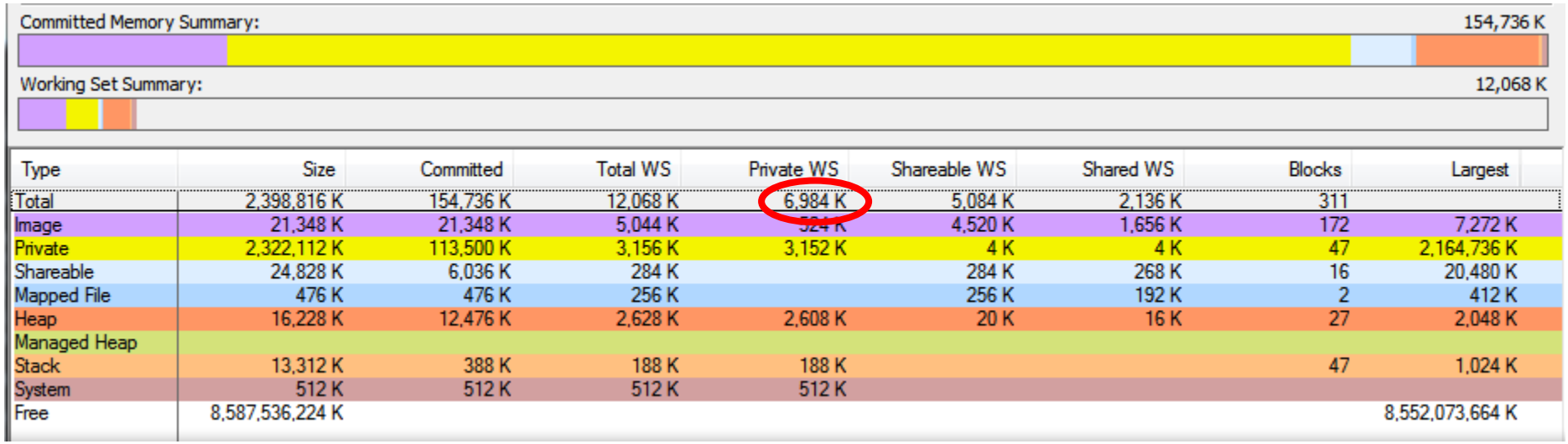
- Classes, interfaces, value types, delegates and methods can be generic.
- Declaration site variance for type parameters of interfaces and delegates.
- Great for performance in collections in combination with value types.
- Instantiations with reference type type parameters typically share implementation, but instantiations with value types do not.

Module System

Module System

- CLR has a great module system
- **internal** access modifier
- .class file is a horrible format
  - too many size limitations
  - support for inner classes in the VM
  - try / finally in the class file format
- Metadata API
- NGEN

NGEN HelloWorld

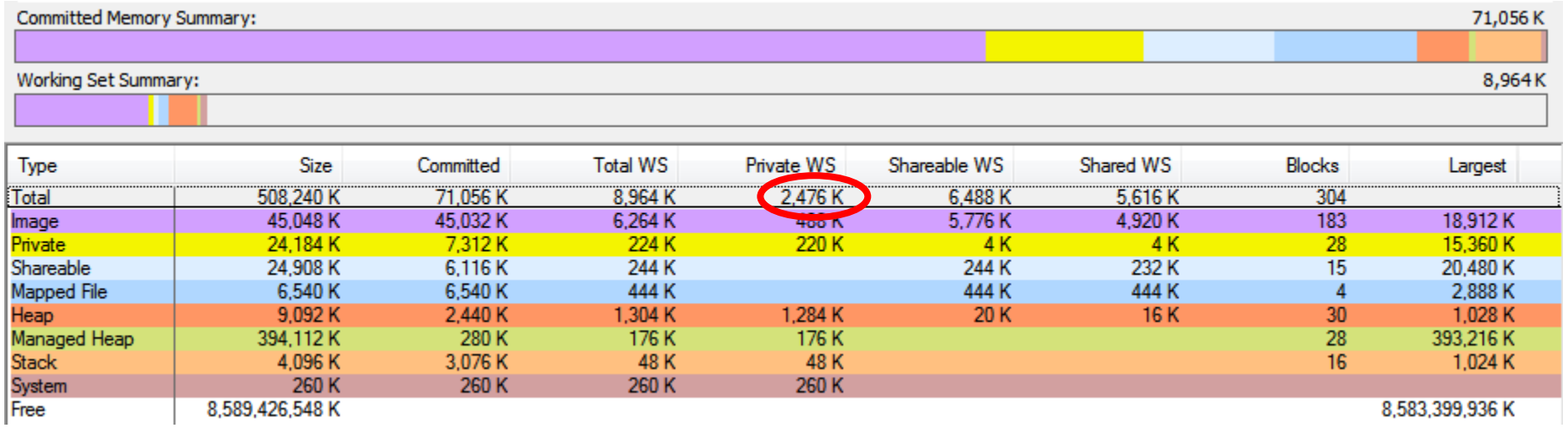


Committed Memory Summary: 154,736 K

Working Set Summary: 12,068 K

Type	Size	Committed	Total WS	Private WS	Shareable WS	Shared WS	Blocks	Largest
Total	2,398,816 K	154,736 K	12,068 K	6,984 K	5,084 K	2,136 K	311	
Image	21,348 K	21,348 K	5,044 K	324 K	4,520 K	1,656 K	172	7,272 K
Private	2,322,112 K	113,500 K	3,156 K	3,152 K	4 K	4 K	47	2,164,736 K
Shareable	24,828 K	6,036 K	284 K		284 K	268 K	16	20,480 K
Mapped File	476 K	476 K	256 K		256 K	192 K	2	412 K
Heap	16,228 K	12,476 K	2,628 K	2,608 K	20 K	16 K	27	2,048 K
Managed Heap								
Stack	13,312 K	388 K	188 K	188 K			47	1,024 K
System	512 K	512 K	512 K	512 K				
Free	8,587,536,224 K							8,552,073,664 K

6,984 K



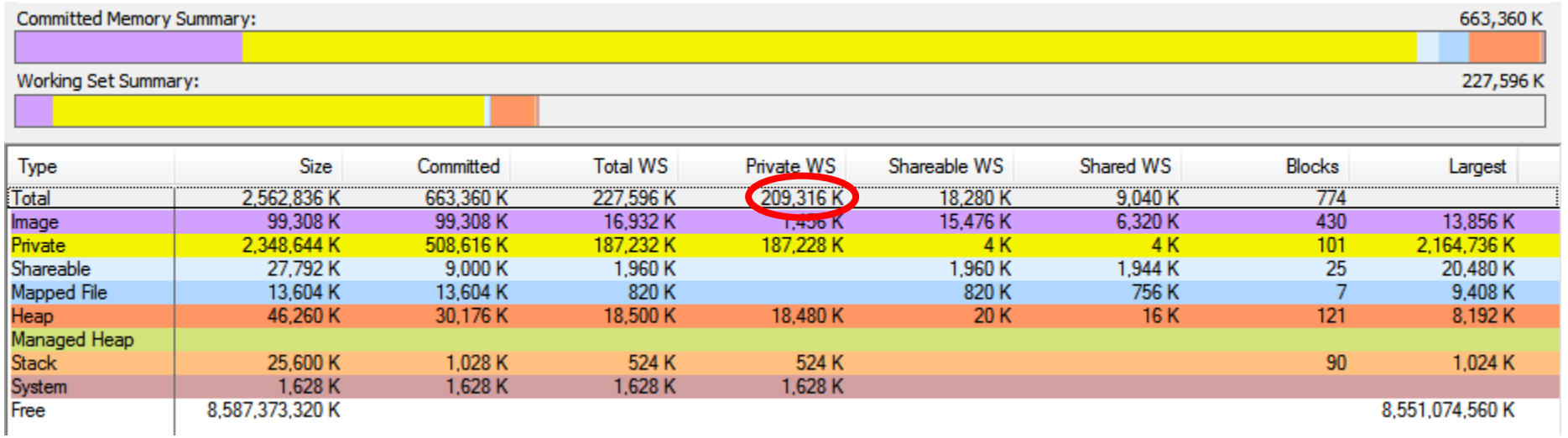
Committed Memory Summary: 71,056 K

Working Set Summary: 8,964 K

Type	Size	Committed	Total WS	Private WS	Shareable WS	Shared WS	Blocks	Largest
Total	508,240 K	71,056 K	8,964 K	2,476 K	6,488 K	5,616 K	304	
Image	45,048 K	45,032 K	6,264 K	488 K	5,776 K	4,920 K	183	18,912 K
Private	24,184 K	7,312 K	224 K	220 K	4 K	4 K	28	15,360 K
Shareable	24,908 K	6,116 K	244 K		244 K	232 K	15	20,480 K
Mapped File	6,540 K	6,540 K	444 K		444 K	444 K	4	2,888 K
Heap	9,092 K	2,440 K	1,304 K	1,284 K	20 K	16 K	30	1,028 K
Managed Heap	394,112 K	280 K	176 K	176 K			28	393,216 K
Stack	4,096 K	3,076 K	48 K	48 K			16	1,024 K
System	260 K	260 K	260 K	260 K				
Free	8,589,426,548 K							8,583,399,936 K

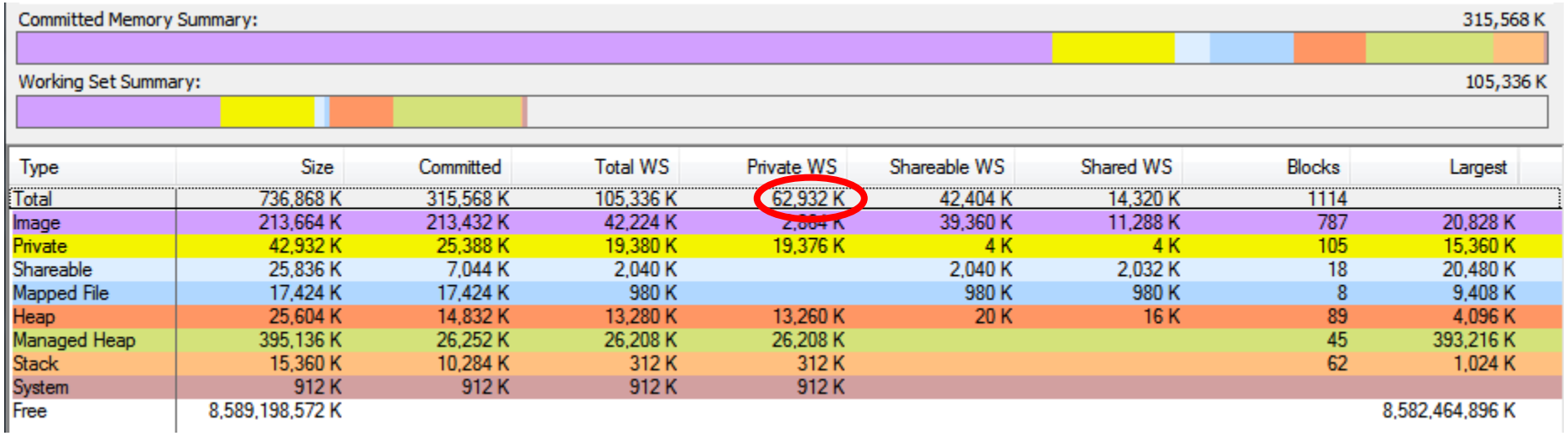
2,476 K

NGEN SwingSet2



Type	Size	Committed	Total WS	Private WS	Shareable WS	Shared WS	Blocks	Largest
Total	2,562,836 K	663,360 K	227,596 K	209,316 K	18,280 K	9,040 K	774	
Image	99,308 K	99,308 K	16,932 K	1,456 K	15,476 K	6,320 K	430	13,856 K
Private	2,348,644 K	508,616 K	187,232 K	187,228 K	4 K	4 K	101	2,164,736 K
Shareable	27,792 K	9,000 K	1,960 K		1,960 K	1,944 K	25	20,480 K
Mapped File	13,604 K	13,604 K	820 K		820 K	756 K	7	9,408 K
Heap	46,260 K	30,176 K	18,500 K	18,480 K	20 K	16 K	121	8,192 K
Managed Heap								
Stack	25,600 K	1,028 K	524 K	524 K			90	1,024 K
System	1,628 K	1,628 K	1,628 K	1,628 K				
Free	8,587,373,320 K							8,551,074,560 K

209,316 K



Type	Size	Committed	Total WS	Private WS	Shareable WS	Shared WS	Blocks	Largest
Total	736,868 K	315,568 K	105,336 K	62,932 K	42,404 K	14,320 K	1114	
Image	213,664 K	213,432 K	42,224 K	2,864 K	39,360 K	11,288 K	787	20,828 K
Private	42,932 K	25,388 K	19,380 K	19,376 K	4 K	4 K	105	15,360 K
Shareable	25,836 K	7,044 K	2,040 K		2,040 K	2,032 K	18	20,480 K
Mapped File	17,424 K	17,424 K	980 K		980 K	980 K	8	9,408 K
Heap	25,604 K	14,832 K	13,280 K	13,260 K	20 K	16 K	89	4,096 K
Managed Heap	395,136 K	26,252 K	26,208 K	26,208 K			45	393,216 K
Stack	15,360 K	10,284 K	312 K	312 K			62	1,024 K
System	912 K	912 K	912 K	912 K				
Free	8,589,198,572 K							8,582,464,896 K

62,932 K

Two Types of Static Initialization

```
using System;

class Type1 {
  static int field = Init();
  static int Init() {
    Console.WriteLine("Hello from Init");
    return 42;
  }
  static void Main() { }
}
```

```
using System;

class Type1 {
  static int field;
  static Type1() {
    Console.WriteLine("Hello from Init");
    field = 42;
  }
  static void Main() { }
}
```

Native Interop

JNI

The CLR has no equivalent of JNI (although there is a hosting API). Instead there are powerful interop mechanisms exposed to managed code:

- P/Invoke
- delegate marshalling
- explicit structure layout
- GCHandle
- unsafe code

Custom Attributes

When Java copied custom attributes, one big mistake was made:

"These annotations do not directly affect the semantics of a program."
– JSR175 Proposed Final Draft

```
[ThreadStatic]
static bool runningFinalize;
```

```
[DllImport("kernel32")]
static extern IntPtr GetStdHandle(int nS
```

```
[SecurityCritical]
public static object allocateInstance(ob
```

```
[StructLayout(LayoutKind.Explicit)]
internal struct jvalue {
  [FieldOffset(0)]
  public jboolean z;
  [FieldOffset(0)]
  public jbyte b;
  [FieldOffset(0)]
  public jchar c;
  [FieldOffset(0)]
  public jshort s;
  [FieldOffset(0)]
  …
```

Finalization

- SafeHandle / CriticalFinalizerObject
- GC.SuppressFinalize() and GC.ReRegisterForFinalize()
- Ephemerons

```
@DllImportAttribute.Annotation("kernel32")
private static native int
  FlushFileBuffers(cli.Microsoft.Win32.SafeHandles.SafeFileHandle handle);
```

Ecosystem

Tools

- ilasm / ildasm
  - Standardized textual representation of CLI module
- peverify
  - Command line tool to run the CLR verifier on a complete module

Are there any Java apps left?

Are there any Java apps left?

- sun.misc.Unsafe is now effectively part of the public API
- Some examples of sun.misc.Unsafe users:
  - Akka
  - Cassandra
  - db4o
  - Scala
  - JRuby
  - Netty
  - XStream

sun.misc.Unsafe vs .NET

sun.misc.Unsafe	.NET
getUnsafe()	[SecurityPermission(LinkDemand, …)]
int getInt(Object o, long offset)	Marshal.ReadInt32(object ptr, int ofs)
void putInt(Object o, long offset, int x)	Marshal.WriteInt32(object ptr, int ofs, int val)
byte getByte(long address)	Marshal.ReadByte(IntPtr ptr)
long allocateMemory(long bytes)	Marshal.AllocHGlobal(IntPtr cb)
void ensureClassInitialized(Class c)	RuntimeHelpers.RunClassConstructor(RuntimeTypeHandle type)
Object allocateInstance(Class cls)	FormatterServices.GetUninitializedObject(Type type)
void monitorEnter(Object o)	Monitor.Enter(object obj)
boolean tryMonitorEnter(Object o)	Monitor.TryEnter(object obj)
boolean compareAndSwapObject(…)	Interlocked.CompareExchange<T>(ref T location, T value, T comp)
Object getObjectVolatile(…)	Thread.VolatileRead(ref object address)

Concurrency

Low Level Concurrency Constructs

- Java has great high level concurrency support, but lacks low level concurrency primitives.
- java.util.concurrent.atomic is not good enough.
- The CLR has APIs to directly manipulate fields (and array locations) using volatile or interlocked operations. This requires reference parameters and reified generics.

```
private int[] counter;
private Node next;

Interlocked.Add(ref counter[i], 42);
Interlocked.CompareExchange(ref next, update, expected);
```

Alternative Design for Atomic Operations

```
import java.util.concurrent.atomic.annotations.*;

class Demo {
  private int counter;

  @CompareAndSet(“counter”)
  private native boolean updateCounter(int expect, int update);

  @PutVolatile(“counter”)
  public native void setCounter(int value);

  @PutVolatileArray
  public native static void setInt(Node[] array, int index, Node value);
}
```

Methods and Fields

Virtual Methods

- newslot
- methodimpl
- No implicit interface implementation across modules

```
.method public hidebysig newslot strict virtual
        instance bool  equals(object obj) cil managed
{
  .override [mscorlib]System.Object::Equals
  …
}
```

Custom Modifiers

- How do you compile C++ without name mangling?
- Surprisingly useful for language interop

```
.method public instance void
   foo(int32 i)

.method public instance void
   foo(int32 modopt([mscorlib]System.Runtime.CompilerServices.IsLong) j)
```

```
.method public instance object
   modopt([IKVM.Runtime]IKVM.Attributes.AccessStub)
   modopt(java.lang.AbstractStringBuilder)
        '<bridge>append'(string A_1)
```

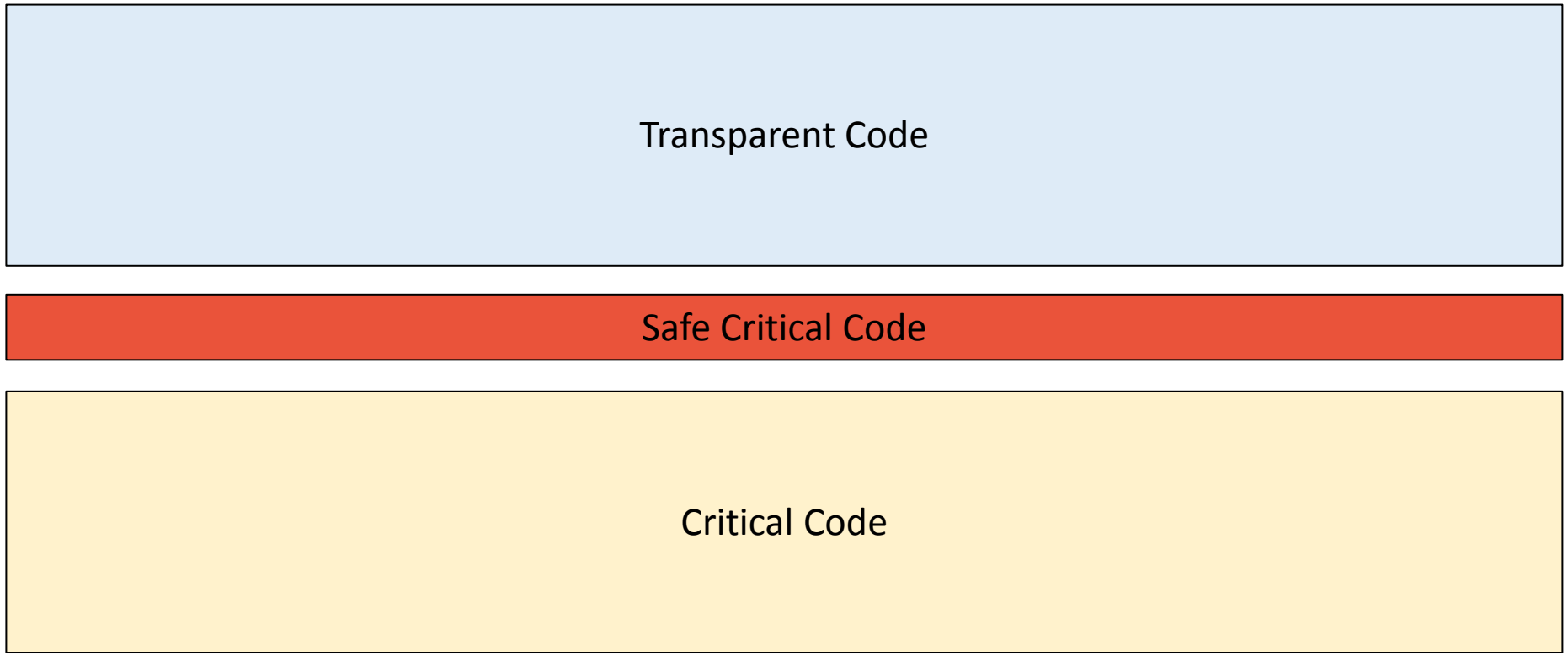
```
.field private static class java.lang.SecurityManager
   modreq([mscorlib]System.Runtime.CompilerServices.IsVolatile) security
```

Member Resolution

- Private members hide inherited members from subclasses. Examples in Java class libraries:
  - MetalScrollBarUI. thumbColor & thumbHighlightColor
  - javax.swing.JTree.AccessibleJTree.AccessibleJTreeNode.accessibleParent
- Accessibility and method overriding is a mess.
- Consider interaction with module system.

Security & Sandbox

Transparent and Security Critical Code



Application Domains

- Isolates?
- JNI is a problem.

Thread.stop() vs Thread.Abort()

- Thread.Abort() is not deprecated!
- It is possible to write Thread.Abort() safe code on the CLR.

Miscellaneous

Interesting IL Opcodes

- conv.ovf.*
- add.ovf, sub.ovf, mul.ovf
- ldelema, ldflda, ldsflda, ldarga, ldloca
- ldind.*, stind.*
- unaligned, volatile
- tailcall
- calli

Static vs Virtual Extension Methods

- Static extension methods are modular, virtual extension methods will forever after clutter up the interface.
- Static extension methods are composable. Third parties can create extension methods, not just the interface definer.

Thank You!

@JeroenFrijters

http://weblog.ikvm.net/

jeroen@frijters.net