Safely Shoot Yourself in the Foot with Java 9

Dr Heinz M. Kabutz

Last Updated 2017-11-04

Javaspecialists.eu
java training

Project Jigsaw: Primary Goals (Reinhold)

- **Make the Java SE Platform, and the JDK, more easily scalable down to small computing devices;**

- **Improve the security and maintainability of Java SE Platform Implementations in general, and the JDK in particular;**

- **Enable improved application performance; and**

- **Make it easier for developers to construct and maintain libraries and large applications, for both the Java SE and EE Platforms.**

javaspecialists.eu

So What Can We Do in Java?

- **Java was supposed to be 100% secure**

- **In early versions we often had to code JNI**

- **sun.misc.Unsafe introduced to allow JDK implementers to**
  - **Create objects without calling their constructors**
  - **Allocate large blocks of native memory and free it again**
  - **Read and write memory locations directly using CAS (peek & poke)**
  - **Fences**
  - **Throw exceptions unchecked**
  - **Release native buffer resources (Java 9)**

javaspecialists.eu

Javaspecialists.eu

Evolution of "Unsafe" Usage

● **ConcurrentLinkedQueue.Node in Java 5 and 6**

```java
private static class Node<E> {
    private volatile E item;
    private volatile Node<E> next;

    private static final AtomicReferenceFieldUpdater<Node, Object> itemUpdater =
        AtomicReferenceFieldUpdater.newUpdater(Node.class, Object.class, "item");

    E getItem() { return item; }

    boolean casItem(E cmp, E val) {
        return itemUpdater.compareAndSet(this, cmp, val);
    }

    void setItem(E val) {
        itemUpdater.set(this, val);
    }
}
```

Evolution of "Unsafe" Usage

- ## ConcurrentLinkedQueue.Node in Java 7 and 8

```java
private static class Node<E> {
  volatile E item;
  volatile Node<E> next;

  boolean casItem(E cmp, E val) {
    return UNSAFE.compareAndSwapObject(this, itemOffset, cmp, val);
  }

  void lazySetNext(Node<E> val) {
    UNSAFE.putOrderedObject(this, nextOffset, val);
  }

  boolean casNext(Node<E> cmp, Node<E> val) {
    return UNSAFE.compareAndSwapObject(this, nextOffset, cmp, val);
  }
  // but there's more ...
```

Huh?

Pointer Arithmetic Into Memory Location

```java
private static final sun.misc.Unsafe UNSAFE;
private static final long itemOffset;
private static final long nextOffset;

static {
  try {
    UNSAFE = sun.misc.Unsafe.getUnsafe();
    Class k = Node.class;
    itemOffset = UNSAFE.objectFieldOffset(k.getDeclaredField("item"));
    nextOffset = UNSAFE.objectFieldOffset(k.getDeclaredField("next"));
  } catch (Exception e) {
    throw new Error(e);
  }
}
```

java.util.concurrent.* Cleaned Up

- **ConcurrentLinkedQueue.Node in Java 9**

```java
static final class Node<E> {
  volatile E item;
  volatile Node<E> next;

  boolean casItem(E cmp, E val) {
    return ITEM.compareAndSet(this, cmp, val);
  }
}
static final VarHandle ITEM;
static {
  try {
    MethodHandles.Lookup l = MethodHandles.lookup();
    ITEM = l.findVarHandle(Node.class, "item", Object.class);
  } catch (ReflectiveOperationException e) {
    throw new Error(e);
  }
}
```

Unsafe in Java 7

	Java 7
java.io.*	**1**
java.lang.**	**7**
java.math.*	**1**
java.net.*	**3**
java.nio.*	**28**
java.util.*	**1**
java.util.concurrent.**	**29**
java.util.zip.*	**0**
Total	**70**

javaspecialists.eu

Unsafe in Java 7, 8

	Java 7	Java 8
java.io.*	1	2
java.lang.**	7	5
java.math.*	1	2
java.net.*	3	3
java.nio.*	28	28
java.util.*	1	1
java.util.concurrent.**	29	36
java.util.zip.*	0	0
Total	70	77

javaspecialists.eu

Unsafe in Java 7, 8, 9

	Java 7	Java 8	Java 9
java.io.*	**1**	**2**	**3**
java.lang.*	**7**	**5**	**29**
java.math.*	**1**	**2**	**2**
java.net.*	**3**	**3**	**3**
java.nio.*	**28**	**28**	**67**
java.util.*	**1**	**1**	**2**
java.util.concurrent.	**29**	**36**	**8**
java.util.zip.*	**0**	**0**	**1**
Total	**70**	**77**	**114**

Java 9 VarHandles

- **CAS operations on non-final fields**
  - **Varying levels of reads and writes**
    - **get/setPlain (non-volatile field semantics)**
    - **get/setOpaque**
    - **getAcquire/setRelease**
    - **get/setVolatile (volatile field semantics)**
  - **Fences (Full, Acquire/Release, LoadLoad, StoreStore)**
- **http://gee.cs.oswego.edu/dl/html/j9mm.html**

Quick Striped64 Tutorial

- **Supports LongAdder and LongAccumulator**
  - **Creates Cell[] to hold values**
    - **Expands on each CAS failure**
    - **Maximum Cell[].length bound by Runtime.availableProcessors()**
    - **Cell objects are marked with @Contended, so very large**
  - **Threads are allocated permanently to a particular Cell**
    - **How?**

ThreadLocalRandom

- **Introduced in Java 7 to improve on Math.random()**
  - **Thread-local pseudo random series**
  - **Very fast**
  - **Initially used ThreadLocal as implementation**
    - **Slow table lookup on every current() call**
- **Java 8 stores ThreadLocalRandom fields inside Thread**
  - **Also protected with @Contended against false sharing**
    - **We'll get back to @Contended later**

javaspecialists.eu

Migration to Java 9

- **Striped64 needs to get direct access to a field in Thread**

  - **We want to avoid Unsafe**

  - **VarHandles typically for fields in our class**

    - **includes inner classes**

javaspecialists.eu

Expires End Of This Talk

- **Free license to latest mini-course on Java.NIO using Java 9**

- **Free subscription to The Java Specialists Newsletter**

- **http://tinyurl.com/javaday17**

Demo of Changing String with VarHandles

● **MethodHandles.privateLookupIn() FTW**

javaspecialists.eu

Turning Off Those Annoying Warnings

- **Who likes seeing these?**

```
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access using Lookup on ChangeString
        to class java.lang.String
WARNING: Please consider reporting this to the maintainers of ChangeString
WARNING: Use --illegal-access=warn to enable warnings of further illegal
        reflective access operations
WARNING: All illegal access operations will be denied in a future release
```

Warnings Be Gone!

- **java --add-opens java.base/java.lang=ALL-UNNAMED ...**

javaspecialists.eu

MappedByteBuffer Unmapping

- **To map a file into memory, do this:**

```
RandomAccessFile raf = new RandomAccessFile(filename, "rw");
FileChannel fc = raf.getChannel();
MappedByteBuffer buf = fc.map(
    FileChannel.MapMode.READ_WRITE, offset, length);
```

- **To unmap the file, wait for buf to be eligible for GC**

  – **A PhantomReference called Cleaner will release the native memory**

- **Demo of how to unmap in Java 7/8 and now in 9**

javaspecialists.eu

javaspecialists.eu

JVisualVM Gone in Java 9

- **We still have jcmd, jconsole, jstat, jmap, jstack**

- **VisualVM is available separately on Github**
  - **https://visualvm.github.io/download.html**

- **Oh, also -Xrunhprof gone**

  - **Although -Xprof remains for now**

  - **But we will have Java Flight Recorder / Java Mission Control soon**
    - **Plan for this to be open sourced as soon as possible**

Making Our Own Fields @Contended

```java
import jdk.internal.vm.annotation.Contended;

public class HighlyContended {
  private int before;
  @Contended
  private volatile int value;
  private int after;


  public int getValue() {
    return value;
  }


  public void setValue(int value) {
    this.value = value;
  }
}
```

Compiling Causes Errors

```
heinz$ javac HighlyContended.java
HighlyContended.java:1: error: package
jdk.internal.vm.annotation is not visible
import jdk.internal.vm.annotation.*;
                      ^

  (package jdk.internal.vm.annotation is declared in module
java.base, which does not export it to the unnamed module)
1 error
```

javaspecialists.eu

--add-exports To The Rescue

- **We compile like so**

```
javac --add-exports java.base/jdk.internal.vm.annotation=ALL-UNNAMED  \
    HighlyContended.java
```

- **And run with**

```
java -XX:-RestrictContended HighlyContendedTest
```

--add-opens vs --add-exports

- **--add-opens allows "deep reflection" on elements**

- **--add-exports allows access to public classes, methods and fields**

  – **--add-opens implies --add-exports**

  – **Both will be removed in a future version of Java (maybe)**

Javaspecialists.eu

Java 9 Stream Changes

- **Stream.takeWhile(Predicate)**

```
IntStream.generate(() -> ThreadLocalRandom.current()
        .nextInt(0, 50))
    .takeWhile(i -> i < 45)
    .forEach(System.out::println);
```

- **Stream.dropWhile(Predicate)**

- **Stream.iterate(seed, Predicate, UnaryOperator)**

```
IntStream.iterate(0, i -> i <= 30, i -> i + 2)
    .forEach(System.out::println);
```

- **Don't use with parallel streams!**

javaspecialists.eu

jshell For Scripting

- **A REPL to make Java more accessible to novices**

- **Quick Demo**

- **Tips:**

  - **Let EDITOR environment variable point to your favourite editor**

  - **Startup is slow, as all the referenced classes are compiled for scripting. Once it's warmed up, execution is Java speed**

Hacking jshell For Unix Style Scripting

- **Start by defining a file exit.jsh that contains one line "/exit"**

- **First line in your file is this**

```
//$JAVA_HOME/bin/jshell --execution local --startup DEFAULT PRINTING $0 $@ exit.jsh; exit
```

- **Notes**

  - **Any parameters you pass in are interpreted as script names**

  - **jshell *always* returns 0, even with System.exit(1)**

    - **--execution local makes System.exit(val) return val**

- **Thanks to Christian Stein for discussions around this**

javaspecialists.eu

GC Changes

- **Java 8 deprecated incremental CMS**
  - **Removed in Java 9**

- **Java 9 deprecated CMS**

- **Default GC is now G1GC instead of Parallel Throughput**
  - **G1GC is child's play to configure**
    - **Set maximum pause time**
    - **Set maximum heap**
    - **You're done!**

Hacking BigInteger

- **Let's say we want to make square() public**

  - Java 8, we could patch our version in using -Xbootclasspath

  - Java 9, we need to create a patch for java.base

Hacking BigInteger

- **Step 1: Compile the hacked BigInteger version**

```
javac --patch-module java.base=BigMathHack/src   \
    -d BigMathHack/mypatches/java.base BigMathHack/src/java/math/BigInteger.java
```

- **– (Optional) Dependency in IntelliJ to shut off compiler warnings**

- **Step 2: Compile our code against the patched BigInteger**

```
javac --patch-module java.base=BigMathHack/mypatches/java.base   \
    -d out/production/FootShootJava9 --source-path src src/BigIntegerTest.java
```

- **Step 3: Run our code against the patched BigInteger**

```
java --patch-module java.base=BigMathHack/mypatches/java.base   \
  -cp out/production/FootShootJava9 BigIntegerTest
```

javaspecialists.eu

Questions?

Dr Heinz M. Kabutz
Email: heinz@javaspecialists.eu
Twitter: @heinzkabutz

Javaspecialists.eu
java training

Expires End Of This Talk

- **Free license to latest mini-course on Java.NIO using Java 9**

- **Free subscription to The Java Specialists Newsletter**

- **http://tinyurl.com/javaday17**

javaspecialists.eu