Title: Public sun.misc.Unsafe Replacement API
Author: Peter Lawrey, Christoph Engelbert
Organization: Higher Frequency Trading Ltd (UK), Hazelcast, Inc.
Owner: Peter Lawrey
Created: 2014/01/21
Type: Feature
State: Draft
Exposure: Open
Component: core/libs
Scope: SE
JSR: TBD
Discussion: core-libs-dev@openjdk.java.net
Start: 2014/Q2
Depends: 193
Blocks:
Effort: M (less than 6 months but more than 3 months)
Duration: L (less than 1 year but more than 6 months)
Template: 1.0
Internal-refs:
Reviewed-by:
Endorsed-by:
Funded-by: Higher Frequency Trading Ltd, Christoph Engelbert

Summary

This JEP is about to create a public API replacements for sun.misc.Unsafe to prevent people from accessing a private package in form of a direct memory kind of buffer and a support class for other *sun.misc.Unsafe* operations independently from buffer like memory operations.
Additionally it is about to add small (maybe optional) things like bounding checks to different kinds of methods for compliance to other Java features and meet programmers expectations on public Java APIs.

Goals

- Remove the need for direct access to internal classes on commonly used features
- Standardization of the resulting API as part of the JCP process into a JSR (target Java 9)
- Meeting security and programmers expectations on using off heap allocations

Non-Goals

No support for deprecated methods, nor *sun.misc.Unsafe* methods not already implemented. Also features that are already available in the Java API are not taken into account like *monitor*Enter or *monitorExit* which are supported through *java.util.concurrent.locks.LockSupport*.

Success Metrics

Proving a clean supported API that is capable of solving the same problems *sun.misc.Unsafe* is normally used for.
It needs to be a mostly full replacement for the direct access of *sun.misc.Unsafe* but is not meant to be seen as a drop-in replacement so API is about to change and security considerations are taken into account.

Motivation
---
sun.misc.Unsafe is currently the only means of building large, thread safe off heap and lock-free data structures. This is useful for minimizing GC and memory fence overhead, sharing memory between processes and implementing embedded databases without having to use C and JNI, which is likely to be slower and less portable.

Description

Provide a wrapper class for off heap memory and direct memory operations like *java.nio.ByteBuffer* but with the following differences / enhancements:
  - 64-bit sizes and offsets
  - Thread safe constructs such as volatile and ordered access, CAS operations maybe by providing the same access principle as discussed in JEP 193
  - JVM optimized bounds checking, or developer control over bounds checking
  - The ability to reuse a slice of buffer for different records within a buffer
  - The ability to map an off heap data structure to such a buffer in such a way that bounds checking is optimized away

Provide a support class for other memory operations supported by *sun.misc.Unsafe*:
  - Non Constructor calling object allocations
  - Array / object based operations like arrayBaseOffset, arrayBaseScale, objectFieldOffset, staticFieldOffset, staticFieldBase, etc.
  - CopyMemory operations, maybe also for objects and not only for arrays
  - Unsafe class defining and ClassLoader injection
  - Get an objects / arrays memory address
  - Retrieving the pagesize, throwing sneaky exception, force class initialization, etc
  - Explicit fences and adder methods were introduced in Java 8 to *sun.misc.Unsafe*

Key functionality to be retained:
  - Support for memory mapped files
  - Support for NIO APIs
  - Support for writes committed to disk

A proposed package name is not meant to be a typical java namespace (like *java.nio.fs*) but will live in javax namespace to clearly mark it as a special purpose API. There are currently many alternatives discussed like for example *javax.native*, *javax.directio* or *javax.offheap*.

Alternatives

Up to Java 9 direct access to *sun.misc.Unsafe* is the alternative and is heavily used in gaming industry, high frequency trading and other businesses.
There is no alternative in Java 9 like using *sun.misc.Unsafe* directly because Project Jigsaw eventually will prevent access to the internal packages. This will break most of the currently in-the-wild applications with a need of high performance data structures and lock-free implementations and also it will break a lot of applications through their used libraries (especially most serialization libraries).

Testing

Enhancement of the TCK is required to test security access checks and the general behavior of the implementation.
In addition to that tests for different platforms (ARM32, ARM64, x86, x64, ...) and different operating systems (Windows, Linux, Solaris, ...) are needed to make sure the implementation is working in terms of endianess and native code.
Maybe some *sun.misc.Unsafe* tests can be reused for this.

Risks and Assumptions

The idea is to lower the risks of ungracefully JVM shutdowns by enabling range and address checks by default and maybe only activating it on purpose by additions to the Java security system (like new / additional permission checks).
It still is possible that some of the APIs proposed in this JEP might break a Java-programmers understanding of Java just as Lambdas did for Java 8.
At worst case the risks sum up to the same as for using *sun.misc.Unsafe* itself.
It might be possible to bring down the JVM ungracefully by emitting bad memory pointer or any kind of other unsafe operations.

Dependences

Currently there is no direct dependency to any other JEP but there might be a small dependency on JEP 162 in the future due to the modulization of the JRE.

Impact

Other JDK / JRE components that will have access to *sun.misc.Unsafe* after introduction of Jigsaw must not be impacted but may want to change to the new API afterwards.

Security objections are unchanged to current situation when no *java.lang.SecurityManager* is installed in the system that blocks reflective access to *sun.misc.Unsafe*. As mentioned above, security is a clearly known problem for all kinds of *sun.misc.Unsafe* like APIs and we will have to come up with a factory implementation and security check to prevent access to this new API or to give access only to a subset.