Minimal Value Types

Welcome to the Minimal Value Types early adopter's project !

What is the Minimal Value Types (MVT) project?

- The Minimal Value Types project is an early prototype for Value Types
    - provides initial subset of Value Type functionality
    - provides a new type which is:
        - immutable, identity-agnostic, non-nullable, non-synchronizable, final
        - does not inherit from java.lang.Object

    - Value Types contained in References, other Value Types or in Arrays are flattenable
    - Value Types can contain primitives or references

Target Audience

- Power users - Java/JVM Language, Framework, Library authors/exports
    - who are comfortable with early experimental software
    - who recognize that everything in the experiment - the model, the classfile extensions, the byte codes is likely to change
    - who want to contribute to early exploration of Value Types
    - who will not build any products based on these prototypes
- Who are willing to provide feedback to the developers on a subset of Value Type features
- Who will provide use cases for the development team to experiment with optimizations

How to Try Minimal Value Types:

Early Access Binaries

http://jdk.java.net/valhalla/

Repository and Build Instructions

To create a new mvt branch:

```
hg clone http://hg.openjdk.java.net/valhalla/valhalla valhalla-mvt

cd valhalla-mvt

hg defpath du <openjdkname>-

hg update -r mvt // name of branch
```

To update repository:

```
cd valhalla-mvt

hg pull

hg update -r mvt  // name of branch
```

To build repository

```
bash configure

make images
```

http://cr.openjdk.java.net/~chegar/docs/sandbox.html // instructions for working with branch repositories (not yet updated for consolidation)

Note: Valhalla is a child of the jdk10/hs repository, to keep current with latest hotspot development.

Programming Model

- Create a POJO using an experimental javac, with an annotation ValueCapableClass (jvm.internal.value.ValueCapableClass)
    - JVM will derive a Derived Value Class (DVC) from this defined "box" which we call the Value Capable Class (VCC)
    - DVC contains an immutable copy of the instance fields from the VCC
- Work with MethodHandles and experimental reflection package ValueType (jdk.experimental.value.ValueType)
- Or spin your own byte codes
    - experimental MethodBuilder (jdk/experimental/value.MethodHandleBuilder)

        - future: ASM support is planned
- Code samples
    - See directory jdk/test/valhalla/mvt in your repository
        - e.g. MethodHandlesTest

Run Experimental MVT

- **java -Xverify:none -XX:+EnableMVT <Test>**
- Command-line flags: Minimal Value Types Command-line Options

Filing Bugs

- Send email to valhalla-dev@openjdk.java.net

Limitations

- platforms: x64 Linux, x64 Mac OS X
- no VarHandles (work-in-progress for an update)
- no support for atomic fields containing value types (work-in-progress for an update)
- no JNI, unsafe, jvmti, redefineclasses, reflection
- -Xint and C2 only, no C1, no tiered-compilation
- interpreter is not optimized, focus is on JIT optimization
- Need additional USE CASES for optimizations
- Working on additional test cases

References

- Experimental Appendix for Java Virtual Machine Specification for Value Classes
- http://cr.openjdk.java.net/~chegar/docs/sandbox.html // instructions for working with branch repositories

- Minimal Value Types Command-line Options
- Youtube JVMLS 2017: Minimal Value Types: Origins and Programming Model
- Youtube JVMLS 2017: Minimal Value Types: Under the Hood