LW2

LW2 is an iteration of a previous prototype, adding further language support and JDK API support for "inline types" (formerly referred to as "value types"). There are also subtle type system, runtime and JIT changes

Javac source support:

- Requires source level >= JDK14 (**since LW2**, the prototype is based on mainline JDK, whose version string is JDK 14 at the time of writing)
- A class declaration of inline type is made by using the "inline class" modifier, or (for IDE syntax parsing convenience) with "@__inline__" annotation (**since LW2**)
    - Interfaces, annotation types, enums can not be inline types
    - Top level, inner, nested, local classes may be inline types. Inline types may declare inner, nested, local types
    - Inline Types are implicitly final, so cannot be abstract
- Inline Types may not declare an explicit super class (except Object). They implicitly extend java.lang.Object akin to enums, annotation types and interfaces
- Inline Types may declare explicit interfaces
- Inline Types constructors may not pass the "this" handle until all instance fields are definitely assigned.
- All instance fields of an inline class are implicitly final
- "Indirect" projections of inline types via the "?" operator (**since LW2**)
    - E.g. the projection of Point is Point?
    - Point? IS also an inline type which allows null for backward compatibility
    - maintains type of a field, array element, or generic parametric argument, but they may be null
    - Point is a subtype of Point?
    - cast between inline type and indirect type
    - Is implicitly a "NullableType"
    - enables compatibility with current generics via erasure to null-object object references
    - Arrays subtyping relationships: InlineType[] <: IndirectType[] <: Object[] <: Object
        - Widening conversions may be carried out from in the order above
        - Narrowing from an array of IndirectType to an InlineType is not possible.
        - This array covariance enables compatibility with existing API, especially generic type erasure
- Inline types may not declare fields of its own type directly. If A and B are inline classes, A may not contain a field B if B contains a field A.
    - There is a method to do this "indirectly" using the "?" operator, but it's value is still final. (**since LW2**
- java.lang.Object methods:
    - javac automatically generates hashCode, equals, and toString computed solely from the instance's state and not from its identity
    - javac does not clone(), finalize(), wait*), notify*() on inline type receivers
- javac allows comparison of inline type using ==, != (**since LW2**)
    - By default this implies a test in the instance state (i.e. field by field "substitutability" test), as inline types have no identity (i.e. not a reference test).
- Inline Types can not be assigned null, null can not be cast to or compared with inline types
    - With the exception of an "indirect" reference using the "?" operator. (**since LW2**)
- Indirect Types cannot be type arguments in generic type parameterizations, type witnesses in generic method invocations, wildcard bounds
    - With the exception of an "indirect" reference using the "?" operator. (**since LW2**)
- Type migration, including partial recompilation is not supported.

Java APIs:

- Class new /modified API:
    - isInlineClass()
    - asPrimaryType()
    - asIndirectType() / isIndirectType()
    - asNullableType() / isNullableType()
    - getName() reflects the Q or L type signatures for arrays of inline types (**since LW2**)
        - In the .class file, the null-free inline type today is represented by a Q type signature, and the indirect projection is represented by an L type descriptor
    - newInstance() on an inline type throws NoSuchMethodException
- setAccessible() on an inline type throws InaccessibleObjectException
- Initial core Reflection and VarHandles support in place (**since LW2**)
- Attempts to build Reference objects with inline type will result in IllegalArgumentException

Runtime:

- IllegalMonitorException thrown on attempts to synchronize or call wait(*) or notify*() on an inline type
- ClassCircularityError thrown if loading an instance field of an inline type which declares its own type either directly
- Attempts to serialize an inline type will throw NotSerializableException
- Cast from indirect type to inline type may result in NullPointerException (**since LW2**)
- Attempts to store "null" into Inline type array will throw NullPointerException
    - Whereas null into indirect type array works as normal (**since LW2**)

Limitations for LW2

This is still a prototype with a lot of components ignored.

- platforms: x64 Linux, x64 Mac OS X, and x64 Windows
- no support for atomic fields containing indirect types
- no support for @Contended inline type fields
- no AOT, CDS, ZGC, serviceability agent, JVMTI, limited JNI
- -Xint and C2 only, no C1, no tiered-compilation, no Graal

- unsafe field and array accessor APIs are not supported for inline types
    - low-level unsafe APIs are UNSAFE and will not be changed to support inline types
    - risks: If an inline type has been flattened in a container, unsafe does not know the layout
        - getObject could return the first flattened element rather than the expected reference
- interpreter is not optimized, focus is on JIT optimization
- Some major changes to JIT compiler are still a work in progress

Future Possibilities

- LW2 updates:
    - fix bugs, add optimizations and support for additional minor features
    - experimenting further with "=="
- LWX:
    - Addition of significant feature support
    - Addition of language level syntax and semantics
    - Eventually a preview with an openjdk release
    - Expect specialized generics and support for primitives as inline types will have their own early access cycles

How to Try L-World Inline Types

Target Audience

- Power users - Java/JVM Language, Framework, Library authors/experts
    - 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 Inline Types
    - who will not build any products based on these prototypes
- Who are willing to provide feedback to the developers on a subset of Inline Type features
- Who will provide use cases for the development team to experiment with optimizations

Early Access Binaries

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

Change log:

- jdk-14-valhalla+2-18 (20190704)
    - Minor bug fixes
- jdk-14-valhalla+1-8 (20190628)
    - Initial LW2 prototype

Repository and Build Instructions

To create a new local repository, based on "lworld" branch:

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

cd valhalla-lworld

hg defpath du <openjdkname>

hg update -r lw2 // name of branch
```

To update repository:

```
cd valhalla-lworld

hg pull

hg update -r lw2  // name of branch
```

To build repository

```
bash configure

make images
```

Instructions for working with branch repositories: http://cr.openjdk.java.net/~chegar/docs/sandbox.html

Note: Valhalla is a child of the jdk/jdk repository, to keep current with latest OpenJDK development.

Programming Model

```
public inline class InlineType implements Comparable<InlineType?> {

    int someValue;

    public InlineType(int value) {
        this.someValue = value;
        /*
          implicitly generates "defaultvalue" followed by "withfield" bytecodes

          cannot pass "this" from here
         */
    }

    // Since existing API can accept "null", we use the indirect projection
    public int compareTo(InlineType? other) {
        if (other == null) {
            return -1;
        }
        return someValue - other.someValue;
    }

    public static void main(String[] args) throws Throwable {
        InlineType def = InlineType.default;
        assert(def == new InlineType(0));

        InlineType[] inlineTypeArray = new InlineType[1];
        // Inline type array cannot store null
        assert(inlineTypeArray[0] == def);

        InlineType?[] indirectTypeArray = new InlineType?[1];
        // Indirect projection of Inline type array can store null
        assert(indirectTypeArray[0] == null);
    }
}
```

A note on "?" operator and indirect projections

This is really only the beginning of a compatibility story, there is more to come (e.g. "null-default inline types"). Today it is a stop gap measure to enable use of current generics, until we get real generic specialization in place. New code should avoid indirect "?" like the plague and only use it for interfacing with code that must deal with "null" (e.g. current generics with type erasure). A good performance story does not exist for indirect, and it is not likely to improve too much in the future.

Run Experimental L-World

- **java <Test>**
- Further experimental flags can be found here

Helpful Feedback Please

This is intended as an early prototype to give you a chance to experiment and provide feedback on the currently supported features.

Please ensure you are on the latest EA binaries before reporting a problem.

Bugs are tracked in JIRA: Start summary with /[lworld/] and label "lworld". You can search for known problems or already reported bugs.

- Known bugs

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

- Use cases that worked for you, including any positive performance or footprint information
- Bugs or questions about surprise behavior
- We have performance holes, if you have a problem, please provide a use case for us to consider

Too Early for Feedback

This is intended as an early prototype to give you a chance to experiment and provide feedback on the currently supported features.

At this time it would premature to provide feedback on

- Language syntax

- Problems already listed under Limitations

References

- Draft Java Virtual Machine Specification Inline Type changes