L-World

Welcome to the L-World early adopter's project !

What are Inline Types?

- Formerly known as "Value Types", renamed "inline-types" (discussion here)
- Inline Types are small, immutable, identity-less types
- User model: "codes like a class, works like an int"
- Use cases: Numerics, algebraic data types, tuples, cursors, ...
- Removing identity commitment enables optimizations such as
    - flattening of inline types in containers such as fields or arrays
        - reducing cost of indirection and locality of reference with attendant cache miss penalties
        - reducing memory footprint and load on garbage collectors
- Combining Immutability and no identity commitment allows inline types to be stored in registers or stack or passed by value

What is the L-World project?

The L-World project is a series of early prototypes for bring Inline Types to the language and JDK.

- builds on work of the previous Minimal Values Types prototype (MVT)
    - provides a new type which is: immutable, identity-agnostic, non-nullable, non-synchronizable, final
    - Inline Types contained in References, other Inline Types or in Arrays are flatten-able
    - Inline Types can contain primitives or references
    - JVMS class file model for MVT:
        - Separate descriptors to distinguish inline types from object types using "Q" signatures (Q-Types) similar to how Object descriptors begin with "L" (L-Types).
        - Separate bytecodes, starting with "v", called "v-bytecodes" to distinguish from reference "a-bytecodes".
- MVT limitations:
    - No direct support for methods
    - No compatibility with existing Objects and Interfaces
- Enabled exploration of potential maximum optimizations for inline types with minimal impact on existing objects (object identity types)

"L-World"

To maximize backward compatibility with existing Objects and interfaces, i.e. existing L-Types, the current prototype incorporates inline types into the L-Type system or "L-World".

- Inline Types may be referred to by the same "L-Type" descriptors the VM has always operated on:
    - May implement interfaces with inline types
    - May pass a inline type as a java.lang.Object, or an interface through existing APIs
- Inline Type characteristics:
    - immutable: unmodifiable instance fields
        - may contain primitives, other inline types, references to mutable objects
    - identity-less:
        - synchronization including use of wait(*), notify*() will fail with exception: IllegalMonitorStateException
        - reference equality with "==" (`if_acmp<eq|ne>`) performs a substitutability test, comparing each field (much like carry out "equals()")
        - freely substitutable when equal, no visible change in behavior if equals()
    - final
- JVMS class file model for "LWorld"
    - Re-uses "L" descriptors
    - Re-uses "a-bytecodes"
    - There are only two new byte-codes, otherwise existing byte-codes have been engineered to accept and maintain inline type characteristics (identity-less, flattenable, pass by value):
        - "defaultvalue" - will create a new default inline type
        - "withfield"- allows updating inline type fields via a copy-on-write semantic, i.e. new value based on the old value combined with new field value.

Recognize that the path to Valhalla is long, there are number of open issues facing inline types. We wish to solve these incrementally. Fully generic specialization of inline types with clear and sensible migration rules are going to take more than a single prototype.

Prototypes

- ~~**"LW1" - Minimal L-World**~~
- **"LW2" - Inline types and indirect projections**