The Zoo of Java Field Types *(Valhalla draft design, adding Extended Primitives)*

**PRIMITIVES** (one quadrant)
- PRIM = ¬ REF, scalar/extended primitive
- field-wise flattenable (like NOID)
- freely copyable (like NOID)
- no identity (like NOID)
- defaults to all-zero bits
- no nulls, no direct recursion
- no construction safety
- no atomicity (tearing)
- no polymorphism

**REFERENCES** (three quadrants)
- REF = ¬ PRIM, all are nullable, recursive
- VALUE OBJECT REF (NOIDs)
  - (includes PRIM boxes)
  - freely copyable
  - field-wise flattenable
  - construction safety
  - atomicity, no identity

**NON-CONCRETE REF**
- polymorphic abstract class or interface

**Island of Classless Scalar Primitives**
- int
- long
- float
- double
- boolean
- char
- byte
- short

**Optional**
- (VBC migrated to value object, a NOID)

**Memory Segment**
- (value object: secure, no box, better VBC)

**List**
- List
- Cursor
- Value Object (marker)

**Point**
- its box

**List Cursor**
- its box

**VALUE OBJECT REF (NOIDs)**
- (includes PRIM boxes)
- freely copyable
- field-wise flattenable
- construction safety
- atomicity, no identity

**THIS MIDDLE SWATH IS WHAT’S NEW IN VALHALLA**

**Identity Object (marker)**

**Object**

**Iterator**

**Abstract List**
- (partially concrete)

**HashMap**
- (mutable object)

**String**
- (identity significant)

**int []**
- (mutable array)

**LocalDate**
- (VBC = value based class, just before migration)

**Island of Classless Scalar Primitives**
- their boxes
The Zoo of Java Field Types *(before Valhalla)*

<table>
<thead>
<tr>
<th>PRIMITIVES (one quadrant)</th>
<th>REFERENCES (three quadrants)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRIM = ¬ REF, scalar primitives only</td>
<td>REF = ¬ PRIM, all are nullable, recursive</td>
</tr>
</tbody>
</table>

- Field-wise flattenable
- Freely copyable
- No identity
- Defaults to all-zero bits
- No nulls
- Uncertain 64-bit atomicity
- No polymorphism

+ int long
+ float double
+ boolean char
+ byte short

**Integer**
(VBC = value based class)

**Long, etc.**
(primitive boxes)

**Optional**
(VBC = value based class)

**Memory Segment**
(VBC = value based class)

**LocalDate**
(VBC = value based class)

**Non-concrete ref**
Polymorphic abstract class or interface

**List**

**Object**

**Iterator**

**Concrete object refs**
Object identity
Construction safety
Fields stay together
Fast pointer compare

**Abstract List**
(partially concrete)

**Hashmap**
(mutable object)

**int[]**
(mutable array)

**String**
(identity significant)