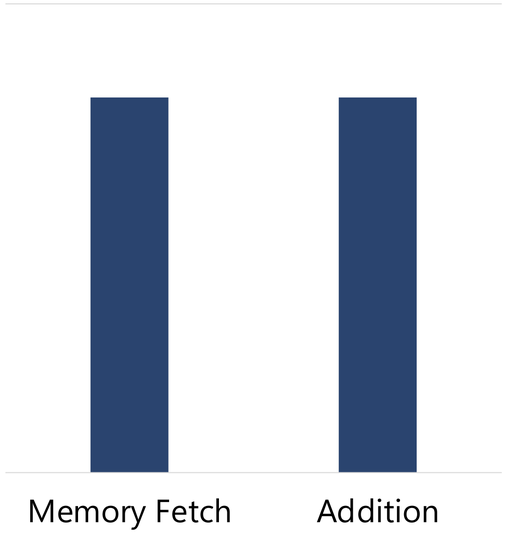
Microsoft

Cache Me If You Can

Speed Up Your JVM With Project Valhalla

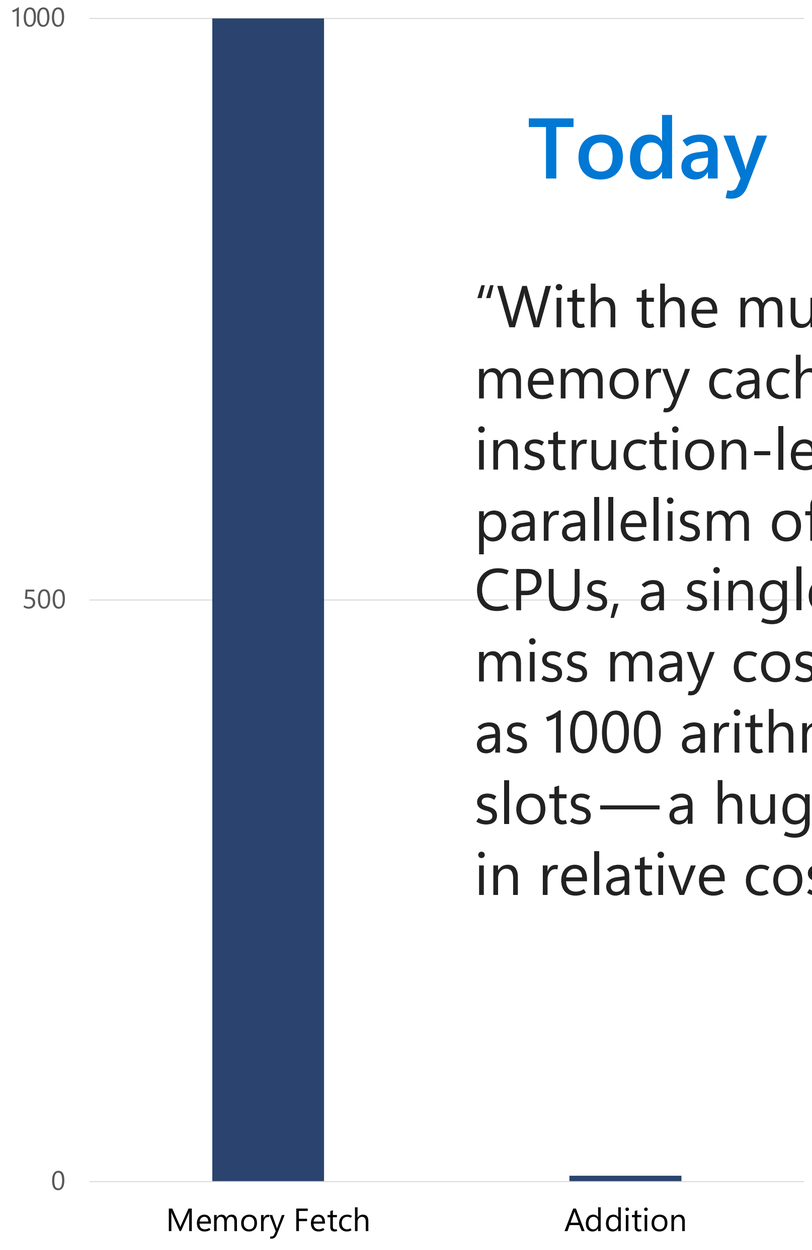
Early Days of Java

"When the Java Virtual Machine was being designed in the early 1990s, the cost of a memory fetch was comparable in magnitude to computational operations such as addition. "



Today

"With the multi-level memory caches and instruction-level parallelism of today's CPUs, a single cache miss may cost as much as 1000 arithmetic issue slots—a huge increase in relative cost."

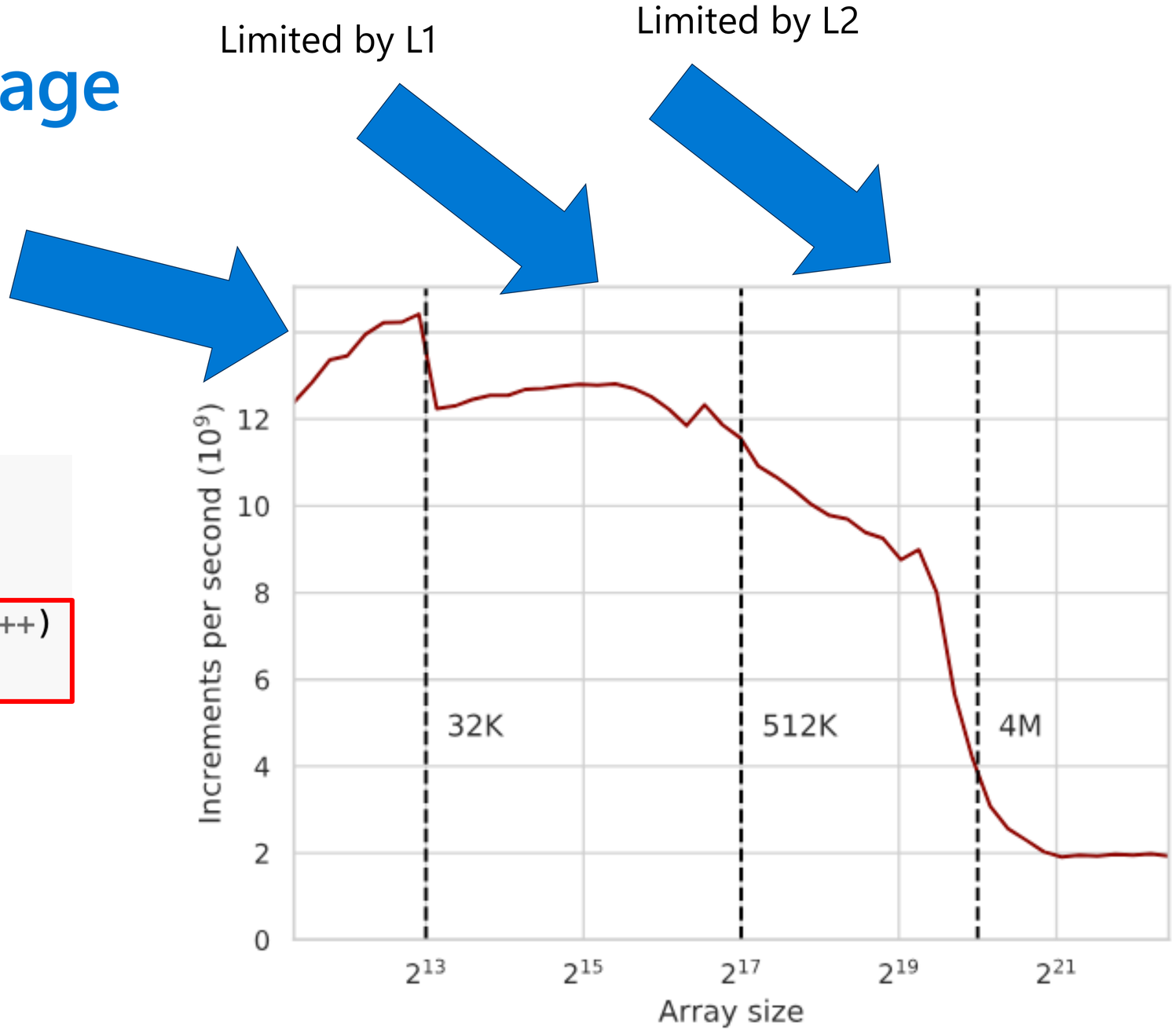


openjdk.org/projects/valhalla/design-notes/state-of-valhalla/01-background

Performance Sabotage

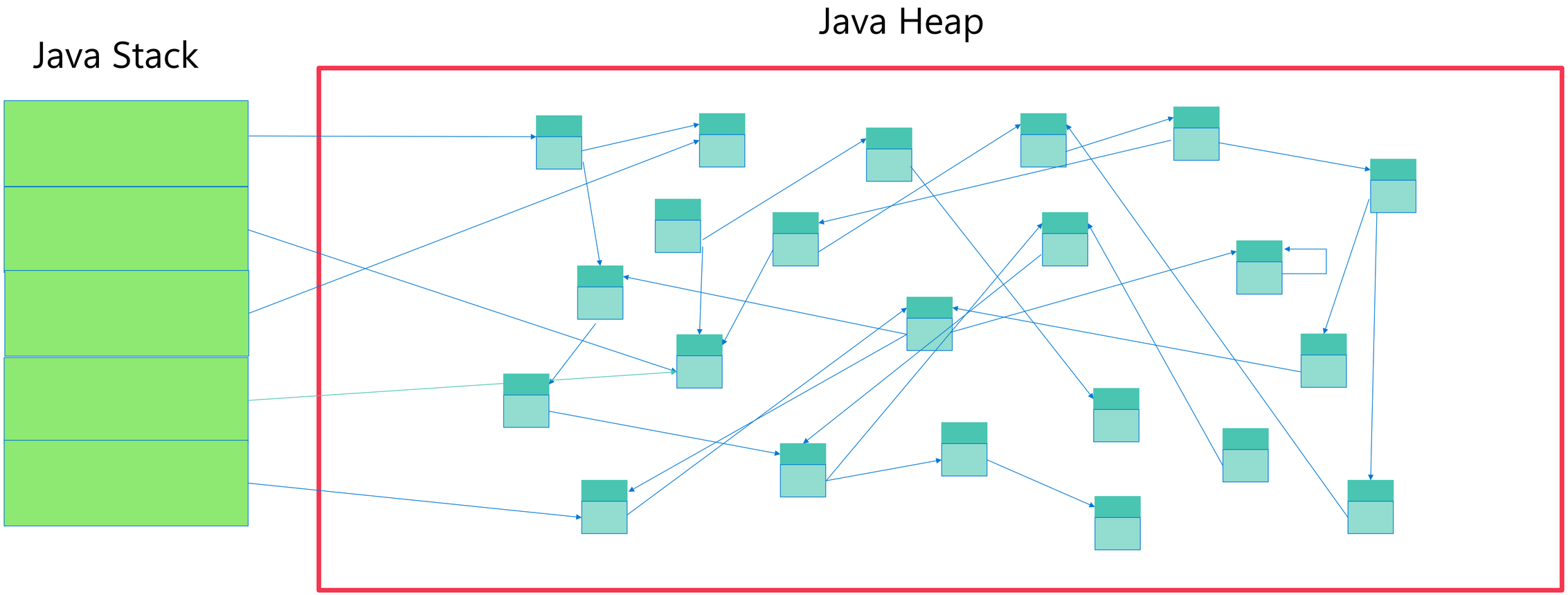
```
int a[N];

for (int t = 0; t < K; t++)
    for (int i = 0; i < N; i++)
        a[i]++;
```



en.algorithmica.org/hpc/cpu-cache/bandwidth/

Object References in the JVM



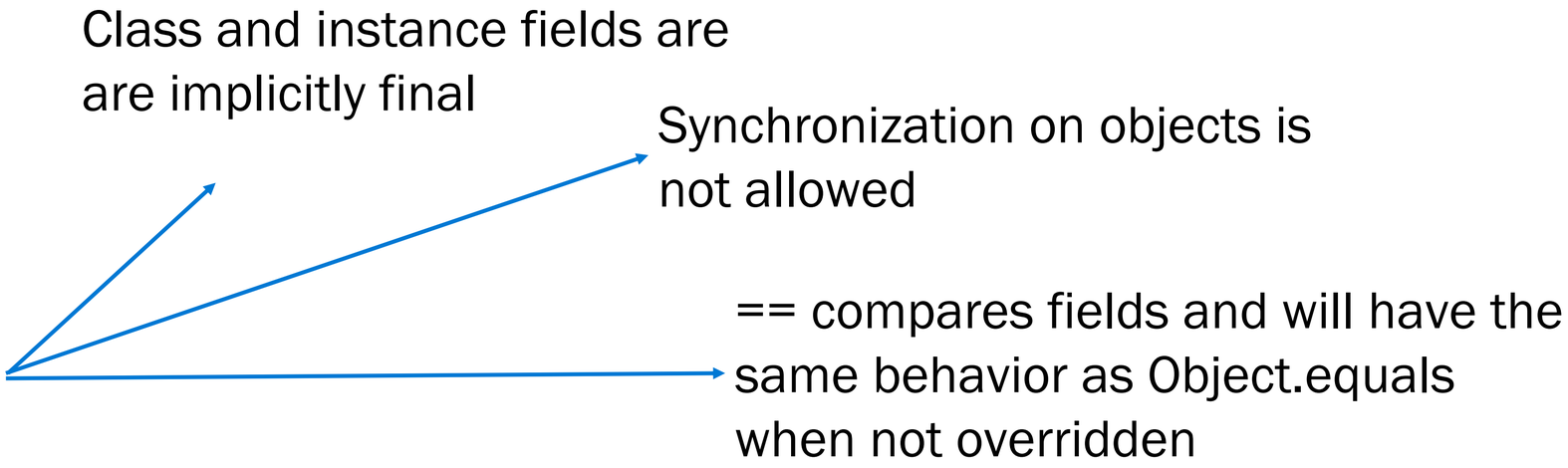
Identity

- Java objects are unique
- Think of == vs Object.equals()
- Allows for field mutability and synchronization
- In a Valhalla world these are known as “Identity Class” or “Identity Object”

openjdk.org/jeps/401

Value Class

- A class without identity
- Nullable
- Accessed atomically



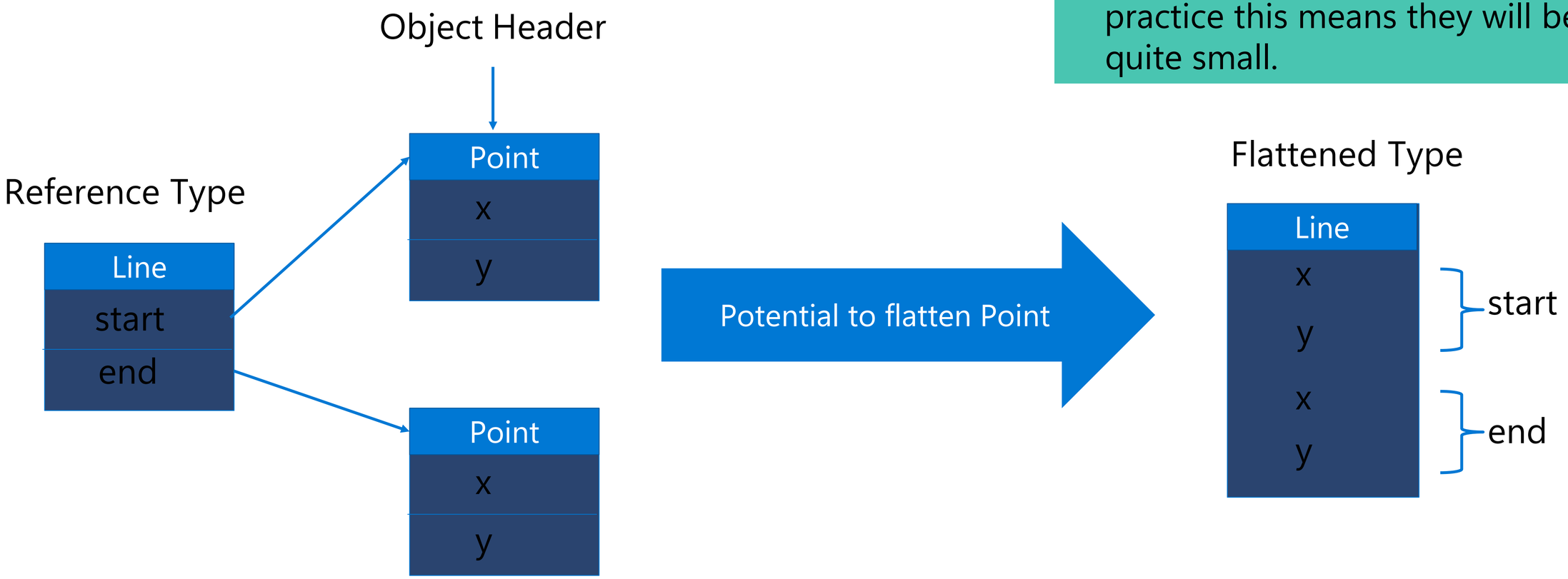

```
value class Line2D {
    Point2D st;
    Point2D en;
    ...
}

value class Point2D {
    int x;
    int y;
    ...
}
```

openjdk.org/jeps/401

Flattening Value Objects

- Flattening is not guaranteed, especially for types > 32 bits
- Objects must be accessed atomically to be flattened… in practice this means they will be quite small.



Null-Restricted Value Class

- Value classes with fields that cannot be null
- Opt-in to automatic creation of default field values
- Allows for flattening of larger value classes

```
value class Line2D {
    Point2D! st;
    Point2D! en;
    ...
}

value class Point2D {
    int x;
    int y;

    public implicit Point2D();
}
```

openjdk.org/jeps/8316779

Even Bigger Flattened Classes

- By default flattened value types need to be small enough to read and write atomically
- Non-volatile primitives (long and double) do not have these restrictions in Java
- Developers can opt-in to to remove this rule for programs that can tolerate non-atomic object access

```
value class Line2D {
    Point2D! st;
    Point2D! en;
    ...
}

value class Point2D
  implements LooselyConsistentValue
{
    int x;
    int y;

    public implicit Point2D();
}
```

openjdk.org/jeps/8316779

Get Involved

> We welcome input from interested Java developers. Keep in mind that most theoretical ideas have been well explored over the last few years! The greatest help can be provided by those who try out concrete prototypes and can share their experiences with real-world code bases.

OpenJDK Early Access Builds

https://jdk.java.net/valhalla/

Build OpenJ9 With Value Types Enabled

```
git clone https://github.com/ibmruntimes/openj9-openjdk-jdk.valuetypes.git
cd openj9-openjdk-jdk.valuetypes
bash ./get_source.sh -openj9-repo=<url> -openj9-branch=<name>
bash ./configure --with-boot-jdk=<jdkpath> --enable-inline-types
make images
```

OpenJ9 Build Instructions: github.com/eclipse-openj9/openj9/tree/master/doc/build-instructions

Eventually it will be available as part of IBM Semeru Runtimes: ibm.com/semeru-runtimes

Microsoft

Thank you

Theresa Mammarella
@t_mammarella
tmammarella
theresa-m