GOTO Chicago RETURN

Cameron Purdy
Senior Vice President
Oracle

ORACLE

About me

- The last time I spoke at a non-Oracle event was QCon in June 2012
- I used to have fun making up “Top 10” lists that would poke fun at awful technology
- Like C++ … and bad programming practices
- I had hair
- It wasn’t all going grey
- I’ve become a very, very, sad old man
- That’s all I got

“Future of Java”

- It’s been done before:
  - http://medianetwork.oracle.com/video/player/1113279726001
  - (Or: http://tinyurl.com/prm2yt4)
- It would require a legal disclaimer
- George Saab is already speaking at this conference

Introduction

It was a dark and stormy night …

(last night)

Title (Take 1)

The Top 10 Top 10 Presentations I ever did Present

Title (Take 2)

There must be Top 10 ways to leave your employer

(Slip out the back, Jack …)

Title (Take 3)

Top 10 PowerPoint Tricks

Title (Final)

Top 10 Stories from my Illustrious Career ™

#10

Begin with the End in Mind

(see what I did there?)

#9

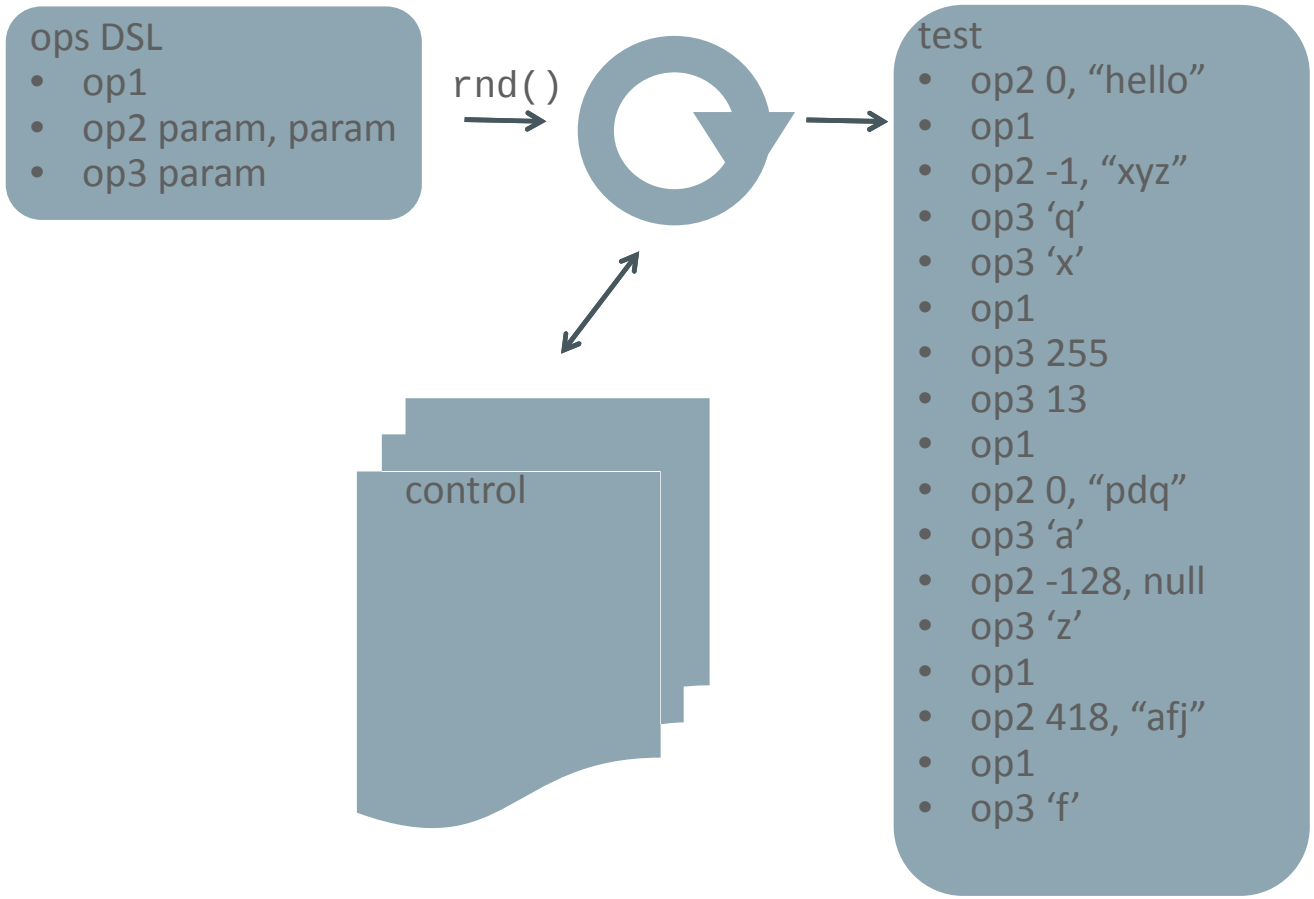
Manage To Your Pain

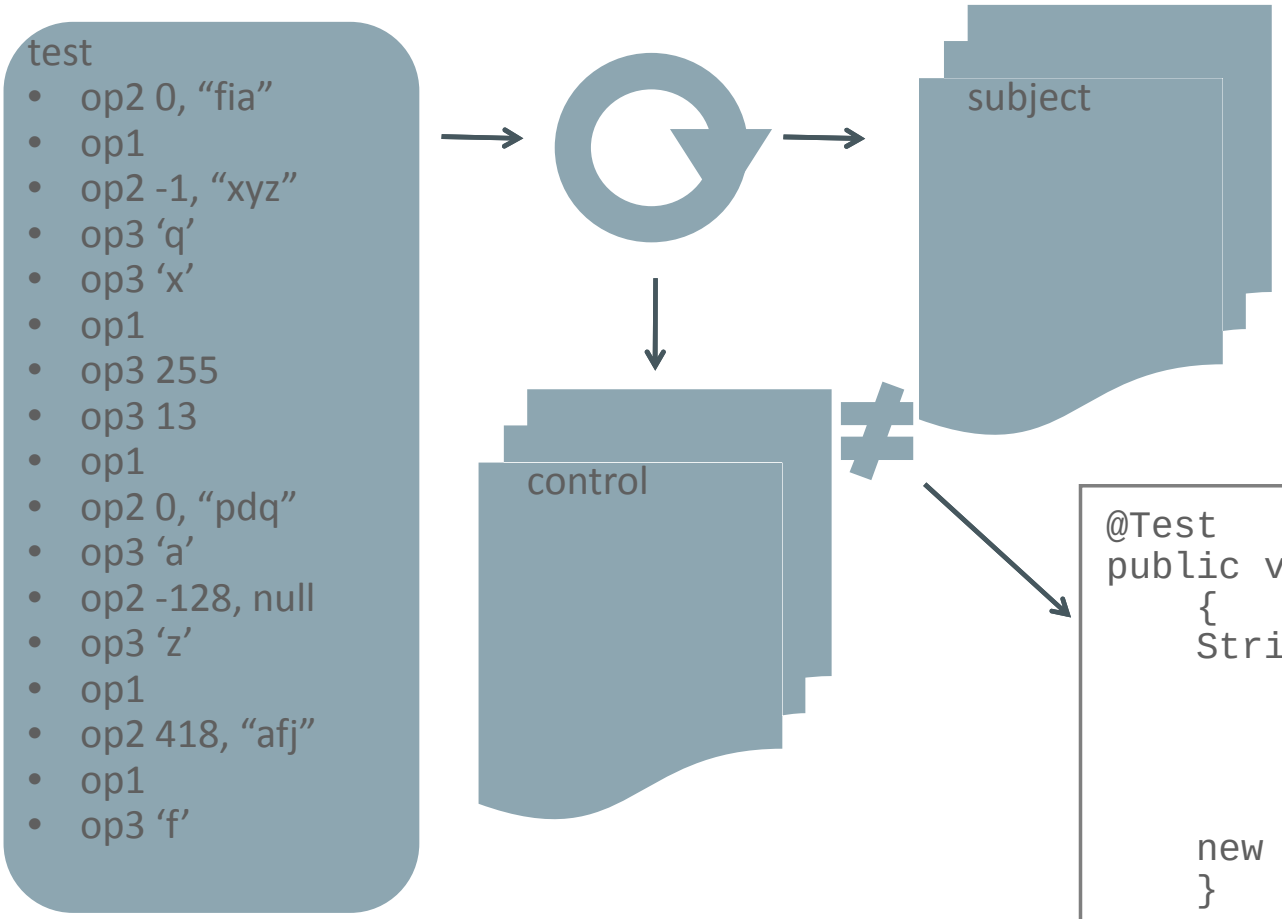
(M2YP)

#8

Random is Awesome

(stochastic is fantastic)






```
@Test
public void test()
    {
    String sTest = ""
            + "op2 0 fia\n"
            + "op1\n"
            + "op2 -1 xyz";

    new TestPlan(sTest).run();
    }
```

#7

Never give in.

(“We have only to persevere to conquer.”)

Elastic Charging Engine

Only 16 CPUs required for charging the 2011 worldwide SMS traffic

Coherence Provides:

- Elastic scalability
  - Handle customer growth dynamically
  - Handle higher load
- Built-in security
  - Built-in secure query with CohQL
- High Throughput and Performance
- Fast Recovery from Hardware or Software Failure
- Very High Availability
  - Online Upgrades and Configuration

Elastic Charging Engine Performance Benchmark Summary	
Servers Used	Exalogic 2-2 3/4 rack
Number of Cores/CPUs	288/48
Busy Hour Events Processed	3 Billion
Throughput for Busy Hour (operations/sec/core)	2,894
99 Percentile Latency	5 ms
Number of Customers in the Grid	400 Million

ORACLE

#6

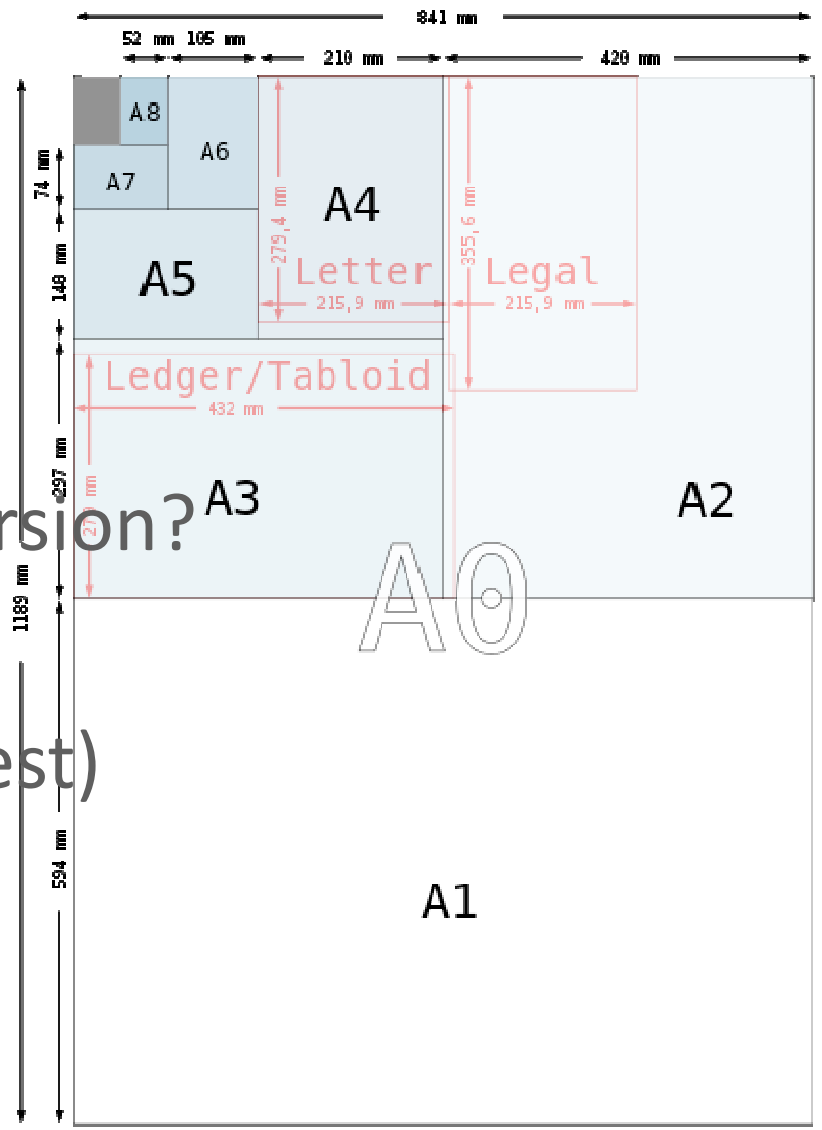
Embrace Criticism, but Require a Better Option

("critic" is only a job on tv)

#5

Does it support recursion?

(the new litmus test)



#4

Inelegant Designs are Almost Always Wrong

(complexity is the enemy)

#3

People and Relationships are Always a Good Investment

(sometimes the investment is you)

#2

The Bark is Worse than the Bite

(imagination is often worse than reality)

#1

Always do what you know to be right

(even when it hurts)

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Value Types

- Working around an “evil design flaw” in Java: Object Identity
- Requires immutability
  - Think of it as a feature, and not a trade-off!
- “Codes like a class; works like an `int`.”
- Provides native language support for tuples
- Might finally get a real `decimal` type – IEEE754-2008 standard!
  - No pun intended
- Perf: eliminates de-refs; smaller footprint; fewer objects=faster GC
- @see http://cr.openjdk.java.net/~jrose/values/values-0.html

Example: Point

```
final __ByValue class Point {
    public final int x;
    public final int y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public boolean equals(Point that) {
        return this.x == that.x && this.y == that.y;
    }
}
```

Non-JDK Example: long values

- HashSet uses almost 600MB of RAM and 7 seconds:
  - RAM=3977272
  - elapsed=6940ms
  - RAM=628839608
- xxxHashSet uses just over 40MB of RAM and 2.5 seconds:
  - RAM=3976960
  - elapsed=2533ms
  - RAM=81787480
- Summary: 15x denser and 3x faster.

Example: `String` values

- Can be implemented in combination with “Arrays 2.0” capabilities
- Fused (1-node) implementation of `java.lang.String`
  - `| header | length | hash | body | → | header2 | char[] |`
- Single allocation
- Single cache line (for short enough `String` values)

Non-JDK Example: `String` values

- 10 million Strings:
- elapsed=19191ms, RAM=140953408 bytes
- elapsed=12459ms, RAM=941366016 bytes

- “Intrinsified” used less than 20 seconds and 140MB
- “Non-intrinsified” used 12.5 seconds and 940MB

- 50% additional time :-(
- 85% less memory :-)

Arrays 2.0

- Support for 64-bit array indexes
  - Current array length limit: 2.1 billion elements

```
interface Array<E> {
  value E getArrayElement(long x) { ... }
  ref   E getArrayElement(long x) { ... } }
```

- Alignment with Value Types
  - Arrays of Value Types can be as efficient as C arrays of structs
  - Cache line optimizations, e.g. B-Tree data structures
- Multi-dimensional arrays, e.g. for vector- and matrix-based calculations
- @see http://cr.openjdk.java.net/~jrose/pres/201207-Arrays-2.pdf

Arrays 2.0 related: GC specialization

- Java GC is optimized for *many* “small” Java objects
- There is basically only one “large” Java class: “Array”
- Until now, extremely large allocations have had to optimize via NIO/Unsafe
  - NIO/Unsafe optimizations for primitives are replaced by Value Types
  - NIO/Unsafe optimizations for large arrays can be replaced by Arrays 2.0
- Arrays can be split into several parts:
  - A header “on the heap”
  - One *or more* storage segments in a separately managed memory area
  - Compaction-based GC (like G1-GC) no longer have to copy the array contents!

We Are the Borg. You Will be Assimilated. Resistance is Futile.™