Java generics...and beyond...

Objectives

- Introduce generics
- Look at what generics lead to
- Java politics

What goes around comes around

- composition → inheritance → composition

- functional programming (lisp)

  → object-oriented programming (java)

  → + functional programming (java8)

Generics

- Java without generics - before Java5
- Storing stuff in a collection – can hold any referenced type

```
List list = new LinkedList();

list.add("abc");  // fine

list.add(new Date());  // fine
```

Java without generics

- What could possibly go wrong?
- Well...reading stuff back from a collection

```
String s = list.get(0);  // compiler error
```

Java without generics

- What could possibly go wrong?
- Well...reading stuff back from a collection

```
String s = list.get(0);  // compiler error
s = (String) list.get(0);  // needs cast
```

Java without generics

- What could possibly go wrong?
- Well...reading stuff back from a collection

```
String s = list.get(0);  // compiler error
s = (String) list.get(0);  // needs cast
Date d = (Date) list.get(1);
d = (String) list.get(0);  // oops again!
```

Java without generics

- What could possibly go wrong?
- Well...reading stuff back from a collection

```
String s = list.get(0);  // compiler error
s = (String) list.get(0);  // needs cast
Date d = (Date) list.get(1);
d = (String) list.get(0);  // oops!
```

- Collection can store any type
- Need to remember what types of objects were stored
- No way to specify the types that a collection should store

Java without generics

- What could possibly go wrong? (cont'd)
- Compiler can be fooled by a cast

```
List list = new LinkedList();

list.add("abc");

list.add(new Date());

String s = (String) list.get(0);

s = (String) list.get(1); //  compiles ok

                // but raises exception at runtime...
```

Java without generics

- What could possibly go wrong? (cont'd)
- Compiler can be fooled by a cast

```
List list = new LinkedList();

list.add("abc");

list.add(new Date());

String s = (String) list.get(0);

s = (String) list.get(1); //  compiles ok

                // but raises exception at runtime...
```

- ...and that's way worse

Generics

- Java with generics – Java5 and up
- Storing and reading stuff in a collection is straightforward

```
// list of Strings

List<String> list = new LinkedList<>();

list.add("abc");  // fine

String s = list.get(0);  // ok
```

Java with generics

- What's been gained?
- Type safety
  - ''In Java, a program is considered type-safe if it compiles without errors and warnings and does not raise any unexpected ClassCastExceptions at runtime.''

    - Angelika Langer
- Compiler detects collection type errors

Java with generics

- What else has been gained?
- See the generics FAQ...all ~~427~~ ~~513~~ ~~297~~ 352 pages!!!
  - Generic types and generic methods...
  - Type parameters, wildcard instantiation...
  - Genericity & inheritance...
  - ...

What people are saying

''I don't know how to ease into this gently. So I'll just spit it out. Generics are a mistake. This is not a problem based on technical disagreements. It's a fundamental language design problem.''

Ken Arnold

- Ken Arnold is the author of *The Java Programming Language*, and he can't understand generics...

What people are saying

“I am completely and totally humbled. Laid low. I realize now that I am simply not smart at all. I made the mistake of thinking that I could understand generics. I simply cannot. I just can't. This is really depressing. It is the first time that I've ever not been able to understand something related to computers, in any domain, anywhere, period.''

What people are saying

''I'm the lead architect here, have a PhD in physics, and have been working daily in Java for 10 years and know it pretty well. The other guy is a very senior enterprise developer (wrote an email system that sends 600 million emails/year with almost no maintenance). If we can't get [generics], it's highly unlikely that the ‘average’ developer will ever in our lifetimes be able to figure this stuff out.''

Java - what happened?

- Initially a pretty simple "blue-collar" language
  - Followed 80 / 20 rule - had 80% of desired features for 20% of the complexity
  - ''When in doubt, leave it out.''
- Then ''feature-envy'' set in prompted by...
  - ...''C# has some feature A and Java doesn't''
  - ...''Ruby, Python, etc. have some feature B and Java doesn't''
  - ...''wouldn't it be cool to add feature C, just because it's cool!''

Java – what happened?

- Eg, inner classes, enumerations, annotations, generics, builtin databases, closures...
- The latest psychodrama concerned closures (lambda functions)

  Finally made it into Java 8
- See
  - Java closures site
  - Bruce Eckel - *Java: Evolutionary Dead End*

Java - what choices

- Freeze the language (à la C)
- And risk developers abandoning for ''sexier'' languages
- Java is steadily losing popularity

Java - what choices

- Add features while maintaining backward compatibility
- And risk the complexity of C++ (backward compatible with C)

Java - what choices

- Add features that may break compatibility with existing codebase
- Python 3 has taken this courageous but sometimes painful approach
- Implementation can be greatly simplified

Java - what choices

- Abandon source-code compatibility while maintaining binary compatibility
- Eg, Scala, Groovy, Clojure, Jython...

Java - what choices

''My argument is that trying to add fundamental new features AND preserve backward compatibility produces bad new features. My previous attitude was to break backwards compatibility in order to add the feature correctly. I now think that the feature shouldn't be added if it requires compromise in order to maintain compatibility.''

Bruce Eckel

Java - what choices

''Java is functional; trying to add every feature will break it. If you really need those features, perhaps using another JVM language that is designed from scratch to support those features (Scala, for example) is a better choice than trying to (badly) force it into Java.''

Bruce Eckel

New and improved Java7/8

- Small language enhancements
  - Strings in `switch`
  - Support for dynamically-typed languages
  - More new I/O APIs
- Major enhancement
  - Lambda expressions (enable closures)
- Etc...

Java futures

- Today – “How many of you know COBOL?”

Java futures

- In 10 years - “How many of you know Java?”
- Takeaway lessons
  - Learn the principles, details are subject to change
  - Don't put all your eggs in one basket

And now for something completely different...

Java is free, right?

- What exactly is the difference between:
  - open source
  - free...as in gratuit (or, free as in beer)
  - free...as in libre

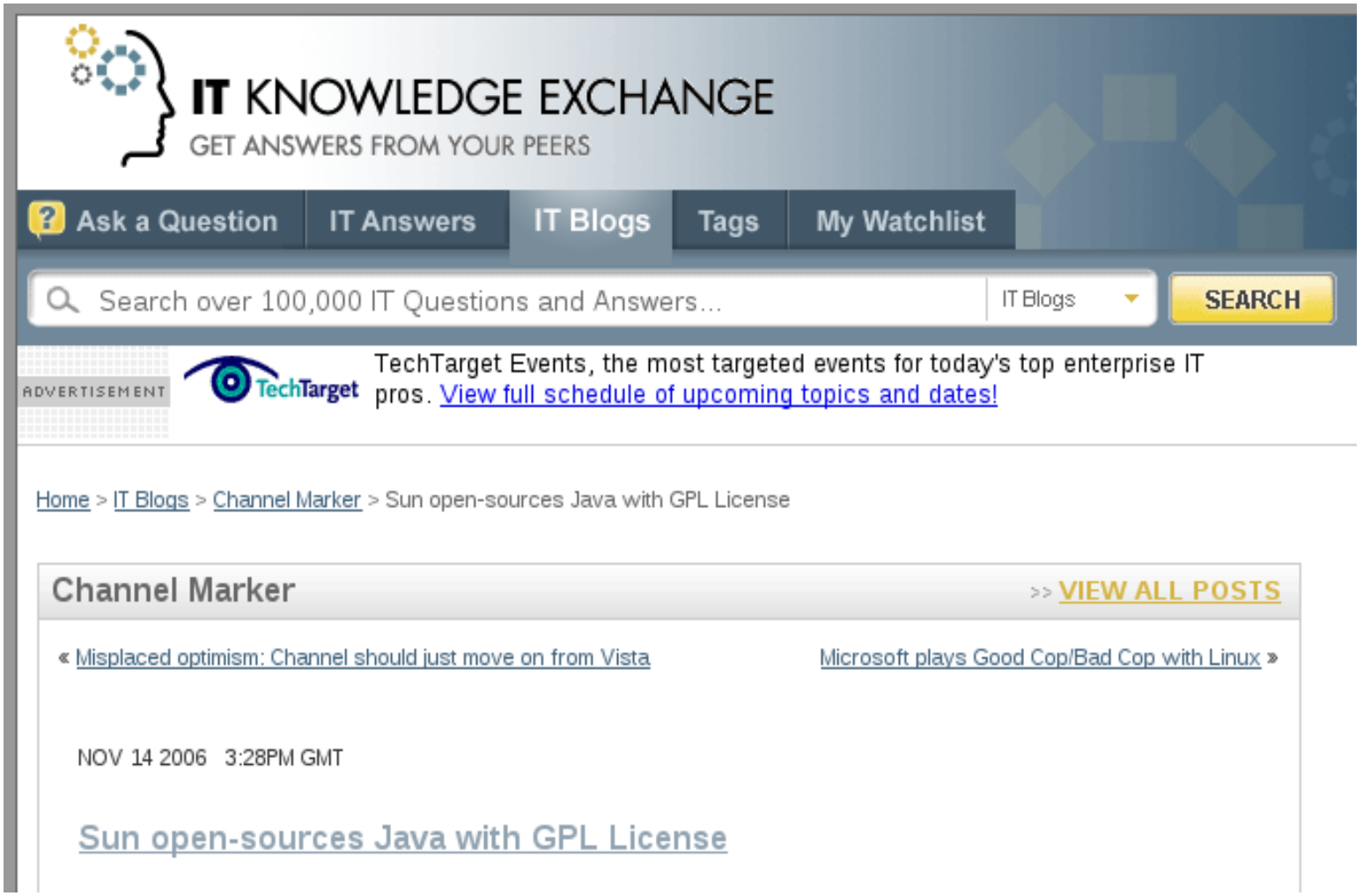
Java is free, right?

- Richard Stallman had warned for years about the Java trap:

“If you develop a Java program on Sun's Java platform, you are liable to use Sun-only features without even noticing. By the time you find this out, you may have been using them for months, and redoing the work could take more months. You might say, “It's too much work to start over.” ...”

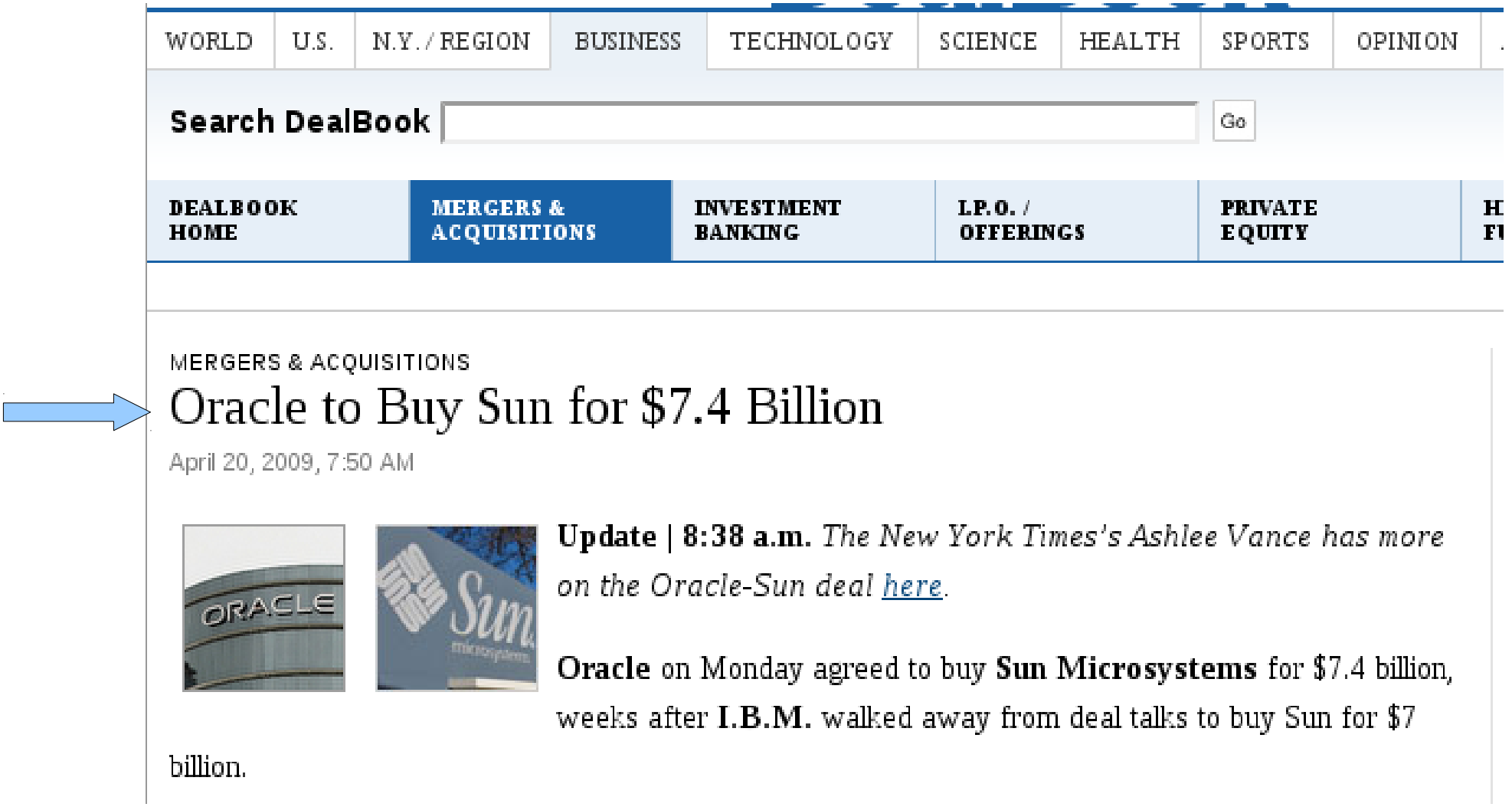
Java is what, exactly?

- Under GPL licence – free as in libre?



Java is what, exactly?

- Oh oh...



WORLD | U.S. | N.Y. / REGION | BUSINESS | TECHNOLOGY | SCIENCE | HEALTH | SPORTS | OPINION

Search DealBook Go

DEALBOOK HOME | MERGERS & ACQUISITIONS | INVESTMENT BANKING | I.P.O. / OFFERINGS | PRIVATE EQUITY

MERGERS & ACQUISITIONS

Oracle to Buy Sun for $7.4 Billion

April 20, 2009, 7:50 AM

Update | 8:38 a.m. *The New York Times's Ashlee Vance has more on the Oracle-Sun deal here.*

Oracle on Monday agreed to buy **Sun Microsystems** for $7.4 billion, weeks after **I.B.M.** walked away from deal talks to buy Sun for $7 billion.

Java is what, exactly?

- Oops...

Java is what, exactly?

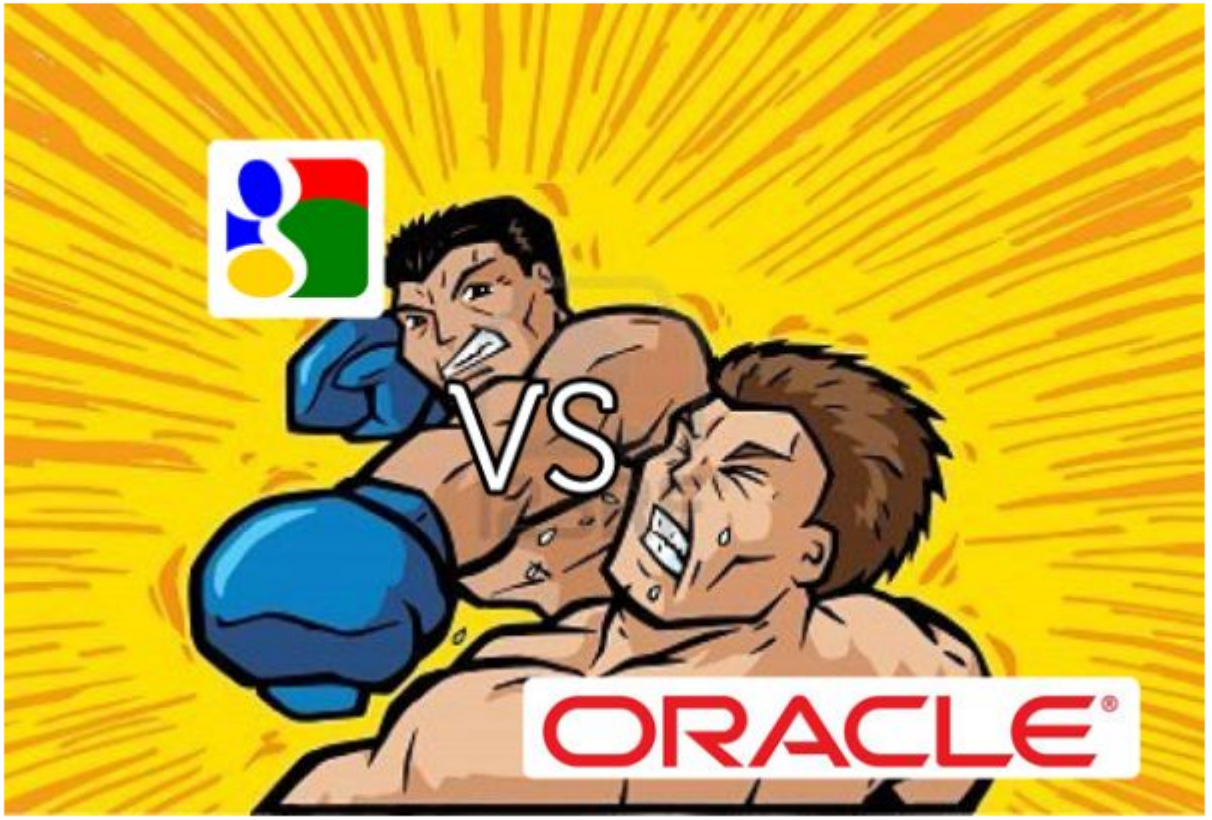
"Now, if you are choosing a programing language, aren't you more likely to consider something truly unencumbered like Ruby or Python — where something like this just wouldn't happen — than you are Java? Joel West points out a problem that Sun always had — that of semi-openness — which now comes back to bite those that trusted it."

Verdict

Pow! Oracle told by judge to cover for Google's legal expenses

by **Bams Sadewo** on Jun 05, 2012 with **5 Comments**

20
Tweet
20
Like
23
+1



The smackdown against Oracle in the courtroom continues. It was only last week that victory

ANDROID NEWS

Verizon Galaxy N
spotted, release
available; U.S. C
it too

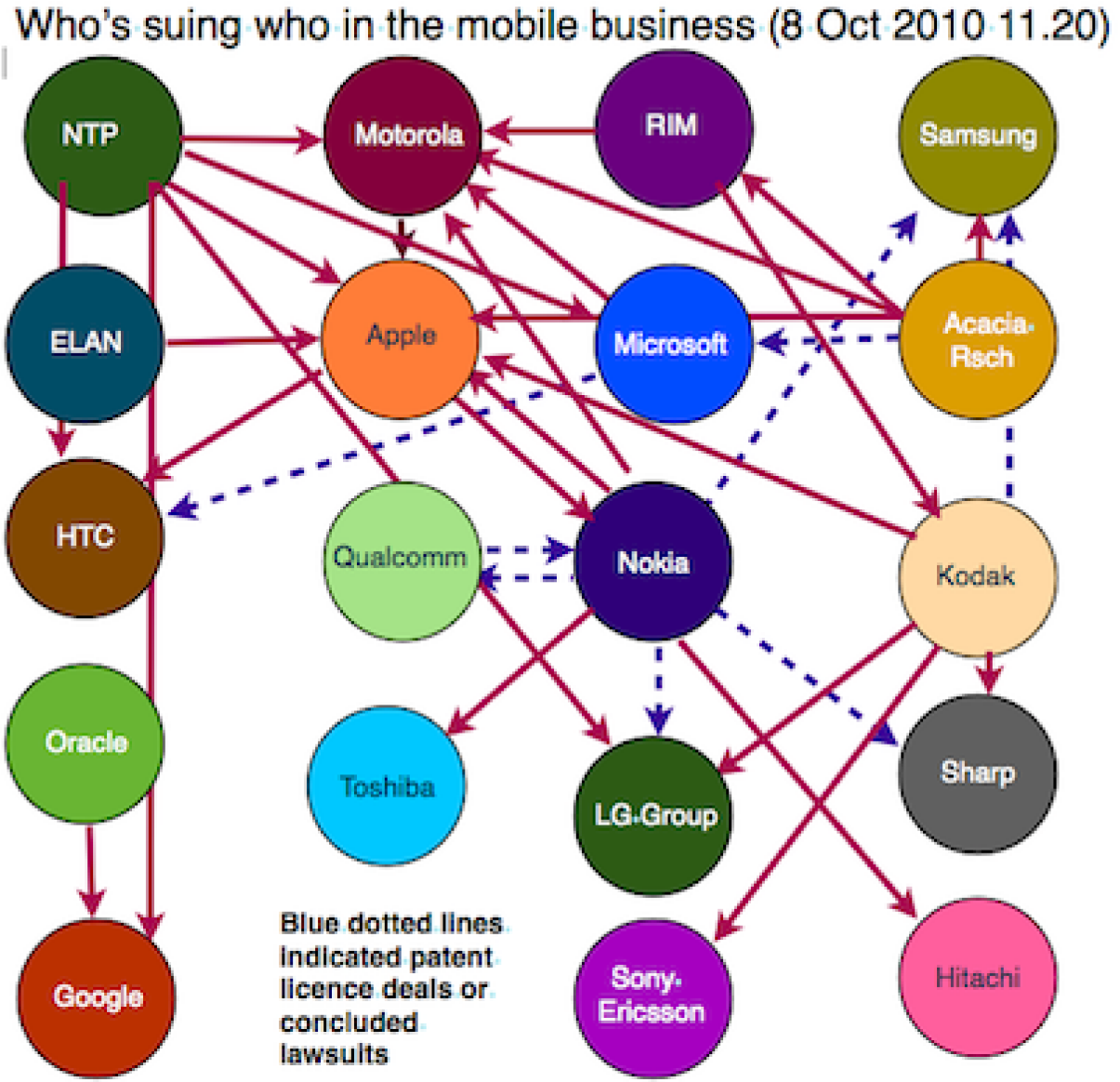
Samsung Galaxy
everything you n
about the Jelly B
before launch

Deal: Cheaper T-
Galaxy S3 availa
Walmart, Target

New Galaxy S3 J
firmware leaks –
– release date fo
Android 4.1.1 up
available [Downl

Android Authorit
Episode 28 – Mo
RAZR HD and Kir

Oracle vs. Google vs...



Zombies never go away...

eWEEK

MOBILE CLOUD SECURITY STORAGE ENTERPRISE APPS INNOVATION

Android Apple IT Management Networking More Slide Shows Video Blogs Research Ce

News & Analysis / Java APIs Copyrightable, Court Rules in Oracle vs. Google

Java APIs Copyrightable, Court Rules in Oracle vs. Google

By Chris Preimesberger | Posted 2014-05-09 Email Print

Two years ago its case seemed dead in the water, but Oracle now can pursue its high-profile copyright suit against Google over Android's use of Java.

A major intellectual property lawsuit involving IT giants Oracle and Google is in the process of determining the legal definitions of "creative" and "utilitarian" in software development.

A three-judge U.S. Court of Appeals in Washington, D.C., on May 9 overturned a 2012 circuit court decision, ruling that Oracle is entitled to copyright protection over application programming interfaces in the Java programming language that are vital components in the open-source Android operating system.

As a result, Oracle now can pursue its high-profile copyright suit against Google over Android's use of Java.

...but they can be wounded!

The Register®
Biting the hand that feeds IT

DATA CENTRE SOFTWARE NETWORKS SECURITY INFRASTRUCTURE DEVOPS BUSINESS HARDWAR

Business ▸ Law

$10bn Oracle v Google copyright jury verdict: Google wins, Java APIs in Android are Fair Use

Big Red tells us it immediately plans to appeal

26 May 2016 at 20:19, Shaun Nichols and Chris Williams

108 91