The Apache Harmony Project

Tim Ellison
Geir Magnusson Jr.

Apache Harmony Project
http://harmony.apache.org

TS-7820

Goal of This Talk

In the next 45 minutes you will...

Learn about the motivations, current status, and future plans of the Apache Harmony project

Agenda

Project History

Development Model

Modularity

VM Interface

How Are We Doing?

Relevance in the Age of OpenJDK

Summary

Agenda

Project History

Development Model

Modularity

VM Interface

How Are We Doing?

Relevance in the Age of OpenJDK

Summary

Apache Harmony

In the Beginning

May 2005—founded in the Apache Incubator

Primary Goals

1. Compatible, independent implementation of Java™ Platform, Standard Edition (Java SE platform) under the Apache License
2. Community-developed, modular architecture allowing sharing and independent innovation
3. Protect IP rights of ecosystem

Apache Harmony

Early history: 2005

Broad community discussion

- Technical issues
- Legal and IP issues
- Project governance issues

Goal: Consolidation and Consensus

Early History

Early history: 2005/2006

Initial Code Contributions

- Three Virtual machines
  - JCHEVM, BootVM, DRLVM
- Class Libraries
  - Core classes, VM interface, test cases
  - Security, beans, regex, Swing, AWT
  - RMI and math

Early History

Early history—2005/2006

- Development activity builds on the initial contributions
  - We got lucky—real VMs to build and test with
    - IBM J9 and BEA JRockit made available to project
      - Used for development
      - Not under open source license, not a contribution
    - Other goals
      - Grow the committers
      - Establish the build/test infrastructure
- October 2006: graduated Incubator, became Apache Harmony project

Agenda

Project History

Development Model

Modularity

VM Interface

How Are We Doing?

Relevance in the Age of OpenJDK

Summary

Development Approach

Enhanced IP-cleanliness

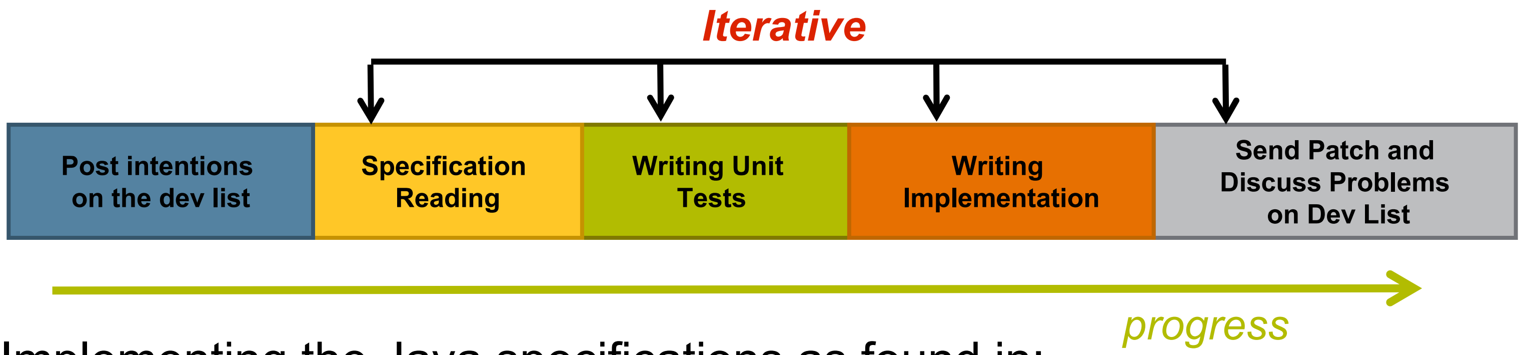
- Contributors detail their prior access via a project questionnaire
- Developers can contribute in functional areas where they have not studied closed-source implementations (exceptions apply)
- Existing code being contributed to the project must provide acceptable pedigree information

This Is in Addition to the Standard Apache Contribution Processes

See: http://harmony.apache.org/auth_cont_quest.html

Spec-Driven Development

Producing a compliant implementation



- Implementing the Java specifications as found in:
  - Java SE platform Javadoc™ tool
  - Java Programming Language specification
  - Java Virtual Machine (JVM™) specification, etc.
- Ambiguities and omissions…
  - Resolved by the reference implementation
  - Determined by functional API testing
- Completeness and compliance determined by Java Compatibility Kit (JCK)

The terms “Java Virtual Machine” and “JVM” mean a Virtual Machine for the Java™ platform.

Test-Driven Development

Producing a robust, compatible implementation

- Functional tests
  - API tests, internal interfaces tests, bug regression tests
  - Component-oriented
- Integration tests
  - Build verification, test with different VMs
  - Component assembly oriented
- Application/system testing
  - Running popular apps, ad hoc, and test suites
  - End-product and compatibility-oriented
- Platform, performance, and stress testing
  - Multi-platform continuous integration, growing performance/stress suites
  - Robustness and quality-oriented

Project Organization

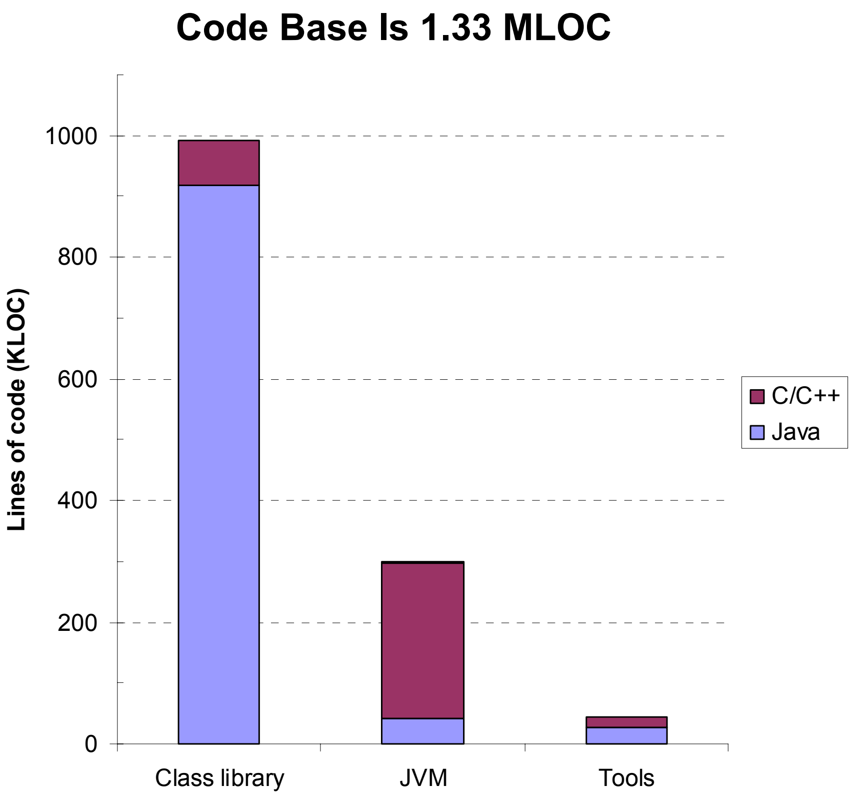
Current codebase

`/classlib`
- the class library code

`/drlvm`
- the VM, JIT, GC

`/jdktools`
- Java technology development tools (javac, javah, javap, ...)

`/trunk`
- federation point (builds everything together)



Federated Build

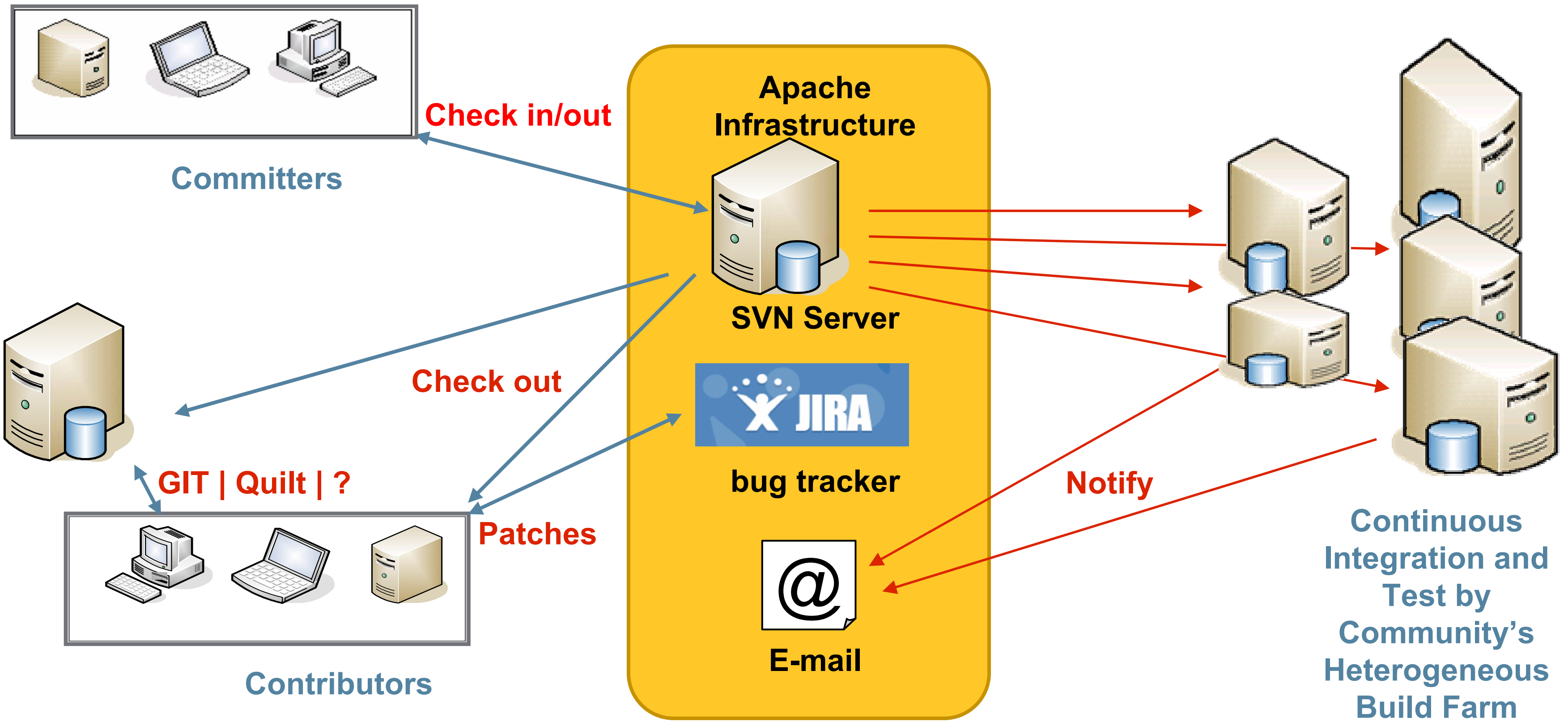
Bringing it all together

```
/trunk
   build.xml
   /working_classlib  -> /classlib
   /working_vm        -> /<your favorite vm>
   /working_jdktools  -> /jdktools
   /current_resources -> /current_resources
```

- Uses `svn switch` “trick”
- Downloads project dependencies
- Running “ant” results in complete HDK, Java Development Kit (JDK™) and Java Runtime Environment (JRE™)
- Can work in any directory (svn commit does Right Thing)
- `ant -Dsvn.revision=X` builds any SVN revision

Distributed Build-test

Heterogeneous build and test farm across community



Apache Infrastructure

SVN Server

bug tracker

E-mail

Committers

Contributors

Continuous Integration and Test by Community's Heterogeneous Build Farm

Agenda

Project History

Development Model

Modularity

VM Interface

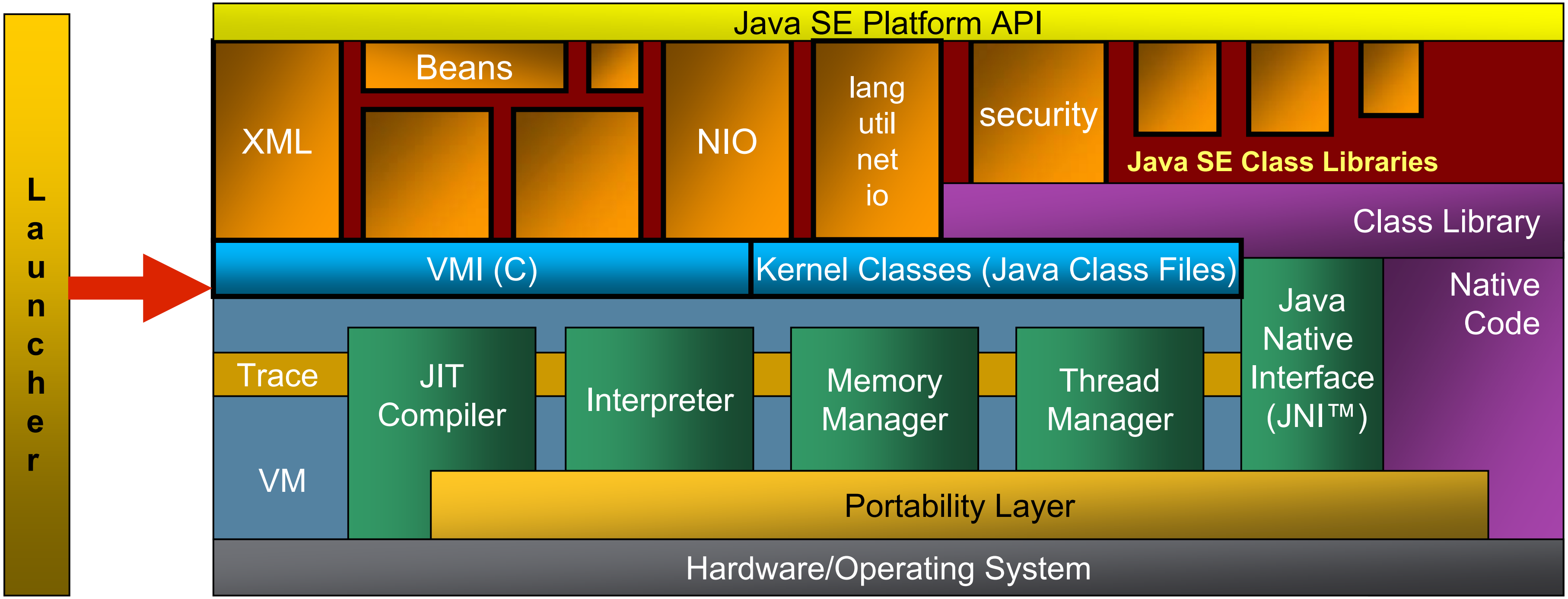
How Are We Doing?

Relevance in the Age of OpenJDK

Summary

Architectural Overview

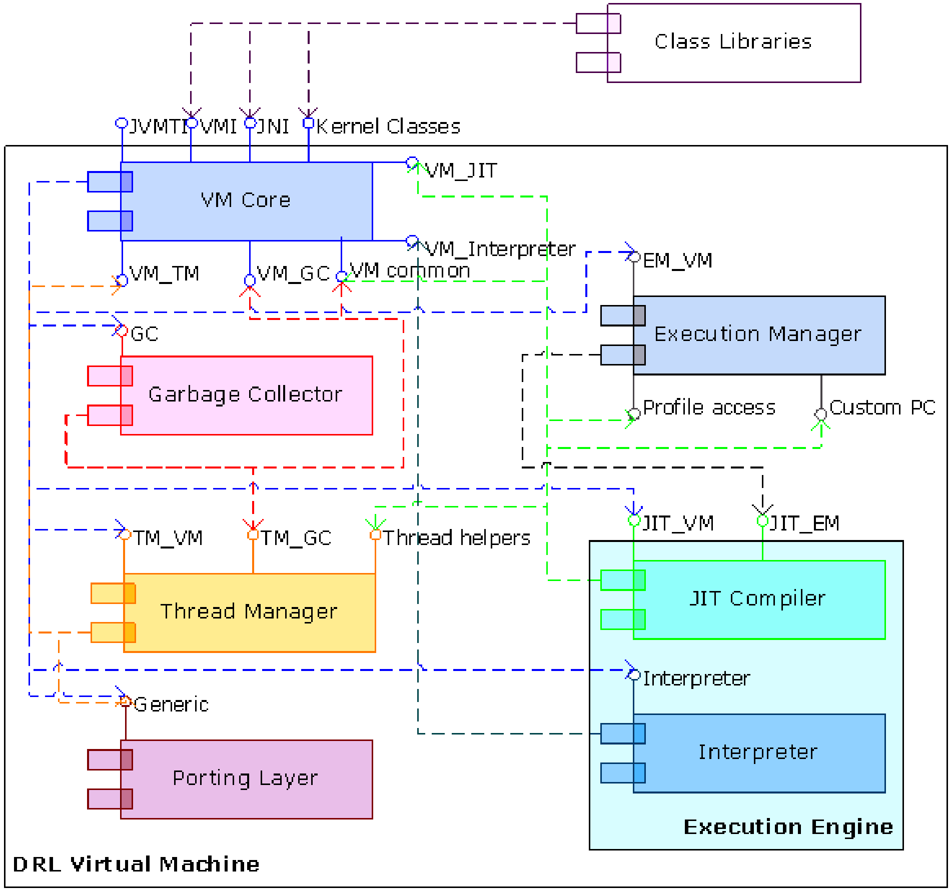
Everything is pluggable



Virtual Machine Modularity

Code execution and memory management

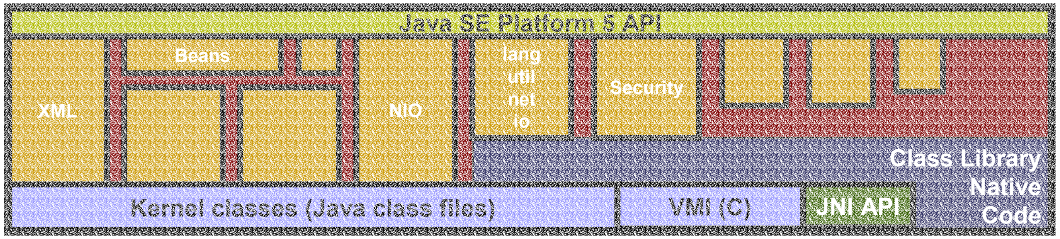
- Well-defined interfaces, consistent across platforms
- Interfaces do not compromise runtime performance
- Modules either build-time or runtime replaceable
- Multiple implementations already exist for some modules



Class Library Modularization

Java SE platform implemented in ~ 30 components

- A module
  - Related functionality scoped by Java technology packages

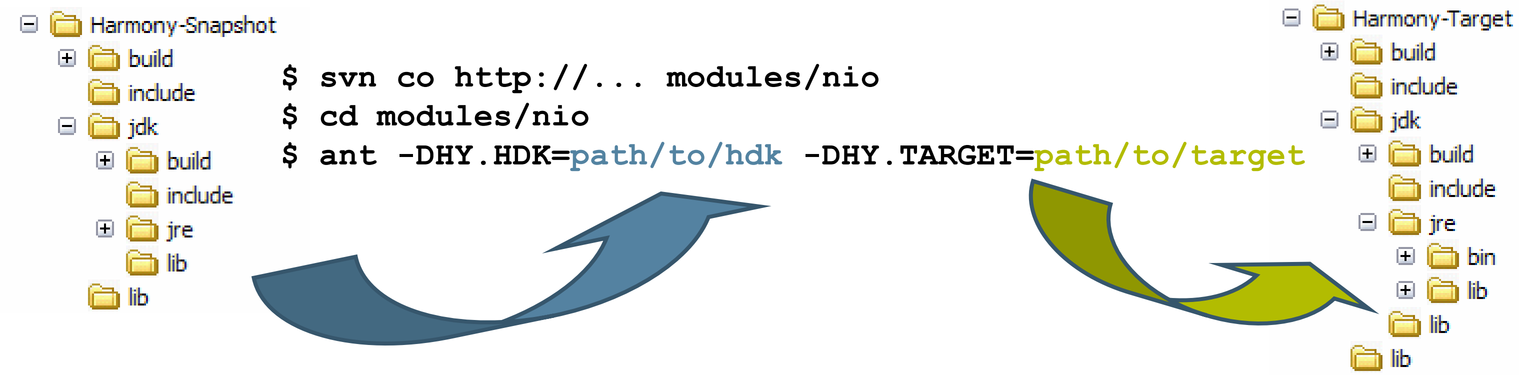


  - ‘Exports’ user API and internal API, hides private internal implementation
  - Defined by dependencies in the Java specification
  - Minimizes coupling by explicit internal APIs
  - Our Java Archive (JAR) files are real OSGi bundles
- Benefits
  - Easier to manage prior exposure
  - Freedom of assembly for module consumers
  - Unit of replacement for fixes and updates
  - Facilitates contributions

Development Time Modularity

The Harmony Development Kit (“HDK”)

- Not a replacement for JDK software
  - HDK → Harmony devs :: JDK software → Java technology devs
  - Contains all files necessary for Harmony development and testing
- Enables fast rebuild of individual modules—Java platform and native
- Removes necessity to check out whole source tree
- Supports separate or in place development of HDK trees

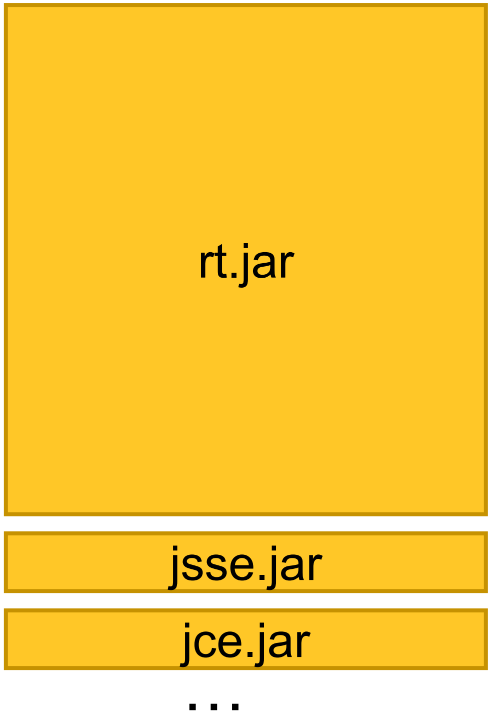


```
$ svn co http://... modules/nio
$ cd modules/nio
$ ant -DHY.HDK=path/to/hdk -DHY.TARGET=path/to/target
```

Packaging Modularity

Software assemblies

Conventional Approach



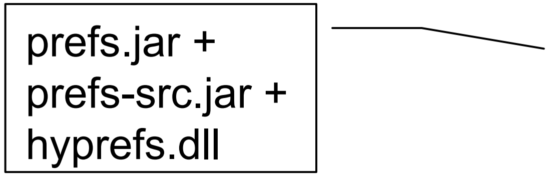
rt.jar

jsse.jar

jce.jar

…

Harmony’s Approach



prefs.jar +
prefs-src.jar +
hyprefs.dll

Runtime Modularity

Multi-VM launcher

- Single launcher program
- Runs command-line Java technology programs, including generic launcher and JDK software tools
- Select VM provider based on command-line option
- Select runtime specified modules (e.g., GC algorithm)

```
jre/bin/
   java.exe
   default/
   drlvm-v1/
   drlvm-v2/
   ibm-j9/
$ java MyClass
$ java -vmdir:ibm-j9 MyClass
$ java -vmdir:drlvm-v1 MyClass
```

Agenda

Project History

Development Model

Modularity

VM Interface

How Are We Doing?

Relevance in the Age of OpenJDK

Summary

Harmony's Class Library/VM Interface

Compatible VMs are required to implement…

- VM-specific 'kernel' classes
  - 23 publicly defined Java SE platform types that the VM typically knows intimately, plus one helper
  - Harmony provides templates for many of these
- Standard Java Native Interface
  - To create objects, etc. from class library natives
- Harmony defined VM interface functions
  - 10 new C functions…

Harmony’s Kernel Class List

VM-specific classes

```
java.lang.Object
java.lang.Class
java.lang.ClassLoader
java.lang.Compiler
java.lang.Package
java.lang.Runtime
java.lang.StackTraceElement
java.lang.System
java.lang.Thread
java.lang.ThreadGroup
java.lang.Throwable
java.security.AccessControlContext
java.security.AccessController
java.lang.reflect.AccessibleObject
java.lang.reflect.Array
java.lang.reflect.Constructor
java.lang.reflect.Field
java.lang.reflect.Method
java.lang.ref.PhantomReference
java.lang.ref.Reference
java.lang.ref.WeakReference
java.lang.ref.SoftReference
org.apache.harmony.kernel.VM
```

- VM implementers provide concrete implementations
  - Either written from scratch, or derived from Harmony’s stub implementations

Harmony’s VMI Functions

Additional C-interface to the VM

- Access to structures and interfaces shared by the VM and class library
- The VMI provides:
  - Access to the operating system abstraction library (port library)
  - Access to per VM storage functions (VMLS) which allows multiple VMs to exist in a single address space
  - Ability to get/set/iterate system properties
  - Major.minor version information to detect incompatible VMI shape changes
- The VMI does not:
  - Require any enhanced VM/class library linkage
  - Prescribe object layout, garbage collection, synchronization, and so on

Harmony’s VMI Functions

Compatible VMs are required to implement

```
/* Version check */
vmiError (JNICALL* CheckVersion) (VMInterface* vmi, vmiVersion* version);


/* Obtain VM structures & interfaces */
JavaVM*               (JNICALL* GetJavaVM)         (VMInterface* vmi);
JavaVMInitArgs*       (JNICALL* GetInitArgs)       (VMInterface* vmi);
HyPortLibrary*        (JNICALL* GetPortLibrary)    (VMInterface* vmi);
HyZipCachePool*       (JNICALL* GetZipCachePool)   (VMInterface* vmi);
HyVMLSFunctionTable*  (JNICALL* GetVMLSFunctions)  (VMInterface* vmi);


/* System properties */
vmiError (JNICALL* GetSystemProperty) (VMInterface* vmi, char* key, char** valuePtr);
vmiError (JNICALL* SetSystemProperty) (VMInterface* vmi, char* key, char* value);
vmiError (JNICALL* CountSystemProperties) (VMInterface* vmi, int* countPtr);
vmiError (JNICALL* IterateSystemProperties) (VMInterface* vmi,
                                 vmiSystemPropertyIterator iterator, void* userData);
```

VMs Expose the VMI Struct

Via two exported functions

```
VMInterface* JNICALL VMI_GetVMIFromJavaVM (JavaVM* vm);

VMInterface* JNICALL VMI_GetVMIFromJNIEnv (JNIEnv* env);
```

Agenda

Project History

Development Model

Modularity

VM Interface

How Are We Doing?

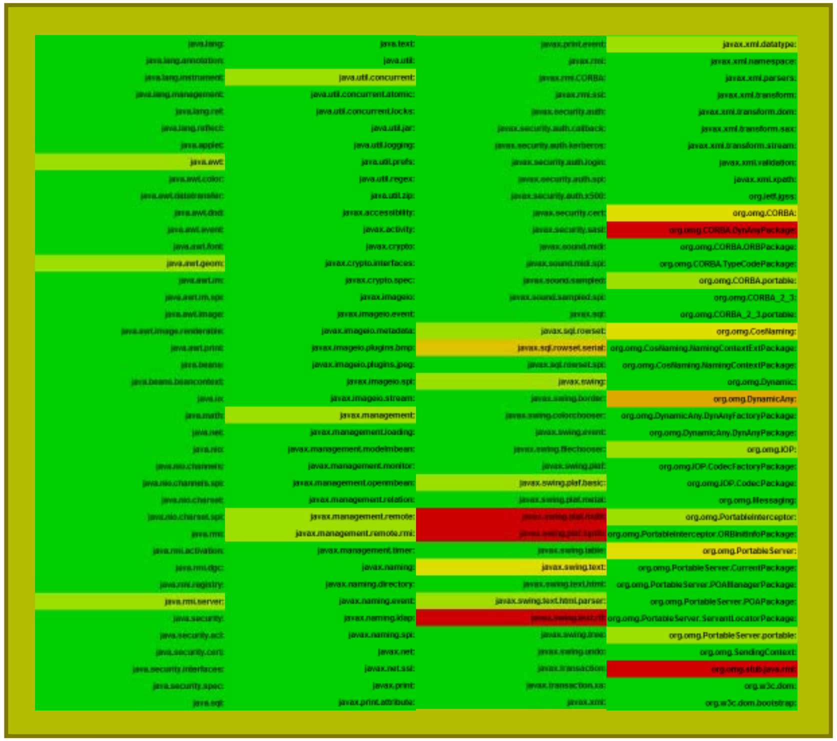
Relevance in the Age of OpenJDK

Summary

API Completeness

Our goal—Java SE platform 5

- Completeness "heat map" courtesy of JAPI tool
  - Compares API compatibility
- Over 96% 1.5 API complete in mid-April
- Missing packages are primarily
  - Swing multi and synth PLAF
  - RTF support
  - ORB dynamic introspection



java.lang:	java.text:	javax.print.event:	javax.xml.datatype:
java.lang.annotation:	java.util:	javax.rmi:	javax.xml.namespace:
java.lang.instrument:	java.util.concurrent:	javax.rmi.CORBA:	javax.xml.parsers:
java.lang.management:	java.util.concurrent.atomic:	javax.rmi.ssl:	javax.xml.transform:
java.lang.ref:	java.util.concurrent.locks:	javax.security.auth:	javax.xml.transform.dom:
java.lang.reflect:	java.util.jar:	javax.security.auth.callback:	javax.xml.transform.sax:
java.applet:	java.util.logging:	javax.security.auth.kerberos:	javax.xml.transform.stream:
java.awt:	java.util.prefs:	javax.security.auth.login:	javax.xml.validation:
java.awt.color:	java.util.regex:	javax.security.auth.spi:	javax.xml.xpath:
java.awt.datatransfer:	java.util.zip:	javax.security.auth.x500:	org.ietf.jgss:
java.awt.dnd:	javax.accessibility:	javax.security.cert:	org.omg.CORBA:
java.awt.event:	javax.activity:	javax.security.sasl:	org.omg.CORBA.DynAnyPackage:
java.awt.font:	javax.crypto:	javax.sound.midi:	org.omg.CORBA.ORBPackage:
java.awt.geom:	javax.crypto.interfaces:	javax.sound.midi.spi:	org.omg.CORBA.TypeCodePackage:
java.awt.im:	javax.crypto.spec:	javax.sound.sampled:	org.omg.CORBA.portable:
java.awt.im.spi:	javax.imageio:	javax.sound.sampled.spi:	org.omg.CORBA_2_3:
java.awt.image:	javax.imageio.event:	javax.sql:	org.omg.CORBA_2_3.portable:
java.awt.image.renderable:	javax.imageio.metadata:	javax.sql.rowset:	org.omg.CosNaming:
java.awt.print:	javax.imageio.plugins.bmp:	javax.sql.rowset.serial:	org.omg.CosNaming.NamingContextExtPackage:
java.beans:	javax.imageio.plugins.jpeg:	javax.sql.rowset.spi:	org.omg.CosNaming.NamingContextPackage:
java.beans.beancontext:	javax.imageio.spi:	javax.swing:	org.omg.Dynamic:
java.io:	javax.imageio.stream:	javax.swing.border:	org.omg.DynamicAny:
java.math:	javax.management:	javax.swing.colorchooser:	org.omg.DynamicAny.DynAnyFactoryPackage:
java.net:	javax.management.loading:	javax.swing.event:	org.omg.DynamicAny.DynAnyPackage:
java.nio:	javax.management.modelmbean:	javax.swing.filechooser:	org.omg.IOP:
java.nio.channels:	javax.management.monitor:	javax.swing.plaf:	org.omg.IOP.CodecFactoryPackage:
java.nio.channels.spi:	javax.management.openmbean:	javax.swing.plaf.basic:	org.omg.IOP.CodecPackage:
java.nio.charset:	javax.management.relation:	javax.swing.plaf.metal:	org.omg.Messaging:
java.nio.charset.spi:	javax.management.remote:	javax.swing.plaf.multi:	org.omg.PortableInterceptor:
java.rmi:	javax.management.remote.rmi:	javax.swing.plaf.synth:	org.omg.PortableInterceptor.ORBInitInfoPackage:
java.rmi.activation:	javax.management.timer:	javax.swing.table:	org.omg.PortableServer:
java.rmi.dgc:	javax.naming:	javax.swing.text:	org.omg.PortableServer.CurrentPackage:
java.rmi.registry:	javax.naming.directory:	javax.swing.text.html:	org.omg.PortableServer.POAManagerPackage:
java.rmi.server:	javax.naming.event:	javax.swing.text.html.parser:	org.omg.PortableServer.POAPackage:
java.security:	javax.naming.ldap:	javax.swing.text.rtf:	org.omg.PortableServer.ServantLocatorPackage:
java.security.acl:	javax.naming.spi:	javax.swing.tree:	org.omg.PortableServer.portable:
java.security.cert:	javax.net:	javax.swing.undo:	org.omg.SendingContext:
java.security.interfaces:	javax.net.ssl:	javax.transaction:	org.omg.stub.java.rmi:
java.security.spec:	javax.print:	javax.transaction.xa:	org.w3c.dom:
java.sql:	javax.print.attribute:	javax.xml:	org.w3c.dom.bootstrap:

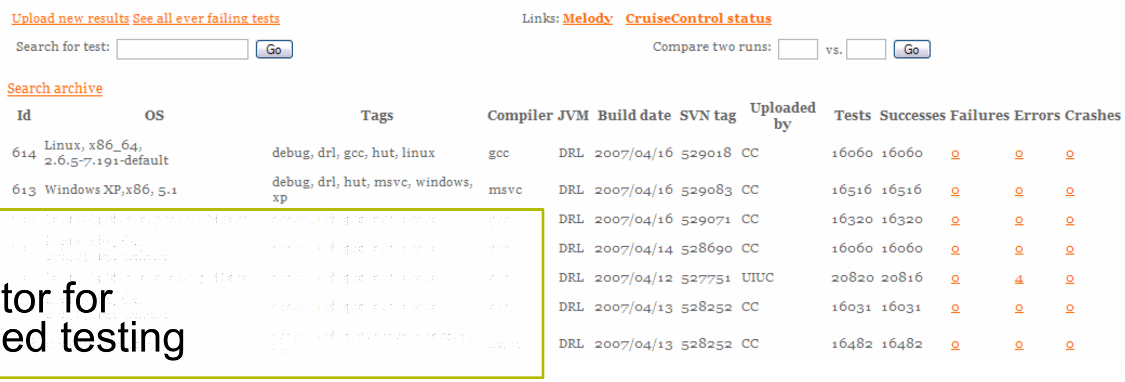
	Good	Minor	Bad	Missing	Abs.add
Total:	96.73%	0.06%	0.02%	3.16%	0.02%

April 2007

See: http://kaffe.org/~stuart/japi

Code Quality

Testing, bug fixing, metrics...



Upload new results See all ever failing tests

Links: Melody CruiseControl status

Search for test: Go

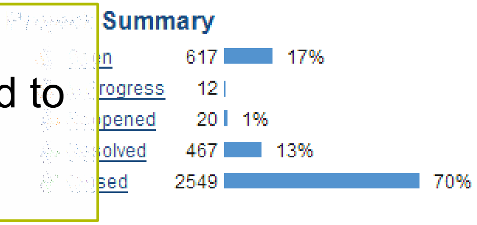
Compare two runs: vs. Go

Search archive

Id	OS	Tags	Compiler	JVM	Build date	SVN tag	Uploaded by	Tests	Successes	Failures	Errors	Crashes
614	Linux, x86_64, 2.6.5-7.191-default	debug, drl, gcc, hut, linux	gcc	DRL	2007/04/16	529018	CC	16060	16060	0	0	0
613	Windows XP,x86, 5.1	debug, drl, hut, msvc, windows, xp	msvc	DRL	2007/04/16	529083	CC	16516	16516	0	0	0
[illegible]	[illegible]	[illegible]	[illegible]	DRL	2007/04/16	529071	CC	16320	16320	0	0	0
[illegible]	[illegible]	[illegible]	[illegible]	DRL	2007/04/14	528690	CC	16060	16060	0	0	0
[illegible]	[illegible]	[illegible]	[illegible]	DRL	2007/04/12	527751	UIUC	20820	20816	0	4	0
[illegible]	[illegible]	[illegible]	[illegible]	DRL	2007/04/13	528252	CC	16031	16031	0	0	0
[illegible]	[illegible]	[illegible]	[illegible]	DRL	2007/04/13	528252	CC	16482	16482	0	0	0

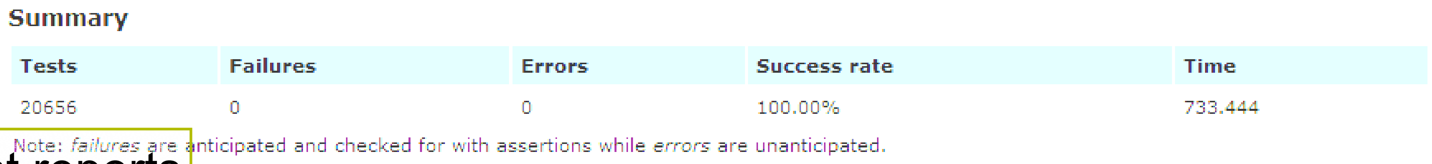
- harmonytest.org
  - Result aggregator for community-based testing

- Bug tracking
  - Commits correlated to fixed bugs
  - Regression tests



Summary

n	617	17%
rogress	12	
pened	20	1%
solved	467	13%
sed	2549	70%



Summary

Tests	Failures	Errors	Success rate	Time
20656	0	0	100.00%	733.444

Note: *failures* are anticipated and checked for with assertions while *errors* are unanticipated.

> 20K JUnit test reports

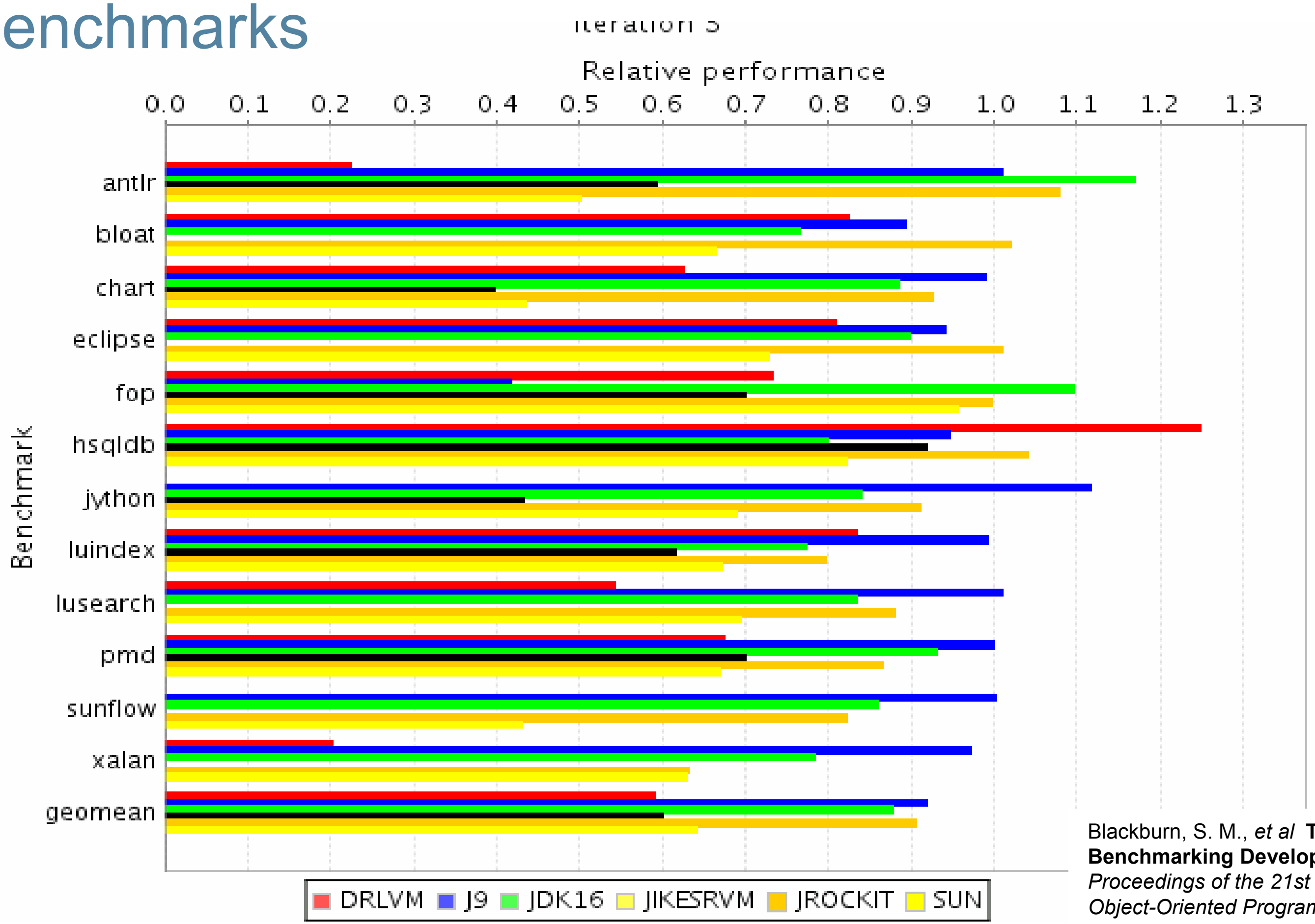
Application-Oriented Testing

Users run applications, not VMs

- Some automated application tests in the Harmony test framework
- Published JDK software/ JRE software builds allow others to easily test and report bugs
  - Apache Tomcat: 100%
  - Apache Geronimo: 96.2%
  - Apache Tuscany SCA: 99.4%
  - Apache Commons: passes 95+% of the 40,000+ tests
  - JBoss
  - Eclipse
  - ...

Performance

Tracked by DaCapo Java technology performance benchmarks



2007-03-07
dacapo-2006-10-MR2

Blackburn, S. M., *et al* **The DaCapo Benchmarks: Java Benchmarking Development and Analysis**, *OOPSLA '06: Proceedings of the 21st annual ACM SIGPLAN conference on Object-Oriented Programming, Systems, Languages, and Applications*, (Portland, OR, USA, October 22-26, 2006)

Source: http://www.dacapobench.org

Multi-Platform Support

Write once, debug everywhere!

- “CruiseControl”-based automated builds and testing
- Community builds, tests, and reports results on:
  - x86 Windows 2000, Windows XP
  - x86 Debian, SUSE9, RHEL AS 4, Ubuntu 6
  - x86_64 SUSE 9, RHEL AS 4, Ubuntu 6
- Known work in progress on:
  - Itanium Linux and Windows
  - PPC 32-bit/64-bit Linux and AIX
  - x86 Mac OS X and FreeBSD
  - zSeries 31-bit/64-bit zLinux and zOS
- Interested in other platforms? Us too!

DEMO

Harmony in action

Agenda

Project History

Development Model

Modularity

VM Interface

How Are We Doing?

Relevance in the Age of Open JDK

Summary

Disclaimer

Caution: The following section contains political and other unfounded statements which may not be suitable for all viewers. It does not represent the position of our respective employers, the Apache Software Foundation, or the Apache Harmony project or our friends or enemies. Sometimes we can’t even believe we say this stuff. Your mileage may vary. Viewer discretion is advised. Offer not valid in CT, TX, CA, or the Canary Islands. Supplies limited. May contain nuts. Contains forward looking statements. We reserve the right to substitute an item of similar value and quality...

For the Avoidance of Doubt…

- We applaud Sun for doing OpenJDK
  - Courageous as predictably internally disruptive
- It's a monumental event in the history of open source
  - We can't think of any contribution to FLOSS of this magnitude
- The adventure is just beginning for Sun and the greater community
- We look forward to find ways to have OpenJDK and Apache Harmony collaborate (hint : javac)

Some Relevant History

- What significant, groundbreaking changes happened to the Java technology ecosystem when Sun released Project Glassfish™?

(Nothing)

The Java Technology Ecosystem

Important underpinnings of the Java Technology Ecosystem

- Specification-based technology
  - *“Collaborate on specifications, compete in implementations” —me*
- Historically—mostly proprietary
  - Started that way by Sun
  - Slowly changing, but much still exists

Moving Forward

- Harmony community members choose to participate for their own, independent reasons
  - “It’s fun”—Dalibor Topic (kaffe)
- Need for FLOSS implementation that permits freedom of downstream licensing persists
  - Sun recognizes this—they will continue to license OpenJDK under proprietary, commercial terms

Moving Forward

- Competition and choice makes Java technology ecosystem healthy
- Community matters, and not all are the same
    - Apache has a well understood, transparent model
        - Collaboration of peers
        - (This doesn’t work for everyone)
    - OpenJDK governance model still undefined
        - Our guess? A model similar to OpenSolaris™ project, Project Glassfish, and NetBeans™ software
            - Ideal for Sun’s users, customers and ISV partners

Agenda

Project History

Development Model

Modularity

VM Interface

How Are We Doing?

Relevance in the Age of OpenJDK

Summary

Summary

The best is yet to come...

- Rapidly approaching a fully compatible, open source, implementation of Java SE platform
- We invested time up-front getting the IP infrastructure right
- The community is focused on building a first-class runtime environment
- Our strong modularity story has given us flexibility and robustness to progress quickly and maintain stability
- We have come a long way in 2 years!

Accomplishments

What we’ve done so far

- ~ 96% of Java platform v.5 class library API coverage
- Modern, Java platform v.5-capable VM with JIT
- Community continues to grow
- Full Apache project
- Self-hosting codebase
- “Runs” many major applications
- Modularity approach successful in classlib, ongoing in VM

Active Areas of Effort

What we still need to do

- Class library completion
- VM performance work
- VM stability work
- Build/test infrastructure
- Improved test case coverage
- Real world application testing
- Community growth

Q&A

Geir Magnusson Jr. geirm@apache.org

Tim Ellison tellison@apache.org

http://harmony.apache.org

Apache Harmony

The Apache Harmony Project

Tim Ellison
Geir Magnusson Jr.

Apache Harmony Project
http://harmony.apache.org

TS-7820