JavaOne℠

Sun's Worldwide Java Developer Conference

Java™ Compatibility

Carla Schroer
Quality Assurance Manager,
JavaSoft

Overview

- What is compatibility and why is it important?
- Testing requirements for compatibility
- Addressing the testing requirements
- Summary

What Is Java™ Compatibility?

- Implementations meet the specifications
- A measure of compliance with the specification, not performance robustness or other quality issues

Components for Compatability

1. Compilers must conform to the Java™ Language specification
2. Virtual Machine (VM) implementations must conform to the Virtual Machine Specification
3. Application Programming Interface (API) support must conform to the API documentation

Why Is Java Compatibility Important?

- To JavaSoft – maintain control of the language
- To developers – it means you write it once and it runs everywhere including across platforms and across products
- To users – can get applications from anywhere and they run on your machine

What Does the Java-Compatible Logo Mean?

- The product has passed the appropriate test suites
- All licensees of the Java technology are required to pass the test suites
- The test suites are tied to a specific version of Java

Who Gets the Logo?

- Today–only licensees of Java-based technology can get the logo, for example:
  - Netscape’s Navigator 2.0
  - Symantec’s Cafe
  - SunSoft’s Java Workshop
- Future–allow ports and “clean room” implementations to get the logo through a certification process
- Lack of certification process is due to concerns about security, malicious intent and completeness of the test suites

Java in Operating Systems

- The Java language will be available directly in many operating systems

–Apple	MAC OS
–HP	HP-UX
–IBM	OS/2, win3.1, MVS, AIX
–Microsoft	Windows 95, Windows NT
–Novell	Netware 4.0
–SCO	UnixWare
–SGI	IRIX
–SunSoft	Solaris
–Tandem	Non-stop Kernel

- All these implementations must pass the JavaSoft compatibility test suites

Three Main Components to Test

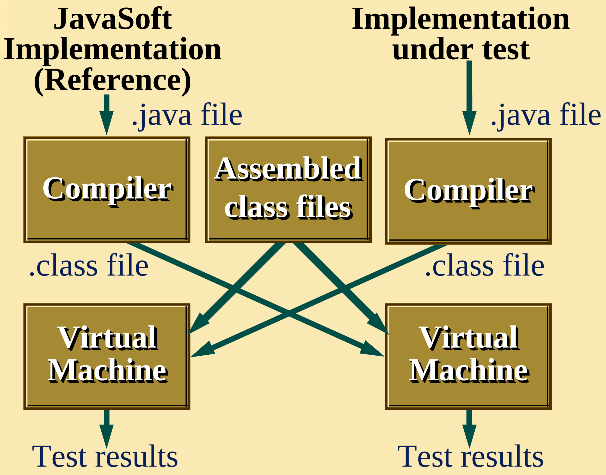
1. Compilers must conform to the Java Language specification
2. Virtual Machine implementations must conform to the Virtual Machine Specification
3. API support must conform to the API documentation
   - Base API is java.lang, java.net, java.io, java.util, java.applet, java.awt packages
   - Additional APIs being developed

Security testing is important for all components

Security Test Examples

- In the compiler
  - Can’t cast pointers
  - Can’t do pointer arithmetic
- In the virtual machine
  - Verifier attacks must fail
  - Can’t overrun arrays
- In an Applet environment
  - Can’t read or write files
  - Can’t make illegal socket connections

Reference Implementation



Examples

• Web browsers	VM, Applet API, Applet Security
• Developer tools	Compiler, VM, Base Java API
• Java in Operating Systems	VM,Base Java API

Examples

- Things not tested for conformance today:
  - Debuggers
  - Java compliant source code generators
  - Compilers for other languages

Other Testing Constraints

- System under test may not have:
  - File system
  - Window system
- Tests must be implementation independent

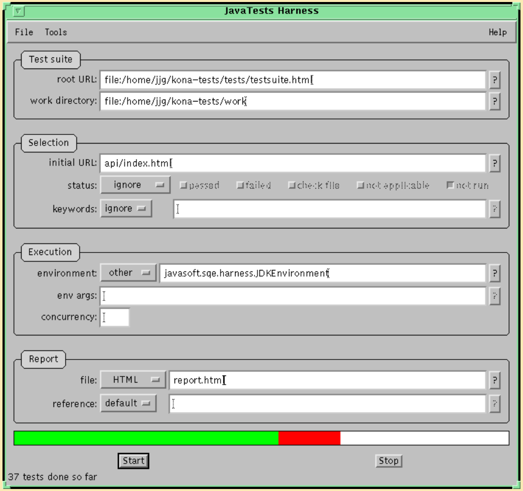
Addressing the Testing Requirements

- Testing Tools
- Compilers
- Virtual Machines
- Base APIs

Testing Tools (Internally Called “Java Test Kit”)

- A set of tools written in Java to manage the testing process
  - Compile test programs
  - Execute test programs
  - Browse results
  - Generate reports
  - Perform code coverage analysis
- Tests, tools and reports are organized around HTML pages, following the browser model
- Used by JavaSoft and licensees of Java-based technology

Example: Harness Application



JavaTests Harness

Test suite

root URL: file:/home/jjg/kona-tests/tests/testsuite.html

work directory: file:/home/jjg/kona-tests/work

Selection

initial URL: api/index.html

status: ignore ☐ passed ☐ failed ☐ check file ☐ not applicable ☐ not run

keywords: ignore

Execution

environment: other javasoft.sqe.harness.JDKEnvironment

env args:

concurrency:

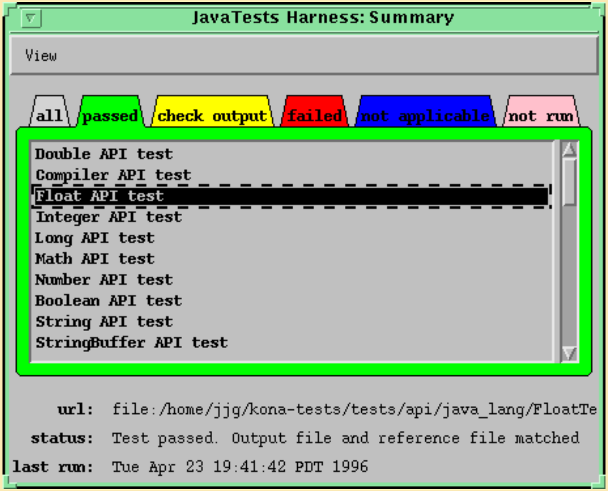
Report

file: HTML report.html

reference: default

37 tests done so far

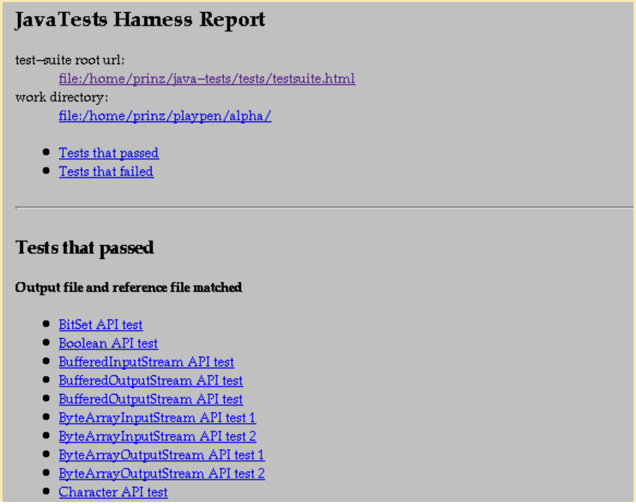
Example: Harness Summary



Double API test
Compiler API test
Float API test
Integer API test
Long API test
Math API test
Number API test
Boolean API test
String API test
StringBuffer API test

url:	file:/home/jjg/kona-tests/tests/api/java_lang/FloatTe
status:	Test passed. Output file and reference file matched
last run:	Tue Apr 23 19:41:42 PDT 1996

Example: Harness Report



JavaTests Harness Report

test–suite root url:

file:/home/prinz/java–tests/tests/testsuite.html

work directory:

file:/home/prinz/playpen/alpha/

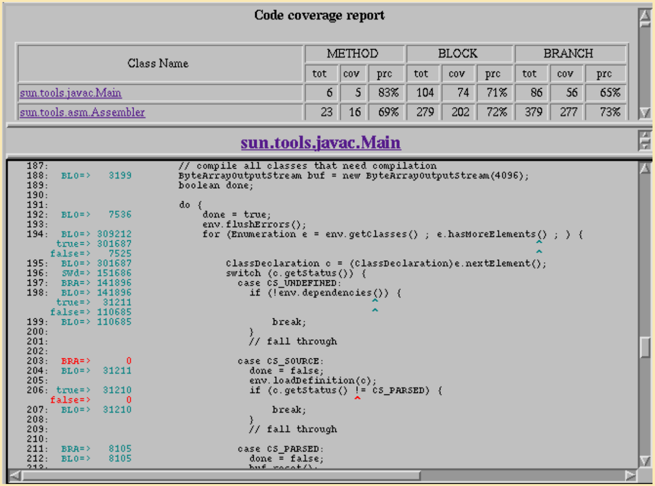
- Tests that passed
- Tests that failed

Tests that passed

Output file and reference file matched

- BitSet API test
- Boolean API test
- BufferedInputStream API test
- BufferedOutputStream API test
- BufferedOutputStream API test
- ByteArrayInputStream API test 1
- ByteArrayInputStream API test 2
- ByteArrayOutputStream API test 1
- ByteArrayOutputStream API test 2
- Character API test

Code Coverage Report



Code coverage report

Class Name	METHOD			BLOCK			BRANCH		
	tot	cov	prc	tot	cov	prc	tot	cov	prc
sun.tools.javac.Main	6	5	83%	104	74	71%	86	56	65%
sun.tools.asm.Assembler	23	16	69%	279	202	72%	379	277	73%

sun.tools.javac.Main

```
187:                      // compile all classes that need compilation
188:  BLO=>   3199        ByteArrayOutputStream buf = new ByteArrayOutputStream(4096);
189:                      boolean done;
190:
191:                      do {
192:  BLO=>   7536            done = true;
193:                          env.flushErrors();
194:  BLO=> 309212            for (Enumeration e = env.getClasses() ; e.hasMoreElements() ; ) {
     true=> 301687                                                                     ^
    false=>   7525                                                                     ^
195:  BLO=> 301687                ClassDeclaration c = (ClassDeclaration)e.nextElement();
196:  SWd=> 151686                switch (c.getStatus()) {
197:  BRA=> 141896                  case CS_UNDEFINED:
198:  BLO=> 141896                    if (!env.dependencies()) {
     true=>  31211                                         ^
    false=> 110685                                         ^
199:  BLO=> 110685                        break;
200:                                  }
201:                                  // fall through
202:
203:  BRA=>      0                  case CS_SOURCE:
204:  BLO=>  31211                    done = false;
205:                                  env.loadDefinition(c);
206: true=>  31210                    if (c.getStatus() != CS_PARSED) {
    false=>      0                                      ^
207:  BLO=>  31210                        break;
208:                                  }
209:                                  // fall through
210:
211:  BRA=>   8105                  case CS_PARSED:
212:  BLO=>   8105                    done = false;
```

Compilers

- Most well understood component
- Lots of language testing experience available
- Easy to automate
- Over 1000 tests written – both positive and negative cases
- More tests being developed

Virtual Machines

- Can use all positive language tests
- Need class files not possible with the compiler
  - Built assembler for the Java language to build test cases
  - Test all byte codes
  - Check that all unused byte codes remain unused
  - Over 400 hand assembled test cases
  - Corrupt class file test cases
- More tests being developed

Base APIs

- Test all public and protected methods on all Java-compatible classes in java.lang, java.net, java.io, java.util
- 56 tests (1 test per class)
- AWT classes are hard to automate
  - Existing capture-playback technology is not cross platform enough
  - Ongoing debate about native “look and feel” vs. Java “look and feel”
  - Today it is native, so can’t detect cross platform behavior differences in an automated way
  - Currently use interactive tests to cover AWT classes
- As additional APIs are developed, conformance tests are also developed

Futures

- More tests
- More test tools
- More tests and tools made available

Summary

- Java compatibility ensures that implementations of Java technology meet the specifications
- “Write once, run anywhere”
- Tests and test tools are well underway