Java Compatibility

February 15 2011

The Legal Framework

Background

- Because Sun wanted to promote WORA and to ensure that there was "one Java" that people could rely on...
- Sun created a legal framework which severely restricts the rights of people to implement “Java” in an incompatible manner.
- IP rights are granted only to those who create *Compatible Implementations*.

The JSPA

“For any Specification produced under a new JSR, the Spec Lead for such JSR shall offer to grant a perpetual, non-exclusive, worldwide, fully paid-up, royalty free, irrevocable license under its licensable copyrights in and patent claims covering the Specification... to anyone who wishes to create and/or distribute an Independent Implementation of the Spec. Such license will authorize the creation and distribution of Independent Implementations provided such Independent Implementations: (a) fully implement the Spec(s) including all its required interfaces and functionality; (b) do not modify, subset, superset or otherwise extend the Licensor Name Space, or include any public or protected packages, classes, Java interfaces, fields or methods within the Licensor Name Space other than those required/authorized by the Spec or Specs being implemented; and (c) pass the TCK for such Spec.”

The Spec license

“Specification Lead also grants you a perpetual, non-exclusive, non-transferable, worldwide, fully paid-up, royalty free, limited license... to create and/or distribute an Independent Implementation of the Specification that: (a) fully implements the Specification including all its required interfaces and functionality; (b) does not modify, subset, superset or otherwise extend the Licensor Name Space, or include any public or protected packages, classes, Java interfaces, fields or methods within the Licensor Name Space other than those required/authorized by the Specification or Specifications being implemented; and (c) passes the Technology Compatibility Kit (including satisfying the requirements of the applicable TCK Users Guide) for such Specification.”

The TCK license

"*Product(s)* means a Licensee product which (i) fully implements the Java Specification(s) identified in Exhibit A including all its required interfaces and functionality; (ii) does not modify, subset, superset or otherwise extend the Licensor Name Space, or include any public or protected packages, classes, Java interfaces, fields, methods or constructors within the Licensor Name Space other than those required/authorized by the Specification or Specifications being implemented; (iii) passes the TCK (including satisfying the requirements of the applicable TCK Users Guide) for such Specification..."

The Compatibility Program

Design Principles and Goals

- The principles of *predictability* and *transparency*, when applied to the Java language and the Java platform, help to promote:
  - *WORA* (Write Once, Run Anywhere)
    - Well-written Java programs will execute with equivalent results on all compatible platforms
  - *WYGIWYS* (What You Get Is What You See)
    - Developers and users clearly understand how their programs will behave

Language: Predictability

- Program behaviour is tightly specified
  - Operates down to the bit level (floating point behavior, hash codes...)
- Some performance cost, but worth it..

Language: Transparency

- Focus is on creating easy-to-read source code.
  - It should be obvious from reading the source code how a Java program will behave.
- Some useful features left out for this reason.
  - Preprocessor, macros, operator overloading...
- Major gains in programmer productivity.
- Keeps each project from "inventing its own language."

Platform: Predictability

- All implementations contain the same APIs.
  - No subsetting or supersetting of standard, specified APIs.
- Behavior is fully specified.
- Class-file semantics are identical across implementations.

Platform: Transparency

- Vendor value-add is clearly identified.
- No magic modes, no hidden behavior.
- Always compatible.

Results

- Compatible products available from multiple vendors.
- Portable applications run identically on all compatible products.
- Developers and customers understand what is part of the platform and what is vendor-specific.
- No surprises; Java programs and platforms behave as expected.

Program Overview (1)

- Compatibility is a contractual obligation.
  - Licensees are prohibited from shipping incompatible products.
  - Compatible products can use the Java name and display the *Java Compatible* logo.
- No need to run (but must be capable of passing) all tests in all possible configurations.
- Compatibility is binary.
  - You can't be “almost compatible” or “a little bit incompatible”
  - JCK contains ~120,000 test cases; if you pass 119,999 you fail.
  - If you pass 120,000 and don't meet all other requirements, you fail.

Program Overview (2)

- The specification defines requirements.
- The TCK verifies conformance with specification.
- *Compatibility Rules* define testing requirements.
- Self-certification of compatibility.
  - Licensees run the tests and follow the rules, reporting to Oracle when they are compatible.
  - Must assert that all other requirements have also been met.
  - Oracle may audit results and investigate complaints.
- Level playing-field: Oracle, as well as licensees, must comply.

Testing is not enough

- Compatibility tests can never cover everything.
- Licensees are free to innovate and add value (subject to constraints.)
- We need the flexibility to adapt to new implementations and changing circumstances.
- Therefore, Compatibility Requirements (*The Rules.*) define additional conditions that must be met.
  - (Some requirements are specified in the License rather than in the Rules...)

Compatibility Requirements

- Typically Chapter 2 of the TCK User's Guide.
- Specify what it means to pass the test suite
  - All required tests must pass.
  - Tests must not be modified.
- Specify additional (untested, or even untestable) criteria that must be met.
  - All specified classes and interfaces must be provided and must function as specified.
  - Syntax & semantics of the Java language must be preserved.
- Requirements evolve over time.

A Brief Tour Through The Rules

Introduction

- Chapter 2 of the TCK User's Guide contains definitions and rules.
- This is only a summary - see the User's Guide for complete details.

Compatibility in all modes

- SE1: The Product must be able to satisfy all applicable compatibility requirements, including passing all Conformance Tests, in every Product Configuration and in every combination of Product Configurations, except only as specifically exempted by these Rules.
- *Product Configuration*: "A specific setting or instantiation of an Operating Mode."
- *Operating Mode*: "Any Documented option of a Product that can be changed by a user in order to modify the behavior of the Product." (Installation options, command-line parameters, system properties...)

Why SE1?

- Make it obvious what's compatible.
- It's simple - all modes are compatible.
- Applications can always rely on a conforming configuration.
- Don't have to worry about how you configured the platform or the application.
- No bait and switch.
  - "Yes, we're compatible, but this other mode makes your app run twice as fast."

Exceptions to Rule SE1

- Options that control resources.
  - Resource: “A computational, location, or security resource that may be required for proper execution of the test suite.”
  - At least one combination of such configurations must pass.
  - Example: if security permissions control access to network resources, OK to require permissions needed by TCK.
- Modes that simply report version, usage, or diagnostic info are not required to pass.
- Backward-compatibility modes.

Forbidden Operating Modes

- *Debug mode* that disables all security checks.
- *Production mode* that disables error checks.
- *Fast mode* that disables functionality.

Don't mess with the tests

- Only materials supplied with the test suite may be used for testing
- SE2: No source or binary test or test property may be modified except as documented in the JCK.
- SE3: Only the testing tools supplied in the JCK may be used for testing.
- SE4: The Exclude List can only be modified by the JCP Maintenance Lead.
- SE8: Only test binaries that are supplied in the JCK may be used for testing.

Modifications to the Rules

- SE5: The Maintenance Lead can define exceptions to these Rules. Such exceptions would be made available to and apply to all licensees.
  - Rules will evolve over time.
  - No waivers or special considerations are ever given to anyone.

Define the Platform

- SE6: All hardware and software necessary to ensure that the product behaves compatibly must be documented and available to all users.
  - Example: If a patch to an operating system is required for the product to perform properly, the patch must be documented and available.

Implement the specified APIs

- SE7: The product must contain the full and exact set of public and protected APIs defined in the specification.
- SE7.1: If a product includes multiple Java technologies, it must contain the union of all public and protected APIs of these technologies.
  - No subsetting, supersetting, or modification of the public or protected APIs is allowed.
  - Checked by Signature Test.

Exception to Rule SE7

- SE7.2: With approval of Maintenance Lead, a product may include a newer version of an *Endorsed Standard* than that defined by the specifications.
  - *Endorsed Standard*: An API defined through a standards process outside of JCP and included in a Java technology.
    - E.g. CORBA, SAX, DOM...
  - The Maintenance Lead may issue updated tests.
  - Newer versions of *Endorsed Standards* included in a product must be documented.

Class-file Semantics

- SE9: The functional programmatic behavior of any binary class or interface must be that defined by the Specifications.
  - API behavior must be consistent with the specification even if no conformance tests check for it.
  - Semantics of user-created programs must conform to the Java Language and JVM Specifications.

Native Interfaces

- SE10: Products that implement Native Code interfaces must pass the JNI tests.
  - If you support Native Code, you must support all of JNI
  - You're allowed to not support JNI only if you don't do any native code access.
  - Native code access in addition to JNI is allowed, but is not portable.

Compiler Rules

- Apply to products that “implement or incorporate a compilation function from source code written in the Java programming language, as specified by Oracle, into Executable Output.”

Compiler Rules

- C1: The compiler must pass the JCK tests for the version of the technology targeted by the compiler.
- C2: Compiler output must behave properly when executed on the associated reference runtime.
- C3: The compiler must not produce executable output from source code that does not conform to the Java Language Specification.

No language extensions

- Preserve the syntax and semantics of the language and JVM.
- C4: Comments or directives must not modify the functional programmatic behavior of a class or interface.
- C5: The Executable Output of the Compiler must be in class file format defined by the Specifications.

What about annotations?

- C6: Annotation processors activated in each Product Configuration must be documented.
- C7: For each technology there must be a configuration that causes the Compiler to process annotations as required by that Technology.
- C8: For each Technology that is included in the Product, there must be a Product Configuration that causes the Compiler to process annotations as required by that Technology.
- C9: For each Product Configuration that causes the Compiler to create Executable Output as well as to process annotations, and for each class or interface for which both the Compiler and the Reference Compiler produce Executable Output, the Executable Output generated by the Compiler must be functionally equivalent to the Executable Output generated by the Reference Compiler.

But wait – there's more!

- Rules for Products that include a Schema Compiler and Generator.
  - SC7: A schema produced by the Schema Generator from source code containing a comment or a directive, or from a class file containing non-standard attributes, must be equivalent to a schema produced by the Schema Generator from the same source code with the comment or directive removed, or from the class file with the attributes removed. Two schemas are equivalent if and only if they validate the same set of XML documents (any XML document is either valid against both schemas or invalid against both schemas).
- Rules for Products that include Java-to-WSDL and WSDL-to-Java Tools
  - JW4: Java-to-WSDL Output must fully meet W3C requirements for the Web Services Description language (WSDL) 1.1.

Summary

- Things were relatively simple when we started.
  - Preserve the syntax and semantics of the language and APIs.
  - No subsetting, supersetting, or changes to core namespaces.
  - All implementations should be compatible.
- New technologies (recently, annotation processors; soon, modularity) make existing Rules obsolete and require increasingly complex modifications and additions.
- Java implementations released under open-source licenses such as GPL by definition cannot require compatibility.
- What next?