

Project Coin: Small Language Changes for JDK 7

Joseph D. Darcy
joe.darcy@sun.com
<http://blogs.sun.com/darcy>

“If this book has convinced the reader that a programming language designer needs the expertise of a scientist, the precision of a mathematician, and the taste of an artist as well as the pragmatism of an engineer, then it has achieved one of its objectives.”

*–R. D. Tennent
Principles of Programming Languages*

Language Evolution Principles

- From JavaOne 2008's *Upcoming Java Programming Language Features*,
 - Principle 1: Encourage high-level practices
 - Principle 2: Covet clarity
 - Principle 3: Prefer static typing
 - Principle 4: Isolate the language from APIs

Language Changes in JDK 7

- Modularity, JSR 294
- Annotations on types, JSR 308
- Project Coin, JSR TBD

What do you want in a coin?

- Make the things programmers do everyday easier
- Support Platform Changes in JDK 7

Stability versus Progress

- Default response is to keep the language as-is
- Burden is on the proposer to convince that a change should get **in**;
burden is *not* to convince to keep a change **out**
- Language changes
 - Slow availability
 - Heavyweight
 - Multiple points in toolchain
- Don't use language changes as first-line solution

What can a language change entail?

- *Update the Java Language Spec.*
- Implementation in a compiler
- Add any essential library support
- Write tests
- Update the JVM Spec.
- *Update the JVM and class file tools*
- Update JNI
- *Update the reflective APIs*
- Update serialization support
- Update javadoc output

Updating the JLS

- Grammar changes?
- Type system changes?
- New conversions?
- Naming conventions?
- Structure of packages, classes, and interfaces?
- Annotation locations?
- Method resolution?
- Source compatibility?
- Binary compatibility?
- Reachability or definitive assignment?

Updating Class File Consumers

- Class file consumers include
 - JVM
 - **javap**
 - **pack200/unpack200**
 - ASM, BCEL, ...
- Implications
 - Even a new class file version requires a minor change
 - Exposure at Java level requires coordination

Updating Reflective APIs

- Core reflection
(**java.lang.Class** and **java.lang.reflect.***)
- Annotation Processing model (from JSR 269)
(**javax.lang.model.***)
- Doclet API
- apt API
- Java Platform Debugger Architecture (JPDA)

Example: Enums

- Update the Java Language Spec.
- Implementation in a compiler
- Add any essential library support
- Write tests
- Update the JVM Spec.
- Update the JVM and class file tools
- Update JNI
- Update the reflective APIs
- Update serialization support
- Update javadoc output

Enum changes

- Update the Java Language Spec.
Are these equivalent?

```
//Implementation 1
enum MetaSyntacticVariable {
    FOO(21),
    BAR(42);
    private int answer;
    MetaSyntacticVariable(int answer) {
        this.answer = answer;
    }

    public int answer() {
        return answer;
    }
}
```

```
//Implementation 2
enum MetaSyntacticVariable {
    FOO {
        public int answer() {
            return 21;
        }
    },
    BAR {
        public int answer() {
            return 42;
        }
    }

    public abstract int answer();
}
```

- Implementation in a compiler
*Tricky rules around **final** and **abstract***

More enum changes

- Add any essential library support
java.lang.Enum
EnumSet and **EnumMap**
java.lang.EnumConstantNotPresentException
- Update the JVM Spec.
ACC_ENUM flag
- Update the JVM and class file tools

Still more enum changes

- Update the reflective APIs

The prolonged tale of `java.lang.Class.isEnum()`

```
public boolean isEnum() {  
    // An enum must both directly extend java.lang.Enum  
    // and have the ENUM bit set; classes for specialized  
    // enum constants don't do the former.  
    return (this.getModifiers() & ENUM) != 0 &&  
        this.getSuperclass() == java.lang.Enum.class;}  

```

- *Third time is the charm!*
- Interaction with `Constructor.newInstance`
- Are synthesized methods
 public static T[] values();
 public static T valueOf(String)
synthetic?

Enum changes, will they ever end?

- Update serialization support
Special support in normal serialization machinery
What about IIOP serialization?
- Update javadoc output
Should take enum modifier rules into account
- Whew!

What is a small change?

- *Simultaneously* small enough in
 - Specification
 - Implementation
 - Testing
- If small in all those ways, should by at most a small barrier to developer adoption

Sizing up past changes

- Normal maintenance: Tiny
- Hexadecimal floating-point literals Very small
- for-each loop Small
- static import
(but more complicated than expected) Small
- enum types Medium
- autoboxing and unboxing Medium
- Annotation types Large
- Generics **Huge**

Too big for this time

- BGGGA closures
- Properties
- Reification

Project Coin: Initial Proposals

- Strings in switch (and class literals too?)
- More concise calls to constructors with type parameters
- Exceptions enhancements:
multi-catch and final rethrow
- Ability to call methods with exotic names
- (Possibly) Bracket notation to access collections

Switch Example

- Strings can be constants, case labels require constants, so allow strings as case labels!
- Improves consistency

Replace

```
if (x.equals("foo") ||  
    x.equals("bar")) {  
    return m1(x);  
} else if (x.equals("baz")) {  
    return m2(x);  
} else {  
    return m3(x);  
}
```

With

```
switch(x) {  
    case "foo":  
    case "bar":  
        return m1(x);  
    case "baz":  
        return m2(x);  
    default  
        return m3(x);  
}
```

Constructor Example

- Repeating type parameters on the left hand side and right side of a variable initialization is verbose
- Allow variables to be listed once; improves clarity

Replace

```
Map<String, TypeElement> unmatched =  
new HashMap<String, TypeElement>(size);
```

with

```
Map<String, TypeElement> unmatched = new HashMap<>(size);
```

Exception Issues

- Often duplicated error handling in separate catch causes
- Catching superclass can lead to bad semantics; catching **Throwable** rarely correct!
- Allow multiple exceptions to be caught; improves conciseness

Exception Enhancement Examples

Replace

```
try {  
    foo();  
} catch (Exception1 ex) {  
    doSomething();  
} catch (Exception2 ex) {  
    doSomething();  
}
```

with

```
try {  
    foo();  
} catch (Exception1 | Exception2 ex) {  
    doSomething();  
}
```

Exotic Names

- Support system programming with JSR 292 facilities
 - Allow methods in non-Java languages to be called easily

```
return this.#'methodWithFunnyName!'();
```

Bracket Notation for Collections?

- Collections very commonly used and have many advantages of arrays (immutability, growable, works with generics, etc.)
- *But*, arrays have the better bracket notation []
- Allow bracket notations for collections too?

Collections Example

Replace

```
v = map.get(foo)
```

with

```
v = map[foo];
```

Replace

```
e = list.get(i)
```

with

```
e = list[i];
```

Replace

```
map.put(k, v)
```

with

```
map[k] = v;
```

Replace

```
list.set(i, e)
```

with

```
list[i] = e;
```

- Details of a proposal matter
- Possible conflict with API independence

Project Coin: Duke wants you!

- Open call for proposals phase beginning soon
 - Fill out a language change proposal form
 - Forms will be sent to an `openjdk.java.net` mailing list for discussion
- Participate in selection process to reveal final five (or so) proposals.
 - Submit ideas
 - **Provide analysis**
 - **Write prototypes**

4032740: (Reflection)Accessibilities of members should be ACCESS-LEVEL sensitive.
4041479: String conversion needs another index entry
4071299: Compilation spec needed: should state if private should imply final for a method
4088441: long double support
4093687: Extension of 'Interface' definition to include class (static) methods.
4171368: VerifyError vs. ArrayStoreException for astore
4191243: Method forwarding - Subclassing without subtyping
4206170: Cloning inner classes
4213096: A Proposal for User-Defined Lightweight Value-Semantics Objects
4223287: Suggestion for "atomic" keyword for try {} blocks
4228585: Explicit (language) support for properties
4230600: Ease the JLS to allow mathematical symbols in indentifiers
4260488: RFE: Please add nested /* */ comments
4262078: Maximum method size is too small (64Kb)
4269827: Allow multiple constants and ranges in case labels
4300222: RFE: Object Oriented Callbacks
4344759: Allow Unicode versions of operators
4351453: `final` variables and inner classes
4364486: Innocuous comment text can appear as ill-formed Unicode escape
4384324: java really needs a complex (number) primitive type
4395264: Allow this/super in certain static contexts
4396260: misleadingly incomplete xref in JLS2 15.28 Constant Expression
4421728: Specification required for mapping from class names to file names
4432337: Catching multiple exceptions simultaneously
4449383: Support For 'Design by Contract', beyond "a simple assertion facility"
4456057: package access for interface methods
4472509: Add support for vertabim string literals
4487555: Java Generics Support should support primitive types
4617197: RFE: Add Immutable types to Java
4630147: silly JLS rule that restricts the throw clause in a hiding method
4632701: Add extensible arrays and dictionaries to the syntax
4649007: Document classfile attributes for compilation date/time + debugging
4652857: Rethink Java FP spec
4662939: wanted: constant pool entries for arrays
4713248: spurious constraint in 8.8.5.1 re: anon classes and enclosing instances
4726340: RFE: Tail Call Optimization
4728767: Inner Classes Need to Support Static
4748349: Multiple return values from a method
4750181: javac should use grammar from JLS sections 4, 6-10, 14, 15
4809540: RFE: permit allocation of new objects on stack
4877954: RFE: Special syntax for core interfaces
4879776: Constructor type inference (JSR14 + JSR65 ++)
4888664: RFE: syntactic sugar to simulate C++ stack-allocated object destructor semantics
4905919: RFE: Operator overloading
4963452: RFE: 64 bit pointers needed
4983159: Typedef (alias)
4988583: AutomaticException chaining for exception in try and exception thrown in finally
4998315: JVM should allow nested classes free access to private members
5008260: @Override should be generalized from 'superclass' to 'supertype'
5009476: Compound assignments don't work well with Byte, Short
5012262: Using Strings and Objects in Switch case statements.
5014235: Closures support instead of anonymous inner classes
5025288: Language support for literal numbers in binary and other bases
5025326: Clarification needed around subroutine recursive calls
5043025: Access to Field, Method and Constructor without the use of Strings
5044723: imports versus erasure of inner classes
5052951: Wildcard should have multiple bounds similar to type variables
5052956: Type variables should have lower/super bounds
5060259: Please introduce a name for the "null" type
5061325: Add language support for functors (function objects)
5066813: Allow annotations to appear more than once on the same element
5092615: RFE: Expression Class Support
5092855: RFE:Change language specification for easier writing of Regular Expressions
5098163: Add reification of generic type parameters to the Java programming language
5108147: Extending finally to allow optional Throwable exception parameter
6176774: arrays should implement Iterable
6179333: Allow guaranteed disposing of iterators in enhanced for loop
6184881: Object.getClass() typing rule could be improved
6193725: Improved method forwarding (requires change to JVM)
6207924: Non-null types
6260924: JLS 15.12.1 should specify that type of Primary may not be primitive
6264216: Overly aggressive application of JLS3 rules to cyclic type references in annotation by javac
6312085: The for/in statement should support Iterators
6324535: Allow static anonymous classes
6350706: Multiply-bounded reference type expressions
350731: Annotation system prohibits certain annotations, drops others
6371674: inference: Inferred types' bounds should help infer unconstrained type variables
6407464: BigInteger should support Autoboxing.6
6412514: Message "except.never.thrown.in.try" have to be reported in extra cases
6427253: Support for Method Level Annotation Inheritance
6433012: A simplified syntax for using Runnable
6444738: need access to method parameter names when processing annotations at runtime
6463976: Add Lambda, Thunk, Predicate, etc., to complement Runnable
6469144: RFE: consider adding traits to Java
6479372: Add self types (type of 'this' aka ThisClass) to the language
6480391: JLS3 5.1.10 ? means ? extends Object
6488666: inference: lub of typevariables does not yield expected results
6500704: Add language support for converting methods to interfaces
6512103: Annotation-qualified optional disallowance of raw generic types
6514490: final fields can be read before being definitely assigned
6519124: RFE: Add Keyword Management
6534270: Support "suppresses" new keyword in method signatures
6557326: Generics RFE: allow <X,Y>foo(), instead of this.<X,Y>foo()
6569520: Omit 'return null;' for Void methods (or allow void as a type parameter)
6570766: Support for public abstract enum declaration
6594979: JVM 2ed 4.9.4 is overly broad on restricting uninitialized objects in backward jumps
6602336: instanceof bytecode instruction is too strict
6640435: inference: propagate >> and == constraints to help uninferred type vars
6643818: Eager-linking VMs should be prohibited to perform early error throwing
6644562: Re-specify membership of intersection types
6674900: Clarify the difference between a Type Variable and a Type Parameter
6674901: Please clarify ReferenceType
6690688: Enhanced-for-loop translation as described in JLS 14.14.2 is not type-safe
6707036: Need SE Platform Spec to list supported classfile versions
6718388: Missing rules for intersection types
6719603: Require NoClassDefFoundError to chain original exception, add JCK test
6720843: Clarify that the throw statement can take null
6742436: Improved Parallelism in the Java Runtime Environment
6746458: writing libraries in Java for non-Java languages requires support for exotic identifiers
6754038: writing libraries in Java for non-Java languages requires support for invokedynamic sites
6775422: OSGi & Dependency Injection
6785114: Return type inference (15.12.2.8) doesn't work with autoboxing
6785612: Generifying a class breaks compatibility wrt >1.5 clients

4032740: (Reflection)Accessibilities of members should be ACCESS-LEVEL sensitive.
4041479: String conversion needs another index entry
4071299: Compilation spec needed: should state if private should imply final for a method
4088441: long double support
4093687: Extension of 'Interface' definition to include class (static) methods.
4171368: VerifyError vs. ArrayStoreException for astore
4191243: Method forwarding - Subclassing without subtyping
4206170: Cloning inner classes
4213096: A Proposal for User-Defined Lightweight Value-Semantics Objects
4223287: Suggestion for "atomic" keyword for try {} blocks
4228585: Explicit (language) support for properties
4230600: Ease the JLS to allow mathematical symbols in indentifiers
4260488: RFE: Please add nested /* */ comments
4262078: Maximum method size is too small (64Kb)
4269827: Allow multiple constants and ranges in case labels
4300222: RFE: Object Oriented Callbacks
4344759: Allow Unicode versions of operators
4351453: 'final' variables and inner classes
4364486: Innocuous comment text can appear as ill-formed Unicode escape
4384324: java really needs a complex (number) primitive type
4395264: Allow this/super in certain static contexts
4396260: misleadingly incomplete xref in JLS2 15.28 Constant Expression
4421728: Specification required for mapping from class names to file names
4432337: Catching multiple exceptions simultaneously
4449383: Support For 'Design by Contract', beyond "a simple assertion facility"
4456057: package access for interface methods
4472509: Add support for vertabim string literals
4487555: Java Generics Support should support primitive types
4617197: RFE: Add Immutable types to Java
4630147: silly JLS rule that restricts the throw clause in a hiding method
4632701: Add extensible arrays and dictionaries to the syntax
4649007: Document classfile attributes for compilation date/time + debugging
4652857: Rethink Java FP spec
4662939: wanted: constant pool entries for arrays
4713248: spurious constraint in 8.8.5.1 re: anon classes and enclosing instances
4726340: RFE: Tail Call Optimization
4728767: Inner Classes Need to Support Static
4748349: Multiple return values from a method
4750181: javac should use grammar from JLS sections 4, 6-10, 14, 15
4809540: RFE: permit allocation of new objects on stack
4877954: RFE: Special syntax for core interfaces
4879776: Constructor type inference (JSR14 + JSR65 ++)
4888664: RFE: syntactic sugar to simulate C++ stack-allocated object destructor semantics
4905919: RFE: Operator overloading
4963452: RFE: 64 bit pointers needed
4983159: Typedef (alias)
4988583: AutomaticException chaining for exception in try and exception thrown in finally
4998315: JVM should allow nested classes free access to private members
5008260: @Override should be generalized from 'superclass' to 'supertype'
5009476: Compound assignments don't work well with Byte, Short
5012262: Using Strings and Objects in Switch case statements.
5014235: Closures support instead of anonymous inner classes
5025288: Language support for literal numbers in binary and other bases
5025326: Clarification needed around subroutine recursive calls
5043025: Access to Field, Method and Constructor without the use of Strings
5044723: imports versus erasure of inner classes
5052951: Wildcard should have multiple bounds similar to type variables
5052956: Type variables should have lower/super bounds
5060259: Please introduce a name for the "null" type
5061325: Add language support for functors (function objects)
5066813: Allow annotations to appear more than once on the same element
5092615: RFE: Expression Class Support
5092855: RFE: Change language specification for easier writing of Regular Expressions
5098163: Add reification of generic type parameters to the Java programming language
5108147: Extending finally to allow optional Throwable exception parameter
6176774: arrays should implement Iterable
6179333: Allow guaranteed disposing of iterators in enhanced for loop
6184881: Object.getClass() typing rule could be improved
6193725: Improved method forwarding (requires change to JVM)
6207924: Non-null types
6260924: JLS 15.12.1 should specify that type of Primary may not be primitive
6264216: Overly aggressive application of JLS3 rules to cyclic type references in annotation by javac
6312085: The for/in statement should support Iterators
6324535: Allow static anonymous classes
6350706: Multiply-bounded reference type expressions
6350731: Annotation system prohibits certain annotations, drops others
6371674: inference: Inferred types' bounds should help infer unconstrained type variables
6407464: BigInteger should support Autoboxing.6
6412514: Message "except.never.thrown.in.try" have to be reported in extra cases
6427253: Support for Method Level Annotation Inheritance
6433012: A simplified syntax for using Runnable
6444738: need access to method parameter names when processing annotations at runtime
6463976: Add Lambda, Thunk, Predicate, etc., to complement Runnable
6469144: RFE: consider adding traits to Java
6479372: Add self types (type of 'this' aka ThisClass) to the language
6480391: JLS3 5.1.10 ? means ? extends Object
6488666: inference: lub of typevariables does not yield expected results
6500704: Add language support for converting methods to interfaces
6512103: Annotation-qualified optional disallowance of raw generic types
6514490: final fields can be read before being definitely assigned
6519124: RFE: Add Keyword Management
6534270: Support "suppresses" new keyword in method signatures
6557326: Generics RFE: allow <X,Y>foo(), instead of this.<X,Y>foo()
6569520: Omit 'return null;' for Void methods (or allow void as a type parameter)
6570766: Support for public abstract enum declaration
6594979: JVM5 2ed 4.9.4 is overly broad on restricting uninitialized objects in backward jumps
6602336: instanceof bytecode instruction is too strict
6640435: inference: propagate >> and == constraints to help uninferred type vars
6643818: Eager-linking VMs should be prohibited to perform early error throwing
6644562: Re-specify membership of intersection types
6674900: Clarify the difference between a Type Variable and a Type Parameter
6674901: Please clarify ReferenceType
6690688: Enhanced-for-loop translation as described in JLS 14.14.2 is not type-safe
6707036: Need SE Platform Spec to list supported classfile versions
6718388: Missing rules for intersection types
6719603: Require NoClassDefFoundError to chain original exception, add JCK test
6720843: Clarify that the throw statement can take null
6742436: Improved Parallelism in the Java Runtime Environment
6746458: writing libraries in Java for non-Java languages requires support for exotic identifiers
6754038: writing libraries in Java for non-Java languages requires support for invokedynamic sites
6775422: OSGi & Dependency Injection
6785114: Return type inference (15.12.2.8) doesn't work with autoboxing
6785612: Generifying a class breaks compatibility wrt >1.5 clients

Q&A

Project Coin: Small Language Changes for JDK 7

Joseph D. Darcy
joe.darcy@sun.com
<http://blogs.sun.com/darcy>

Backup Slides

Hexadecimal Floating-Point Literals

- Handy to have textual representation that is
 - Human readable
 - Clearly unambiguous
 - Mappable to floating-point format

- Examples

$(2-2^{-52}) \cdot 2^{1023}$ as **0x1.fffffffffffffp1023**

2^{-1074} as **0x1.0P-1074** or **0x0.0000000000000001P-1022**