Evolving the Java™ Language

Mark Reinhold <mr@sun.com>
Java SE Chief Engineer
Sun Microsystems
Contents

Java Language Principles
Language-Level XML
Other Language Ideas
A New JVM Bytecode
Contents

Java Language Principles
Language-Level XML
Other Language Ideas
A New JVM Bytecode
Philosophy
Philosophy

Language != Collection<Feature>
Philosophy

Style Matters
Philosophy

Style Matters
Philosophy
Learning from C++
Learning from C++

POWER
Learning from C++

POWER

Flexibility
Learning from C++

\[ a = b \]
Learning from C++

\[ a \ll b \]
Java Language Principles

“Programs should be written for people to read, and only incidentally for machines to execute.”

Gerald Jay Sussman & Hal Abelson
Structure and Interpretation of Computer Programs
Java Language Principles

Readability
Simplicity
Universality
Evolving the Language
Evolving the Language

“First, do no harm.”

Hippocrates
Evolving the Language

2030
Evolving the Language

Developer
Productivity
Non-features

const
Macros
Continuations
Multiple Dispatch
Multiple Inheritance
Multiple Return Values
Operator Over-Loading
Aspect-Oriented Programming
Evolving the Platform

Java Platform ≠ Java Language
Evolving the Platform

Java Platform ≠ Java Language
Evolving the Platform

JScheme  AspectJ  ??
JRuby  Jython  Groovy

Java  JavaScript  BeanShell

Java  VM  Standard Libraries
Contents

Java Language Principles

Language-Level XML

Other Language Ideas

A New JVM Bytecode
An Example

<feature>
   <id>29</id>
   <rfe>4057701</rfe>
   <votes>649</votes>
   <name>Method to find free disk space</name>
   <engineer>iris.garcia</engineer>
   <release>mustang</release>
   <state>submitted</state>
</feature>
An Example

<feature>
   <id>29</id>
   <rfe>4057701</rfe>
   <votes>649</votes>
   <name>Method to find free disk space</name>
   <engineer>iris.garcia</engineer>
   <release>mustang</release>
   <state>submitted</state>
   <reviewed>
      <who>graham.hamilton</who>
      <when>2004-11-07T13:44:25</when>
   </reviewed>
</feature>
An Example

```java
void addReviewer(Element feature,
                 String user, String time)
{
    ...
}
```
void addReviewer(Element feature, 
       String user, String time)
{
    Document doc = feature.getOwnerDocument();
    Element review = doc.createElement("reviewed");
    Element who = doc.createElement("who");
    who.setTextContent(user);
    review.appendChild(who);
    Element when = doc.createElement("when");
    when.setTextContent(time);
    review.appendChild(when);
    feature.appendChild(review);
}
void addReviewer(Element feature,
        String user, String time)
{
    feature
        .addContent(new Element("reviewed"))
        .addContent(new Element("who")
            .addContent(user))
        .addContent(new Element("when")
            .addContent(time));
}
An Example: JAXB

```java
void addReviewer(Feature feature,
                 String user,
                 XMLGregorianCalendar time)
{
    Review review = new Review();
    review.setWho(user);
    review.setWhen(time);
    feature.setReviewed(review);
}
```
An Example: JAXB

```xml
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:complexType name="Review">
    <xsd:sequence>
      <xsd:element name="who" type="xsd:string"/>
      <xsd:element name="when" type="xsd:dateTime"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="Feature">
    <xsd:sequence>
      <xsd:element name="id" type="xsd:integer"/>
      <xsd:element name="rfe" type="xsd:integer"/>
      <xsd:element name="votes" type="xsd:integer" minOccurs="0"/>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="engineer" type="xsd:string"/>
      <xsd:element name="release" type="xsd:string"/>
      <xsd:element name="state" type="xsd:string"/>
      <xsd:element name="reviewed" type="Review" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="feature" type="Feature"/>
</xsd:schema>
```
An Example: JAXB

```java
void addReviewer(Feature feature,
                 String user,
                 XMLGregorianCalendar time)
{
    Review review = new Review();
    review.setWho(user);
    review.setWhen(time);
    feature.setReviewed(review);
}
```
void addReviewer(Element feature,
                   String user, String time)
{
    ...
}

Can We Do Better?
void addReviewer(Element feature,
    String user, String time)
{
    feature.add(<reviewed><who>{ user } </who>

        <when>{ time } </when>

    </reviewed>);
}
Language-Level XML

java.lang.XML
Language-Level XML

java.lang.String    java.lang.XML

""                <>
void addReviewer(XML feature,
                   String user, String time)
{
    feature.add(<reviewed>
                  <who><![CDATA[{ user }]]></who>
                  <when><![CDATA[{ time }]]></when>
                </reviewed>);
}
Selection

<feature>
    <id>29</id>
    <rfe>4057701</rfe>
    <votes>649</votes>
    <name>Method to find free disk space</name>
    <engineer>iris.garcia</engineer>
    <release>mustang</release>
    <state>submitted</state>
    <reviewed>
        <who>graham.hamilton</who>
        <when>2004-11-07T13:44:25</when>
    </reviewed>
</feature>
Selection

<feature>
  <id>29</id>
  <rfe>4057701</rfe>
  <votes>649</votes>
  <name>Method to find free disk space</name>
  <engineer>iris.garcia</engineer>
  <release>mustang</release>
  <state>submitted</state>
  <reviewed>
    <who>graham.hamilton</who>
    <when>2004-11-07T13:44:25</when>
  </reviewed>
</feature>
Selection

```java
String getState(XML feature) {
    return feature.________;
}
```
String getState(XML feature) {
    return feature.#state;
}
Mutation

<feature>
  <id>29</id>
  <rfe>4057701</rfe>
  <votes>649</votes>
  <name>Method to find free disk space</name>
  <engineer>iris.garcia</engineer>
  <release>mustang</release>
  <state>submitted</state>
  <reviewed>
    <who>graham.hamilton</who>
    <when>2004-11-07T13:44:25</when>
  </reviewed>
</feature>
Mutation

<feature>
  <id>29</id>
  <rfe>4057701</rfe>
  <votes>649</votes>
  <name>Method to find free disk space</name>
  <engineer>iris.garcia</engineer>
  <release>mustang</release>
  <state>approved</state>
  <reviewed>
    <who>graham.hamilton</who>
    <when>2004-11-07T13:44:25</when>
  </reviewed>
</feature>
Mutation

```java
String getState(XML feature) {
    return feature.#state;
}

void approve(XML feature) {
    feature.#state = "approved";
}
```
Iteration

<feature-list>
  <feature>
    <id>29</id>
    <name>Method to find free disk space</name>
    <engineer>iris.garcia</engineer>
    <state>approved</state>
  </feature>
  <feature>
    <id>201</id>
    <name>Improve painting (fix gray boxes)</name>
    <engineer>scott.violet</engineer>
    <state>approved</state>
  </feature>
  <feature>
    <id>42</id>
    <name>Zombie references</name>
    <engineer>mark.reinhold</engineer>
    <state>submitted</state>
  </feature>
</feature-list>
Iteration

```java
void list(XML featureList) {
```
Iteration

void list(XML featureList) {
    for (XML f : featureList.#feature)
void list(XML featureList) {
    for (XML f : featureList.#feature)
        System.out.printf("%3s: %s\n", f.#id, f.#name);
}
void list(XML featureList) {
    for (XML f : featureList.#feature)
        System.out.printf("%3s: %s%n", f.#id, f.#name);
}

29: Method to find free disk space
201: Improve painting (fix gray boxes)
42: Zombie references
void rejectOpenFeatures(XML featureList) {
    for (XML f : featureList.#feature) {
        if (!f.#state.equals("approved"))
            f.#state = "rejected";
    }
}
<feature-list>
  <feature>
    <id>29</id>
    <name>Method to find free disk space</name>
    <engineer>iris.garcia</engineer>
    <state>approved</state>
  </feature>
  <feature>
    <id>201</id>
    <name>Improve painting (fix gray boxes)</name>
    <engineer>scott.violet</engineer>
    <state>approved</state>
  </feature>
  <feature>
    <id>42</id>
    <name>Zombie references</name>
    <engineer>mark.reinhold</engineer>
    <state>rejected</state>
  </feature>
</feature-list>
Descendants

List<XML> getSortedIds(XML featureList) {
    List<XML> ids = new ArrayList<XML>();
    for (XML f : featureList)
        ids.add(f.#id);
    return Collections.sort(ids);
}
Descendants

```java
List<XML> getSortedIds(XML featureList) {
    List<XML> ids = new ArrayList<XML>();
    for (XML f : featureList)
        ids.add(f.#id);
    List<XML> ids = featureList..#id;
    return Collections.sort(ids);
}
```
Language-Level XML

java.lang.String java.lang.XML

" <>
DOM Requirements

Model contents, not syntax
Well-formed documents only
Gracefully handle large documents
Leverage relevant language features

As simple as possible, but not simpler
The Micro-DOM
class XML { // An element
    Name name();
    Map<Name, Attribute> attributes();
    List<Content> children();
}
The Micro-DOM

class XML {  // An element
    Name name();
    Map<Name, Attribute> attributes();
    List<Content> children();
}

class Name {
    URI ns();  // null => no namespace
    String name();
    static Name of(URI ns, String name);
}
The Micro-DOM

class XML {
    // An element
    Name name();
    Map<Name, Attribute> attributes();
    List<Content> children();
}
The Micro-DOM

class XML { // An element
    Name name();
    Map<Name,Attribute> attributes();
    List<Content> children();
}

class Attribute { // An attribute
    Name name();
    String value();
    Attribute set(String v);
}
The Micro-DOM

class XML { // An element
    Name name();
    Map<Name, Attribute> attributes();
    List<Content> children();
}

class Attribute { // An attribute
    Name name();
    String value();
    Attribute set(String v);
}

class Text // Some text
    implements CharSequence
{
    Text set(CharSequence cs);
    Text add(CharSequence cs);
}
The Micro-DOM

class XML { // An element
    Name name();
    Map<Name,Attribute> attributes();
    List<Content> children();
}

class Attribute { // An attribute
    Name name();
    String value();
    Attribute set(String v);
}

class Text { ... } // Some text

class Content { } // Elements | text
The Micro-DOM

class XML extends Content { // An element
    Name name();
    Map<Name,Attribute> attributes();
    List<Content> children();
}

class Attribute { // An attribute
    Name name();
    String value();
    Attribute set(String v);
}

class Text extends Content { ... } // Some text

class Content { } // Elements | text
class XML extends Content { // An element
    Name name();
    Map<Name,Attribute> attributes();
    List<Content> children();
}

class Attribute { // An attribute
    Name name();
    String value();
    Attribute set(String v);
}

class Text extends Content { ... } // Some text

class Content { } // Elements | text

class Node { // Attributes | elements | text
    XML parent();
    Node detach();
}
The Micro-DOM

class XML extends Content {
    // An element
    Name name();
    Map<Name,Attribute> attributes();
    List<Content> children();
}

class Attribute extends Node {
    // An attribute
    Name name();
    String value();
    Attribute set(String v);
}

class Text extends Content {
    // Some text
}

class Content extends Node {
    // Elements | text
}

class Node {
    // Attributes | elements | text
    XML parent();
    Node detach();
}
The Micro-DOM

class XML extends Content { // An element
    Name name();
    Map<Name,Attribute> attributes();
    List<Content> children();
    List<Node> nodes();
}

class Attribute extends Node { // An attribute
    Name name();
    String value();
    Attribute set(String v);
}

class Text extends Content { ... } // Some text

class Content extends Node { } // Elements | text

class Node { // Attributes | elements | text
    XML parent();
    Node detach();
}
The Micro-DOM

class XML extends Content { // An element
    Name name();
    Map<Name,Attribute> attributes();
    List<Content> children();
    List<Node> nodes();
}
class Attribute extends Node { // An attribute
    Name name();
    String value();
    Attribute set(String v);
}
class Text extends Content { ... } // Some text

class Content extends Node { } // Elements | text

class Node { // Attributes | elements | text
    XML parent();
    Node detach();
}
Language-Level XML

Syntax
Mutability
Validation
Namespaces
Typechecking
Document types
Processing instructions

Coming soon to a JSR near you!
Contents

Java Language Principles
Language-Level XML
Other Language Ideas
A New JVM Bytecode
Other Language Ideas

Still in the Brainstorming Phase ...

- Friends
  - Not all public APIs are meant for outside use
  - Want richer model, finer control
  - Tightly tied to JSR 277: Java Modules
Other Language Ideas

Still in the Brainstorming Phase ...

- Friends
  - Not all public APIs are meant for outside use
  - Want richer model, finer control
  - Tightly tied to JSR 277: Java Modules

- Method references/Closures
  - Some love them
  - Some hate them
  - Many still don’t see the point
Method References/Closures

Advantages are greatest on day one

- Powerful libraries
  - Map, reduce, *etc.*
- User-defined control constructs
  - Assert
  - Foreach
  - Try-finally-finally-finally-finally-...
  - If-but-unless-maybe
Method References/Closures

Costs are at least as high today as on day one
- Lose most big advantages
- Multiple ways of doing things
  - Iterators
  - Inner classes
- Restricted subsets under consideration
  - Main goal: Simplify GUI event-handling code
Contents

Java Language Principles
Language-Level XML
Other Language Ideas
A New JVM Bytecode
Support for Other Languages

**What**
- Some people program in several languages
  - Especially scripting languages
- We want to improve JVM support for these

**Why**
- Useful for Java platform users for certain tasks
- Broaden community

**How**
- New `invokedynamic` byte code
Support for Other Languages

- Today, implementations of other languages simply ride existing JVMs
  - JavaScript, Jython, Kawa, JRuby, Groovy, ...
- Easy for single inheritance, single dispatch, statically typed OOPLs
- The real interest is in languages that are different
  - Scripting languages are all dynamically typed
  - Many have multiple inheritance or mixins
- This can be challenging to do well
Method Invocation

invokevirtual
invokeinterface
invokestatic
invokespecial
invokevirtual

- General form is:
  - `invokevirtual TargetObjectType.methodDescriptor`
  
  `MethodDescriptor -> methodName(ArgTypes) ReturnType`

- Very close to Java language semantics
  - Only overloading (and generics) left to javac
  - Single inheritance
  - Single dispatch
  - Statically typed: TargetObjectType is constant

- Verifier ensures that types are correct
newSize(c) {
    // Collection has grown; figure out
    // the next increment in size
    return c.size() * c.growthFactor()
}
And Here My Troubles Began

```java
newSize(c) {
    // Collection has grown; figure out
    // the next increment in size
    return c.size() * c.growthFactor()
}
```

How do we compile method invocations?
And Here My Troubles Began

newSize(c) {
    // Collection has grown; figure out
    // the next increment in size
    return c.size() * c.growthFactor()
}

How do we compile method invocations?

invokevirtual
    UnknownType.growthFactor() UnknownReturnType
And Here My Troubles Began

```java
newSize(c) {
    // Collection has grown; figure out
    // the next increment in size
    return
    ((Interface91)(((Interface256)c)).size())
    * ((Interface91)(((Interface42)c)).
growthFactor());
}
```

Solutions are complex,
involving many synthetic interfaces and casts
Solution: invokedynamic

A loosely typed invokevirtual

- Target type need not be statically known
  - VM determines type at each invocation
  - No need for a host of synthetic interfaces

- Actual arguments need not be statically known to match method descriptor
  - Instead, cast at invocation time to ensure integrity
Only a Partial Solution

- No direct support for multiple inheritance or multiple dispatch
  - General support is hard
  - Each language has its own rules
- Calling platform libraries from scripting languages brings additional problems
  - How do you resolve overloading?
- These issues become easier to address once you have invokedynamic
Summary

- The Java programming language will evolve
  - But don't expect fundamental changes
- Value as an easily-understood lingua franca is key
  - Anything undermining that is unlikely to happen
- We are serious about language-level XML
- Additional features are under consideration
- The Java platform will support additional languages
  - Experiment with new stuff in that context
Thank you!

Questions?

Mark Reinhold
<mr@sun.com>