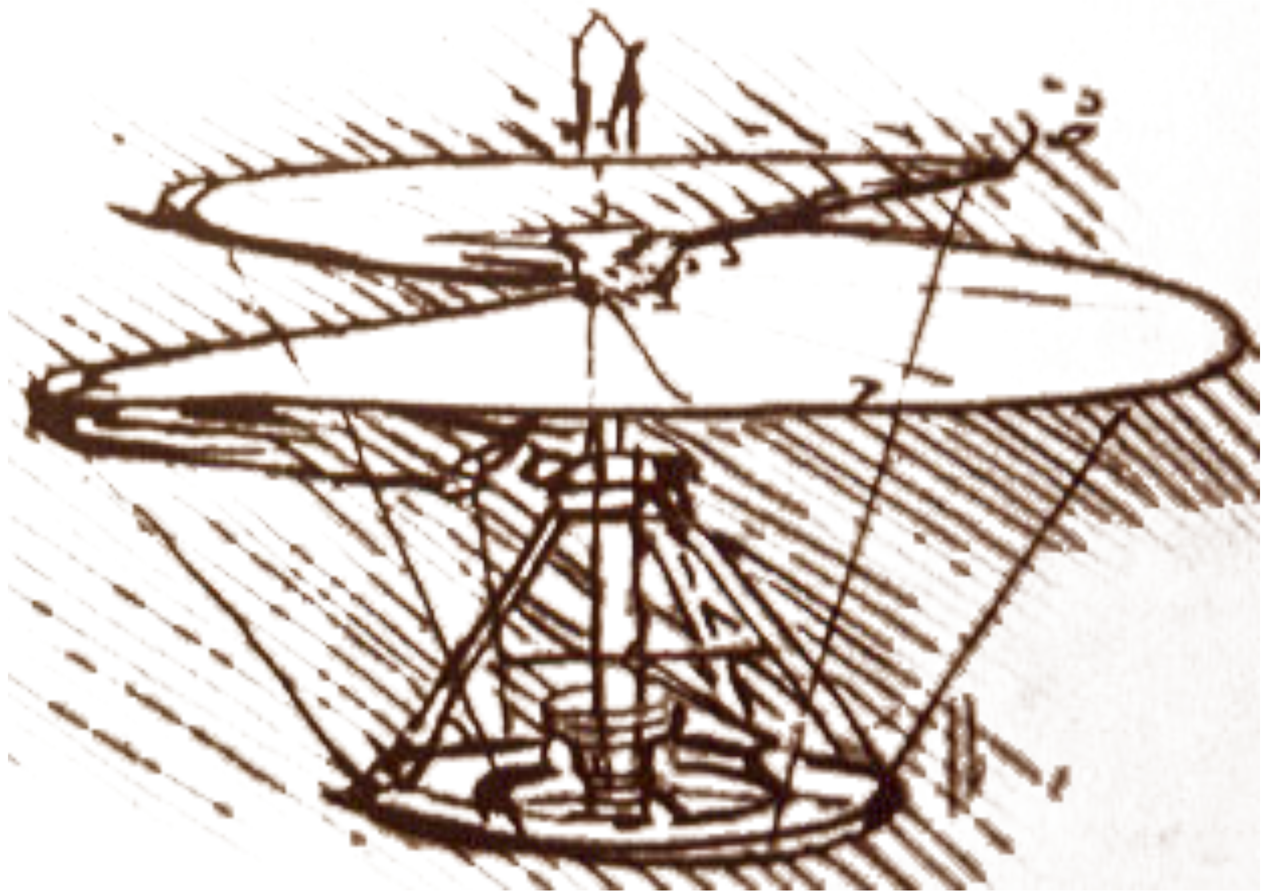
# Future of the Java platform



Ola Bini
JRuby Core Developer
ThoughtWorks Studios

ola.bini@gmail.com
http://olabini.com/blog

## About me

## About me

Ola Bini

## About me

Ola Bini

Works for ThoughtWorks

## About me

- Ola Bini
- Works for ThoughtWorks
- From Sweden - duh

## About me

Ola Bini

Works for ThoughtWorks

From Sweden - duh

Involved with several languages on the JVM:

## About me

- Ola Bini
- Works for ThoughtWorks
- From Sweden - duh
- Involved with several languages on the JVM:
  - Ioke

## About me

- Ola Bini
- Works for ThoughtWorks
- From Sweden - duh
- Involved with several languages on the JVM:
  - Ioke
  - JRuby

## About me

- Ola Bini
- Works for ThoughtWorks
- From Sweden - duh
- Involved with several languages on the JVM:
  - Ioke
  - JRuby
  - Jatha

## About me

- Ola Bini
- Works for ThoughtWorks
- From Sweden - duh
- Involved with several languages on the JVM:
  - Ioke
  - JRuby
  - Jatha
- Member of the JSR292 expert group

## About me

- Ola Bini
- Works for ThoughtWorks
- From Sweden - duh
- Involved with several languages on the JVM:
  - Ioke
  - JRuby
  - Jatha
- Member of the JSR292 expert group

## About me

- Ola Bini
- Works for ThoughtWorks
- From Sweden - duh
- Involved with several languages on the JVM:
  - Ioke
  - JRuby
  - Jatha
- Member of the JSR292 expert group

## Java as a platform

## Java as a platform

Other languages

# Java as a platform

- Other languages
- Libraries

## Java as a platform

Other languages

Libraries

Platform independence

## Java as a platform

Other languages

Libraries

Platform independence

Higher level abstractions

# Java as a platform

Other languages

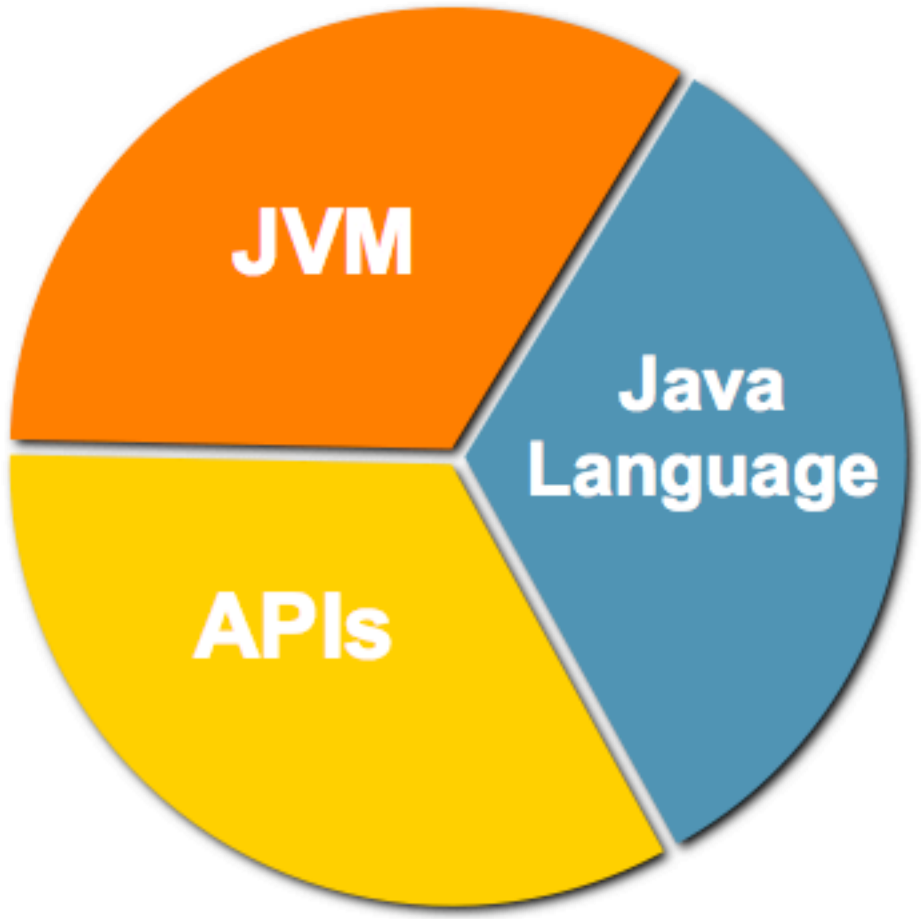
Libraries

Platform independence

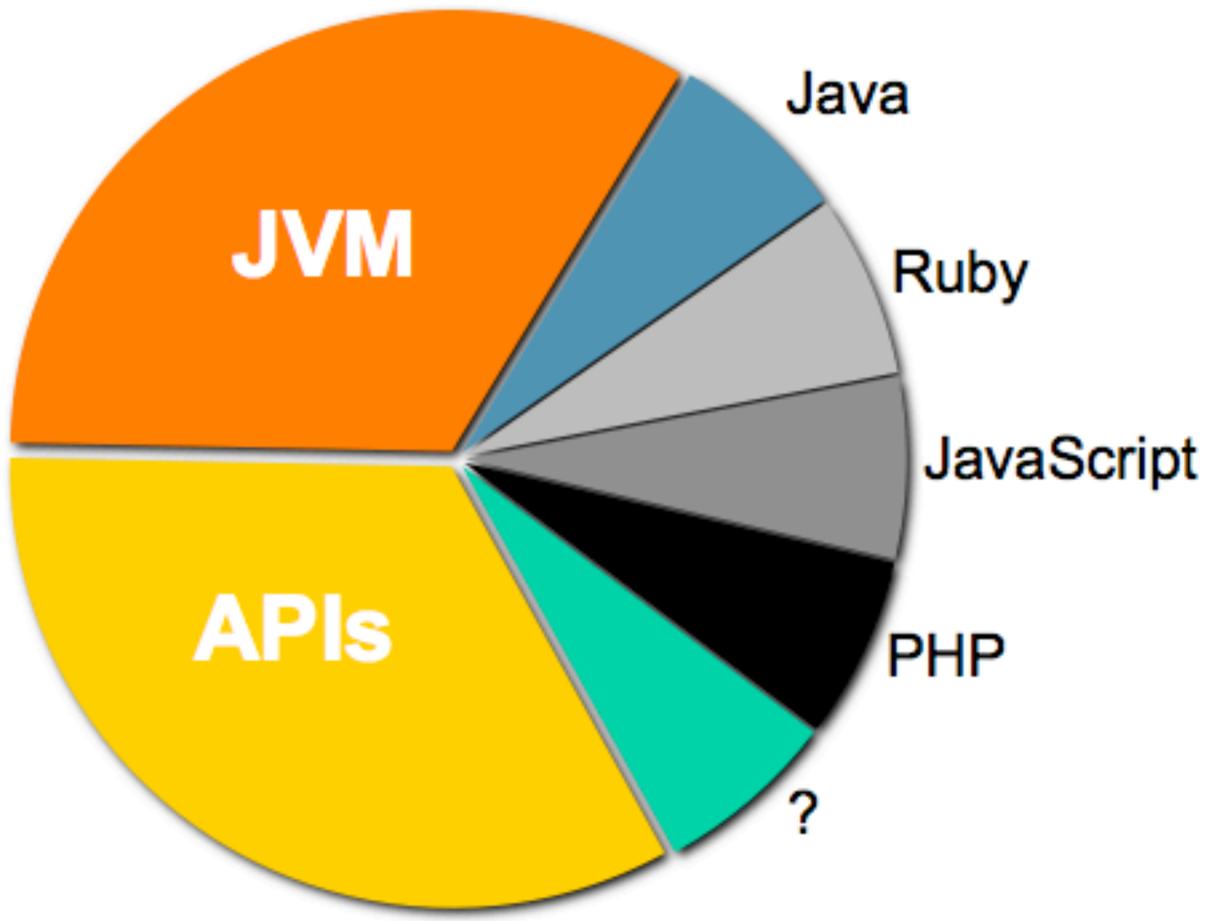
Higher level abstractions

Java the language as systems language?

## The Java platform?



## The Java platform?



# Other languages

| | | | |
|---|---|---|---|
| Hecl | HotScheme | tuProlog | WLShell |
| Jacl | webLISP | JLog | JudoScript |
| Clojure | Jaja | LL | JRuby |
| Ync/Javascript | JScheme | javalog | Jickle |
| JoyJ | Skij | SmallWorld | Rhino |
| v-language | Kawa | Bistro | BeanShell |
| CAL | uts | Talks2 | Resin |
| Aardappel | JBasic | Obol | Jython |
| Funnel | Mapyrus | Groovy | Pnuts |
| Mini | CONVERT | Nice | Janino |
| PLAN | HotTEA | Scala | Join Java |
| Sixx | COCOA | Anvil | JMatch |
| BDC Scheme | NetLogo | dSelf | iScript |
| ABCL | StarLogo | Hojo | Yassl |
| Lili | AJLogo | Correlate | Yoix |
| Jatha | Turtle Tracks | MetaJ | W4F |
| Bigloo | rLogo | Sather | PERCobol |
| SISC | Yoyo | Quercus | Bex Script |
| Lisp | TermWare | FScript | Demeter/Java |
| PS3i | XProlog | Sleep | CKI Prolog |

## Other languages: Clojure

## Other languages: Clojure

Lisp dialect - code as data

# Other languages: Clojure

Lisp dialect - code as data

Designed for the JVM

## Other languages: Clojure

Lisp dialect - code as data

Designed for the JVM

Powerful macros

## Other languages: Clojure

Lisp dialect - code as data

Designed for the JVM

Powerful macros

Good interoperability with Java

# Other languages: Clojure

Lisp dialect - code as data

Designed for the JVM

Powerful macros

Good interoperability with Java

Functional programming language

## Other languages: Clojure

Lisp dialect - code as data

Designed for the JVM

Powerful macros

Good interoperability with Java

Functional programming language

Concurrency

## Other languages: Groovy

## Other languages: Groovy

Dynamic, strongly typed

## Other languages: Groovy

Dynamic, strongly typed

Object oriented

## Other languages: Groovy

- Dynamic, strongly typed
- Object oriented
- Designed for the JVM

## Other languages: Groovy

Dynamic, strongly typed

Object oriented

Designed for the JVM

Inspired by Python, Ruby and Smalltalk

## Other languages: Groovy

Dynamic, strongly typed

Object oriented

Designed for the JVM

Inspired by Python, Ruby and Smalltalk

Good integration with Java

## Other languages: Scala

## Other languages: Scala

Multiparadigm language

## Other languages: Scala

- Multiparadigm language
  - Object orientedness

## Other languages: Scala

- Multiparadigm language
  - Object orientedness
  - Functional programming

## Other languages: Scala

- Multiparadigm language
  - Object orientedness
  - Functional programming
- Designed for the JVM

## Other languages: Scala

- Multiparadigm language
  - Object orientedness
  - Functional programming
- Designed for the JVM
- Concurrency: Immutability and actors

## Other languages: Scala

- Multiparadigm language
  - Object orientedness
  - Functional programming
- Designed for the JVM
- Concurrency: Immutability and actors
- Includes many advanced language features

## Other languages: Scala

- Multiparadigm language
  - Object orientedness
  - Functional programming
- Designed for the JVM
- Concurrency: Immutability and actors
- Includes many advanced language features
  - Pattern matching, closures, parametric polymorphism

## Other languages: Scala

- Multiparadigm language
  - Object orientedness
  - Functional programming
- Designed for the JVM
- Concurrency: Immutability and actors
- Includes many advanced language features
  - Pattern matching, closures, parametric polymorphism
  - Sequence comprehensions, mixins, infix or postfix statements

## The Java Virtual Machine

## The Java Virtual Machine

The JVM is a great virtual machine

## The Java Virtual Machine

The JVM is a great virtual machine

Flexible online code loading (with safe bytecodes)

# The Java Virtual Machine

The JVM is a great virtual machine

Flexible online code loading (with safe bytecodes)

GC & object structure

# The Java Virtual Machine

The JVM is a great virtual machine

Flexible online code loading (with safe bytecodes)

GC & object structure

- Mature and provies lots of algorithms and parameters

## The Java Virtual Machine

- The JVM is a great virtual machine
- Flexible online code loading (with safe bytecodes)
- GC & object structure
  - Mature and provies lots of algorithms and parameters
- Reflective access to classes and objects

## The Java Virtual Machine

- The JVM is a great virtual machine
- Flexible online code loading (with safe bytecodes)
- GC & object structure
  - Mature and provies lots of algorithms and parameters
- Reflective access to classes and objects
- Tools (JMM, JVMTI, dtrace)

## The Java Virtual Machine

- The JVM is a great virtual machine
- Flexible online code loading (with safe bytecodes)
- GC & object structure
    - Mature and provies lots of algorithms and parameters
- Reflective access to classes and objects
- Tools (JMM, JVMTI, dtrace)
- Good libraries and a useful language to write more

## The Java Virtual Machine

## The Java Virtual Machine

Optimizing Just-In-Time compiler

## The Java Virtual Machine

Optimizing Just-In-Time compiler

Clever performance techniques

# The Java Virtual Machine

- Optimizing Just-In-Time compiler
- Clever performance techniques
    - Type inference

## The Java Virtual Machine

- Optimizing Just-In-Time compiler
- Clever performance techniques
  - Type inference
  - Customization

## The Java Virtual Machine

- Optimizing Just-In-Time compiler
- Clever performance techniques
  - Type inference
  - Customization
  - Profiling

## The Java Virtual Machine

- Optimizing Just-In-Time compiler
- Clever performance techniques
  - Type inference
  - Customization
  - Profiling
  - Deoptimizing

# The Java Virtual Machine

- Optimizing Just-In-Time compiler
- Clever performance techniques
  - Type inference
  - Customization
  - Profiling
  - Deoptimizing
  - Fast/slow paths

## The Java Virtual Machine

- Optimizing Just-In-Time compiler
- Clever performance techniques
  - Type inference
  - Customization
  - Profiling
  - Deoptimizing
  - Fast/slow paths
- The JVM is <u>mature</u>

## Needs of higher level languages

# Needs of higher level languages

Very late binding (runtime linking, typing, code gen)

## Needs of higher level languages

- Very late binding (runtime linking, typing, code gen)
- Automatic storage management (GC)

## Needs of higher level languages

Very late binding (runtime linking, typing, code gen)

Automatic storage management (GC)

Environmental queries (reflection, stack walking)

## Needs of higher level languages

Very late binding (runtime linking, typing, code gen)

Automatic storage management (GC)

Environmental queries (reflection, stack walking)

Exotic primitives (tail calls, bignums, call/cc)

## Needs of higher level languages

Very late binding (runtime linking, typing, code gen)

Automatic storage management (GC)

Environmental queries (reflection, stack walking)

Exotic primitives (tail calls, bignums, call/cc)

Code management integrated with execution

## Needs of higher level languages

Very late binding (runtime linking, typing, code gen)

Automatic storage management (GC)

Environmental queries (reflection, stack walking)

Exotic primitives (tail calls, bignums, call/cc)

Code management integrated with execution

Robust handling of incorrect inputs

# Needs of higher level languages

Very late binding (runtime linking, typing, code gen)

Automatic storage management (GC)

Environmental queries (reflection, stack walking)

Exotic primitives (tail calls, bignums, call/cc)

Code management integrated with execution

Robust handling of incorrect inputs

Helpful runtime support libraries (regexps, math)

## Needs of higher level languages

Very late binding (runtime linking, typing, code gen)

Automatic storage management (GC)

Environmental queries (reflection, stack walking)

Exotic primitives (tail calls, bignums, call/cc)

Code management integrated with execution

Robust handling of incorrect inputs

Helpful runtime support libraries (regexps, math)

A compiler that understands it all

# Needs of higher level languages

Very late binding (runtime linking, typing, code gen)

Automatic storage management (GC)

Environmental queries (reflection, stack walking)

Exotic primitives (tail calls, bignums, call/cc)

Code management integrated with execution

Robust handling of incorrect inputs

Helpful runtime support libraries (regexps, math)

A compiler that understands it all

## What’s missing?

## What’s missing?

Dynamic invocation

## What’s missing?

Dynamic invocation

Lightweight method objects

## What's missing?

Dynamic invocation

Lightweight method objects

Lightweight bytecode loading

## What's missing?

Dynamic invocation

Lightweight method objects

Lightweight bytecode loading

Continuations and stack introspection

## What's missing?

Dynamic invocation

Lightweight method objects

Lightweight bytecode loading

Continuations and stack introspection

Tail calls and tail recursion

## What’s missing?

Dynamic invocation

Lightweight method objects

Lightweight bytecode loading

Continuations and stack introspection

Tail calls and tail recursion

Tuples and value-oriented types

## What’s missing?

Dynamic invocation

Lightweight method objects

Lightweight bytecode loading

Continuations and stack introspection

Tail calls and tail recursion

Tuples and value-oriented types

Immediate wrapper types

## What's missing?

Dynamic invocation

Lightweight method objects

Lightweight bytecode loading

Continuations and stack introspection

Tail calls and tail recursion

Tuples and value-oriented types

Immediate wrapper types

## Dynamic invocation

## Dynamic invocation

Non-Java call site in the bytecodes

## Dynamic invocation

Non-Java call site in the bytecodes

Language-specific handler

## Dynamic invocation

Non-Java call site in the bytecodes

Language-specific handler

Determines linking at runtime

## Dynamic invocation

Non-Java call site in the bytecodes

Language-specific handler

Determines linking at runtime

Works in a reflective style

## Dynamic invocation

Non-Java call site in the bytecodes

Language-specific handler

Determines linking at runtime

Works in a reflective style

Installs direct (non-reflective) methods

## Dynamic invocation

Non-Java call site in the bytecodes

Language-specific handler

Determines linking at runtime

Works in a reflective style

Installs direct (non-reflective) methods

Stateful: can be updated or revoked over time

## Dynamic invocation

Non-Java call site in the bytecodes

Language-specific handler

Determines linking at runtime

Works in a reflective style

Installs direct (non-reflective) methods

Stateful: can be updated or revoked over time

Any dynamic language will benefit greatly

## Symbolic freedom

## Symbolic freedom

Allow any identifier as name

## Symbolic freedom

Allow any identifier as name

JVM identifiers originally based on the Java language

## Symbolic freedom

Allow any identifier as name

JVM identifiers originally based on the Java language

No real reason for this

## Symbolic freedom

Allow any identifier as name

JVM identifiers originally based on the Java language

No real reason for this

Support for Ruby style names

# Symbolic freedom

Allow any identifier as name

JVM identifiers originally based on the Java language

No real reason for this

Support for Ruby style names

- empty?

## Symbolic freedom

- Allow any identifier as name
- JVM identifiers originally based on the Java language
- No real reason for this
- Support for Ruby style names
  - empty?
  - value=

## Symbolic freedom

- Allow any identifier as name
- JVM identifiers originally based on the Java language
- No real reason for this
- Support for Ruby style names
  - empty?
  - value=
  - clear!

## Symbolic freedom

- Allow any identifier as name
- JVM identifiers originally based on the Java language
- No real reason for this
- Support for Ruby style names
  - empty?
  - value=
  - clear!
- Canonical name mangling

## Closures

## Closures

Several closure proposals right now

## Closures

Several closure proposals right now

All of them will benefit from the previous ideas

## Closures

Several closure proposals right now

All of them will benefit from the previous ideas

But it isn't required

## Closures

Several closure proposals right now

All of them will benefit from the previous ideas

But it isn't required

Most of the machinery for closures is already in place

## Closures

Several closure proposals right now

All of them will benefit from the previous ideas

But it isn't required

Most of the machinery for closures is already in place

It's just a question of deciding ...

## The DaVinci machine

## The DaVinci machine

Evolutionary adaptation of the present JVM

## The DaVinci machine

- Evolutionary adaptation of the present JVM
- Open-ended experiment

## The DaVinci machine

- Evolutionary adaptation of the present JVM
- Open-ended experiment
    - Wild ideas are considered, but must prove useful

## The DaVinci machine

- Evolutionary adaptation of the present JVM
- Open-ended experiment
  - Wild ideas are considered, but must prove useful
  - While incubating, features are disabled by default

## The DaVinci machine

- Evolutionary adaptation of the present JVM
- Open-ended experiment
  - Wild ideas are considered, but must prove useful
  - While incubating, features are disabled by default
- Eventual convergence

## The DaVinci machine

- Evolutionary adaptation of the present JVM
- Open-ended experiment
  - Wild ideas are considered, but must prove useful
  - While incubating, features are disabled by default
- Eventual convergence
- Prototype JVM extensions to run non-Java languages efficiently

## The DaVinci machine

- Evolutionary adaptation of the present JVM
- Open-ended experiment
  - Wild ideas are considered, but must prove useful
  - While incubating, features are disabled by default
- Eventual convergence
- Prototype JVM extensions to run non-Java languages efficiently
- First class architectural support (no hack or side-cars)

## The DaVinci machine

- Evolutionary adaptation of the present JVM
- Open-ended experiment
  - Wild ideas are considered, but must prove useful
  - While incubating, features are disabled by default
- Eventual convergence
- Prototype JVM extensions to run non-Java languages efficiently
- First class architectural support (no hack or side-cars)
- New languages to co-exist gracefully with Java

## The DaVinci machine

## The DaVinci machine

Most of the features mentioned above have or will be implemented here

## The DaVinci machine

Most of the features mentioned above have or will be implemented here

Will eventually decide what makes it in Java 7

## The DaVinci machine

Most of the features mentioned above have or will be implemented here

Will eventually decide what makes it in Java 7

Why?

## The DaVinci machine

- Most of the features mentioned above have or will be implemented here
- Will eventually decide what makes it in Java 7
- Why?
  - Language implementers know what they want

## The DaVinci machine

Most of the features mentioned above have or will be implemented here

Will eventually decide what makes it in Java 7

Why?

- Language implementers know what they want
  - and how to simulate it at 100x slowdown

## The DaVinci machine

- Most of the features mentioned above have or will be implemented here
- Will eventually decide what makes it in Java 7
- Why?
  - Language implementers know what they want
    - and how to simulate it at 100x slowdown
  - VM implementers know what VMs can do

## The DaVinci machine

- Most of the features mentioned above have or will be implemented here
- Will eventually decide what makes it in Java 7
- Why?
  - Language implementers know what they want
    - and how to simulate it at 100x slowdown
  - VM implementers know what VMs can do
  - Let's bring them together

## JSR 292

## JSR 292

Supporting dynamically type languages

## JSR 292

Supporting dynamically type languages

Main features

## JSR 292

- Supporting dynamically type languages
- Main features
    - invoke_dynamic

## JSR 292

- Supporting dynamically type languages
- Main features
  - invoke_dynamic
  - Method handles

## JSR 292

- Supporting dynamically type languages
- Main features
    - invoke_dynamic
    - Method handles
    - Hotswapping

## JSR 292

- Supporting dynamically type languages
- Main features
  - invoke_dynamic
  - Method handles
  - Hotswapping
- Representatives from JRuby, Groovy, Jython, among others

# JSR 292

- Supporting dynamically type languages
- Main features
  - invoke_dynamic
  - Method handles
  - Hotswapping
- Representatives from JRuby, Groovy, Jython, among others
- Focus on VM support

## The JVM languages group

## The JVM languages group

Focus on library level support for languages running on the JVM

## The JVM languages group

Focus on library level support for languages running on the JVM

Discussions about current pain points

## The JVM languages group

Focus on library level support for languages running on the JVM

Discussions about current pain points

Meta-object protocol

## The JVM languages group

Focus on library level support for languages running on the JVM

Discussions about current pain points

Meta-object protocol

Java method overload resolution at runtime

## The JVM languages group

Focus on library level support for languages running on the JVM

Discussions about current pain points

Meta-object protocol

Java method overload resolution at runtime

Representatives from Java, JRuby, Jython, Groovy, Pnuts, Ioke, Scala, Clojure, Nice, Ng, and many more

## Java 7

## Java 7

Probably early 2010

## Java 7

Probably early 2010

Provisional:

## Java 7

- Probably early 2010
- Provisional:
  - Modularization

## Java 7

- Probably early 2010
- Provisional:
  - Modularization
  - JSR 292 - Dynamic languages

# Java 7

Probably early 2010

Provisional:

- Modularization
- JSR 292 - Dynamic languages
- JSR 203 - Better I/O support, asynch I/O, revamped file system API

## Java 7

- Probably early 2010
- Provisional:
  - Modularization
  - JSR 292 - Dynamic languages
  - JSR 203 - Better I/O support, asynch I/O, revamped file system API
  - Safe rethrow

## Java 7

Probably early 2010

Provisional:

- Modularization
- JSR 292 - Dynamic languages
- JSR 203 - Better I/O support, asynch I/O, revamped file system API
- Safe rethrow
- Null dereferencing

## Java 7

- Probably early 2010
- Provisional:
    - Modularization
    - JSR 292 - Dynamic languages
    - JSR 203 - Better I/O support, asynch I/O, revamped file system API
    - Safe rethrow
    - Null dereferencing
    - Better type inferencing for generics

## Java 7

- Probably early 2010
- Provisional:
  - Modularization
  - JSR 292 - Dynamic languages
  - JSR 203 - Better I/O support, asynch I/O, revamped file system API
  - Safe rethrow
  - Null dereferencing
  - Better type inferencing for generics
  - Multi catch

## Java 7

- Probably early 2010
- Provisional:
    - Modularization
    - JSR 292 - Dynamic languages
    - JSR 203 - Better I/O support, asynch I/O, revamped file system API
    - Safe rethrow
    - Null dereferencing
    - Better type inferencing for generics
    - Multi catch
    - Better Swing

# Java 7

- Probably early 2010
- Provisional:
  - Modularization
  - JSR 292 - Dynamic languages
  - JSR 203 - Better I/O support, asynch I/O, revamped file system API
  - Safe rethrow
  - Null dereferencing
  - Better type inferencing for generics
  - Multi catch
  - Better Swing
  - 6u10 features - Java Kernel, Quickstarter, new plugin

# Java 7

- Probably early 2010
- Provisional:
  - Modularization
  - JSR 292 - Dynamic languages
  - JSR 203 - Better I/O support, asynch I/O, revamped file system API
  - Safe rethrow
  - Null dereferencing
  - Better type inferencing for generics
  - Multi catch
  - Better Swing
  - 6u10 features - Java Kernel, Quickstarter, new plugin

## Not in Java 7

## Not in Java 7

Closures

## Not in Java 7

Closures

Reified generics

## Not in Java 7

Closures

Reified generics

1st class propertiers

## Not in Java 7

Closures

Reified generics

1st class propertiers

Operator overloading

# Not in Java 7

Closures

Reified generics

1st class propertiers

Operator overloading

BigDecimal syntax

# Not in Java 7

Closures

Reified generics

1st class propertiers

Operator overloading

BigDecimal syntax

JSR 295 - Beans binding

## Not in Java 7

Closures

Reified generics

1st class propertiers

Operator overloading

BigDecimal syntax

JSR 295 - Beans binding

## After?

## After?

Look to C# - more advanced language features

## After?

- Look to C# - more advanced language features
  - But not as much backwards compatibility

## After?

- Look to C# - more advanced language features
    - But not as much backwards compatibility
- JavaFX

# After?

- Look to C# - more advanced language features
  - But not as much backwards compatibility
- JavaFX
- Java as a platform

## After?

- Look to C# - more advanced language features
    - But not as much backwards compatibility
- JavaFX
- Java as a platform

Q and A