


Ioke



A Folding Language

Ola Bini

computational metalinguist

ola.bini@gmail.com

<http://olabini.com/blog>



Vanity slide

From Stockholm, Sweden

ThoughtWorks

JRuby core developer

Ioke creator

... ask me about programming languages



ThoughtWorks®

ThoughtWorks

Global consulting firm

US, Canada, UK, Australia, India, China - and Scandinavia

1100 people worldwide

Ruby

Agile

Open Source

Martin Fowler

Studios - Mingle, Cruise, Twist



Goal

ThoughtWorks®

How expressive can you make a language?



What is Ioke?

An experiment

A programming language

Dynamic and strong typing

Prototype based object orientation

Homoiconic

Inspirations: Io, Ruby, Self/Smalltalk, Lisp

Hosted on the JVM and the CLR

Current version: Ioke E ikj 0.3.1, Ioke E ikc 0.1.1



Philosophy

Expressiveness over performance

Communication over implementation

Abstraction over low level interfaces

Higher order functionality over explicitness

First class over implicit functionality

“Right is better” over “Worse is better”

Language oriented programming over APIs

“Code as data” over “data as code”

Homoiconicity over syntax

Syntax over explicit API's

What is expressiveness?

Power

Is Perl 5 more powerful than Perl 4?

Ruby 1.8 vs Ruby 1.6?

Java 1.5 vs Java 1.4?

Basic vs assembler?

Cobol vs C++?

C++ vs Java?

Java vs Lisp?

Abstraction

Mapping from mind to code



Getting started

Make sure you have Java 5 or better

Download latest tar/zip from <http://ioke.org/download.html>

Unpack into \$IOKE_HOME

Add \$IOKE_HOME/bin to \$PATH

```
git clone git://github.com/olabini/ioke.git
```

```
cd ioke
```

Make sure you have ant installed

```
ant
```

Add `pwd`/bin to \$PATH



Syntax

ThoughtWorks®

No dots for method calls

Period is just that - the end of a sentence

Semicolon is the start of a comment

Most other things will remind you of Ruby



Demo
A taste

Prototype based OO

Only one kind of object

Every object has zero or more *mimics*

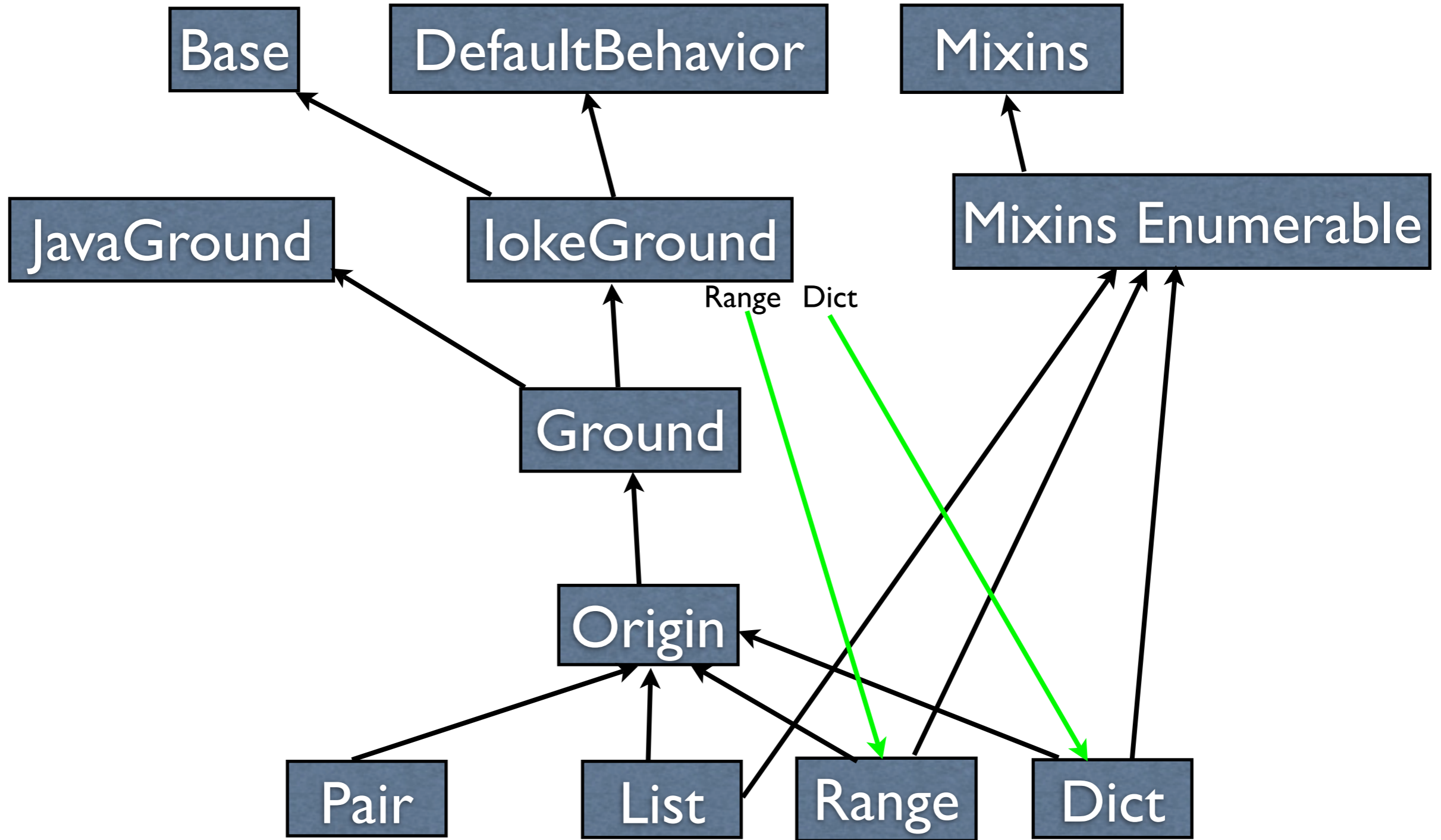
Every object has zero or more *cells*

Convention: everything that has a name that starts with a capital letter, is called a *kind*. A kind can be thought of as a class like object whose main purpose is to be the mimic of other objects.



Model

ThoughtWorks®





Methods

Positional arguments

Keyword arguments

Rest arguments

Keyword rest arguments

Splatting

Conditions

Generalization of Exceptions

Concepts:

Condition - something that needs to be signalled

Rescue - a piece of code that rescues from a condition signal

Handler - a piece of code that handles a condition signal

Restart - a piece of code that can recover from the condition

How do you handle exceptions?

Exceptions are only for errors - what about warnings?

What about other things?



Demo
Conditions



Homoiconic?

```
class Foo
  def bar(x, y)
    puts "hello"
    puts [1,2,3].map {|z| z*z}
    x*y
  end
end
```



JRuby AST (the MRI is worse)

```
RootNode
  NewlineNode
  ClassNode
    NewlineNode
    DefnNode |bar|
      ArgsPreTwoArgNode
      BlockNode
        NewlineNode
        FCallOneArgNode |puts|
          ArrayNode
            StrNode == "hello"
        NewlineNode
        FCallOneArgNode |puts|
          ArrayNode
            CallNoArgBlockNode |map|
              ArrayNode
                FixnumNode == 1
                FixnumNode == 2
                FixnumNode == 3
              IterNode
                DAsgnNode |z| &0 >0
                NilImplicitNode |nil|
              NewlineNode
              CallOneArgNode |*|
                DVarNode |z| &0 >0
              ArrayNode
                DVarNode |z| &0 >0
            NewlineNode
            CallOneArgNode |*|
              LocalVarNode |x| &0 >0
            ArrayNode
              LocalVarNode |y| &1 >0
```



The Ioke version

```
Foo = Origin mimic do(  
  bar = method(x, y,  
    "hello" println  
    [1,2,3] map(z, z*z) println  
    x*y))
```

AST:

```
=(Foo, Origin mimic do(  
  =(bar, method(x, y,  
    "hello" println  
    [] (1,2,3) map(z, z *(z)) println  
    x *(y))))))
```



Messages

Everything is a message

Message is a kind in Ioke - and can be created from scratch

A message has a name

A message has zero or more arguments

A message has a prev pointer

A message has a next pointer

This is the AST



Macros

Macro

- Activates at runtime

- Always runs

- Doesn't evaluate arguments at all

- Gives access to argument messages and caller ground

Syntax

- Activates at runtime

- Only runs the first time, then replaces itself with the result of itself

- Is expected to return a message chain

- Doesn't evaluate arguments

- Gives access to argument messages and caller ground



Aspects

before, after and around returns pointcuts

Pointcuts have methods to add and remove advice

Only affects those cells that use it

Typical usage:

Origin `around(:mimic)` defines aspect to call initialize

Implemented fully in Ioke



ISpec

ThoughtWorks®

Minimal, inspired by RSpec

A few hundred lines of code

Mocking support

Used for the Ioke test spec - currently about 3900 specs.



DokGen

Generates HTML documents

Walks the living structure of the Ioke system

Combines this structure with ISpec information



Java Integration

Support calling and using Java methods and fields

Implementing interfaces

Extending classes



Demo
Swinging
loke



ikc - Ioke CLR runtime

Runs on Mono and .NET

No CLR integration features yet



Future

Ioke P, Ioke F

Concurrency support

Units

Important libraries

IO, Sockets

Cane - Package management tool

SQL DSL?

JRuby integration

V8 runtime?

Parrot runtime?



The project

Git at Github - <http://github.com/olabini/ioke>

Git mirror at Kenai - <git://git.kenai.com/ioke~main>

Mailing lists at Kenai - <http://ioke.kenai.com>

ioke.org

- Programming guide

- Wiki

- IRC logs

- Current release Doks

IRC: *#ioke* at Freenode

Q

and

A



Q



A



^
v