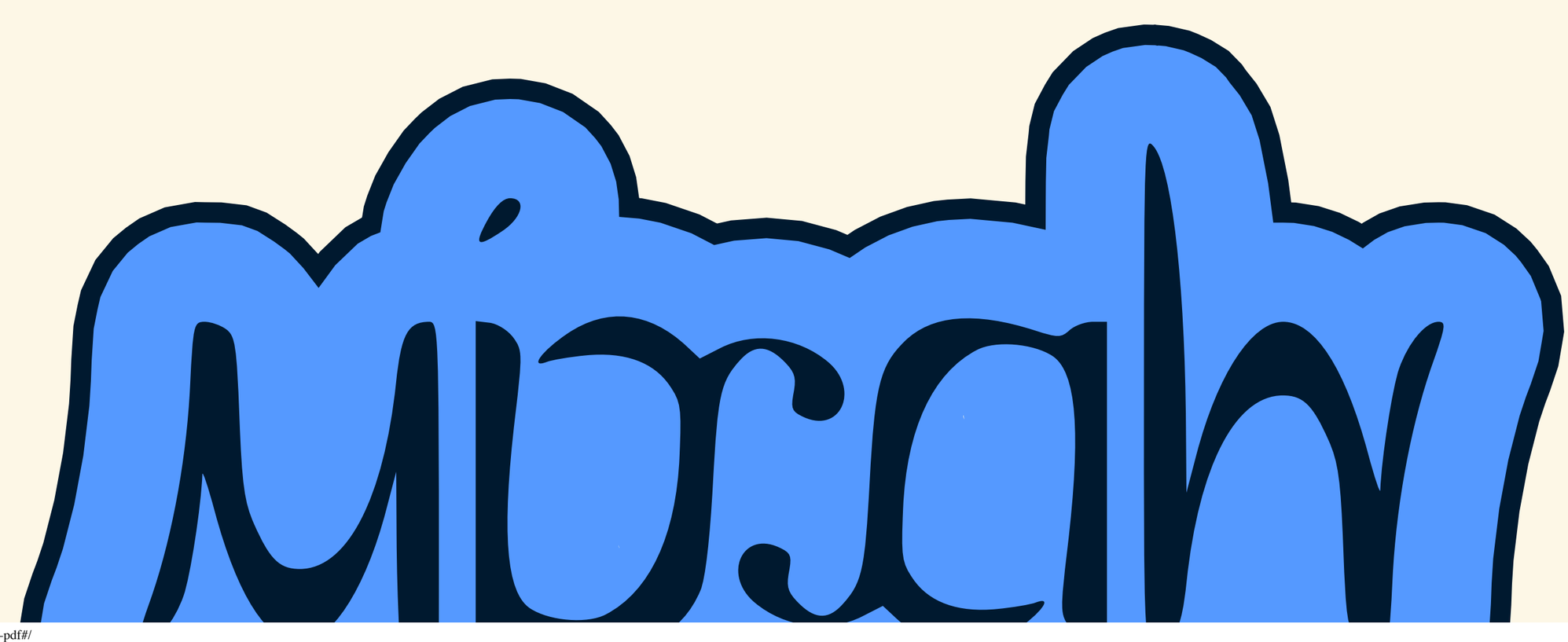
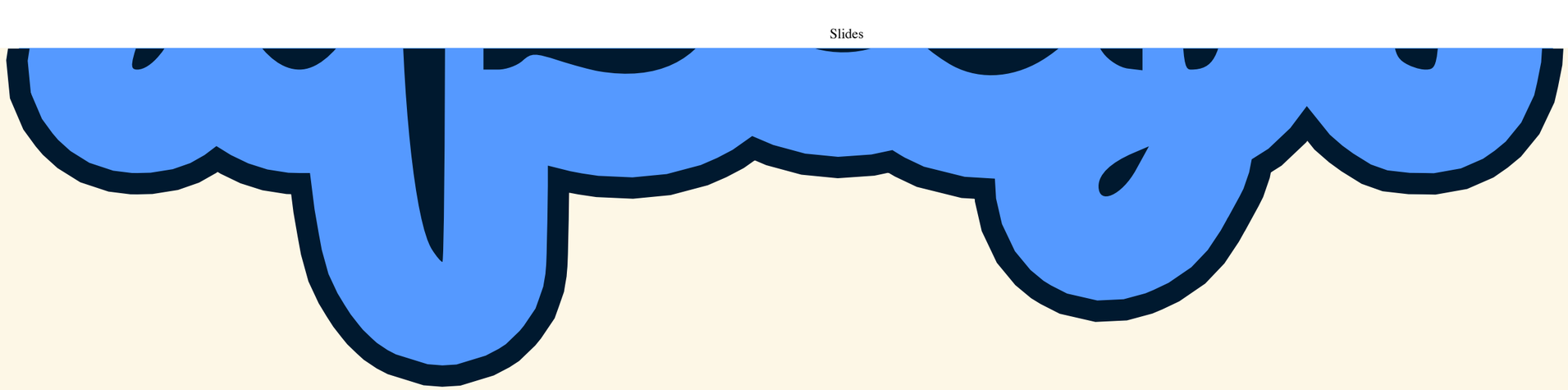
Sprinkle Mirah in Your Java





Understand Java?

You might like Mirah

Static

Ruby Syntax

```
1 puts "Hi!"
```

Java Types

```
1 package org.mirah.jvm.model
2
3 import java.util.Arrays
4 import java.util.ArrayList
5 import java.util.EnumMap
```

No Runtime Jars

Compile Time Macros

Mirah's Backstory

2009 - Duby by @headius

AppEngine and Dubious

Renamed

Mirah

Unpacking Mirah

Classic Example: Fibonacci

In Ruby,

```
1 def fib(a)
2   if a < 2
3     a
4   else
5     fib(a - 1) + fib(a - 2)
6   end
7 end
```

In Java,

```
1 public static int fib(int a) {
2     if (a < 2) {
3         return a;
4     } else {
5         return fib(a - 1) +
6                fib(a - 2);
7     }
8 }
```

In Mirah?

In Mirah,

```
1 def fib(a: int)
2   if a < 2
3     a
4   else
5     fib(a - 1) + fib(a - 2)
6   end
7 end
```

Almost identical to Ruby!

Only difference:

```
1 def fib(a: int)
```

From Ruby:

Mirah's not a statically typed Ruby

Type system:

Java's

+ Type Inference

In Mirah,

```
1 def fib(a: int)
2   if a < 2
3     a
4   else
5     fib(a - 1) + fib(a - 2)
6   end
7 end
```

```
1 public static int fib(int a) {
```

Bytecode, you say

javap mirah_fib.class

```
1  public static int fib(int);
2    Code:
3       0: iload_0
4       1: iconst_2
5       2: if_icmpge     9
6       5: iload_0
7       6: goto          22
8       9: iload_0
9      10: iconst_1
10     11: isub
11     12: invokestatic  #15                 // Method fib:(I)I
12     15: iload_0
13     16: iconst_2
14     17: isub
15     18: invokestatic  #15                 // Method fib:(I)I
16     21: iadd
17     22: ireturn
```

And Java?

javap JavaFib.class

```
1 public static int fib(int);
2   Code:
3      0: iload_0
4      1: iconst_2
5      2: if_icmpge     7
6      5: iload_0
7      6: ireturn
8      7: iload_0
9      8: iconst_1
10     9: isub
11    10: invokestatic  #2                  // Method fib:(I)I
12    13: iload_0
13    14: iconst_2
14    15: isub
15    16: invokestatic  #2                  // Method fib:(I)I
16    19: iadd
17    20: ireturn
```

~Same as Mirah's

No Runtime library

No runtime metaprogramming

Not new

It's a remix

familiar Ruby syntax

How to get Mirah?

Release Archive

github.com/mirah/mirah/releases

unzip and add mirah/bin to your path

Just the jar?

Also, Maven

```
<dependency>
  <groupId>org.mirah</groupId>
  <artifactId>mirah</artifactId>
  <version>0.1.4</version>
</dependency>
```

Use The Source

github.com/mirah/mirah

```
$ mirah -e 'puts "hello, JavaONE!"'
```

```
$ mirah fib.mirah
102334155
```

```
$ mirahc fib.mirah # =>
fib.class
```

Virgo

Add Mirah to your project's build

مرحبا

Maven!

Pom it up

```
<plugin>
  <groupId>org.mirah.maven</groupId>
  <artifactId>maven-mirah-
plugin</artifactId>
  <version>1.2-SNAPSHOT</version>
  <executions>
    <execution>
      <phase>compile</phase>
      <goals><goal>compile</goal></goals>
    </execution>
  </executions>
</plugin>
```

src/main/mirah

Runs after java compile

```
[INFO] --- maven-compiler-plugin:2.3.2:compile (default-compile)
@ my-app ---
[INFO] Compiling 1 source file to
/Users/nick/mycoolproject/target/classes
[INFO]
[INFO] --- maven-mirah-plugin:1.2-SNAPSHOT:compile (default) @
my-app ---
```

Java's got good tooling

Java's got excellent tooling

Mirah uses Java's static types

Static typing

IDEs IDEs IDEs

Netbeans

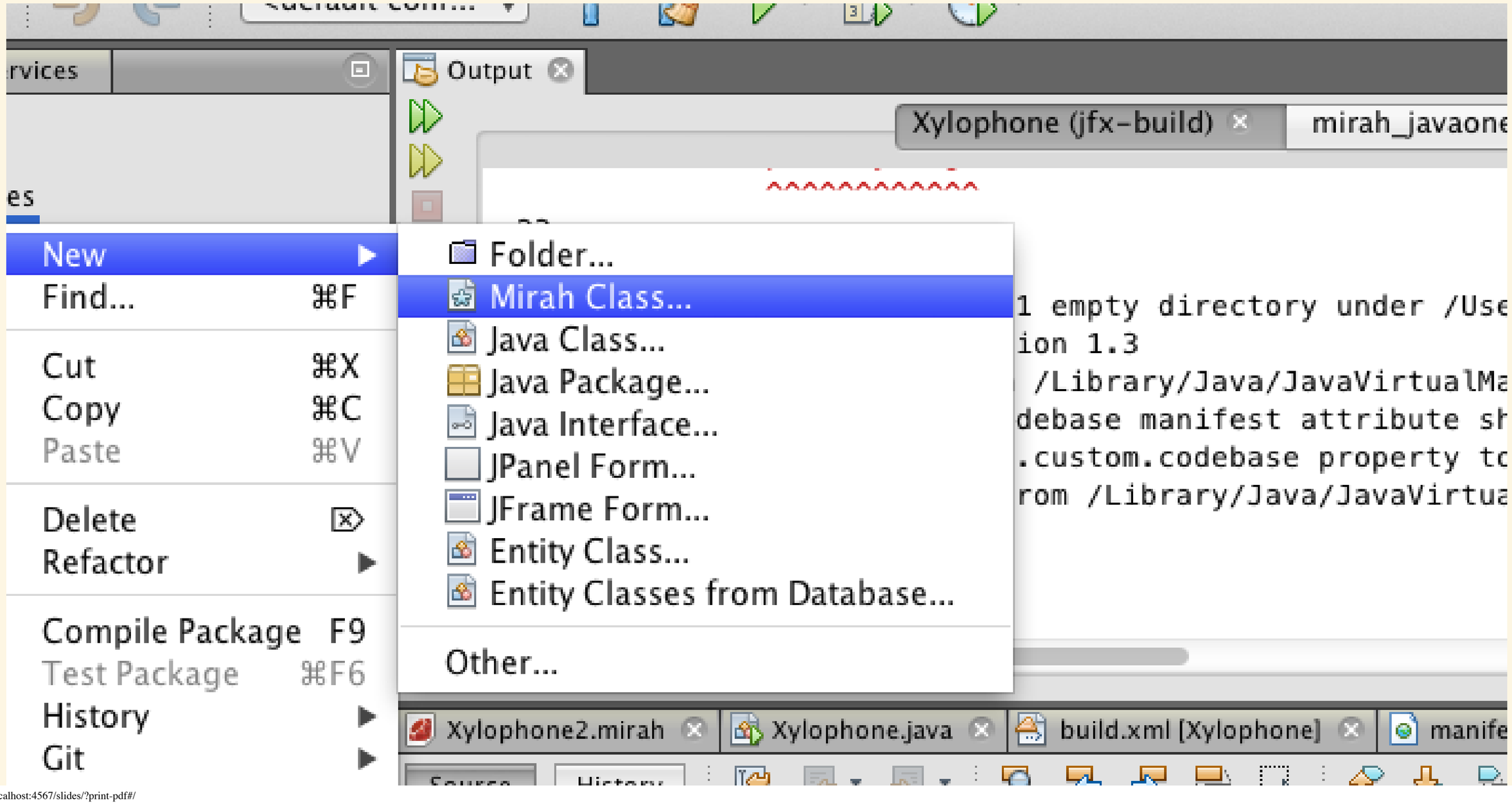
Steve Hannah rocks!

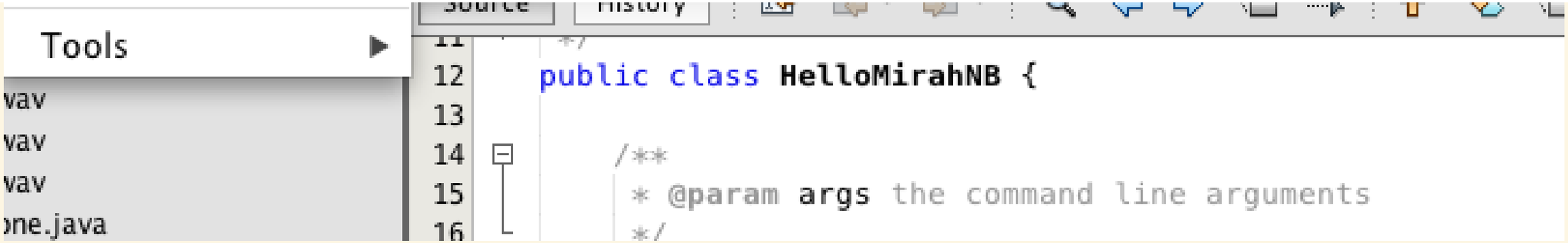
Mirah for Netbeans

Featureful, yet still alpha!

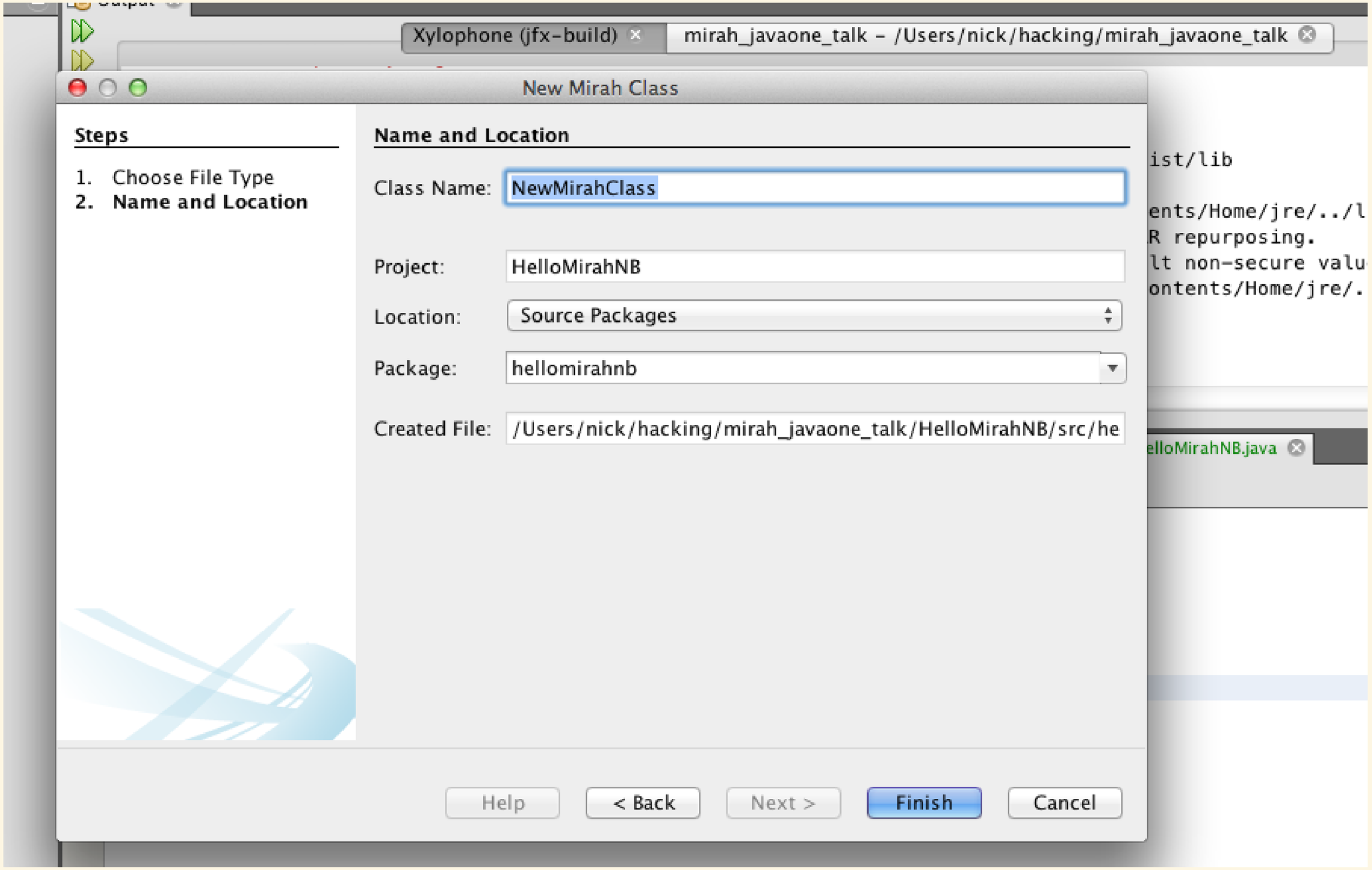
It's got

Wizards


Output
Xylophone (jfx-build)
mirah_javaone
New
Find... ⌘F
Cut ⌘X
Copy ⌘C
Paste ⌘V
Delete
Refactor
Compile Package F9
Test Package ⌘F6
History
Git
Folder...
Mirah Class...
Java Class...
Java Package...
Java Interface...
JPanel Form...
JFrame Form...
Entity Class...
Entity Classes from Database...
Other...
1 empty directory under /Use
ion 1.3
/Library/Java/JavaVirtualMa
debase manifest attribute sh
.custom.codebase property to
rom /Library/Java/JavaVirtua
Xylophone2.mirah
Xylophone.java
build.xml [Xylophone]


Tools
vav
vav
vav
one.java
Source
History
*/
public class HelloMirahNB {
/**
* @param args the command line arguments
*/

Wizards


Xylophone (jfx-build)
mirah_javaone_talk - /Users/nick/hacking/mirah_javaone_talk
New Mirah Class
Steps
1. Choose File Type
2. Name and Location
Name and Location
Class Name: NewMirahClass
Project: HelloMirahNB
Location: Source Packages
Package: hellomirahnb
Created File: /Users/nick/hacking/mirah_javaone_talk/HelloMirahNB/src/he
Help
< Back
Next >
Finish
Cancel
ist/lib
ents/Home/jre/../l
R repurposing.
lt non-secure valu
ontents/Home/jre/.
elloMirahNB.java

Wizards

NewMirahClass.mirah

Source Visual History

```
1  /*
2   * To change this license header, choose License Headers in Project Propert
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6
7  package hellomirahnb
8
9  /**
10  *
11  * @author nick
12  */
13 class NewMirahClass
14
15 end
16
17
```

It's got

Syntax Highlighting

Source Visual History

```
8
9   class Greeter
10    def initialize name: String
11      @name = name
12    end
13    def greet
14      puts "Hello, #{name
15    end
16  end
17  puts "Hello from Mirah"
18
19  Greeter.new("JavaOne").greet
20
21
```

It's got code completion!

Mirah to Mirah completion

```
7   package net.tomitamb
8
9   class Greeter
10    def initialize name: String
11      @name = name
12    end
13    def greet
14      puts "Hello, #{name}"
15    end
16  end
17  puts "Hello from Mirah"
18
19  Greeter.
20          new(String )
```

```
21
```

Mirah to Mirah completion

```
 7  package hellomirahnb
 8
 9  class Greeter
10    def initialize name: String
11      @name = name
12    end
13    def greet
14      puts "Hello, #{name}"
15    end
16  end
17  puts "Hello from Mirah"
18
19  Greeter.new("JavaOne").gr
20                          greet:PrintStream
21
```

h

Java method completion

```
10    def initialize name: St
11      @name = name
12    end
13    def greet
14      puts "Hello, #{name}"
15    end
16
17    def shout
18      puts "Hello, #{name}".to
19    end
20  end
21  puts "Hello from Mirah"
22
23  Greeter.new("JavaOne").gr
24
25
```

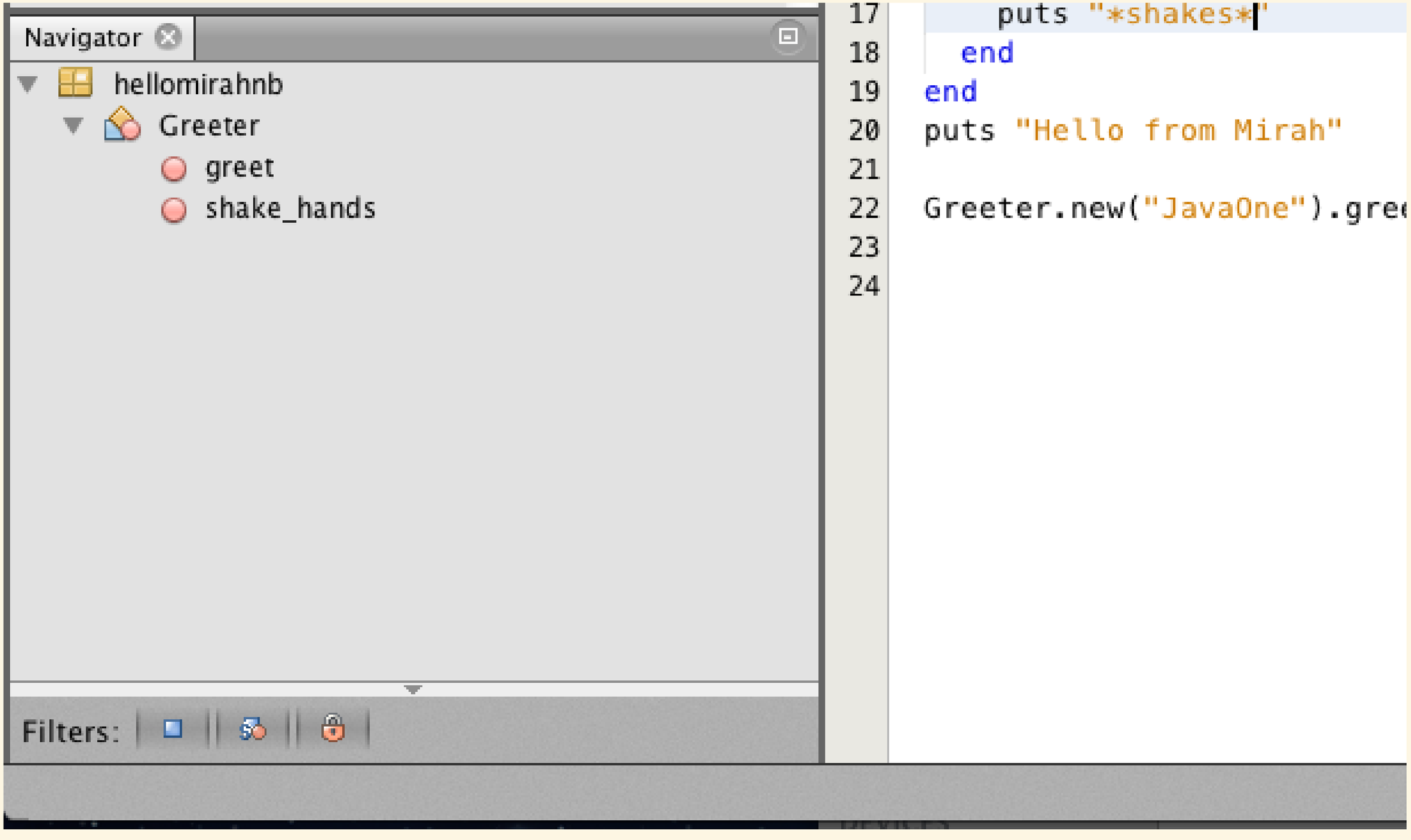
```
toCharArray:char[]
toLowerCase(Locale ):String
toLowerCase:String
toString:String
toUpperCase:String
toUpperCase(Locale ):String
```

Java method completion

```
10    def initialize name: St
11      @name = name
12    end
13    def greet
14      puts "Hello, #{name}"
15    end
16
17    def shout
      puts "Hello, #{name}".to
19    end
20  end
21  puts "Hello from Mirah"
22
23  Greeter.new("JavaOne").gr
24
25
```

```
toCharArray:char[]
toLowerCase(Locale ):String
toLowerCase:String
toString:String
toUpperCase:String
toUpperCase(Locale ):String
```

Also, Navigation!


Navigator
hellomirahnb
Greeter
greet
shake_hands
Filters:
17 puts "*shakes*"
18 end
19 end
20 puts "Hello from Mirah"
21
22 Greeter.new("JavaOne").gree
23
24

Get it

github.com/shannah/mirah-nbm#download

Apache 2.0

Download

ca-weblite-netbeans-mirah.nbm

Installation Instructions

1. Download the .nbm file.
2. In Netbeans, select "Tools" > "Plugins"

2. In NetBeans, select "Tools" > "Plugins"
3. Click on the "Downloaded" tab, and click the "Add Plugins..." button.
4. In the file dialog, select the ca-weblite-netbeans-mirah.nbm module that click "Open".
5. Follow the prompts as it is installed.

Other IDEs

IntelliJ:

github.com/uujava/mirah-idea-plugin

Covered

Getting Mirah

IDE Setup

Code comparison time!

Before Java 8 & lambdas

Java

```
1 Thread t = new Thread(new Runnable() {
2   public void run() {
3     System.out.println("Hello");
4   }
5 });
6 t.start();
7 t.join();
```

Same code, in Mirah:

Mirah

```
1 t = Thread.new { puts "Hello" }
2 t.start
3 t.join
```

Java lambdas!

With Lambdas

```
1 Thread t = new Thread(()-> {
2   System.out.println("Hello");
3 });
4 t.start();
5 t.join();
```

With Lambdas

```
1 String name = "JavaOne";
2 Thread t = new Thread(()-> {
3   System.out.println(
4     "Hello, " + name);
5 });
6 t.start();
7 t.join();
```

In Mirah

```
1 name = "JavaOne"
2 t = Thread.new {
3   puts "Hello, #{name}" }
4 t.start
5 t.join
```

Method definitions

Java: left typed

Mirah: right typed

Java Signature

```
1 public int gcd(int u, int v) {}
```

Mirah Signature

```
1 def gcd(u: int, v: int): int
```

Mirah Signature

```
1 def gcd(u: int, v: int)
```

```
1 interface Maths
2   def gcd(u: int, v: int): int;
3 end
  end
```

Mirah Signature

```
1 def gcd(u, v)
```

Class Definitions

```
1 class CoffeeMug {
2   private double fullness;
3   private double temp;
4   CoffeeMug(double fullness,
5             double temp) {
6     this.fullness = fullness;
7     this.temp = temp;
8   }
9 }
```

```
1 class CoffeeMug
2   def initialize(fullness:
3 double,
4                  temp: double)
5     @fullness = fullness
6     @temp = temp
7   end
  end
```

Getters and Setters

```
1 public double getFullness() {
2   return fullness;
3 }
4
5 public void setFullness(double
6 fullness) {
7   this.fullness = fullness;
8 }
  // ...
```

How about Mirah?

Getters and Setters

```
1 class CoffeeMug
2   attr_accessor fullness:
3 double,
                  temp: double
```

Getters and Setters

```
1 attr_reader fullness: double,
2              temp: double
```

Getters and Setters

```
1 attr_reader fullness: double,
2               temp: double
3 protected attr_writer fullness:
4 double,
                          temp:

  double
```

How?

Macros!?

No runtime

Only at compile time

Macros

Macros

Macros

```
1 attr_reader a: int, b: int
```

```
1 def a: int
2   @a
3 end
4
5 def b: int
6   @b
7 end
```

```
1 @@log.debug_log "just in case
2     #{dump_stats()}"
```

```
1 if @@log.isLevel(Level::DEBUG)
2   @@log.debug_log "..."
3 end
```

```
1 macro def debug(msg)
2   quote do
3     if `@call.target`.isLevel(Level::DEBUG)
4       `@call.target`.debug_log `msg`
5     end
6   end
7 end
```

```
1 macro def debug(msg)
```

```
1 quote do
2   # ...
3 end
```

```
1 `@call.target`.debug_log `msg`
```

```
1 macro def debug(msg)
2   quote do
3     if `@call.target`.isLevel(Level::DEBUG)
4       `@call.target`.debug_log `msg`
5     end
6   end
7 end
8
```

```
1 if @@log.isLevel(Level::DEBUG)
2   @@log.debug_log "just in case
3     #{dump_stats()}"
4 end
```

Macros

Language level compiler plugins

metaprogramming

at compile time

not runtime

Familiar Ruby Syntax

Java Behavior

Familiar Ruby Syntax

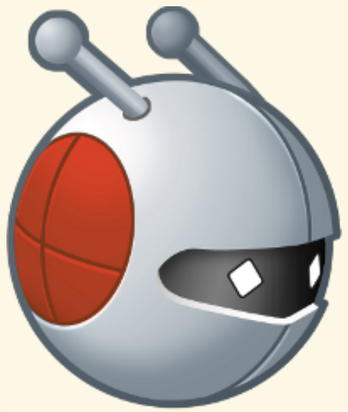
Java Runtime Size

Good for Mobile?

You bet

Android?

Checkout Pindah



Pindah

github.com/mirah/pindah

```
1 package
2 com.baroquebobcat.example
3 import android.app.Activity
4 import android.widget.TextView
5
6 class HelloFromMirah < Activity
7   def onCreate(state)
8     super state
9     tv = TextView.new self
10    tv.setText "Hello,
```

```
11 JavaOne!"
12       setContentView tv
     end
   end
```

Mirah 1.0: The Future

Future Plans and Schemes

1st Class Generics

Current State:

Mirah understands Java Generics

```
1 nums = [1, 2, 3]
2 nums << 7
3 nums.each { |n| puts n + 2 }
```

BUT

No Literals

Adding Java syntax

```
1 class Foot<ToeType>; end
```

Next.

Better Macros

Not easy

```
1  macro def each(block:Block)
2    if block.arguments && block.arguments.required_size() > 0
3      arg = block.arguments.required(0)
4      name = arg.name.identifier
5      type = arg.type if arg.type
6    else
7      name = gensym
8      type = TypeName(nil)
9    end
10   it = gensym
11
12   getter = quote { `it`.next }
13   if type
14     getter = Cast.new(type.position, type, getter)
15   end
16
17   quote do
```

```
18      while `it`.hasNext
19        init {`it` = `@call.target`.iterator}
20        pre {`name` = `getter`}
21        `block.body`
22      end
23    end
24  end
```

Macros: Access to compiler

```
1 macro def foo
2
3 @mirah.typer.change_how_stuff_wo
  end
```

Great for

library writers

compiler hackers

Bad for day to day work

Make it better!

Extension methods

Extension methods

```
1 using org.mine.StringInflections
2
3 count = 5
4 puts "banana".pluralize(count)
```

* Scoped with declarations

```
1 using org.mine.StringInflections
```

* Dispatch via macros

```
1 puts "banana".pluralize(count)
```

Add stack frame at runtime

```
Exception in thread "main"
  at
org.example.StringReflections.pluralize(string_reflections.mirah:1
  at org.example.myapp.App(app.mirah:7)
```

Both Runtime and Compile-time

Define extensions

```
1  extension StringInflections
2    extending String
3      def pluralize count
4        if count == 1
5          "#{count} #{self}"
6        elsif self.endswith "s"
7          "#{count} #{self}es"
8        else
9          "#{count} #{self}s"
10       end
11     end
```

```
12    end
13 end
```

Prior art:

C#'s extension methods

Ruby refinements

Finally, Gradual Typing

JEP 276

Virgo

Mirah's Principles

No Runtime

```
1 $ ls my-app/jars/mirah*.jar
2 ls: my-app/jars/mirah*.jar: No
  such file or directory
```

Prefer Ruby's conventions over Java's

```
1 def first_bageler bags: List
2   bags.each do |b|
3     return b if not b.bagels.all_gone?
4   end
5   nil
6 end
```

Interop well with Java

```
1 JavaClass.new.callMethods
```

```
1 new MirahClass().callMethods()
```

00 macros

Programmer Happiness

Virgo

Get Involved

Apache License 2.0 Grant

committer policy TBD

Code of Conduct