Calling Java

JNIPort for VisualWorks

Joachim Geidel – ESUG Smalltalk Conference 2007

blueCarat Software & Consulting

Agenda

- JNIPort
- Java Native Interface – Invocation Interface
- JNIPort Implementation
  - The Low Level Interface
  - The Twilight Zone
  - Ghost classes
- Tools
- Plans

JNIPort

Use any Java classes with any Java VM in Smalltalk

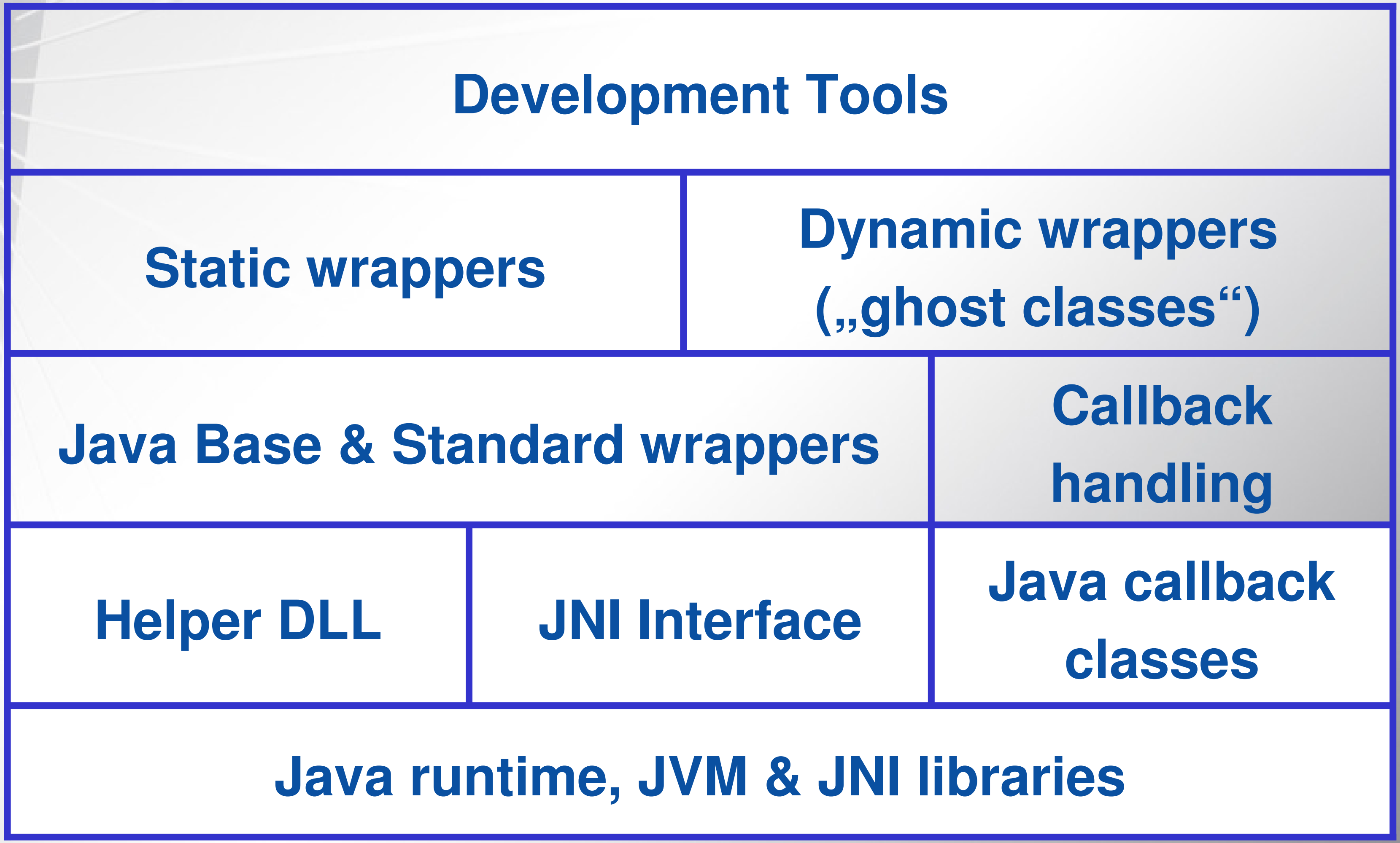
Free open source class library

Developed by Chris Uppal for Dolphin Smalltalk

Ported to VisualWorks in 2006/2007

```
| jvm class |
jvm := JVM current.
class := jvm findClass: #'java.lang.System'.
class currentTimeMillis_null. "--> 1045217556089"
```

JNIPort components



Development Tools

Static wrappers

Dynamic wrappers („ghost classes“)

Java Base & Standard wrappers

Callback handling

Helper DLL

JNI Interface

Java callback classes

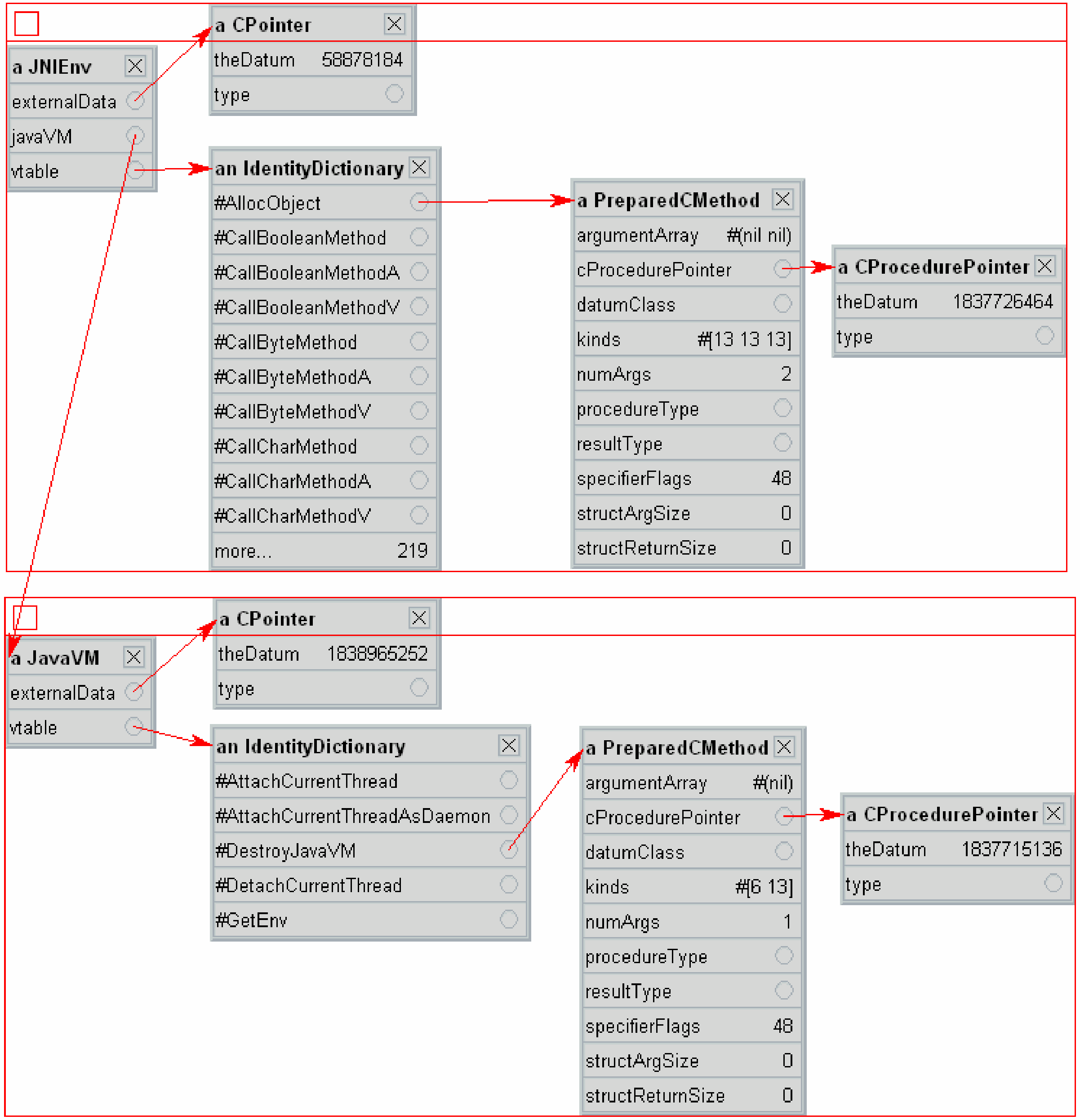
Java runtime, JVM & JNI libraries

JNI – Invocation Interface

- The Java VM is a library, not an executable
- Function tables (vtable): JavaVM, JNIEnv
- JavaVM – Start / stop the Java VM
- JNIEnv – Access classes and instances
  - Look up Java classes
  - Send messages to Java classes and instances
  - Access Java objects using reference objects
    - Local references – valid inside a thread
    - Global references – valid in all threads

Lowest Level API

```
jniEnv :=  JNILibrary new createFirstJNIEnv: JavaVMInitArgs new.

math := jniEnv FindClass_name: 'java/lang/Math'
                    onException: [:error | “error handling“].

absID := jniEnv GetStaticMethodID_class: math
                    name: 'abs' sig: '(D)D'
                    onException: [:error | “error handling“].

arguments := JNIValueArray fromArray: #(-321.2d) types: #(jdouble).

result := jniEnv CallStaticDoubleMethodA_class: math
                    methodID: absID args: arguments
                    onException: [:error | “error handling“].

env javaVM DestroyJavaVM.
```



a JNIEnv ☒
externalData ○
javaVM ○
vtable ○

a CPointer ☒
theDatum 58878184
type ○

an IdentityDictionary ☒
#AllocObject ○
#CallBooleanMethod ○
#CallBooleanMethodA ○
#CallBooleanMethodV ○
#CallByteMethod ○
#CallByteMethodA ○
#CallByteMethodV ○
#CallCharMethod ○
#CallCharMethodA ○
#CallCharMethodV ○
more... 219

a PreparedCMethod ☒
argumentArray #(nil nil)
cProcedurePointer ○
datumClass ○
kinds #[13 13 13]
numArgs 2
procedureType ○
resultType ○
specifierFlags 48
structArgSize 0
structReturnSize 0

a CProcedurePointer ☒
theDatum 1837726464
type ○

a JavaVM ☒
externalData ○
vtable ○

a CPointer ☒
theDatum 1838965252
type ○

an IdentityDictionary ☒
#AttachCurrentThread ○
#AttachCurrentThreadAsDaemon ○
#DestroyJavaVM ○
#DetachCurrentThread ○
#GetEnv ○

a PreparedCMethod ☒
argumentArray #(nil)
cProcedurePointer ○
datumClass ○
kinds #[6 13]
numArgs 1
procedureType ○
resultType ○
specifierFlags 48
structArgSize 0
structReturnSize 0

a CProcedurePointer ☒
theDatum 1837715136
type ○

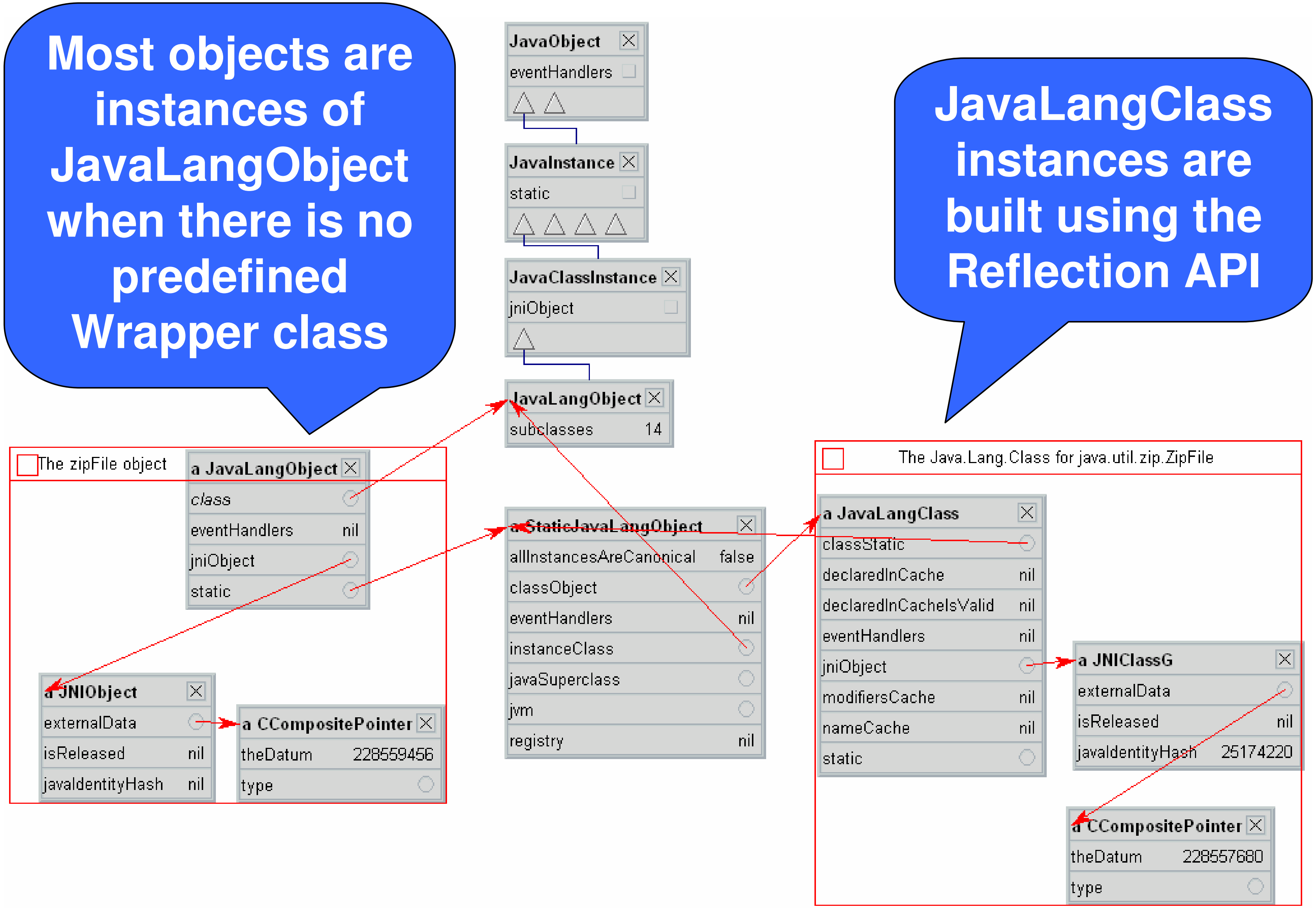
The Twilight Zone

- References are automatically encapsulated by Smalltalk objects
- Statics, Java Instances
- Messages are sent to Statics and Instances
  - No need to talk to the JNIEnv
  - But still low level
- Automatic Life Cycle Management (GC)
- Access to Java Reflection

```
jvm := JVM current.
zipfileClass := jvm findClass: #'java.util.zip.ZipFile'.
args := (JNIValueArray new: 1).
args objectAt: 1 put: (‘file.zip‘ asJavaString: jvm).
zipfile := zipfileClass
   callConstructorSignature: '(Ljava/lang/String;)V'
   withArguments: args.
zipfile callIntMethod: 'size'.           "--> 6"
entries := zipfile
   callObjectMethod: 'entries'
   signature: '()Ljava/util/Enumeration;'.
```

Most objects are instances of JavaLangObject when there is no predefined Wrapper class

JavaLangClass instances are built using the Reflection API



JavaObject ☒
eventHandlers □

JavaInstance ☒
static □

JavaClassInstance ☒
jniObject □

JavaLangObject ☒
subclasses 14

□ The zipFile object

a JavaLangObject ☒
class ○
eventHandlers nil
jniObject ○
static ○

a JNIObject ☒
externalData ○
isReleased nil
javaIdentityHash nil

a CCompositePointer ☒
theDatum 228559456
type ○

a StaticJavaLangObject ☒
allInstancesAreCanonical false
classObject ○
eventHandlers nil
instanceClass ○
javaSuperclass ○
jvm ○
registry nil

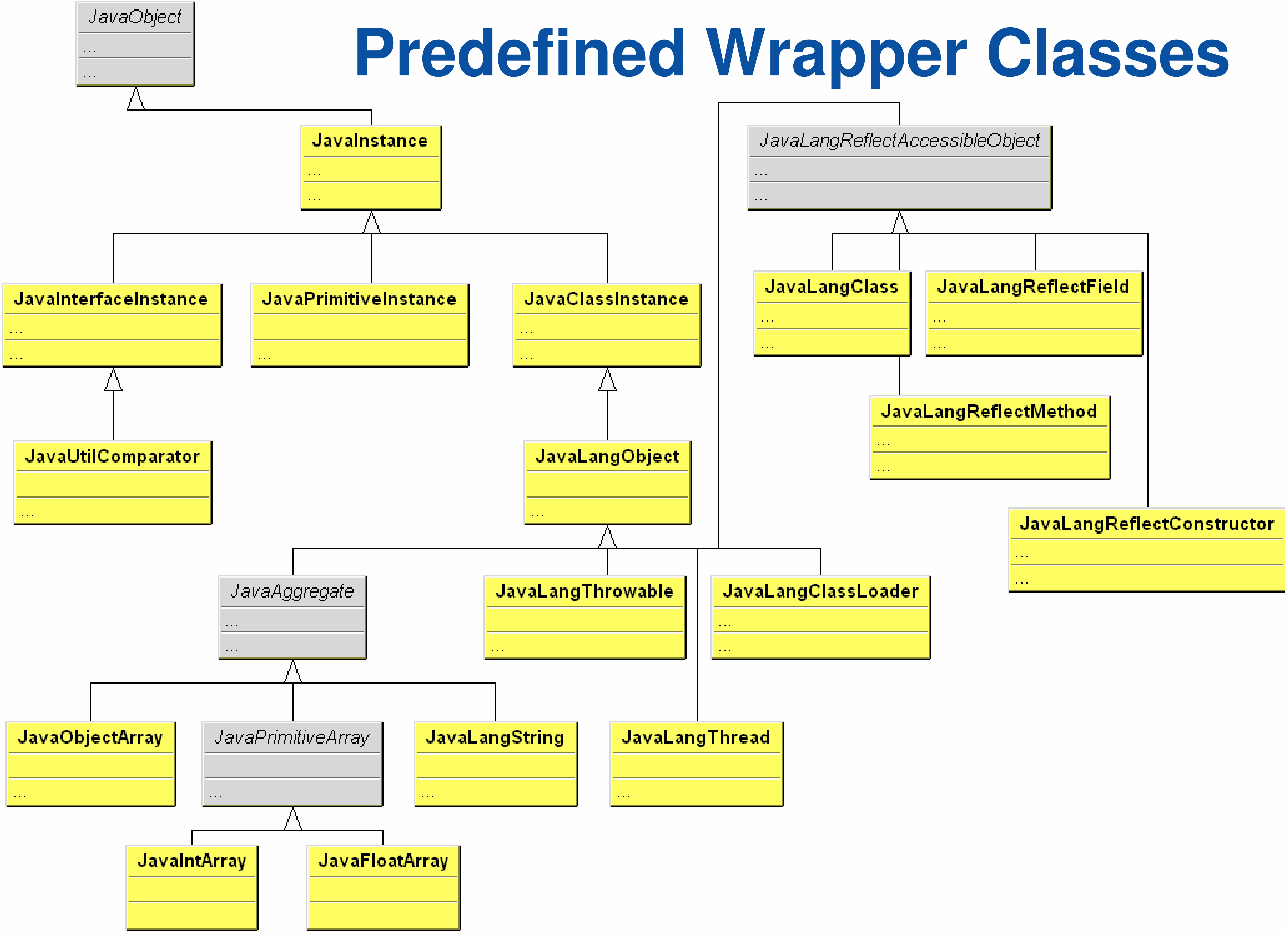
□ The Java.Lang.Class for java.util.zip.ZipFile

a JavaLangClass ☒
classStatic ○
declaredInCache nil
declaredInCacheIsValid nil
eventHandlers nil
jniObject ○
modifiersCache nil
nameCache nil
static ○

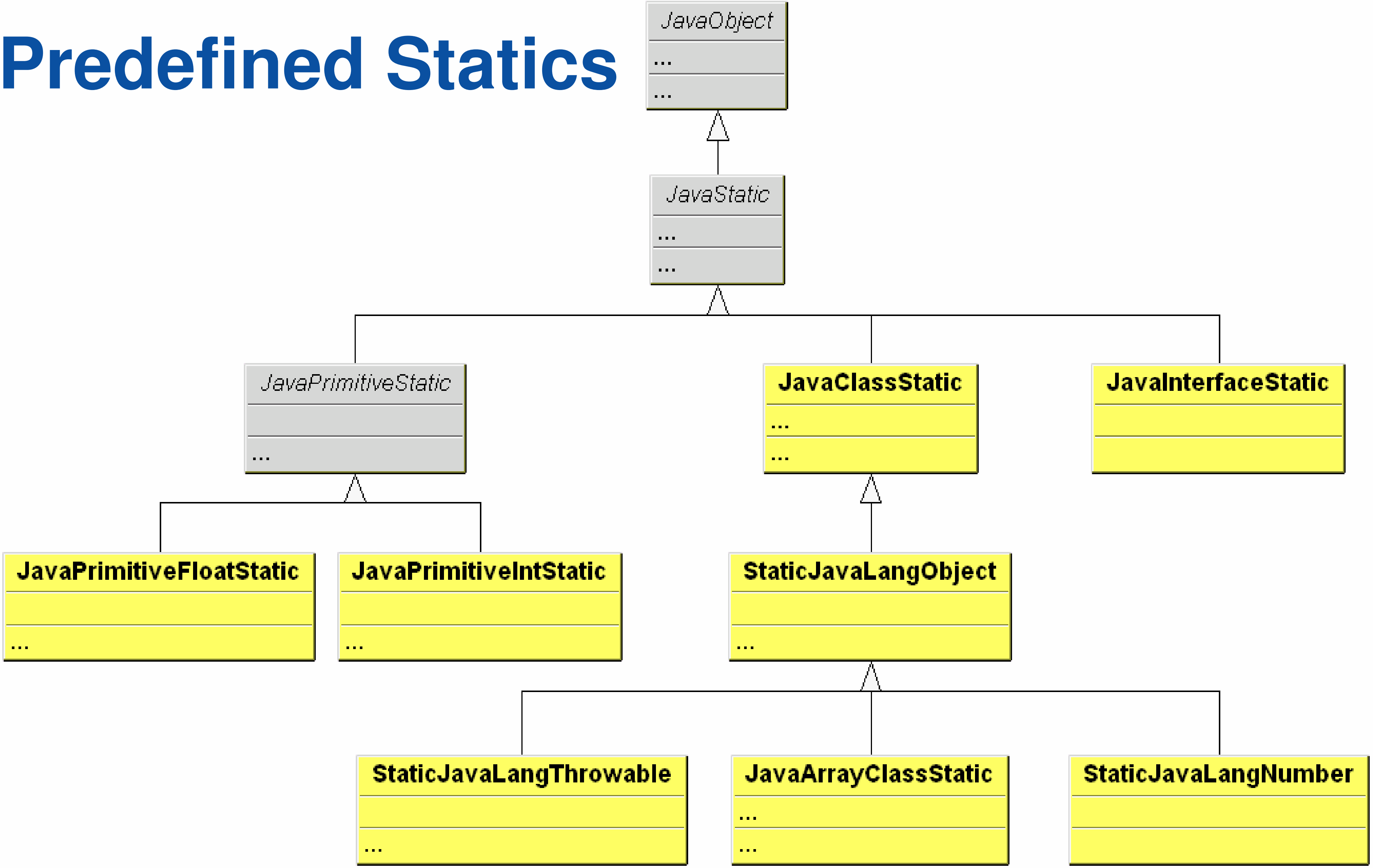
a JNIClassG ☒
externalData ○
isReleased nil
javaIdentityHash 25174220

a CCompositePointer ☒
theDatum 228557680
type ○

Predefined Wrapper Classes



Predefined Statics



Ghost classes

Use Java objects just like Smalltalk objects

Dynamically generated…

- …wrapper classes
- …wrapper methods

Code generation triggered by creating the first reference to an instance of a Java class

~~Ghost classes disappear when they have no instances~~

This is not true!

Ghost methods

Source code generated using information from Reflection API, querying a Java class' protocol

Context specific information embedded into CompiledCode as „literals“

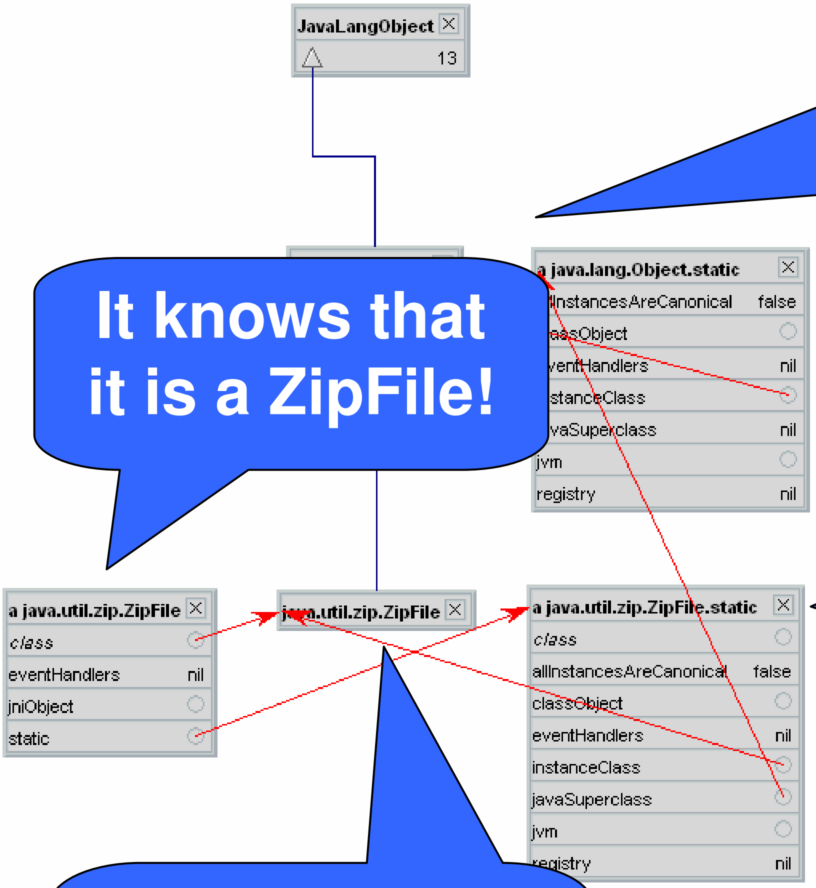
Source code is discarded

- Can be kept in image (JNIPort configuration option)

Augmented tools to handle ghost methods

```
jvm := JVM current.

zipfileClass := jvm findClass: #'java.util.zip.ZipFile'.

zipfile := zipfileClass new_String: 'MyZipFile.zip'.

zipfile size_null.

entries := zipfile entries_null.

entries asAnEnumeration
   do: [:each | Transcript cr; print: each].
```

No need to implement these methods – they are generated on the fly.



JavaLangObject
13

a java.lang.Object.static

llInstancesAreCanonical	false
assObject	○
ventHandlers	nil
stanceClass	○
vaSuperclass	nil
jvm	○
registry	nil

a java.util.zip.ZipFile

class	○
eventHandlers	nil
jniObject	○
static	○

java.util.zip.ZipFile

a java.util.zip.ZipFile.static

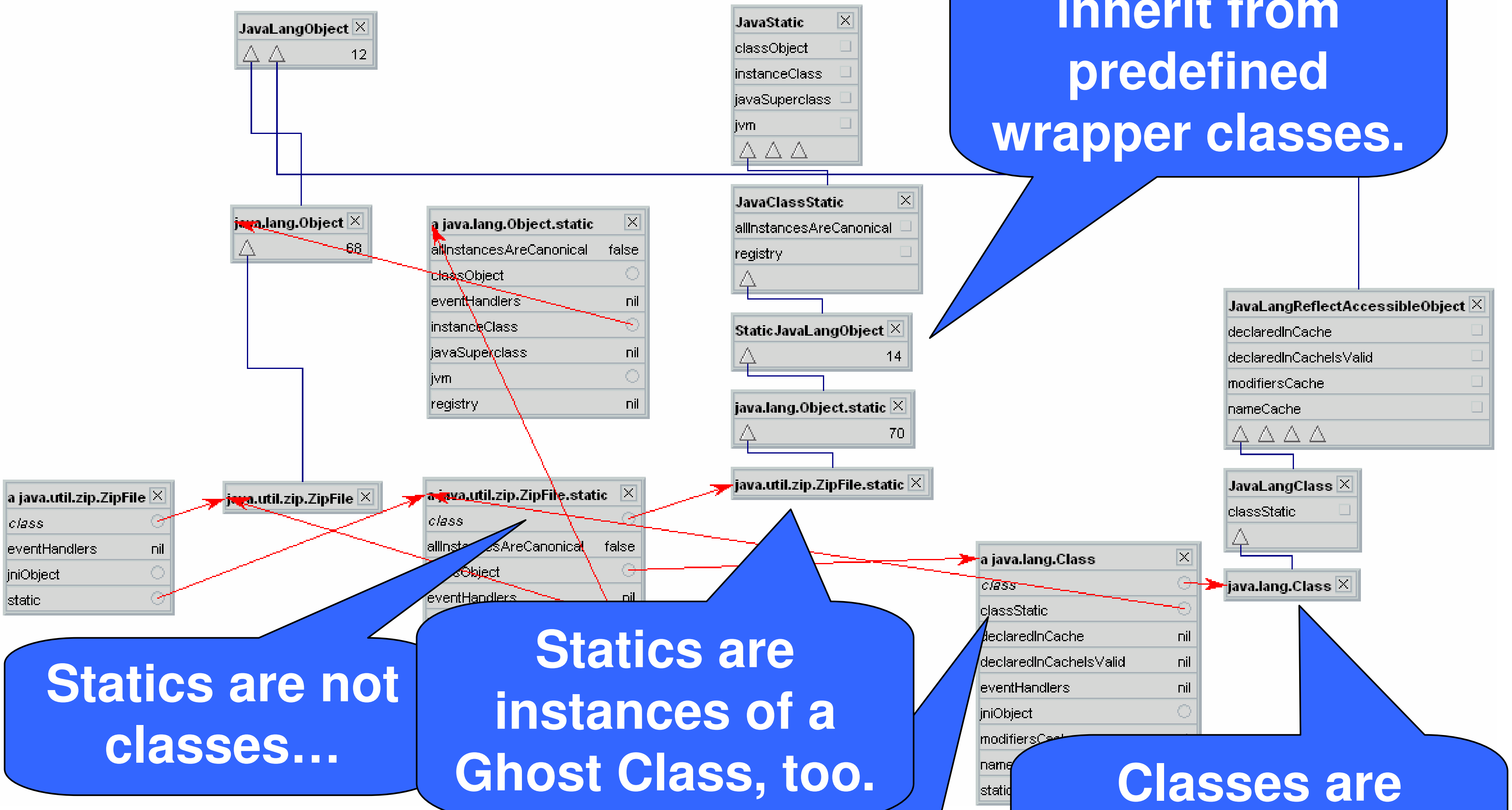
class	○
allInstancesAreCanonical	false
classObject	○
eventHandlers	nil
instanceClass	○
javaSuperclass	○
jvm	○
registry	nil

Parallel hierarchies – similar to Class / Metaclass hierarchy in Smalltalk

It knows that it is a ZipFile!

Static – wraps the static part of the Java class

Ghost Class – a Smalltalk class



Ghost Classes inherit from predefined wrapper classes.

Statics are not classes…

Statics are instances of a Ghost Class, too.

…but Classes are.

Classes are instances of a Ghost Class, too.

Don‘t worry – you don‘t have to know that.

Just write your code.

JNIPort does the rest.

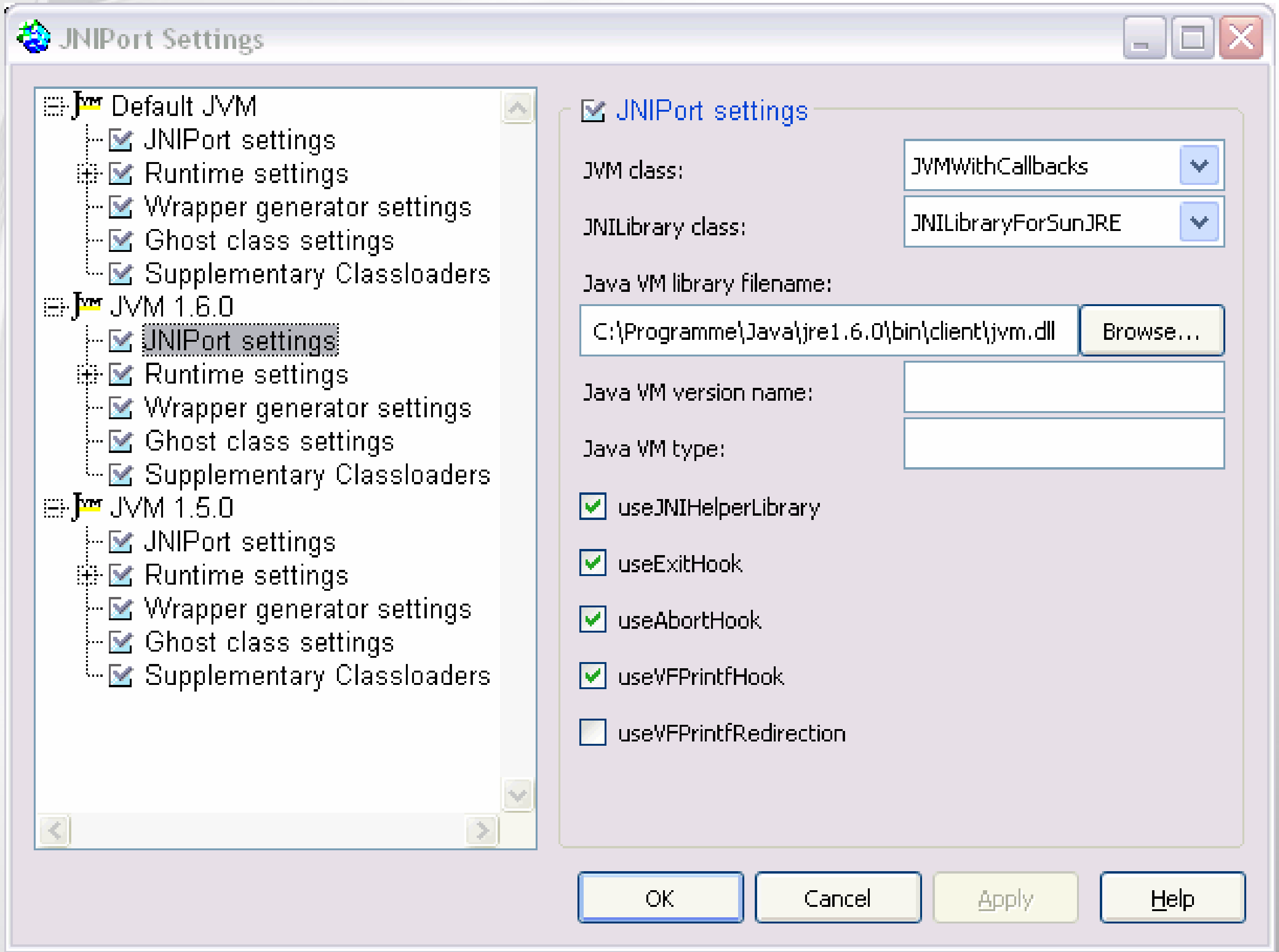
Tools

Settings Tool

Class Wrapper Browser

Inspector

Decompiler



☑ JNIPort settings

JVM class: JVMWithCallbacks

JNILibrary class: JNILibraryForSunJRE

Java VM library filename: C:\Programme\Java\jre1.6.0\bin\client\jvm.dll

Java VM version name:

Java VM type:

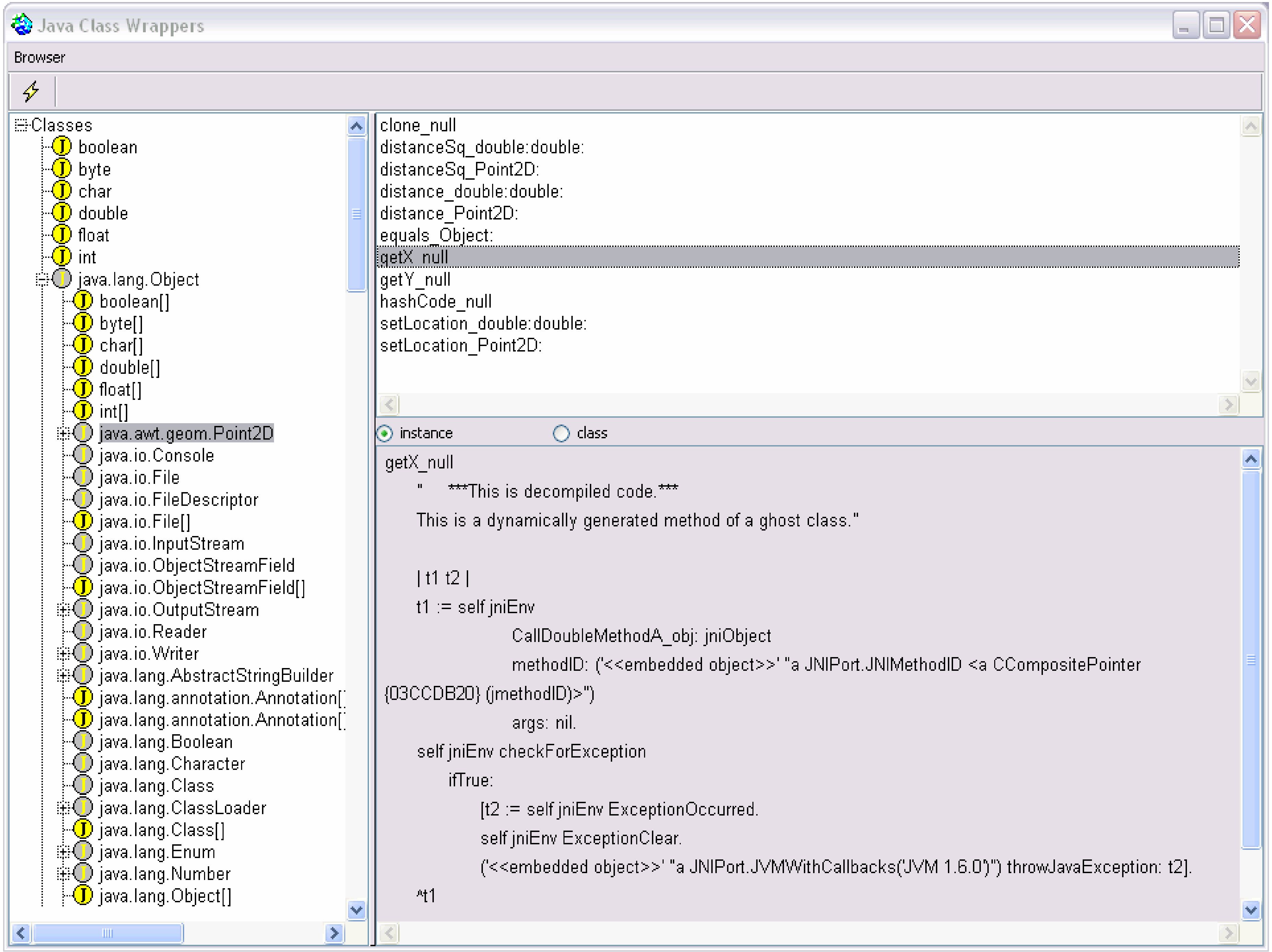
☑ useJNIHelperLibrary

☑ useExitHook

☑ useAbortHook

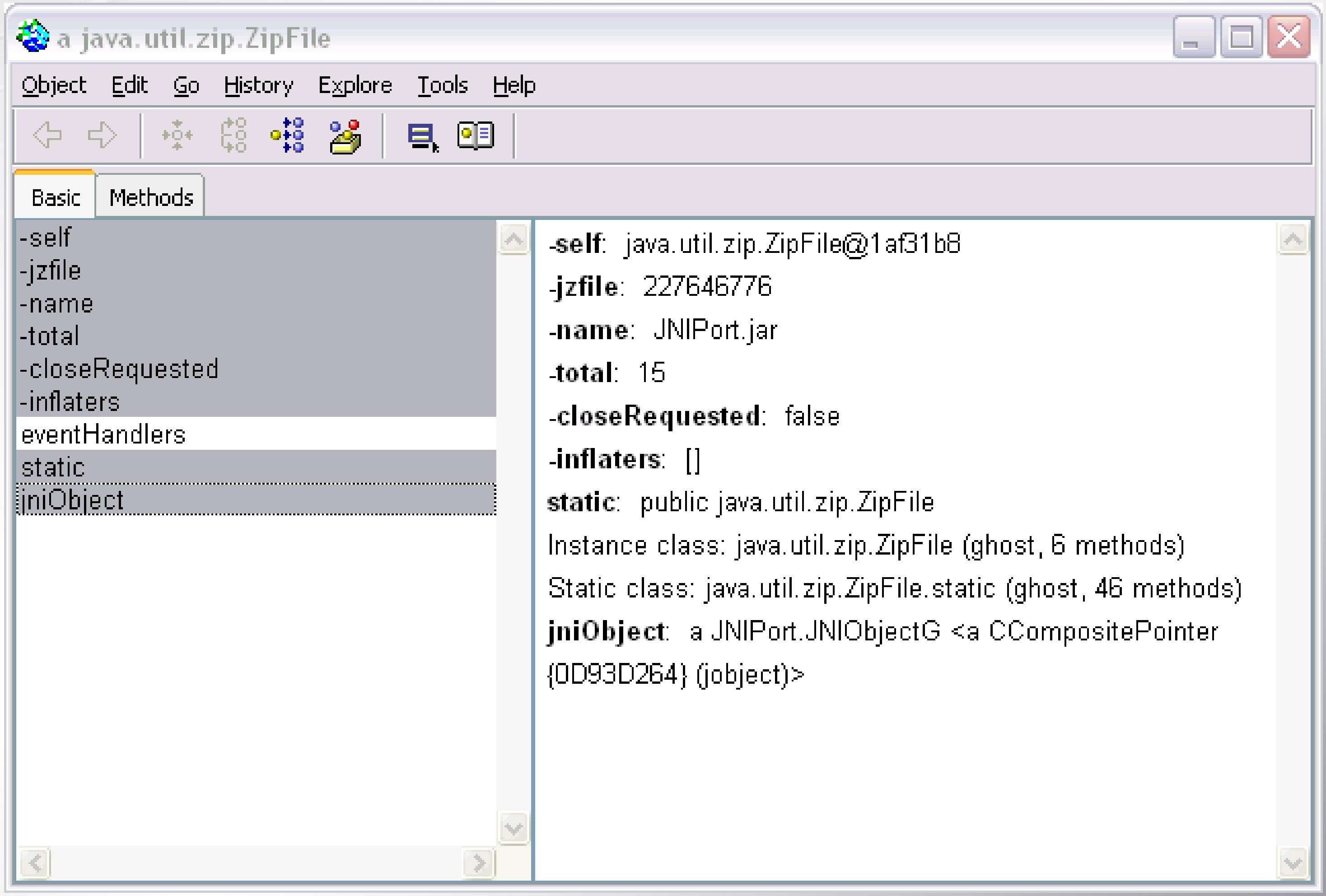
☑ useVFPrintfHook

☐ useVFPrintfRedirection



```
getX_null
    "    ***This is decompiled code.***
    This is a dynamically generated method of a ghost class."

    | t1 t2 |
    t1 := self jniEnv
                CallDoubleMethodA_obj: jniObject
                methodID: ('<<embedded object>>' "a JNIPort.JNIMethodID <a CCompositePointer
{03CCDB20} (jmethodID)>")
                args: nil.
    self jniEnv checkForException
        ifTrue:
            [t2 := self jniEnv ExceptionOccurred.
            self jniEnv ExceptionClear.
            ('<<embedded object>>' "a JNIPort.JVMWithCallbacks('JVM 1.6.0')") throwJavaException: t2].
    ^t1
```

Inspector



-self: java.util.zip.ZipFile@1af31b8

-jzfile: 227646776

-name: JNIPort.jar

-total: 15

-closeRequested: false

-inflaters: []

static: public java.util.zip.ZipFile

Instance class: java.util.zip.ZipFile (ghost, 6 methods)

Static class: java.util.zip.ZipFile.static (ghost, 46 methods)

jniObject: a JNIPort.JNIObjectG <a CCompositePointer {0D93D264} (jobject)>

Plans

Tools for generating static wrapper classes

Linux, MacOS X

Java packages as Smalltalk Namespaces:

```
java.lang.System currentTimeMillis_null.
```

instead of

```
class := JVM current findClass:
  #'java.lang.System'.
class currentTimeMillis_null.
```

Resources

- Cincom Public Repository
  - Registry (version 16 or later)
  - FastCMethodPointers (version 1.1 or later)
  - Weaklings (version 12 or later)
  - JNIPort Prerequisites
  - JNIPort
  - JNIPort Tools
  - JNIPort Tests
- Chris Uppal‘s web site: http://www.metagnostic.org
  - Documentation: http://www.metagnostic.org/DolphinSmalltalk/JNIPort.html
  - Extras directory in http://www.metagnostic.org/DolphinSmalltalk/JNIPort-Complete.zip