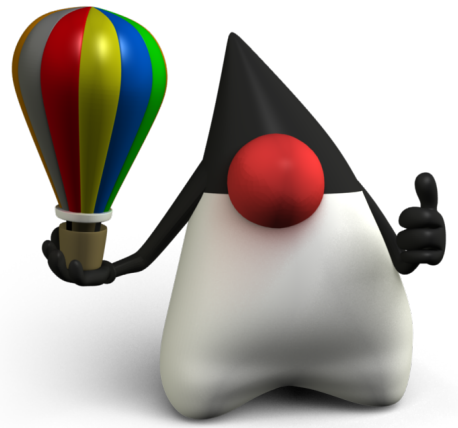
Smalltalk
redline

Smalltalk on the JVM

By James Ladd
object@redline.st
http://redline.st
@redline_st

- ESUG 2011



Our Promise:

Our promise is to provide you with high quality Smalltalk to enjoy every day. If you are not 100% satisfied, let us know and we will happily try to improve.

NUTRITION INFORMATION:

Servings per package: Endless
Serving size: Minimal

	% Daily Intake* Per Serving
Smalltalk	100%
Java Byte Code	100%
- Java integration	100%
Fat, total	0%

Note: All values are considered averages unless otherwise indicated and relate to uncooked product.

* Smalltalk is naturally a source of productivity.

Ingredients:

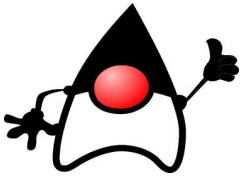
Smalltalk, for the Java Virtual Machine.

Storage:

Download and keep on disk. Do not delete, if downloaded try within 24 hours.

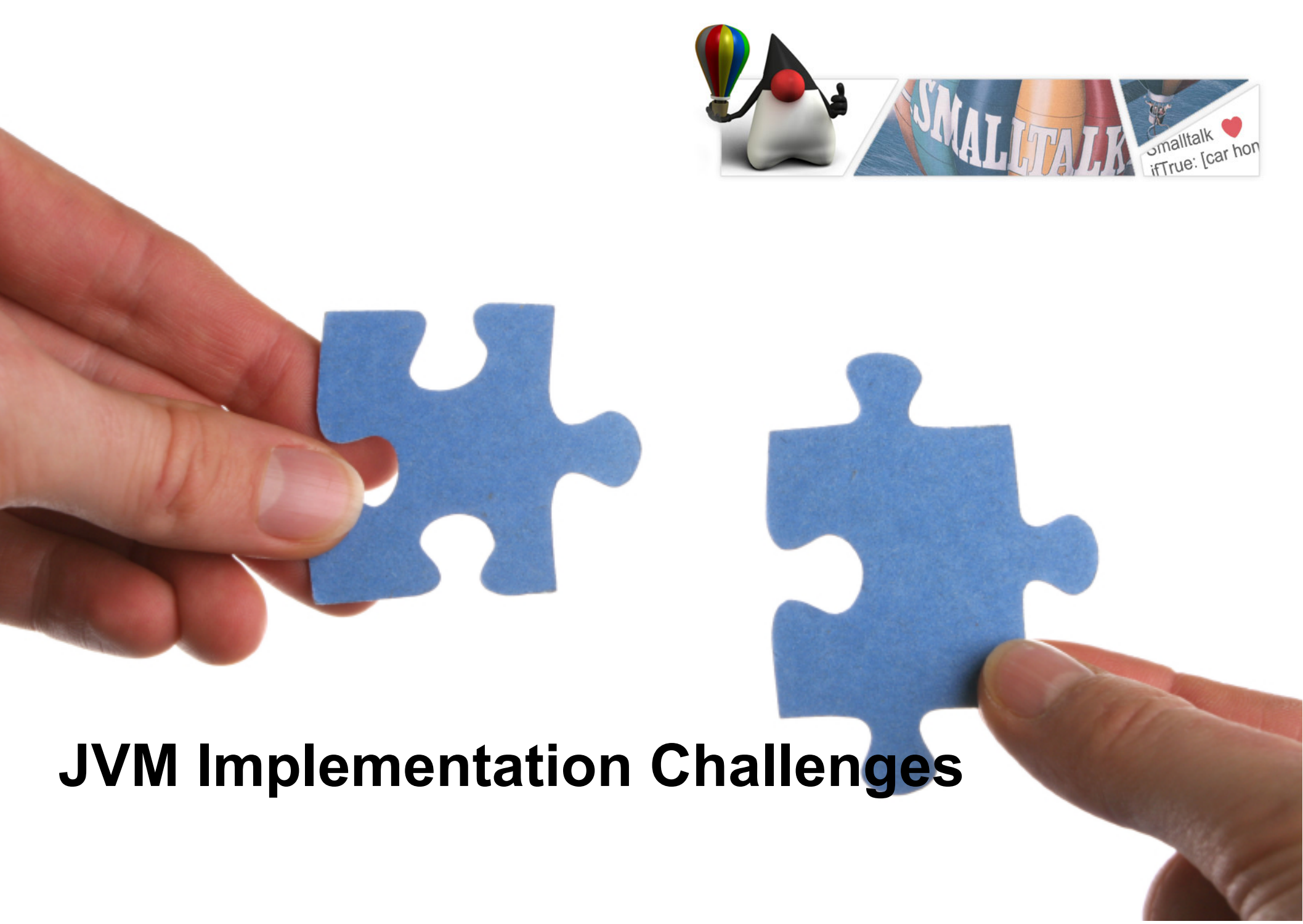
Information:

Product of Australia.
Redline Smalltalk is open source and free.



Please Share

BEST BEFORE TOMORROW



JVM Implementation Challenges

JVM Implementation Challenges:

JVM Unit of execution is a class

Sequence of bytes in class format

Class is not an Object
(more like data structure)

JVM Implementation Challenges:

JVM Unit of execution is a class

Load Class

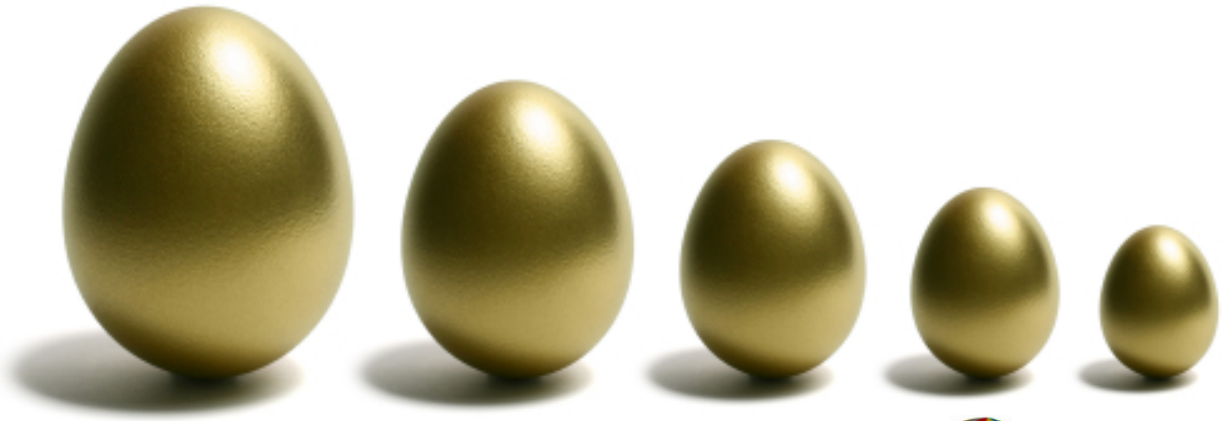
newInstance()

<init> called

JVM Implementation Challenges:

JVM loads classes with a Class Loader

loadClass("java.lang.String")

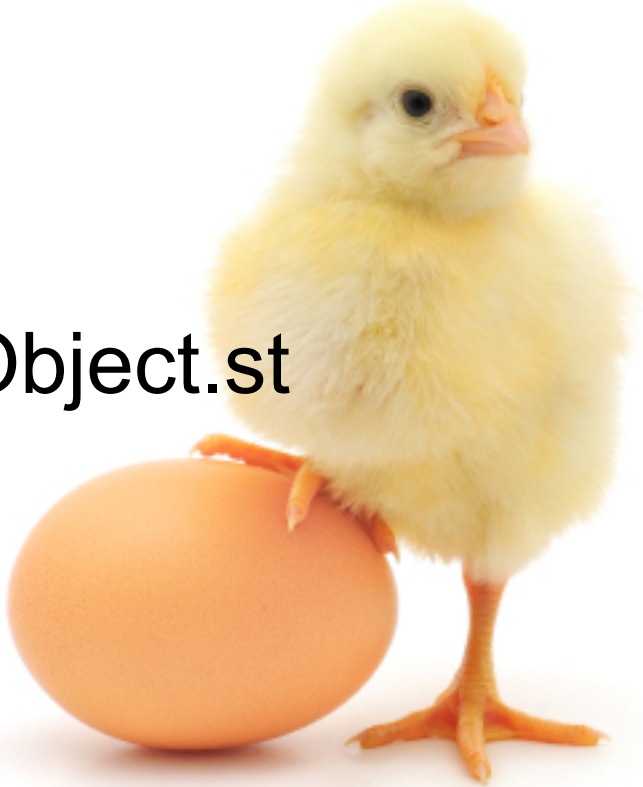


JVM Implementation Challenges:

JVM loads classes with a Class Loader

SmalltalkClassLoader

loadClass('Object') = Object.st



JVM Implementation Challenges:

JVM classes are namespaced

java.lang, st.redline

Package

Must specify fully qualified name

JVM Implementation Challenges:

Calling methods is cumbersome

Very static

Must specify exact type

JVM Implementation Challenges:

Calling methods is cumbersome

St/redline/ProtoObject

(Ljava/lang/Object;)Ljava/lang/StringBuilder;

JVM Implementation Challenges:

Calling methods is cumbersome

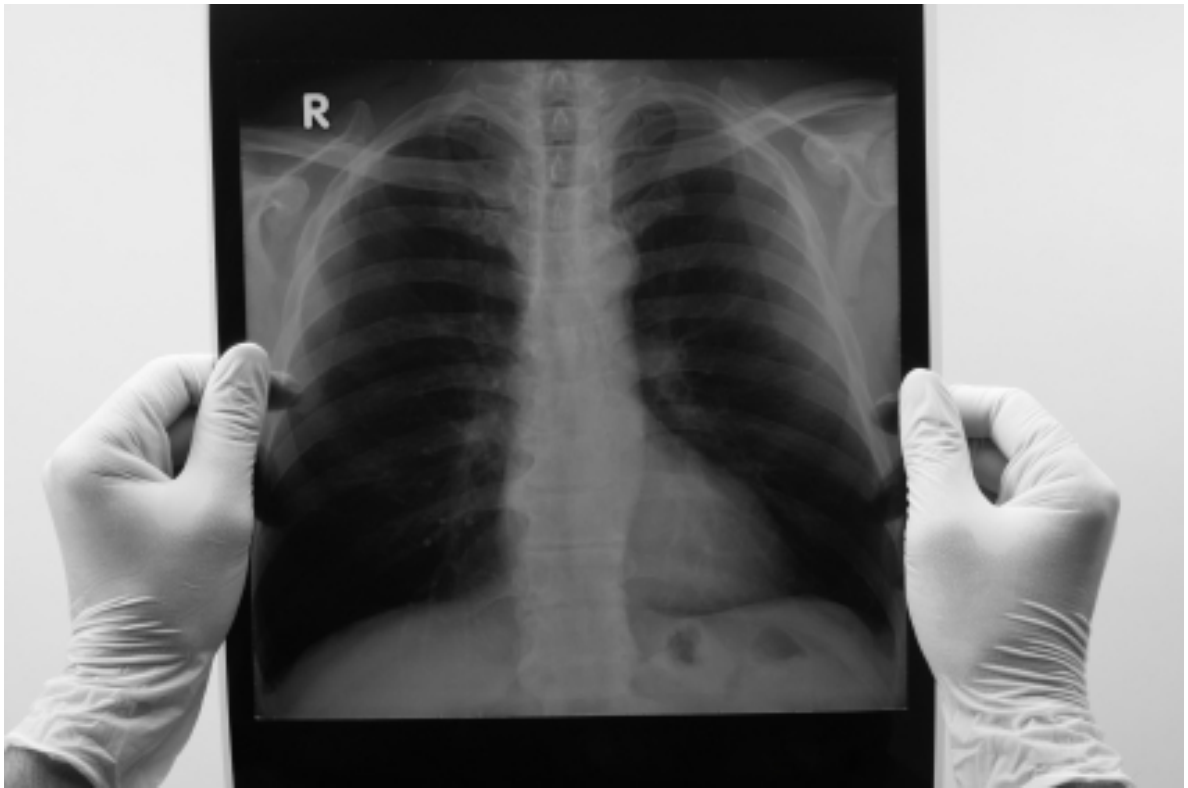
Luckily everything in Smalltalk is an Object (ProtoObject)

JVM Implementation Challenges:

Methods are part of a class

Change class to change method

Redline Smalltalk Internals



Redline Smalltalk Internals

Uses ANTLR

Smalltalk.g

PreProc.g

Redline Smalltalk Internals

Users ANTLR

Jim Idle wrote pre-processor

Will eventually make grammar skinnable

Redline Smalltalk Internals

Uses ObjectWeb ASM

Bytecode class writing library

Redline Smalltalk Internals

Base of hierarchy is ProtoObject

Java object that implements primitives

Each primitive is a static Java method

Redline Smalltalk Internals

Base of hierarchy is ProtoObject

Smalltalk Objects built using message sends to ProtoObject

Compiler's job is to generate message sends

Redline Smalltalk Internals

Two other Java Objects

ProtoMethod

ProtoBlock

Redline Smalltalk Internals

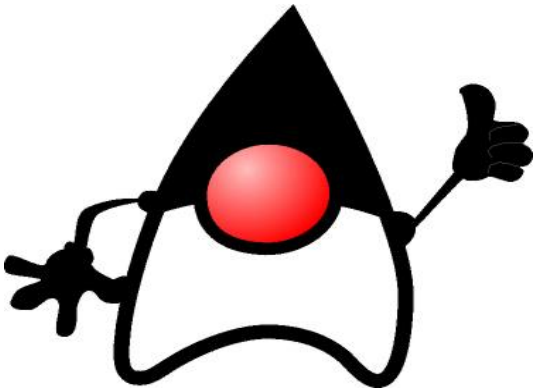
Redline Java Class Loader

SmalltalkClassLoader

Searches for source (.st file)

Redline Smalltalk Internals

Redline Java Class Loader



Can use Smalltalk classes from Java

Redline Smalltalk Internals

Namespace support

Modelled on Java Packages

File path is package

Redline Smalltalk Internals

Namespace support

st/redline/Object = st/redline package

Each class has own 'set' of imports

Redline Smalltalk Internals

SmalltalkClassLoader

Partitions applications within single JVM

Redline Execution

Redline Execution

What happens when st.redline.Example is Executed?

Redline Execution: executing st.redline.Example

Invoke class from command line

./stic st.redline.Example

Redline Execution: executing st.redline.Example

Stic

Creates instance of SmalltalkClassLoader

Sets as context ClassLoader

Redline Execution: executing st.redline.Example

Stic

Asks SmalltalkClassLoader to bootstrap

Asks ProtoObject to resolve
'st.redline.Example'

Redline Execution: executing st.redline.Example

ProtoObject loads class

Class.forName(“st.redline.Example”);

Redline Execution: executing st.redline.Example

SmalltalkClassLoader

Checks cache – returns object if present

Searches source path for 'Example.st'
ie: src/main/smalltalk/st/redline/Example.st

Redline Execution: executing st.redline.Example

SmalltalkClassLoader

Invokes compiler on source file

Resulting class loaded into JVM

Instance created class.newInstance()

Redline Execution: executing st.redline.Example

Method newInstance()

Runs Class <init> method

<init> method contains message sends from Source

Redline Execution: executing st.redline.Example

Compiler

Creates Java class to contain logic

Class is subclass of ProtoObject

Package is based on file path

Redline Execution: executing st.redline.Example

Compiler

Creates <init> method, which is executed when instance created.

Logic in 'st.redline.Example' is encoded as message sends: ProtoObject.primitiveSend(...)

Redline Execution: executing st.redline.Example

Compiler – Smalltalk Methods

Methods are encoded as a message send to compile the method source.

Because class doesn't exist yet.

Redline Execution: executing st.redline.Example

Method Compilation

Methods are 1st class objects

Subclass of ProtoMethod

Added to receivers method dictionary

Redline Execution: executing st.redline.Example

Method Compilation

All Methods Objects have only 1 method

applyTo(...)

Contains logic embodied in Smalltalk method

Redline Execution: executing st.redline.Example

Block Compilation

Blocks are 1st class objects

Subclass of ProtoBlock

Redline Execution: executing st.redline.Example

Block Compilation

Blocks instance created when used

Have a Java applyTo(...) method

^ semantics handled correctly

Redline Demo

Questions?

Please visit:

http://redline.st