JDK 7 — *and beyond*

Mark Reinhold
Principal Engineer
Sun Microsystems

http://blogs.sun.com/mr
http://openjdk.java.net

2009/10/16 — JVM Language Summit 2009

Modularization

Multiple languages

Java language evolution

Modularization

~~Multiple languages~~

Java language evolution

http://flickr.com/photos/chefranden/908539119

```
// Strings in switch

String s = ...;
switch (s) {
    case "quux":
        processQuux(s);
        // fall through
    case "foo":
    case "bar":
        processFooOrBar(s);
        break;
    case "baz":
        processBaz(s);
        // fall through
    default:
        processDefault(s);
        break;
}
```

```
// Automatic resource management

void copy(File src, File dst) throws IOException {
    try (InputStream in = new FileInputStream(src);
         OutputStream out = new FileOutputStream(dst))
    {
        byte[] buf = new byte[8192];
        int n;
        while ((n = in.read(buf)) >= 0)
            out.write(buf, 0, n);
    }
}
```

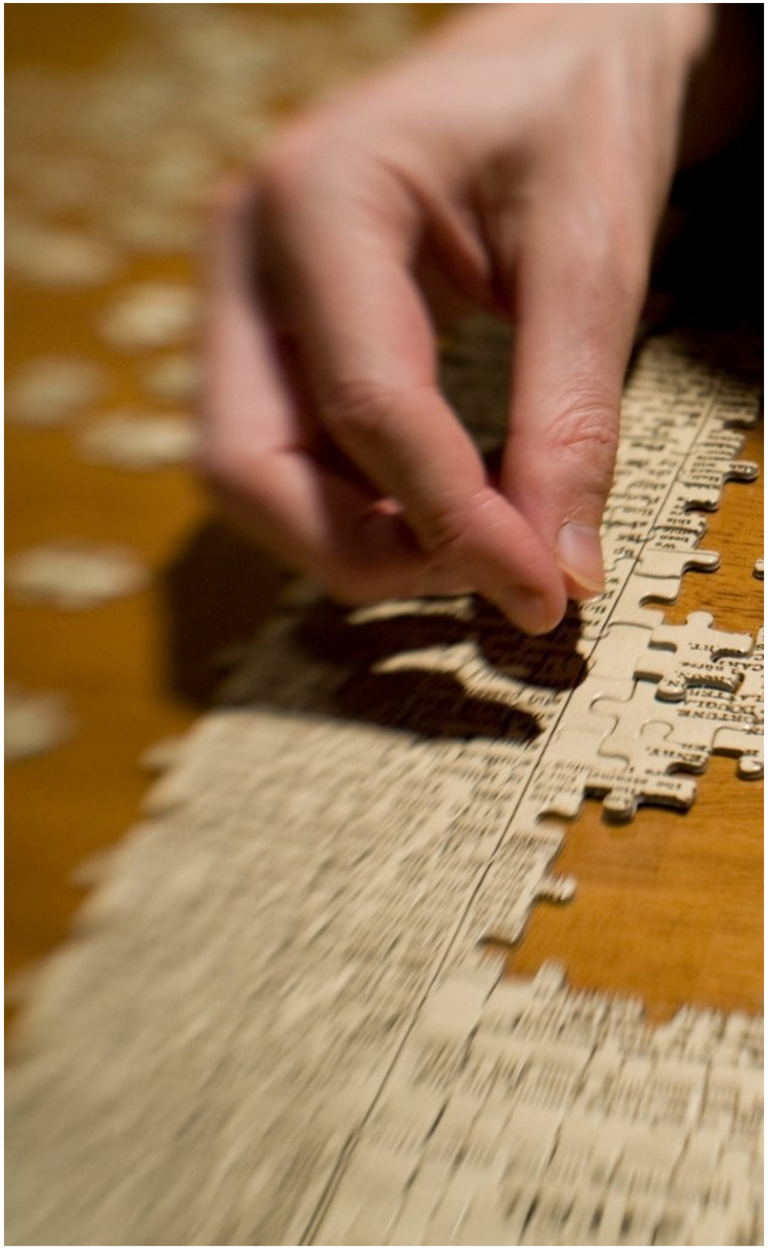
```
// Improved type inference ("diamond")

Map<String,Map<Integer,List<String>>> map
    = new HashMap<>();
```

```
// Improved integer literals

long i = 0b10100010000101100001011100100110000101101100;

long j = 418_530_815_043;

assert i == j;
```

```
// Collection literals

List<Integer> piDigits = [ 3, 1, 4, 1, 5, 9, 2, 6, 5 ];

Set<Integer> primes = { 2, 7, 31, 127, 8191, 131071 };

Map<Integer,String> platonicSolids
    = { 4  : "tetrahedron",
        6  : "cube",
        8  : "octahedron",
        12 : "dodecahedron" };
```

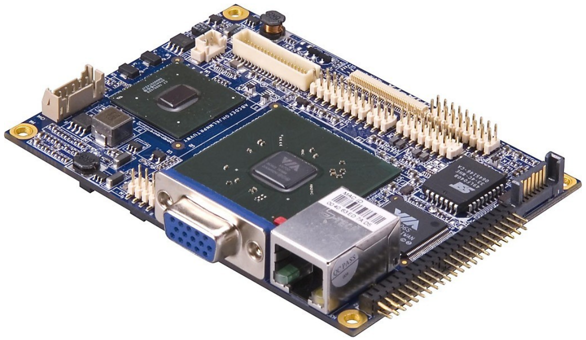
```
// Collection literals

List<Integer> piDigits = [ 3, 1, 4, 1, 5, 9, 2, 6, 5 ];

Set<Integer> primes = { 2, 7, 31, 127, 8191, 131071 };

Map<Integer,String> platonicSolids
    = { 4  : "tetrahedron",
        6  : "cube",
        8  : "octahedron",
        12 : "dodecahedron" };


// Collection accessors

assert piDigits[3] == 1;

assert platonicSolids[6].equals("cube");

platonicSolids[20] = "icosahedron";
```

```
// Language support for JSR 292 (InvokeDynamic)


// Exotic identifiers

static int #"op~"(int x) { return ~x; }
int x = #"op~"(0);


// InvokeDynamic instructions

InvokeDynamic<int>.#"exotic!"(3);


// Method-handle invocation

MethodHandle mh = ...;
int x = mh.<int>invoke(3);
boolean x = mh.<boolean>invoke("foo");
```



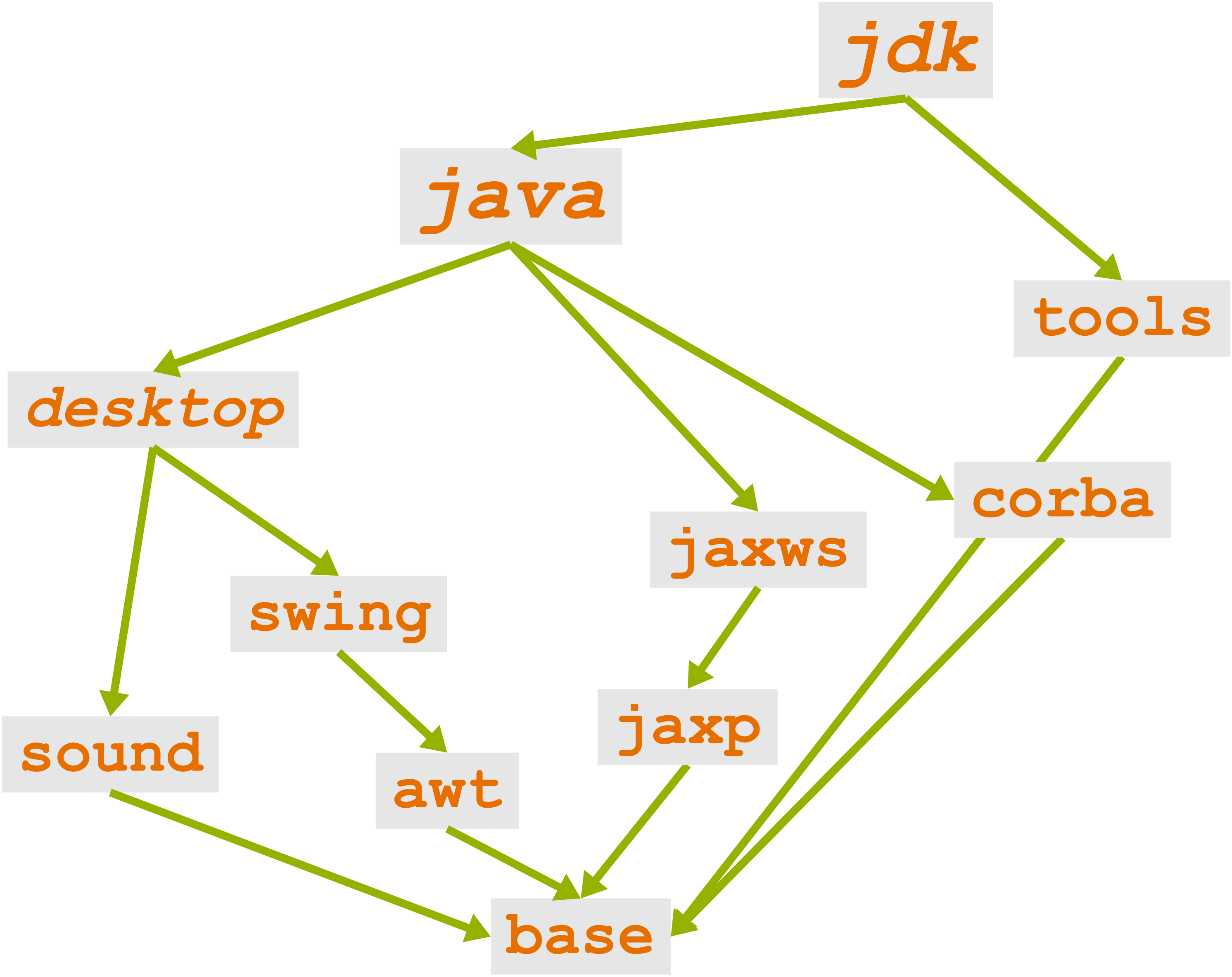
http://flickr.com/photos/lizadaly/2944362379

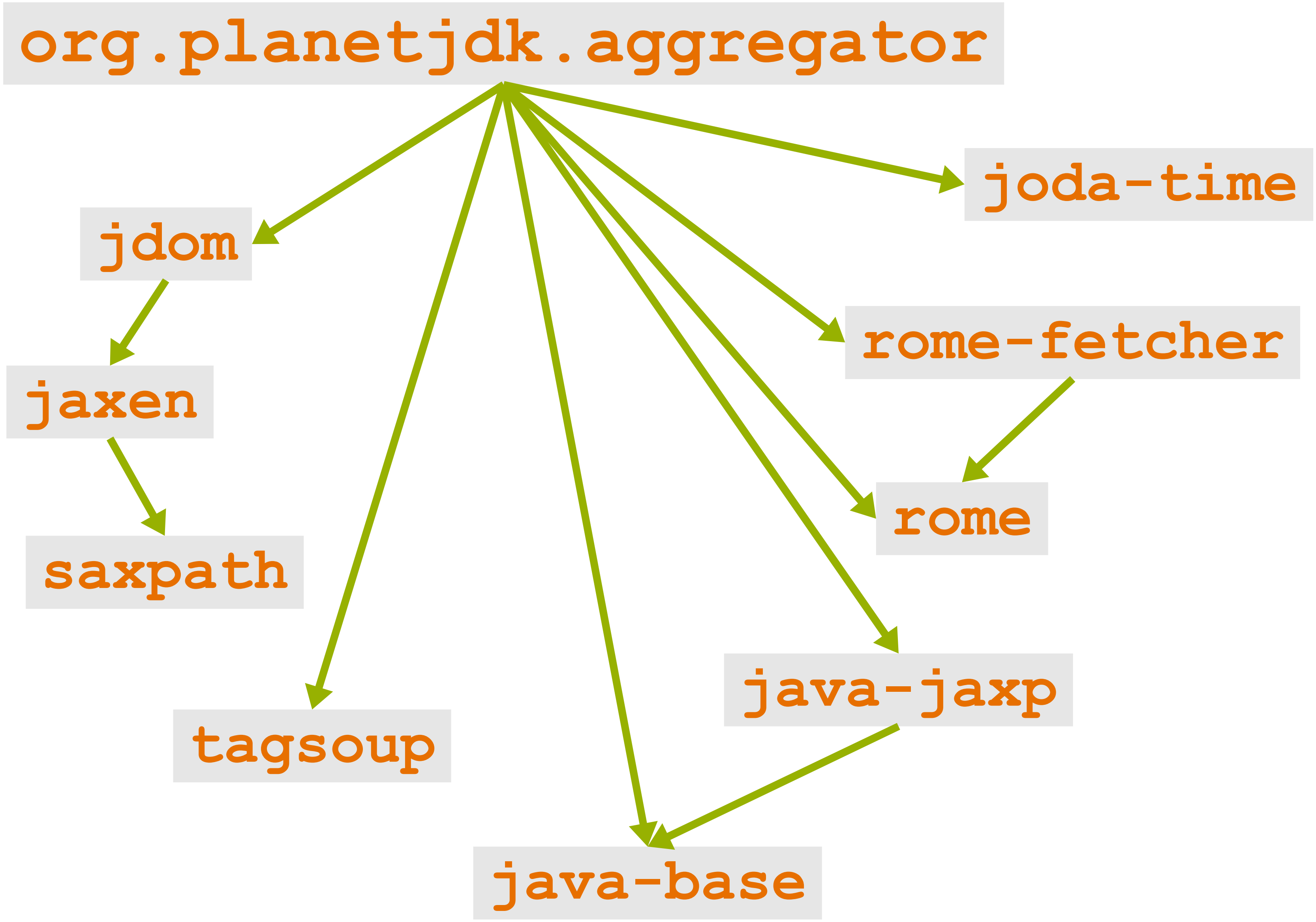
13MB



http://flickr.com/photos/viagallery/2290654438

http://flickr.com/photos/thatguyfromcchs08/2300190277


jdk
java
tools
desktop
corba
jaxws
swing
jaxp
sound
awt
base


org.planetjdk.aggregator
joda-time
jdom
rome-fetcher
jaxen
rome
saxpath
java-jaxp
tagsoup
java-base

Why should I care?

```
$ wget -q http://www.scala-lang.org/downloads/\
scala-2.7.6.final.tgz
$ tar xfz scala-2.7.6.final.tgz
$ export PATH=~/scala-2.7.6.final/bin:$PATH
$ which scala
~/scala-2.7.6.final/bin/scala
$ scala -version
Scala version 2.7.6.final -- Copyright 2002-2009, LAMP/EPFL
$
```

```
$ jmod install http://www.scala-lang.org/downloads/\
scala-2.7.6.final.jmod
$ which scala
~/bin/scala
$ scala -version
Scala version 2.7.6.final -- Copyright 2002-2009, LAMP/EPFL
$
```

```
$ sudo jmod install http://www.scala-lang.org/downloads/\
scala-2.7.6.final.jmod
$ which scala
/usr/bin/scala
$ scala -version
Scala version 2.7.6.final -- Copyright 2002-2009, LAMP/EPFL
$
```

```
#!/bin/bash
# -----------------------------------------------------------------------------
# jruby.sh - Start Script for the JRuby interpreter
#
# Environment Variable Prequisites
#
#   JRUBY_OPTS    (Optional) Default JRuby command line args
#   JRUBY_SHELL   Where/What is system shell
#
#   JAVA_HOME     Must point at your Java Development Kit installation.
#
# -----------------------------------------------------------------------------

cygwin=false

# ----- Identify OS we are running under --------------------------------------
case "`uname`" in
  CYGWIN*) cygwin=true;;
  Darwin) darwin=true;;
esac

#
# Figure out the OS and cpu the same as JNA would, so the library path can be set
#
case "`uname -m`" in
  i[34567]86) JNA_CPU=i386; JNA_ALT_CPU=amd64;;
  i86pc) JNA_CPU="x86"; JNA_ALT_CPU=amd64;;
  amd64|x86_64) JNA_CPU=amd64; JNA_ALT_CPU=i386;;
  sparc*) JNA_CPU=sparc; JNA_ALT_CPU=sparcv9;;
esac
```

```
# ----- Verify and Set Required Environment Variables -------------------------

## resolve links - $0 may be a link to  home
PRG=$0
progname=`basename "$0"`

while [ -h "$PRG" ] ; do
  ls=`ls -ld "$PRG"`
  link=`expr "$ls" : '.*-> \(.*\)$'`
  if expr "$link" : '.*/.*' > /dev/null; then
    if expr "$link" : '^/' > /dev/null; then
      PRG="$link"
    else
      PRG="`dirname ${PRG}`/${link}"
    fi
  else
    PRG="`dirname $PRG`/$link"
  fi
done

JRUBY_HOME_1=`dirname "$PRG"`           # the ./bin dir
if [ "$JRUBY_HOME_1" = '.' ] ; then
  cwd=`pwd`
  JRUBY_HOME=`dirname $cwd` # JRUBY-2699
else
  JRUBY_HOME=`dirname "$JRUBY_HOME_1"`  # the . dir
fi

if [ -z "$JRUBY_OPTS" ] ; then
  JRUBY_OPTS=""
fi
```

```
if [ -z "$JAVA_HOME" ] ; then
  JAVA_CMD='java'
else
  if $cygwin; then
    JAVA_HOME=`cygpath -u "$JAVA_HOME"`
  fi
  JAVA_CMD="$JAVA_HOME/bin/java"
fi

# If you're seeing odd exceptions, you may have a bad JVM install.
# Uncomment this and report the version to the JRuby team along with error.
#$JAVA_CMD -version

JRUBY_SHELL=/bin/sh

# ----- Set Up The Boot Classpath -------------------------------------------

CP_DELIMITER=":"

# add jruby jars for command-line execution
for j in "$JRUBY_HOME"/lib/{jruby*,bsf}.jar; do
    if [ "$JRUBY_CP" ]; then
        JRUBY_CP="$JRUBY_CP$CP_DELIMITER$j"
        else
        JRUBY_CP="$j"
    fi
done

if $cygwin; then
    JRUBY_CP=`cygpath -p -w "$JRUBY_CP"`
fi
```

```
# ----- Set Up The System Classpath ------------------------------------------

if [ "$JRUBY_PARENT_CLASSPATH" != "" ]; then
    # Use same classpath propagated from parent jruby
    CP=$JRUBY_PARENT_CLASSPATH
else
    # add other jars in lib to CP for command-line execution
    for j in "$JRUBY_HOME"/lib/*.jar; do
        if [ "$CP" ]; then
            CP="$CP$CP_DELIMITER$j"
            else
            CP="$j"
        fi
    done

    if $cygwin; then
        CP=`cygpath -p -w "$CP"`
    fi
fi

if $cygwin; then
    # switch delimiter only after building Unix style classpaths
    CP_DELIMITER=";"
fi

# ----- Execute The Requested Command -----------------------------------------

if [ -z "$JAVA_MEM" ] ; then
  JAVA_MEM=-Xmx500m
fi

if [ -z "$JAVA_STACK" ] ; then
  JAVA_STACK=-Xss1024k
fi
```

```
JAVA_VM=-client
JAVA_ENCODING=""

declare -a java_args
declare -a ruby_args

java_class=org.jruby.Main

# Split out any -J argument for passing to the JVM.
# Scanning for args is aborted by '--'.
while [ $# -gt 0 ]
do
    case "$1" in
    # Stuff after '-J' in this argument goes to JVM
    -J*)
        val=${1:2}
        if [ "${val:0:4}" = "-Xmx" ]; then
            JAVA_MEM=$val
        elif [ "${val:0:4}" = "-Xss" ]; then
            JAVA_STACK=$val
        elif [ "${val}" = "" ]; then
            $JAVA_CMD -help
            echo "(Prepend -J in front of these options when using 'jruby' command)"
            exit
        elif [ "${val}" = "-X" ]; then
            $JAVA_CMD -X
            echo "(Prepend -J in front of these options when using 'jruby' command)"
            exit
        elif [ "${val}" = "-classpath" ]; then
            CP="$CP$CP_DELIMITER$2"
            shift
        elif [ "${val}" = "-cp" ]; then
            CP="$CP$CP_DELIMITER$2"
            shift
        else
            if [ "${val:0:3}" = "-ea" ]; then
                VERIFY_JRUBY="yes"
            elif [ "${val:0:16}" = "-Dfile.encoding=" ]; then
                JAVA_ENCODING=$val
            fi
            java_args=("${java_args[@]}" "${1:2}")
        fi
        ;;
     # Match switches that take an argument
     -C|-e|-I|-S) ruby_args=("${ruby_args[@]}" "$1" "$2"); shift ;;
     # Match same switches with argument stuck together
     -e*|-I*|-S*) ruby_args=("${ruby_args[@]}" "$1" ) ;;
     # Run with the instrumented profiler: http://jiprof.sourceforge.net/
     --profile)
       PROFILE_ARGS="-javaagent:$JRUBY_HOME/lib/profile.jar
-Dprofile.properties=$JRUBY_HOME/lib/profile-ruby.properties"
       JRUBY_OPTS=("${JRUBY_OPTS[@]}" "-X+C")
       VERIFY_JRUBY="yes"
       ;;
     # Run with the instrumented profiler: http://jiprof.sourceforge.net/
     --profile-all)
       PROFILE_ARGS="-javaagent:$JRUBY_HOME/lib/profile.jar
-Dprofile.properties=$JRUBY_HOME/lib/profile-all.properties"
       JRUBY_OPTS=("${JRUBY_OPTS[@]}" "-X+C")
       VERIFY_JRUBY="yes"
       ;;
     # Run with JMX management enabled
     --manage)
        java_args=("${java_args[@]}" "-Dcom.sun.management.jmxremote") ;;
     # Don't launch a GUI window, no matter what
     --headless)
        java_args=("${java_args[@]}" "-Djava.awt.headless=true") ;;
```

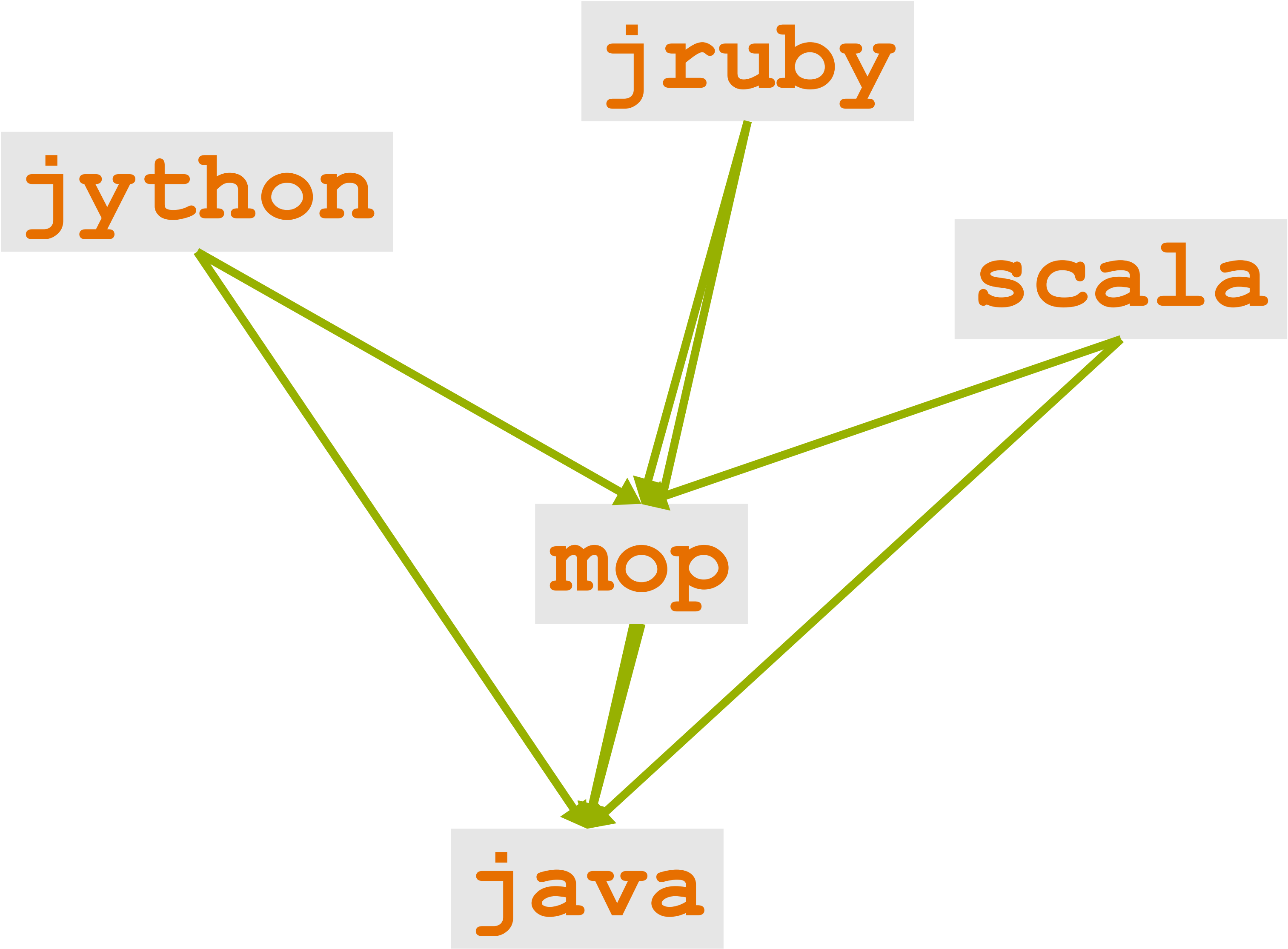
```
# Run under JDB
--jdb)
   if [ -z "$JAVA_HOME" ] ; then
     JAVA_CMD='jdb'
   else
     if $cygwin; then
       JAVA_HOME=`cygpath -u "$JAVA_HOME"`
     fi
     JAVA_CMD="$JAVA_HOME/bin/jdb"
   fi
   java_args=("${java_args[@]}" "-sourcepath" "$JRUBY_HOME/lib/ruby/1.8:.")
   JRUBY_OPTS=("${JRUBY_OPTS[@]}" "-X+C") ;;
--client)
   JAVA_VM=-client ;;
--server)
   JAVA_VM=-server ;;
--sample)
   java_args=("${java_args[@]}" "-Xprof") ;;
--1.9)
   java_args=("${java_args[@]}" "-Djruby.compat.version=RUBY1_9") ;;
--1.8)
   java_args=("${java_args[@]}" "-Djruby.compat.version=RUBY1_8") ;;
--ng-server)
   # Start up as Nailgun server
   java_class=com.martiansoftware.nailgun.NGServer
   JAVA_VM=-server
   VERIFY_JRUBY=true ;;
--ng)
   # Use native Nailgun client to toss commands to server
   nailgun_client=true ;;
# Abort processing on the double dash
--) break ;;
```

```
     # Other opts go to ruby
     -*) ruby_args=("${ruby_args[@]}" "$1") ;;
     # Abort processing on first non-opt arg
     *) break ;;
    esac
    shift
done

# Force file.encoding to UTF-8 when on Mac, since Apple JDK defaults to MacRoman (JRUBY-
if [[ $darwin && -z "$JAVA_ENCODING" ]]; then
  java_args=("${java_args[@]}" "-Dfile.encoding=UTF-8")
fi

# Add a property to report memory max
JAVA_OPTS="$JAVA_OPTS $JAVA_VM -Djruby.memory.max=${JAVA_MEM:4} -Djruby.stack.max=${JAVA_

# Append the rest of the arguments
ruby_args=("${ruby_args[@]}" "$@")

# Put the ruby_args back into the position arguments $1, $2 etc
set -- "${ruby_args[@]}"

JAVA_OPTS="$JAVA_OPTS $JAVA_MEM $JAVA_STACK"
JNA_OS="`uname -s | tr '[:upper:]' '[:lower:]'`"
case "$JNA_OS" in
darwin) JNA_PATH="$JRUBY_HOME/lib/native/darwin";;
  *) JNA_PATH="$JRUBY_HOME/lib/native/${JNA_OS}-${JNA_CPU}:$JRUBY_HOME/lib/native/${JNA_
{JNA_ALT_CPU}";;
esac
#JAVA_OPTS="$JAVA_OPTS -Djna.boot.library.path=$JNA_PATH"
JAVA_JNA="-Djna.boot.library.path=$JNA_PATH"
```

```
JFFI_BOOT=""
for d in $JRUBY_HOME/lib/native/*`uname -s`; do
  if [ -z "$JFFI_BOOT" ]; then
    JFFI_BOOT="$d"
  else
    JFFI_BOOT="$JFFI_BOOT:$d"
  fi
done
JFFI_OPTS="-Djffi.boot.library.path=$JFFI_BOOT"

if $cygwin; then
  JRUBY_HOME=`cygpath --mixed "$JRUBY_HOME"`
  JRUBY_SHELL=`cygpath --mixed "$JRUBY_SHELL"`

  if [[ ( "${1:0:1}" = "/" ) && ( ( -f "$1" ) || ( -d "$1" )) ]]; then
    win_arg=`cygpath -w "$1"`
    shift
    win_args=("$win_arg" "$@")
    set -- "${win_args[@]}"
  fi

  # fix JLine to use UnixTerminal
  stty -icanon min 1 -echo > /dev/null 2>&1
  if [ $? = 0 ]; then
    JAVA_OPTS="$JAVA_OPTS -Djline.terminal=jline.UnixTerminal"
  fi

fi
```

```
if [ "$nailgun_client" != "" ]; then
  if [ -f $JRUBY_HOME/tool/nailgun/ng ]; then
    exec $JRUBY_HOME/tool/nailgun/ng org.jruby.util.NailMain $JRUBY_OPTS "$@"
  else
    echo "error: ng executable not found; run 'make' in tool/nailgun"
    exit 1
  fi
else
if [ "$VERIFY_JRUBY" != "" ]; then
  if [ "$PROFILE_ARGS" != "" ]; then
      echo "Running with instrumented profiler"
  fi

  "$JAVA_CMD" $PROFILE_ARGS $JAVA_OPTS "$JAVA_JNA" "$JFFI_OPTS" "${java_args[@]}" -classp
"$JRUBY_CP$CP_DELIMITER$CP$CP_DELIMITER$CLASSPATH" \
    "-Djruby.home=$JRUBY_HOME" \
    "-Djruby.lib=$JRUBY_HOME/lib" -Djruby.script=jruby \
    "-Djruby.shell=$JRUBY_SHELL" \
    $java_class $JRUBY_OPTS "$@"

  # Record the exit status immediately, or it will be overridden.
  JRUBY_STATUS=$?

  if [ "$PROFILE_ARGS" != "" ]; then
      echo "Profiling results:"
      cat profile.txt
      rm profile.txt
  fi
  if $cygwin; then
    stty icanon echo > /dev/null 2>&1
  fi
  exit $JRUBY_STATUS
```

```
else
  if $cygwin; then
    # exec doed not work correctly with cygwin bash
    "$JAVA_CMD" $JAVA_OPTS "$JAVA_JNA" "$JFFI_OPTS" "${java_args[@]}"
-Xbootclasspath/a:"$JRUBY_CP" -classpath "$CP$CP_DELIMITER$CLASSPATH" \
      "-Djruby.home=$JRUBY_HOME" \
      "-Djruby.lib=$JRUBY_HOME/lib" -Djruby.script=jruby \
      "-Djruby.shell=$JRUBY_SHELL" \
      $java_class $JRUBY_OPTS "$@"

    # Record the exit status immediately, or it will be overridden.
    JRUBY_STATUS=$?

    stty icanon echo > /dev/null 2>&1

    exit $JRUBY_STATUS
  else
    exec "$JAVA_CMD" $JAVA_OPTS "$JAVA_JNA" "$JFFI_OPTS" "${java_args[@]}"
-Xbootclasspath/a:"$JRUBY_CP" -classpath "$CP$CP_DELIMITER$CLASSPATH" \
      "-Djruby.home=$JRUBY_HOME" \
      "-Djruby.lib=$JRUBY_HOME/lib" -Djruby.script=jruby \
      "-Djruby.shell=$JRUBY_SHELL" \
      $java_class $JRUBY_OPTS "$@"
  fi
fi
fi

# Be careful adding code down here, you might override the exit
# status of the jruby invocation.
```



jruby

jython

scala

mop

java

120
140
160
180
90
100
110
120
km/h
1

Why should I care?

Download/install experience
Implementation simplification
Startup time

http://openjdk.java.net/projects/jdk7

coin
jigsaw
mlvm

Mark Reinhold
Principal Engineer
Sun Microsystems

http://blogs.sun.com/mr
http://openjdk.java.net