

GNU Classpath

Core Classes for a Diversity
of Java Virtual Machines

February 2004



- **Sascha Brawer**
brawer@dandelis.ch
<http://www.dandelis.ch/people/brawer/>
- **Mark Wielaard**
mark@klomp.org
<http://www.klomp.org/mark/>

Overview

- Overview
 - Motivation
 - Anatomy of a Java-like system
 - Documentation, Quality assurance, Releases
- Demo
- Details
 - Current state
 - VMs using GNU Classpath
- How you can help

Motivation

- Currently, most free software is written in C

The Good

Close to hardware
Fast execution, even
with bad compilers
Large code base,
many libraries
Ubiquitous

The Bad

Easy to make bugs
Lacks “modern”
language concepts
 (“modern” = 1980’ies)
Hard to write
portable code

The Ugly

Libraries not
well integrated
Difficult to learn

Motivation

- Java is a good foundation for many projects
 - Type-safety
 - Avoids many typical bugs, crashes, security problems
 - More opportunities for optimizing compilers
 - Modular, object-oriented, concurrent, dynamic
 - Easier to write structured, portable code
 - Very rich library, reasonably well designed (mostly)
 - Large developer base (“COBOL of the 1990’ies”)
 - There exist lots of other good (even better?) languages: Oberon, Eiffel, O’Caml, Haskell, Erlang, TOM, Cedar, ...
 - But: They have not many free applications/libraries

Motivation

- But: Java does not solve every problem
 - Over-hyped as a solution to everything
 - Bad reputation for wasting resources (CPU, RAM)
 - It is easy to write inefficient programs
 - Early implementations were very slow
 - Type-safety and memory management have their cost
 - Often over-estimated: Array bounds checking (→ no buffer overflows) costs ~2% execution time, avoids ~50% security-related bugs
 - Syntax similar to C++ → not very easy to learn

Motivation

- Java is a compromise
 - Not necessarily ideal, but reasonable
- Plenty of free software is being written in Java
 - Sourceforge.net: 11'032 Java projects

Why GNU Will Be Compatible with UNIX

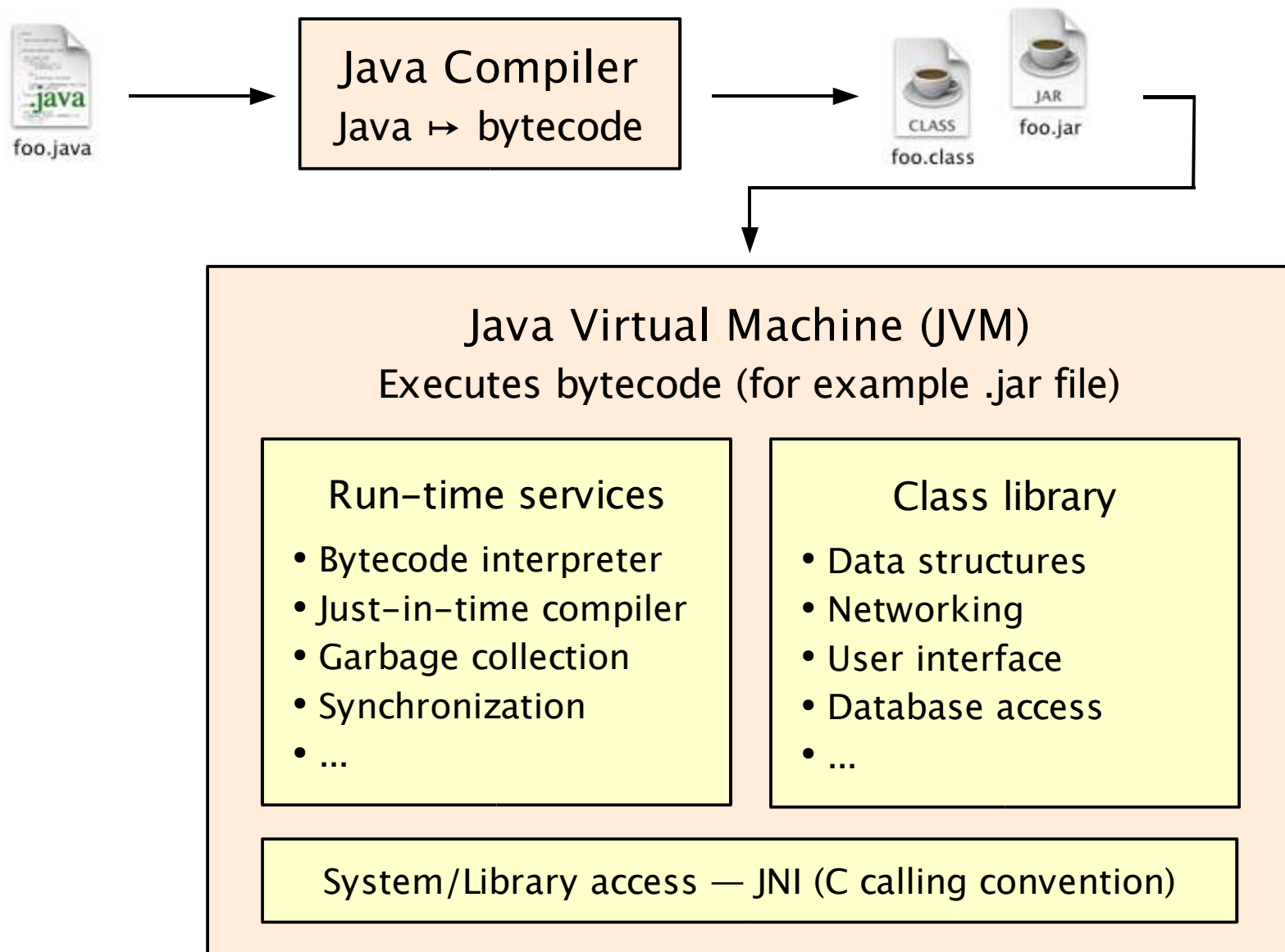
Unix is not my ideal system, but it is not too bad. The essential features of Unix seem to be good ones, and I think I can fill in what Unix lacks without spoiling them. And a system compatible with Unix would be convenient for many other people to adopt.

— *Richard Stallman*
The GNU Manifesto, 1985

Motivation

- Freedom!
 - Freedom to use, study, adapt, improve and share
 - Freedom to innovate
 - Not controlled by any single entity
 - Future: Possibility for natural, proven (vs. committee-dictated) standards

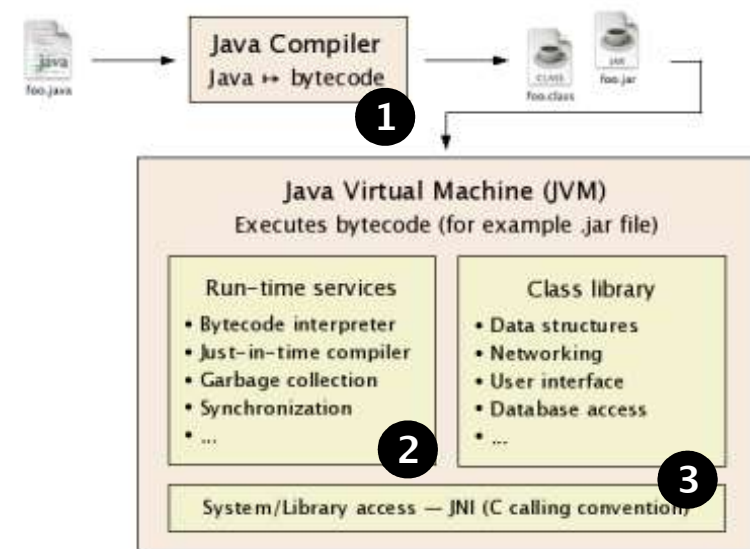
Anatomy of a Java-like system



Anatomy of a Java-like system

① Java → bytecode compiler

- gcj, jikes, kjc, Eclipse, ...
- Complete up to Java 1.4
 - 1.5 will need some work for generic types: `List<A>`



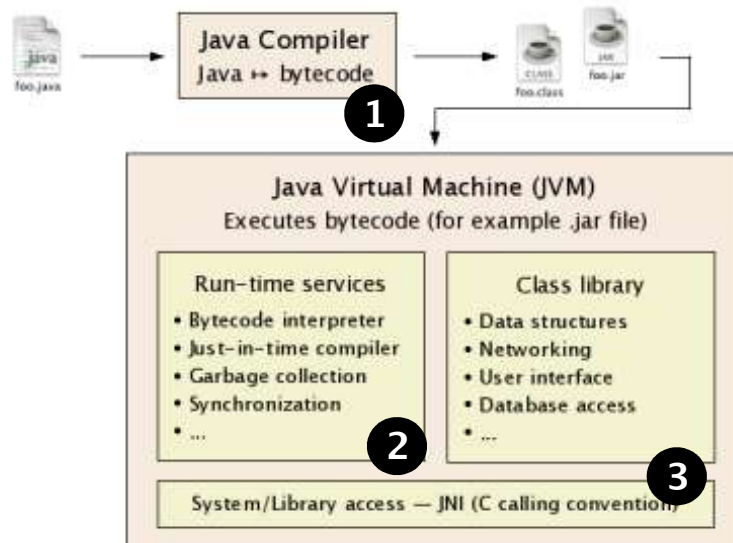
② Run-time services

- About 15 JVM projects
 - Very diverse goals
 - Almost no common code (yet)

③ Class library

- `java.{lang, math, ...}`,
`javax.{mail, crypto, ...}`
- Some C code for POSIX-like systems

Anatomy of a Java-like system



Very coarse estimates!

Generated using <http://dwheeler.com/sloccount/>

“Basic COCOMO”

1 Java compiler

Ex.: Jikes Compiler
71,000 lines of code
~ 17.5 person-years
~ USD 2.4 Mio.

2 Run-time serv.

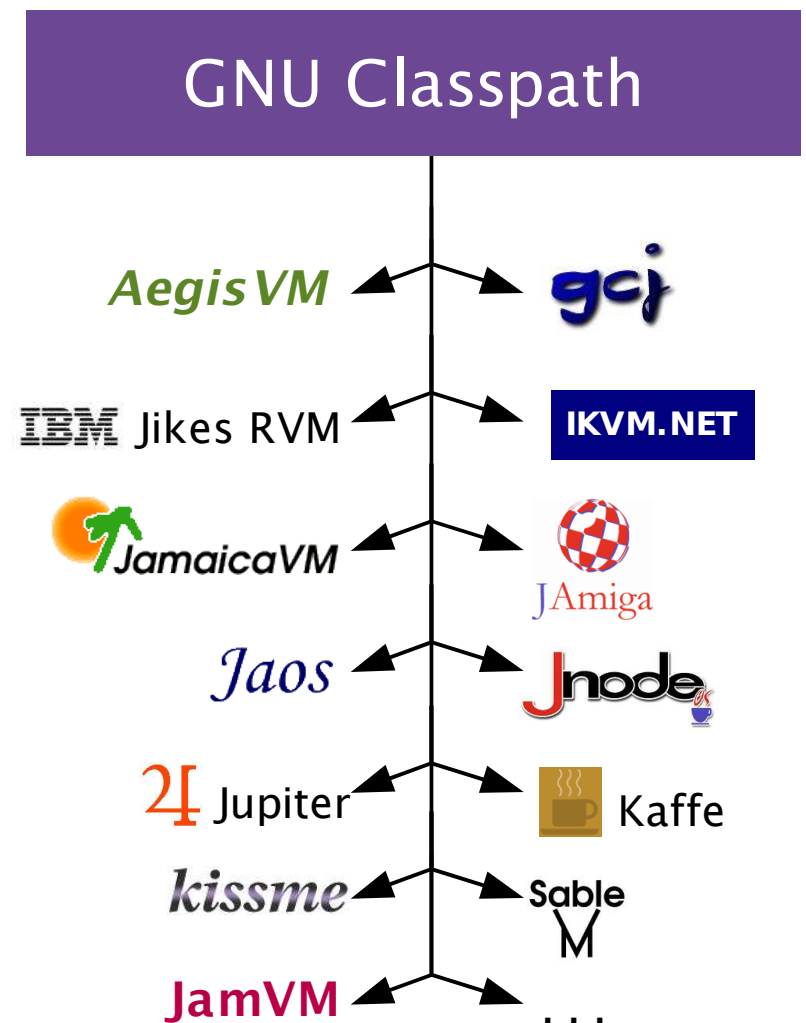
Ex.: Kissme VM
59,000 lines of code
~ 14.5 person-years
~ USD 2.0 Mio.

3 Class library

Ex.: GNU Classpath
233,000 lines of code
~ 61.5 person-years
~ USD 8.3 Mio.

GNU Classpath

- GNU Classpath is shared among many VM projects
 - “Upstream” provider for the class library
 - java.*, most javax.*
 - Mainly for historical reasons, certain javax.* are provided by other projects
 - Common code base, bug tracking, ...
 - Very permissive license



Free software is flexible

- Done with GNU Classpath:
 - Compiling Java source → stand-alone executable
 - Compiling Java bytecode → CIL (bytecode of .NET)
 - Operating Systems with type-safe kernels
 - JVM for multiprocessor clusters (128 CPUs, Myrinet)
 - Embedded systems with real-time guarantees
 - Alternative access to native system/libs (non-JNI)

Free Software can be adapted
to arbitrary needs → innovation

Documentation

- Documentation is very important
 - Often neglected
 - Many APIs have reasonable docs
 - There certainly are exceptions
- Urgently needed:
 - High-level overview
 - Manual for free environments

the stream as well. The class implementing this interface must figure out how

This interface can be used to provide object persistence. When an object is to be saved, the `writeExternal` method is called to save state. When the object is restored, an instance is created using the constructor and the `readExternal` method is used to restore the state.

Author:

Aaron M. Renn (arenn@urbanophile.com)

Method Summary	
void	readExternal(ObjectInput in) This method restores an object's state by reading in the instance data from the stream.
void	writeExternal(ObjectOutput out) This method is responsible for writing the instance data of an object to the stream.

Method Details

readExternal

```
public void readExternal(ObjectInput in)
```

This method restores an object's state by reading in the instance data from the stream. Note that this stream is not a subclass of `InputStream`, but rather is a class that implements the `ObjectInput` interface. That interface provides a mechanism for reading in Java data types from a stream.

Note that this method must be compatible with `writeExternal`. It must read the instance data in the exact order they were written.

If this method needs to read back an object instance, then the class for that operation fails, then this method throws a `ClassNotFoundException`.

Parameters:

`in` - An `ObjectInput` instance for reading in the object state

Quality assurance



MAUVE

<http://sources.redhat.com/mauve/>

- Test suite
 - ~100,000 tests
 - We need many more!
- Key to reliability
- Tests are very easy to write

```
// Tags: JDK1.0
package gnu.testlet.java.util.Stack;
import java.util.Stack;
import gnu.testlet.*;

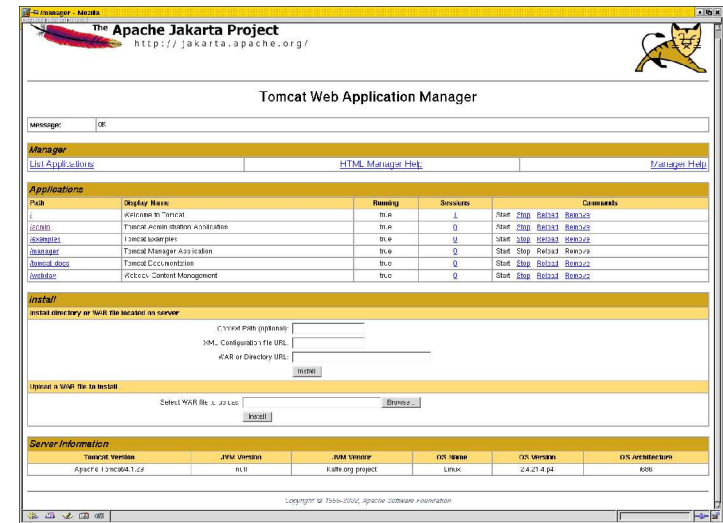
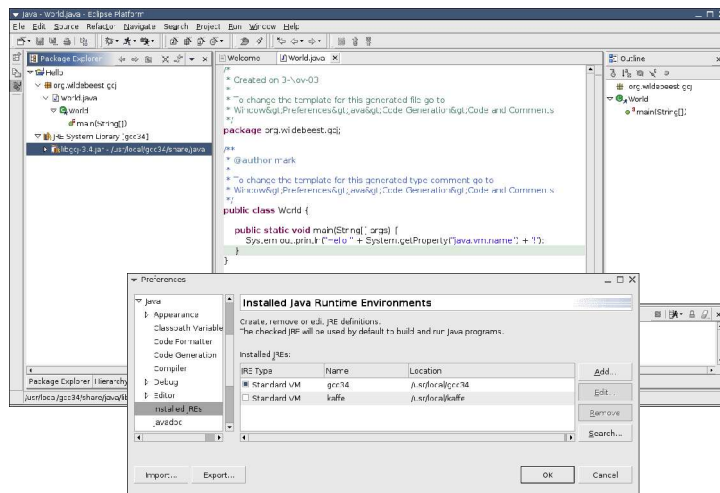
public class empty
    extends Testlet
{
    public void test(TestHarness h)
    {
        Stack stack = new Stack();
        // Check #1.
        h.check(stack.empty());
        // Check #2.
        stack.push("abc");
        h.check(!stack.empty());
    }
}
```

Test for java.util.Stack.empty()

Releases

- Now: Version 0.08
- Time-based release schedule
 - Every three months, a new 0.0x release
- Already quite usable for real applications
 - “0.08” may be too modest a name
- For v1.0, we need:
 - Fixed VM interface
 - Complete implementation of supported packages
 - Define “supported”!
 - Full test coverage
 - Full documentation of the API
 - Start of a real manual?

Demos: Eclipse, Tomcat



- Eclipse IDE

- UI: SWT (not AWT)
- eclipse.org

- Tomcat

- Container for Servlets and JavaServer Pages
- jakarta.apache.org



Package Explorer

- Hello
 - org.wildebeest.gcj
 - World.java
 - World
 - main(String[])
- JRE System Library [gcc34]
 - libgcj-3.4.jar - /usr/local/gcc34/share/java

Package Explorer | Hierarchy

/usr/local/gcc34/share/java/libgcj-3.4.jar

```
Welcome | World.java x
/*
 * Created on 3-Nov-03
 *
 * To change the template for this generated file go to
 * Window>Preferences>Java>Code Generation>Code and Comments
 */
package org.wildebeest.gcj;

/**
 * @author mark
 *
 * To change the template for this generated type comment go to
 * Window>Preferences>Java>Code Generation>Code and Comments
 */
public class World {

    public static void main(String[] args) {
        System.out.println("Hello " + System.getProperty("java.vm.name") + "!");
    }
}
```

Outline

- org.wildebeest.gcj
 - World
 - main(String[])

Console [<terminated> /usr/local/kaffe/bin/java (11/03/2003 8:55 PM)]

Hello Kaffe!

Preferences

- Java
 - Appearance
 - Classpath Variable
 - Code Formatter
 - Code Generation
 - Compiler
 - Debug
 - Editor
 - Installed JREs
 - Javadoc

Installed Java Runtime Environments

Create, remove or edit JRE definitions.
The checked JRE will be used by default to build and run Java programs.

Installed JREs:

JRE Type	Name	Location
<input checked="" type="checkbox"/>	Standard VM gcc34	/usr/local/gcc34
<input type="checkbox"/>	Standard VM kaffe	/usr/local/kaffe

Buttons: Add..., Edit..., Remove, Search...

Buttons: Import..., Export..., OK, Cancel



Tomcat Web Application Manager

Message: OK

Manager

[List Applications](#)[HTML Manager Help](#)[Manager Help](#)

Applications

Path	Display Name	Running	Sessions	Commands
/	Welcome to Tomcat	true	1	Start Stop Reload Remove
/admin	Tomcat Administration Application	true	0	Start Stop Reload Remove
/examples	Tomcat Examples	true	0	Start Stop Reload Remove
/manager	Tomcat Manager Application	true	0	Start Stop Reload Remove
/tomcat-docs	Tomcat Documentation	true	0	Start Stop Reload Remove
/webdav	Webdav Content Management	true	0	Start Stop Reload Remove

Install

Install directory or WAR file located on server

Context Path (optional): XML Configuration file URL: WAR or Directory URL:

Upload a WAR file to install

Select WAR file to upload

Server Information

Tomcat Version	JVM Version	JVM Vendor	OS Name	OS Version	OS Architecture
Apache Tomcat/4.1.29	null	Kaffe.org project	Linux	2.4.21.4.p4	i686

Current state

- J2SE 1.3 / 1.4 (Desktop)
 - Source: GNU Classpath
 - License: GPL + “Compiler exception”
 - Allows static linking with proprietary closed-source code and into embedded systems → very permissive
 - **No formal compliance with any specification**

java.beans[.beancontext], java.io, java.lang[.ref, .reflect], java.math,
java.net, java.nio.charset[.spi] (only SPI), java.rmi[.activation, .dgc,
.registry, .server], java.security[.acl, .cert, .interfaces, .spec], java.sql,
java.text, java.util[.jar, .logging, .zip]
javax.naming[.directory, .ldap, .spi], javax.swing.[border, plaf],
javax.transaction[.xa]

Current state

- J2SE 1.4 (Desktop), partial implementation

java.applet, java.awt, java.awt.event, java.awt.geom,
java.awt.image[.renderable], java.nio[.channels, .spi], java.util.prefs
javax.accessibility, javax.print[*] (only SPI), javax.swing.undo

- J2SE 1.4 (Desktop), lots of work/nothing there

java.awt.color, java.awt.font, java.awt.im[.spi],
java.nio.charset (service providers)
javax.imageio[*], javax.print (service providers), javax.rmi[.CORBA],
java.security.auth[*], javax.swing[.colorchooser, .event,
.filechooser, .plaf.basic, .plaf.metal, .plaf.multi, .table,
.text, .text.html, .text.html.parser, .text.rtf, .tree]
org.ietf.jgss, org.omg[*]

Current state

- Various standard packages
 - Licenses: Various (but all “free”)
 - **No formal compliance with any specification**

javax.crypto

Service providers for
many crypto algorithms
GNU Crypto

java.util.regex

GNU regex

javax.sound

Service providers for
ALSA, esd, lame, ogg
Tritonus Project

javax.net

javax.net.ssl
javax.security.cert
SSL/TLS Support
Jessie

javax.activation

javax.infobus
javax.mail
javax.servlet
GNU Classpath
Extensions

javax.speech

Text-to-Speech
(no recognition)
FreeTTS

Current state

- J2EE 1.4 (Server)
 - Multi-tier stack: web services/enterprise computing
 - Multiple free implementations: JBoss.org, Jonas
 - Needs some (not much) work to combine it all
 - Conformance: JBoss.org has recently licensed the J2EE1.4 compliance kit from Sun microsystems
 - Now making sure that JBoss.org satisfies the test suite
 - Will afterwards be certified as compliant

javax.activation, javax.ejb[.spi], javax.enterprise[.*], javax.jms,
javax.mail[.*], javax.management[.*], javax.resource[.*],
javax.security.jacc, javax.servlet[.*], javax.transaction[.xa], javax.xml.*

Current state

- Plenty of free software is being written in Java
 - sourceforge.net: 11,032 Java projects
 - Support for multiple other programming languages
 - GNU KAWA: Scheme, Common Lisp, eLisp, ECMAScript, ...
 - Jython, JRuby
- Traditional free libraries with Java bindings
 - GTK+, GNOME, glade, gconf, vte, ...
 - Will be part of the official language bindings in GNOME 2.6
 - GNU gettext, getopt, readline, ...

VMs using Classpath: gcj



GNU Compiler for the Java Progr. Language

Free Software Foundation, Boston, USA

<http://gcc.gnu.org/java/>

- Java front-end to the GCC compiler
 - Java \simeq C++
 - GCC backend performs \pm same optimizations as with other languages (C, C++, Fortran, Ada, Pascal, Cobol, ...)
 - fast execution
 - Static compilation produces stand-alone executables
 - But Java is a dynamic language!
 - → Run-time library contains a simple bytecode interpreter for dynamically loaded classes

VMs using Classpath: gcj



GNU Compiler for the Java Progr. Language

Free Software Foundation, Boston, USA

<http://gcc.gnu.org/java/>

- GCJ uses its own class library (“libjava”)
 - Being merged with GNU Classpath
 - Some parts are tricky to merge (optimizations)
 - Non-standard interface for native code
 - “Compiled Native Interface” = C++
 - Support for standard JNI has been added, but JNI is slower than CNI (and much more verbose)

VMs using Classpath: Jikes RVM



Jikes RVM

Jikes Research VM (“Jalapeño”)

IBM T.J. Watson Research Lab, Hawthorne NY, USA

<http://www-124.ibm.com/developerworks/oss/jikesrvm/>

- Research platform for high-speed VMs
 - ~ 70 academic publications
 - Just-in-time compilers produce very efficient code
- Fully written in Java
 - Compiler, Assembler, Garbage Collector, ...

VMs using Classpath: IKVM.NET

IKVM.NET

Jeroen Frijters, Sumatra Software
Wassenaar, The Netherlands

<http://www.ikvm.net/>

- JVM for Mono and Microsoft .NET
 - Java bytecodes ⇒ “parse trees” ⇒ CIL instructions
 - Uses run-time services of the CLI platform
 - Synchronization, Garbage Collection, etc.
 - But also optimization: Loaded CIL instructions get optimized by platform compiler

VMs using Classpath: JamVM

JamVM

Rob Lougher, United Kingdom
<http://jamvm.sourceforge.net/>

- Very compact, interpreted VM
 - Reasonably fast for a pure interpreter: 4x faster than Kaffe's interpreter, 2x than Blackdown int. (on PPC)
 - Efficient synchronization ("Tazuki bi-modal locks")
 - Architectures: IA32, PowerPC, ARM
- Supports full specification
 - Including reflection API, JNI and Thread.interrupt

VMs using Classpath: Jaos

Jaos

Patrik Reali, ETH Zürich, Switzerland

<http://www.oberon.ethz.ch/jaos/>

- JVM for the Active Oberon System
 - Uses run-time services of “Bluebottle” object kernel
 - No interpreter (all code is JIT-compiled)
 - Java classes can extend Oberon classes and vice versa
 - All low-level methods in Active Oberon (no C code)

VMs using Classpath: JNode OS



<http://www.jnode.org/>

- Operating System fully written in Java
 - Including device drivers etc.
 - JVM written in Java and Assembly, no C code

VMs using Classpath: Jupiter



Jupiter VM

University of Toronto, Canada
Computer Engineering Research Group

<http://www.eecg.toronto.edu/jupiter/>

- Highly scalable JVM
 - Key research questions:
 - Heuristics for memory locality
 - Parallel Garbage Collection
 - Efficient synchronization primitives
 - Cluster of 128 PCs, connected via Myrinet
 - Not in active maintenance anymore (?)

VMs using Classpath: Kaffe



Kaffe

<http://www.kaffe.org/>

- First free JVM
 - Traditional VM: interpreter + JIT compiler
 - Complete tool set: compiler, appletviewer, ...
 - Ported to 56 platforms: “NetBSD of free Java”
 - License: GPL
 - Very active development, mostly the first to adopt new libraries and extensions

VMs using Classpath: SableVM



SableVM

Sable Group, Univ. McGill and UQAM,
Montréal, Canada

<http://www.sablevm.org/>

- Clean, modular code
 - Strongly aims for “real world” application support
 - Also ideal for academic research into traditional byte code interpreters
 - Easy to maintain and extend
 - About 60K lines of very portable code
 - “1 hour for a new port” [to a POSIX system]

VMs using Classpath: Others



“Coffee for your Amiga”

Your project here?

kissme

Object Persistence

AegisVM

Modular bytecode
verification

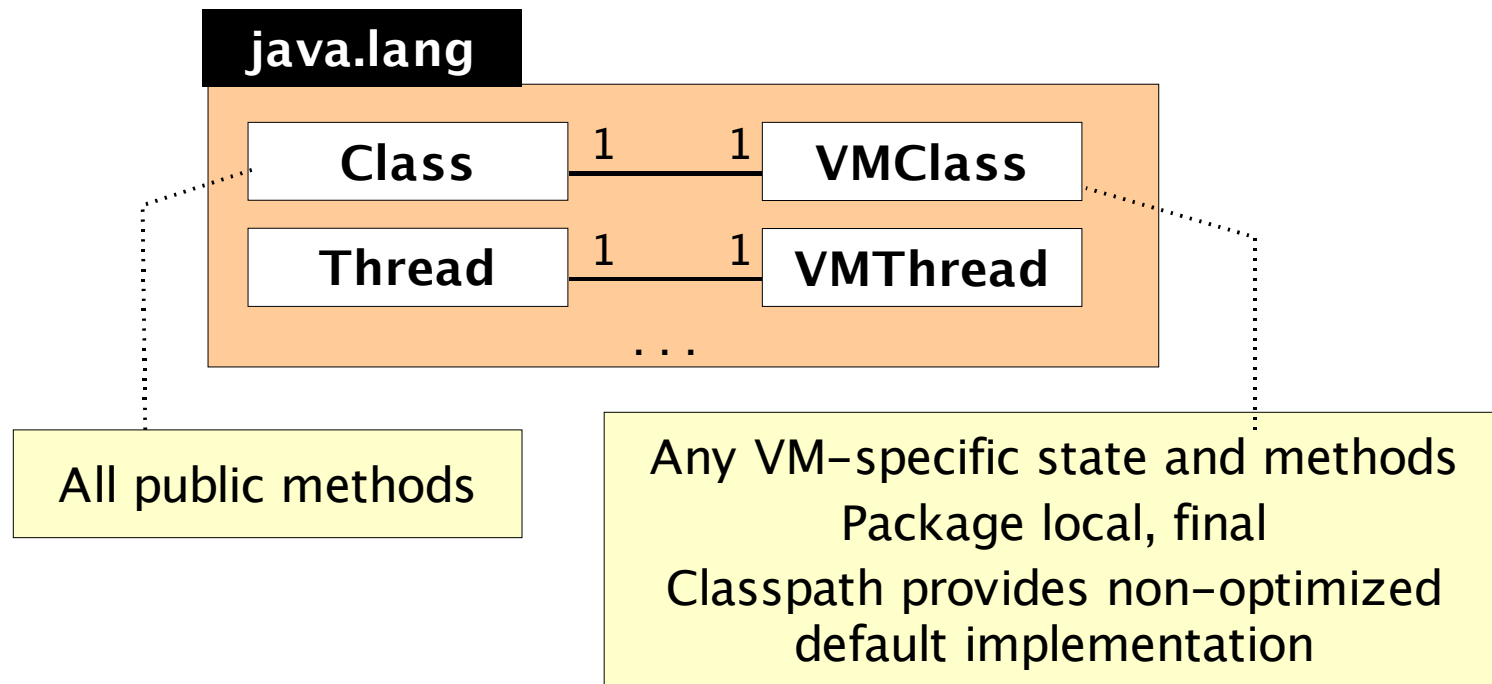


Real-time JVM
for embedded systems

Very different VMs

- GNU Classpath is used by very diverse VMs
 - Very different (sometimes conflicting) design goals
 - Extremes: gcj ↔ Jaos, JNode OS
 - gcj: Java \simeq C++
 - Jaos, JNode OS: Cannot support any C code, no POSIX

Very different VMs



- Diverse VMs → delegation/façade pattern
 - Clear interface between Classpath and VM
 - Plan: Common glibj.zip/.jar with all core classes
 - VM should inline calls to pack.-local final methods

How you can help

- Any help is greatly appreciated
 - Test Java programs on free Virtual Machines
 - Write test cases for Mauve
 - Write intellegible documentation
 - Implement library classes
 - Fund development
- How to proceed
 - Look at the task list
 - <http://www.gnu.org/software/classpath/>
 - Both easy and tricky tasks
 - Ask us for details: classpath@gnu.org
 - Copyrights must get assigned to FSF
 - Happy hacking!