Small-Footprint Implementations of the Java Virtual Machine

John Garvin

March 22, 2000
Platforms targeted:

- PalmPilots
- pagers
- cellular phones
- PDAs
Existing implementations:

Sun — KVM for a variety of platforms

Wabasoft — open-source WabaVM for PalmOS and Windows CE

NSIcom — JSCP (Java Software Co-processor) for embedded systems; licensed by Microsoft for Windows CE

TransVirtual — Kaffe
Regular JVM open-source; recommended by GNU
Website: “Versions for Internet appliances and embedded systems can be licensed”
Characteristics of small JVMs:

- small code size

- simplified runtime environment—GC, etc.

- Not all there (nonessential elements left out)
KVM

Sun’s JVM for small devices (16 or 32-bit processors 25MHz and up; approximately 128K memory)

Code size: 40K in most common configuration

Part of Java 2 Micro Edition (J2ME)—includes KVM and library of APIs
Prevalence of KVM

- PalmPilot
- BellSouth wireless
- NTT DoCoMo
Sun’s other small JVMs:

PersonalJava — uses regular JVM, libraries in J2ME and more

EmbeddedJava — full JVM, except loading and verification are optional; almost all packages in API are optional. For closed, black-box configurations.

JavaCard — very small subset of JVM
KVM (J2ME) includes a subset of libraries, called core APIs

Each configuration must include core APIs and whatever other classes are appropriate for the platform.

Example: KVM for PalmPilot

`java.*`:

io: (78 in full JVM)
`IOException.java  InputStream.java  OutputStream.java  Serializable.java`

lang: (80+ in full JVM)
`ArithmeticException.java  *Math.java`
`ArrayIndexOutOfBoundsException.java  NegativeArraySizeException.java`
`ArrayStoreException.java  *NullPointerException.java`
`*Boolean.java  *NumberFormatException.java`
`*Byte.java  *Object.java`
`*Character.java  *OutOfMemoryError.java`
`Class.java  Runnable.java`
ClassCastException.java *Runtime.java
ClassNotFoundException.java *RuntimeException.java
Error.java *Short.java
*Exception.java StackOverflowError.java
IllegalArgumentException.java *String.java
IllegalAccessException.java *StringBuffer.java
IllegalMonitorStateException.java *System.java
IndexOutOfBoundsException.java Thread.java
InstantiationException.java *Throwable.java
InterruptedException.java *VirtualMachineError.java
InstantiationException.java *Void.java
Long.java

net: (39 in full JVM)
Socket.java SocketInputStream.java
SocketException.java SocketOutputStream.java

util: (54+ in full JVM)
*BitSet.java *Enumeration.java NoSuchElementException.java
*Dictionary.java *Hashtable.java *Vector.java

* = core APIs
KVM leaves out java.applet, awt, beans, math, rmi, security, sql, text

Subset of methods in each library
Many constraints found in the JVM specification are relaxed in KVM.

- Large data types (long, float, double) are optional

- Multi-dimensional arrays are optional

- Verification can be done by an outside system (Sun is working on a mechanism for outside verification and delivery)

- Errors can be handled in a platform-dependent way
• Threads and Event Handling can follow a different, simpler model than that of the JVM specification

• JNI—can use a simpler method of calling native functions

• Class loaders—different mechanism for loading classes than Java class loaders

• Object finalization is optional

• Many maximum size limitations can be reduced: classes in a package, elements per array, etc.

• Startup; initial class and method
KVM sets forth a standard of sorts

...but no other JVM implementations seem to follow it.

- WabaVM follows its own modifications of JVM specs

- JSPC is based on PersonalJava, not KVM
KVM does not include AWT.

Project kAWT seeks to create a lightweight version of AWT for KVM.

Currently version 0.25; not fully functional

Not open-source; free for private and educational use
WabaVM: a subset of the JVM for Palm and Windows CE

Omitted parts are specified by opcode:

- datatypes: longs and doubles omitted
- threads: monitorenter and monitorexit
- exeptions: athrow

Doesn't do quick or reserved opcodes, either.
Garbage Collection

Both KVM and WabaVM use a simple mark-sweep GC rather than more complicated types. Mark-sweep GC works well for small heap sizes.