VM/ESA JAVA Update

VM/VSE Technical Conference
Orlando, May/June 2000
Session # M61

Christine T. Casey
VM/ESA Development, Endicott
caseyct@us.ibm.com

Richard F. Lewis
IBM Washington System Center
rflewis@us.ibm.com

RETURN TO INDEX

Disclaimer

Trademarks

The following are trademarks of International Business Machines Corporation. Those identified with an (*) are registered trademarks of International Business Machines

- Enterprise Systems Architecture/370
- Enterprise Systems Architecture/390
- ESA/370
- ESA/390
- IBM*
- System/370
- VM/ESA*
- OpenEdition

The following are trademarks of the corporations identified

- Windows 95 - is a registered trademark of Mircrosoft Corporation
- JAVA - is a trademark of Sun Microsystems
- "Write once, run anywhere" - is a trademark of Sun Microsystems

Agenda

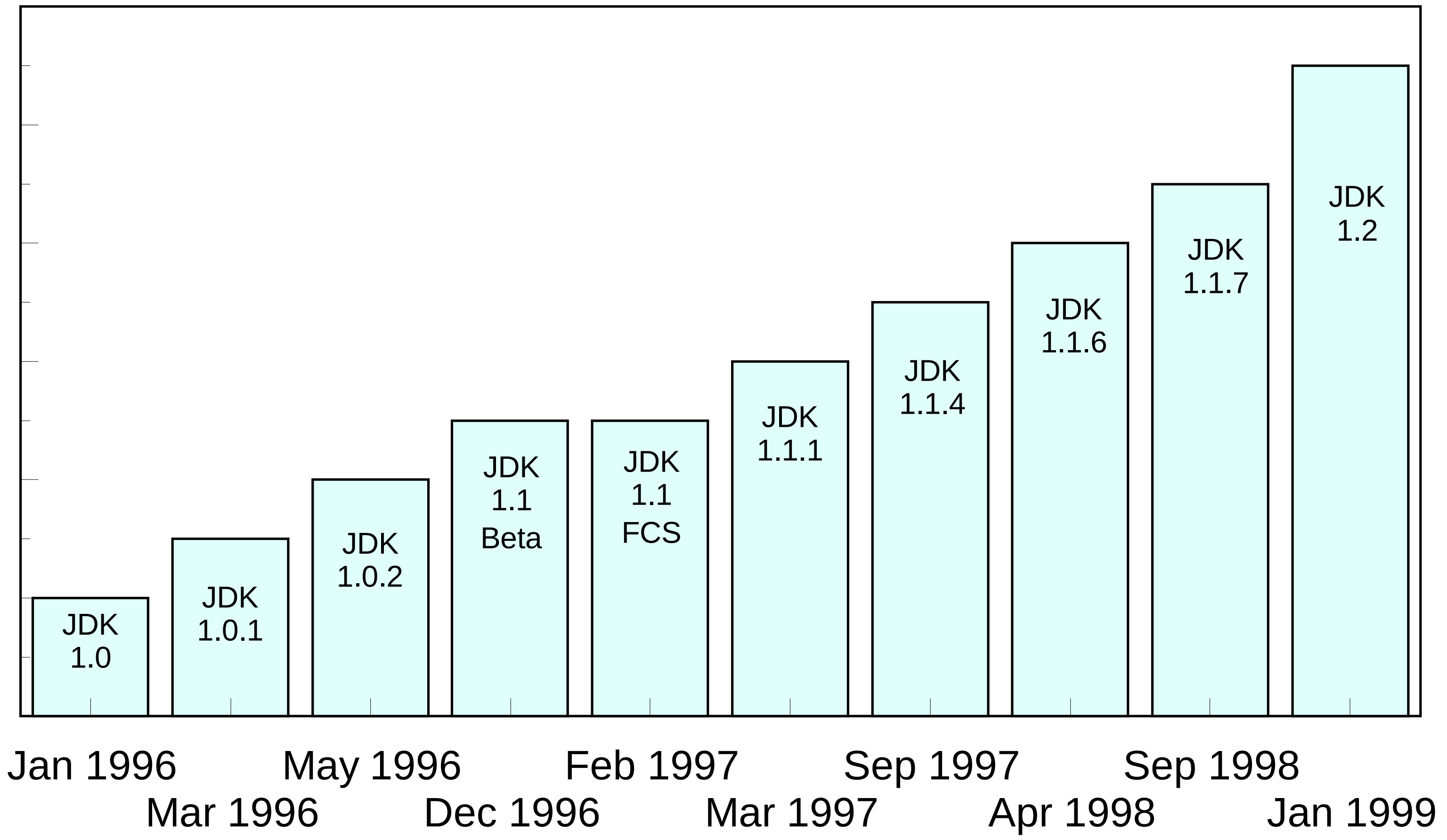
- Introduction
- Java on VM/ESA (V2R3 and V2R4)
- Notes on JDK utility programs
- Sample server
- Summary

Introduction

Java History

- Started life in 1991 as OAK (project green)
  - Failed to win bid in consumer electronic market
  - Small, portable, fast, and safe
- Explosive growth in WWW
  - Provided Sun an ideal opportunity to capitalize on this technology
  - Early 1995 - Sun releases alpha level of web browser called HotJava
  - Ushered in new generation of Internet programming

Java History (Cont.)



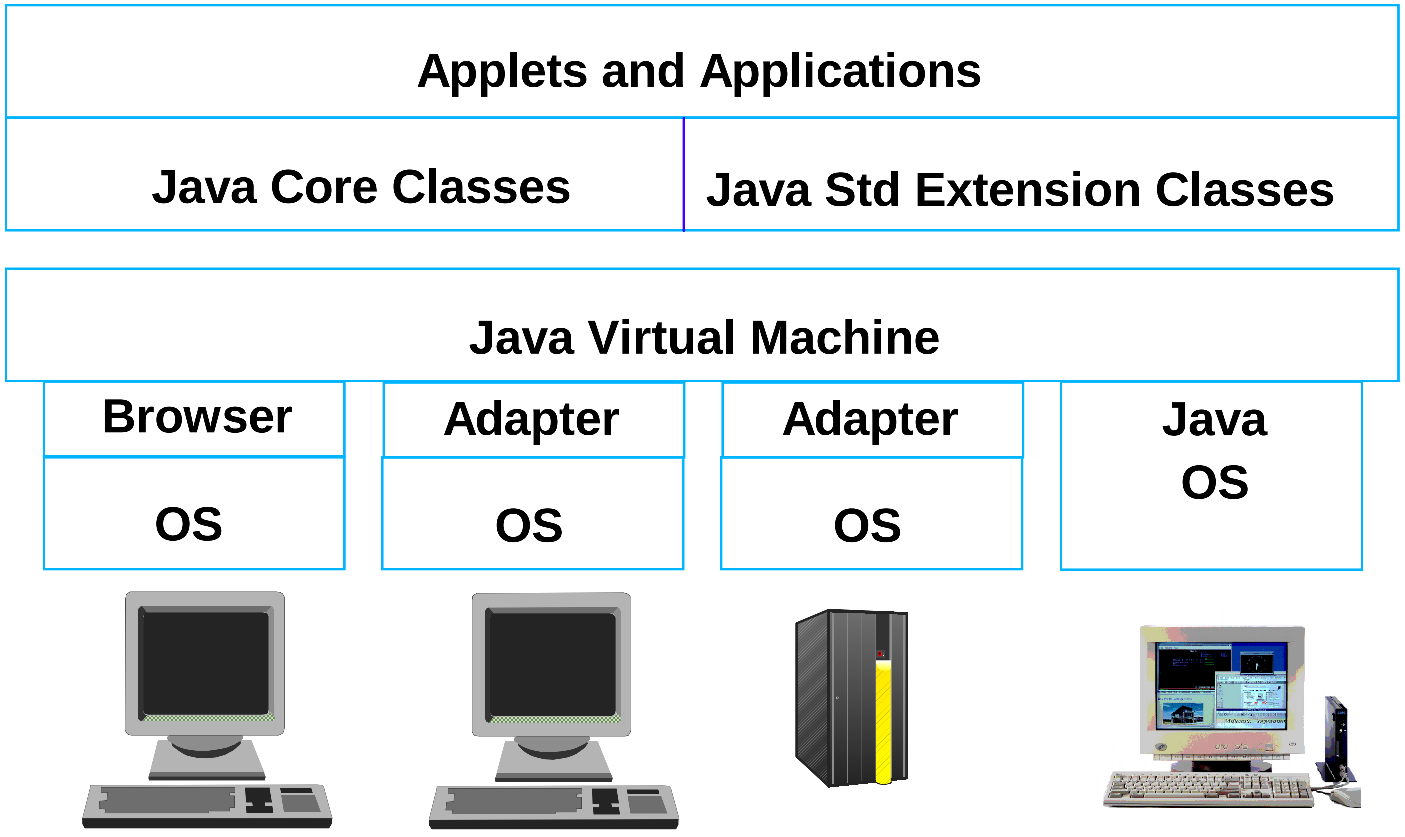
What Is Java?

- A programming language that is
  - A simpler, safer dialect of C and C++ with things that are complex, dangerous, or error-prone eliminated
    - No preprocessor, pointer, include files, operator overloading...
  - Object oriented with a rich set of classes
    - Single inheritance (interfaces mimic multiple inheritance)
    - Networking, Threading, Windowing ....
  - Robust
    - Java compiler and run time environment do extensive checking
  - Portable
    - "Write once, run anywhere"
- A single API across all platforms

What Is Java (Cont.)?

- An execution environment (Java Virtual Machine)
  - Software microprocessor with its own instruction set and op-codes (byte codes)
  - JVM interprets byte codes produced by the Java compiler
    - Architecture independent
    - Dynamically linked
  - JVM performs run time checking
  - Just-in-time (JIT) compilation may improve efficiency
- Java has one Application Binary Interface (ABI) on all platforms
  - Consistent program environment across all platforms

Java System Architecture



The Java Developer's Kit

- Packages (class libraries)
  - Common to every implementation
  - Java source included
    - java.lang, java.util, java.math, java.text [strings, numbers, date/time...]
    - java.io, java.net [file and network I/O]
    - java.awt [abstract window toolkit], java.applet [animation, audio]
    - java.security [public keys, cryptography]
    - java.sql [database] java.rmi [remote methods]
    - java.beans [library of pluggable components]
- Programs

javac	Compiles Java source into bytecodes
java	Invokes the JVM to run a compiled application
appletviewer	Previews a compiled applet
javadoc	Extracts interface documentation from source
javah	Generates C header files for native methods
javap	Disassembles Java class files
jdb	Runs the Java debugger
jar	Create Java archive files

- Samples and demos to illustrate usage

Java on VM/ESA

What Is Currently Being Provided

- Port of JDK 1.1.6
  - The IBM Java Port for VM/ESA, Developer Release 1.1.6
  - Installed into BFS
  - Installed using VMSES/E, or manually
- For the first time includes a Just-in-Time Compiler (JIT) for VM/ESA
- Integrate Remote AWT (RAWT)
  - Host code preinstalled
  - Obtain workstation code from http://www.s390.ibm.com/java/rawt.html
  - View RAWT -readme.html from the java root directory
- Euro support, IEEE floating point
- IBM Java Port for VM/ESA, Developer Release 1.1.6 is ONLY available from our web page: http://www.s390.ibm.com/vm/java

What Is Currently Being Provided (cont.)

- NetRexx 1.160
  - Major improvements to inner classes and select instruction
  - Many code generation improvements
  - Some new compiler options
- CMS Multitasking and BFS performance enhancements (V2R4)
  - Extend kernel DSA frame size to prevent frame overflows (6->64kb)
  - Eliminate large number of ThreadCreates
  - Eliminate redundant BFS QueueOpen calls
  - Eliminate large number of calls to CMSSTOR

Highlights of JDK 1.1.6

- Includes:
  - Compiler
  - Debugger
  - Java Virtual Machine (JVM)
  - JDK 1.1.6 class files
  - Java Native Interface (JNI)
  - JDK portion of the JDBC
    - DB/2 on VM does not have a JDBC interface
  - Other utilities
    - Appletviewer not supported on VM/ESA

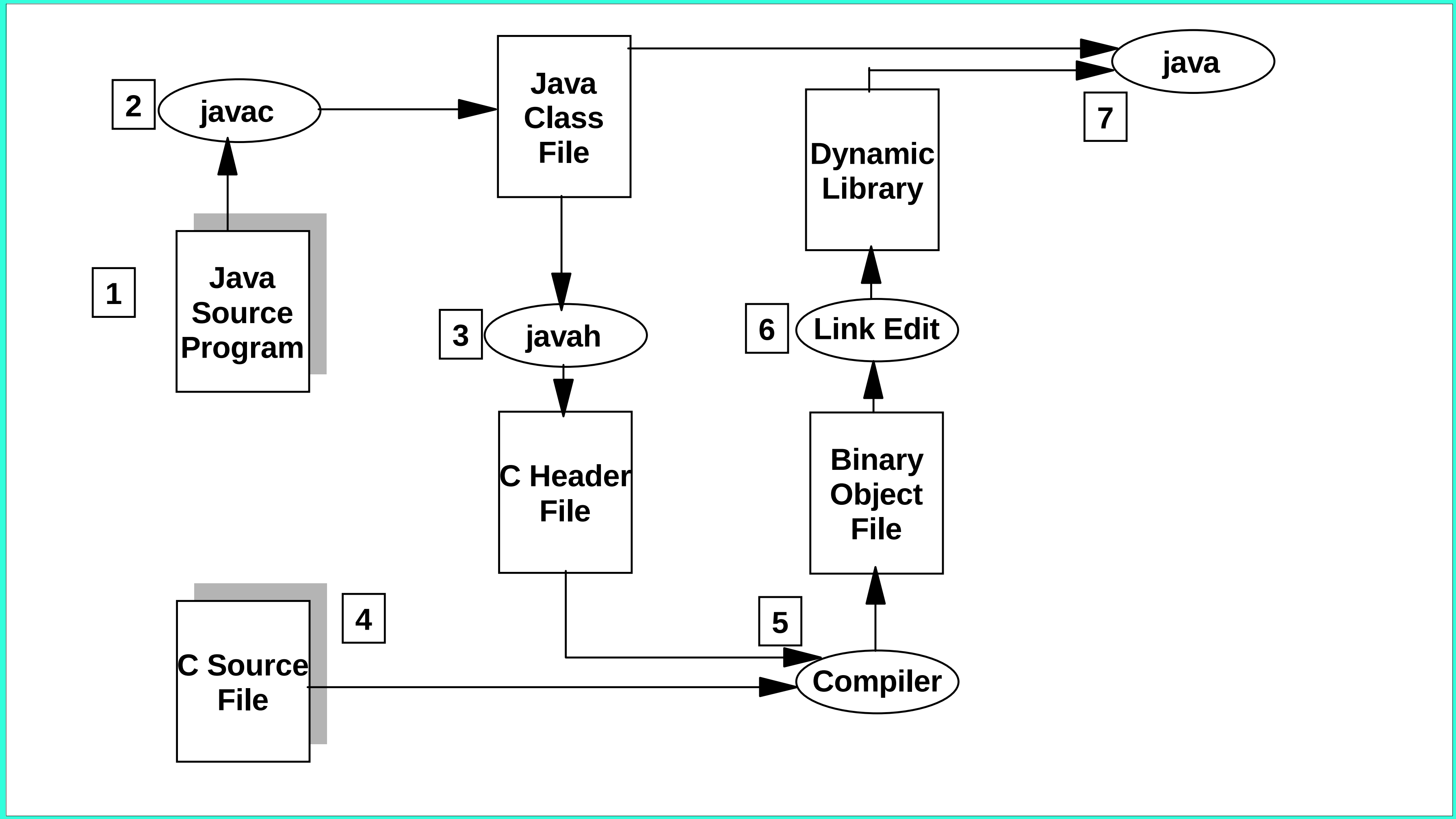
Java Native Interface

- An interface enabling code running inside a JVM to interoperate with applications and libraries written in other languages
- Supports need to get things done outside a JVM
- Imposes no restrictions on implementation of underlying Java VM
  - Application writer can write one version of native application and expect it to work with all Java VMs supporting JNI
- When is 100% pure Java not enough?
  - Need to use a platform specific feature not supported by Java class library
  - Desire to make existing application accessible to Java code
  - Need to implement time critical code in low-level language

Java Native Interface (cont.)

- With the JNI you can use native methods to:
  - Create, inspect, and update Java objects (includes arrays and strings)
  - Call Java methods
  - Catch and throw exceptions
  - Load classes and obtain class information
  - Perform runtime type checking
- Reason for including in JDK is to avoid problems associated with vendors offering specific native method interfaces in their Java VM implementations

Java Native Interface (cont.)



Java Data Base Connectivity

- Java API for executing SQL statements
  - Classes and interfaces written in Java
  - Facilitate writing data base applications in pure Java
- Supports sending SQL statements to any database
  - SYBASE, ORACLE, INFORMIX, DB/2 Universal Database
  - Write once run anywhere
- With JDBC your program can
  - Establish a connection with a database
  - Send SQL statements
  - Process the results
- Benefit
  - Bring company data to end users on diverse platforms

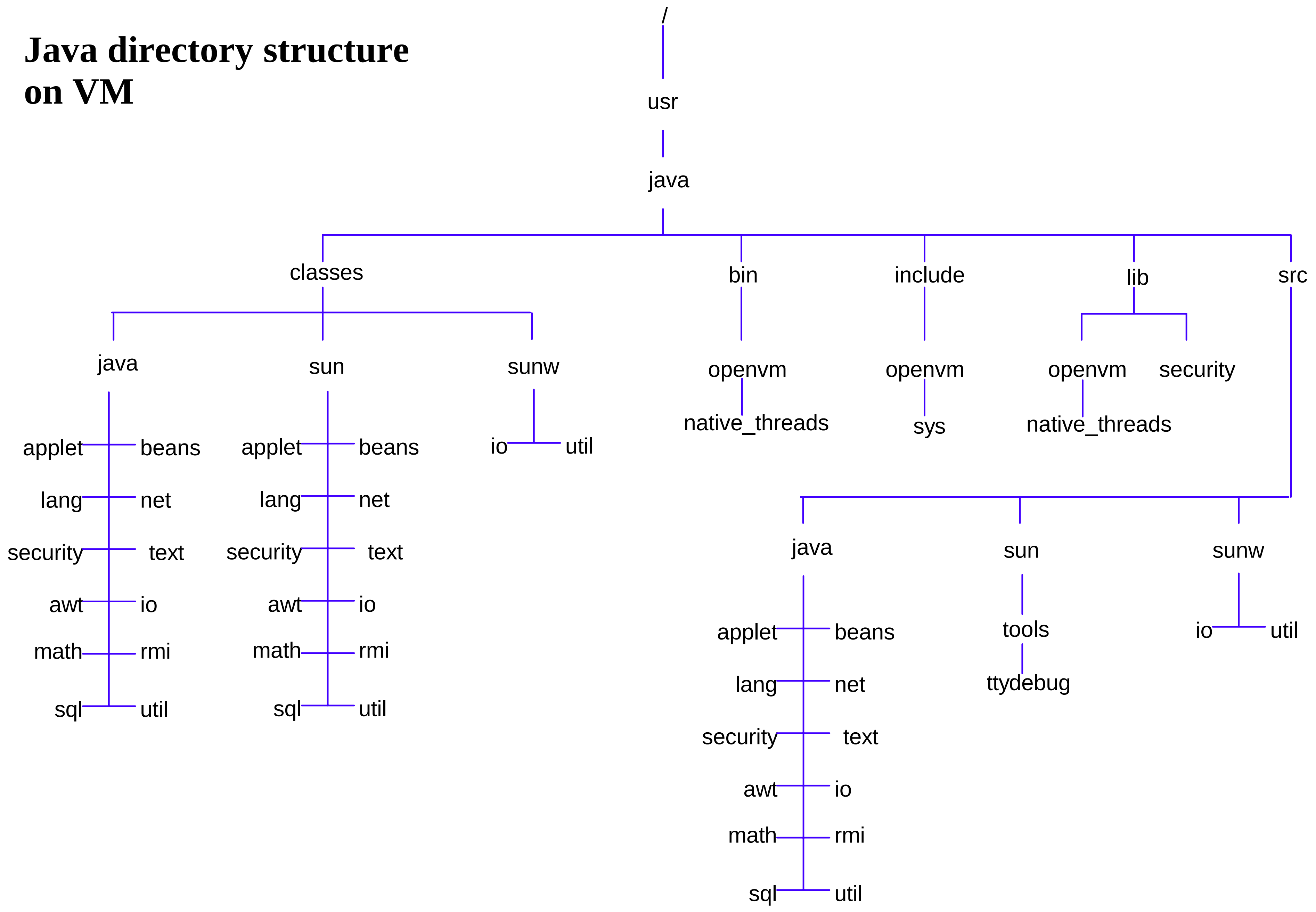
Java Data Base Connectivity (cont.)

- JDBC is an API that is uniform and database independent
- Two parts:
  - Application layer
    - Standard API used by Java class files
    - Layer used by application developers
  - Driver layer
    - Translate JDBC calls into specific calls required by a particular database
    - Each supported database must have a driver
    - An application is written once and moved to systems with various drivers
    - Application remains constant, drivers are the part that change
    - Allows for implementation of database specific functionality
- The application layer comes with the JDK
- From VM, use solution with applet and DB/2 Connect on a workstation platform (Win95/98, NT, Linux...)

Just-In-Time Compiler

- Code generator that runs concurrently with JVM
- Determine methods called most often and generate machine code on the fly
- Makes use of specific hardware or coprocessors
- May get as much as 2 to 3 times performance improvement

Java directory structure on VM



Software Prerequisites

- At least VM/ESA V2R3
- OpenEdition Shell and Utilities (feature 6059)
  - Not required for installation
  - Still recommended for execution environment
- LE/370 1.8 (included in base VM/ESA V2R3 and V2R4)
- SFS filepools installed (VMSYS:, VMSYSU:)
- BFS
- VMARC MODULE (depending how Java code is ordered for VM/ESA V2R3 files)
- **Refer to our VM/ESA java website for installation instructions**

Verifing Installation

- Logon to userid with posixinfo
- Mount bfs root
  - mount /../VMBFS:VMSYS:ROOT/ /
- Enter shell
  - openvm shell
- Verify java version level
  - java -version
- Response
  - java version "1.1.6"
- cd /usr/java
  - java hw
  - Response should be Hello World

Using JDK Utility Programs

- Java compiler
  - javac [options] filename.java ...
  - Options
    - -classpath path - overrides default or classpath environment variable
    - -d directory - specifies root directory of class hierarchy, destination directory for compiled classes (for writing)
    - Other options
  - Can compile more than one class at a time
- Java interpreter
  - java filename~~.class~~
  - Executes java bytecodes

Using JDK Utility Programs (Cont.)

- Java Debugger
  - Compile program with -g compiler option which generates debugging tables
  - Run program with jdb command
    - jdb fn
  - Debugger loads class, and displays prompt to enter command
    - List thread information
    - Set break points
    - Step through a line at a time
    - Break at specified exceptions
    - Many more commands
- Run process in background then <KILL -SIGINT pid> to obtain thread dump
  - java <classname> & (to run in background)

Using JDK Utility Programs (Cont.)

- Java Documentation Generator
  - javadoc command
  - Parses declarations and comments
  - Produces html pages describing classes, interfaces, constructors, methods, fields
  - Produces class hierarchy and index of all members
  - javadoc [options] fn ... (where fn is list of source files or package names)
    - -d path - specifies target directory for output files
    - -classpath path - specifies where to look for source files
    - -sourcepath path - synonym for -classpath
    - Other options

Using JDK Utility Programs (Cont.)

- Java Archive Tool (jar)
  - Combines many class files and other resources into single specific format file
  - Can be compressed using ZIP compression format
  - Useful for applets consisting of multiple classes with other resource files such as image or sound files
    - Reduces download time by allowing browser to obtain all files with one request
  - jar [options] infile ...
  - Options
    - c - creates a new or empty archive and adds files to it
    - f - specifies JAR file name as second command line argument
    - o - stores without ZIP compression

Sample Application

```
openvm shell

IBM
Licensed Material - Property of IBM
5654-030 (C) Copyright IBM Corp. 1995
(C) Copyright Mortice Kern Systems, Inc., 1985, 1993.
(C) Copyright Software Development Group, University of Waterloo, 1989.

All Rights Reserved.

U.S. Government users - RESTRICTED RIGHTS - Use, Duplication, or
Disclosure restricted by GSA-ADP schedule contract with IBM Corp.

IBM is a registered trademark of the IBM Corp.


#



cms 'x SAMPLServer.java (namet bfs'
```

SAMPLServer.java

```
import java.net.*;
import java.io.*;
import java.util.*;
import java.text.*;

// This class is defined to implement the Runnable interface, so
// we can define and use multiple threads of execution.  We could
// have constructed the program such that each new connection would
// cause a new thread to be defined.  However, this could potentially
// cause serious performance problems as our server would be subject
// to memory shortages, and other undesireable effects when the
// number of concurrent threads became large.  To avoid these problems
// we will only define 3 threads.  We could have defined more
// without any serious side effects, but 3 will suffice for the demo.
// Implementing threads in this manner also keeps the logic very
// simple for those who are just beginning to understand java threads.

public class SAMPLServer implements Runnable {
  private ServerSocket ss;           // Define a server socket
                                     // accessible by all threads

  public static void main(String args[]) throws Exception {
    SAMPLServer w = new SAMPLServer();
    w.go();                          // Define instance of this class
  }
```

SAMPLServer.java

```
// Since SAMPLServer implements Runnable, we can pass it as an
// instance to the constructors of the Thread objects.  We pass the
// same instance to all Thread objects, so that they can share code
// and data (specifically, the serversocket object.

  public void go() throws Exception {
    ss = new ServerSocket(5050, 5);
    Thread t1 = new Thread(this, "1"); // Thread named 1
    Thread t2 = new Thread(this, "2"); // Thread named 2
    Thread t3 = new Thread(this, "3"); // Thread named 3
    t1.start();                        // Start each of the threads
    t2.start();
    t3.start();
  }

// Each thread that is started will execute the instance method run()
// When we defined this class as one that implements the Runnable
// interface, one consequence of that is that we had to define an
// instance method named run().  This is the code each thread will
// execute.
```

SAMPLServer.java

```
  public void run() {
    Socket s = null;
    BufferedWriter out = null;
    String myname = Thread.currentThread().getName();
    String enc = "8859_1";

    for(;;) {
      try {
        System.out.println("Thread " + myname + " about to accept..");
        s = ss.accept();
        System.out.println("Thread " + myname + " accepted a connection");
        out = new BufferedWriter(
               new OutputStreamWriter(s.getOutputStream(), enc));       ;
        Date dt = new Date();
        DateFormat df = DateFormat.getDateTimeInstance(DateFormat.FULL,
                        DateFormat.DEFAULT);
        out.write("Thread " + myname + ": " + df.format(dt));
        Thread.sleep(10000); // we put a sleep here to force the
                             // threads to run somewhat in sequence
                             // and cycle through the set
        out.write("\n");
        out.close();
      }
      catch (Exception e) {
        e.printStackTrace();
      }
    }
  }
}
```

SAMPLServer.nrx

```
/******/
import java.net.
import java.io.
import java.util.
import java.text.

class SAMPLServer implements Runnable
  properties private
    ss = ServerSocket

  method main(argwords=String[]) static
    w = SAMPLServer SAMPLServer()
    w.go()

  method go() public signals IOException
    ss = ServerSocket ServerSocket(5050, 5)
    t1 = Thread Thread(this, "1")
    t2 = Thread Thread(this, "2")
    t3 = Thread Thread(this, "3")
    t1.start()
    t2.start()
    t3.start()
```

SAMPLServer.nrx

```
method run() public
  s = Socket
  out = BufferedWriter
  myname = String Thread.currentThread().getName()
  enc = String "8859_1"

  Loop Forever
    Do
        Say "Thread " myname " about to accept.."
        s = ss.accept()
        Say "Thread " myname " accepted a connection"
        out = BufferedWriter( -
               OutputStreamWriter(s.getOutputStream(), enc))
        dt = Date Date()
        df = DateFormat DateFormat.getDateTimeInstance(DateFormat.FULL, -
                       DateFormat.DEFAULT)
        disp = String "Thread " myname ": " df.format(dt) "\n"
        out.write(disp, 0, disp.length())
        Thread.sleep(10000)

        out.close()
      catch e = Exception
        e.printStackTrace()
    End
  End
```

```
#
nrc SAMPLServer.nrx
NetRexx portable processor, version 1.142
Copyright (c) IBM Corporation, 1998.  All rights reserved.
Program SAMPLServer.nrx
  === class SAMPLServer ===
    function main(String[])
      signals IOException
    method go
    method run
      implements Runnable.run
Compilation of 'SAMPLServer.nrx' successful
#
java SAMPLServer
Thread  1  about to accept..
Thread  3  about to accept..
Thread  2  about to accept..
Thread  1  accepted a connection
Thread  1  about to accept..
```

```
 pipe tcpclient 9.82.2.222 5050 linger 25 | xlate from codepage 819 to codepage
  1047 | console
Thread  1 :  Monday, February 15, 1999 4:29:49 PM
```

Summary

Summary

- Java started life in consumer electronic market
- Explosive growth with WWW
- Suited to characteristics of web
  - Portable
    - Single API and ABI
    - Secure
    - Most browsers have built-in Java interpreter
- Object-oriented language
  - Everything is in a class
  - Core set of classes
    - Language extended through additional libraries
  - C-like syntax

Summary

- Interpreted language
  - Java Virtual Machine
- JDK 1.1.6 implemented on VM/ESA V2R3 & V2R4
  - Installed in BFS (using VMSES/E)
    - Classes in directories, not zip file
  - AWT can be used on VM with RAWT classes
  - Exploitation of CMS facilities is attractive but subverts portability
    - Java Native Interface to be exploited to open this up
- Add VM to list of Java platforms