Java™ : A Quick Look

somchaip@chula.ac.th

Programming Languages

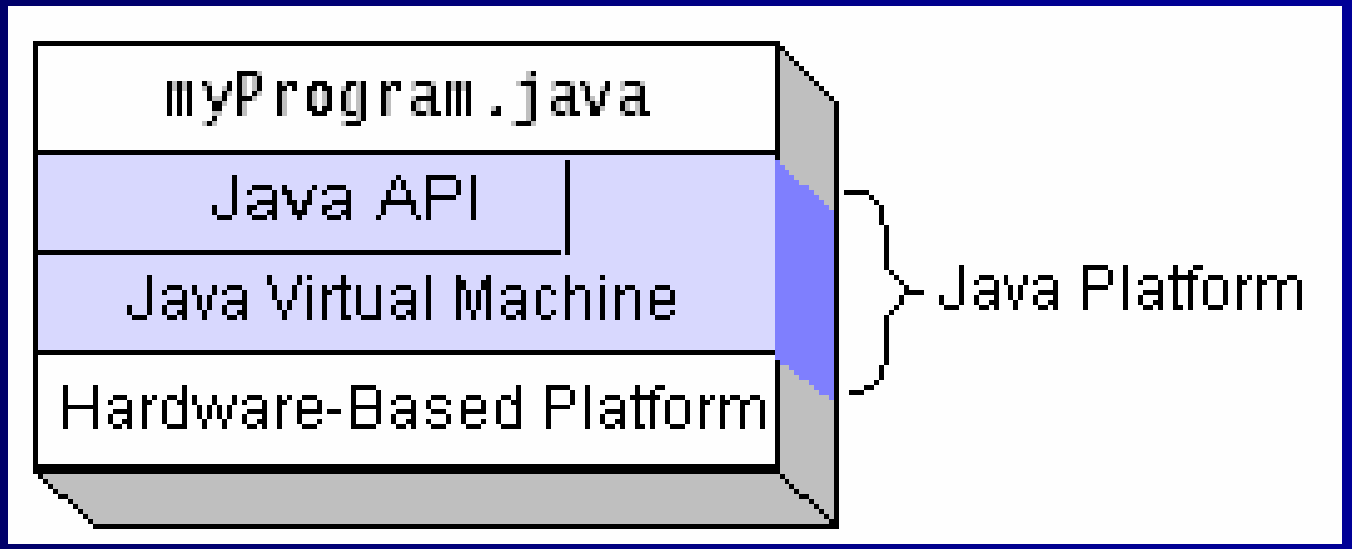
- 1957 : Fortran
- 1959 : Cobol，Lisp
- 1964 : Basic
- 1971 : Pascal
- 1972 : C，Prolog
- 1983 : C++
- 1995 : Java
- 2001 : C#，J#

Sun's Green Project

- 1990 : Sun's Green project
  - "The convergence of digitally controlled consumer devices and computers."
- 1992 : StarSeven (*7) handheld wireless PDA using Oak as prog. lang.
  - *7 went ch11
- 1995 : Oak was renamed to Java at the time when Internet & WWW was booming.

Java Technology

- Java Programming Language
- Java Platform
  - Java Virtual Machine
  - Java API



Java VM

- Solaris
- Linux
- Windows
- Mac OS X
- HP-UX
- IBM-AIX, OS/390
- PDA
- Java-enabled cell phones
- GameBoy Advance
- ...

Java API

	version	#packages	#classes
	1.0	8	212
	1.1	23	504
Java 2	1.2	59	1520
Java 2	1.3	76	~1800
Java 2	1.4	135	~2800
Java 2	1.5	???	????

Java Platforms

- Standard Edition (J2SE)
  - client-side general-purpose applications
- Enterprise Edition (J2EE)
  - multi-tier server-centric applications
- Micro Edition (J2ME)
  - consumer and embedded devices

Java Technology

- James Gosling :
  - “... software VLSI, end-to-end, side-to-side, homogenous view of heterogeneity reality ...”

Java Logos

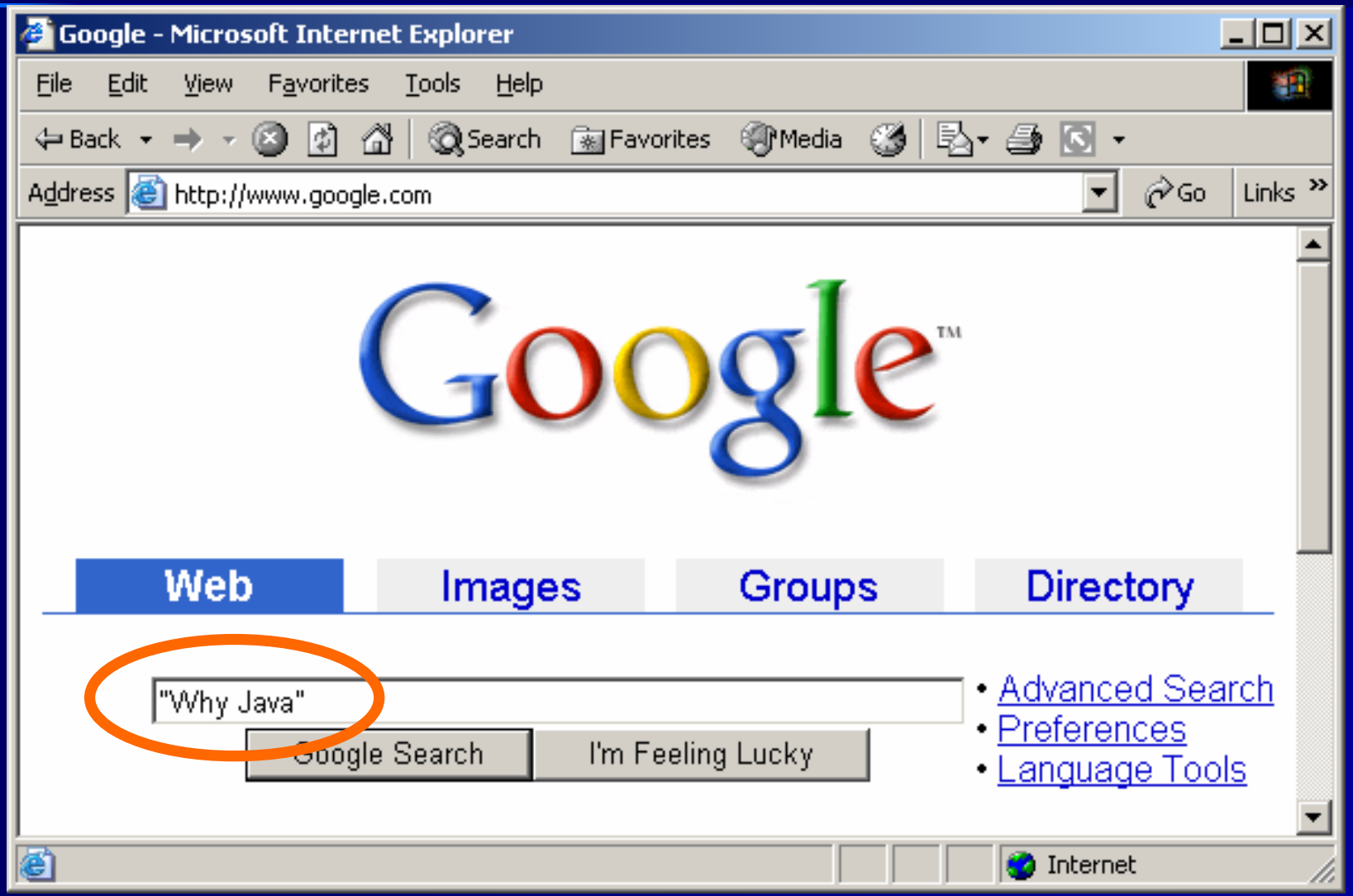
1995
-
2002

2003

Sun's Stat.

- 500 million desktop PCs run Java applets
- 3 million+ professional Java developers
- Of all Java developers (Evans Data, 2002):
  - 58% develop desktop applications
  - 50% develop intranet applets
  - 38% develop internet applets
- >36 million Java runtime env. downloads
- JRE downloads surging, now about 3 million per month

Why Java ?


Google - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.google.com
Go Links
Google
Web Images Groups Directory
"Why Java"
Google Search I'm Feeling Lucky
Advanced Search
Preferences
Language Tools
Internet

Popularity

Position	Programming Language	Ratings
1	Java	47.4
2	C	36.7
3	C++	33.8
4	Perl	18.3
5	(Visual) Basic	15.2
6	PHP	9.5
7	SQL	6.1
8	C#	4.1
9	Delphi/Pascal/Kylix	3.8
10	JavaScript	3.7

The Biggest Open Problem In Programming Languages

- Increasing Programmer Productivity
  - Write programs correctly
  - Write programs quickly
  - Write programs easily

http://research.microsoft.com/~toddpro/papers/disruptive.ppt

Java Programming Lang.

- James Gosling :
  - Life is too short, Java lets him do more coding, less debugging
  - Java beats C and C++ by a factor of 2 in developer productivity.

Java : Design Goals

Java technology must enable the development of secure, high performance, and highly robust applications on multiple platforms in heterogeneous, distributed networks.

Java Programming Lang.

- Simple, Familiar, Object-Oriented
- Robust, Secure
- Architecture Neutral, Portable
- High Performance
- Interpreted, Threaded, Dynamic

Platform independence is the main reason for using Java (ComputerWorld's survey)

Simple, Familiar, and OO

- easily program
- Bill Joys : Java = C++ - -
- provide a clean & efficient object-based development platform

Robust and Secure

- Robust
  - strict compile time checking
  - strict run-time checking
  - automatic garbage collection
  - support exception handling & assertion
- Secure
  - no pointer
  - verify bytecode before class loading
  - sandbox

Arch Neutral & Portable

- Write Once Run Anywhere (WORA)
  - compiled to Java bytecode
  - run on Java VM
  - fixed sizes, formats, and behaviors of all primitive data types
  - behave (almost) the same on multi-platforms
- solve distribution & version problems
- can be deployed into heterogeneous network environments

High Performance

- Hotspot VM is highly tunable
- the garbage collector runs as a low-priority thread
- support native codes for compute-intensive operations
- JVM performance keeps improving

Interpreted, Thread, Dynamic

- linking is lightweight and incremental
- faster development cycle
- support multithreading at the language level w/ sync. primitives
- language and run-time system are dynamically linked on demand

Java's WORA

- promote readability
- Java code will survive.
  - Write today, use next year.
  - Easy to maintain at a fixed level of functionality
- Java enables evolution
  - bad code dies
  - good code lives on (modified, evolves, becomes better)

http://www.wgrosso.com/Articles/Presentations/PresentationonJavaasaProg.html

WORA == Community

- Reuse lots of code from different places
  - Very little need to worry about platform dependencies
  - Widespread adoption of coding conventions
  - And large-scale libraries
- New programming style
  - Write Half, Steal the Rest

http://www.wgrosso.com/Articles/Presentations/PresentationonJavaasaProg.html

Language Features

- 1995 (1.0) : first public release
- 1997 (1.1) : nested classes
- 2001 (1.4) : assert
- 2003 (1.5) :
  - generics, enum, enhanced for, autoboxing/unboxing, varargs, static import, metadata.

increase expressiveness, increase safety

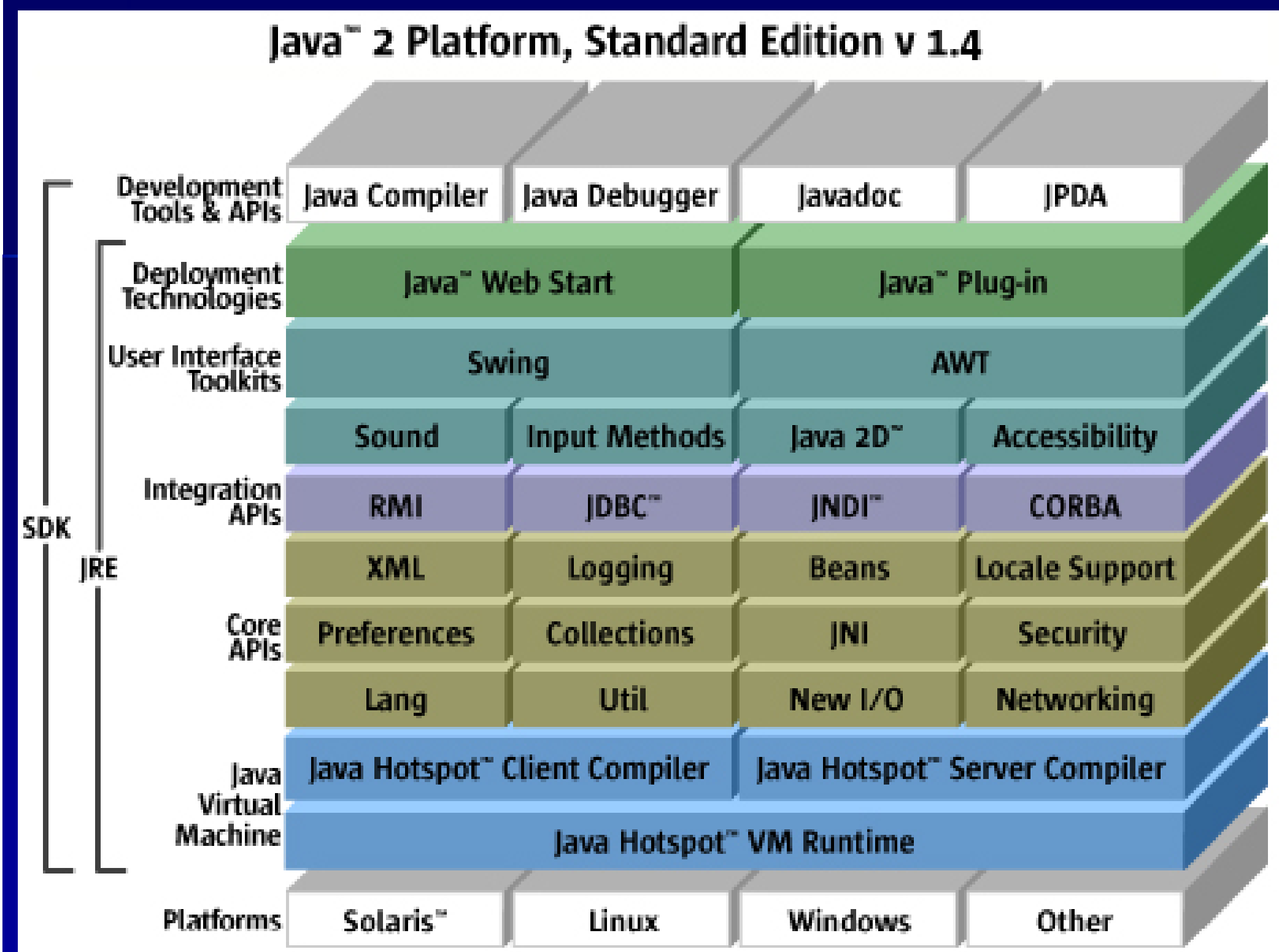
Features Removed from C++

- No typedef, define, preprocessor
- No multiple inheritance
- No operator overloading
- No automatic coercion
- No pointers

Demo

Object Technology in Java

- Encapsulation
- Polymorphism
- Single Inheritance
- Dynamic binding


Java™ 2 Platform, Standard Edition v 1.4
Development Tools & APIs
Java Compiler
Java Debugger
Javadoc
JPDA
Deployment Technologies
Java™ Web Start
Java™ Plug-in
User Interface Toolkits
Swing
AWT
Sound
Input Methods
Java 2D™
Accessibility
Integration APIs
RMI
JDBC™
JNDI™
CORBA
XML
Logging
Beans
Locale Support
Core APIs
Preferences
Collections
JNI
Security
Lang
Util
New I/O
Networking
Java Virtual Machine
Java Hotspot™ Client Compiler
Java Hotspot™ Server Compiler
Java Hotspot™ VM Runtime
Platforms
Solaris™
Linux
Windows
Other
SDK
JRE

javadoc

- generates HTML pages of API documentation from Java source files
- automatically adds cross-reference links
- produces one complete document each time it is run
- relies on the java compiler to do its job

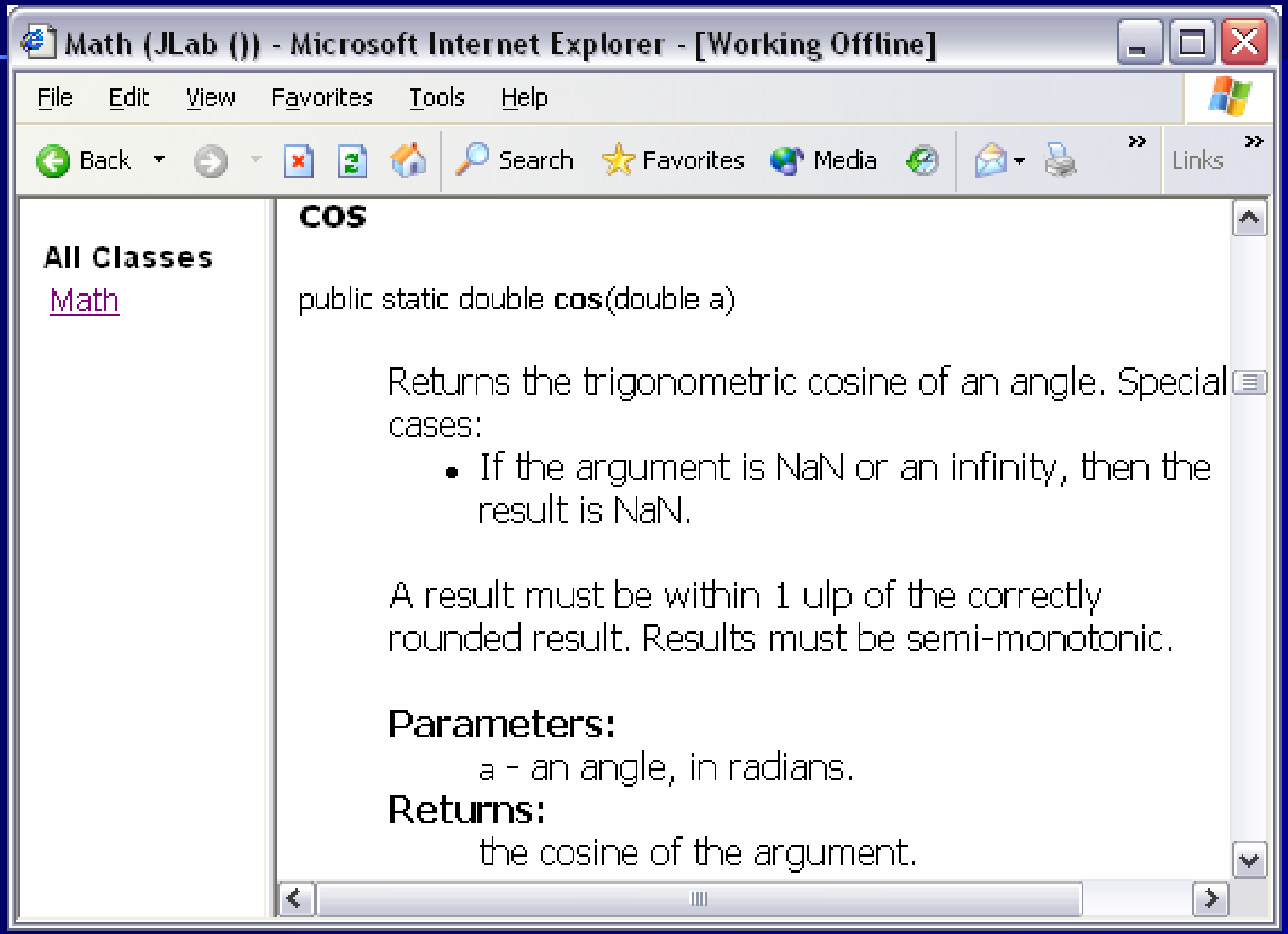
Commenting the Source

- javadoc comment /** */
- comments are written in HTML
- special tags embedded in comments
  - @author @deprecated
  - @param @throws
  - @return @since
  - @see ...

javadoc : Comments

```
/**
 * Returns the trigonometric cosine of an angle. Special cases:
 * <ul><li>If the argument is NaN or an infinity, then the
 * result is NaN.</ul>
 * <p>A result must be within 1 ulp of the correctly rounded
 * result.  Results must be semi-monotonic.
 *
 * @param   a   an angle, in radians.
 * @return  the cosine of the argument.
 */
public static double cos(double a) {
   return StrictMath.cos(a);
}
```

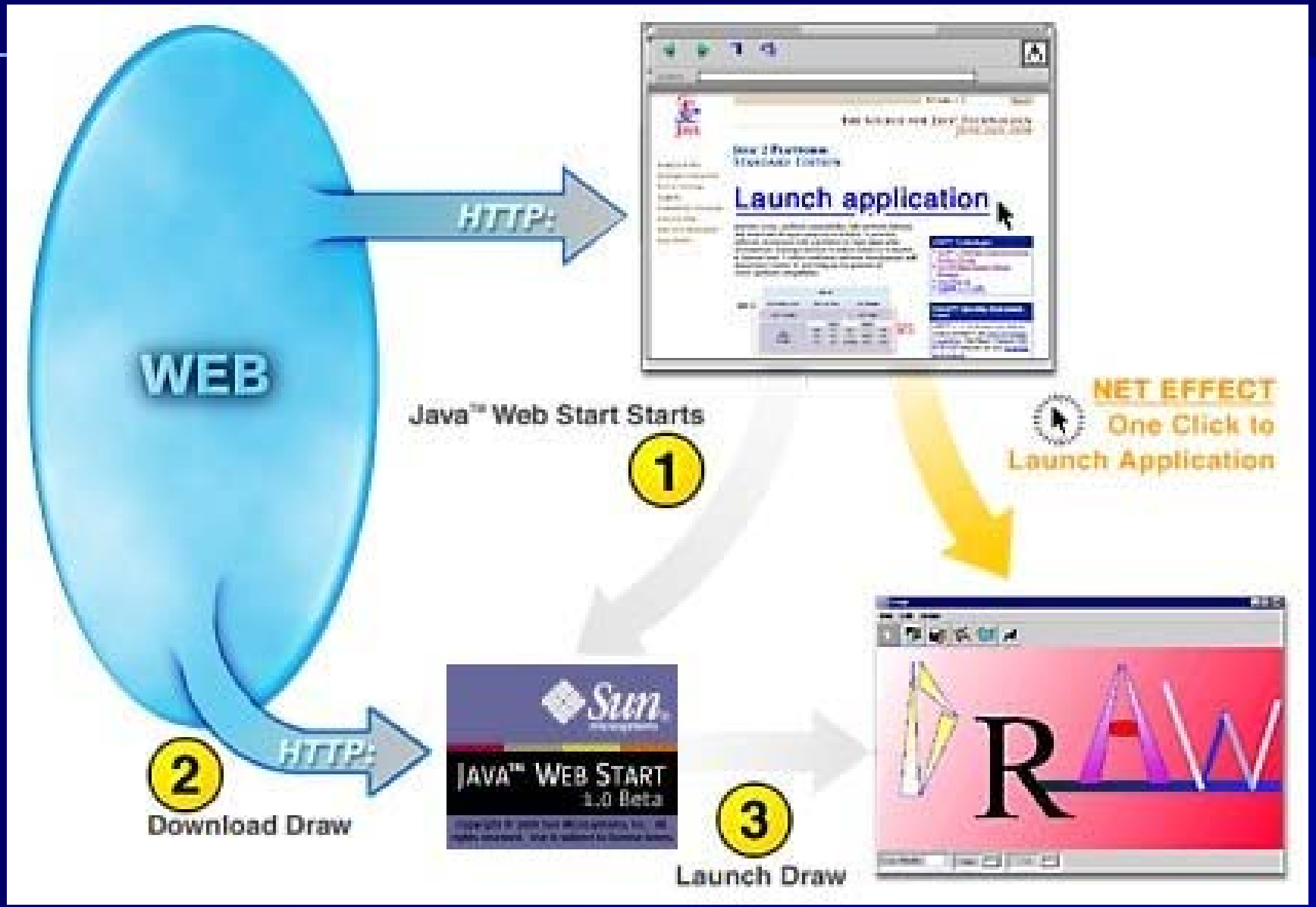
javadoc : HTML



javadoc

- can run on .java source files that are pure stub files with no method bodies
- can generate documentation for source files whose code is incomplete
- can customize the content and format of the Javadoc tool's output by using doclets (HTML, XML, MIF, RTF)
- use 3rd party s/w to produce other formats e.g., Windows HTMLHLP

Java Web Start



Java Plug-in

- enables Java 2 applets to be run under Sun's JRE in Netscape Navigator and Internet Explorer.
- supports cookies
- allow applet persistence across sessions
- applet caching
- Java-to-JavaScript communication
- Common DOM API

User Interface API

- AWT
- Swing
- Accessibility API
- Java 2D API
- Input Methods
- Sound

Abstract Window Toolkit

- rich set of UI components
- robust event-handling model
- graphics and imaging classes
- layout managers
- data transfer class (cut-&-paste through native clipboard)

Swing

- Graphical User Interface (GUI) component toolkit.
- Complete set of user-interface elements written entirely in Java.
- Integrated into the Java 2 platform (1.2 or higher).

Swing vs. AWT

- Swing
  - Lightweight
  - Complex
  - Pluggable L&F
  - Pure Java
  - Java 1.2, ...
- AWT
  - Heavyweight
  - Primitive
  - Single L&F
  - Not Pure Java
  - Java 1.0, ...

Swing vs. AWT

- JApplet
- JFrame
- JButton, JCheckbox, JRadioButton, JComboBox, JDialog, JLabel, JList, JSlider, JTextComponent, ...

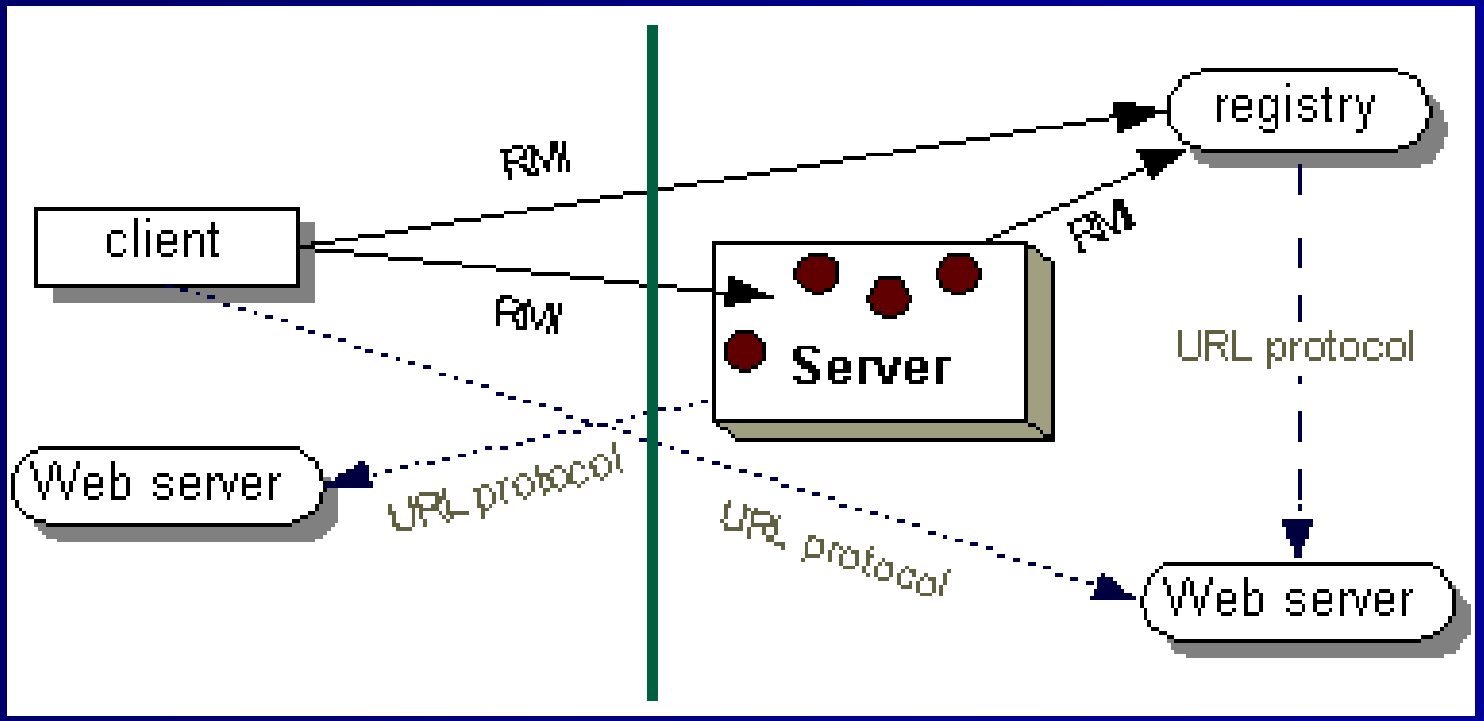
- Applet
- Frame
- Button, Checkbox, Choice, Dialog, Label, List, Scrollbar, TextComponent, ...

Distributed Object App.

- Server
  - create some remote objects
  - make references to the objects accessible
  - wait for client to invoke methods on these objects
- Client
  - get remote references
  - invoke methods on these references

Remote Method Invocation

- RMI allows an object running in one JVM to invoke methods on an object running in another JVM



Distributed App.

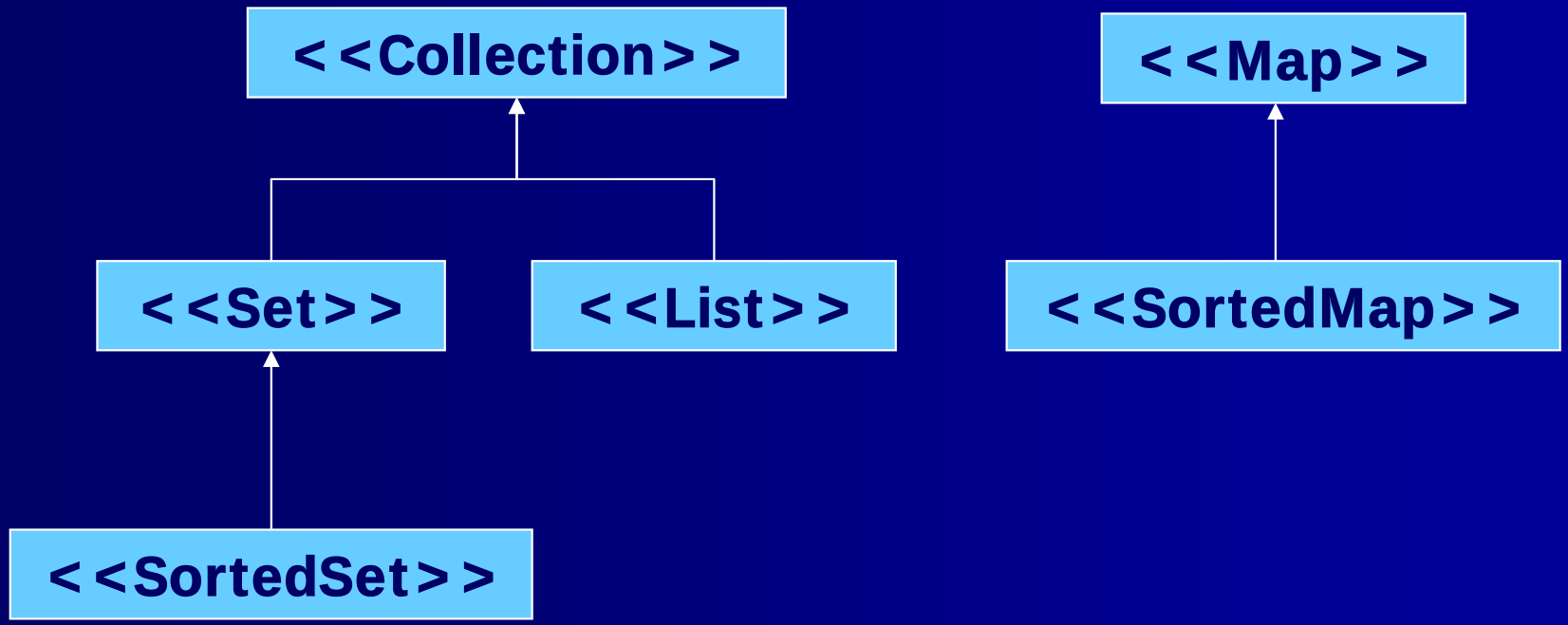
- Cross-language, cross-vendor interoperability is achieved via the IIOP (a transport protocol for distributed app. written in either IDL or Java RMI)
- Java 1.1 : RMI
  - a natural outgrowth of RPC
- Java 1.2 : Java IDL
  - API for interoperability and integration with CORBA
- Java 1.3 : RMI over IIOP
  - enabled remote objects written in Java to be accessible from any language via IIOP

Core API

- lang.
- util.
  - logging
  - perference
  - collection
  - regexp
  - jar, zip
- I/O & new I/O
- networking
- security
- locale
- JavaBean
- XML
- native interface

Collection Framework

- Java 1 : Vector, Hashtable
- Java 2 :

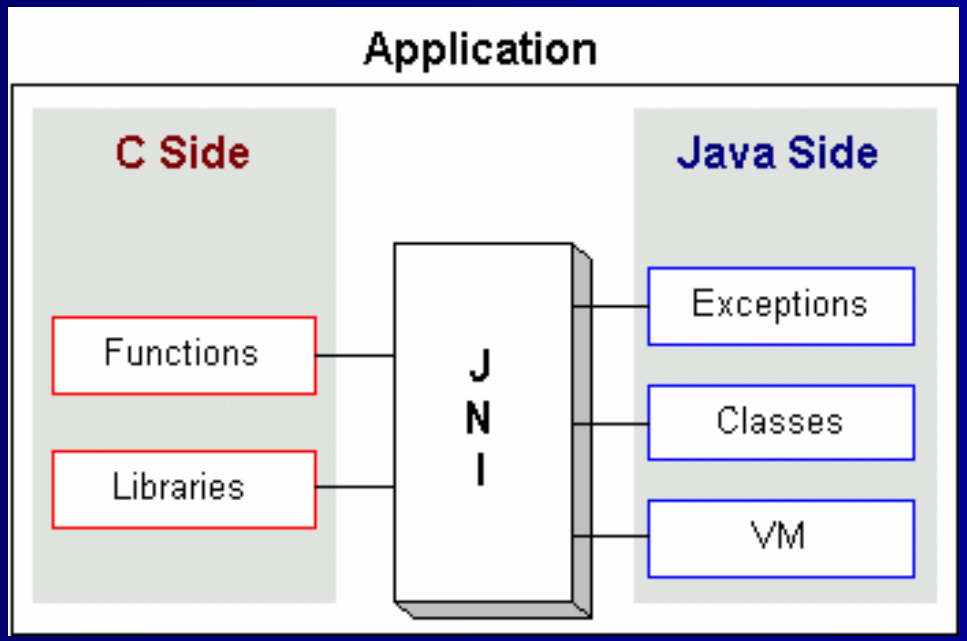


Implementations

- General-purpose
  - ArrayList, LinkedList
  - TreeSet, HashSet, LinkedHashSet,
  - TreeMap, HashMap, LinkedHashMap
- Wrapper
  - synchronized
  - unmodifiable
- Convenience
  - Array.asList
  - ...
- Legacy
  - Vector, Hashtable
- Special purpose
  - WeakHashMap, IdentityHashMap
- Algorithms
  - sort, search, fill, ...

JNI : Native Interface

- native method can
  - create, access, and update its own objects including objects passed to it
  - call Java methods
  - load and get information about Java classes



An Example

HelloWorld.java

```
class HelloWorld {
  public native void displayHelloWorld();

  static {
    System.loadLibrary("hello");
  }

  public static void main(String[] args) {
    new HelloWorld().displayHelloWorld();
  }
}
```

javac HelloWorld.java

javah –jni HelloWorld

An Example

HelloWorld.c

```
#include <jni.h>
#include "HelloWorld.h"
#include <stdio.h>

JNIEXPORT void JNICALL
Java_HelloWorld_displayHelloWorld(JNIEnv *env,
jobject obj)
{
    printf("Hello world!\n");
    return;
}
```

```
cl -Ic:\java\include -Ic:\java\include\win32
    -LD HelloWorld.c -Fehello.dll
```

Reflection API

- examine classes
- manipulate an instance of a classes whose name is not known until runtime
- invoke a method whose name is not known until runtime
- usually used for writing development tools e.g., debuggers, class browsers, ...

An Example : Java


```
public void init() {
  Button btOK = new Button("OK");
  btOK.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      System.out.println(“I’m OK”);
    }
  } );
  add(btOK);
  Button btCancel = new Button("CANCEL");
  btCancel.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      System.out.println(“I was cancelled”);
    }
  } );
  add(btCancel);
}
```

An Example : VB

```
Private Sub OK_Click()
  Debug.print "I'm OK"
End Sub

Private Sub CANCEL_Click()
  Debug.print "I was cancelled"
End Sub
```


```
 public void onButtonPushed_OK()
   System.out.println("I'm OK");
 }
 public void onButtonPushed_CANCEL() {
   System.out.println("I'm Cancelled");
 }
```

```
public void init() {
  ActionListener btListener = new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      methodInvoke("onButtonPushed_" +
                   e.getActionCommand());
    } };
  Button btOK = new Button("OK");
  btOK.addActionListener(btListener);
  add(btOK);
  Button btCancel = new Button("CANCEL");
  btCancel.addActionListener(btListener);
  add(btCancel);
}

public void methodInvoke(String metName) {
  try {
    Class c = this.getClass();
    Method met = c.getMethod(metName, new Class[0]);
    met.invoke(this, new Object[0]);
  } catch (Exception ex) { ex.printStackTrace(); }
}
```