



Java Programming Language Overview

Bok, Jong Soon

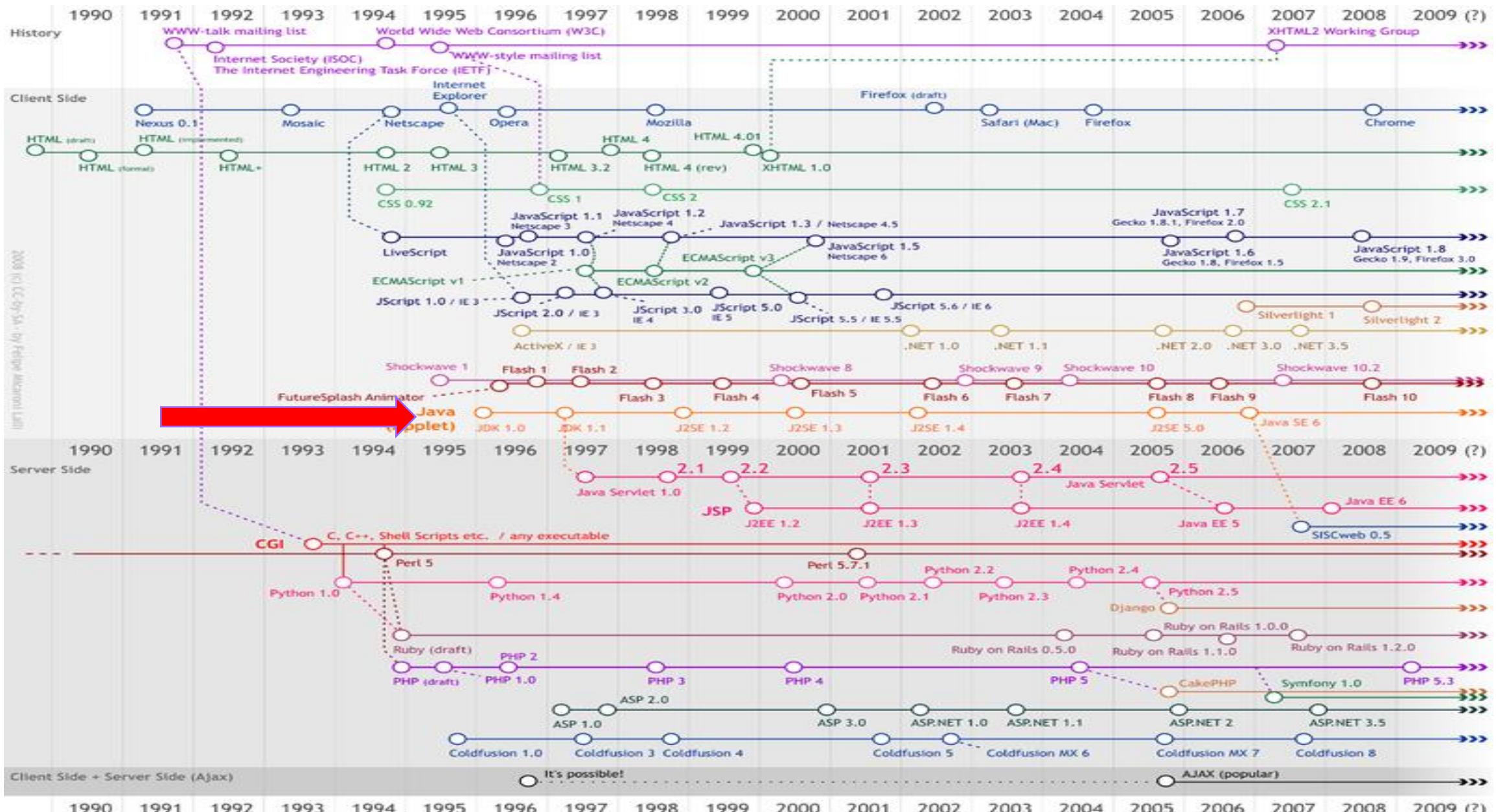
javaexpert@nate.com

<https://github.com/swacademy/JavaSE>

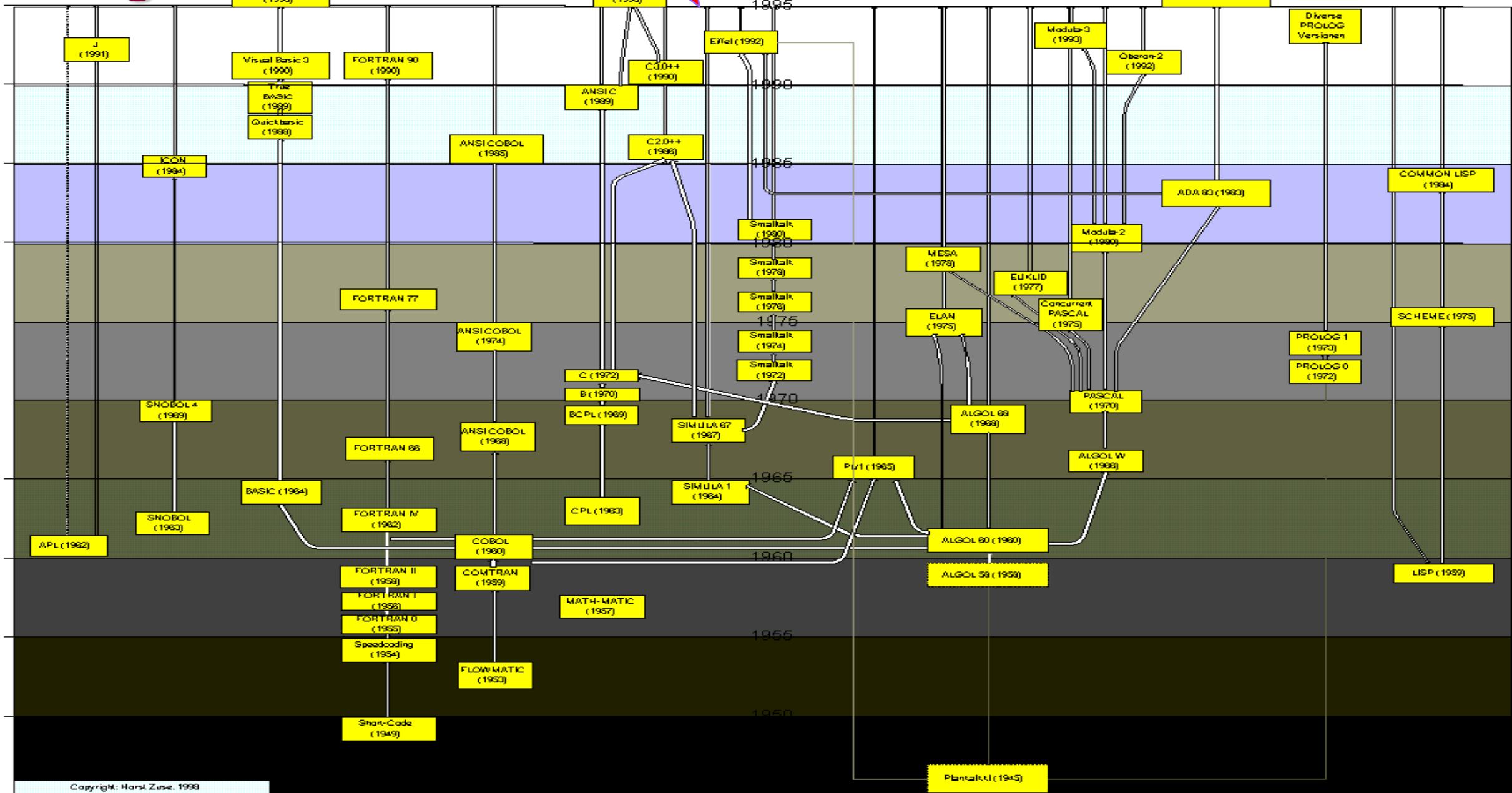
What is Java Technology?

- Is a programming language.
- Is a platform.

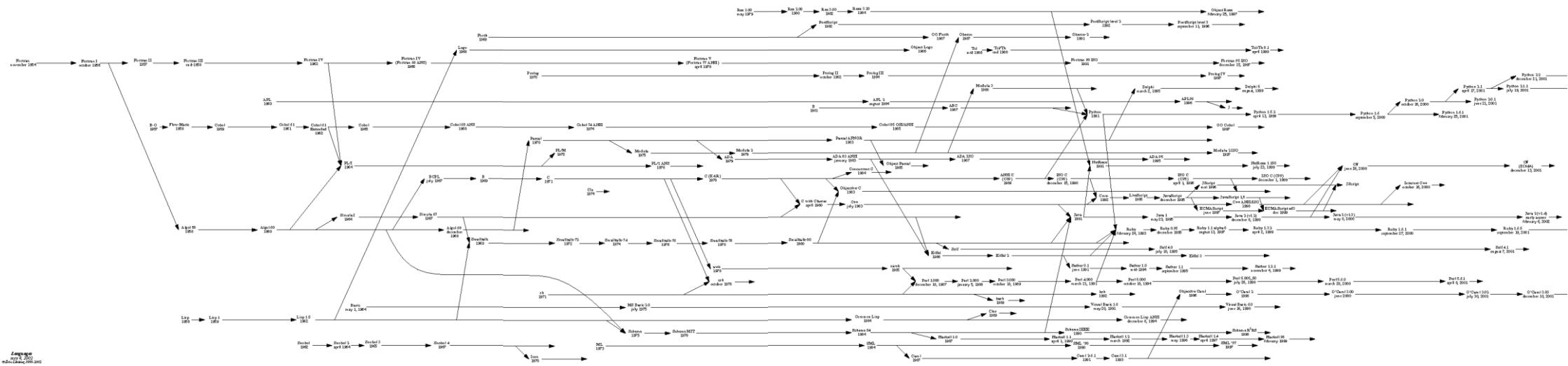
Web Timeline



Language Timeline



Language Timeline (Cont.)



Java Programming Language

- Is platform independent programming language.
- Similar to C++ in syntax.
- Similar to SmallTalk in mental paradigm.
- Is one of today's most popular software-development languages.
- Is used for Web programming
- Is used for developing standalone applications across platforms on servers, desktops, and mobile devices.
- Is a high-level language.

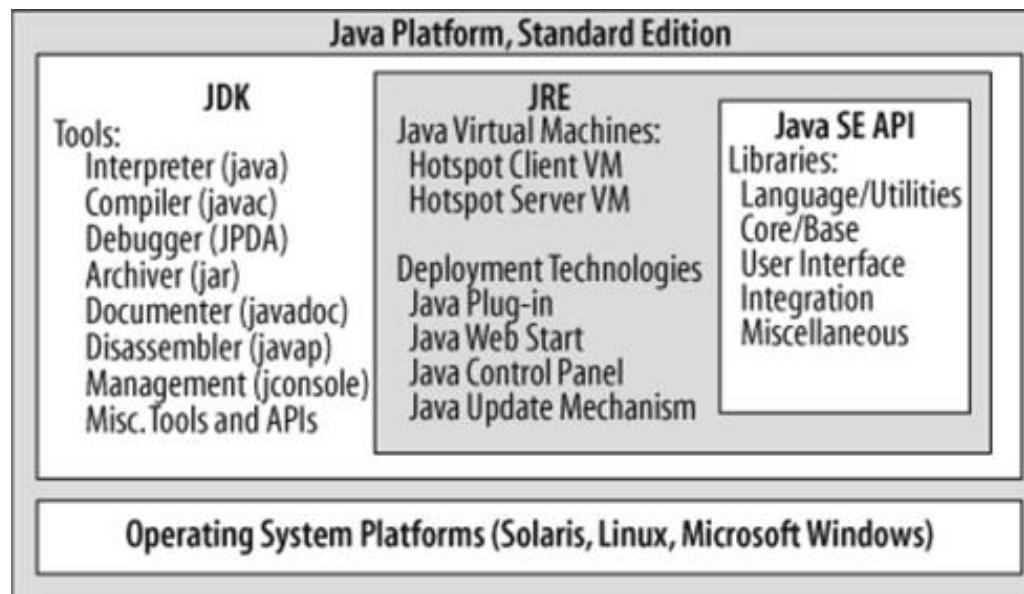
Java Programming Language (Cont.)

Jan 2016	Jan 2015	Change	Programming Language	Ratings	Change
1	2	▲	Java	21.465%	+5.94%
2	1	▼	C	16.036%	-0.67%
3	4	▲	C++	6.914%	+0.21%
4	5	▲	C#	4.707%	-0.34%
5	8	▲	Python	3.854%	+1.24%
6	6		PHP	2.706%	-1.08%
7	16	▲	Visual Basic .NET	2.582%	+1.51%
8	7	▼	JavaScript	2.565%	-0.71%
9	14	▲	Assembly language	2.095%	+0.92%
10	15	▲	Ruby	2.047%	+0.92%
11	9	▼	Perl	1.841%	-0.42%
12	20	▲	Delphi/Object Pascal	1.786%	+0.95%
13	17	▲	Visual Basic	1.684%	+0.61%
14	25	▲	Swift	1.363%	+0.62%
15	11	▼	MATLAB	1.228%	-0.16%
16	30	▲	Pascal	1.194%	+0.52%
17	82	▲	Groovy	1.182%	+1.07%
18	3	▼	Objective-C	1.074%	-5.88%
19	18	▼	R	1.054%	+0.01%
20	10	▼	PL/SQL	1.016%	-1.00%

Source : <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>

The Java Platform

- Platform : The hardware or software environment in which a program runs.
- Has two components:
 - The Java Virtual Machine
 - The Java Application Programming Interface (API)



From : [Java Pocket Guide], Robert
Liguori ; Patricia Liguori, O'Reilly, 2008,
978-0-59-651419-8, p191

The Java Platform (Cont.) - JRE

- Java Runtime Environment
- Provides the backbone for running Java application.
- Is a collection of software.
- Allows a computer system to run a Java application.
- Consists of
 - JVMs, Java Virtual Machines, interpret Java *bytecode* into machine code.
 - Standard class libraries
 - User interface toolkits
 - A variety of utilities.

The Java Platform (Cont.) - JDK

- Java Development Kit
- Provides all of the components and necessary resources to develop Java applications.
- Is a programming environment for compiling, debugging, and running Java applets, applications, and Java Beans.
- Includes the JRE, Java Programming language, development tools and tool APIs.
- Refer to <http://java-virtual-machine.net/other.html>

The Java Platform (Cont.) - JDK

- Download the most recent version at
<http://java.sun.com>
- Download older versions at
<http://java.sun.com/products/archive>

Java Development Kits	Codename	Release
Java SE 8 with JDK 1.8.0	Spider	2014
Java SE 7 with JDK 1.7.0	Dolphin	2011
Java SE 6 with JDK 1.6.0	Mustang	2006
Java 2 SE 5.0 with JDK 1.5.0	Tiger	2004
Java 2 SE with SDK 1.4.0	Merlin	2002
Java 2 SE with SDK 1.3	Kestrel	2000
Java 2 with SDK 1.2	Playground	1998

Table 1-1. Java Development Kits

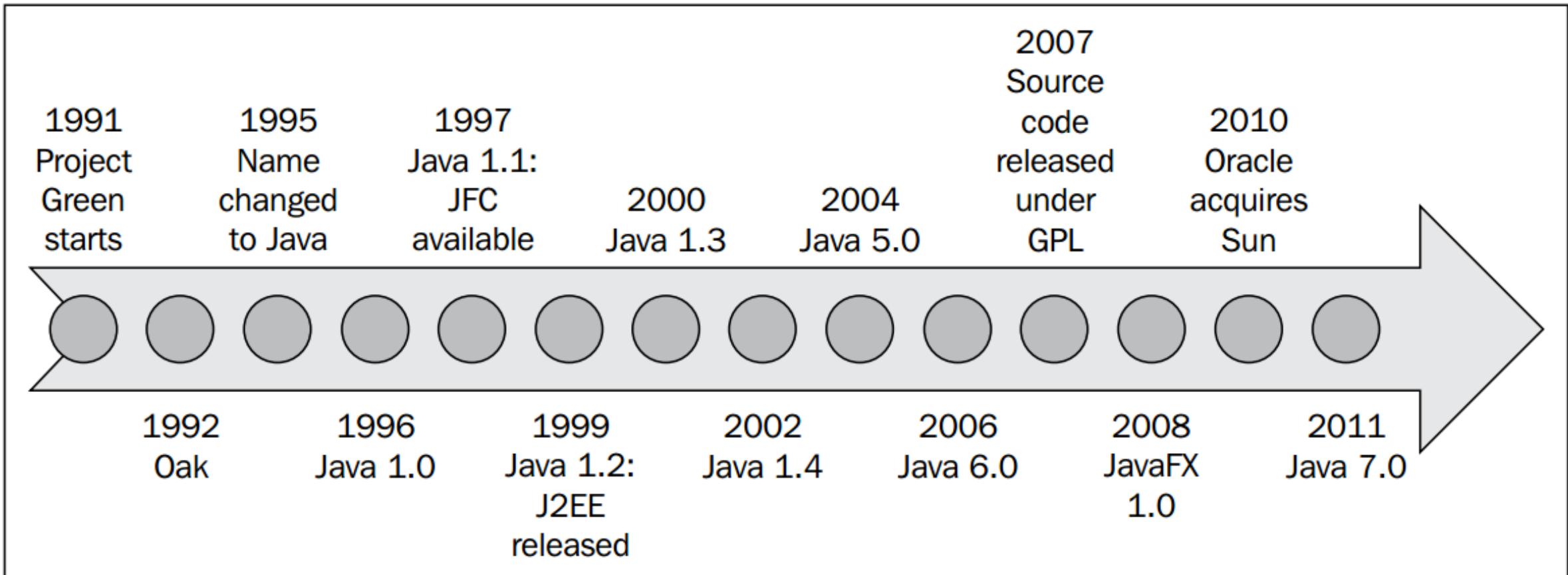
Java SE Code Names

Version	Code Name	Release Date
JDK 1.1.4	Sparkler	1997. 10. 11
JDK 1.1.5	Pumpkin	1997. 11. 03
JDK 1.1.6	Abigail	1998. 04. 24
JDK 1.1.7	Brutus	1998. 09. 28
JDK 1.1.8	Chelsea	1999. 04. 08
J2SE 1.2	Playground	1998. 11. 04
J2SE 1.2.1	(none)	1999. 03. 30
J2SE 1.2.2	Cricket	1999. 07. 08
J2SE 1.3	Kestrel	2000. 08. 05
J2SE 1.3.1	Ladybird	2001. 05. 17
J2SE 1.4.0	Merlin	2002. 02. 13
J2SE 1.4.1	Hopper	2002. 09. 16
J2SE 1.4.2	Mantis	2003. 06. 26
Java SE 5.0(1.5.0)	Tiger	2004. 09. 29
Java SE 6.0(1.6.0)	Mustang	2005. 11. 20
Java SE 7.0(1.7.0)	Dolphin	2011. 07. 28
Java SE 8.0(1.8.0)	Spider	2014. 03. 18

History

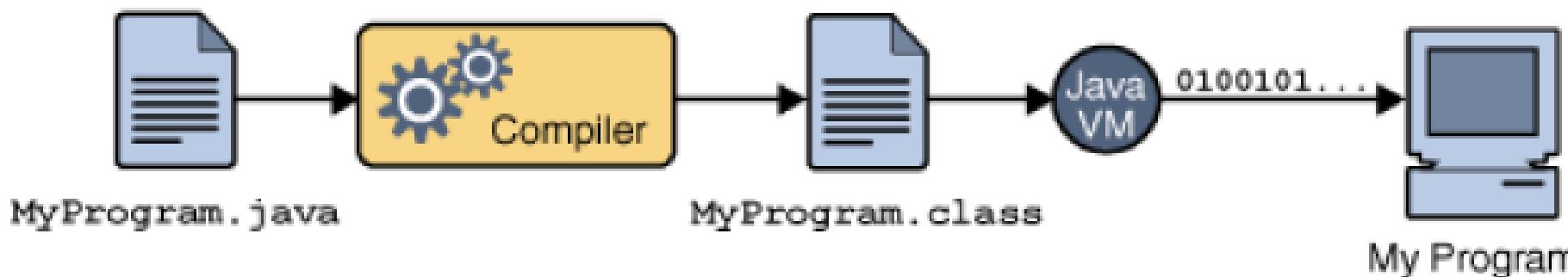
- Originally named *Oak*, designed in 1991.
- Main team members : Bill Joy, Patrick Naughton, Mike Sheridan, James Gosling.
- Original goal : use in embedded consumer electronic appliances.
- In 1994, team realized Oak was perfect for *Internet*.
- In 1995, renamed Java, was redesigned for developing Internet applications.
- Announced *in May 23 in 1995* at SunWorld'95.
- First non-beta release January 23 in 1996.
- Refer to <http://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html>

History (Cont.)



Features

- Simple
- Object-Oriented
- Distributed
- Multithreaded
- Dynamic
- Write Once, Run Anywhere™
- <http://java.sun.com/docs/white/langenv/>
- Architecture neutral
- Portable
- High performance
- Robust
- Secure



How it works...!

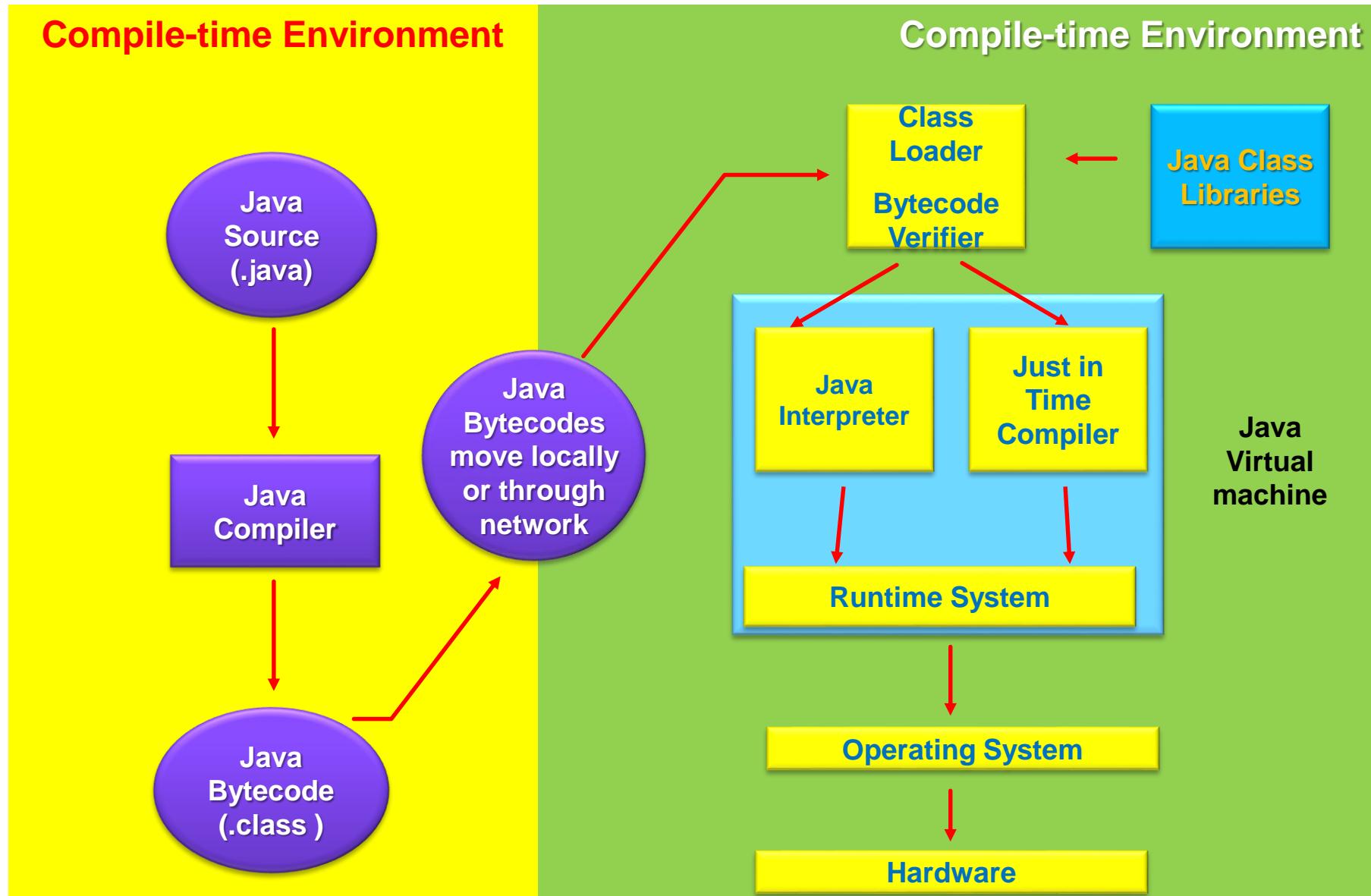


Figure 1.1 J2SE vs. J2EE vs. J2ME

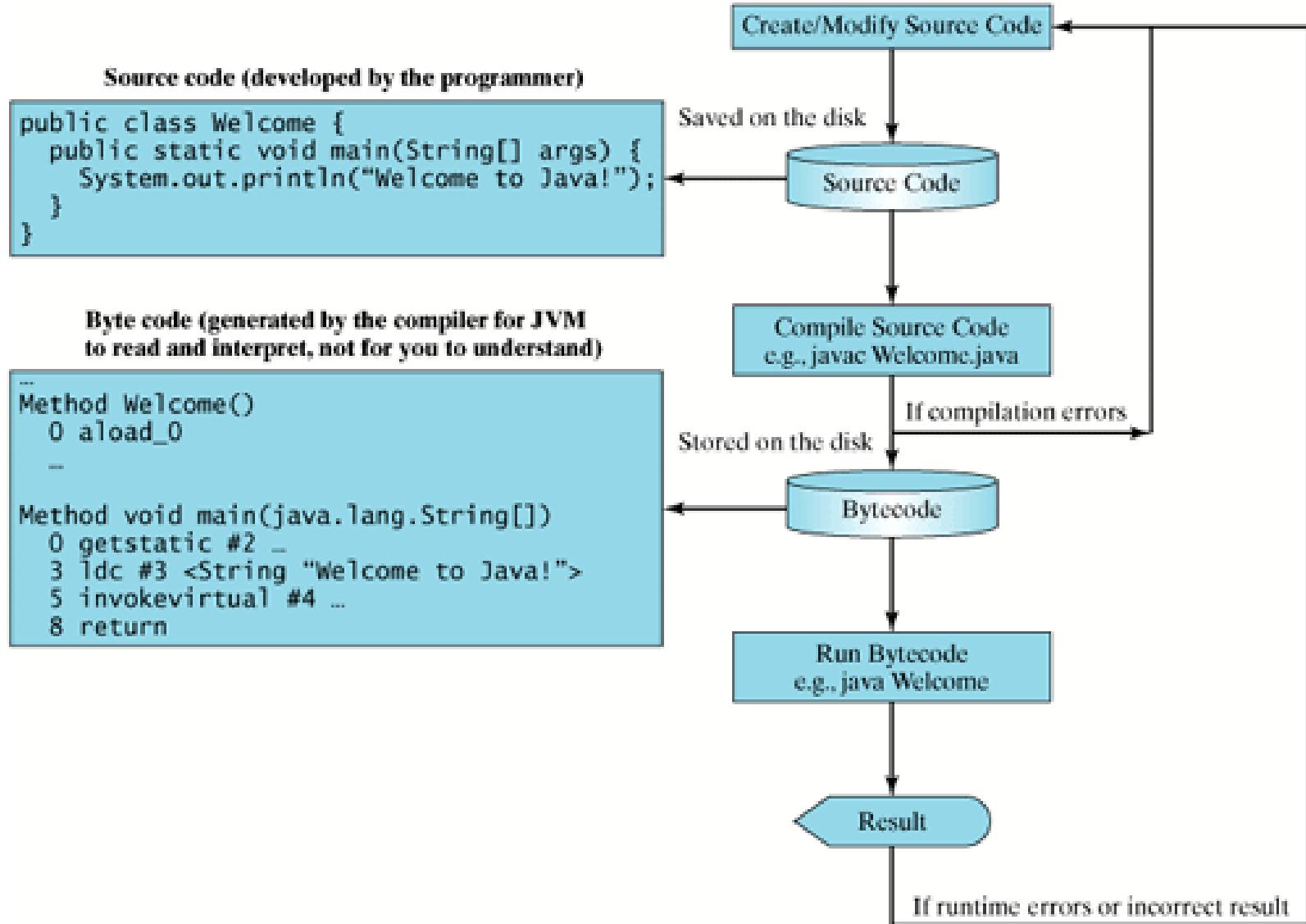


Figure 1.2 Java SE 6 Platform at a Glance

		Java Language									Java SE API	
JDK	Java Language	java	javac	javadoc	apt	jar	javap	JPDA	JConsole			
		Security	Int'l	RMI	IDL	Deploy	Monitoring	Troubleshoot	Scripting	JVM TI		
	Tools & Tool APIs	Deployment		Java Web Start				Java Plug-in				
		AWT				Swing			Java 2D			
	Deployment Technologies	Accessibility		Drag n Drop		Input Methods		Image I/O	Print Service	Sound		
		IDL	JDBC		JNDI		RMI		RMI-IIOP			
	User Interface Toolkits	Beans	Intl Support		Input/Output		JMX		JNI	Math		
		Networking	Override Mechanism			Security	Serialization		Extension Mechanism	XML JAXP		
	Integration Libraries	lang and util		Collections	Concurrency Utilities		JAR		Logging	Management		
		Preferences API	Ref Objects		Reflection		Regular Expressions		Versioning	Zip	Instrumentation	
	Other Base Libraries	Java Hotspot Client VM					Java Hotspot Server VM					
		Solaris		Linux		Windows			Other			

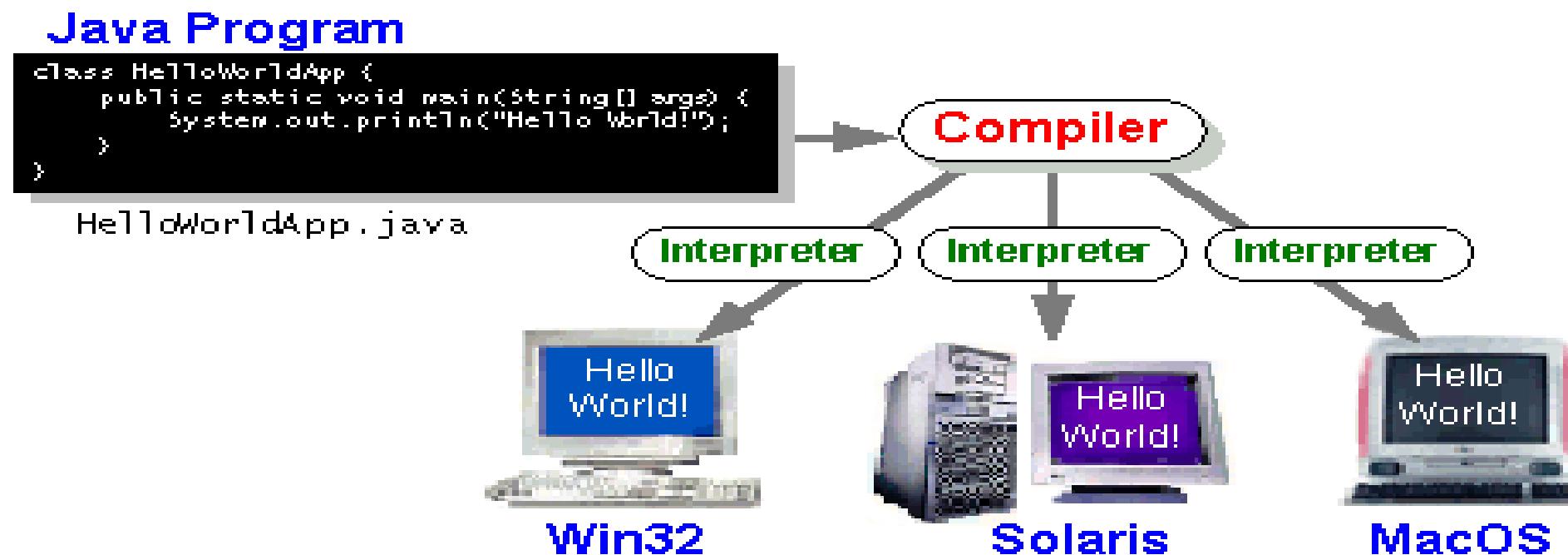
※ <http://java.sun.com/javase/technologies/index.jsp>

Development Process



Development Process (Cont.)

1. Create a *source file*
2. Compile the source file into a *bytecode* file
3. Run the program contained in the *bytecode* file



Creating a Source Code – HelloWorld.java

```
instructor@Ubuntu64-00:/home/instructor/JavaRoom
/*
 * Author : Peter Bok
 * When : August, 1, 2014
 * Objective : Hello World Code Test
 * Environment : Ubuntu 14.04 LTS Desktop, JDK 1.8.0_11, vi editor
 */

public class HelloWorld {
    public static void main(String [] args) {
        String msg = "Hello, World";
        System.out.println("msg = " + msg);
    }
}
~
~
~
~
~
~
~
:q
```

Compiling the Source Code – HelloWorld.java

- Java Compiler – **javac.exe**

```
instructor@Ubuntu64-00:~/JavaRoom$ ls  
HelloWorld.java  
instructor@Ubuntu64-00:~/JavaRoom$ javac HelloWorld.java  
instructor@Ubuntu64-00:~/JavaRoom$  
instructor@Ubuntu64-00:~/JavaRoom$ ls  
HelloWorld.class  HelloWorld.java
```

Interpreting the *bytecode* – HelloWorld.class

- Java Interpreter – **java.exe**

```
instructor@Ubuntu64-00:~/JavaRoom$ ls  
HelloWorld.class  HelloWorld.java  
instructor@Ubuntu64-00:~/JavaRoom$  
instructor@Ubuntu64-00:~/JavaRoom$  
instructor@Ubuntu64-00:~/JavaRoom$ java HelloWorld  
msg = Hello, World  
instructor@Ubuntu64-00:~/JavaRoom$ █
```

Command Line Tools

- JDK provides several command-line tools.
- Commonly used tools is a compiler, launcher/interpreter, archiver, documenter.
- Refer to
<http://java.sun.com/javase/6/docs/technotes/tools>

Command Line Tools - Compiler

- Translates Java source files into Java *bytecode*.
- Creates a *bytecode* file with the same name as the source file but with the **.class** extension.
- **javac [-options] [source files]**
 - `javac HelloWorld.java`
 - `javac -cp ./dir/classes/ HelloWorld.java`
 - `javac -d ./opt/hwapp/classes HelloWorld.java`
 - `javac -source 1.4 HelloWorld.java`
 - `javac -version`
 - `javac -help`
- Refer to <http://www.javaexpert.co.kr/entry/22>
- Refer to
<http://pllab.kw.ac.kr/j2seAPIs/tooldocs/windows/java.html>

Command Line Tools - Interpreter

- Handles the program execution, including launching the application.
- `java [-options] class [arguments...]` or `java [-options] -jar jarfile [arguments...]`
 - `java HelloWorld`
 - `java -cp .:/dir/Classes HelloWorld`
 - `java -ea HelloWorld`
 - `java -version`
 - `java -help`
 - `javaw <classname>`
- Refer to <http://javaexpert.co.kr/entry/23>
- Refer to
<http://pllab.kw.ac.kr/j2seAPIs/tooldocs/windows/java.html>

Command Line Tools - Packager

- JAR, Java Archive, utility is an archiving and compression tool.
- Used to combine multiple files into a single file called a JAR file.
- JAR consists of a ZIP archive containing a manifest file (JAR content describer) and optional signature files (for security).
- **jar [options] [jar-file] [manifest-files] [entry-point] [-C dir] files...**
 - **jar cf** files.jar HelloWorld.java kr/co/javaexpert/HelloWorld.class
 - **jar tfv** files.jar
 - **jar xf** files.jar
- Refer to <http://javaexpert.co.kr/entry/24>
- Refer to
<http://pllab.kw.ac.kr/j2seAPIs/tooldocs/windows/jar.html>

Command Line Tools – JAR Execution

- Can be created to be executable.
- Specifies the file within the JAR where the **main** class resides.
- Refer to <http://javaexpert.co.kr/entry/25>
 1. Compile **.java** file with package option.
 2. Create a file **Manifest.txt** using editor.
 3. Create a JAR file that adds the Manifest.txt contents to the manifest file, **MANIFEST.MF**.
 4. Display the contents of the JAR file.
 5. Execute the JAR file using **java -jar** option.

Figure 1.3. shows an examples of this basic communication.

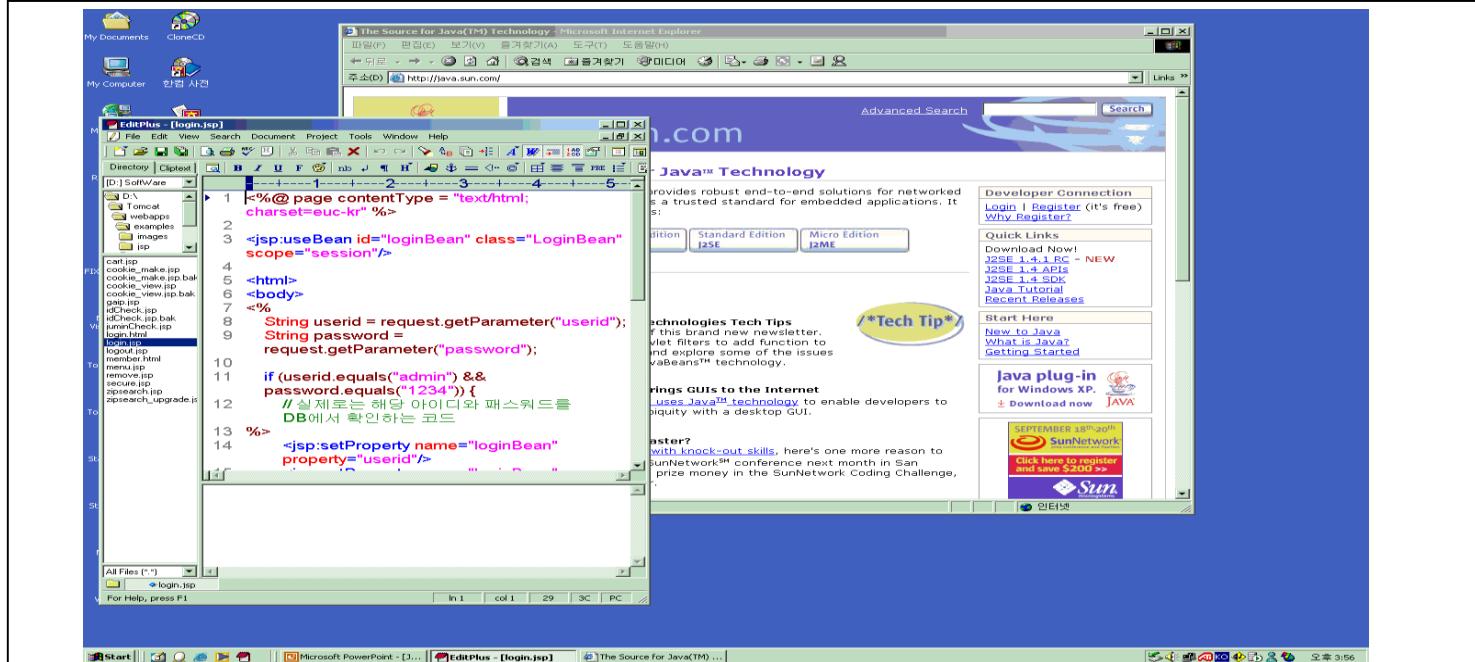
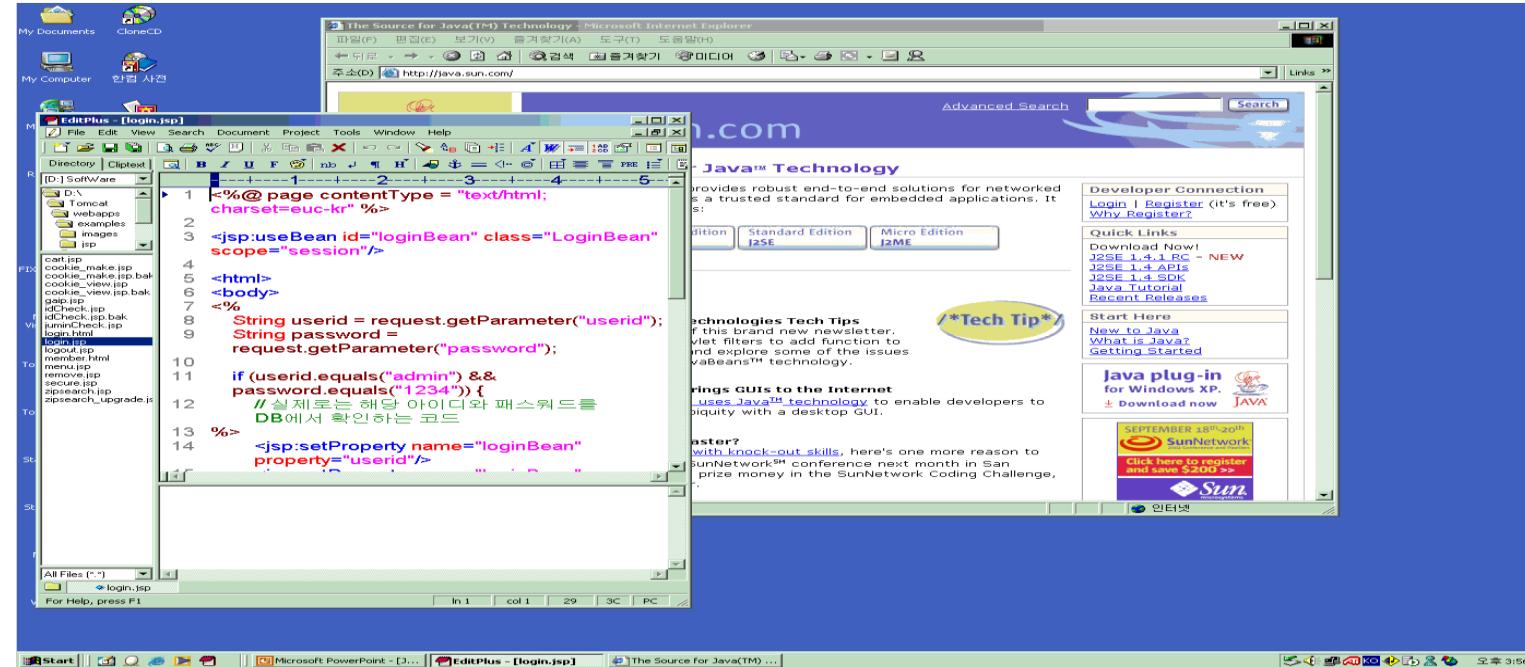


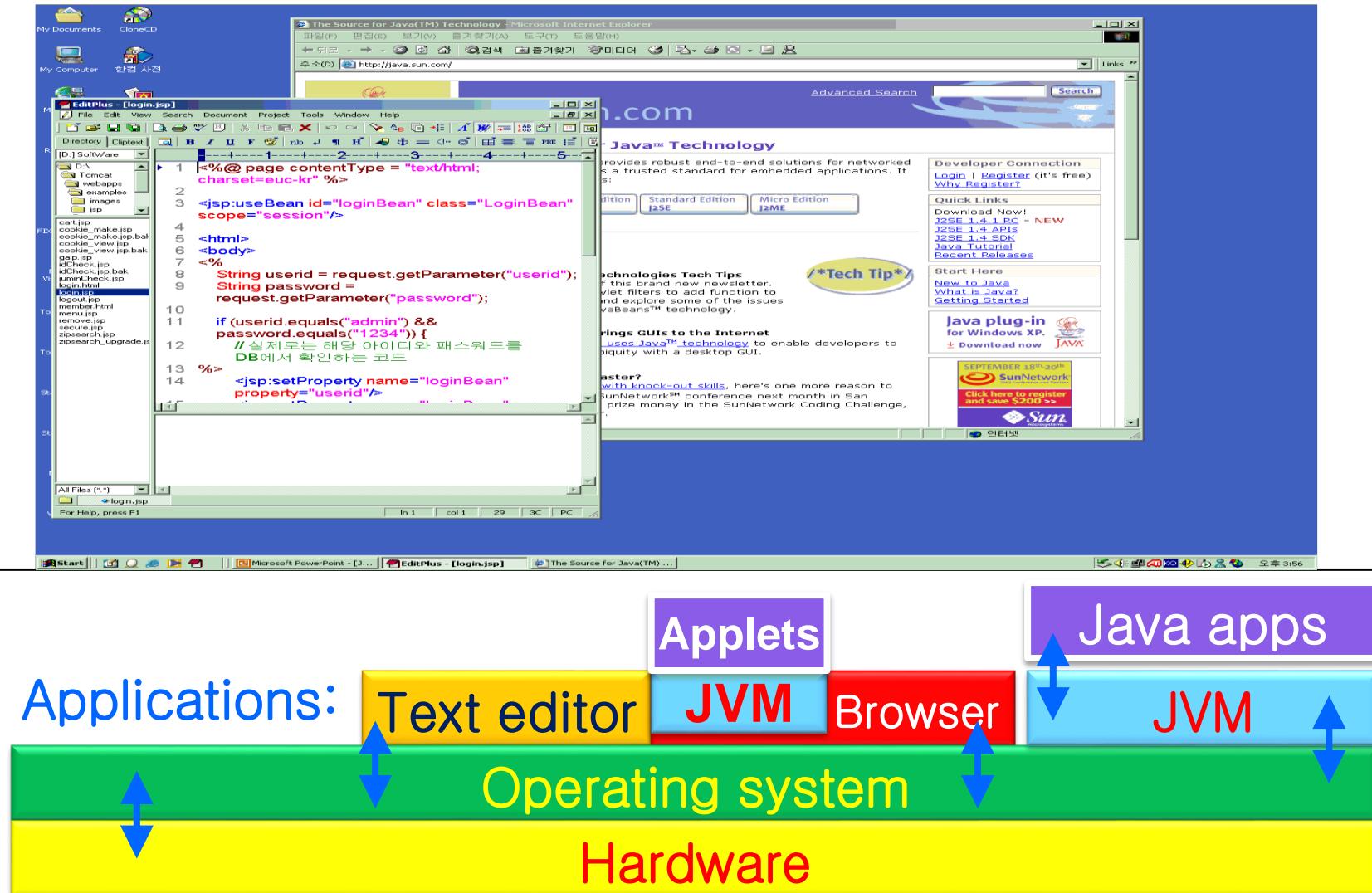
Figure 1.4. Computer Communication Problem



How Java Technology Solves the Communication Problem

- Uses compiling and interpretation.
- A little slower than compiled programs, but runs on any operating system.
- Compiles source code to *bytecode*.
- Uses Java virtual machine (JVM™), interprets *bytecode*.
- Uses a different JVM for every operating system.

Figure 1.5. How the JVM Interacts With the Operating System and Java Applets



Java API Documentation

- Detailed information API
- Very valuable resource: download, or view online at:

<http://docs.oracle.com/javase/7/docs/api/>

Figure 1.6. Java 2 Platform Specification, java.lang Package, Integer Class

The screenshot shows a Java documentation interface for the Java 2 Platform Specification, specifically the `java.lang` package. The left sidebar lists various Java classes and interfaces, with `java.lang` and `Integer` highlighted by red boxes. The main content area displays the `Class Integer` documentation, which includes the class hierarchy (`Object`, `Number`, `Integer`), implemented interfaces (`Serializable`, `Comparable<Integer>`), and a detailed description of the `Integer` class. The description notes that it wraps a primitive `int` value and provides methods for conversion between `String` and `int`. It also mentions implementation details from "Hacker's Delight". The interface includes tabs for Overview, Package, Class (which is selected), Use, Tree, Deprecated, Index, and Help, along with links for Summary, Nested, Field, Constr, Method, Detail, and Field, Constr, Method.

java.awt.image
java.awt.image.renderable
java.awt.print
java.beans
java.beans.beancontext
java.io
java.lang
java.lang.annotation
java.lang.instrument
java.lang.invoke
java.lang.management
java.lang.ref
java.lang.ThreadLocal
java.lang.Integer
java.lang.Long
java.lang.Math
java.lang.Number
java.lang.Object
java.lang.Package
java.lang.Process
java.lang.ProcessBuilder
java.lang.ProcessBuilder.Redirect
java.lang.Runtime
java.lang.RuntimePermission
java.lang.SecurityManager
java.lang.Short
java.lang.StackTraceElement
java.lang.StrictMath
java.lang.String
java.lang.StringBuffer
java.lang.StringBuilder
java.lang.System
java.lang.Thread
java.lang.ThreadGroup
java.lang.ThreadLocal

Overview Package **Class** Use Tree Deprecated Index Help

Java™ Platform Standard Ed. 7

Prev Class Next Class Frames No Frames

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

java.lang

Class Integer

java.lang.Object
java.lang.Number
java.lang.Integer

All Implemented Interfaces:

Serializable, Comparable<Integer>

public final class **Integer**
extends Number
implements Comparable<Integer>

The `Integer` class wraps a value of the primitive type `int` in an object. An object of type `Integer` contains a single field whose type is `int`. In addition, this class provides several methods for converting an `int` to a `String` and a `String` to an `int`, as well as other constants and methods useful when dealing with an `int`.

Implementation note: The implementations of the "bit twiddling" methods (such as `highestOneBit` and `numberOfTrailingZeros`) are based on material from Henry S. Warren, Jr.'s *Hacker's Delight*, (Addison Wesley, 2002).

Since:

JDK1.0

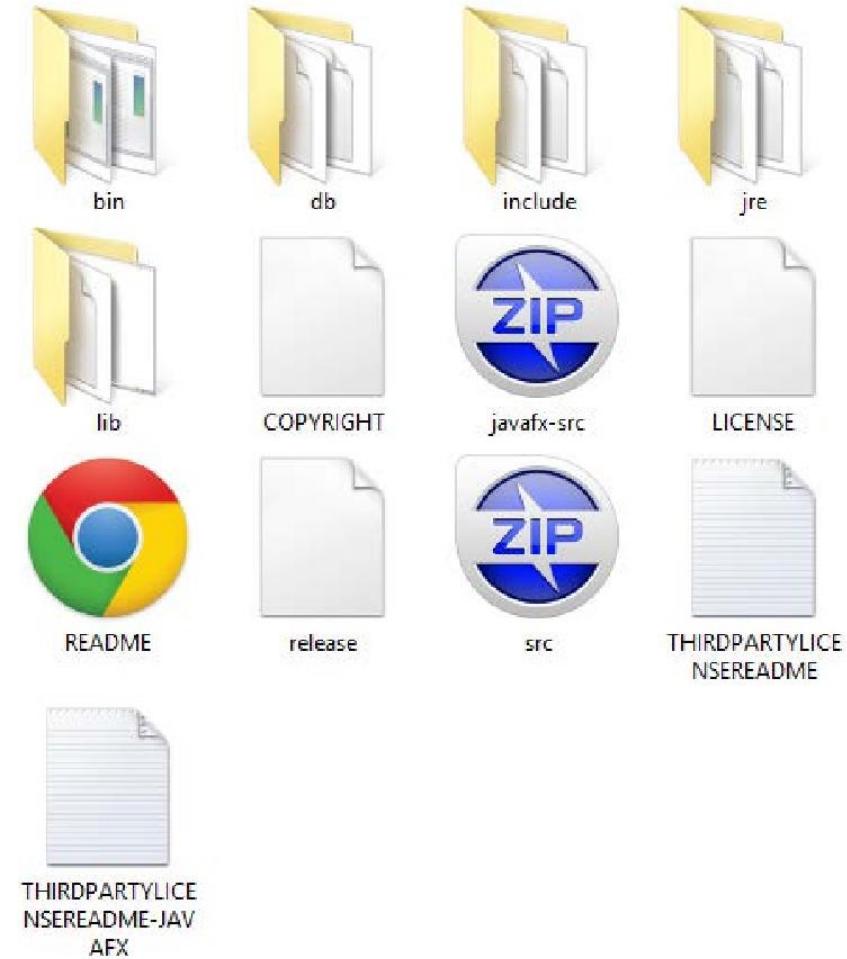
See Also:

Serialized Form

Field Summary

JDK File Structure

- bin : contains the tools used for developing a Java application including the compiler and JVM.
- db : is the Apache Derby relational database.
- include : contains header files used to interact with C applications.
- jre : is a JRE used by the JDK.
- src.zip : includes the actual code for the core classes, called the SDK.



Additional Resource

- Java Technology : An Early History
 - <http://oracle.com.edgesuite.net/timeline/java/>
 - <http://www.cs.umd.edu/class/spring2002/cmsc434-0101/MUIseum/applications/index.html>
- Java Language and Virtual Machine Specifications
 - <http://docs.oracle.com/javase/specs/>
- Java SE6 API Hangul Documentation
 - <http://docs.xrath.com/java/se/6/docs/ko/>
 - <http://docs.xrath.com/java/se/6/docs/ko/api/index.html>
- Comparison of Java and C++
 - http://en.wikipedia.org/wiki/Comparison_of_Java_and_C%2B%2B
 - http://verify.stanford.edu/uli/java_cpp.html