



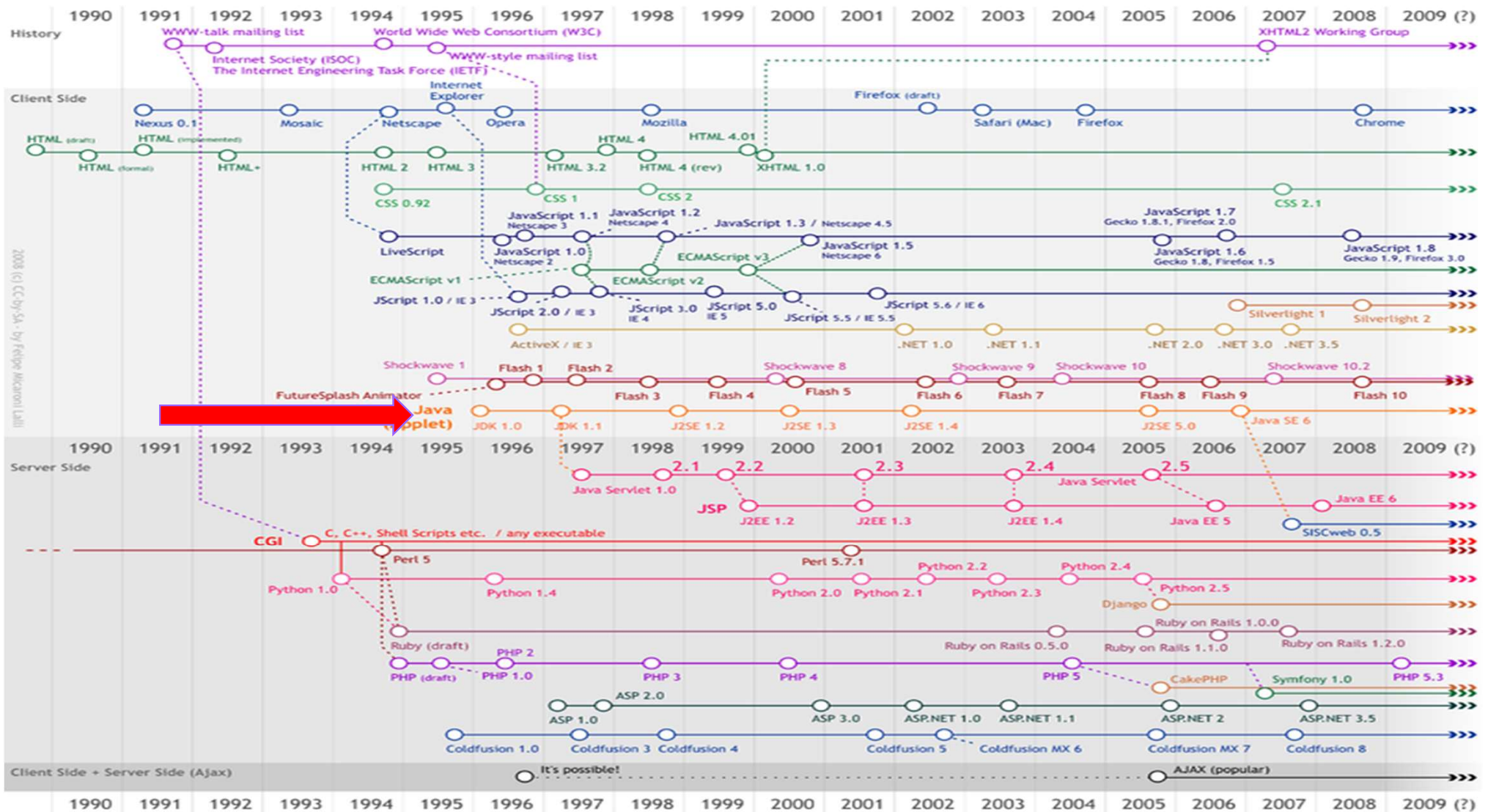
Java Programming Language Overview

Bok, Jong Soon
javaexpert@nate.com
<https://github.com/swacademy/Core-Java>

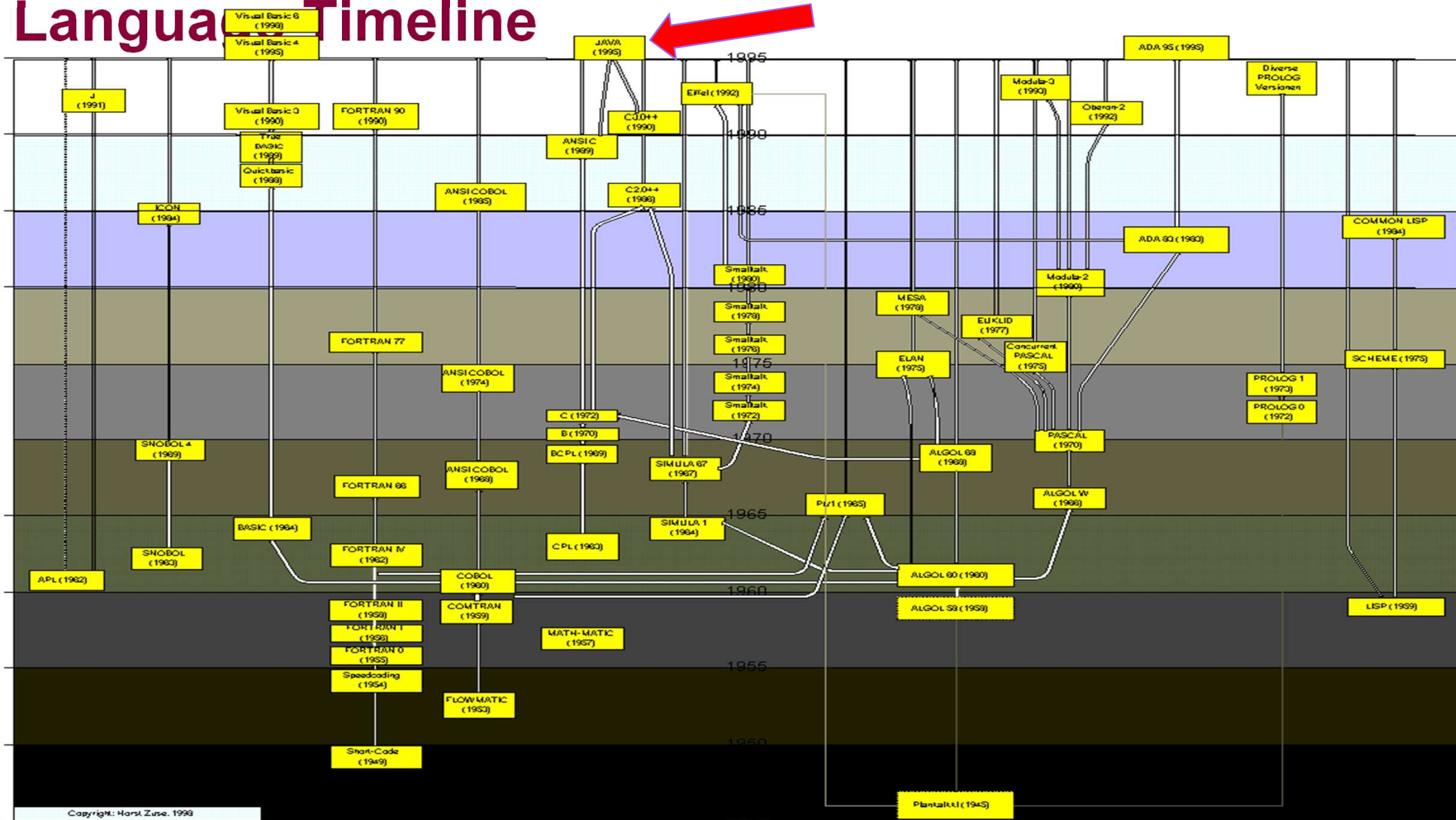
What is Java Technology?

- Is a programming language.
- Is a platform.

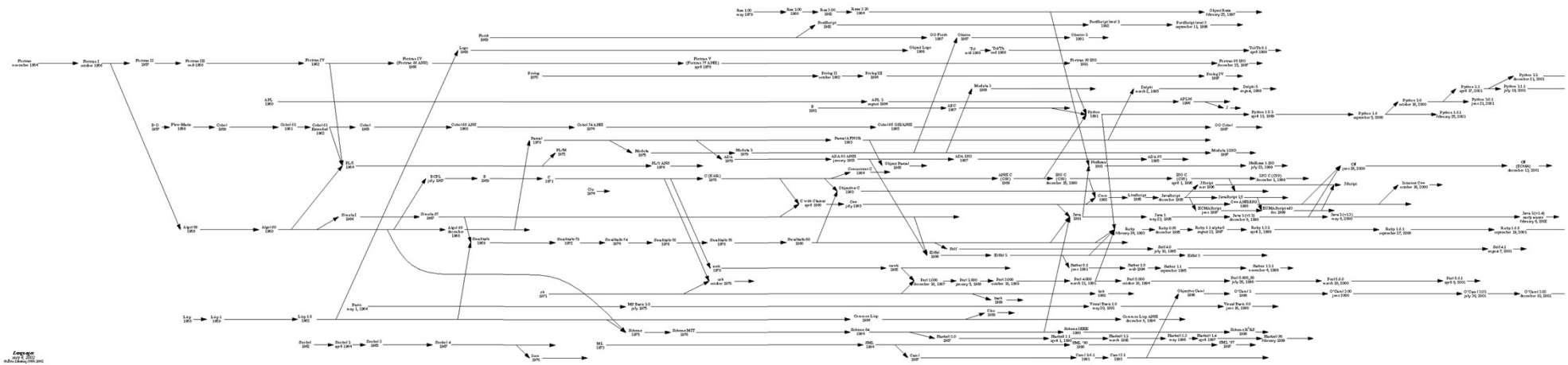
Web Timeline



Language Timeline








<http://www.cs.umd.edu/class/spring2002/cmsc434-0101/MUIseum/applications/langtl2.html>



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.)

May 2024	May 2023	Change	Programming Language		Ratings	Change
1	1			Python	16.33%	+2.88%
2	2			C	9.98%	-3.37%
3	4	⬆		C++	9.53%	-2.43%
4	3	⬇		Java	8.69%	-3.53%
5	5			C#	6.49%	-0.94%
6	7	⬆		JavaScript	3.01%	+0.57%
7	6	⬇		Visual Basic	2.01%	-1.83%
8	12	⬆		Go	1.60%	+0.61%
9	9			SQL	1.44%	-0.03%
10	19	⬆		Fortran	1.24%	+0.46%
11	11			Delphi/Object Pascal	1.24%	+0.23%
12	10	⬇		Assembly language	1.07%	-0.13%
13	18	⬆		Ruby	1.06%	+0.26%

Source : <https://www.tiobe.com/tiobe-index/>

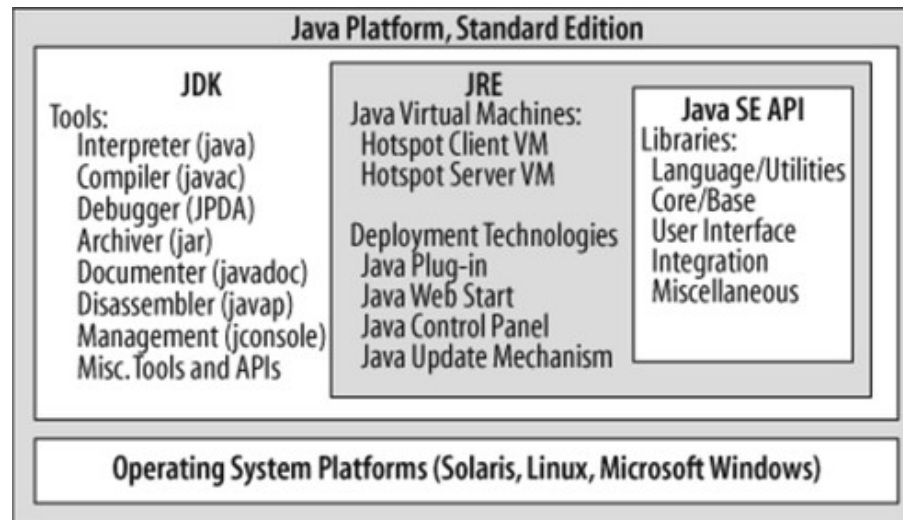
Java Programming Language (Cont.)

Programming Language	2024	2019	2014	2009	2004	1999	1994	1989
Python	1	4	8	6	10	28	22	-
C	2	2	1	2	2	1	1	1
C++	3	3	4	3	3	2	2	2
Java	4	1	2	1	1	15	-	-
C#	5	6	5	7	8	25	-	-
JavaScript	6	7	9	9	9	20	-	-
Visual Basic	7	19	-	-	-	-	-	-
SQL	8	9	-	-	7	-	-	-
PHP	9	8	6	5	6	-	-	-
Go	10	18	36	-	-	-	-	-
Objective-C	30	10	3	36	45	-	-	-
Lisp	35	30	14	20	15	13	6	3
(Visual) Basic	-	-	7	4	5	3	3	7

Source : <https://www.tiobe.com/tiobe-index/>

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 <https://www.oracle.com/java/technologies/>
- Download older versions at <https://www.oracle.com/java/technologies/downloads/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

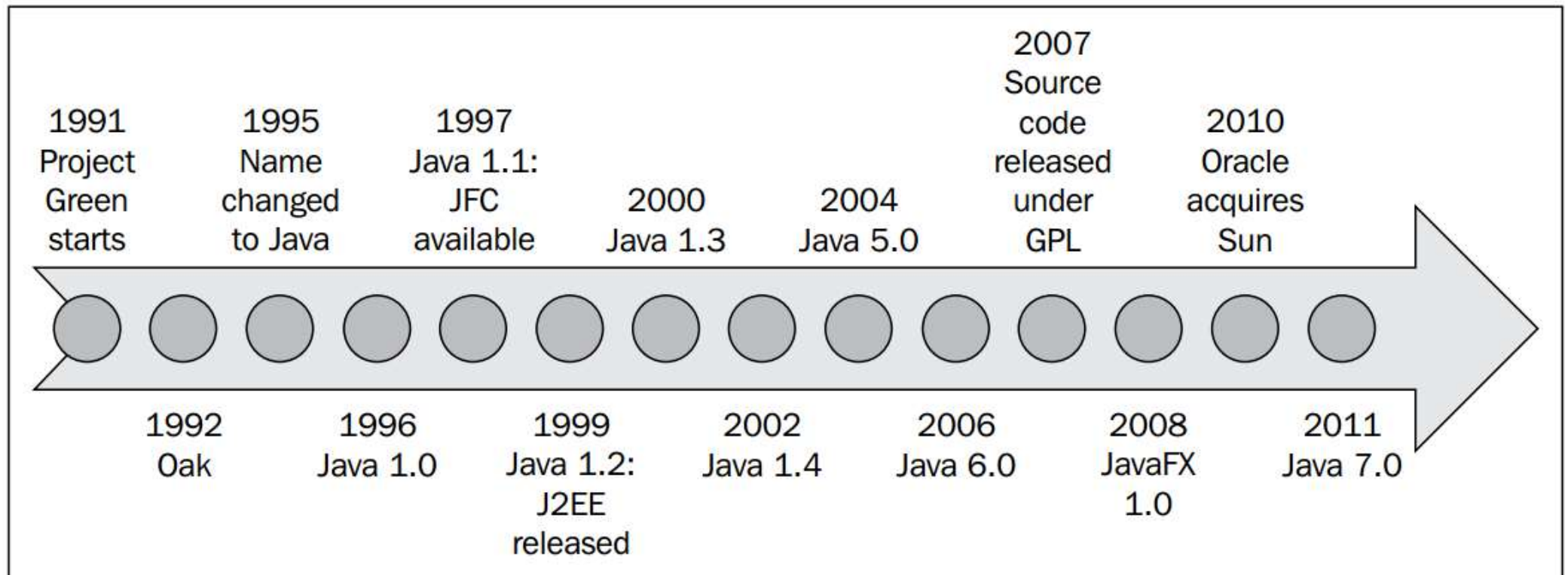
Java SE Code Names

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

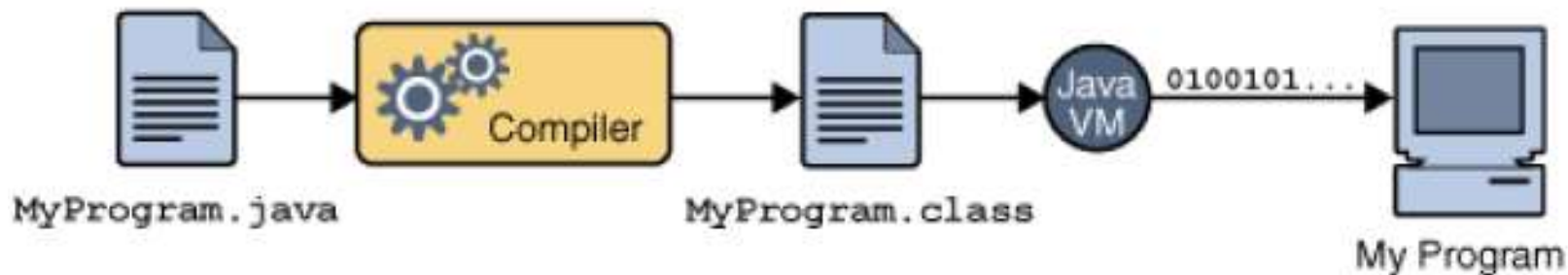
- Originally named *Oak*, designed in 1991.
- Main team members : Bill Joy, Patrick Naughton, Mike Sheridan, James Gosling.
- Original goal : use in embedded consumer electronic appliances.
- In 1994, team realized Oak was perfect for *Internet*.
- In 1995, renamed Java, was redesigned for developing 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.)

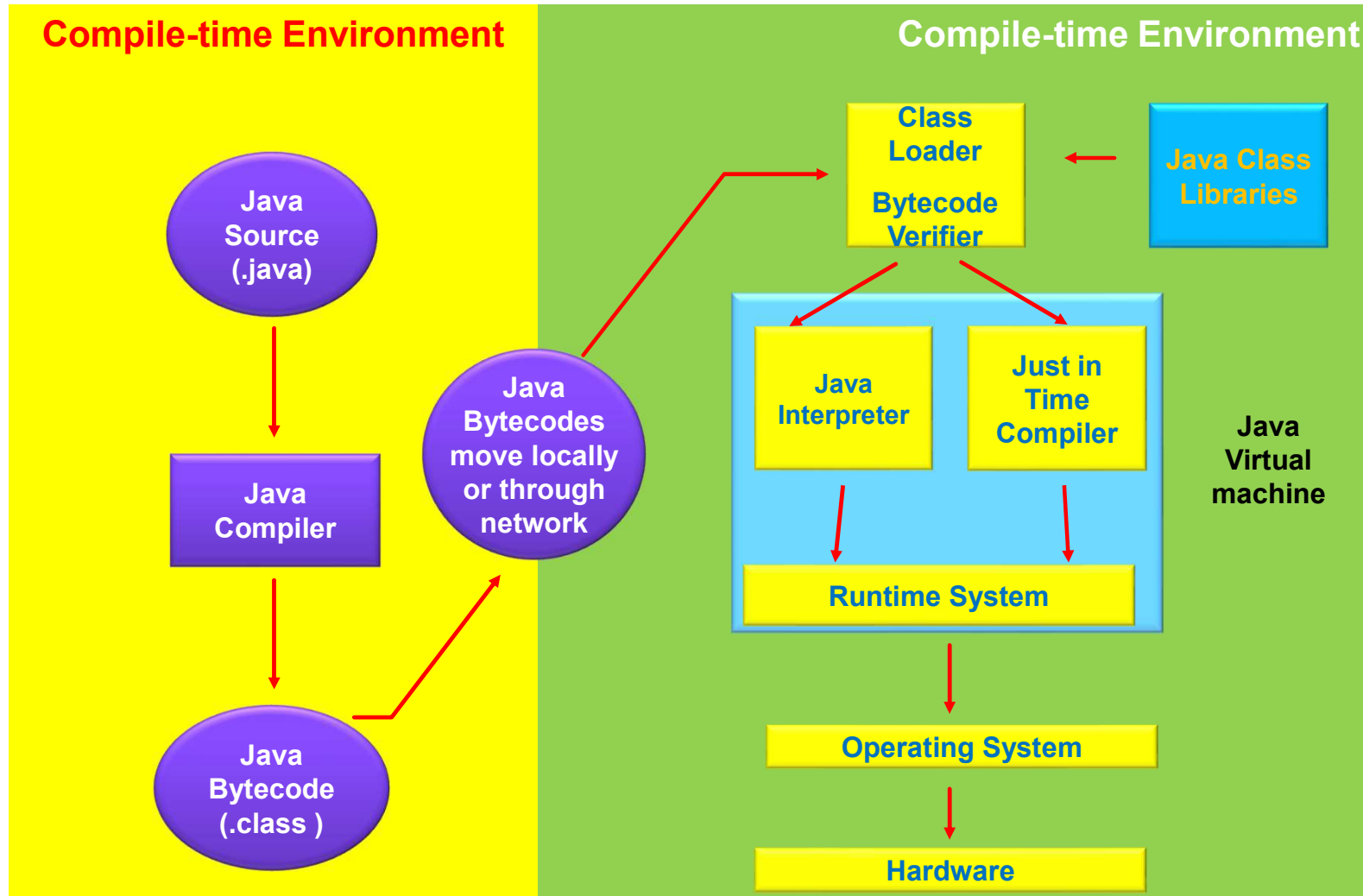


Features

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



How it works...!

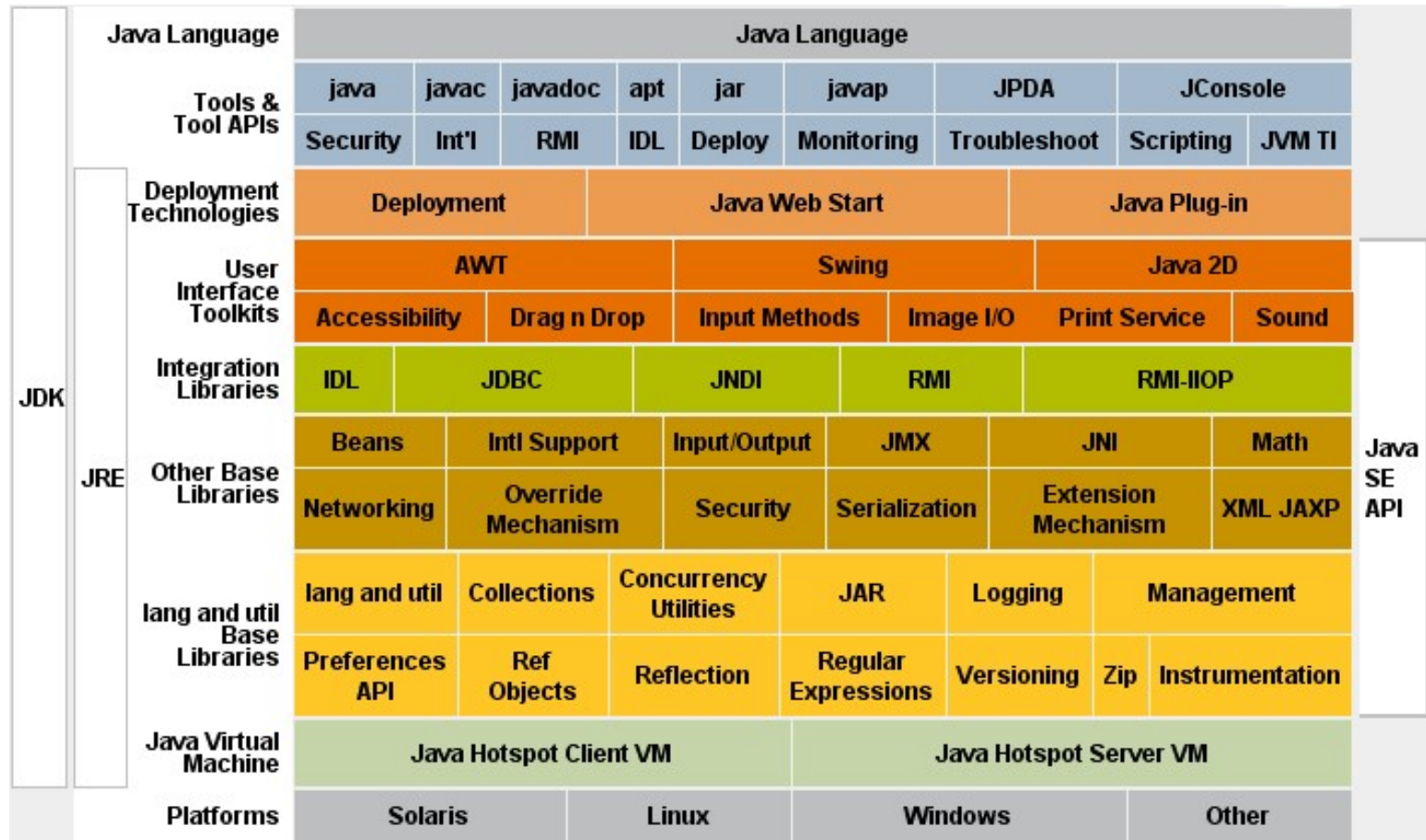


Source : Java tutorial PPT, by Intelligo Technologies on Mar 08, 2011

Figure 1.1 J2SE vs. J2EE vs. J2ME

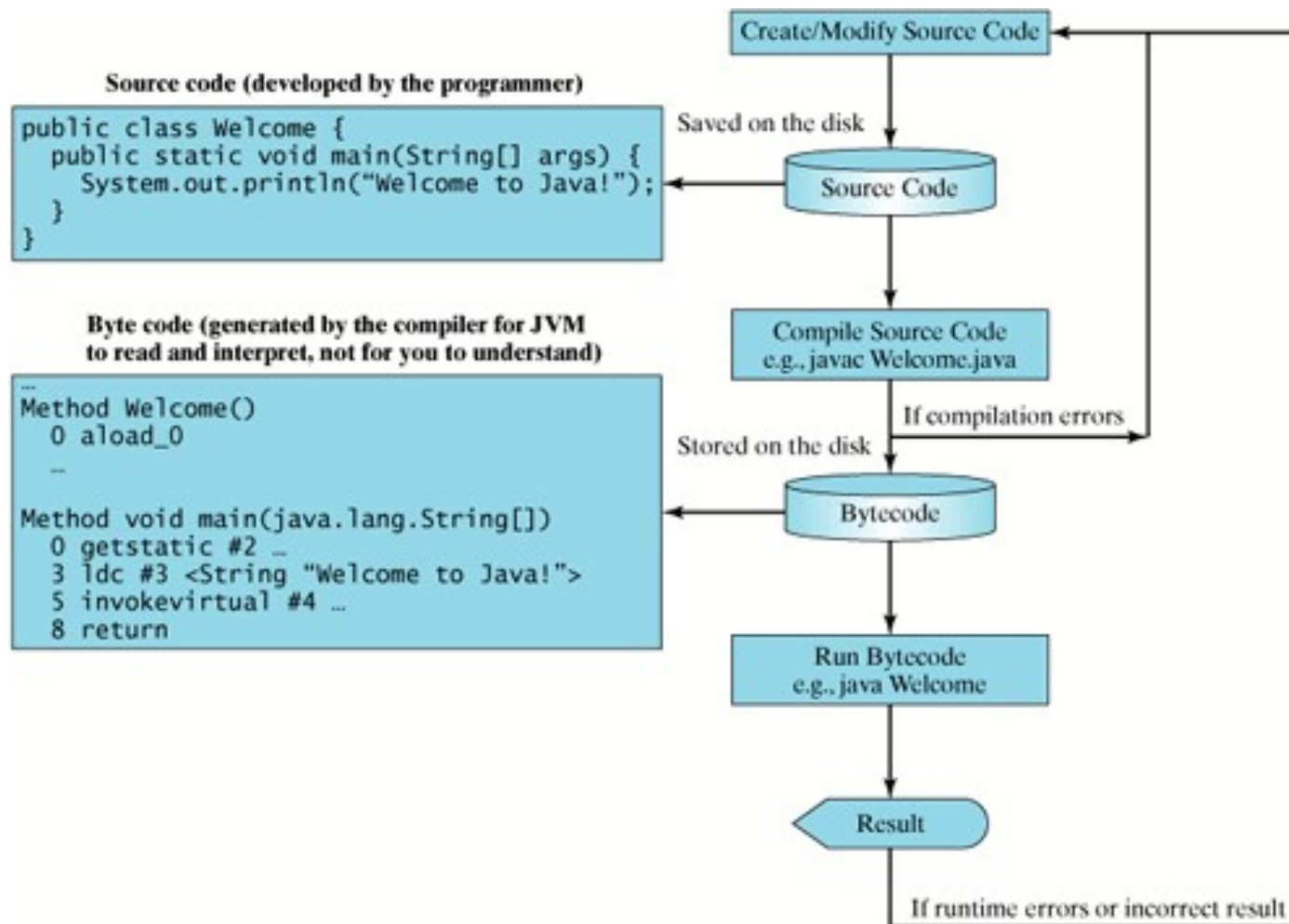


Figure 1.2 Java SE 6 Platform at a Glance



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

Development Process



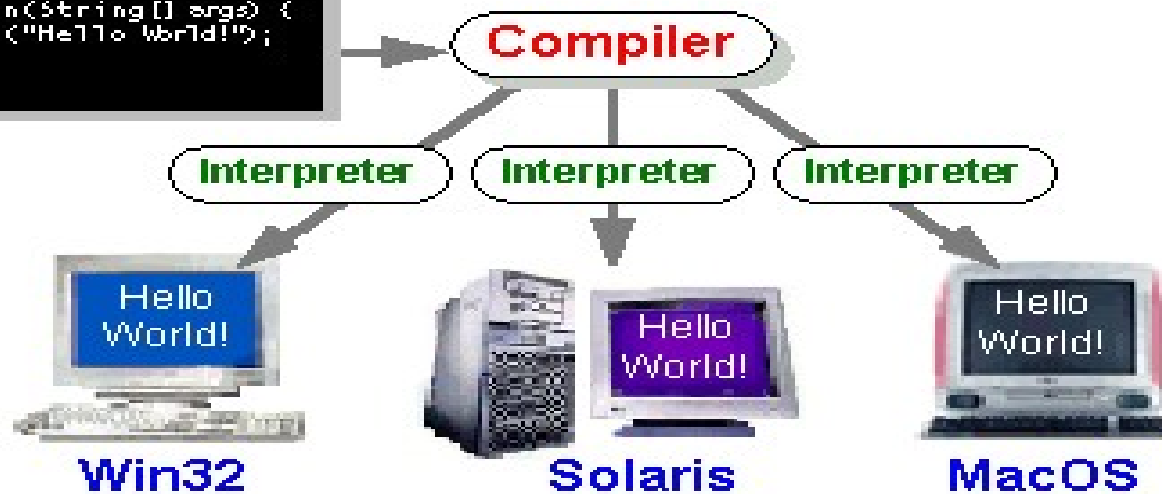
Development Process (Cont.)

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

Java Program

```
class HelloWorldApp {  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```

HelloWorldApp.java



Creating a Source Code – HelloWorld.java

```
1  /*
2   * Author : Henry
3   * When : Jul, 1, 2024
4   * Objective : Java First Coding
5   * Environment : Windows 11 Enterprise Ed., JDK 17.0.10, Microsoft Visual Studio Code 1.87.0
6   *
7   */
8  public class HelloWorld {
9      Run | Debug
10     public static void main(String [] args){
11         String str = "Hello, World";
12         System.out.printf(format:"str = %s\n", str);
13     }
14 }
```

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 <https://docs.oracle.com/en/java/javase/17/docs/specs/man/index.html>

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 <https://bluemonnd.tistory.com/entry/22>

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 <https://bluemonnd.tistory.com/entry/23>

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 <https://bluemonnd.tistory.com/entry/24>

Command Line Tools – JAR Execution

- Can be created to be executable.
- Specifies the file within the JAR where the **main** class resides.
- Refer to <https://bluemonad.tistory.com/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.

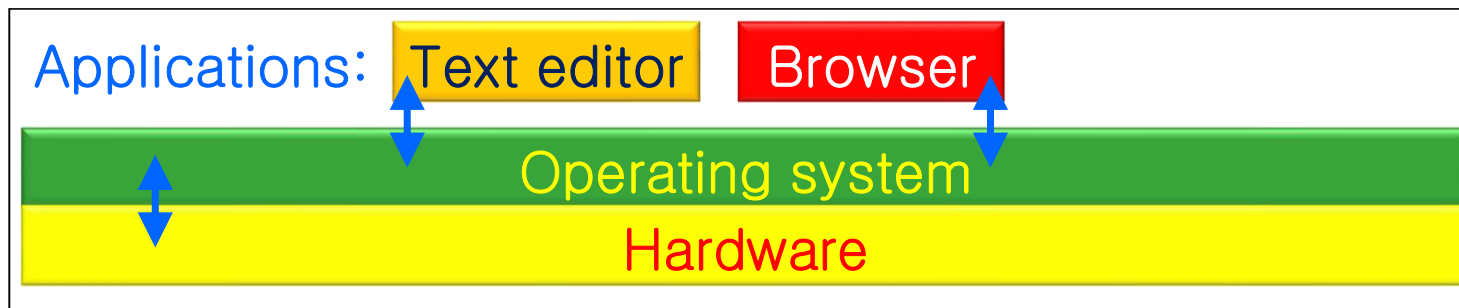
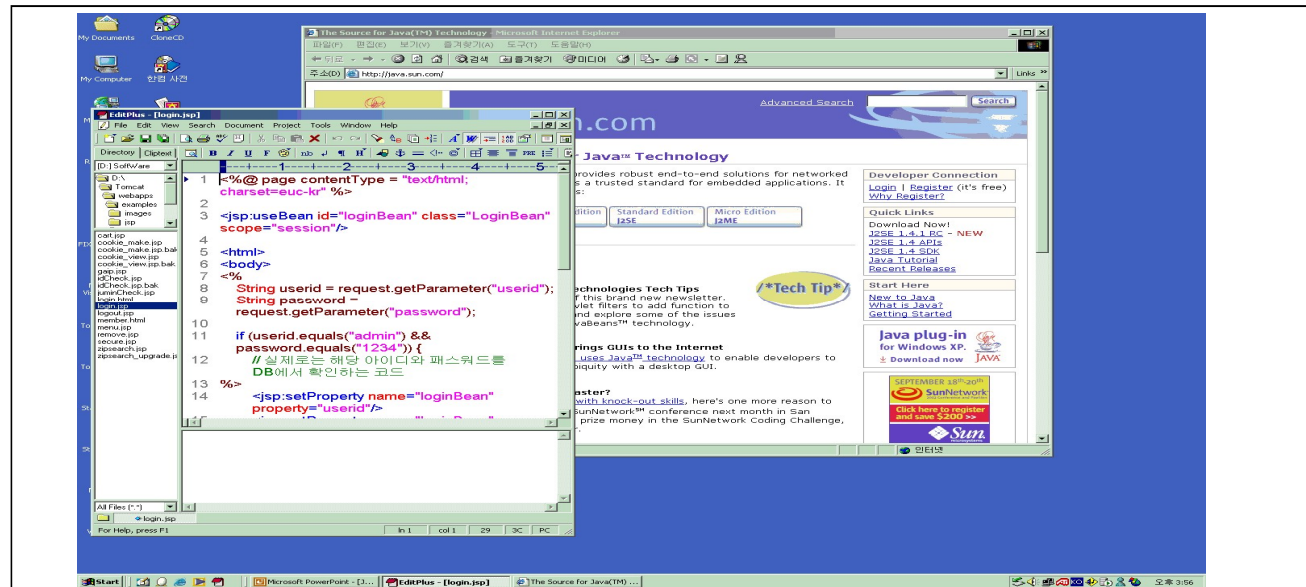
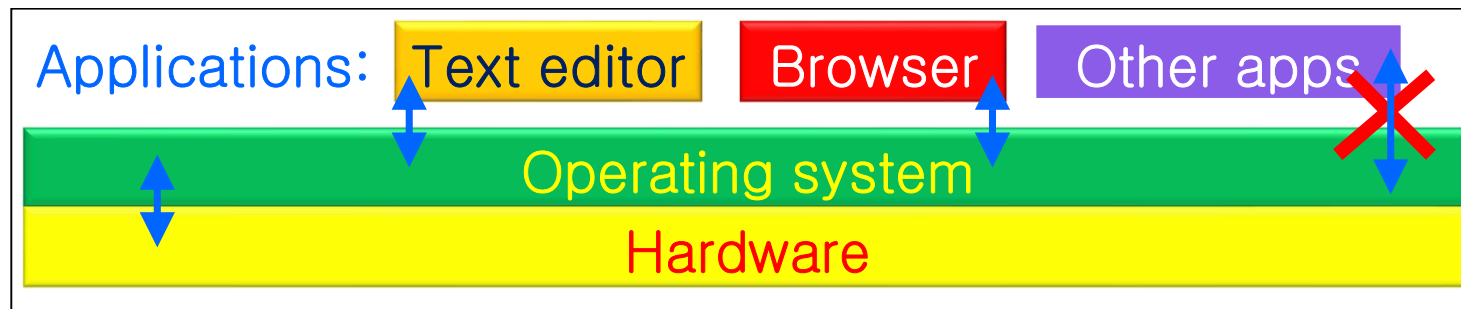
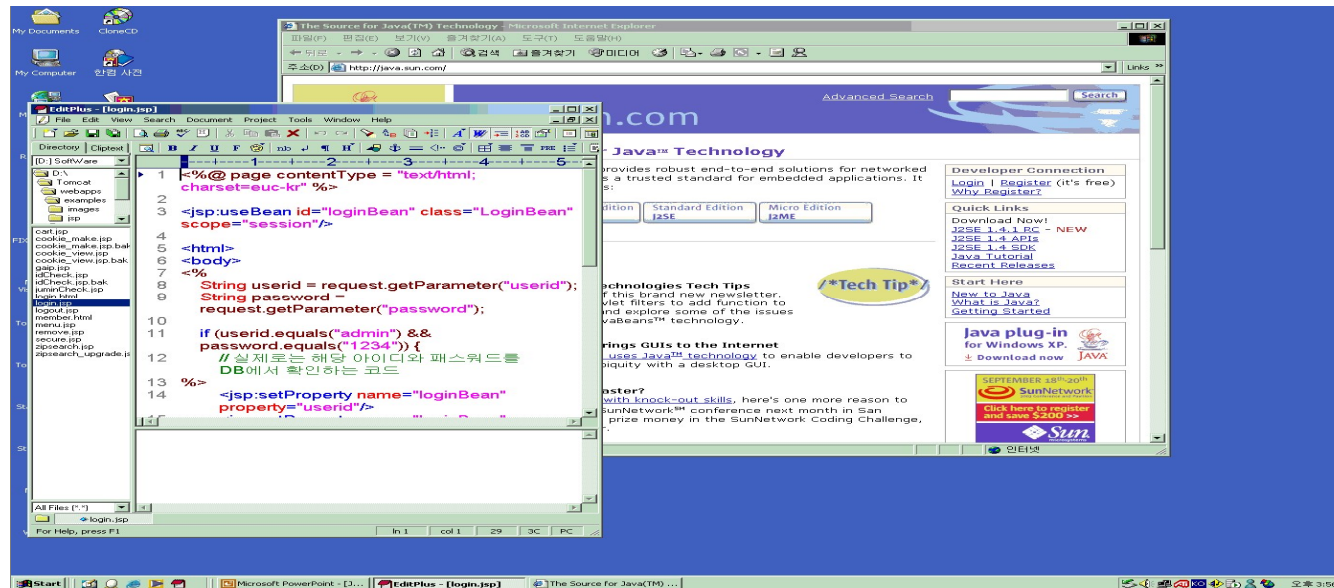


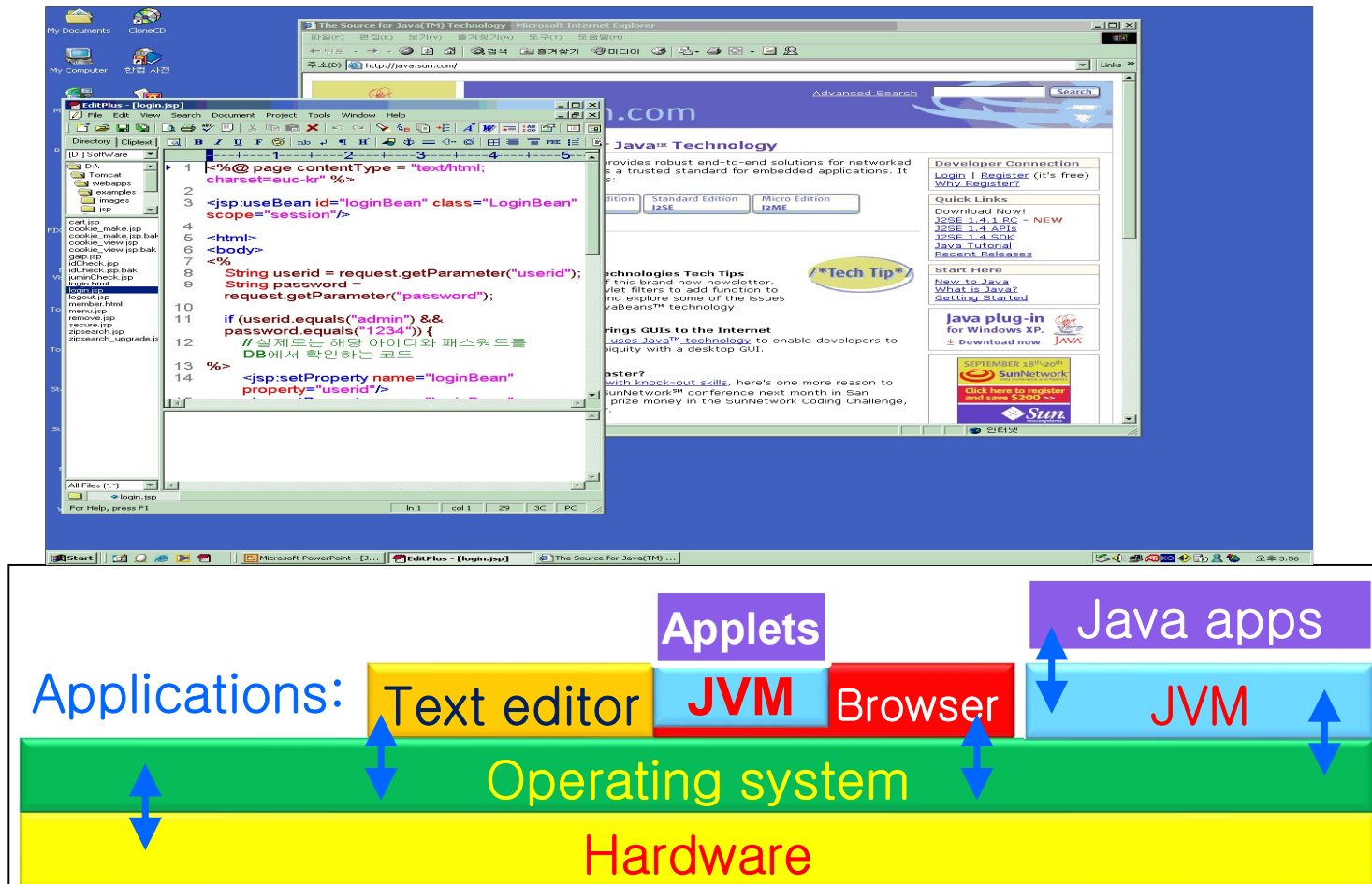
Figure 1.4. Computer Communication Problem



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:

<https://docs.oracle.com/en/java/javase/17/docs/api/index.html>

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

The screenshot shows the Java SE 17 & JDK 17 documentation page for the `Integer` class. The page is titled "Class Integer" and is part of the `java.lang` package. The class is highlighted with a red box. The page includes a navigation bar with tabs for Overview, Module, Package, Class, Use, Tree, Preview, New, Deprecated, Index, and Help. The "Class" tab is selected. The page content includes the class hierarchy, implemented interfaces, class declaration, and a field summary table.

Module `java.base`
Package `java.lang`
Class `Integer`

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

All Implemented Interfaces:
`Serializable`, `Comparable<Integer>`, `Constable`, `ConstantDesc`

```
public final class Integer
    extends Number
    implements Comparable<Integer>, Constable, ConstantDesc
```

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`.

This is a value-based class; programmers should treat instances that are equal as interchangeable and should not use instances for synchronization, or unpredictable behavior may occur. For example, in a future release, synchronization may fail.

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:
1.0

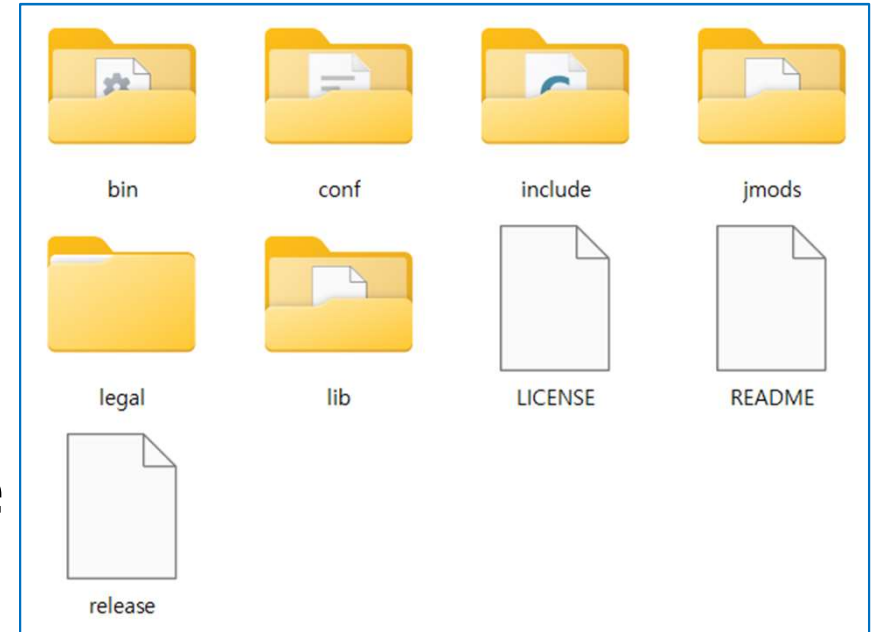
See Also:
[Serialized Form](#)

Field Summary

Fields	Modifier and Type	Field	Description
	static final int	<code>BYTES</code>	The number of bytes used to represent an <code>int</code> value in two's complement binary form.
	static final int	<code>MAX_VALUE</code>	A constant holding the maximum value an <code>int</code> can have, $2^{31}-1$.
	static final int	<code>MIN_VALUE</code>	A constant holding the minimum value an <code>int</code> can have, -2^{31} .
	static final int	<code>SIZE</code>	The number of bits used to represent an <code>int</code> value in two's complement binary form.
	static final Class<Integer>	<code>TYPE</code>	The Class instance representing the primitive type <code>int</code> .

JDK File Structure

- bin : contains the tools used for developing a Java application including the compiler and JVM.
- include : contains header files used to interact with C applications.
- lib/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