Introduction to Java and the SDK Tools

Qiuyan Huo 霍秋艳
Software Engineering Institute
qyhuo@mail.xidian.edu.cn

计算机事业发展的下一个浪潮就是 Java，并且将很快会发生的。

Tim Berners-Lee

James Gosling

May 19, 1955

'father of Java'

History of Java

一切从C++开始！

History of Java

Oak:窗外的一棵老橡树！

History of Java

Oak → Java

History of Java

太平洋上一个盛产咖啡的岛屿

History of Java

WWW → WebRunner

History of Java

1995.5.23 HotJava

History of Java

Internet上的世界语！

History of Java

Web era

Java for websites

- **Popular sites using Java**
  - Linkedin.com
  - Msn.com
  - Ebay.com
  - 163.com
  - Paypal.com
  - Sohu.com
  - Odnoklassniki.ru
  - Mywebsearch.com
  - Aol.com
  - Ebay.de
- **Random selection of sites using Java**
- **Sites using Java only recently**
  - Mobile.de
  - 12306.cn
  - Marktplaats.nl
  - Caixa.gov.br
  - Brainyquote.com

History of Java

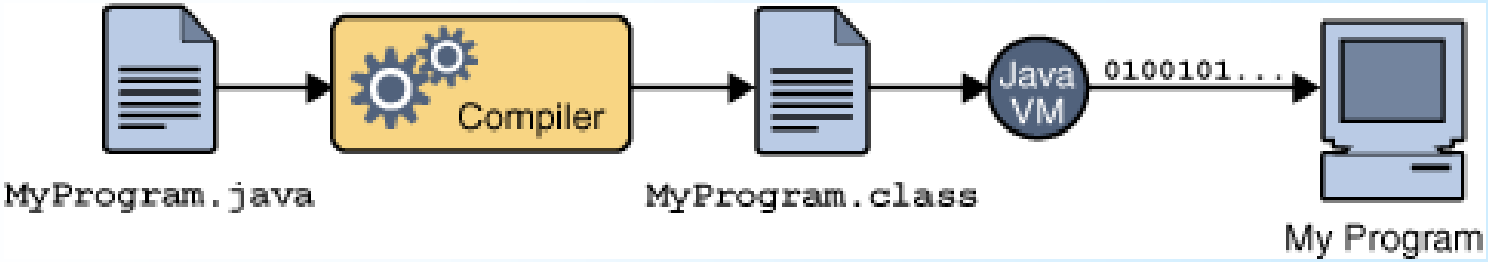
- Java technology was created as a programming tool in a small, closed-door project initiated by Patrick Naughton, Mike Sheridan, and James Gosling of Sun in 1991. But creating a new language wasn't even the point of "the Green Project."
- The TV set-top box and video-on-demand industries (Demo)
- 1994: Change of direction – The Internet. By March 1995, there were still only seven or eight binary copies of what they called "1.0a" outside of Sun. The team was getting ready to post a "full public" alpha version ("1.0a2") of the Java source code on the Internet.
- 13 November 2006 ~ 8 May 2007, Sun released all of Java's core code free and open-source, aside from a small portion of code to which Sun did not hold the copyright.
- 2009, Oracle收购Sun~ 20 October 2010, 完成了合并
- 2011 Java 7
- 2014 Java 8
- 2017 Java 9

What's Java?

- 一种编程语言→programming
- 一种开发环境→developing
- 一种应用环境→application
- 一种发布环境→deployment
- Java是一种高级编程语言

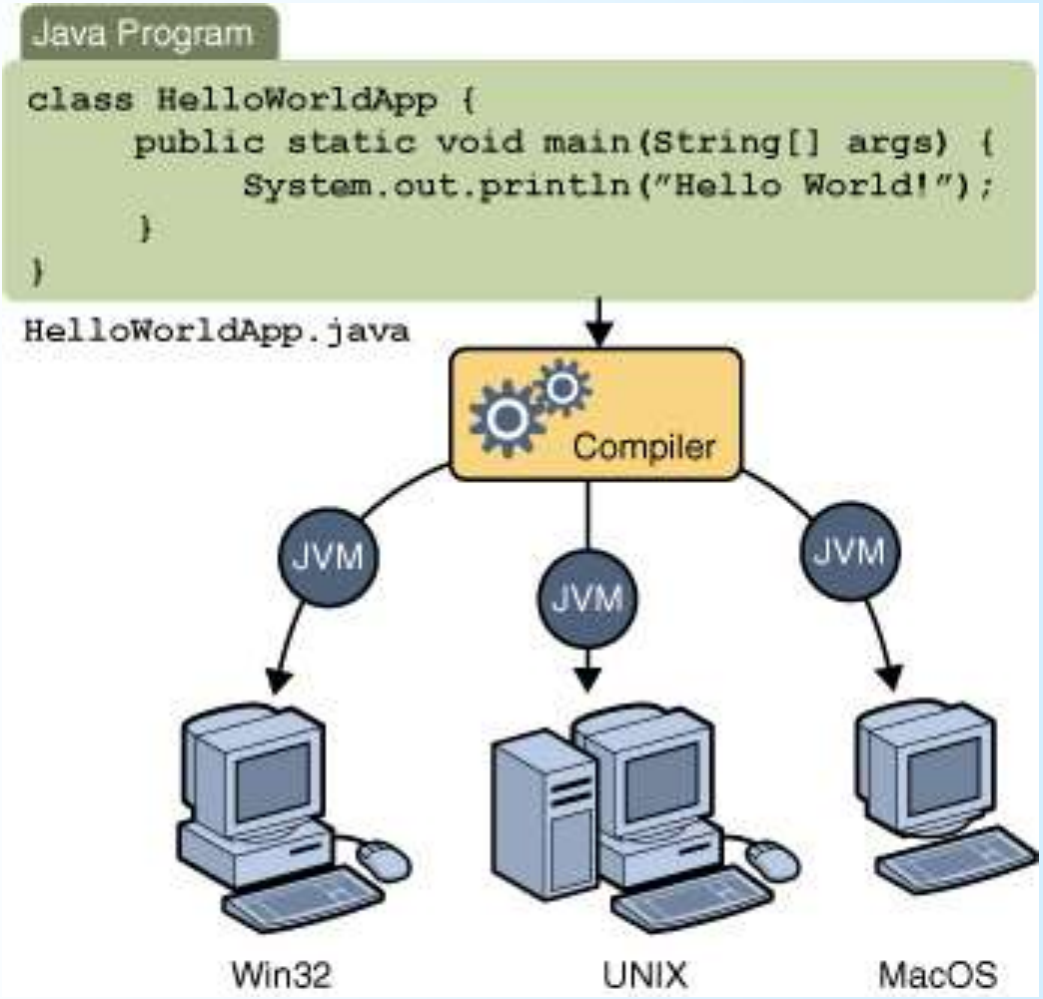
What's Java?

Java虚拟机(JVM)和Java字节码(ByteCode):



What's Java?

"*Write once, Run anywhere*"



Why Java?

- 避免其它语言的缺点（指针，内存管理）
- 更快的开发
- 更好的编写
- 更少的代码
- 一次编写，到处执行
- 更容易的发布

Java (How?)

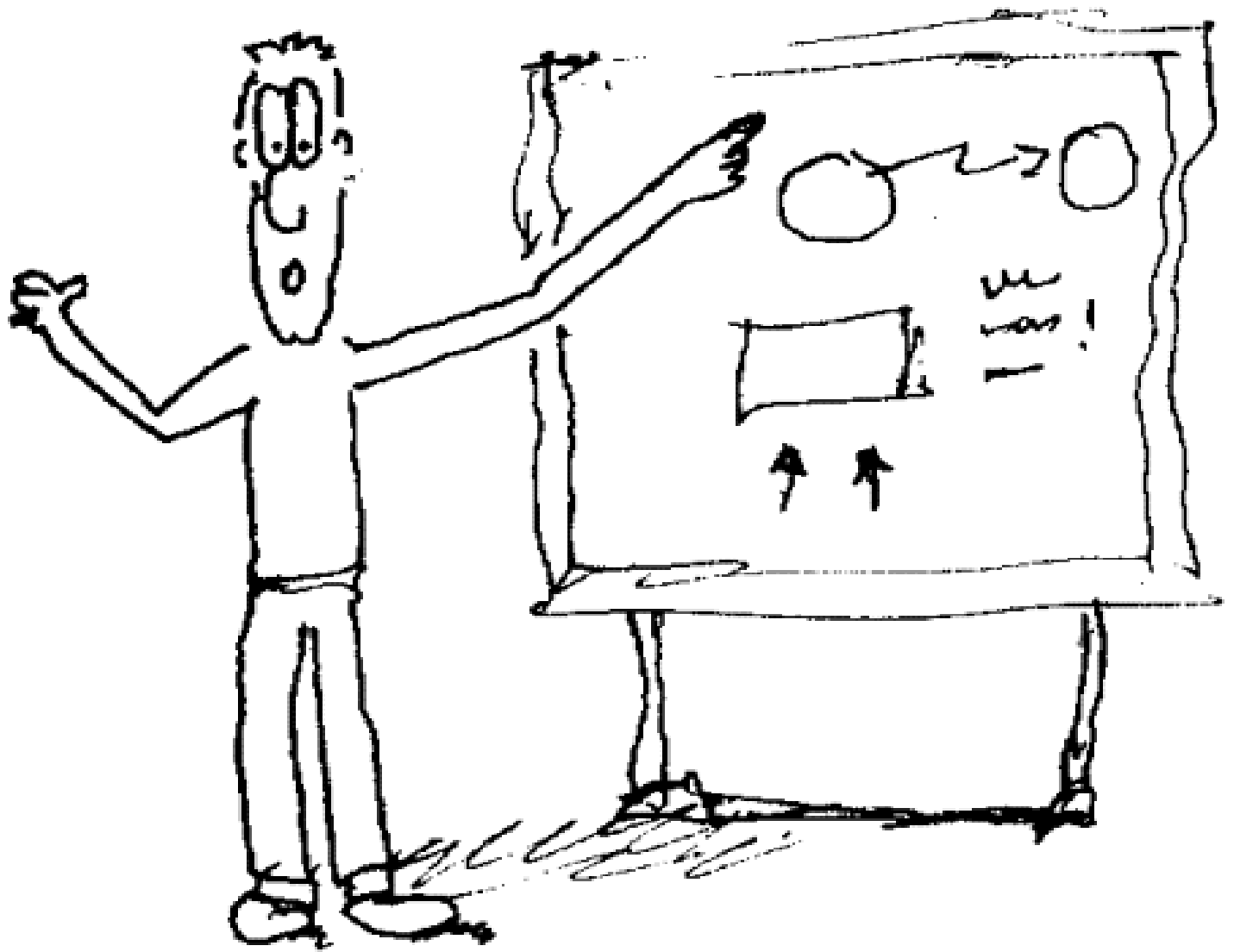
- Java虚拟机 — JVM
  - Operating System independence.
  - Safe, well defined, operating environment.
  - Portability
  - Performance Issues
- Garbage Collection
  - 内存的分配 —— 指针 —— 释放
  - 不用再释放内存
  - C&C++ 程序员的责任 —— memory leak
  - 使用系统层次的线程追踪内存的分配
  - 自动运行
  - 同JVM同步
- 代码的安全 — Security

JRE: Java运行时环境

- 装载 —— load
  - 隔离——名字空间(namespace)——private
  - local & network—— 调用
  - 完全装载——内存分配——搜索表
  - 都在执行前
- 认证 —— verify
  - 代码的专一性
  - 对系统完整性的破坏
  - 堆栈的上溢和下溢
  - 参数、数据类型、对象的引用
  - 无非法数据转换
- 执行 —— execute

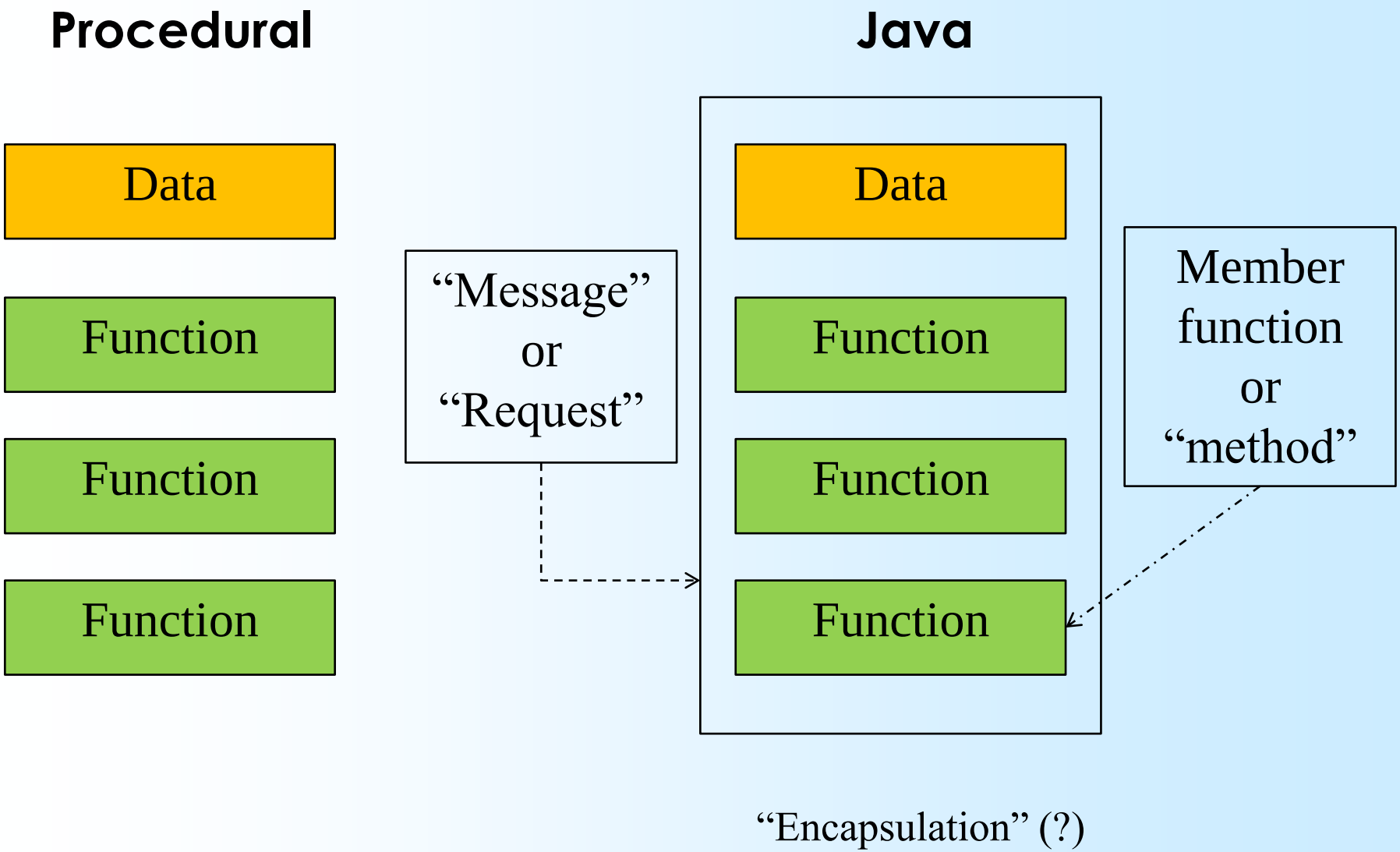
Objects, Objects, Objects

-
-
-



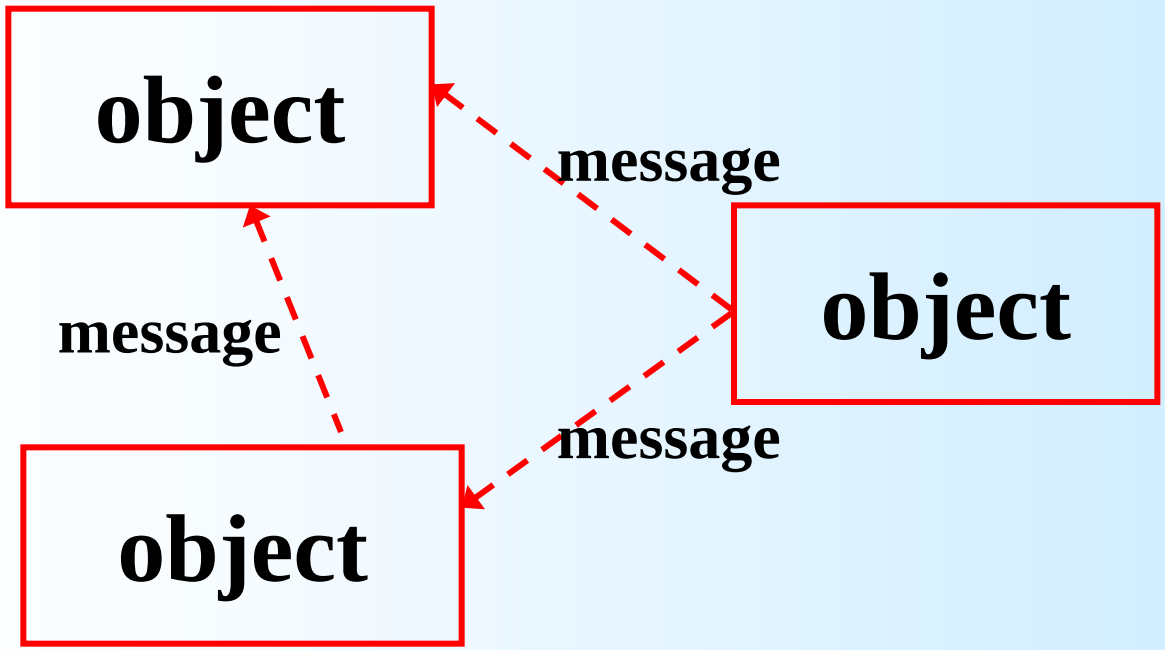
ile.

Procedural vs. Java



"Encapsulation" (?)

Object-Oriented programs



- Made up of objects sending messages to each other's interfaces

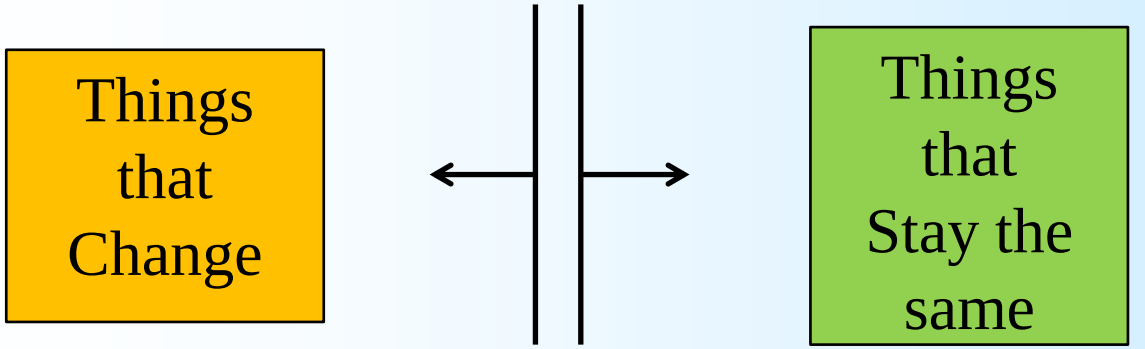
面相对象编程描述的是对象之间的相互作用

OOP Development

- Object-Oriented *Components*
  - Formalized in Java EE - Java Beans
- Incremental Development
- Unified Modeling Language (UML)

OOP Design

- Most designs can be simplified by adding another level of indirection/abstraction



- In the design
- At compile-time
- At run-time
- In the environment
- Etc

- Hard part decomposing system into objects
- What interfaces do you need?

Features & Benefits

- Platform Independence
- High Performance
- Easy to Learn
- Standards-Based
- Worldwide Prevalence
- Consistent Runtime Environments
- Optimized for Embedded
- High-performance, Portable Applications
- Proven Security Model
- Java Platform, Enterprise Edition

Java vs. C++

- Similar syntax/control structures.
- No preprocessor or include files.
- No pointers
- No global variables
- No **`struct`** or **`union`** types.
- All primitive types have well defined size.

More Java vs. C++

- No operator overloading.
- Single inheritance only
  - there is another approach used - interfaces.
- Error handling is well defined (and somewhat enforced!).
- No memory leaks!

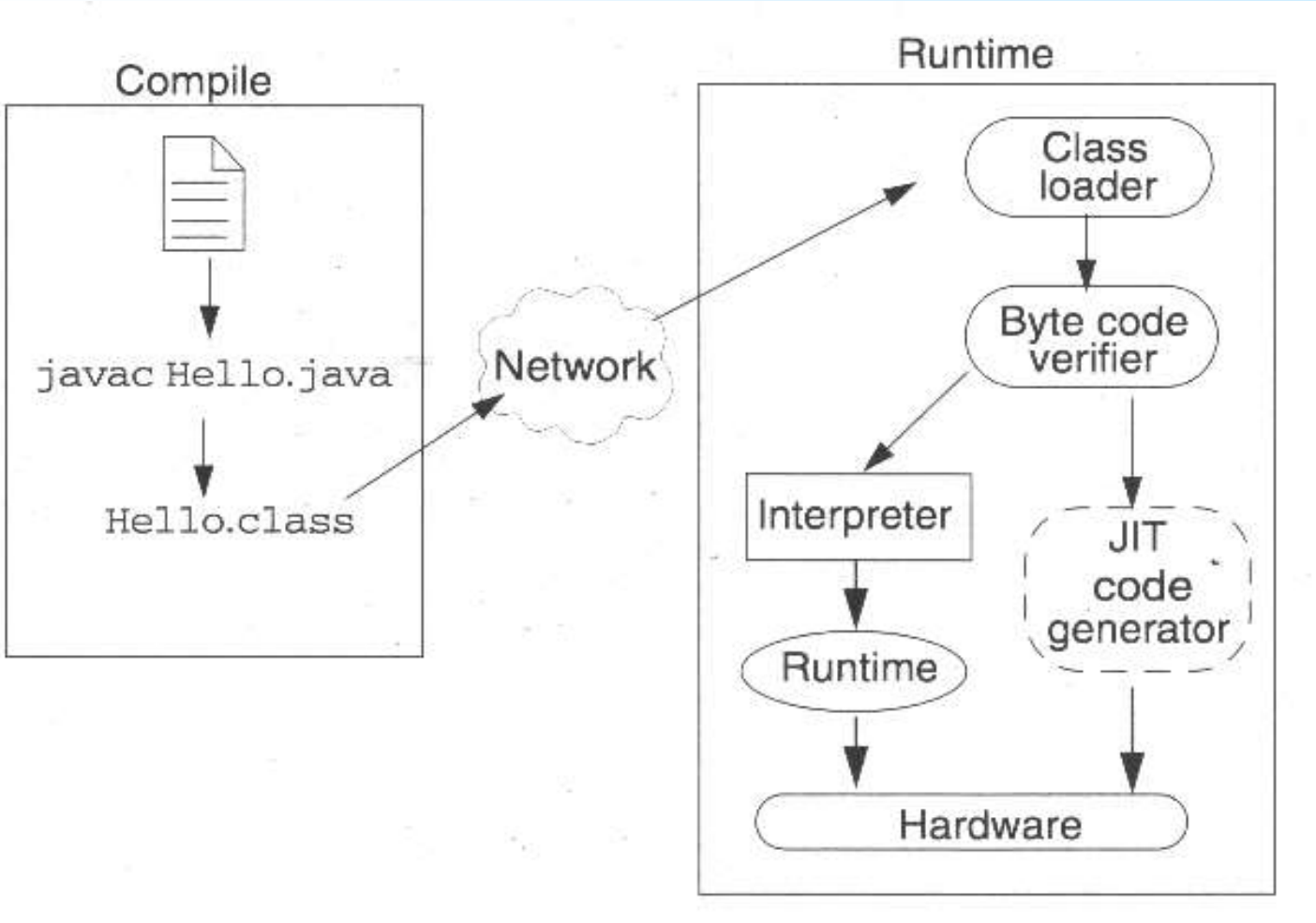
and more…

- Safety designed into the language and VM
  - bytecode verification
  - array access bounds checking
  - security manager/ security policies

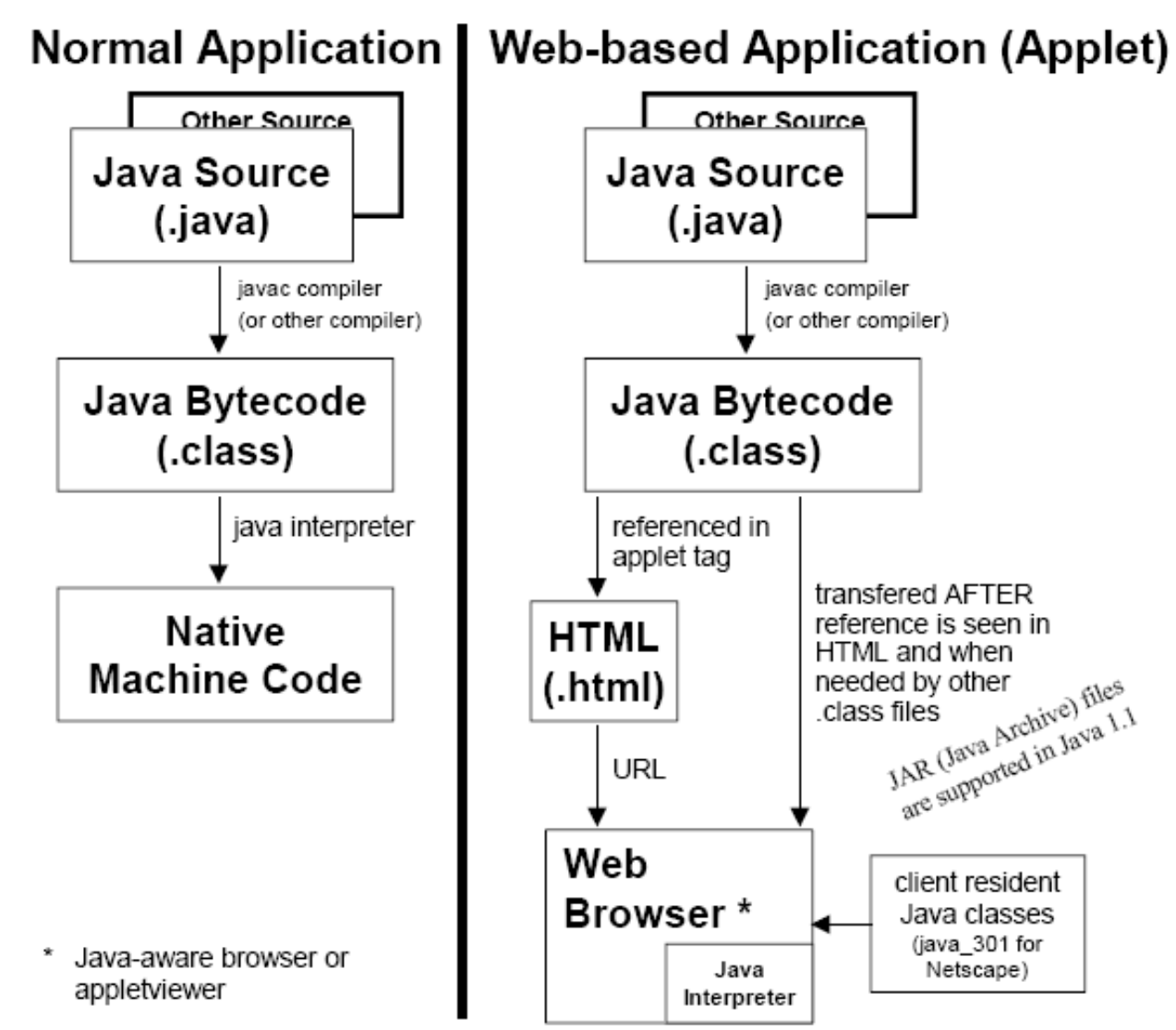
Performance Issues

- Using a VM is slower than compiling to *native* instructions.
  - JIT compilers convert Java Bytecode to machine language.
- Safety/Security slow things down
  - all array accesses require bounds check
  - Many I/O operations require security checks

The big picture



The big picture (cont.)



Java Strengths

- Simplicity (the language itself)
- Networking
- Object model
  - Graphical User Interface (GUI) programming
  - Large and Very Large systems.
  - Portable libraries

SDK Tools

- **`javac`**: the Java compiler.
  - Reads source code and generates bytecode.
- **`java`**: the Java interpreter
  - Runs bytecode.
- **`jar`**: Java Archive utility
- **`javadoc`**: create documentation from code.
- **`jdb`**: Java debugger (command line).
- There are others…**`appletviewer, javah, javap, extcheck,`**…

The Java Compiler

- Usage: **`javac filename.java`**
  - You can also do: **`javac *.java`**
  - Creates **`filename.class`** (if things work)

  - Use "-g" to compile for use with the debugger.

The Java Interpreter

- Usage: **`java classname`**
  - You tell the interpreter a class to run, not a file to run!
    - It uses the CLASSPATH to find the named class.
    - The named class should have a method with prototype like:
      - **`public static void main()`**

jar

- Like Unix tar command.
- Used to create (and extract from) an archive file:
  - collection of files.
  - compressed.
- Java can find classes (bytecode) that are stored in jar files.

jar usage

- To extract files:

```
jar xf filename.jar
```

- To list files:

```
jar tf filename.jar
```

- To create and archive:

```
jar cf filename.jar file1 file2 dir1 dir2 …
```

javadoc

- Creates documentation from properly commented Java source code.
- The output of javadoc includes HTML files in the same format as the Java SDK documentation.
  - we all need to get used to this format…
  - learning to find and understand the documentation on classes/methods is 1/2 of learning Java!

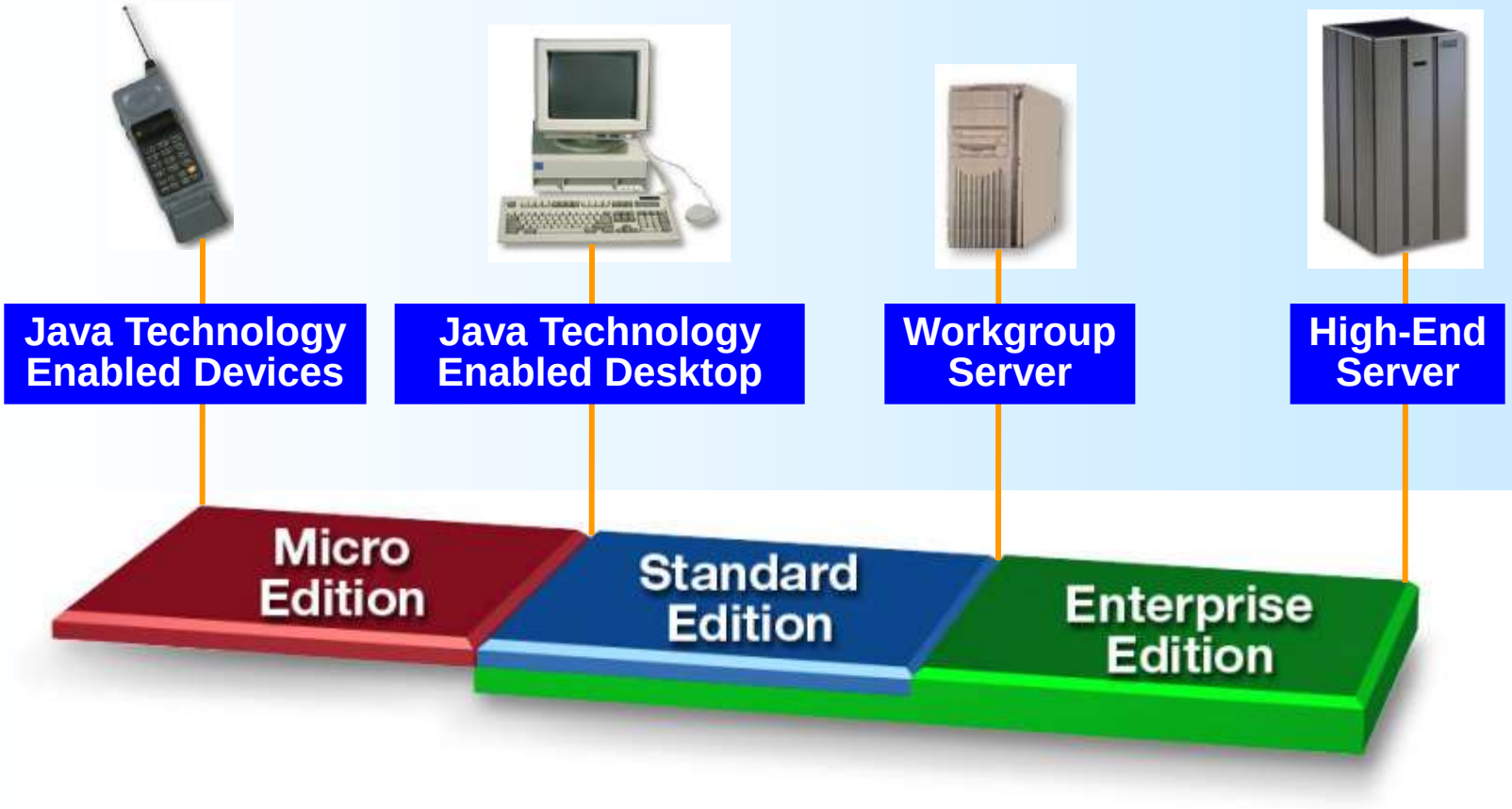
Using the tools

- We will look at these and other tools in more detail later…
- For now (HW1) we just need to be able to compile and execute a simple Java class.

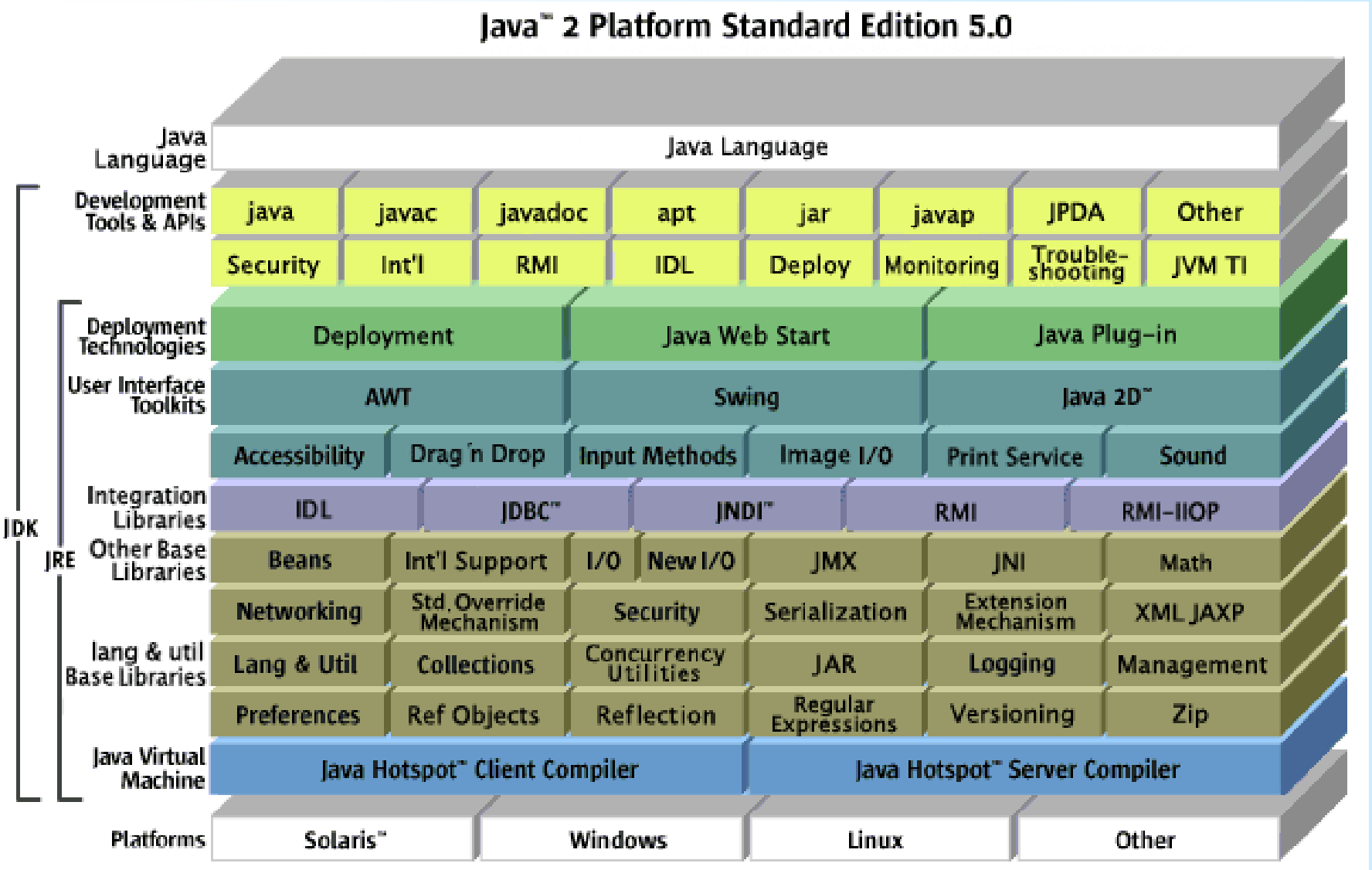
Java Versions

- **Java™ Platform, Standard Edition (Java SE)**
- **Java™ Platform, Enterprise Edition (Java EE)**
- **Java™ Platform, Micro Edition (Java ME)**

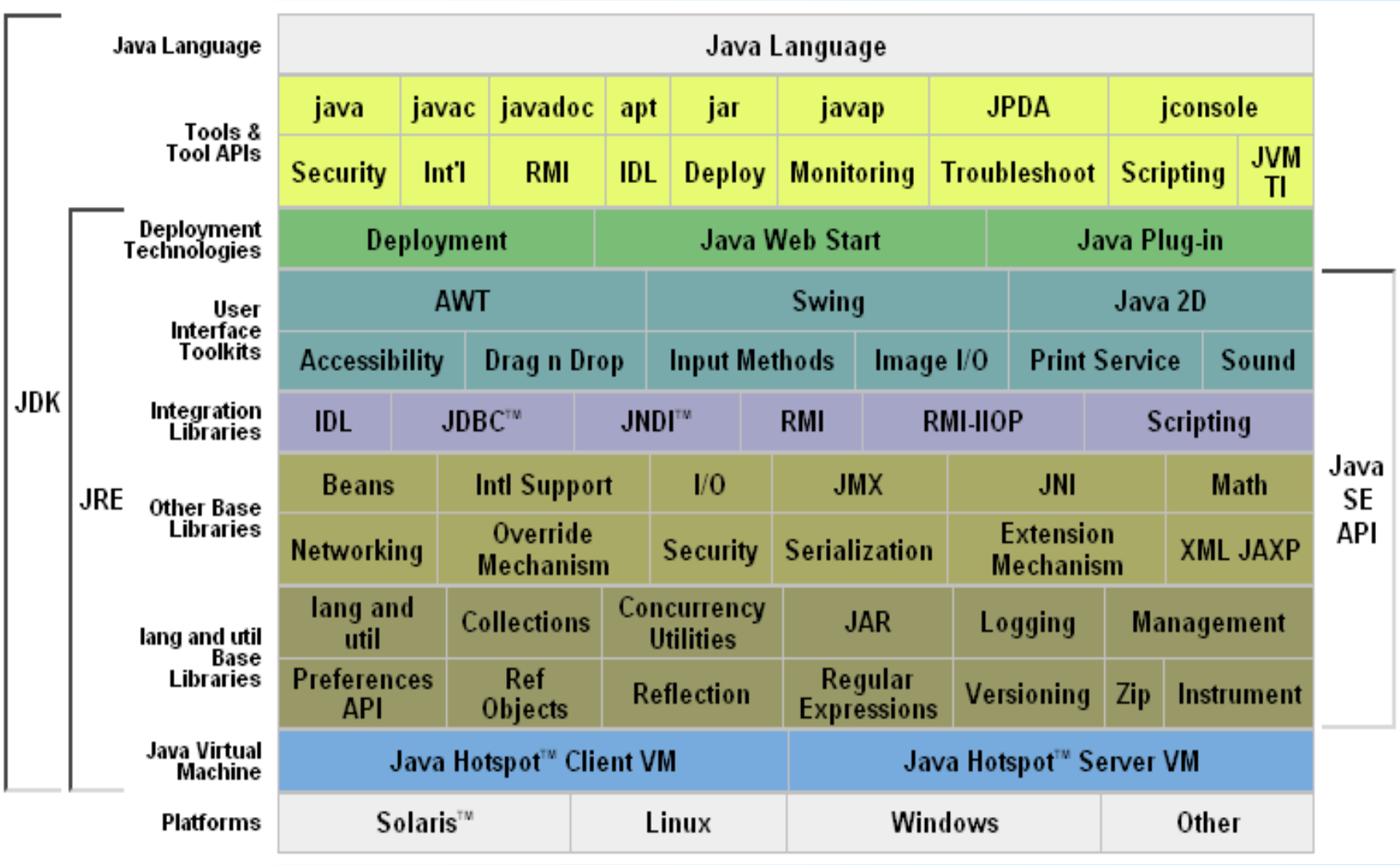
The Java™ Platform



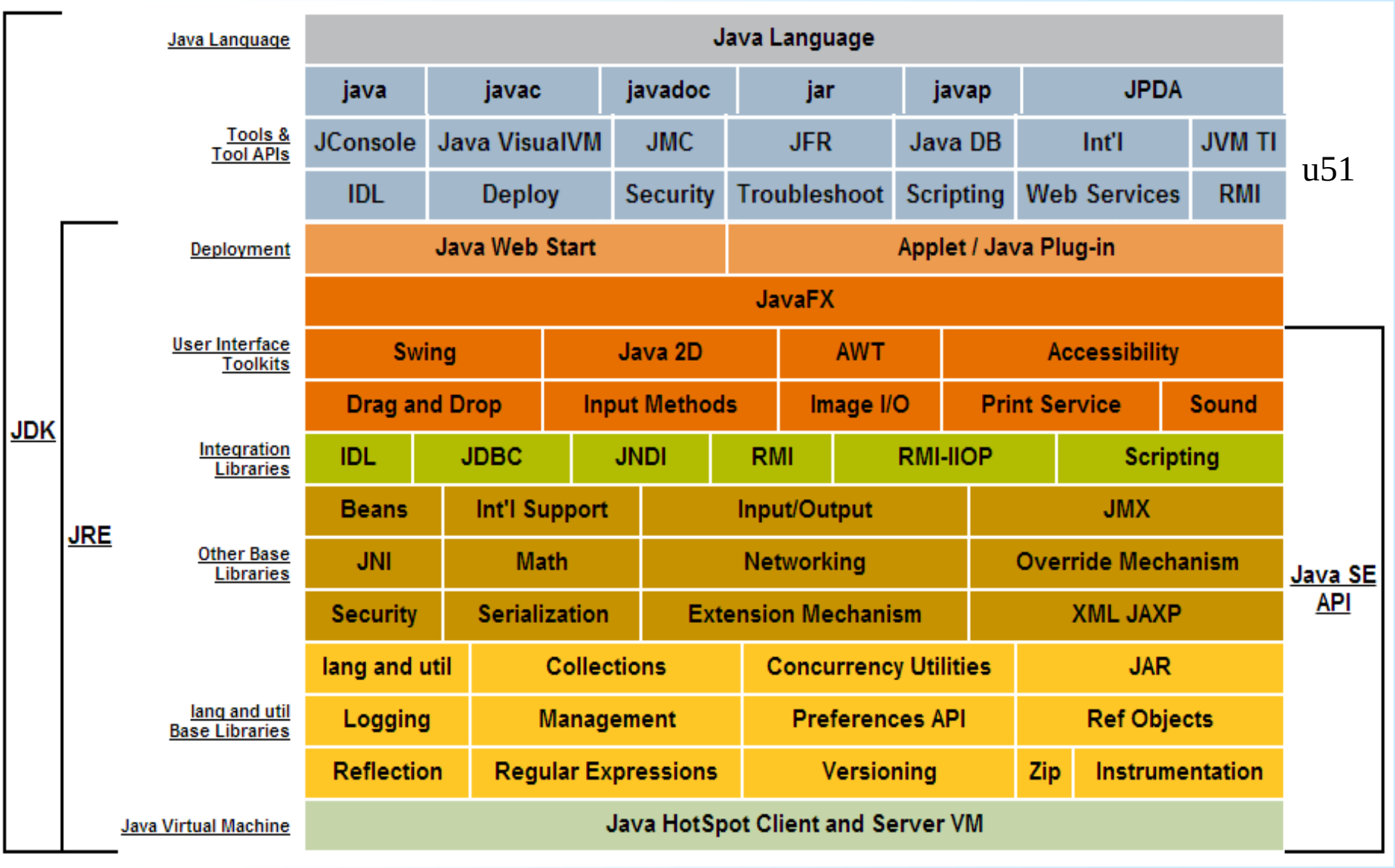
Java™ SE 5


Java™ 2 Platform Standard Edition 5.0
Java Language
Java Language
Development Tools & APIs
java
javac
javadoc
apt
jar
javap
JPDA
Other
Security
Int'l
RMI
IDL
Deploy
Monitoring
Trouble-shooting
JVM TI
Deployment Technologies
Deployment
Java Web Start
Java Plug-in
User Interface Toolkits
AWT
Swing
Java 2D™
Accessibility
Drag 'n Drop
Input Methods
Image I/O
Print Service
Sound
Integration Libraries
IDL
JDBC™
JNDI™
RMI
RMI-IIOP
Other Base Libraries
Beans
Int'l Support
I/O
New I/O
JMX
JNI
Math
Networking
Std. Override Mechanism
Security
Serialization
Extension Mechanism
XML JAXP
lang & util Base Libraries
Lang & Util
Collections
Concurrency Utilities
JAR
Logging
Management
Preferences
Ref Objects
Reflection
Regular Expressions
Versioning
Zip
Java Virtual Machine
Java Hotspot™ Client Compiler
Java Hotspot™ Server Compiler
Platforms
Solaris™
Windows
Linux
Other
JDK
JRE

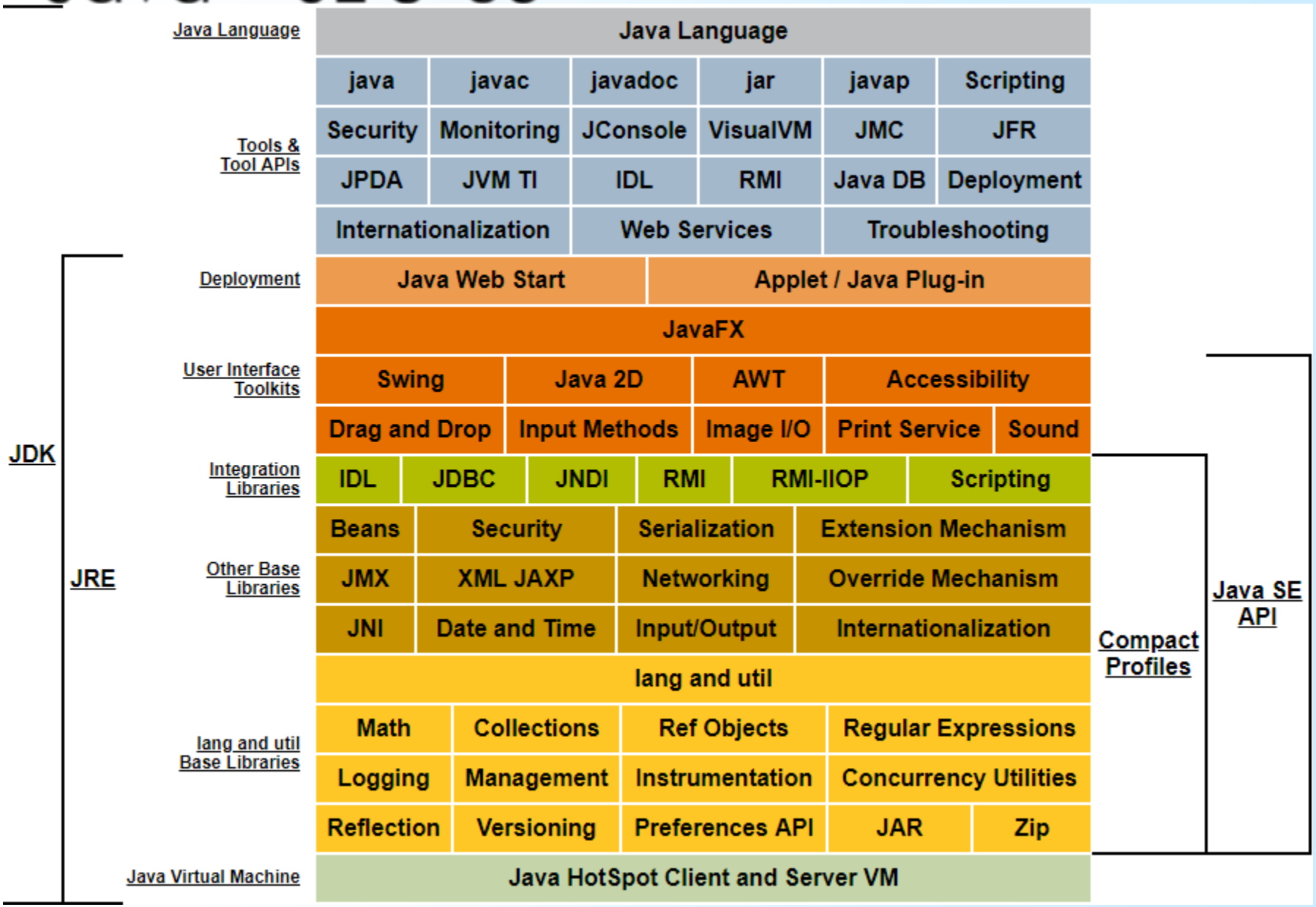
Java™ SE 6


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

Java™ SE 7 –u01


Java Language
Java Language
java
javac
javadoc
jar
javap
JPDA
Tools & Tool APIs
JConsole
Java VisualVM
JMC
JFR
Java DB
Int'l
JVM TI
IDL
Deploy
Security
Troubleshoot
Scripting
Web Services
RMI
u51
Deployment
Java Web Start
Applet / Java Plug-in
JavaFX
User Interface Toolkits
Swing
Java 2D
AWT
Accessibility
Drag and Drop
Input Methods
Image I/O
Print Service
Sound
Integration Libraries
IDL
JDBC
JNDI
RMI
RMI-IIOP
Scripting
Other Base Libraries
Beans
Int'l Support
Input/Output
JMX
JNI
Math
Networking
Override Mechanism
Security
Serialization
Extension Mechanism
XML JAXP
lang and util Base Libraries
lang and util
Collections
Concurrency Utilities
JAR
Logging
Management
Preferences API
Ref Objects
Reflection
Regular Expressions
Versioning
Zip
Instrumentation
Java Virtual Machine
Java HotSpot Client and Server VM
JDK
JRE
Java SE API

Java™ SE 8–u5


Java Language
Java Language
java
javac
javadoc
jar
javap
Scripting
Tools & Tool APIs
Security
Monitoring
JConsole
VisualVM
JMC
JFR
JPDA
JVM TI
IDL
RMI
Java DB
Deployment
Internationalization
Web Services
Troubleshooting
Deployment
Java Web Start
Applet / Java Plug-in
JavaFX
User Interface Toolkits
Swing
Java 2D
AWT
Accessibility
Drag and Drop
Input Methods
Image I/O
Print Service
Sound
Integration Libraries
IDL
JDBC
JNDI
RMI
RMI-IIOP
Scripting
Other Base Libraries
Beans
Security
Serialization
Extension Mechanism
JMX
XML JAXP
Networking
Override Mechanism
JNI
Date and Time
Input/Output
Internationalization
lang and util
lang and util Base Libraries
Math
Collections
Ref Objects
Regular Expressions
Logging
Management
Instrumentation
Concurrency Utilities
Reflection
Versioning
Preferences API
JAR
Zip
Java Virtual Machine
Java HotSpot Client and Server VM
JDK
JRE
Compact Profiles
Java SE API

Getting, installing and testing the Java SDK

Installing SDK

- Make sure you have room! (150MB).
- Download the appropriate file from java.sun.com:
  - JDK Standard Edition 1.5.0 (or higher)
- Run the file (start the installation). Install somewhere easy to access, I liked **`D:\Java\jdk1.7.0_51`**, but now I just use the default directory.
- Download and install documentation
  - Java™ SDK Standard Edition documentation.
  - HTML files.

PATH Environment Variable

- To run the tools (compiler, JVM, etc) from the DOS command line you need to add the location of the tools to your PATH.
  - Follow the instructions found in the documentation at:
    `http://docs.oracle.com/javase/7/docs/webnotes/install/windows/jdk-installation-windows.html`

http://docs.oracle.com/javase/7/docs/webnotes/install/index.html

CLASSPATH environment variable

- The Java compiler and VM use the CLASSPATH environment variable to decide where to look for class definitions (Java code).
- For now it's enough the Java will look in the "current working directory".
- If your CLASSPATH is already set to something, you may need to add "." to it.

  ("." is *the* ***current directory***).
- Different directories is separated by "**;**"

Testing things

- Probably you need to reboot to have any changes to the PATH take effect. (Win2k, WinXP, Win7 do not need to reboot)
- Open a MS-DOS prompt window (command prompt for Win2K and up).
- Type "**`java -version`**". If you get "unknown command" or something like that, your PATH is not right…

Testing a Java Program

- In general, you do the following:
  - create a folder to hold the java code.
  - use an editor (notepad will work, but other editors will work better – check out editplus at www.editplus.com) to create a Java program.
  - compile the program (using **`javac`**)
  - run the program (using **`java`**)

Sun "first cup of Java" tutorial

- on the web at:
  http://download.oracle.com/javase/tutorial/getStarted/cupojava/win32.html

- Goes through all the steps, explains how to resolve common problems

First Cup of Coffee

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

Be Careful When You Type 

Type all code, commands, and file names exactly as shown. Both the compiler (javac) and launcher tool (java) are *case-sensitive*, so you must capitalize consistently.

HelloWorld ≠ helloworld

Naming Conventions(协定)

Part of JavaSoft programming standard

- Words run together, no underscores
- Intermediate words capitalized
- Classes: first letter capitalized
- Methods and variables (including references): first letter lowercase
- Constants: all caps with underscores to separate words (like C).

Homework

- Get JDK installed and working.
- Modify a simple sample program.
- Use tag @author follows your name, Student No. and Email address.
- Compile and run the program.
- No need to submit.

Next

- Introduction to Objects
- Everything is an Object

- Language Basics