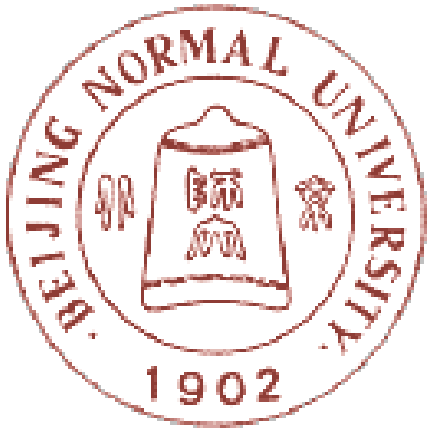
# Java概述




宋杰 硕士研究生
songjiesdnu@163.com

北京师范大学教育技术学院
现代教育技术研究所

# Java 4-ever


土豆网
优酷
- You can do fine tuning and balancing
which you can't do with Microsoft.

# Java is everwhere

What will be possible tomorrow ?
you decide it .
Because java is everywhere.

# 主要内容

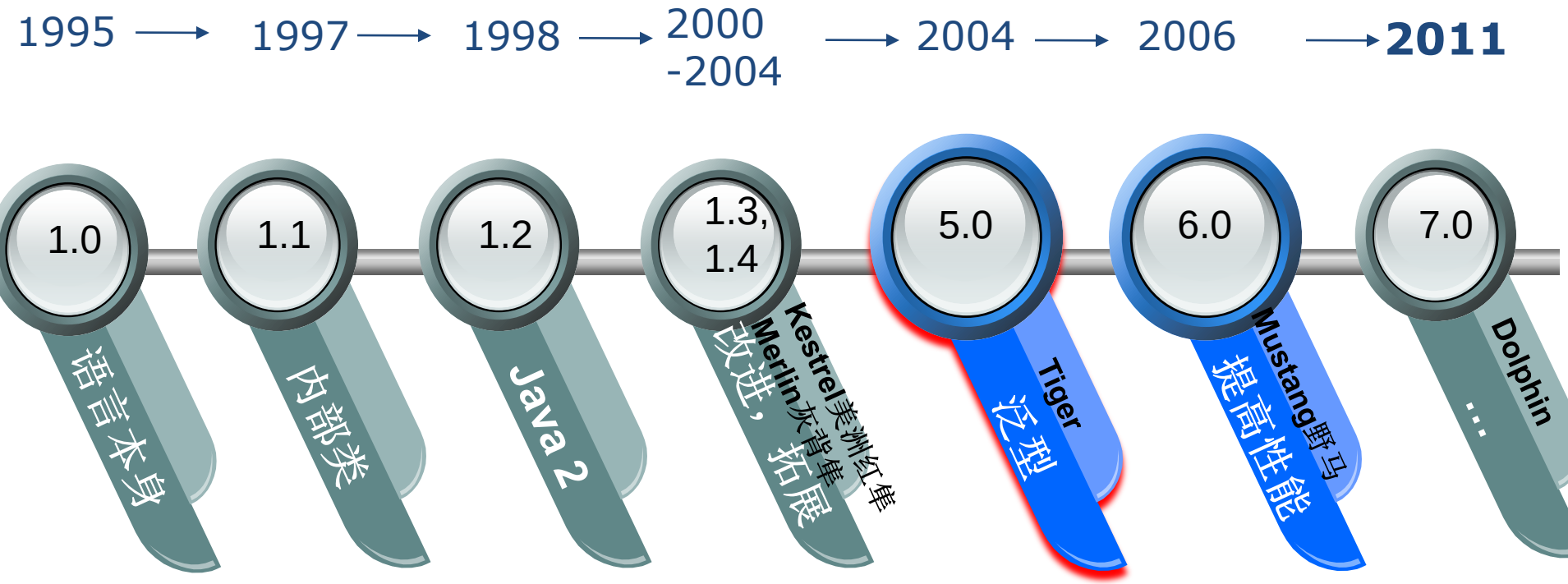
1. **Java的历史与发展**
2. **Java语言概貌**
3. **Java开发环境及开发工具**
4. **Java的程序示例**
   - Application程序
   - Applet程序
   - Servlet程序
   - Jsp程序

# 1 JAVA的历史与发展

# Java的历史与发展

- **前身：Oak (James Gosling 1990)**
- **Java大事记**
  - 90年，Sun公司开始Green工程
  - 93年，WWW席卷全球
  - 94年，开始定位于Internet
  - 95.1，Java命名

# Java版本更新




网址 (查询新动向) ~~http://java.sun.com~~

http://www.oracle.com/technetwork/java/index.html

**课外阅读**

从Oracle收购Sun 公司谈起 http://caisu.blog.sohu.com/114902355.html

纪念SUN的历史时刻 http://caisu.blog.sohu.com/115299503.html

# 2 JAVA语言概貌

# Java语言的特点

- **Simple (易学，自动内存管理，简化重载，去掉指针及C++中一些不是绝对必要的功能)**
  **Java语言与C++语言的风格极为相似，但却比C++语言简单得多，去掉了C++语言中容易引发程序错误的地方。**

- **Object-Oriented (纯面向对象语言，程序代码以类的形式组织，由类来定义对象的各种状态和行为)**
  **具备面向对象的三大特点：封装、继承、多态。**

- **Distributed (丰富的网络编程功能-轻松处理TCP/IP，通过URL访问远地资源；字节码可来自网络)**

# Java语言的特点（续）

- **Interpreted**
  - Java是解释型的，但Java通过预先将源代码编译为接近于机器指令的字节码，有效地克服了传统解释型语言的性能瓶颈，同时又保持了解释型语言的可移植性。
  - Java解释器能直接在任何机器上执行Java字节码

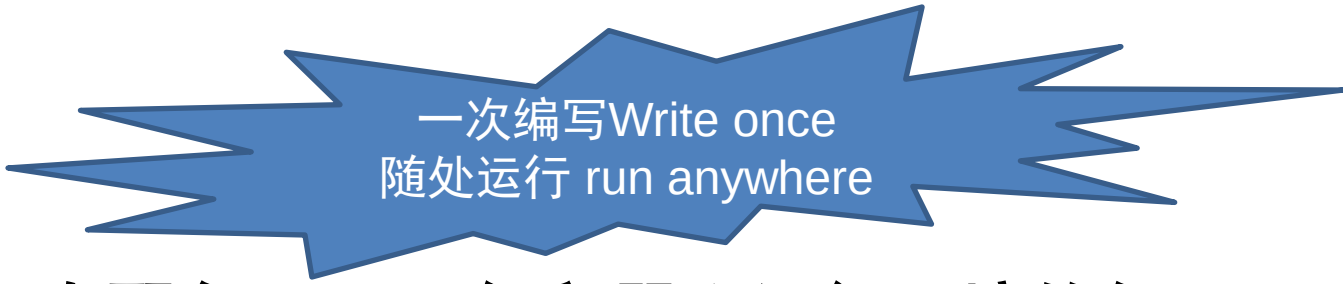
- **Robust**
  - 静、动态检查，排除出现错误的条件，异常处理，取消指针，内存保护。
  - Java语言系统仔细检查对内存的每次访问，确认它是合法的，不致引起任何问题，如果出现某种意料之外的事，系统不会崩溃，而是把该例外抛弃。
  - 取消指针，从而杜绝了对内存的非法访问。

# Java语言的特点（续）

- **Secure(适用于网络/分布式运算环境，确保建立无病毒且不会被侵入的系统。内存分配及布局由Java运行系统决定，字节码加密传输，客户端校验)**
  - Java程序分两种，Application在本地执行，而Applet小程序可在网上发布，但需要浏览器执行，为保证从远端下载的小程序不会对用户造成危害，Java引入了砂盒( sandbox )安全模型，限制小程序访问本地资源。

- **architecture-neutral(让Java应用程序能够在网络上任何地方执行，字节代码 - 平台无关性、完全统一的语言版本 - 实现无关性， 访问底层操作系统功能的扩展类库 – 不依赖于具体系统)**
  - Java语言源程序被编译成一种高层次的与机器无关的以及结构中立的字节码语言，该格式语言在Java虚拟机上运行，只要有Java语言运行系统的机器都能执行这种中间代码。

# Java语言的特点（续）




- **Portable**

  **Java程序可在配备了Java解释器和运行环境的任何机器上运行，这成为Java软件便于移植的良好基础。**

- **high-performance(字节码-> 目标代码)**

  **Java开发者设计了just in time编译器(也叫代码生成器)，这种编译器可以在运行时把Java的字节码翻译成特定的机器代码，提高了其高性能。**

# Java语言的特点（续）

- multi-threaded(支持多任务。在语言级嵌入了对并发控制的功能 - 多线程控制，大大简化了多线程应用程序的开发)

  使应用程序可以并行执行，在一个程序里可同时执行多个小任务，同步机制保证了对共享数据的正确操作。

# Java语言的特点（续）

- **Dynamic(可动态增加和修改类库内容, 是面向对象设计的延伸。 Java的基本组成单元是类，而Java的类又是运行时动态装载的。可以在分布环境中动态地维护应用程序和类库的一致性。更能适应时刻变化的环境，Java不会因程序库的更新，而必须重新编译程序)**

Java语言的设计使它适合于一个不断发展的环境，在类库中可以自由地加入新的方法和实例变量而不会影响用户程序的执行，它同时允许程序动态地装入运行过程中所需要的类。

# Java是什么


- A simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high - performance, multi-threaded, dynamic language.

“作为一种计算机语言，Java的广告词实在有点夸大其辞。
然而，Java确实是一种优秀的程序设计语言。
对于一个名符其实的程序设计人员来说，使用Java无疑是一个好的选择。”

小故事：M$的J++

# Java语言与C\C++语言的比较

- **全局变量**

  **在Java语言程序中，不能在所有类之外定义全局变量，只能通过在一个类中定义公用、静态的变量来实现一个全局变量，这样在Java语言对全局变量进行了更好的封装。而在C\C++语言中依赖于不加封装的全局变量常常造成系统的崩溃。**

- **goto**

  **Java语言不支持C\C++语言中的goto语句，而是通过异常处理语句try、catch、finally等来代替C\ C++语言中用goto来处理遇到错误时的跳转情况，使程序更可读且更结构化。**

# Java语言与C\C++语言的比较

- 指针

  指针是C\C++语言中最灵活，也是最容易产生错误的数据类型。Java语言对指针进行了完全的控制，程序员不能直接进行任何指针操作。同时，数组作为类在Java语言中实现，它很好地解决了数组访问越界这一C\C++语言中不作检查的错误。

- 数据类型的支持

  在C\C++语言中，对于不同的平台，编译器对于简单数据类型如int,float等分别分配不同长度的字节数，但在Java语言中，对于这些数据类型总是分配固定长度的位数，从而保证了Java语言的平台无关性。

# Java语言与C\C++语言的比较


- 内存管理

  在C 语言中，程序员通过库函数malloc()和free()来分配和释放内存， C++语言中则通过运算符new和delete来分配和释放内存。而在Java语言中，所有的数据结构都是对象，通过运算符new为它们分配内存堆。通过new得到对象的处理权，而实际分配给对象的内存可能随程序的运行而改变，Java运行系统对此自动进行管理并且进行垃圾收集，有效地防止了由于程序员的误操作而导致的错误，并且更好地利用了系统资源。

# Java语言与C\C++语言的比较

- 类型转换

  在C\C++语言中，可以通过指针进行任意的类型转换，常常带来不安全性，而Java语言中，运行时系统对对象的处理要进行类型相容性检查，以防止不安全的转换。

- 结构与联合

  C\C++语言中的结构体与共用体中的所有成员均为公有，这就带来了安全性问题。 Java语言中不包含结构体与共用体，所有的内容都封装在类中。

# Java语言与C\C++语言的比较

- 宏定义

  C\C++语言中用宏定义来实现的代码给程序的可读性带来了困难。在 Java语言中不支持宏，它通过关键字final来声明一个常量，以实现宏定义中广泛使用的常量定义。

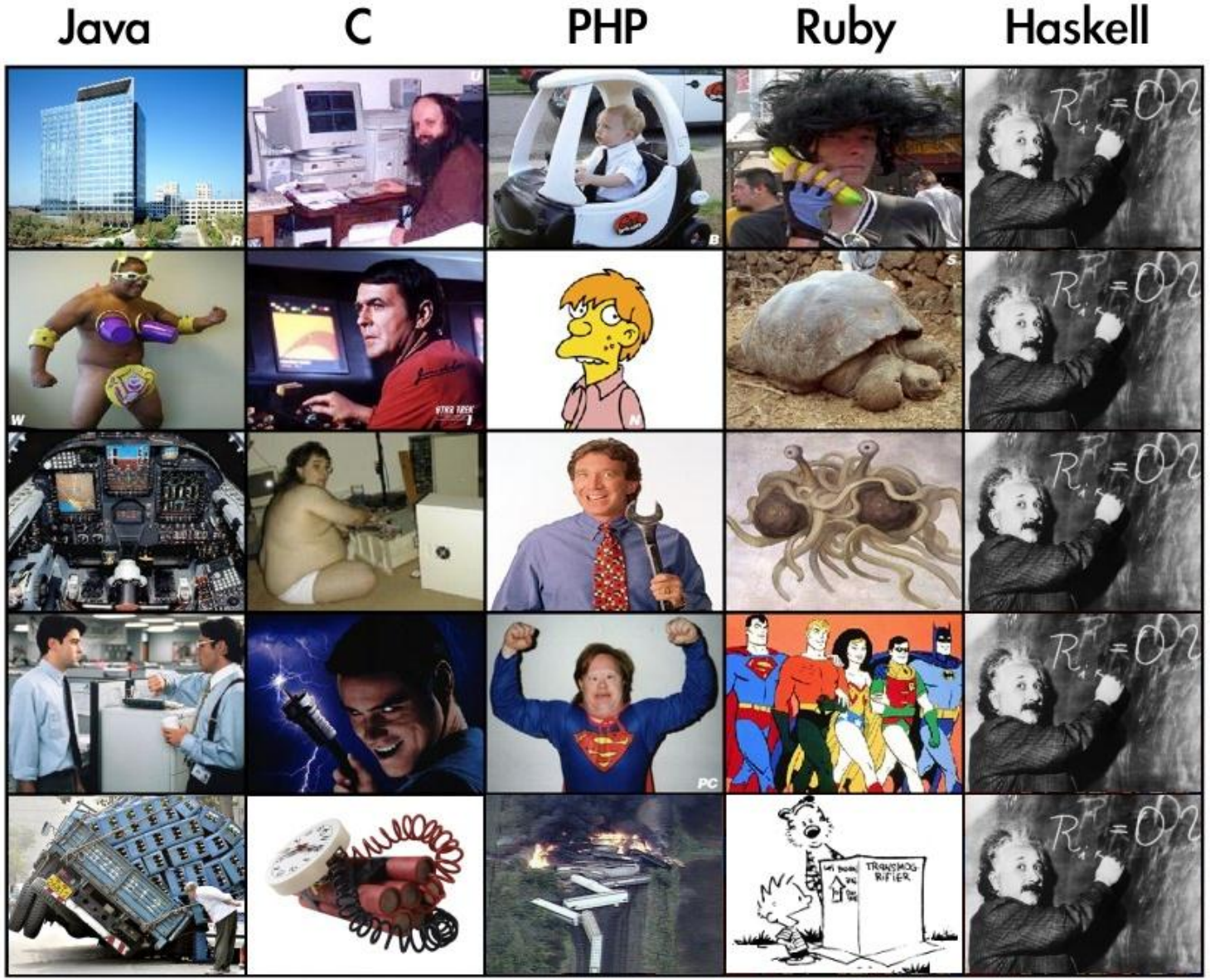
- 头文件

  C\C++语言中用头文件来声明全局变量、库函数等，在大的系统中，维护这些头文件很困难。而Java不支持头文件， Java语言中用import语句与其他类进行通信，以便使用它们的方法。

# Java语言的缺点

- **缺点**
  - 解释型语言，对于桌面程序来说运行速度慢（但在1.2之后性能大为改观）

# 不同编程语言粉丝眼中的Java



# 另1：不同IT员工如何看待对方

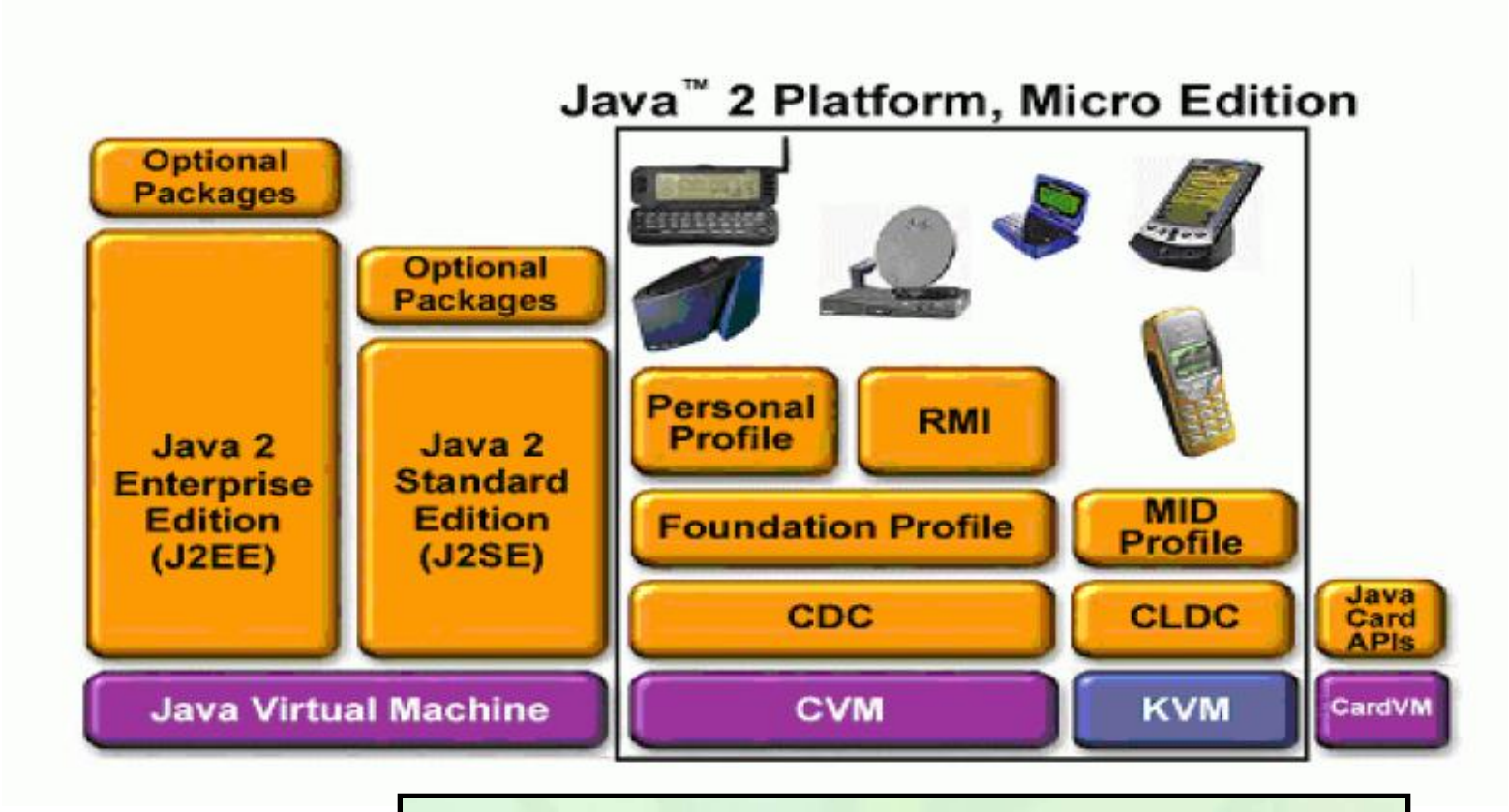
| 市场 | 产品 | 技术 | As seen by… |
| --- | --- | --- | --- |
|  |  |  | 市场 |
|  |  |  | 产品 |
|  |  |  | 技术 |
|  |  |  | 老板 |

# Java是什么

- **Java是面向对象的程序设计语言**
  - Java是Internet上的世界语。
  - Java是最佳的网络应用开发语言。
- **Java是环境，是平台**
- **Java是产业**
- **……**

**Java已经逐步从一种单纯的计算机高级编程语言发展为一种重要的Internet平台，并进而引发、带动了Java产业的发展和壮大，成为当今计算机业界不可忽视的力量和重要 的发展潮流与方向 ！**

# Java技术体系




1999年

- 企业计算：Java 2 Enterprise Edition J2EE
- 桌面计算：Java 2 Standard Edition J2SE
- 嵌入计算：Java 2 Micro Edition J2ME

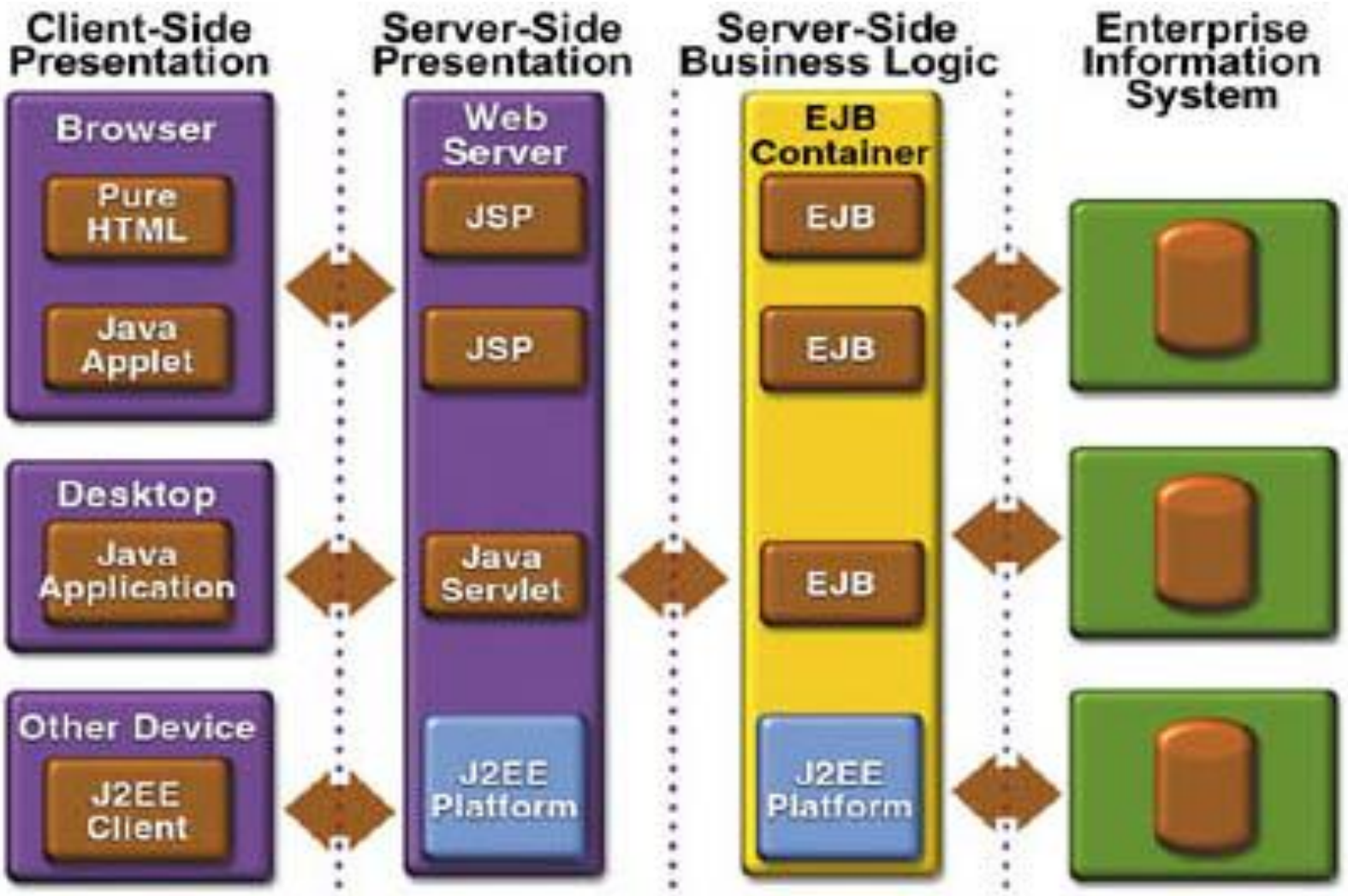
# JSE

- (1) 以Web为中心的客户端或服务器端的软件开发
- (2) JSE 的实现： Java Software Development Kit (SDK), Standard Edition + Java Runtime Environment, Standard Edition
- 以前称为 JDK 1.2，JKD1.3，JDK1.4
- 从JDK1.5起，改称为JDK 5或J2SE5,最终弃用2,改为Java SE5
- 目前最新版本为Java SE 7

# JME & JEE

- **JME**
  - (1) 针对消费类的电子设备如蜂窝电话、数字置顶盒、汽车导航系统等
  - (2) 语言精简、运行环境高度优化
- **JEE**
  - (1) 开发企业级和服务器端的应用
  - (2) JSE+Enterprice JavaBeans（EJB）,Java Servlets API + Java Server Pages（JSP）

# J2EE Application Model



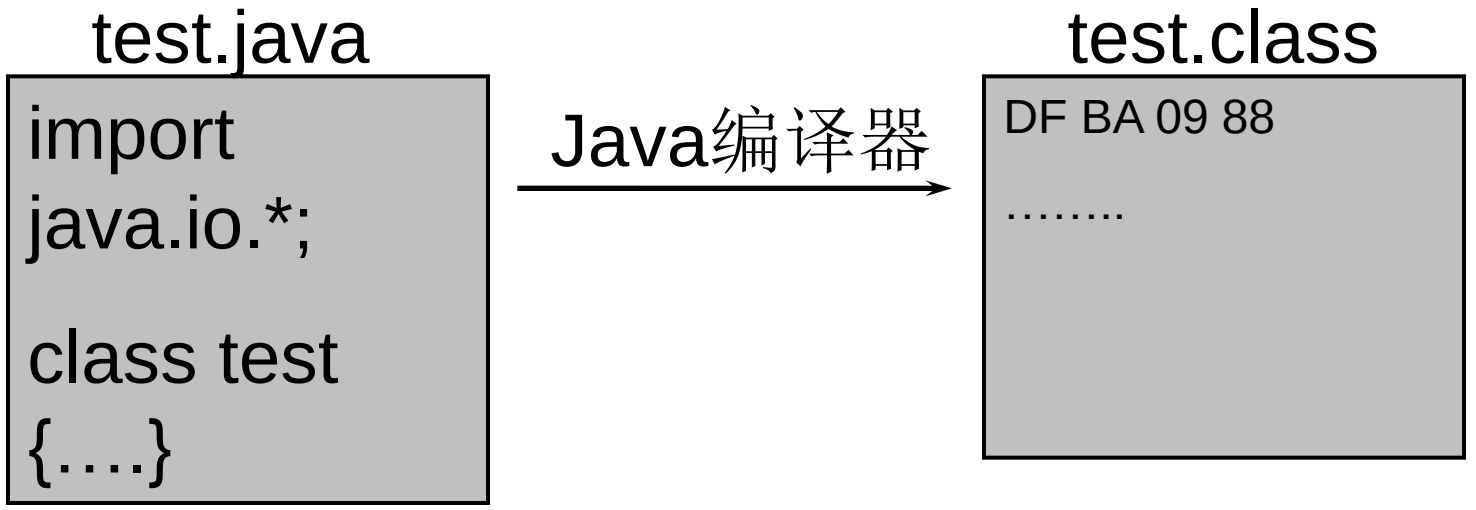
# Java技术体系

- JEE:除微软以外的软件公司都各自有自己的J2EE产品，如BEA的WebLogic, IBM的Websphere, Oracle的Application Server, CA, HP。体现IT行业两种技术思想的碰撞：封闭技术（微软）和开放标准
- JSE:由于跨平台的优势，许多桌面软件已经用Java来开发，尤其是在图形用户界面设计方面
- JME: 充分占领了消费电子市场，占世界市场份额95%的厂商已经签约采用JavaCard技术，手机上的应用开发平台已经广泛应用（Nokia 和Motolora）;数字电视机顶盒上的应用开发平台标准已经采用Java技术（欧洲MHP标准和美国ATSC标准）

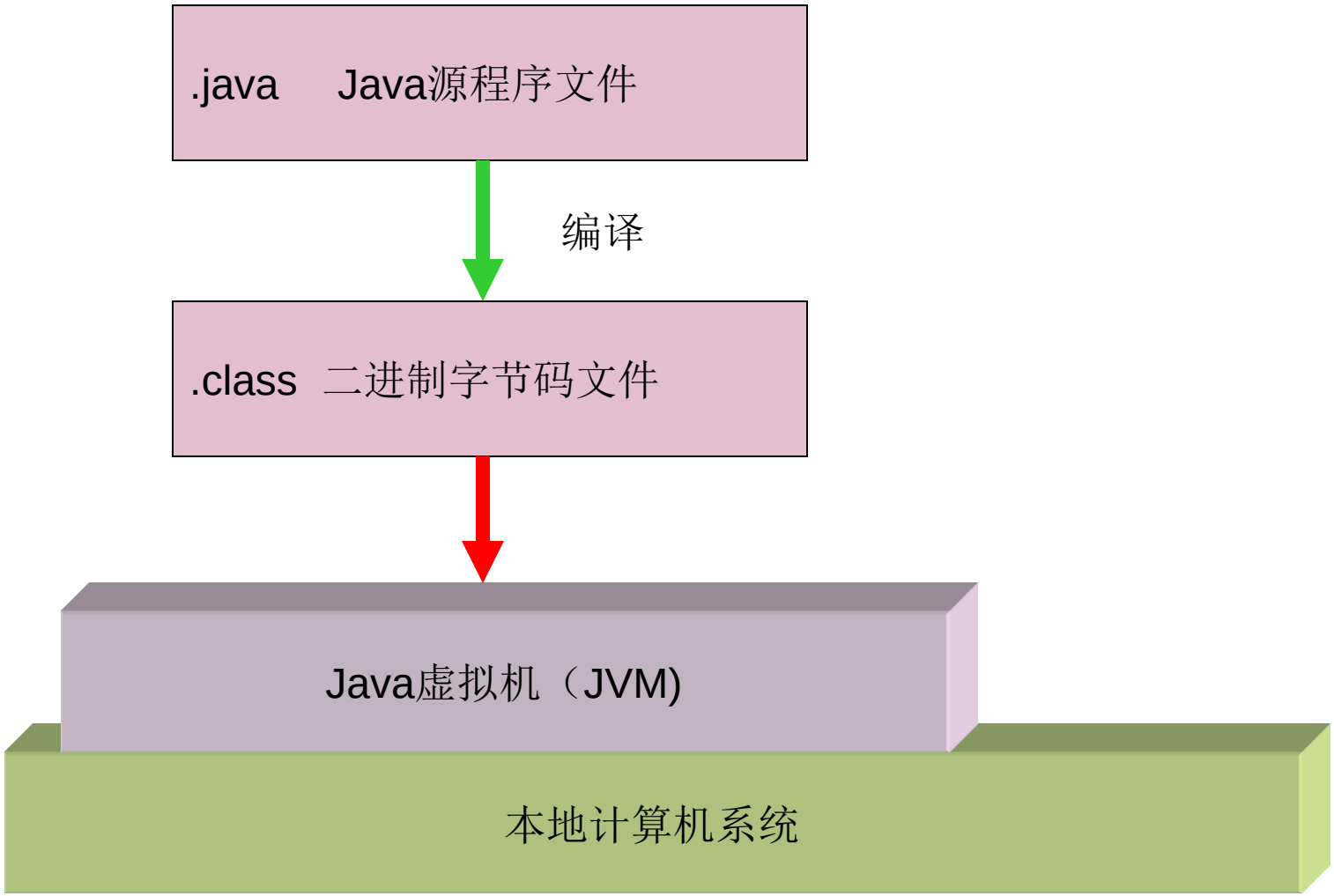
# 3 JAVA开发环境及开发工具

# Java的工作方式

- **文件类型**
  - .java　Java源文件
  - .class　二进制字节码文件
- **编译过程**



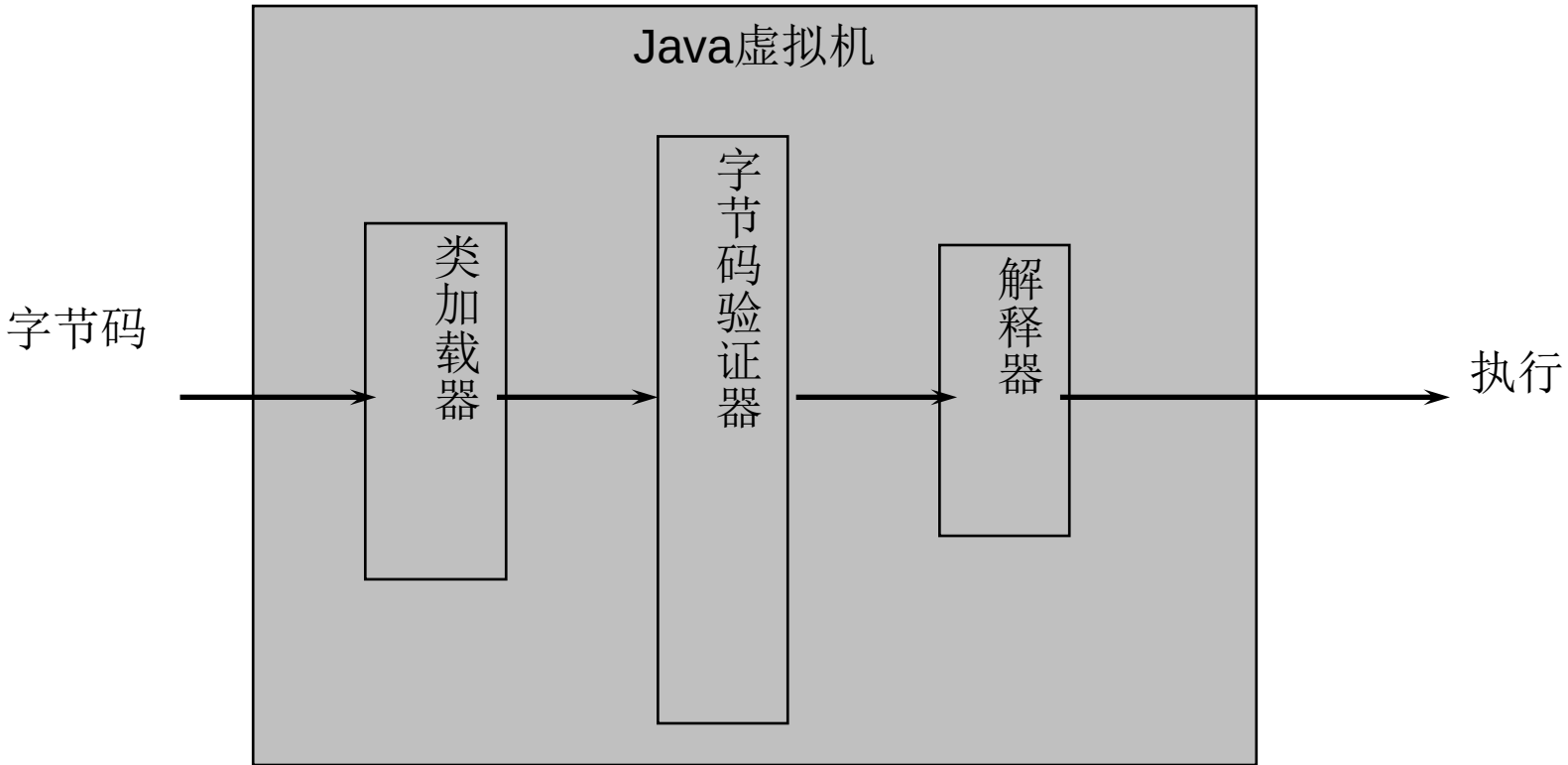
# Java的工作方式



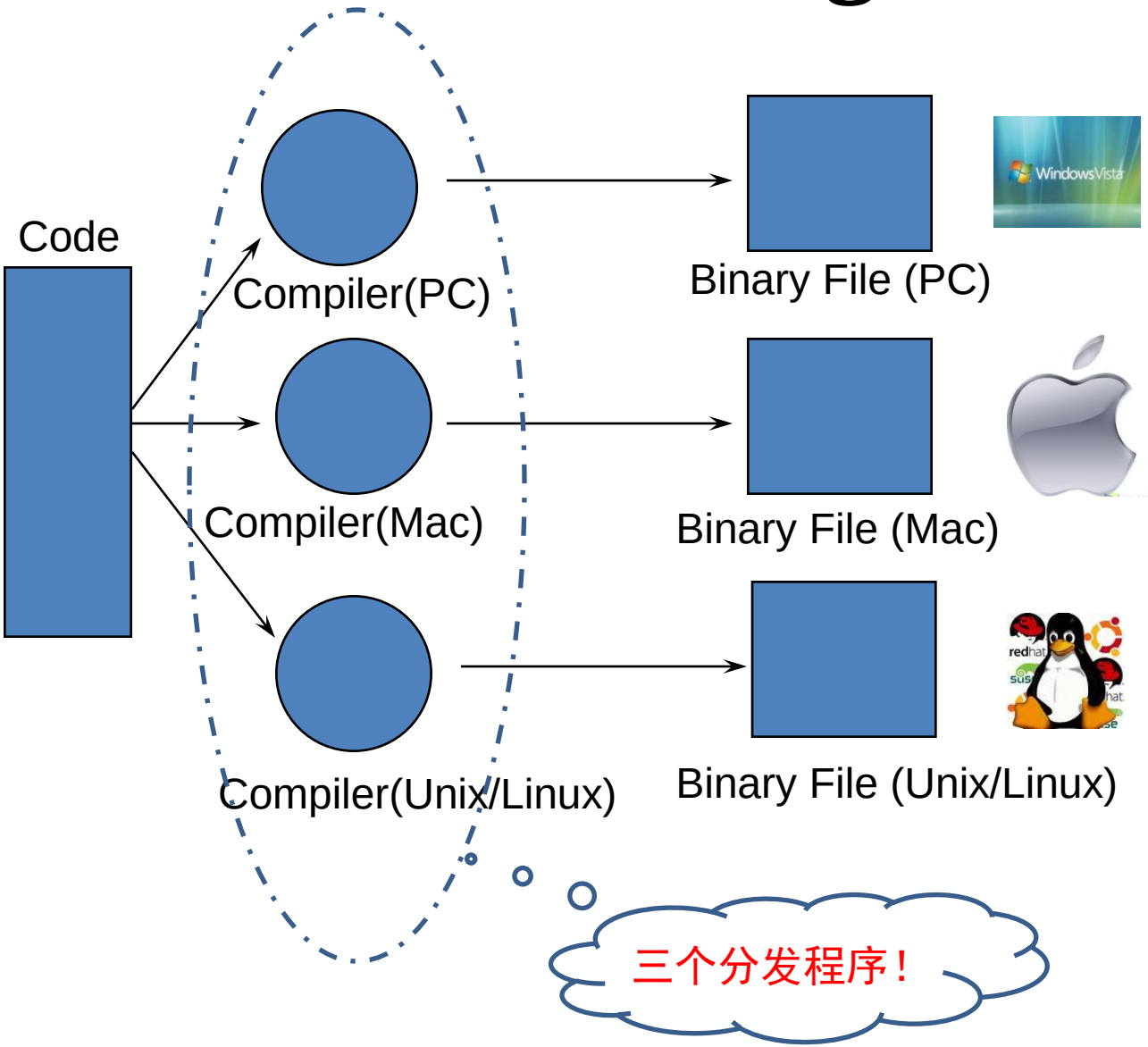
# Java的工作方式

- **Java虚拟机（JVM）**

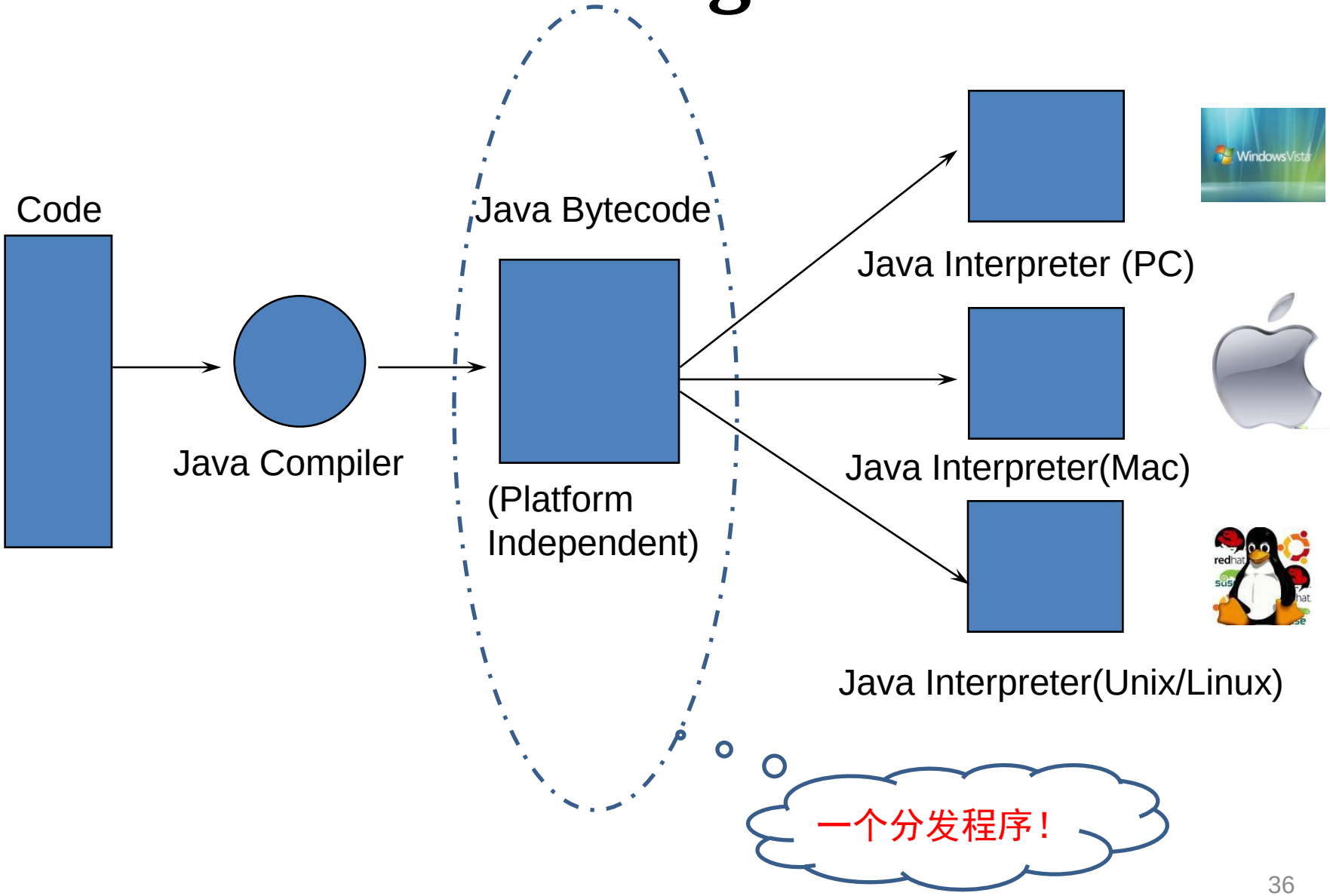
  **Java虚拟机类似于一个小巧而高效的CPU，Java处理器“芯片”，一般由软件实现**



# Traditional Programs



# Java Programs

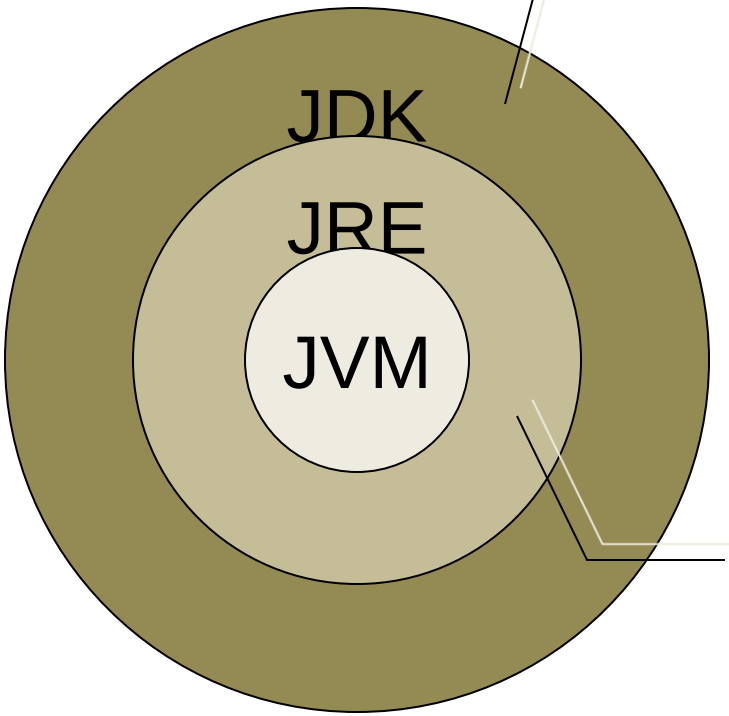


# Java术语—JVM、JRE

- JVM －Java Virtual Machine
- JVM 虚拟机将 Java 字节码程序翻译成机器语言，然后由计算机执行
- JVM 没有其他相关的文件
- JVM 本身不足以支持Java Application和Applet的运行
- JRE － Java 运行环境
- JRE 是由JVM 和 Java Platform 核心类以及相关支撑文件组成

# Java术语— JDK

- JDK — Java developer’s kit (Java开发工具包)
- JDK 包含JVM和其他工具，以及所有的API和相关文件。
- Java 2 — Java 2 Platform, J2 SDK –Java技术的新名称
- SDK — Software Development Kit




JDK plus all APIs, compilers , tools, and documentation (what you need in order to write Java technology programs).

JVM plus basic APIs (what you need to distribute to people who will run your Java programs)

# Java术语— API

- **API: Application programming interface.**
- **API 是rules(syntax) ：在Java技术中如何编程。**
- **API包括数百个类**
  - SUN公司预先编好的代码，你可以在编程中充分利用它们的功能。

# Java 开发工具

- 文本编辑器+JDK(Sun)　命令行方式
- Eclipse 集成开发环境 （Opensource, supported by IBM）
- JBuilder(Borland)
- IntelliJ IDEA(JetBrains)
- Visual J++(Microsoft)
- Netbeans (Sun)
- Visual Age for Java(IBM)

# Java SDK实用程序

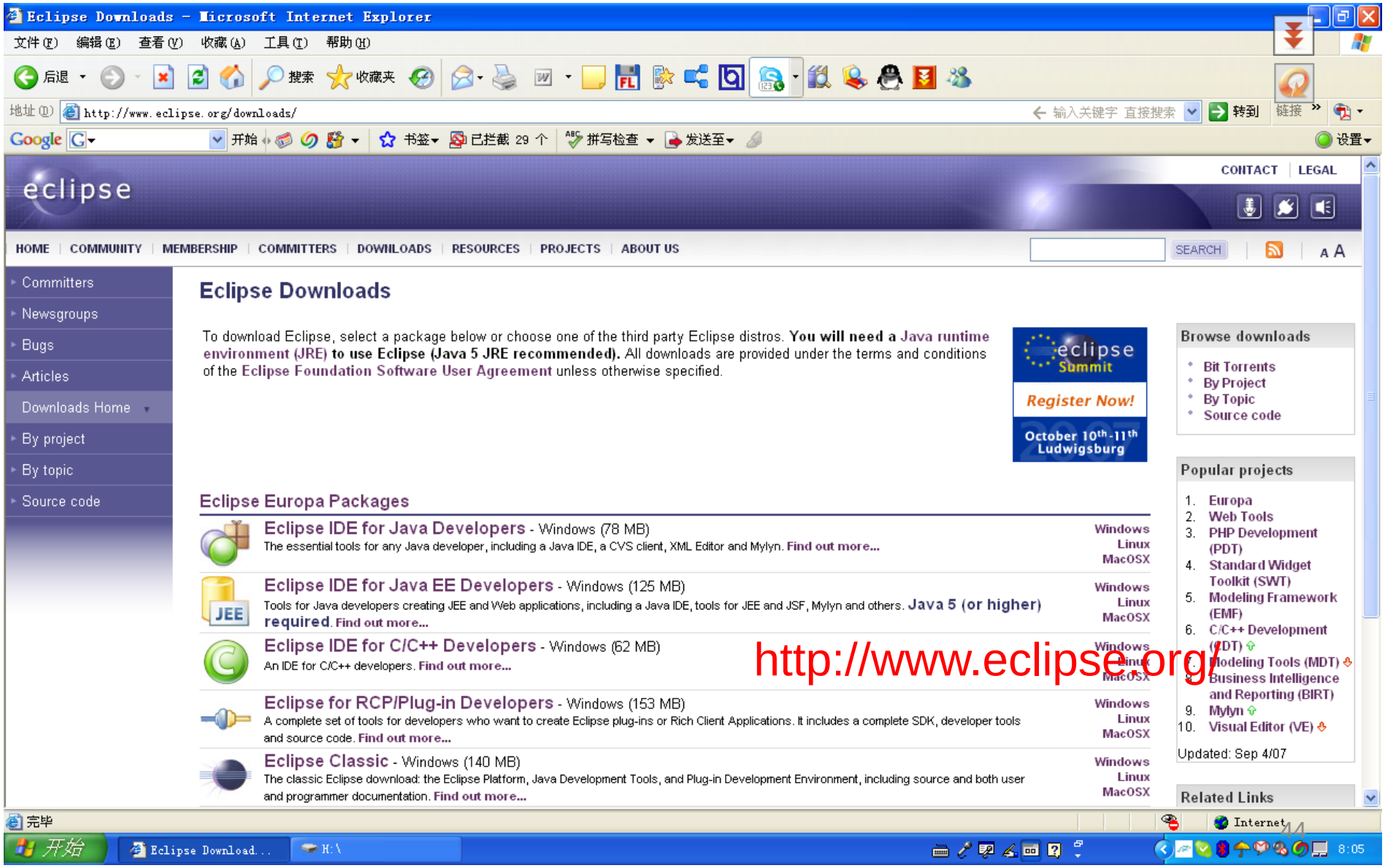
- **javac** Java编译器，将Java源程序编译成字节码
- **java** Java解释器，直接从类文件执行Java应用程序，即Application
- **appletviewer** 小程序浏览器，执行html文件中的Java小程序，即Applet

# 本课程的开发工具

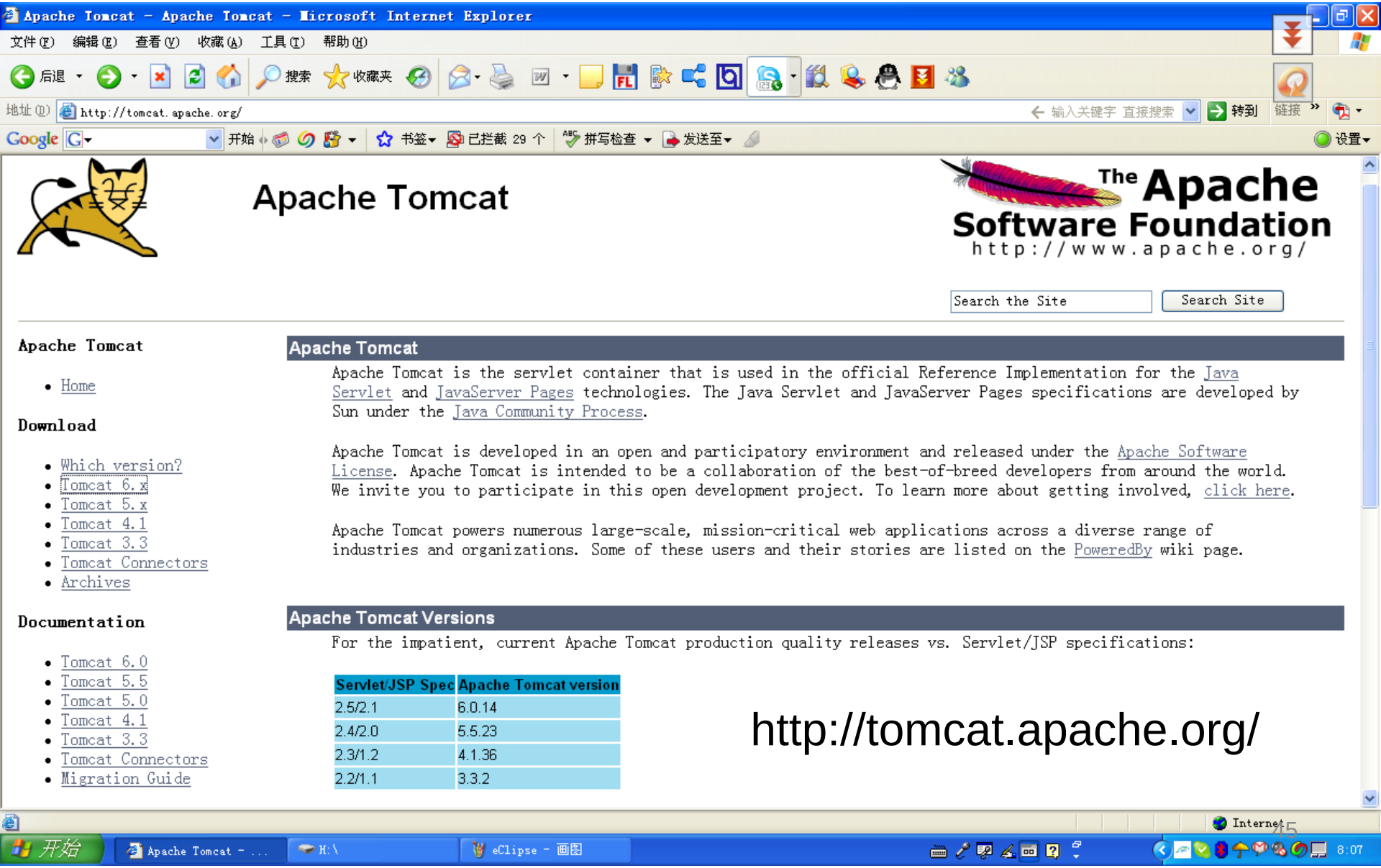
- **UltraEdit 12.0/Editlpus**
- **JDK1.6**

- **Eclipse 3.4（后期使用）**

# eclipse, IBM支助的Java IDE开源项目



# Tomcat, 执行Jsp/Servlet的容器



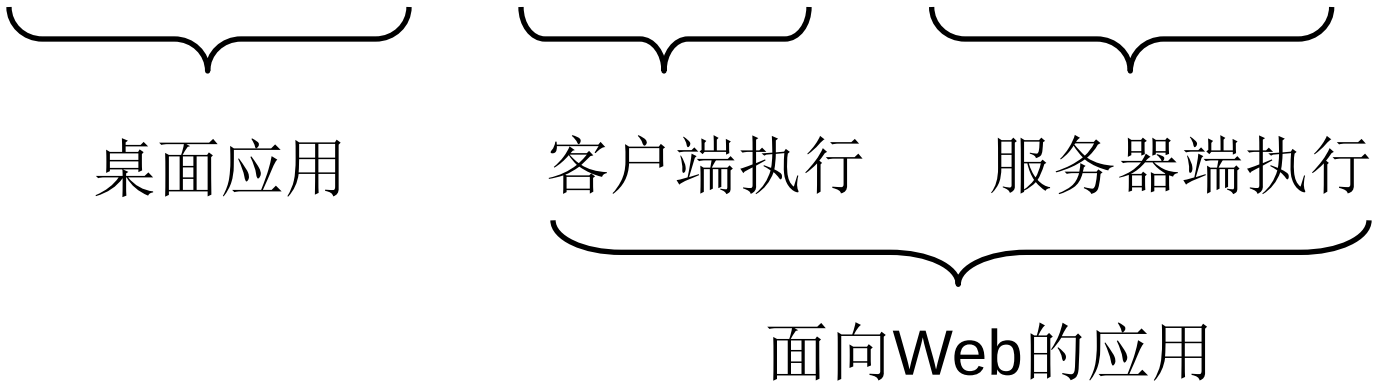
# 企业级应用的典型环境

项目开发及运行环境
操作系统：Windows\Unix\Linux
开发环境：JBuilder,eclipse
发布环境：Tomcat
数据库：MYSQL\Oracle\SQL Server

# 4 JAVA程序实例

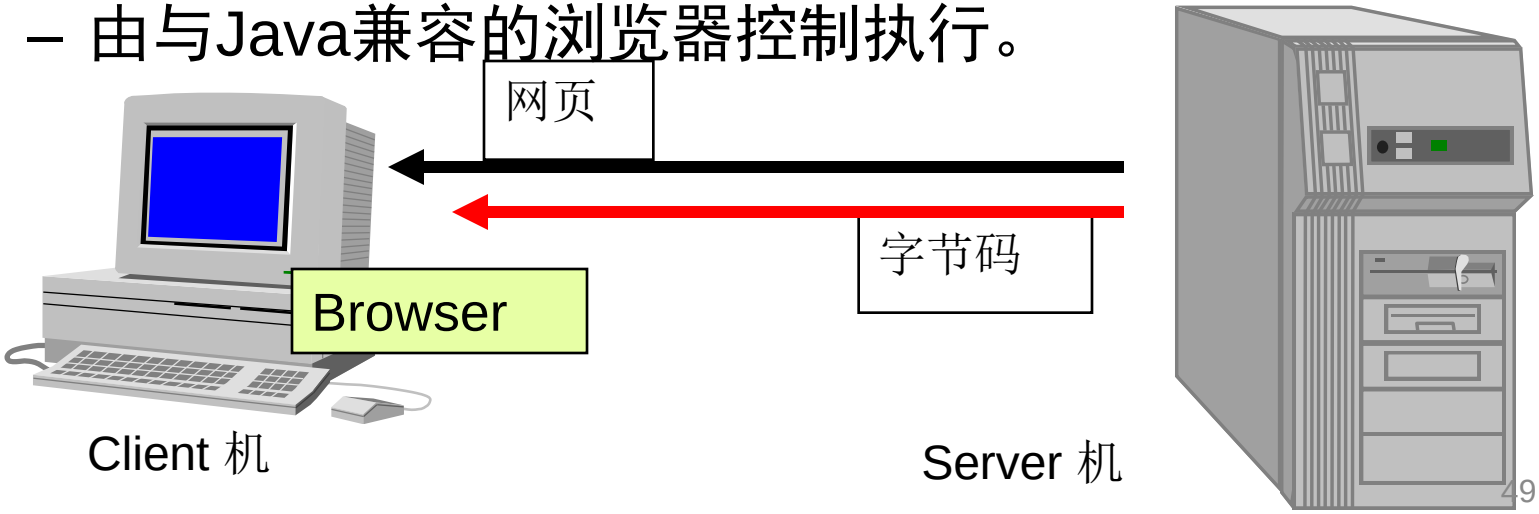
# Java程序的几种类型

Application, Applet, JSP, Servlet



# Java程序的几种类型

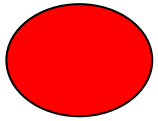
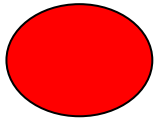
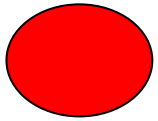
- **Application**
  - “Java应用”是可以独立运行的Java程序。
  - 由Java解释器控制执行。
- **Applet**
  - “Java小程序”不能独立运行，而是被嵌入到Web页中。
  - 由与Java兼容的浏览器控制执行。



# Java程序的几种类型

- **Servlet**
  - 是Java技术对CGI 编程的解决方案。
  - 是运行于Web server上的、作为来自于Web browser 或其他HTTP client端的请求 和 在HTTP server上的数据库及其他应用程序之间的中间层程序。
- **Servlet的工作是：**
  - 读入用户发来的数据（通常在web页的form中）
  - 找出隐含在HTTP请求中的其他请求信息（如浏览器功能细节、请求端主机名等)
  - 产生结果(调用其他程序、访问数据库、直接计算结果）
  - 格式化结果（网页）
  - 设置HTTP response参数(如告诉浏览器返回文档格式)
  - 将文档返回给客户端。

# 程序类型与用户界面

| | 字符界面 | 图形界面 |
|---|---|---|
| Application |  |  |
| Applet | |  |
| Servlets | | |

# JDK实用程序

- **javac**

  Java编译器，将Java源程序编译成字节码

- **java**

  Java解释器，直接从类文件执行Java应用程序，即application

- **applet viewer**

  小程序浏览器，执行html文件中的Java小程序，即Applet

# Application程序

**程序名称：HelloApp. java**

```
1 class HelloApp
2 {
3       public static void main(String args[])
4       {
5
6         System.out.println("welcome to java world!");
7
8       }
9 }
```

类声明与定义
关键字/类头与类体

类中的方法
main方法/方法头与方法体

用大括号括起语句组

语句结尾以分号标志

**程序的功能：输出“Welcome to java world!”**

# Java语言规则

- **Java语言区分大小写**
  - 类名第一个字母大写　　PhoneCard
  - 方法名第一个字母小写　　getBalance()
  - 变量名第一个字母小写　　cardNumber
  - 内含单词首字母大写

# Application程序基本结构

- 由一个或多个“类”组成。
- 其中必须有一个“类”定义了main() 方法,该方法是自动执行的类方法，是Java应用运行的起始点。而这个类也就称之为“主类”。
- 编辑Java源文件。如 HelloApplication.java:

```
import java.io.*;
public class HelloApplication  {
public static void main(String args[ ])  {
      System.out.println("Hello, Java world! ");
    }
}
```

# Application程序编译与执行

- **编译 — javac**

  **javac HelloApplication.java**

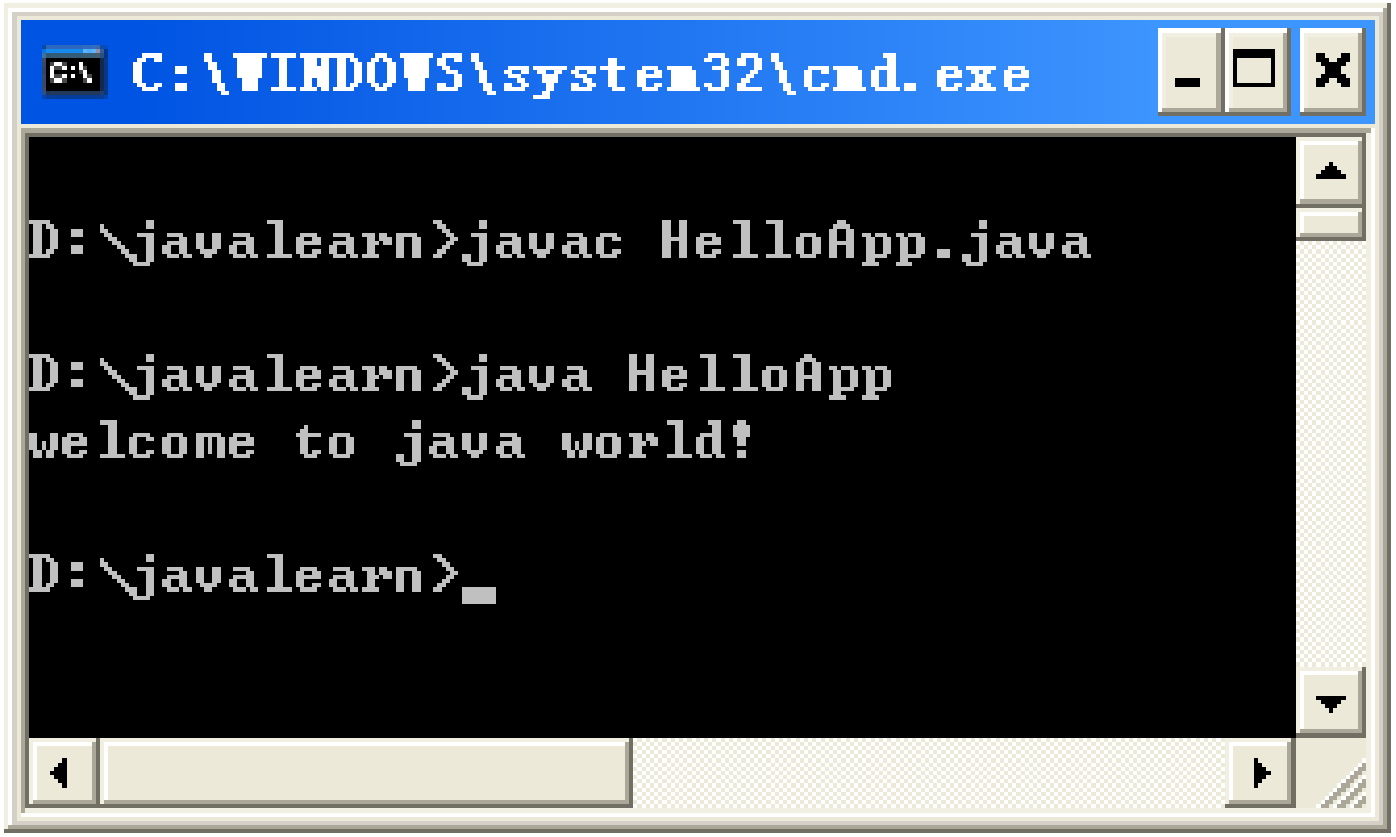
  **产生HelloApplication.class 文件**

- **解释执行 — java**

  **java HelloApplication （隐含.class文件）**

- **Java程序文件**
  - 源文件名要与主类名同名（包括大小写）
  - 一个类产生一个.class文件

# 编译、运行程序


C:\WINDOWS\system32\cmd.exe
D:\javalearn>javac HelloApp.java
D:\javalearn>java HelloApp
welcome to java world!
D:\javalearn>

# Applet程序

**程序名称：HelloApplet. java**

```
1 import java.applet.*;
2 import java.awt.*;
3
4 public class HelloApplet extends Applet{
5
6         public void paint(Graphics g){
7
8             g.drawString("Welcome to Java World!",25,50);
9
10        }
11 }
```

引入本程序所需要的类
API概念/本程序中引用的类/包空间

继承
子类与父类

该程序的入口？

**程序的功能：输出“Welcome to java world!”**

# Applet程序

- Applet程序中必须包含java.applet.Applet类的子类。该子类就是Applet程序的主类
- 系统类Applet中已经定义了很多的成员域和成员方法，它们规定了Applet小程序如何与执行它的解释器 — Web浏览器配合工作

# 程序名称：HelloApplet.html

```
<HTML>
  <HEAD>
    <TITLE>HelloApplet小程序 </TITLE>
  </HEAD>
  <BODY>
    <APPLET CODE="HelloApplet.class" HEIGHT=200 WIDTH=300></APPLET>
  </BODY>
</HTML>
```

将Applet小程序的代码嵌入到HTML文件中，并用Appletviewer或支持java的浏览器来运行

# Applet小程序的编译与执行过程

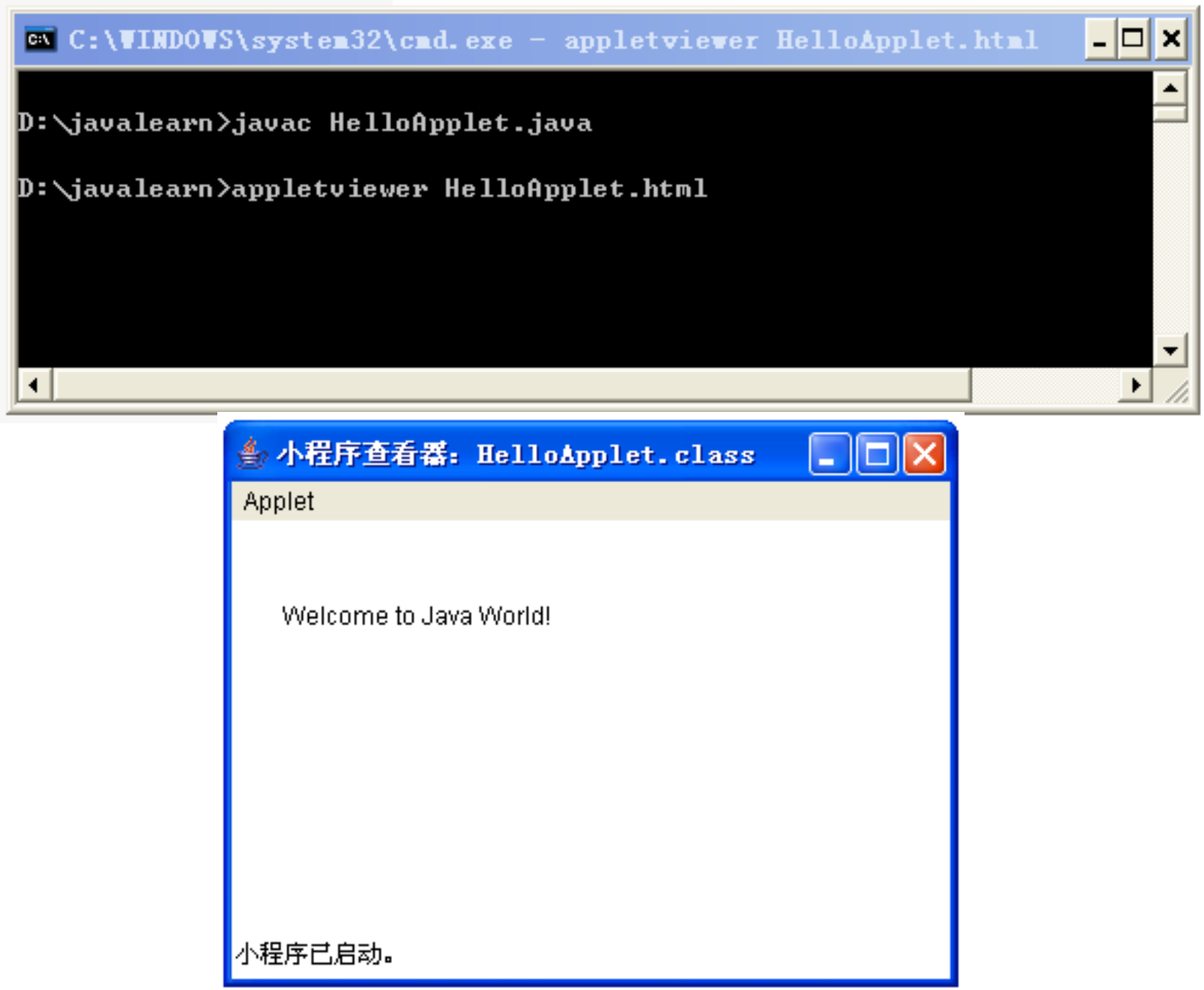
**(1) 编译Java小程序  javac HelloApplet.java**

**(2) 建立HelloApplet.html文件，嵌入HelloApplet.class**

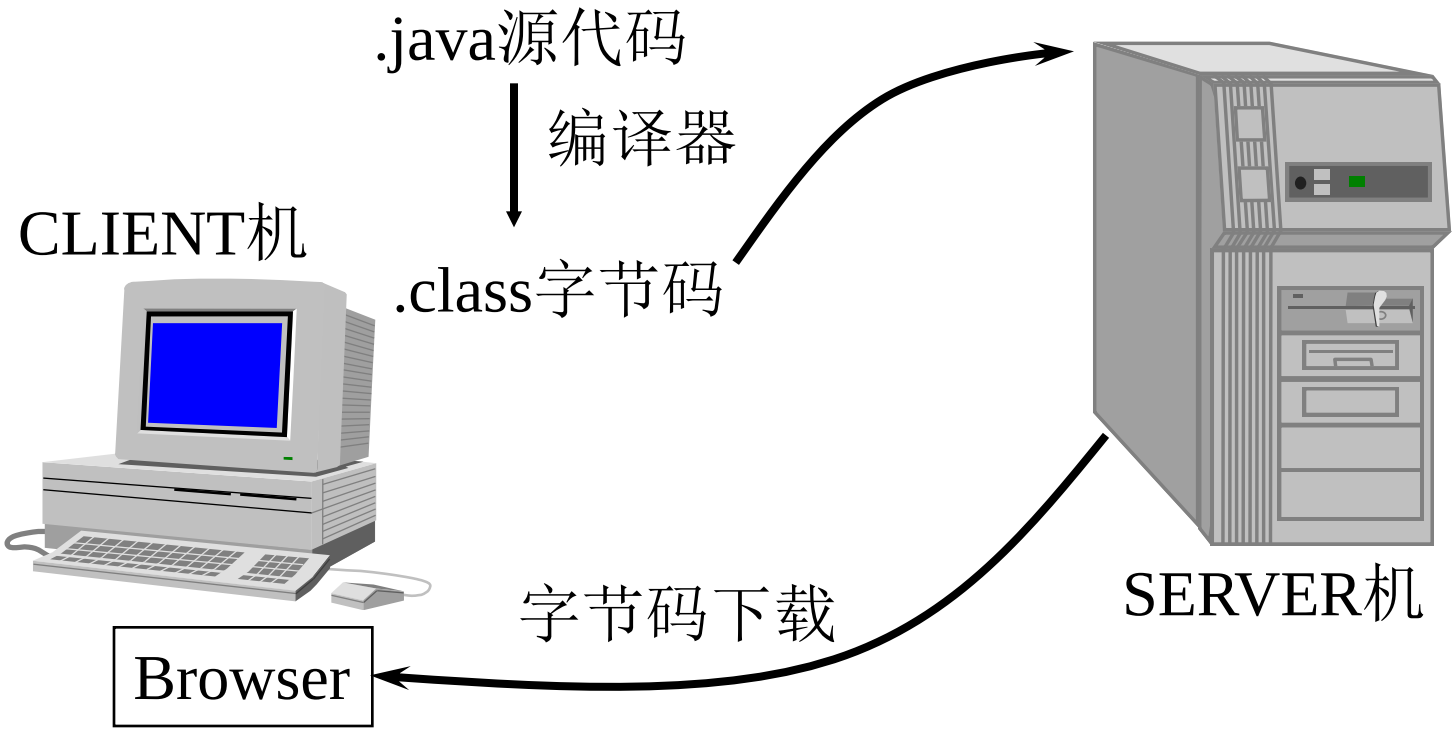
**(3) 浏览HelloApplet.html文件,有两种方式**

① 使用appletviewer (appletviewer是一个简化的浏览器)

② 使用常规的浏览器


C:\WINDOWS\system32\cmd.exe - appletviewer HelloApplet.html
D:\javalearn>javac HelloApplet.java
D:\javalearn>appletviewer HelloApplet.html
小程序查看器: HelloApplet.class
Applet
Welcome to Java World!
小程序已启动。

# Applet开发、执行模式



# Jsp/Servlet程序类型

- **Jsp/Servlet需要运行在能够解释JSP，执行Servlet的容器中。**
- Jsp程序

```
1 <html>
2 <head><title>Hello! First JSP!</title></head>
3 <body>
4  <%out.println("Hello! World!");%>
5 </body>
6 </html>
```

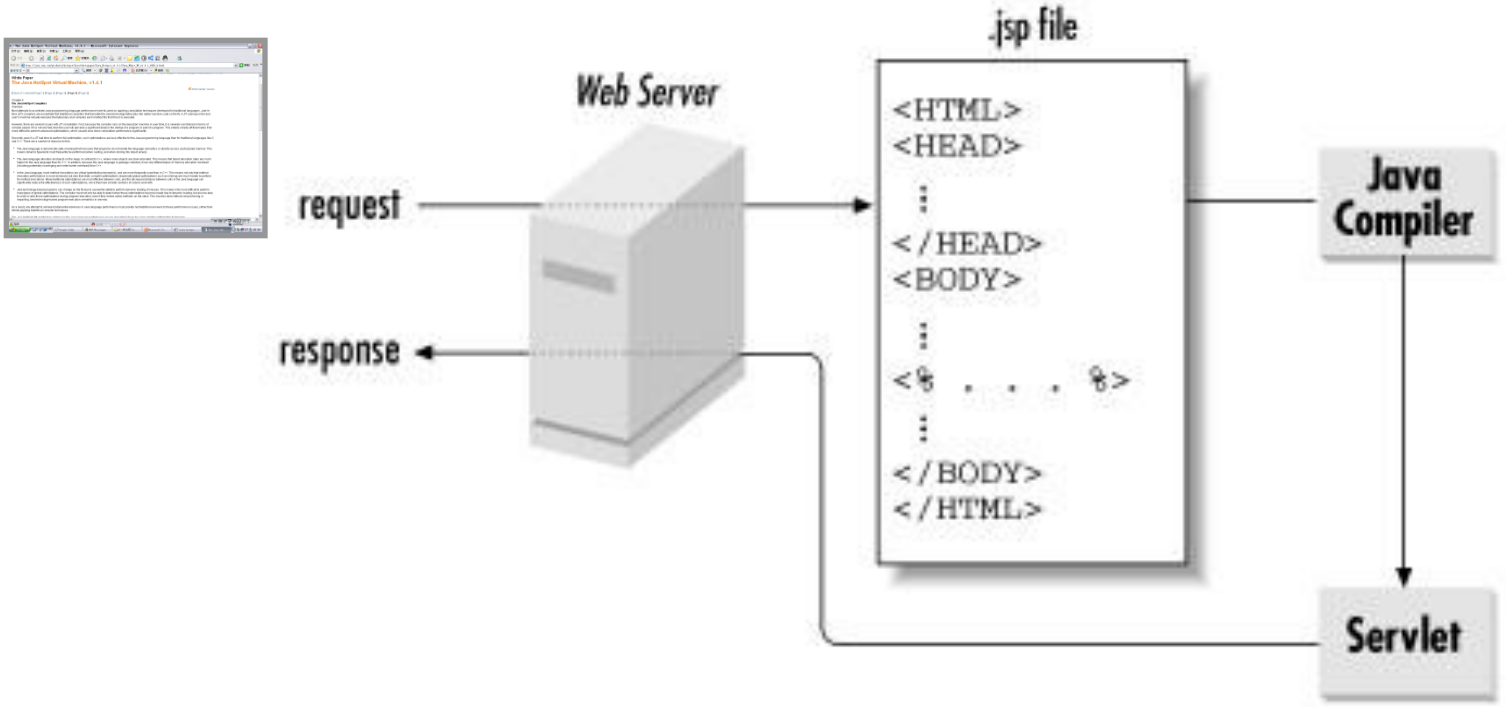
# Servlet

```
1 import java.io.*;
2 import javax.servlet.*;
3 import javax.servlet.http.*;
4
5 public class HelloWorld  extends HttpSerevlet{
6         public void init() throws ServeletException{ }
7         public void doGet(HttpServeletRequest request, HttpServeletResponse response)
8                 throws ServeletException,IOException
9         {
10           PrintWriter out = response.getWriter();
11           out.println("Hellow World1");
12        }
13
14        public void destory(){}
15 }
```

Servlets的工作是：

(1) 读入用户发来的数据（通常在web页的form中）
(2) 找出隐含在HTTP请求中的其他请求信息（如浏览器功能细节、请求端主机名等)
(3) 产生结果(调用其他程序、访问数据库、直接计算结果）
(4) 格式化结果（网页）
(5) 设置HTTP response参数(如告诉浏览器返回文档格式)
(6) 将文档返回给客户端。

# JSP/Servlet背后的运行机制



# 简单输出方法

- **字符界面（用于Application）**
  - System.out.println(“ 字符串 ”)；
- **图形界面（在Applet中）**
  - 在paint()方法中
  - g.drawString(“ 字符串 ” , xInt, yInt);

    注：(xInt, yInt)为输出字符串的起始位置坐标，即x表示第一个字符的左边界， y表示整个字符串的基准线位置坐标。

# 上机实践

- 熟悉学习元平台。
- 能独立安装Java开发环境并在命令行模式下编译、执行Java程序。

# Java概述


宋杰 硕士研究生
songjiesdnu@163.com

北京师范大学教育技术学院
现代教育技术研究所