



第一章 Java概述

主讲：丛小茗



第1章 Java概述

- **1.1 了解Java**

- **1.2 Java开发环境搭建与测试运行**




1.2 Java的开发环境搭建和测试运行

- 1 Java** 开发平台介绍
- 2 Java SE** 平台结构
- 3 Java SE** 开发环境的安装
- 4 Java SE** 程序的编辑、编译与运行



1.2 Java的开发环境搭建和测试运行

- 1 Java 开发平台介绍**
- 2 Java SE 平台结构
- 3 Java SE 开发环境的安装
- 4 Java SE 程序的编辑、编译与运行



1: Java开发平台(Platform)

(1) Java SE:

(Java Platform, Standard Edition)

(2) Java EE:

(Java Platform, Enterprise Edition)

(3) Java ME:

(Java Platform, Micro Edition)



(1) Java SE:

- 允许开发和部署在桌面、服务器、嵌入式环境和 实时环境中使用的 **Java** 应用程序。
- 为**Java EE**和**ME**提供语言基础
- Java Platform, Standard Edition (Java SE) lets you develop and deploy Java applications on desktops and servers. Java offers the rich user interface, performance, versatility, portability, and security that today's applications require.
- 新版本**Java SE 8 (JSR337)**
- 最新版本**Java SE 11(JSR384 LTS), 14 (JSR389)**



(2) Java EE:

- 以**SE**为基础定义了一些**API**,服务和**规范**
- 适用于开发分布式,基于**组件**的企业级应用
- 最新版本是**Java EE 8 (JSR 366)**



(3) Java ME:

-- 为在移动设备和嵌入式设备（比如手机、**PDA**、电视机顶盒和打印机）上运行的应用程序提供一个健壮且灵活的环境

--**Java ME SDK 8.1**



1.2 Java的开发环境搭建和测试运行

- 1 Java 开发平台介绍
- 2 Java SE 平台结构**
- 3 Java SE 开发环境的安装
- 4 Java SE 程序的编辑、编译与运行



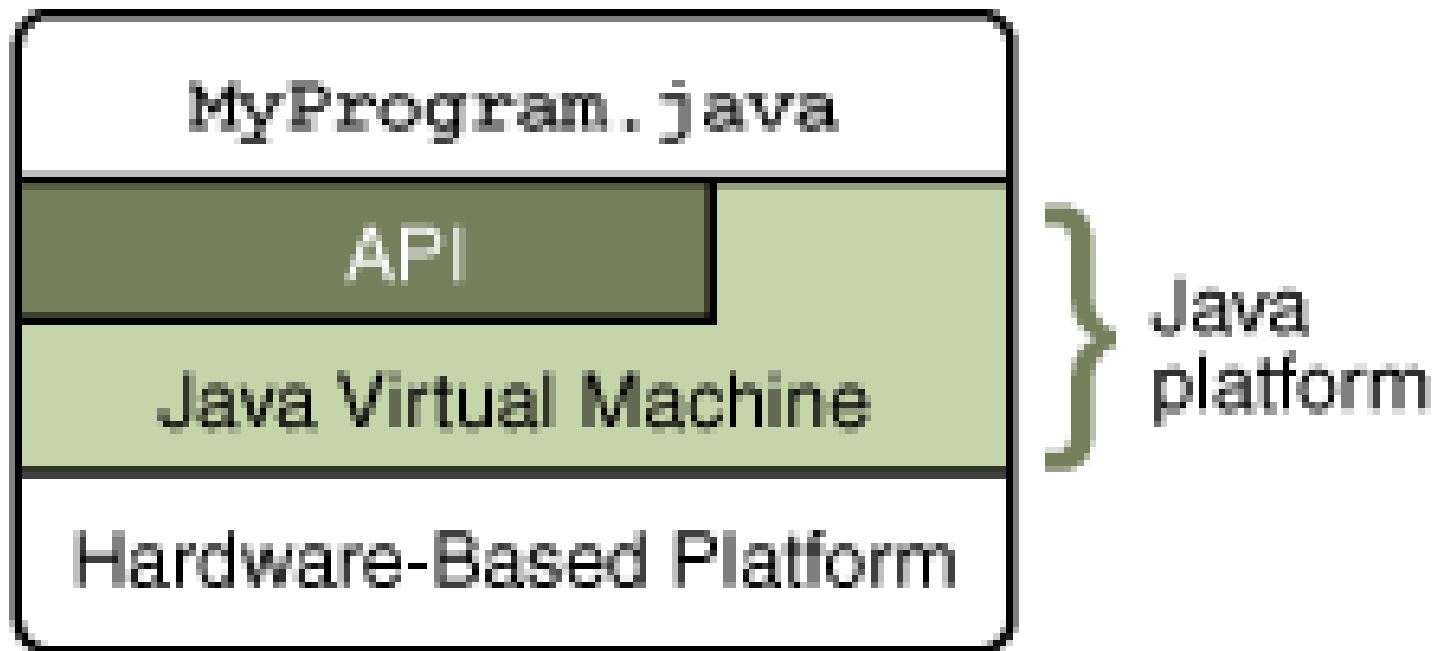
Java SE Platform由:

- ---Java 虚拟机**JVM**

(Java Virtual Machine)

---应用编程接口**API**

(Application Programming Interface)



来自:

<http://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html>



What's JVM?

- The Java HotSpot Virtual Machine is a **core** component of **the Java SE platform**.
- It implements the **Java Virtual Machine Specification(JSR924)**, and is delivered as a shared library in the Java Runtime Environment.
- As the **Java bytecode execution engine**, it provides Java runtime facilities, such as thread and object synchronization, on a variety of operating systems and architectures.
- It includes dynamic compilers that adaptively compile Java bytecodes into optimized machine instructions and efficiently manages the Java heap using garbage collectors, optimized for both low pause time and throughput.
- It provides data and information to profiling, monitoring and debugging tools and applications.

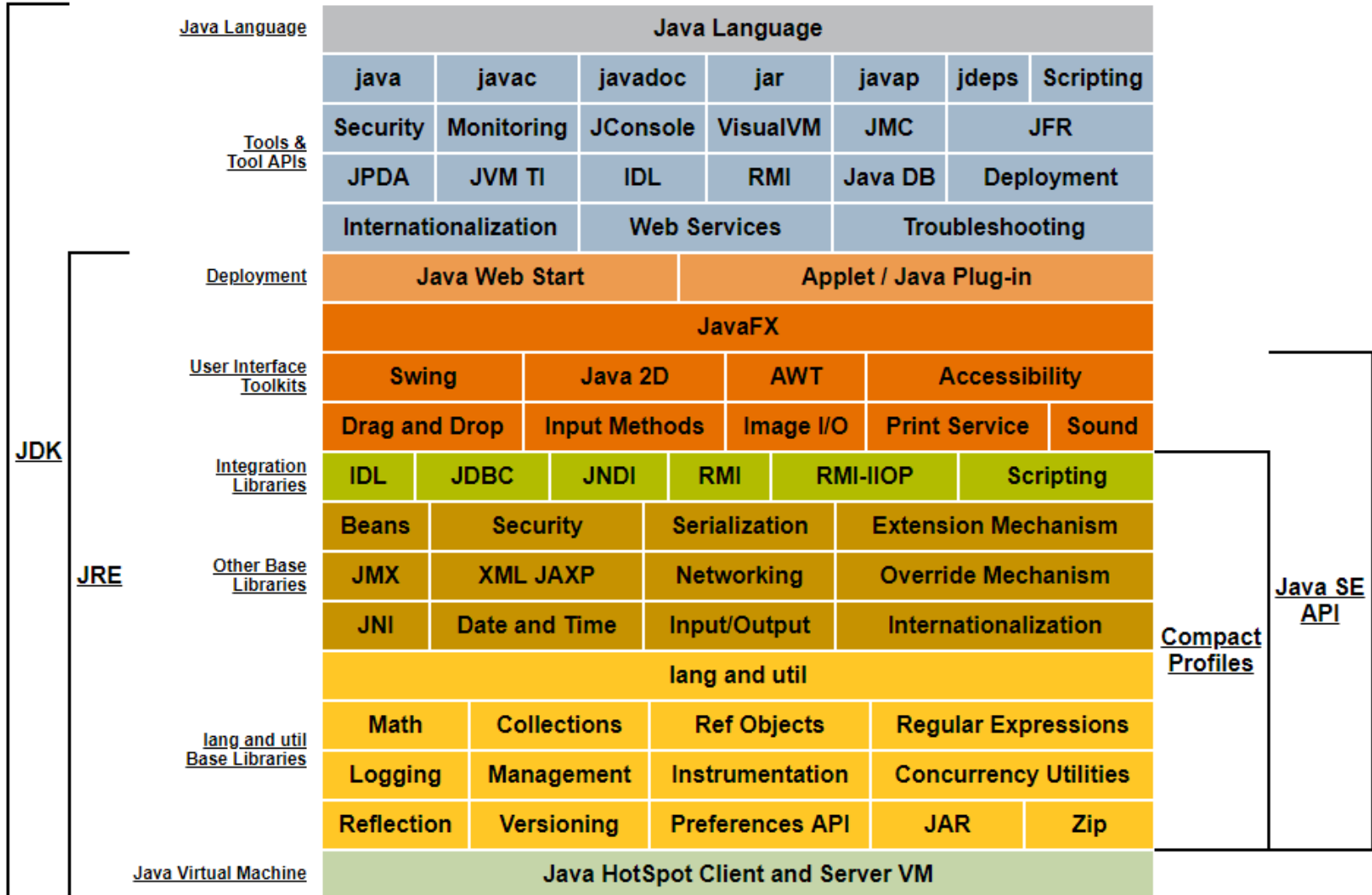


JVM

■ java 虚拟机(Java Virtual Machine)

- 是用软件模型实现的虚拟计算机，它定义了指令集，寄存器集，类文件结构栈，垃圾收集堆，内存区域,处理器等，提供了跨平台能力的基础框架
- Java虚拟机是建立在实际的处理器基础上的用软件实现的计算机
- Java 8 参考规范：
<https://docs.oracle.com/javase/specs/jvms/se8/html/>

Java Conceptual Diagram





Java APIs

Java SE

The Java Platform, Standard Edition (Java SE) APIs define the core Java platform for general-purpose computing. These APIs are in modules whose names start with `java`.

JDK

The Java Development Kit (JDK) APIs are specific to the JDK and will not necessarily be available in all implementations of the Java SE Platform. These APIs are in modules whose names start with `jdk`.

Java™ Platform
Standard Ed. 8

All Classes All Profiles

Packages

java.applet
java.awt
java.awt.color
java.awt.datatransfer

All Classes

AbstractAction
AbstractAnnotationValueVisitor6
AbstractAnnotationValueVisitor7
AbstractAnnotationValueVisitor8
AbstractBorder
AbstractButton
AbstractCellEditor
AbstractChronology
AbstractCollection
AbstractColorChooserPanel
AbstractDocument
AbstractDocument.AttributeContext
AbstractDocument.Content
AbstractDocument.ElementEdit
AbstractElementVisitor6
AbstractElementVisitor7
AbstractElementVisitor8
AbstractExecutorService
AbstractInterruptibleChannel
AbstractLayoutCache
AbstractLayoutCache.NodeDimensions
AbstractList
AbstractListModel
AbstractMap

Java™ Platform
Standard Ed. 8

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV NEXT FRAMES NO FRAMES

Java™ Platform, Standard Edition 8
API Specification

This document is the API specification for the Java™ Platform, Standard Edition.

See: Description

Profiles

- compact1
- compact2
- compact3

Packages

Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two



1.2 Java的开发环境搭建和测试运行

- 1 Java 开发平台介绍
- 2 Java SE 平台结构
- 3 Java SE 开发环境的安装**
- 4 Java SE 程序的编辑、编译与运行



3 Java SE开发环境的安装

1)JDK开发包下载(本课基于JDK8)

2)JDK开发包安装过程

3)JDK Documentation的”安装”过程



3 Java SE开发环境的安装

1)JDK开发包下载(本课基于JDK8)

2)JDK开发包安装过程

3)JDK Documentation的”安装”过程



1)JDK开发包下载

🔗 **JDK开发包**
下载链接

<https://www.oracle.com/java/technologies/javase-jdk8-downloads.html>

🔗 **JDK开发文档(jdk-8u261-docs-all.zip)**

下载链接<https://www.oracle.com/java/technologies/javase-jdk8-doc-downloads.html>

[Products](#)[Resources](#)[Support](#)[Events](#)

Java SE 8u261

Java SE 8u261 includes important bug fixes. Oracle strongly recommends that all Java SE 8 users upgrade to this release.

- [Documentation](#)
- [Installation Instructions](#)
- [Release Notes](#)
- [Oracle License](#)
 - [Binary License](#)
 - [Documentation License](#)
 - [BSD License](#)
- [Java SE Licensing Information User Manual](#)
 - [Includes Third Party Licenses](#)
- [Certified System Configurations](#)
- [Readme Files](#)
 - [JDK ReadMe](#)
 - [JRE ReadMe](#)

Oracle JDK

[JDK Download](#)[Server JRE Download](#)[JRE Download](#)[Documentation Download](#)[Demos and Samples Download](#)

Which Java package do I need?



3 Java SE开发环境的安装

1)JDK开发包下载(本课基于JDK8)

2)JDK开发包安装过程

3)JDK Documentation的”安装”过程



2)JDK开发包的安装过程(Windows)

- 安装JDK
- 测试JDK是否安装成功



2)JDK开发包的安装过程(Windows)

安装指导

https://docs.oracle.com/en/java/javase/11/install/overview-jdk-installation.html?xd_co_f=27cdc8e7c71c7b0ca741597080343553#GUID-8677A77F-231A-40F7-98B9-1FD0B48C346A

6 Installation of the JDK on Microsoft Windows Platforms



This topic includes the following sections:

- [System Requirements for Installing the JDK on 64-Bit Windows Platform](#)
- [JDK Installation Instruction Notation for Windows](#)
- [JDK Installation Instructions for Windows](#)
- [Beginning to Use the JDK](#)
- [Uninstalling the JDK on Windows](#)
- [JDK Installation Troubleshooting](#)

System Requirements for Installing the JDK on 64-Bit Windows Platform

For supported processors and browsers, see [Oracle JDK Certified Systems Configurations](#).

JDK Installation Instruction Notation for Windows

For any text in this document that contains the following notation, you must substitute the appropriate update version number:

interim.update.patch

For example, if you are downloading the JDK installer for 64-bit systems for update 11 Interim 0, Update 0, and Patch 0, then the file name `jdk-11.interim.update.patch_windows-x64_bin.exe` becomes `jdk-11_windows-x64_bin.exe`.



安装JDK8

- 只需运行 **`jdk-*_windows-x64_bin.exe`** 。
- 安装到默认目录下，
例如 **`C:\Program Files\Java`** 。



测试JDK安装是否成功

— **javac**命令（**path**设置）

测试Jdk开发环境是否成功安装

— **java -version**

测试Java版本是否与安装一致



3 Java SE开发环境的安装

1)JDK开发包下载(本课基于JDK8)

2)JDK开发包安装过程

3)JDK Documentation的“安装”过程



3)JDK Documentation的”安装”过程

- 只需解压 **jdk-*_doc-all.zip** 即可。

建议！！

将**Documentation** 解压到**C:\Program Files\Java**

- 或者使用在线**Documentation**

<https://docs.oracle.com/javase/11/>



4)安装目录说明:

目录	说明
bin	Java 编译器等开发工具
jre	JDK 自带的 JRE
lib	库文件
demo	演示程序
docs	库文档， html 格式
src	库源文件



4 Java程序的编辑、编译与运行

- **Java**程序分为:

- Application(应用程序)**

- Applet(小应用程序)**



(1) Application:

■ Application 结构:

```
public class ClassName{  
    public static void main(String[ ] args)  
    {  
        .....  
    }  
}
```



(1) Application:

■ 第一个Application:

```
public class Hello
{
    public static void main(String[ ] args)           //main
    {
        System.out.println("Hello World!");           //用于在屏幕上显示内容
    }
}
```

- 注：源程序编写可使用任意的纯文本编辑器，**notepad**，**UltraEdit**等。

(1) Application:



步骤：

☞ 编码：

- 1) 打开文本编辑器(**记事本**、写字板、**word**均可)或者**IDE**集成的编辑器。
- 2) 将**Hello.java**代码输入到编辑器中
- 3) 文件保存为**Hello.java** (**注**：一定要注意扩展名)

(1) Application:



步骤：

☞ 编译与运行：-----Terminal模式

- 1) 进入命令行运行模式
- 2) 进入保存**Hello.java**的目录
- 3) 运行**javac**编译源程序，**javac Hello.java**
- 4) 查看是否生成了字节码文件” **Hello.class**”
- 5) 执行程序，**java Hello**



补充: Path 设置

■ Path作用:

- 环境变量是**OS**中定义的变量。
- 用来告诉**OS**到什么目录下寻找需要的应用程序



补充: Path 设置(书P8)

■ 设置path:

- **Windows**下, 启动控制面板, 选择系统→环境变量→系统变量→**PATH**, 加入
C:\Program Files\Java \bin
(或: 在**dos**命令行方式下进行临时**path**的设置)



补充: **Path** 设置

- **测试path:**

(启动一个**dos**命令行窗口, 输入下面的命令)

- **javac:** 测试**Java**编译器是否正确安装。

(1) Application:



步骤：

☞ 编译与运行：-----**IDE**模式

在**IDE**中使用集成的一键式“**Run**”命令



Java IDE

- **NB or Eclipse or MyEclipse or IntelliJ**

Java IDE

- 
- **NetBeans**是Apache的开源项目(www.netbeans.org)
[NetBeans12-EE&SE][Java 14 or older]

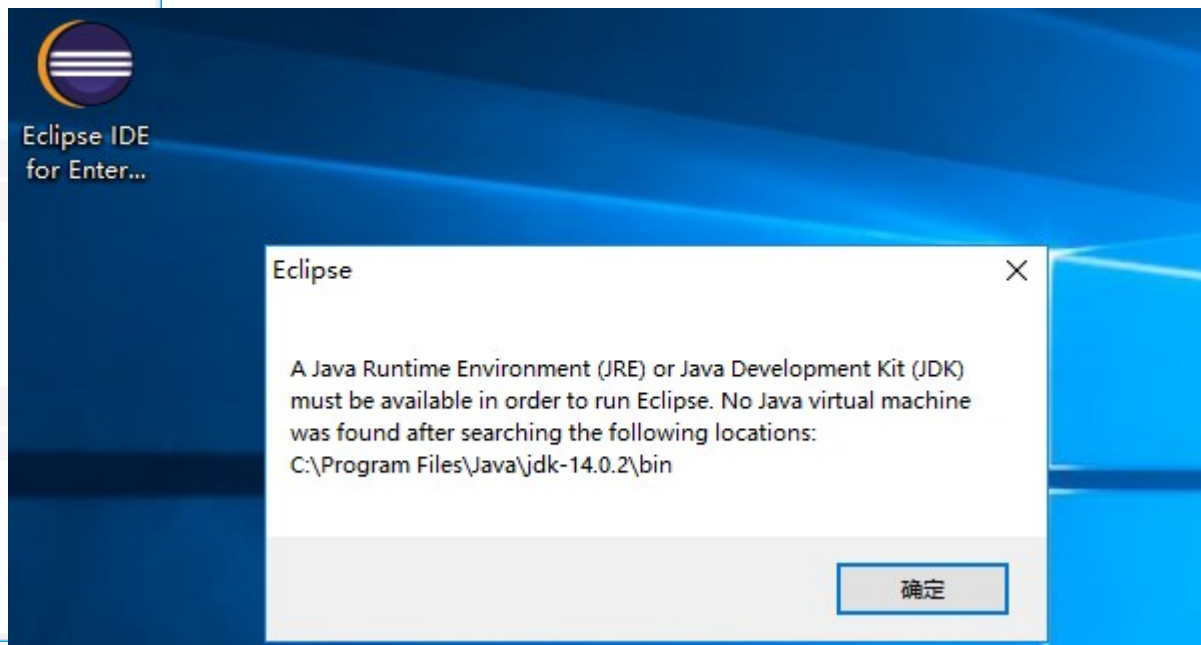
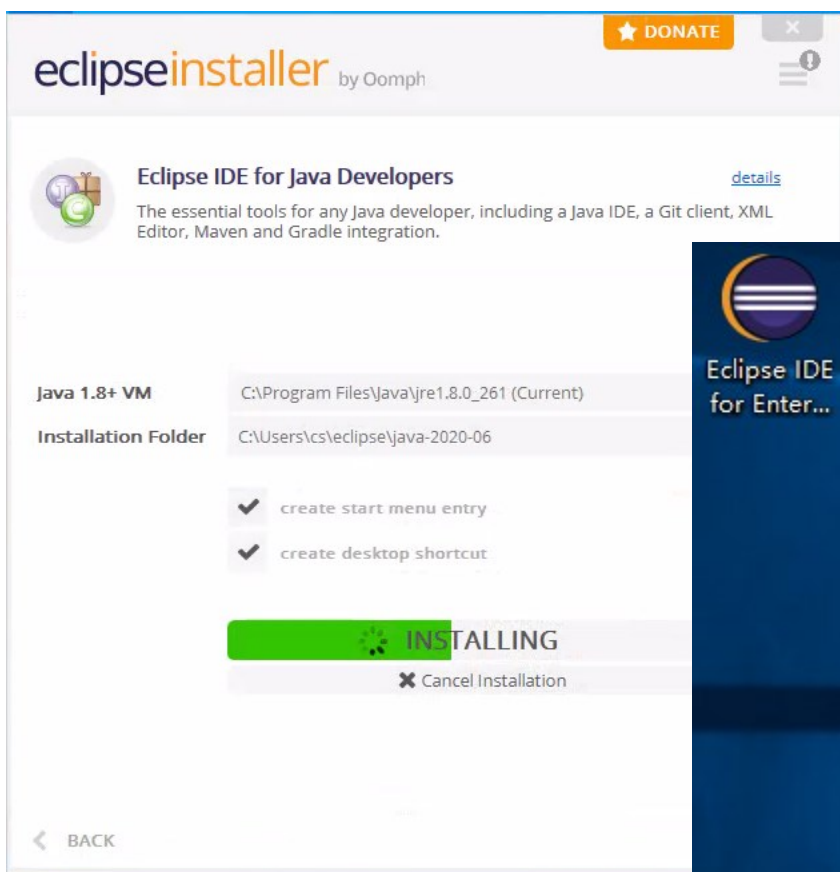
- **Eclipse**是IBM公司作为主赞助商的开源项目
(www.eclipse.org)[新版本:
4.16,4.12,4.11,..4.8PHOTON]

<https://www.eclipse.org/downloads/packages/>

- **MyEclipse** 是基于**Eclipse**的一个插件,收费软件
[MyEclipse 2020.5.18b]
- **IntelliJ IDEA 2020.2.1**

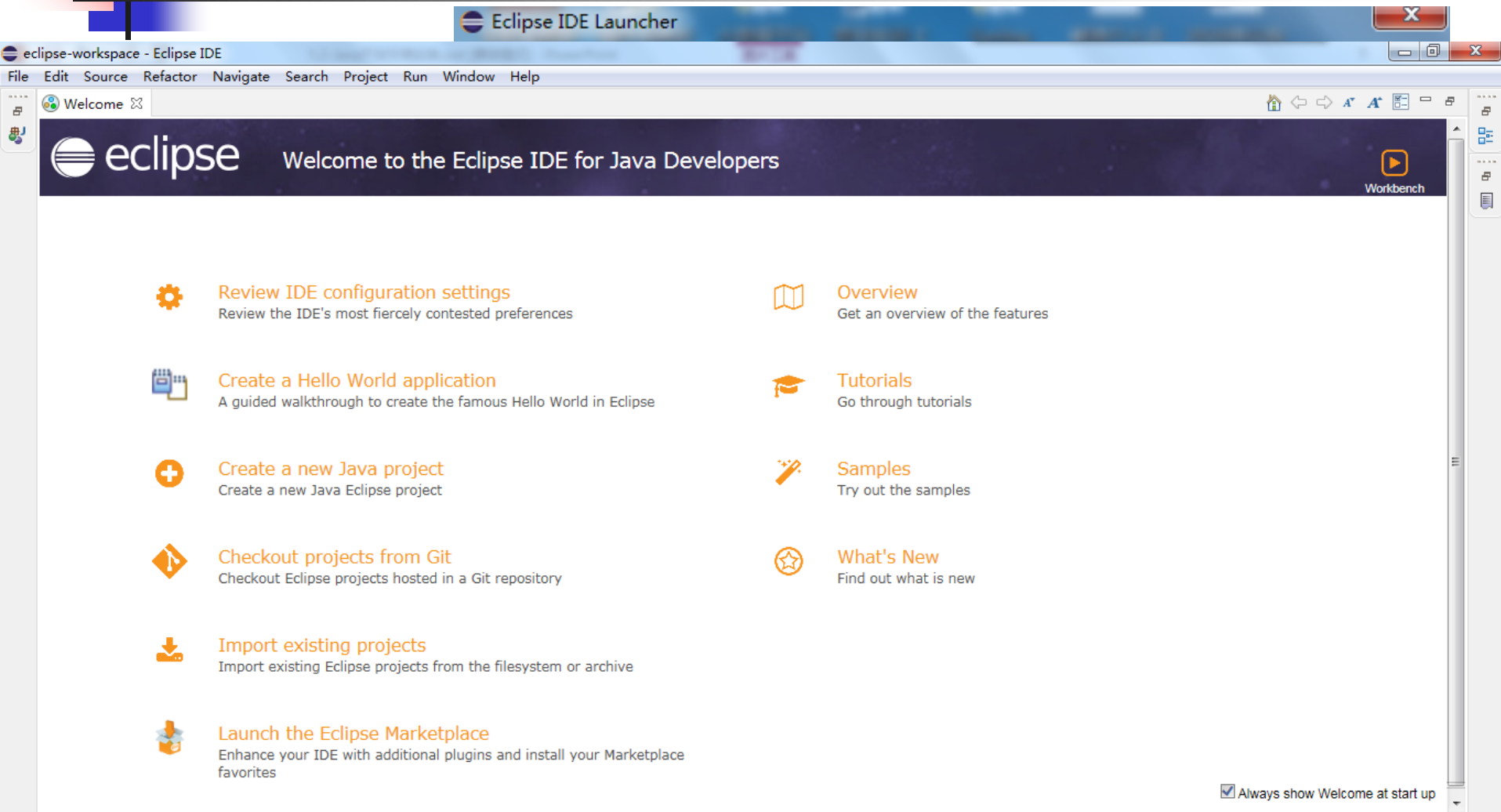
Eclipse的安装

安装之前必须先装**JDK**并设置环境变量**path**

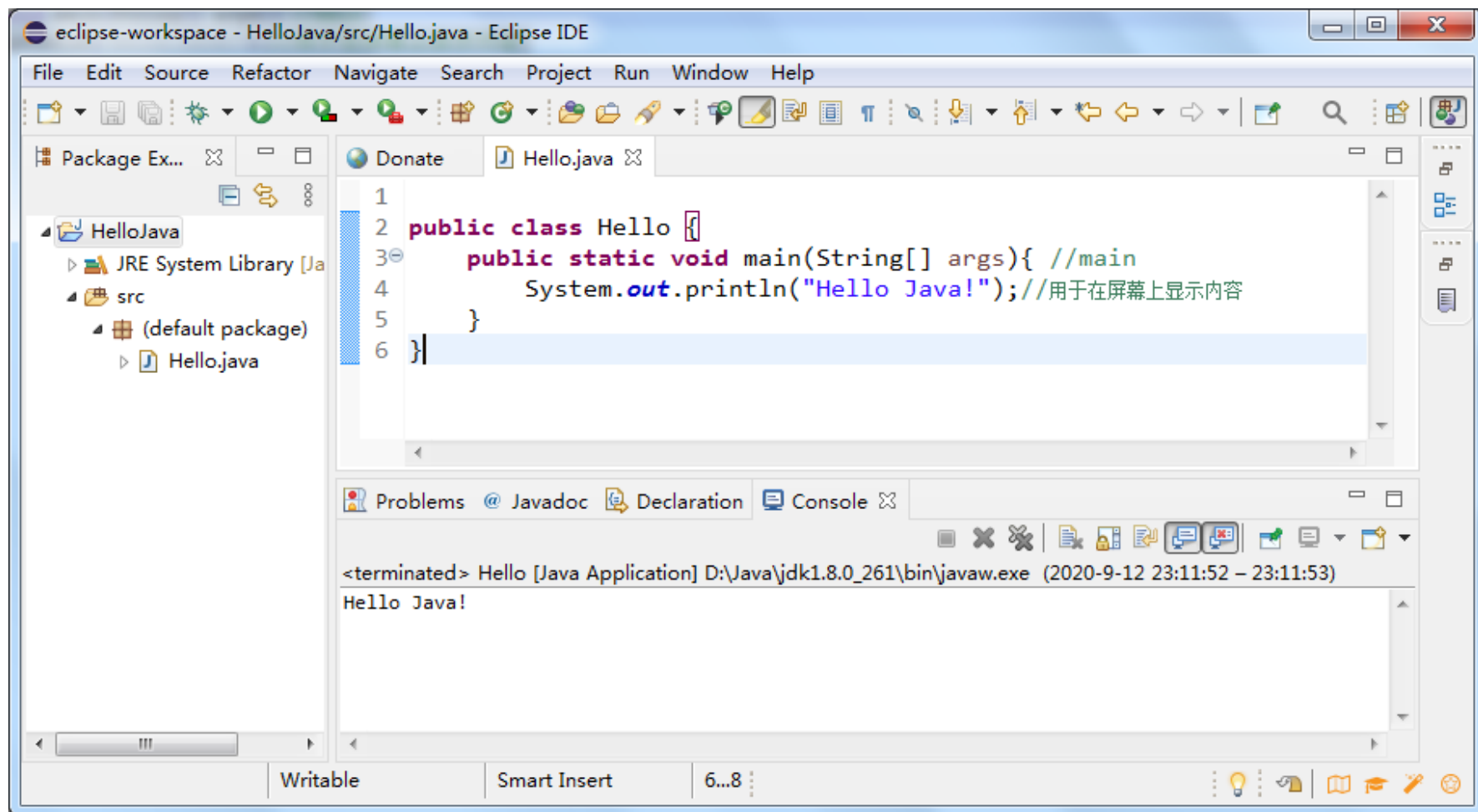


(没有安装**JDK**或者设置**Path**)

Eclipse的启动



Eclipse下运行Hello.java



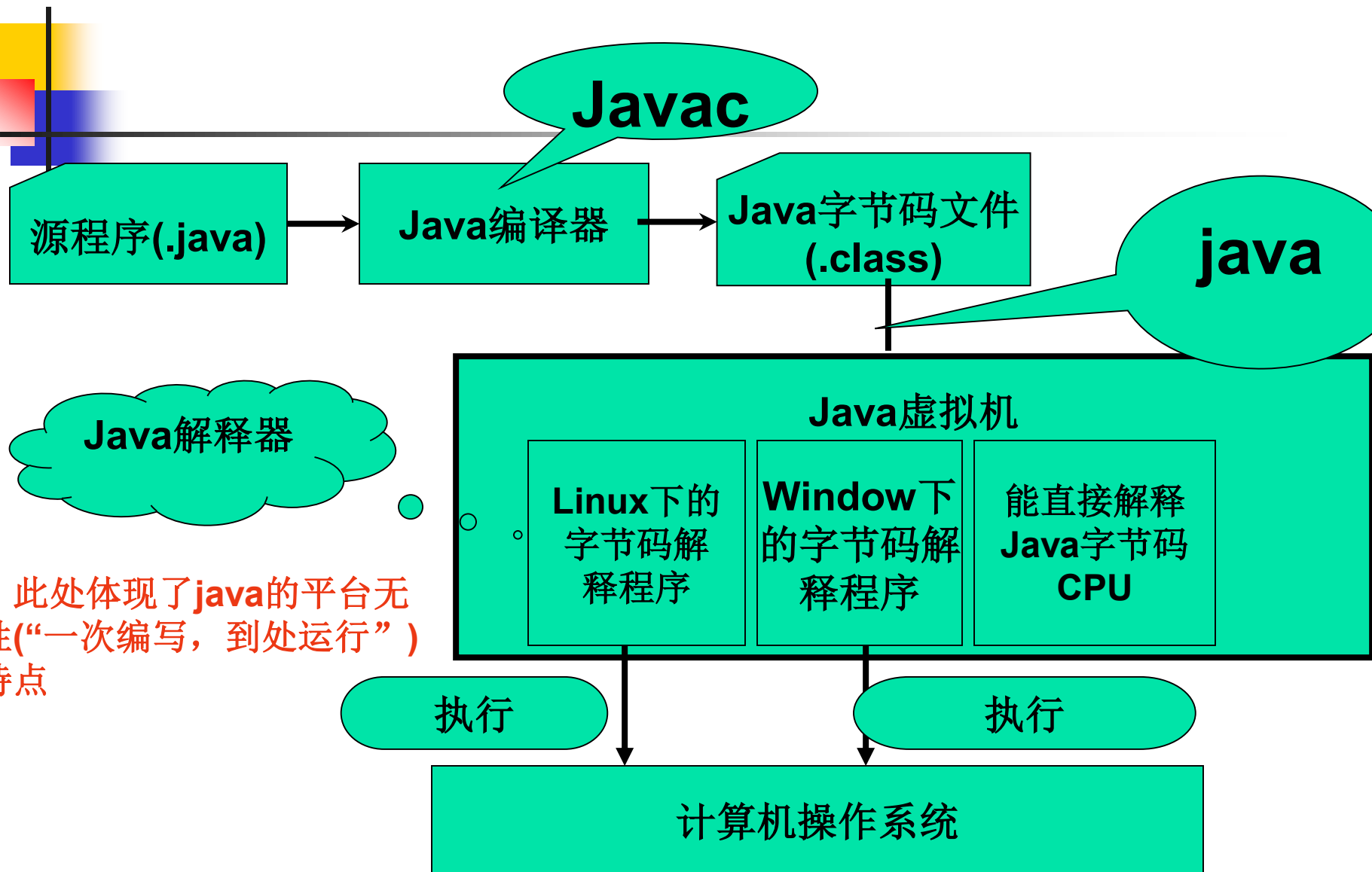
Eclipse下的main参数设置

The screenshot displays the Eclipse IDE interface with the following components:

- Package Explorer:** Shows the project structure with 'HelloJava' containing 'src' and 'HelloToXXX.java'.
- Source Editor:** Displays the code for 'HelloToXXX.java':

```
1 public class HelloToXXX
2 {
3     public static void main(String args[])
4     {
5         if(args.length==0)
6             System.out.println("Hello!");
7         else
8             for(int i=0; i<args.length; i++)
9                 System.out.println("Hello "+args[i]);
10    }
11 }
```
- Run Configurations Dialog:** A modal window titled 'Run Configurations' is open, showing the configuration for 'HelloToXXX'. The 'Main' tab is active, displaying:
 - Name:** HelloToXXX
 - Program arguments:** Students
 - VM arguments:** (empty)
 - Working directory:** Default: \${workspace_loc:HelloJava}
- Problems View:** Shows a message: '<terminated> HelloToXXX: Hello!Students'.

Java Application的执行过程:



注：此处体现了java的平台无关性(“一次编写，到处运行”)的特点

Java程序编译运行流程



(1) Application: -- 总结

- 易犯错误
- 引申问题



易犯错误：-----A

---文件名和类名不一致(应使用**public**修饰的类命名)

- **Java**区分大小写，不一致时以类名为准。



引申问题-----A 基础!!

- 一个**.java**源文件中只能有一个**public**修饰的类
- 但是可以有若干个非**public**修饰的类



易犯错误: -----B

---main方法声明错误

- **public static void**: 公开、静态、无返回值
- **String[] args**: 参数为字符串数组



引申问题 -----B

- **Java** 的**Application**的入口是**main**方法
- **main**方法声明及执行问题



引申问题 -----B

main参数是一个字符串数组，可以接受字符串输入。

main方法中的参数名字可以随便定义，但是不能为空。



引申问题 -----B:

- **main**方法修饰符可以没有**public**，但是必须有**static**。
- **static**修饰的类成员，由类可以直接访问，不需要实例化对象



易犯错误：-----C

---Java关键字拼写错误

- **Java**关键字全部小写。



补充:

关于Code Conventions for Java

- 官方规范 English Version:

<http://www.oracle.com/technetwork/java/codeconvtoc-136057.html>

- 阿里巴巴Java开发规范



简单补充 -----命名规范:

- 类名称：以**所有首字母开头大写**词，代表类功能。
---UserOperation
-
- 属性：以**小**写字母开头的名词其余单词首字母大写。**---strXXX,isXXX**等。
- 方法：以**小**写字母开头的动词或动宾词组，中间单词用**大**写开头
---getName(),setTitle(),isLogon()



简单补充 -----注释规范

☞ `//` `--`用于单行注释，注释从`//`开始，终止于该行行尾

☞ `/*..*/` `--`用于多行注释，这种注释不能互相嵌套

☞ `/**..*/` `--java`所特有的**doc**注释，主要是为了支持**JDK**中的**javadoc** 工具(**API**文档生成器)。



4 Java程序的编辑、编译与运行

(1)Application 编辑,编译与运行

(2)Applet编辑,编译与运行

(3)Application与Applet的区别



(2) Applet:

- 什么是**Applet**:

Applet就是使用**Java**语言编写的一段代码，它可以在**浏览器**环境中运行。

- **Applet**结构:

- **.java**源文件

- **.html**浏览器文件



(2) Applet:

---java源文件

```
import java.applet.*;           //包导入语句!!!
import java.awt.*;
public class AppletEx extends Applet //必须的!
{
    public void paint(Graphics g)
    {
        g.setColor(Color.red);
        g.drawString("我一边喝着咖啡，一边学Java呢",2,30);
        g.setColor(Color.blue);
        g.drawString("我学得很认真",10,50);
    }
}
```



(2) Applet:

---HTML文件

将Applet嵌入HTML

Html中Applet标记的完整语法格式为:

<Applet

code = AppletEx.class

width = Applet宽度

height = Applet高度

.....

>

[<param name=参数名 value=参数值>]

.....

</Applet>

保存为**myapplet.html**

(2) Applet:

创建步骤:

👉 编码:

1) 打开文本编辑器

(记事本、写字板、word均可)

2) 将**Applet**代码输入到编辑器中

3) 文件保存为**AppletEx.java**

(注: 一定要注意扩展名)

4) 在记事本中创建**myapplet.html**

(2) Applet:

创建步骤:

☞ 编译与运行:

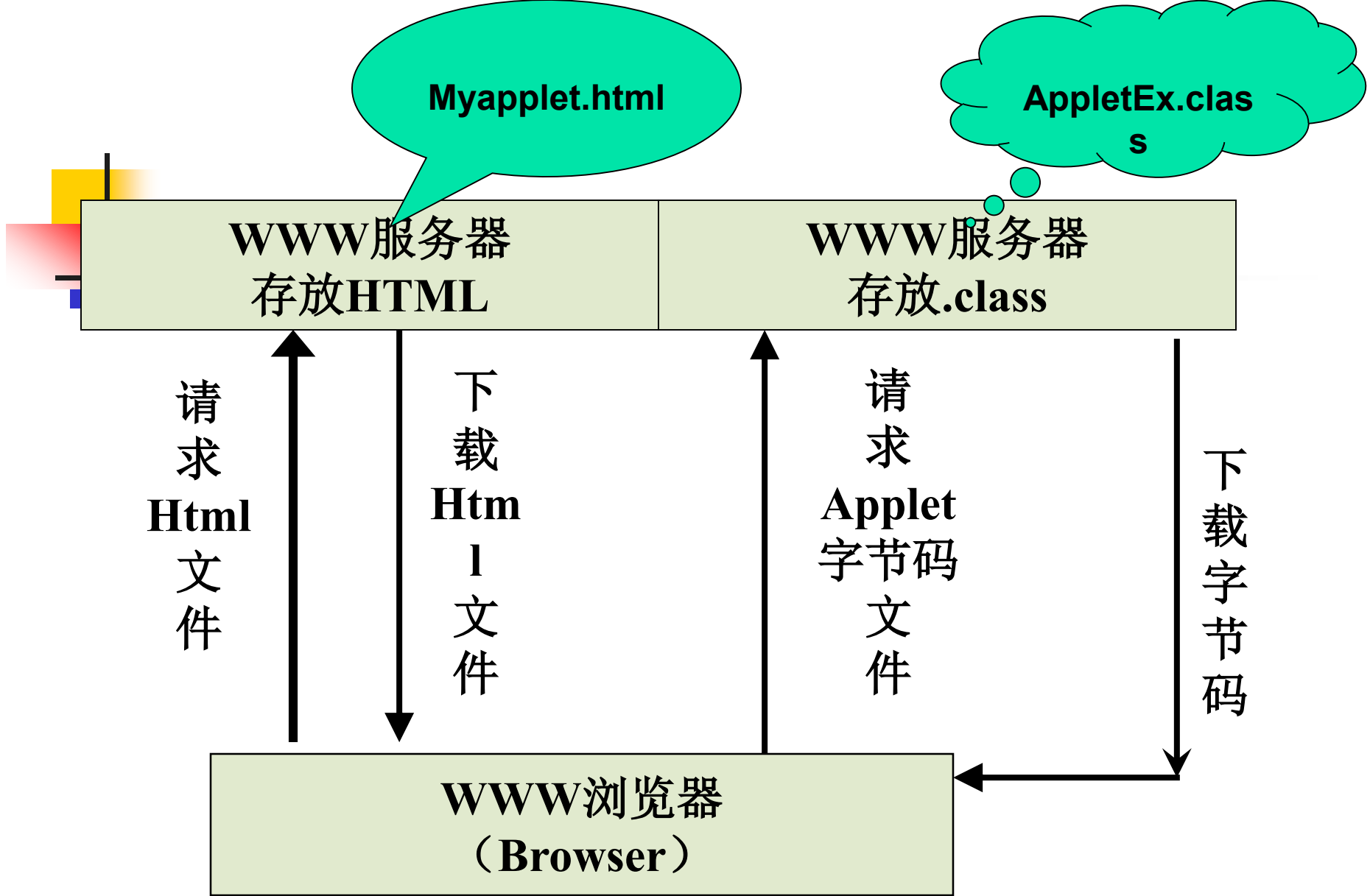
或在支持**Applet**的浏览器上运行**1)** 进入命令行运行模式

2) 进入保存**AppletEx.java**的目录

3) 运行**javac**编译源程序, **javac AppletEx.java**

4) 查看是否生成了字节码文件” **AppletEx.class**”

5) 执行程序, **appletviewer myapplet.html**



解释执行Html文件 解释执行字节码(plugin支持)

Applet执行过程



4 Java程序的编辑、编译与运行

(1)Application 编辑,编译与运行

(2)Applet编辑,编译与运行

(3)Application与Applet的区别



(3) 区别:

执行方式不同

- **application**是从其中的**main()**方法开始运行的
- **Applet**一般在浏览器中运行，必须创建一个**HTML**文件，通过编写**HTML**语言代码告诉浏览器载入何种**Applet**以及如何运行。

Java 程序执行过程比较

