

Java语言

主讲：张荣

授课计划

- 绪论
- 标识符、关键字、数据类型
- 表达式和流程控制
- 面向对象编程
- 类设计
- 字符串
- **Java**集合
- 数组

授课计划

- 例外处理
- 线程与同步
- 输入与输出
- GUI设计
- Applet
- 网络编程
- 设计模式
- UML

Java基础

本章内容

- 程序设计语言
- Java历史及发展
- Java语言特点
- Java开发环境
- Java类库
- Java程序

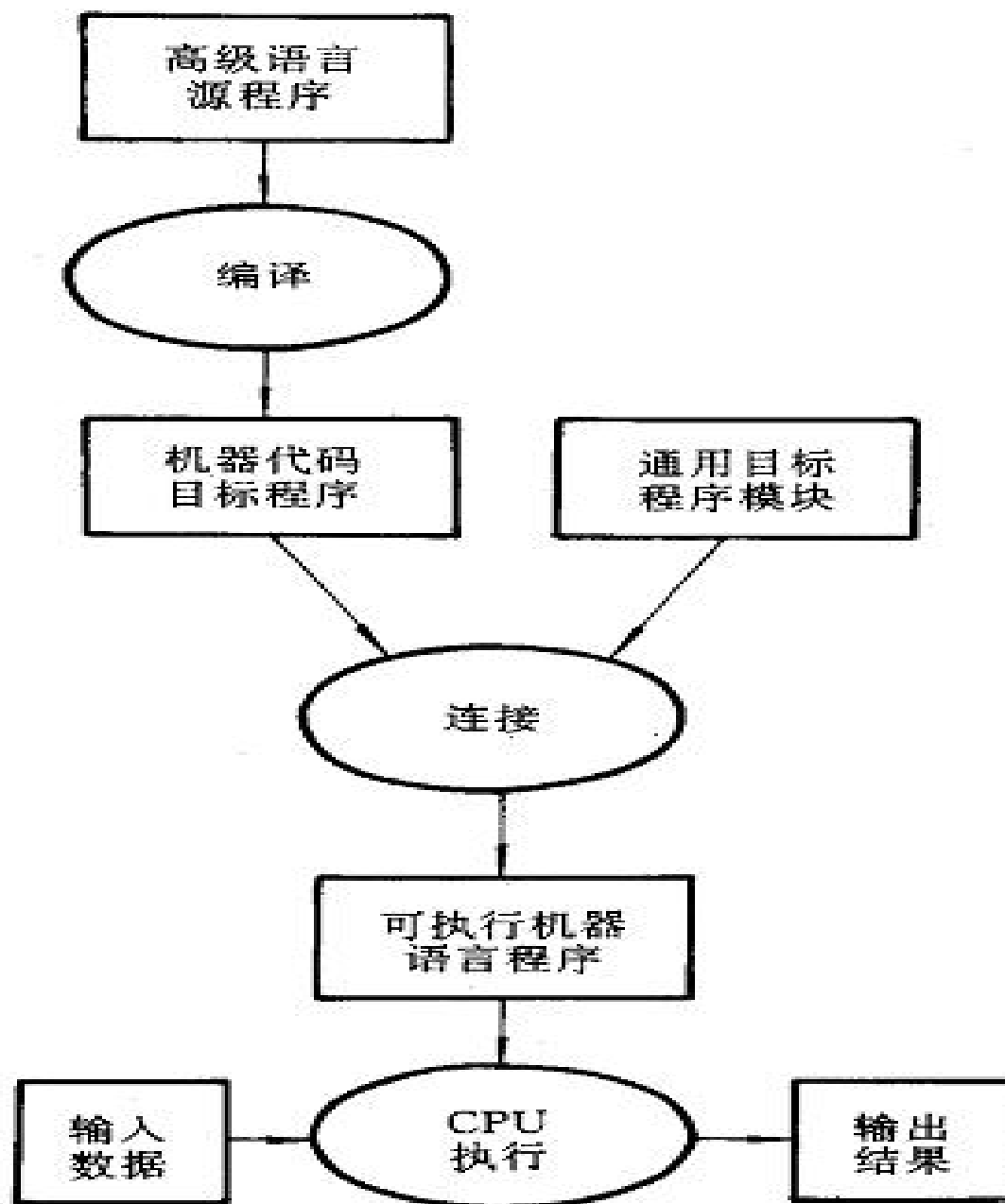
1 程序设计语言

用高级语言书写的程序不可能直接地在计算机上执行，要在计算机上执行高级语言书写的程序，有两种基本方法：

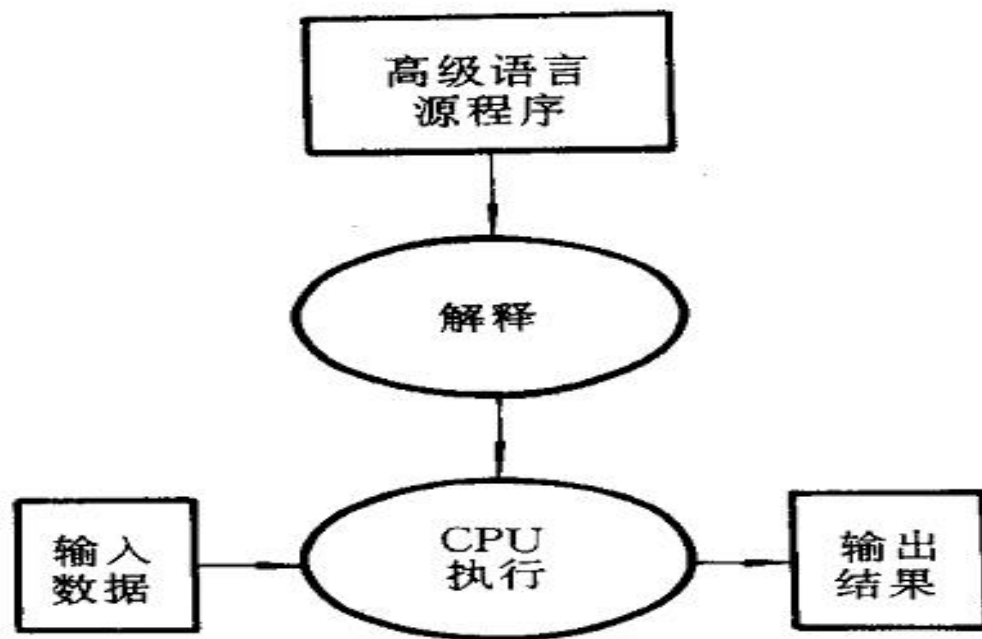
q 编译（Compilation）执行：这种方法是把源程序转换成为机器语言(计算机直接执行的程序)，即转变为“可执行（Executable）程序”。

q “解释”（Interpretation）执行：这种方法是**即时**把源程序转换为机器可执行的指令。

编译过程

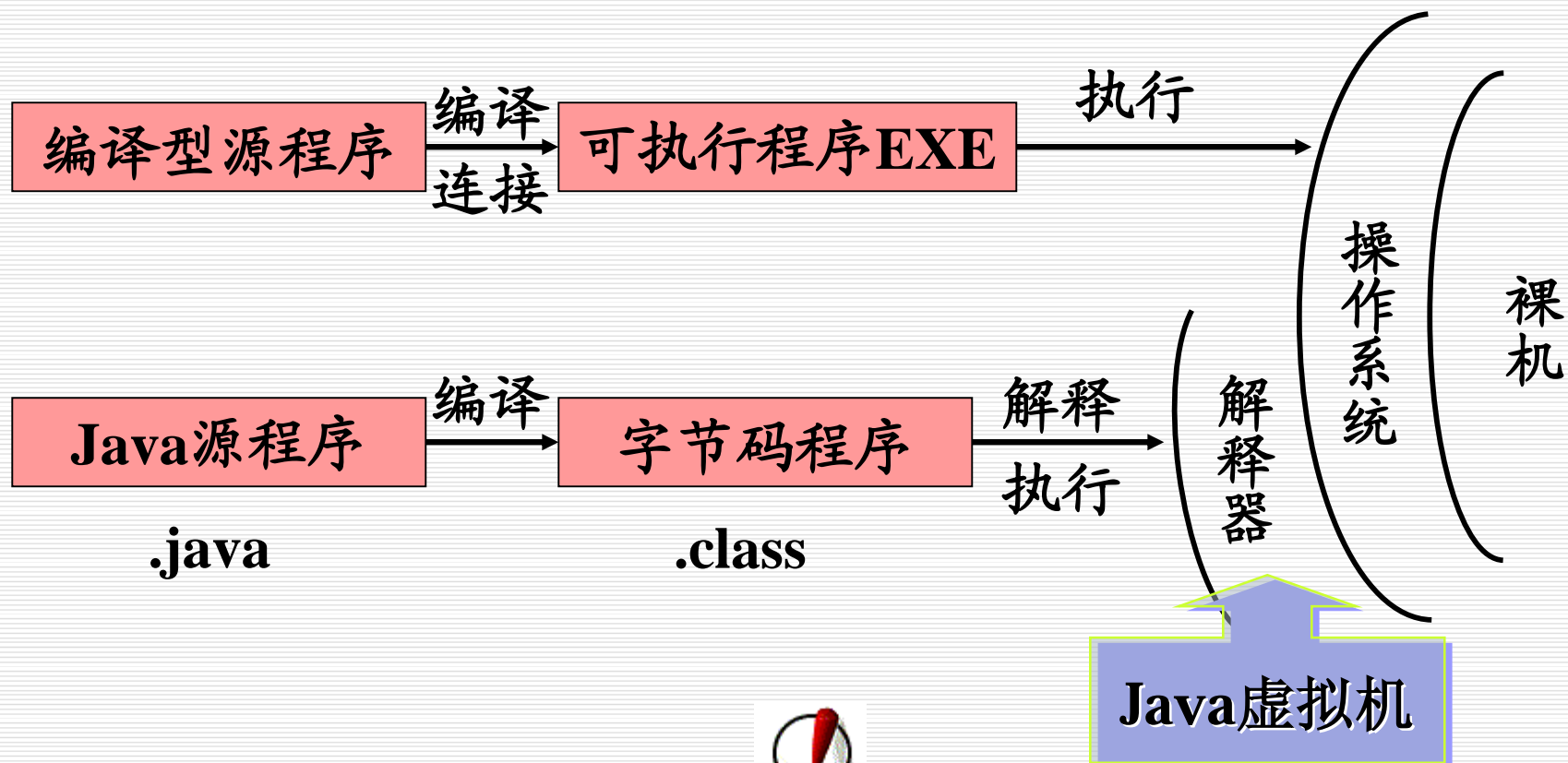


对于“解释执行”方式，人们实现了一种称为“解释器”（Interpreter）的软件来完成转换工作。解释器在工作方式上与编译器不同，它不对源程序进行翻译，而是直接对源程序的语句进行分析和解释，实现源程序所描述的功能。



解释过程

Java是解释执行的高级编程语言



2 Java历史及发展



JAVA 时代的到来

- 1990 年Sun 公司的 James Gosling等人开始开发名称为 Oak 的语言。希望用于控制嵌入在有线电视交换盒、PDA等的微处理器。
- 1993 年交互式电视和 PDA 市场开始滑坡，而 Internet 正处于增长时期，因此Sun 公司将目标市场转向 Internet 应用程序。
- 1994年将Oak语言更名为Java。
- 1995年Sun 公司的 HotJava 浏览器问世。

2 Java历史及发展

嵌入式系统

手机、PDA、电视机顶盒、冰箱、微波炉等消费类电子设备

J2ME(Java 2 Micro Edition)

传统桌面系统

单机应用软件

J2SE(Java 2 Standard Edition)

企业级应用系统

基于网络的分布式安全的信息系统

J2EE(Java 2 Enterprise Edition)



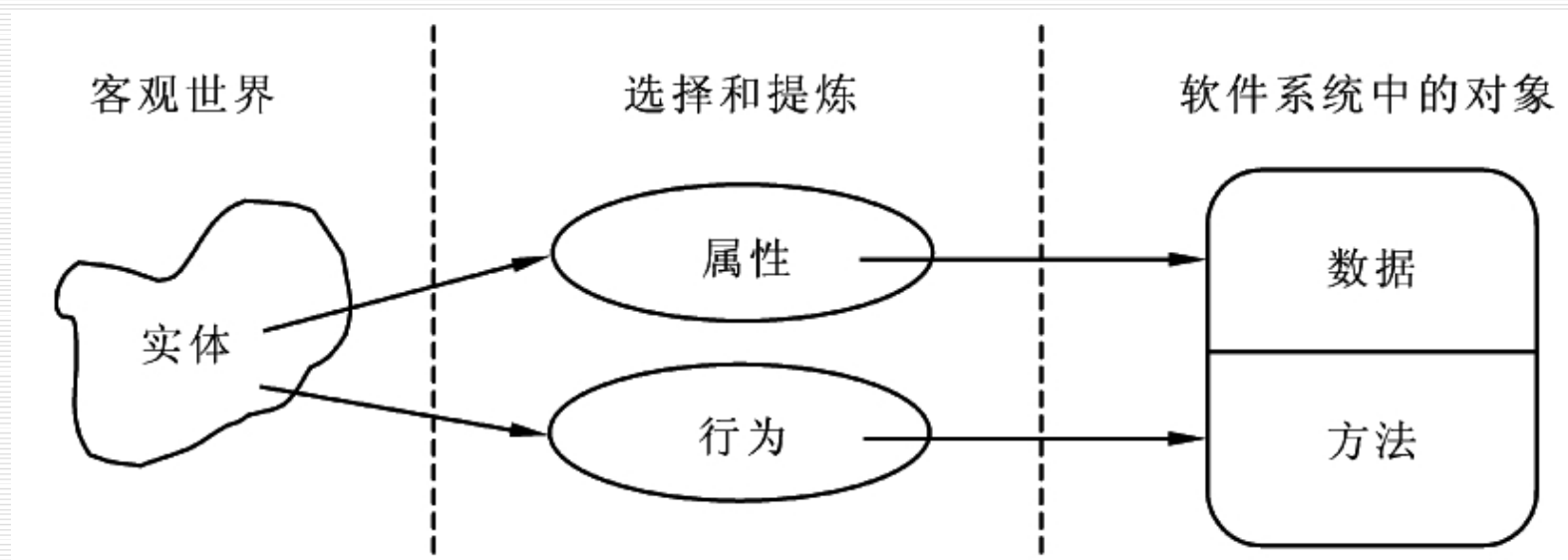
Java的特点

- 创建一种**面向对象**的程序设计语言，而不是面向过程的语言；
- 提供一个**解释执行**的程序运行环境，是程序代码独立于平台；
- 实现**多线程**，使得程序能够同时执行多个任务；
- 提供代码校验机制以保证安全性；

3 Java语言特点- (1) 面向对象

对象：

对象有两个层次的概念，现实生活中对象指的是可观世界的实体；而程序中对象就是一组变量和相关方法的集合，其中变量表明对象的状态，方法表明对象所具有的行为。



面向对象概念：

其实是现实世界模型的自然延伸。现实世界中任何实体都可以看作是对象。

对象之间通过消息相互作用。面向对象的编程语言则是以对象为中心以消息为驱动。

面向对象编程语言为：

程序=对象+消息。

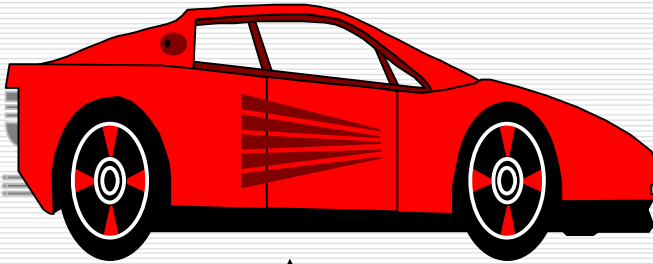
类:

相似的对象可以归并到同一个类中去，就像传统语言中的变量与类型关系一样。

具有相同或相似性质的对象的抽象就是类。因此，对象的抽象是类，类的具体化就是对象，也可以说类的实例是对象。

类具有属性，它是对象的状态的抽象，用数据结构来描述类的属性。类具有操作，它是对象的行为的抽象，用操作名和实现该操作的方法来描述。

抽象数据类型



现实生活中的实体

可以将现实生活中的对象经过抽象，映射为程序中的对象。对象在程序中是通过一种抽象数据类型来描述的，这种抽象数据类型称为类（Class）。

```
class Vehicle
{
    int color;
    int type;
    int speed;
    void start() { ... }
    void brake() { ... }
    void speedUp() { ... }
    void slowDown() { ... }
}
```

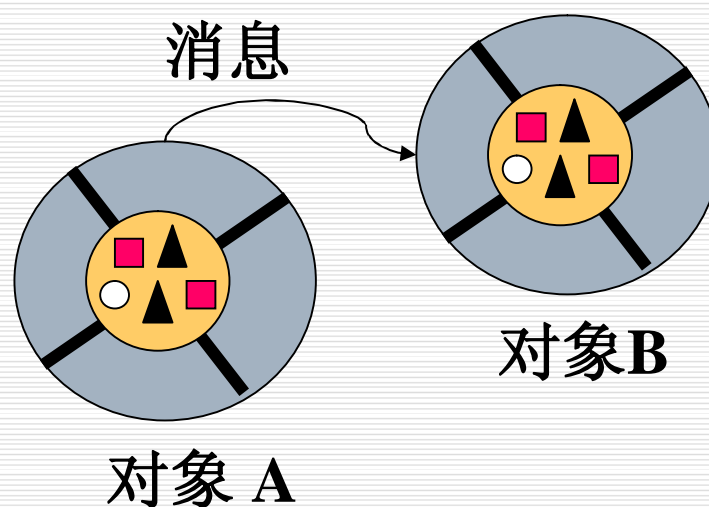



Vehicle car;
Vehicle truck;
... ..
Vehicle pcar;

消息:

对象通过相互间传递消息来相互作用和通信，一个消息由三部分组成:

1. 接受消息的对象
2. 接收对象要采取的方法
3. 方法需要的参数

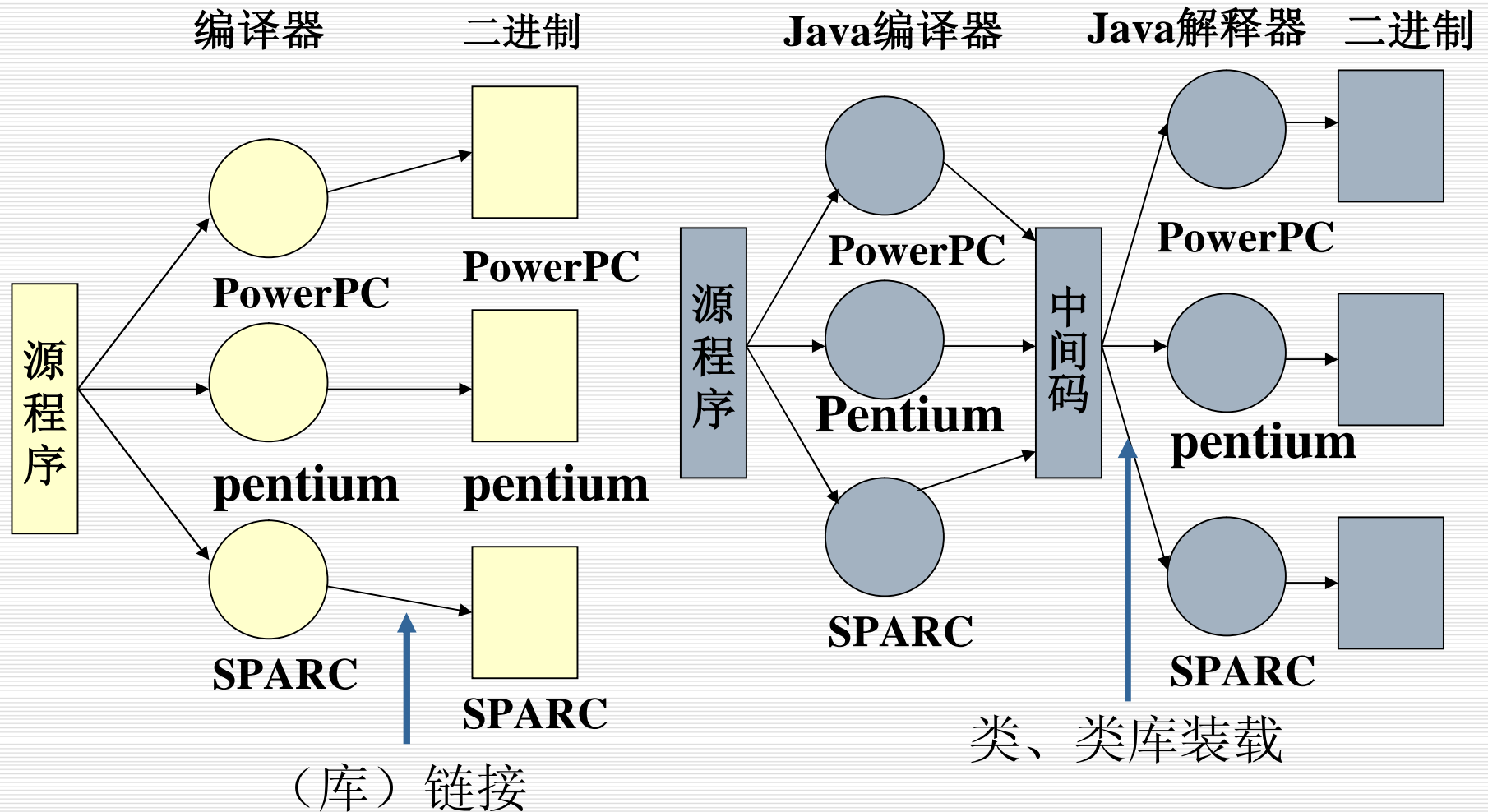


3 Java语言特点— (2)平台无关性

p “一次编写，到处执行”，即用Java语言编写的程序可以不作修改、**不重新编译**而在任何时候在任何一台计算机上正确运行。编译好的Java代码可以在网络上传输、并在不同的计算机平台(通常是指硬件与操作系统的组合)上运行。

p Java的这种平台无关性与其本身的运行方式是密切相关的。与其他计算机语言或是编译的或是解释的不同，Java是一种**半编译半解释的语言**。

编译型和解释型语言的工作模式

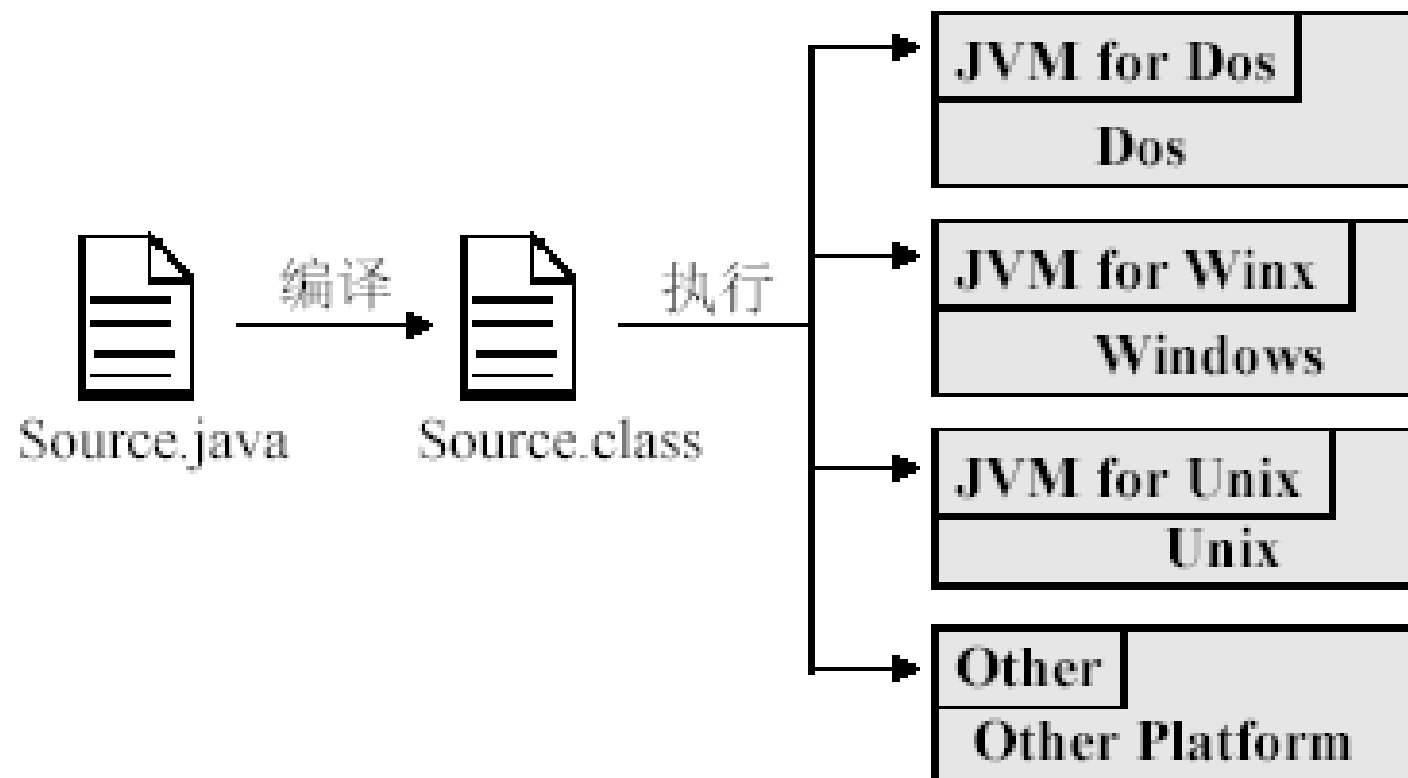


字节代码的平台无关性

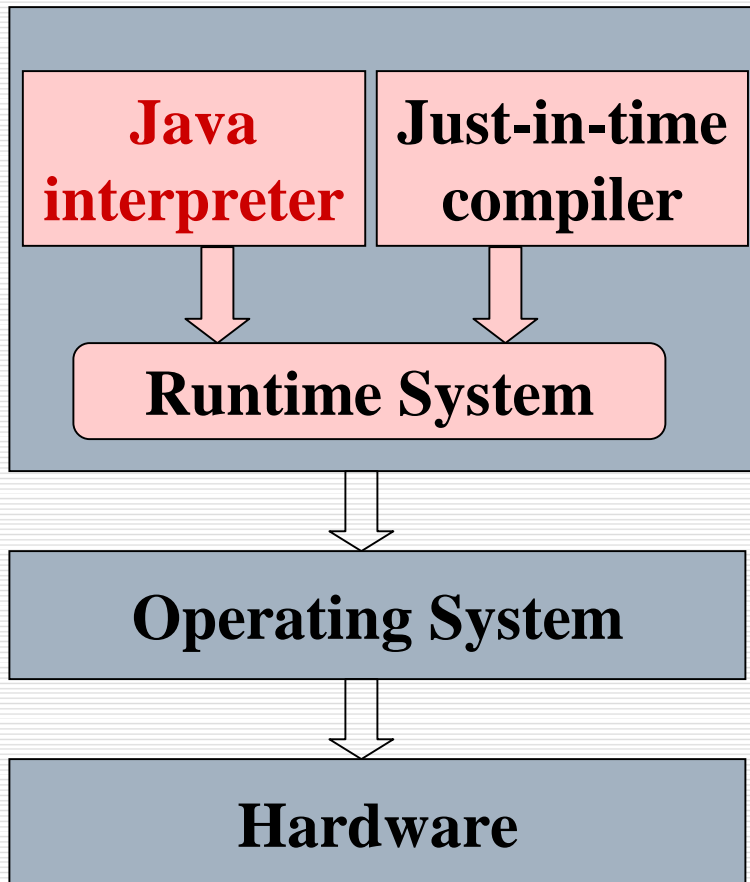
字节代码也就是Java虚拟机代码，它与具体的计算机处理器代码无关。

Java虚拟机(JVM, java virtual machine)是一个抽象的计算机处理器。由Java编译器产生的字节码文件(其扩展名为.class)就是在这个Java虚拟机上运行的目标程序。与某种具体的计算机处理器不同，Java虚拟机通常不是由硬件而是由软件实现的。这个软件就是Java解释器。

Java虚拟机



Java虚拟机(Java解释器)



不同的操作系统有不同的虚拟机。

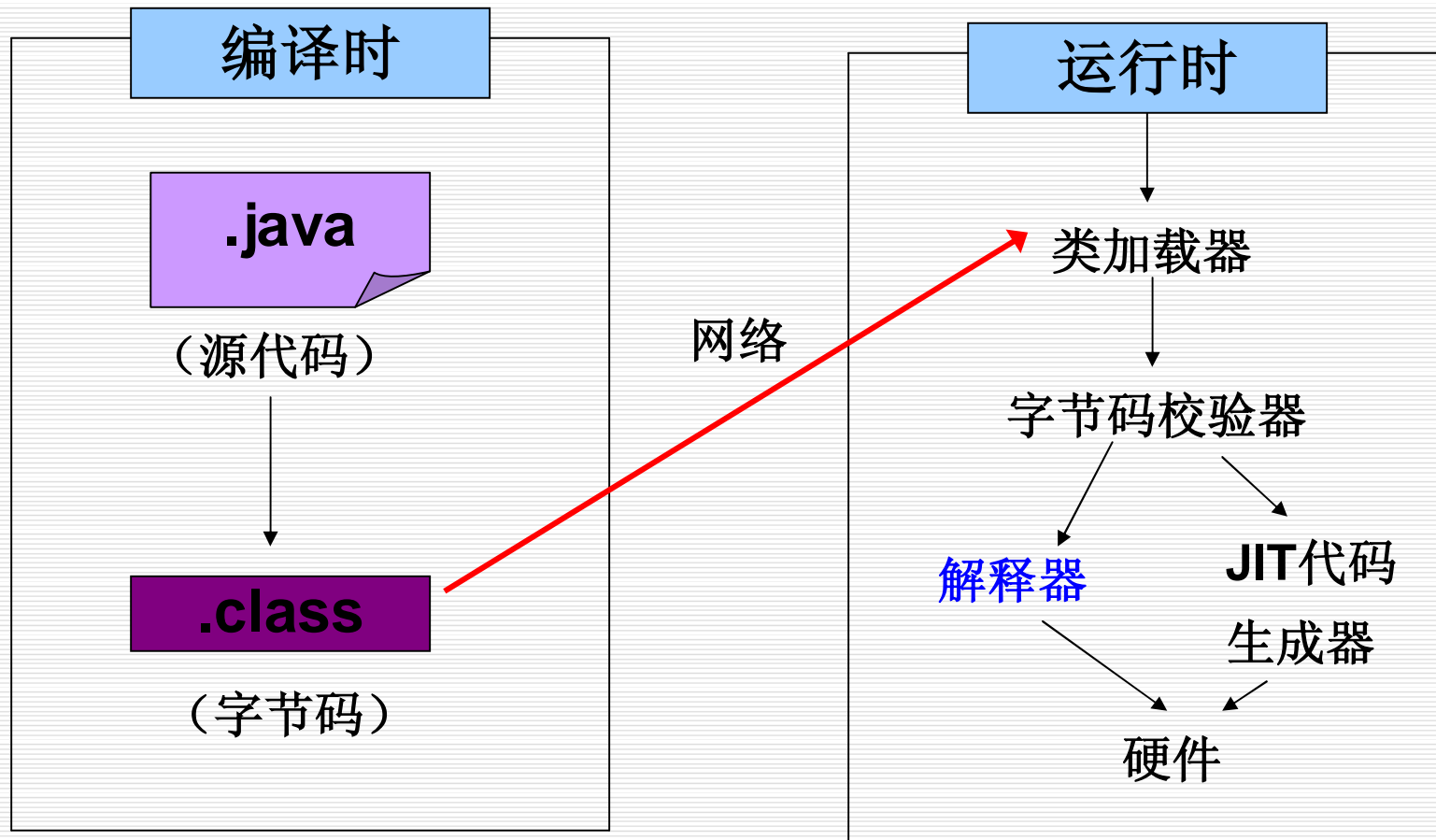
它类似一个小巧而高效的CPU。

Bytecode代码是与平台无关的是虚拟机的机器指令。

Java字节代码运行的两种方式:

Interpreter(解释方式)

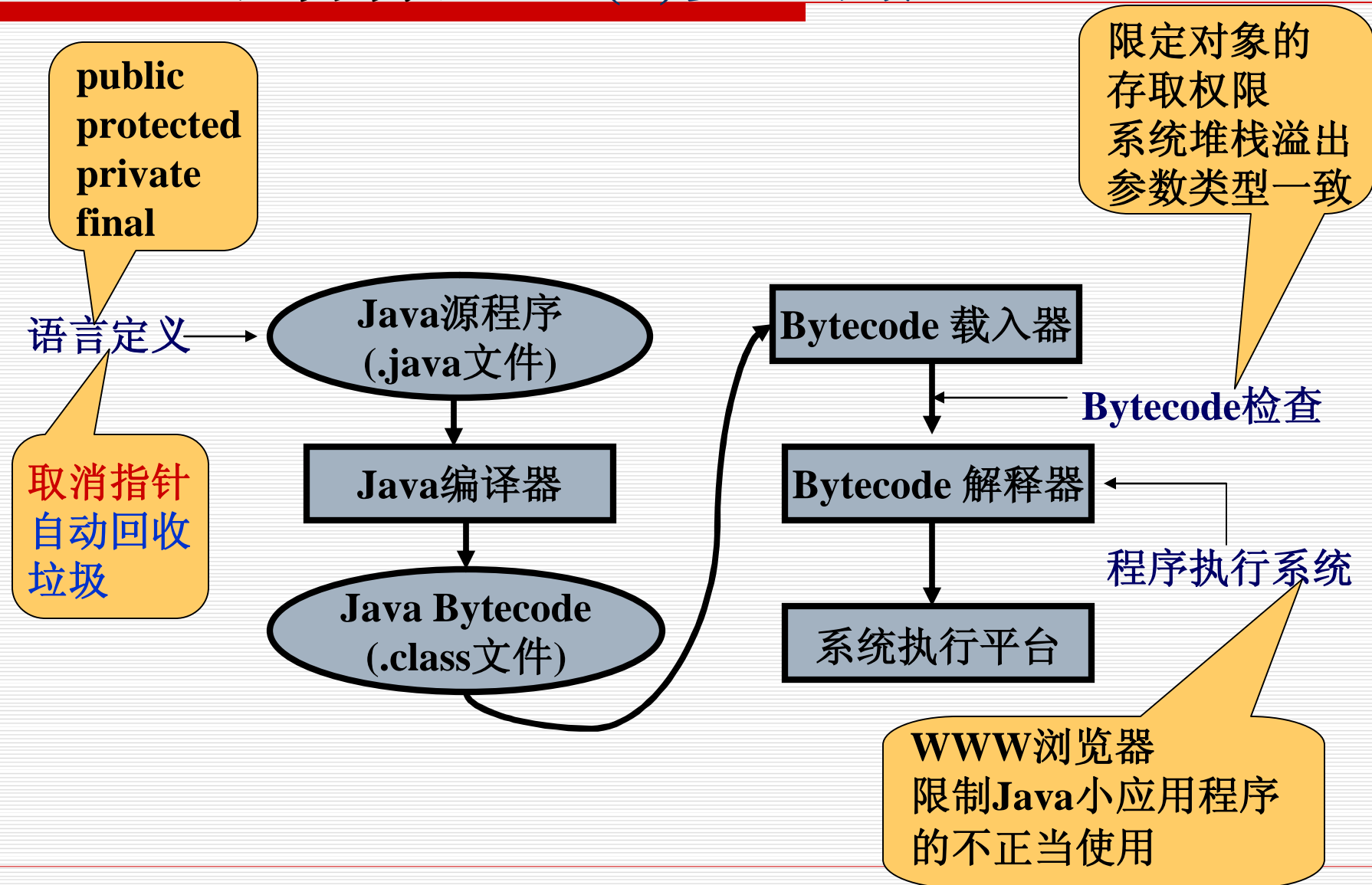
Just-in-time(即时编译):由代码生成器将字节代码转换成本机的机器代码,然后可以以较高速度执行。



Java虚拟机规范定义了：

- n 指令集
- n 寄存器集
- n 类文件结构
- n 堆栈
- n 垃圾收集堆
- n 内存区域

3 Java语言特点— (3)安全问题



3 Java语言特点— (4)多线程

q 单线程程序一个时间只能做一件事情，多线程程序允许在同一时间同时做多件事情。

q 其它大多数高级语言，包括C，C++等，都不支持多线程，只能编写顺序执行的程序（除非有操作系统API的支持）。

q Java提供现成的类Thread，只要继承这个类就可以编写多线程的程序。

3 Java语言特点— (5)与C及C++的区别

- 不再有全局变量
- 不再有#include 和#define 等预处理功能
- 不再有struct、union及typedef等
- 不再不再有指针、不再有多重继承
- 不再有goto语句
- 不再有操作符重载(Operator Overloading)
- 取消自动类型转换，要求强制转换
- 自动进行内存管理

3 Java语言特点—(6)垃圾回收机制(GC)

垃圾收集:不再使用的内存空间应回收

- 在C/C++等语言中, 由程序员负责回收无用内存。
- Java语言解除了程序员回收无用内存空间的责任。它提供一种系统级线程跟踪存储空间的分配情况。并在JVM的空闲时, 检查并释放那些可被释放的存储器空间。
- 垃圾收集在Java程序运行过程中自动进行, 程序员无法精确控制和干预。

3 Java语言特点—(6)垃圾回收机制(GC)

垃圾收集:不再使用的内存空间应回收

- Java 虚拟机使用两个独立的堆内存，分别用于静态内存分配和动态内存分配。
- 其中一个是非垃圾收集堆内存，用于存储所有类定义、常量池和方法表。
- 另一个堆内存再分为两个可以根据要求往不同方向扩展的小块。
- 用于垃圾收集的算法适用于存放在动态堆内存中的对象。

3 Java语言特点—(6)垃圾回收机制(GC)

垃圾收集:不再使用的内存空间应回收

- 垃圾收集器将在收集对象实例之前调用**finalize**方法。
- 即使显式调用垃圾收集**System.gc()**方法,也不能保证立即运行。
- 垃圾收集线程的运行优先级很低所以可能经常会被中断。

3 Java语言特点—其他

(7) 可移植性

(8) 分布性

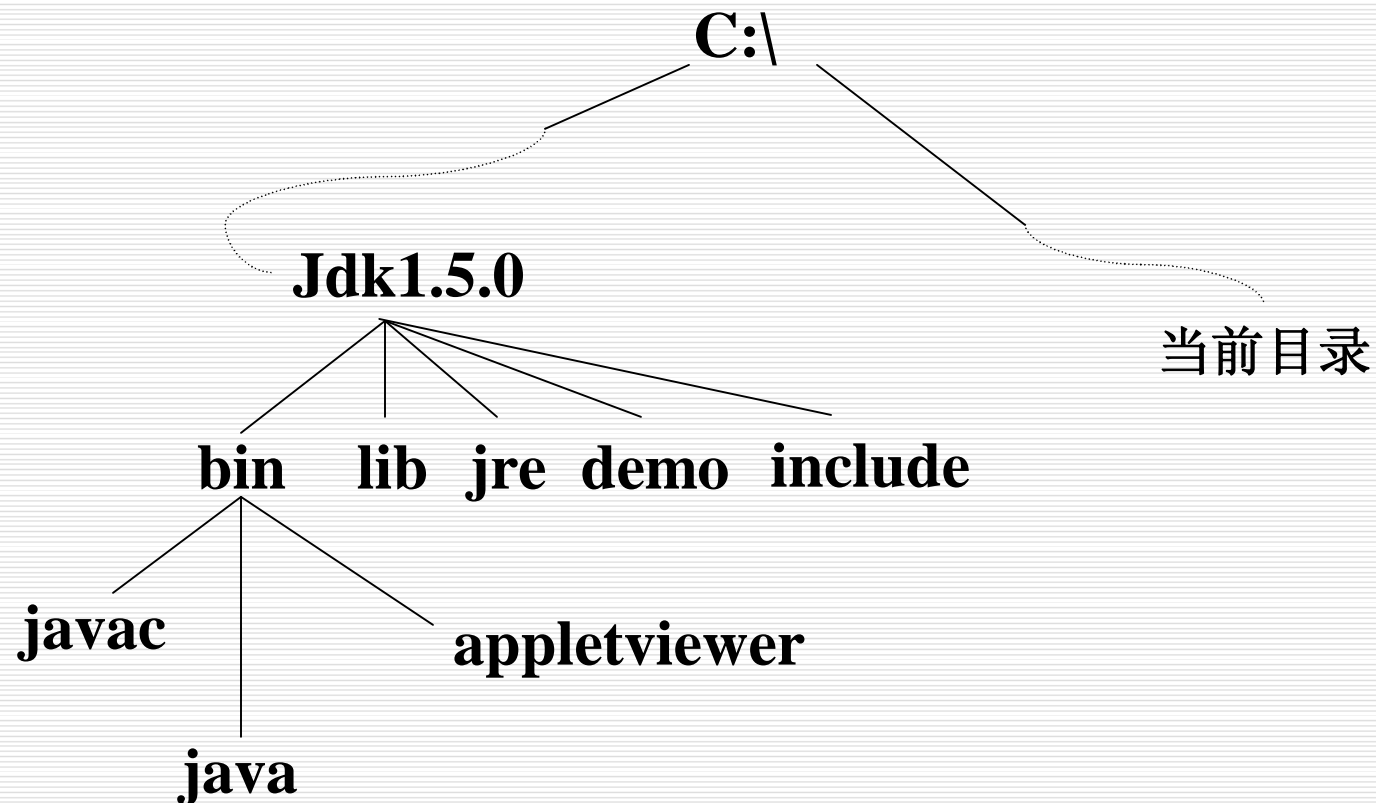
(9) 高性能（相对于其他解释型语言）

(10) 健壮性

4 Java开发环境

JDK (可从<http://java.sun.com>下载)

§ jdk1.5.0_04-win.exe



安装JDK后产生如下目录结构：

- \bin目录：Java开发工具，包括Java编译器、解释器等
- \demo目录：一些实例程序
- \lib目录：Java开发类库
- \jre目录：Java运行环境，包括Java虚拟机、运行类库等
- ...

Java开发工具包括:

- **Javac**: 编译器, 用来将java程序编译成Bytecode。
- **Java**: 解释器, 执行已经转换成Bytecode的java应用程序。
- **Jdb**: 调试器, 用来调试java程序。
- **Javap**: 反编译, 将类文件还原回方法和变量。
- **Javadoc**: 文档生成器, 创建HTML文件。
- **Appletviewer**: Applet解释器, 用来解释已经转换成Bytecode的java小应用程序。

○ 设置环境变量:

§ JAVA_HOME = c:\jdk1.5.0_04;

§ PATH = c:\jdk1.5.0_04\bin;

§ CLASSPATH= .;C:\jdk1.5.0_04\lib\dt.jar;
C:\jdk1.5.0_04\lib\tools.jar;C:\jdk1.5.0_04\jre\lib\rt.jar;

○ Windows NT/2000/XP:

控制面板 → 系统 → 高级 → 环境变量

5 Java类库

○Java程序由各种类（class）组成

- n 你自己编写的类
- n （Java class libraries——Java API）
 - n JDK——Java Development Kit(1.1~1.6)
 - n 独立软件供应商、免费软件/共享软件

○几点提示

- n 使用类库可以[提高软件重用/提高效率/提高移植性](#)
- n 网络上有许多开放的Java源码资源

6 Java开发工具

○ 常用开发工具

JDK（推荐初学者使用）

NetBeans

Eclipse

JBUILDER

Visual Age for Java

JCreator

○ 开发过程

编辑——编译——运行
(文本编辑器) (javac) (java)

7 Java程序

//这是名称为“HelloWorld.java”的简单程序

import java.io.*;

class HelloWorld {

public static void main(String args[])

{

System.out.println(“欢迎访问 Java 世界! ”);

}

}

单行注释

导入
必要的包

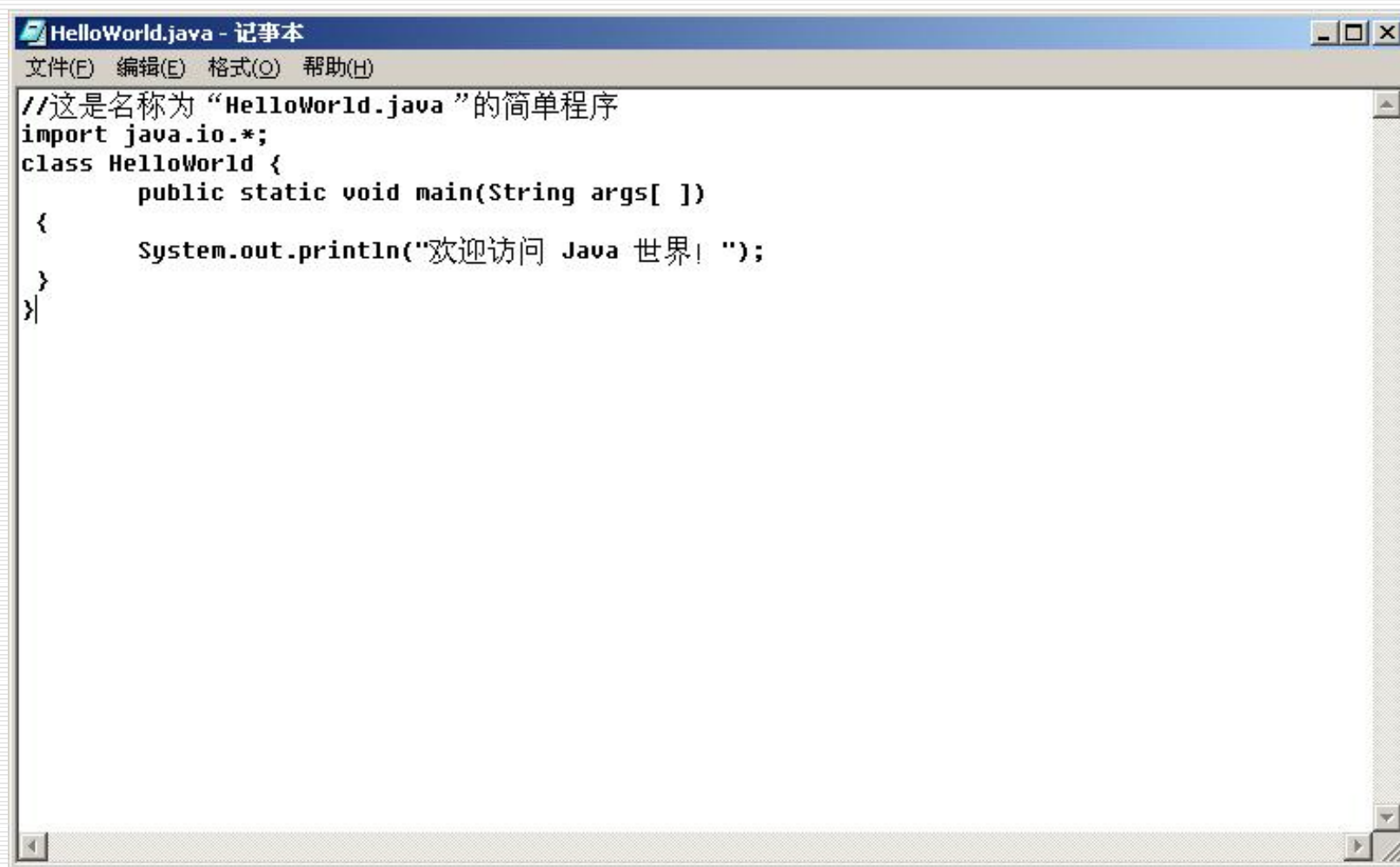
定义类

Main函数

在控制台上输出



在记事本编辑Java程序



```

HelloWorld.java - 记事本
文件(E) 编辑(E) 格式(O) 帮助(H)
//这是名称为“HelloWorld.java”的简单程序
import java.io.*;
class HelloWorld {
    public static void main(String args[ ])
    {
        System.out.println("欢迎访问 Java 世界! ");
    }
}

```


编译和运行

```
命令提示符
Microsoft Windows 2000 [Version 5.00.2195]
(C) 版权所有 1985-2000 Microsoft Corp.

C:\>cd java

C:\java>javac HelloWorld.java

C:\java>java HelloWorld
欢迎访问 Java 世界!

C:\java>
```

输出结果

开始

7.1 Java程序—第一个Java应用程序

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

- n 编辑存盘：文件名和公共类名(用public声明)要一致 HelloWorldApp.java
- n 编译程序：javac HelloWorldApp.java
- n 运行程序：java HelloWorldApp
- n 运行结果：Hello World!

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

- p** 声明一个类： `public class HelloWorldApp{}`，类名第一个字母大写。
- p** 一个类中可有很多方法，`main`方法是运行程序的第一个方法，方法名的第一个字母小写。
- p** `System.out.println`是向屏幕输出，相当于C中的`printf()`。

8 Java程序结构

Java程序结构:

- p package语句:** 零个或一个, 必须放在文件开始
- p import语句:** 零个或多个, 必须放在所有类定义之前
- p Interface Definition:** 零个或多个
- p public ClassDefinition:** 零个或一个
- p Class Definition:** 零个或多个

-
- ❑ 源文件命名：若有**public**类，源文件必须按该类命名
 - ❑ 标识符：区分大小写
 - ❑ 类定义：定义程序所需的类及接口，包括其内部的变量、方法等。
 - ❑ **main()**方法：应用程序的入口，与标准C中**main()**函数的地位是一样的。一个应用程序有且只有一个**main()**方法，**main()**方法必须包含在一个类中，该类即为应用程序的外部标志。
 - ❑ 程序注释：与C++类似，`/*...*/` 或者 `//...`

新手常见错误

编译时： HelloWorld.java:1: class helloworld is public, should be declared in a file named helloworld.java

n public class helloworld

n ^

n 1 error

这种错误是因为你的类名（就是源程序中紧跟在关键字**class**中的那个）和程序名称不一致，注意，**Java**是大小写敏感的。

运行时: **Exception in thread "main"**
java.lang.NoSuchMethodError: main

这种错误是因为没有定义一个main方法，它的基本结构如下：

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

运行时: Exception in thread "main"

java.lang.**NoClassDefFoundError**:helloWorld (wrong name:
HelloWorld)

运行时, 系统**没有找到这个要执行**的文件。原因:

没有将要执行的这个文件名称写对, 比如, 在这边如果想执行这个HelloWorld应用, 则必须在控制台中输入:

java HelloWorld

并且严格注意大小写。

Java编程规范

包名：

包名是全小写的名词，中间可以由点分隔开，例如：

java.awt.event;

类名：

首字母大写，通常由多个单词合成一个类名，要求每个单词的首字母也要大写，例如

class HelloWorldApp

接口名：

命名规则与类名相同，例如

interface Collection

Java编程规范

方法名：

往往由多个单词合成，第一个单词通常为动词，首字母小写，中间的每个单词的首字母都要大写，例如：

balanceAccount, isButtonPressed, getStringLength, setSquareWidth等；

变量名：

全小写，一般为名词，例如：**length**；

常量名：

基本数据类型的常量名为全大写，如果是由多个单词构成，可以用下划线隔开，例如：**int YEAR, int WEEK_OF_MONTH**；如果是对象类型的常量，则是大小写混合，由大写字母把单词隔开。

使用Scanner取得输入

在J2SE 5.0中，可以使用`java.util.Scanner`类取得使用者的输入

```
Scanner scanner = new Scanner(System.in);  
System.out.print("请输入您的名字: ");  
System.out.printf("Hello! %s!\n", scanner.next());
```

可以使用这个工具的`next()`功能，来取得用户的输入字符串

```
System.out.print("请输入一个数字: ");  
System.out.printf("您输入了%d! \n",  
                    scanner.nextInt());
```

使用BufferedReader取得输入

n 可使用java.io.InputStreamReader
BufferedReader bufferedReader =
new BufferedReader(
new InputStreamReader(System.in));
System.out.print("请输入一系列文字，可包括空白：
");
String text = bufferedReader.readLine();
System.out.println("您输入的文字：" + text);

输出格式控制

若是使用J2SE5.0或更高的版本

//输出19的十进制表示

```
System.out.printf("%d%n", 19);
```

//输出19的八进制表示

```
System.out.printf("%o%n", 19);
```

//输出19的十六进制表示

```
System.out.printf("%x%n", 19);
```

格式字符	作 用
%%	在字符串中显示%
%d	以10进位整数方式输出，提供的数必须是Byte、Short、Integer、Long、或BigInteger
%f	将浮点数以10进位方式输出，提供的数必须是Float、Double或BigDecimal
%e, %E	将浮点数以10进位方式输出，并使用科学记号，提供的数必须是Float、Double或BigDecimal
%a, %A	使用科学记号输出浮点数，以16进位输出整数部份，以10进位输出指数部份，提供的数必须是Float、Double、BigDecimal
%o	以8进位整数方式输出，提供的数必须是Byte、Short、Integer、Long、或BigInteger
%x, %X	将浮点数以16进位方式输出，提供的数必须是Byte、Short、Integer、Long、或BigInteger
%s, %S	将字符串格式化输出
%c, %C	以字符方式输出，提供的数必须是Byte、Short、Character或 Integer
%b, %B	将"true"或"false"输出（或"TRUE"、"FALSE"，使用 %B）。另外，非null值输出是"true"，null值输出是"false"
%t, %T	输出日期/时间的前置，详情请看在线API文件

输出格式控制

可以在输出浮点数时指定精度

```
System.out.printf("example:%.2f%n", 19.234);
```

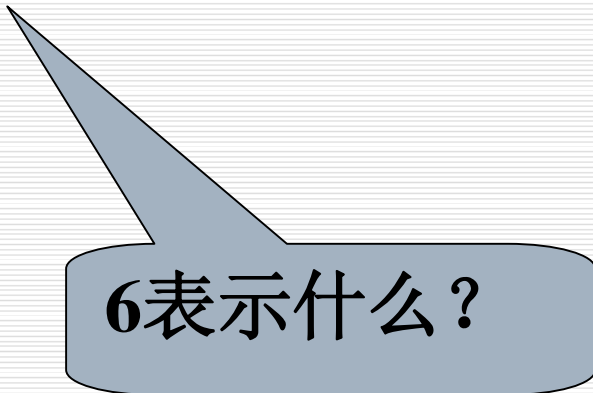
example:19.23

可以指定输出时，至少要预留的字符宽度

```
System.out.printf("example:%6.2f%n", 19.234);
```

example: 19.23

补上一个[↑]空白在前端



6表示什么？

总结

- **JDK6 API**的使用
- **bin**目录下的命令用法
- **Eclipse**用法

下课! 😊

Thank you!