1. 初识Java

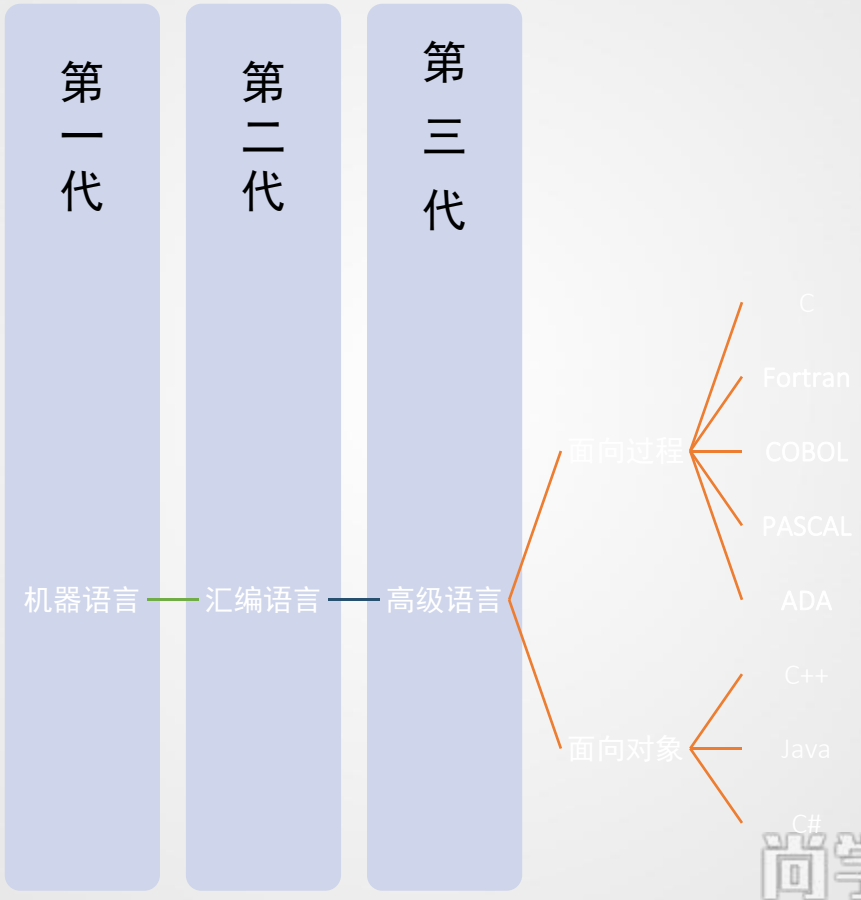
CONTENTS

- Java历史和三大版本
- Java特点
- Java跨平台原理（ 虚拟机 字节码文件 ）
- Java开发过程 编译 解释
- 环境变量的配置
- Java注释类型
- 反编译工具

本章技能点列表

技能点名称	难易程度	认知程度	重要程度
Java历史和三大版本	易	记忆	*
Java特点	易	了解	*
Java跨平台原理 虚拟机 字节码文件	中	理解	***
DOS命令入门	中	应用	*
Java开发过程 编译 解释	难	应用	***
环境变量的配置	难	应用	**
Java注释类型	易	记忆	**
反编译工具	易	了解	*

计算机语言发展历史



Java简史

- SUN公司是一家什么样的公司?

  美国SUN(Stanford University Network)公司

  在中国大陆的正式中文名为“太阳计算机系统（中国）有限公司”

  在台湾中文名为“升 阳电脑公司”。

- Java为什么被发明?

  Green项目。

  应用环境：像电视盒这样的消费类电子产品

  要求：

  语言本身是中立的，也就是跨平台

- Java的发明人?

  James Gosling

Java简史

- 1991年，Sun公司的Green项目，Oak
- 1995年，推出Java测试版
- 1996年，JDK1.0
- 1997年，JDK1.1
- 1998年，JDK1.2，大大改进了早期版本的缺陷，是一个革命性的版本，更名为Java2
- 1999 Java被分成J2SE、J2EE 和J2ME，JSP/Servlet技术诞生
- 2004年，J2SE 5.0 (1.5.0) Tiger老虎.为了表示这个版本的重要性，J2SE1.5更名为J2SE5.0。
- 2006年，J2SE 6.0 (1.6.0) Mustang野马.此时，Java的各种版本被更名，取消其中的数字"2"：J2EE更名为Java EE, J2SE更名为Java SE，J2ME更名为Java ME
- 2009年4月20日甲骨文收购Sun公司，交易价格达74亿美元
- 2011年，JavaSE7.0
- 2014年，JavaSE8.0

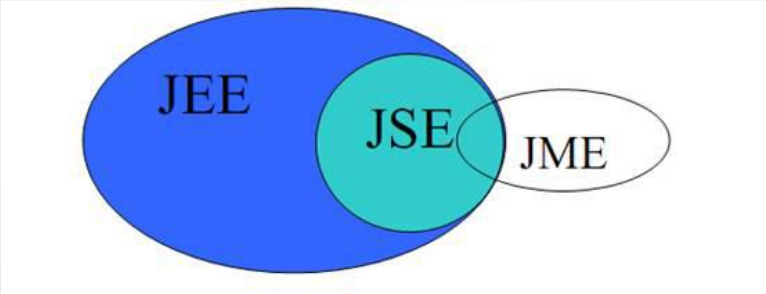
Java简史

- Sun公司的主要竞争对手是IBM，业务高度重合
- Sun抱有很多先进的技术，但在策略上一直奉行技术保护主义，在定价策略上能多高就多高，争取利润最大化。一味抓眼前利益，而失去了前瞻的视野。
- Sun是一家极具创新能力的公司， 但是没能利用Java构建一个强有力、可变现的生态系统，没打好Java这张牌。

- 2008年金融危机给sun公司致命的打击
- 2009年4月20日甲骨文以现金收购Sun微系统公司，交易价格达74亿美元

Java三大版本

- J2SE Java的标准版本 (Java2 Standard Edition) 定位在客户端，主要用于桌面应用软件的编程
- J2ME (Java2 Micro Edition) 主要应用于嵌入式系统开发，如手机和PDA的编程
- J2EE 企业版本(Java2 Enterprise Edition)定义在服务器端Java2的企业版，主要用于分布式网络程序的开发，如电子商务网站
- 2005 JavaOne大会召开，Sun公司公开Java SE6。此时，Java的各种版本被更名，取消其中的数字"2"：J2EE更名为Java EE, J2SE更名为Java SE，J2ME更名为Java ME



Java特点

- Java是跨平台的
- Java是简单的
- Java是安全的
- Java是完全面向对象的
- Java是健壮的

Java特点

- Java是跨平台的
  - Java程序的跨平台主要是指字节码文件可以在任何具有Java虚拟机的计算机或者电子设备上运行，Java虚拟机中的Java解释器负责将字节码文件解释成为特定的机器码进行运行。
- Java是简单的
  - 不再有#include 和#define 等预处理功能
  - 不再有struct,union及typedef
  - 不再有函数、
  - 不再有指针、不再有多重继承
  - 不再有goto
  - 不再有操作符重载(Operatior Overloading)
  - 不再有全局变量　　取消自动类型转换,要求强制转换
  - 不再有手动内存管理

Java特点

- Java是安全的
  - Java取消了强大但又危险的指针。由于指针可进行移动运算，指针可随便指向一个内存区域，而不管这个区域是否可用，这样做是危险的，因为原来这个内存地址可能存储着重要数据或者是其他程序运行所占用的， 并且使用指针也容易数组越界。
  - Java提供了自动内存管理机制，由垃圾回收器在后台自动回收,
  - Java在字节码的传输过程中使用了公开密钥加密机制(PKC)。
  - 而在运行环境提供了四级安全性保障机制：
  - 字节码校验器 -类装载器 -运行时内存布局 -文件访问限制

Java特点

- Java是完全面向对象的
  - Java和C++都是面向对象语言。也就是说，它们都能够实现面向对象思想（封装，继承，多态）。
  - 由于C++为了照顾大量C语言使用者而兼容了C，使得自身仅仅成为了带类的C语言，多少影响了其面向对象的彻底性！
  - Java则是完全的面向对象语言，它句法更清晰，规模更小，更易学。它是在对多种程序设计语言进行了深入细致研究的基础上，据弃了其他语言的不足之处，从根本上解决了c++的固有缺陷。

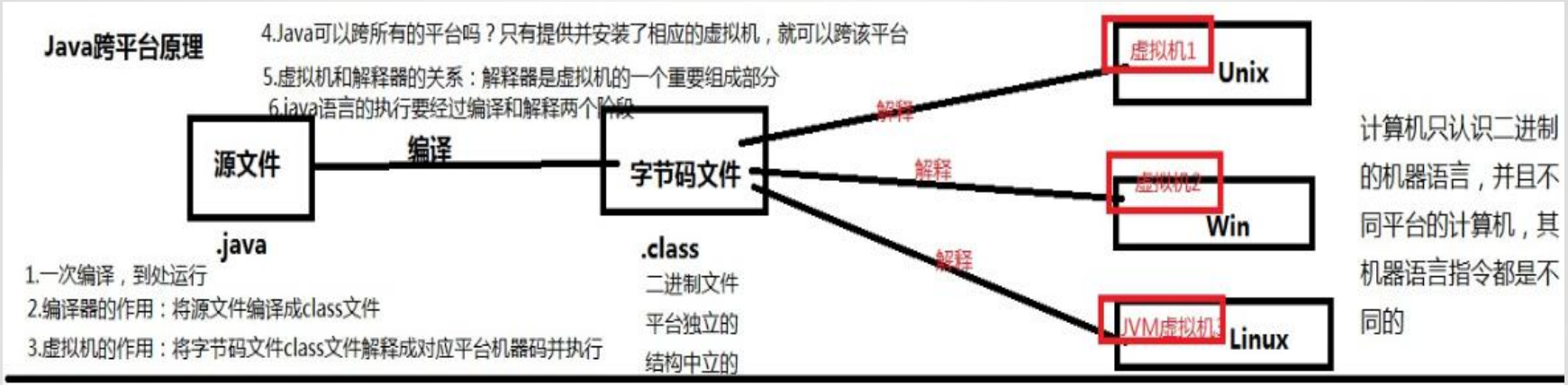
Java特点

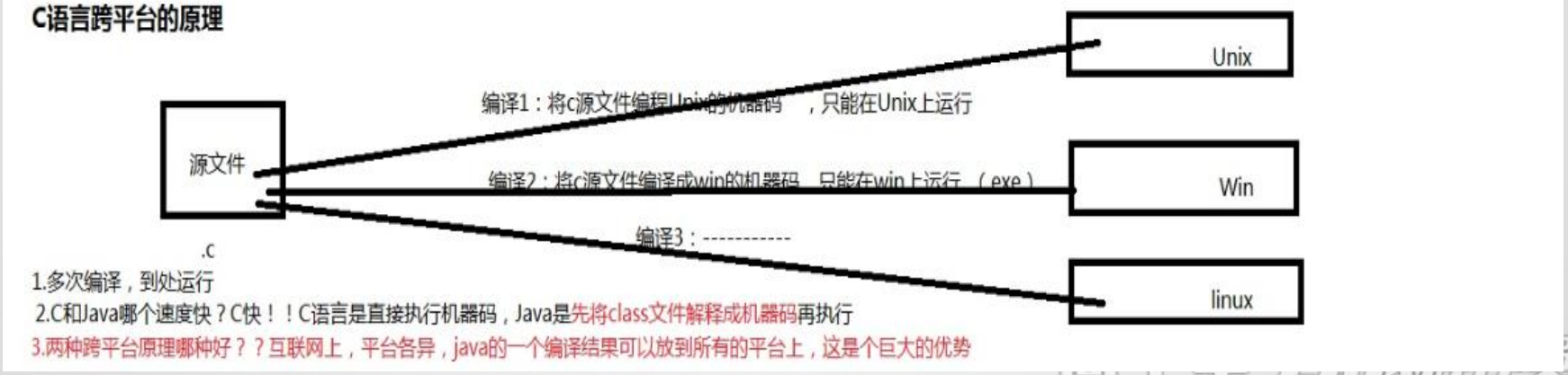
- Java是健壮的
  - Java的强制类型机制、异常处理、垃圾的自动收集等是Java程序健壮性的重要保证。
  - 对指针的丢弃是Java的明智选择。
  - Java的安全检查机制使得Java更具健壮性。

Java跨平台原理

- Java为什么能够流行
  - 外部环境
    - 互联网的爆发式发展
    - 互联网上的电脑硬件不同，软件环境差异较大。需要一个跨平台的语言。
  - Java核心优势：
    - 跨平台

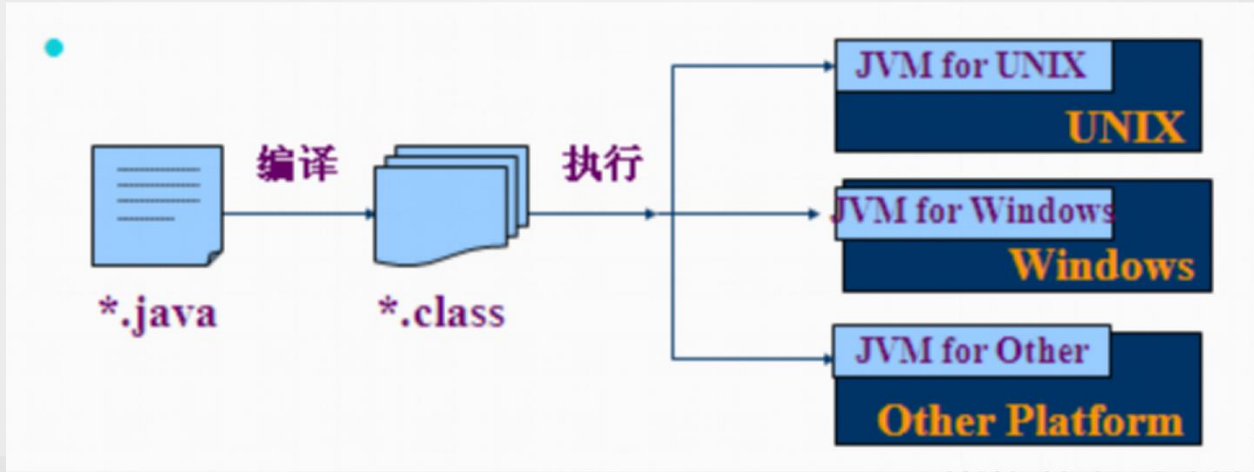
Java跨平台原理


Java跨平台原理
4.Java可以跨所有的平台吗？只有提供并安装了相应的虚拟机，就可以跨该平台
5.虚拟机和解释器的关系：解释器是虚拟机的一个重要组成部分
6.java语言的执行要经过编译和解释两个阶段
源文件
.java
编译
字节码文件
.class
二进制文件
平台独立的
结构中立的
解释
解释
解释
虚拟机1
Unix
虚拟机2
Win
JVM虚拟机3
Linux
计算机只认识二进制的机器语言，并且不同平台的计算机，其机器语言指令都是不同的
1.一次编译，到处运行
2.编译器的作用：将源文件编译成class文件
3.虚拟机的作用：将字节码文件class文件解释成对应平台机器码并执行


C语言跨平台的原理
源文件
.c
编译1：将c源文件编程Unix的机器码 ，只能在Unix上运行
编译2：将c源文件编译成win的机器码，只能在win上运行（exe）
编译3：----------
Unix
Win
linux
1.多次编译，到处运行
2.C和Java哪个速度快？C快！！C语言是直接执行机器码，Java是先将class文件解释成机器码再执行
3.两种跨平台原理哪种好？？互联网上，平台各异，java的一个编译结果可以放到所有的平台上，这是个巨大的优势

Java跨平台原理

- 总结1：Java运行过程
  - Java程序的运行分为两步：先编译再解释执行
  - 通过“编译器”将Java源程序编译成Java 字节码文件（.class）(字节码文件采用结构中立的中间文件格式)
  - 通过不同的“虚拟机”将Java字节码文件解释为对应机器语言并执行



Java跨平台原理

- 总结2：Java跨平台和C跨平台的区别
  - Java：一次编译，到处运行　C：多次编译，到处运行
  - 在互联网情况下，平台各异，Java的跨平台更具有优势
  - Java可以跨所有平台吗：要看有没有提供并安装相应的虚拟机
  - Java的运行速度没有C语言快
  - Java需要将class文件解释成机器码再执行；C执行执行机器码

- 总结3：字节码文件bytecode
  - .class文件 二进制文件
  - 格式中立、平台无关的二进制文件
  - 是编译的产物，是解释的原料

Java跨平台原理

- 总结4：Java虚拟机 JVM
  - JVM是Java Virtual Machine（Java虚拟机）的缩写
  - JVM是一种用于计算设备的规范，它是一个虚构出来的计算机，是通过在实际的计算机上仿真模拟各种计算机功能来实现的。
  - JVM就是一个虚拟的用于执行bytecodes字节码的计算机

  - Java虚拟机是Java最核心技术，也是跨平台的基础。

  - Java语言使用Java虚拟机屏蔽了与具体平台相关的信息，使得Java语言编译程序只需生成在Java虚拟机上运行的目标代码（字节码），就可以在多种平台上不加修改地运行。
  - Java虚拟机在执行字节码时，把字节码解释成具体平台上的机器指令执行。这就是Java的能够“一次编译，到处运行”的原因

Java跨平台原理

- 总结5：JDK、JRE、JVM的区别联系
  - JDK:
    - Java Development Kit
    - 针对Java开发员的产品
  - JRE:
    - Java Runtime Environment
    - 是运行Java程序所必须的环境集合
  - JVM
    - Java Virtual Machine
    - 解释运行Java字节码文件，跨平台的核心
  - 联系：JDK 包含JRE，JRE包含JVM。

开始Java开发之前的准备

- 安装JDK
- 安装Java开发工具
- 准备JDK API

开始Java开发之前的准备

- 安装JDK
  - 卸载JDK
  - 安装JDK
  - 验证JDK安装正确
    - Java -version

```
C:\Documents and Settings\Administrator>java -version
java version "1.6.0_14"
Java(TM) SE Runtime Environment (build 1.6.0_14-b08)
Java HotSpot(TM) Client VM (build 14.0-b16, mixed mode, sharing)
```

开始Java开发之前的准备

- 常用Java开发工具
  - 文本编辑器(选择任意一个)
    - UltraEdit
    - EditPlus
    - notepad++
  - 集成开发环境（IDE：Integrated Development Environment）
    - JBuilder （http://www.borland.com）（基本淘汰）
    - Eclipse （http://www.eclipse.org）****开源 解压即可（重点）
    - MyEclipse：若进行J2EE开发，还要加MyEclipse插件。MyEclipse是Eclipse的插件，也是一款功能强大的J2EE集成开发环境。现在直接安装集成Eclipse的MyEclipse即可。商业软件
    - NetBeans（http://java.sun.com）

开始Java开发之前的准备

- JDK帮助文档
  - SUN公司为JDK工具包提供了一整套文档资料，我们习惯上称之为JDK文档。
  - JDK文档中提供了Java中的各种技术的详细资料，以及JDK中提供的各种类的帮助说明。
  - JDK文档是Java语言的完整说明，大多数书籍中的类的介绍都要参照它来完成，它是编程者经常查阅的资料。

第一个Java程序

- 代码编写：（程序员编辑代码并保存在磁盘上）

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

  - 保存为：Welcome.java
- 编译阶段（编译器创建class字节码文件 ）
  - 进入java文件所在目录，执行命令：javac Welcome.java
  - 编译时必须加上扩展名.java。
- 执行阶段：
  - 进入java文件所在目录，执行命令：java Welcome
  - 运行的是类而非class文件，所以类名后不能加扩展名.class

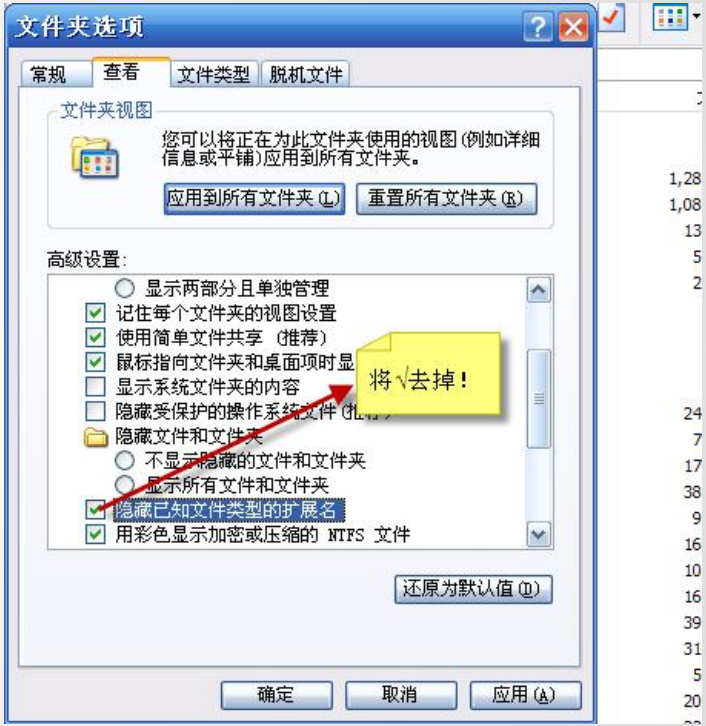
第一个Java程序

- 环境变量配置
  - Java_HOME：C:\Program Files\Java\jdk1.6.0_14
  - path（执行dos命令，系统会在path指定的路径中寻找该命令对应的可执行文件）
  - 将 “%Java_HOME%\bin” 增加到path中；多个目录用分号隔开。
  - classpath
  - JDK5.0以上版本，可以不对其进行配置！

- 配置完java环境变量后，需要重启DOS窗口。重启后新配置的环境变量才能生效。

第一个Java程序

- 第一个程序可能出现的错误
  - java 不是内部或外部命令，也不是可运行的程序或批处理文件。
  - 设置path和重启DOS窗口
  - 编译javac Test.java，明明看到该文件，为何找不到？
  - Java区分大小写。敲代码时注意



第一个Java程序

- Java对大小写敏感，如果出现了大小写拼写错误，程序无法运行
- 关键字class 表明Java 程序中的全部内容都包含在类中，Java是一种面向对象的语言。
- main方法是Java应用程序的入口方法，它有固定的书写格式：

public static void

- 用花括号的代码都必须以“{”开始，以“}”结束， 由于编译器忽略空格，所以花括号风格不受限制。

关键字

类名与文件名完全一样

main()方法四要素必不可少

main()方法是Java程序执行的入口点

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

从控制台输出信息

{ 和 }一一对应，缺一不可

第一个Java程序

- 编程风格
  - 注意缩进!
    - 一定要有缩进。缩进就像人得体的衣着一样!
  - 成对编程!
    - 括号、引号都应该写完后，再往里加内容。
  - 见名知意!
    - 最基本的要求!

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

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

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

第一个Java程序

- 注释
  - 作用
    - 注释就是程序员为读者作的说明，是提高程序可读性的一种手段
  - 类型
    - // 单行注释　注释内容从//到本行结尾
    - /*　*/ 多行注释 /* */ 注释不能嵌套
    - /**　*/文档注释　可以通过JDK提供的Javadoc命令，生成程序的API文档
    - (面向对象编程时再讲)
  - 注意
    - 注释不会出现在字节码文件中。
    - 即Java编译器编译时会
    - 跳过注释语句。

```
public class Welcome {
    //我是单行注释
    public static void main(String[] args/*我是行内注释 */) {
        System.out.println("Hello World!");
    }
    /*
     我是多行注释！
     我是多行注释！
    */
}
```

第二个Java程序

- 环境变量配置
  - Java_HOME：C:\Program Files\Java\jdk1.6.0_14
  - path（执行dos命令，系统会在path指定的路径中寻找该命令对应的可执行文件）
  - 将 “%Java_HOME%\bin” 增加到path中；多个目录用分号隔开。
  - classpath
  - JDK5.0以上版本，可以不对其进行配置！

- 配置完java环境变量后，需要重启DOS窗口。重启后新配置的环境变量才能生效。

反编译

- 编译
  - 源代码----->class
- 反编译
  - class---->源代码
- 反编译软件
  - jd-gui.exe
- 因为编译的时候不会对注释进行处理，所以反编译时不可能得到注释

总结

- Java语言历史
  - Oak--->Java　詹姆斯-高斯林 Sun--->Oracle
  - 1995年，推出Java测试版，目前最高版本1.8
- Java跨平台原理
  - Java最大的优势所在
  - 一次编译，到处运行
  - 编译成格式独立的字节码文件，字节码文件需要虚拟机来解释执行
- Java开发运行原理
  - 编辑、编译、解释运行
  - Java开发环境配置
  - path classpath
- Java注释
  - 单行注释、多行注释、文档注释