

1.1 Java 的基本了解

1.1.1 Java 的历史（参考了一些网上的资料）

Java 的历史充满了戏剧性。他的诞生完全是一个巧合。

在 1990 年，Sun 公司想要为智能家电制作一个通用的控制系统，例如微波炉、固话、电视等。起初使用的语言是 C++ 语言，不过这个语言有一些问题，例如缺少垃圾回收机制，复杂的多继承、指针等。并不适合用来做一个控制系统。

后来，Sun 公司决定在 C++ 的基础之上，对 C++ 进行一些修改，已达到设计控制系统的目的。后来因为一些原因没有成功。唯一行得通的办法是设计一个新的语言，起名 Oak。

1994 年，Sun 公司的一个小组用 Oak 语言编写了一个叫做 WebRunner 的网页浏览器，该浏览器受到使用者的高度评价。Oak 语言可算是真正的出名了。无奈，Oak 商标已被注册，只好改名 Java。

随着 Java 语言的成熟，越来越多的人开始访问 Sun 公司的网站；越来越多人开始学习 Java 语言；用 Java 做的程序、游戏越来越多，他成为了一个广为人知的编程语言！

1997 年，JDK 1.1 版本发布。对效率有很大的提升。

次年，JDK 1.2 发布。Sun 将 Java 分成了 J2EE、J2SE、J2ME 三个版本。

2004 年，里程碑式的更新：JDK 1.5 发布。公司将其改名为 Java SE/ME/EE 5.0。

2006 年，JDK 1.6 发布。

2009 年，Oracle 宣布收购 Sun 公司，通过收购获得了 Java 该项资产。虽然 Sun 公司倒下了，但是 Java 的前途猎猎作响，JDK 1.7/1.8 相继发布！

这只是一个开始，Java 的未来必定更加辉煌。

1.1.2 为什么我要选择 Java？

目前来讲，Java 是世界上最流行的语言之一。数年来，其在编程语言排名中名列前茅。2015 年 6 月份，Java 以第一名的成绩击败了其他的编程语言，可见选择 Java 语言是十分明智的。而且现在市场上对 Java 的程序员需求也十分地广。

当然了，我相信打开这个帖子的读者们也并不一定是想要以 Java 作为一个职业。要编写 Bukkit 的一些插件或 mods 需要很深的 Java 基础。所以说要先打好根基。

2015 年 6 月排行榜：

Jun 2015	Jun 2014	Change	Programming Language	Ratings	Change
1	2	▲	Java	17.822%	+1.71%
2	1	▼	C	16.788%	+0.60%
3	4	▲	C++	7.756%	+1.33%
4	5	▲	C#	5.056%	+1.11%
5	3	▼	Objective-C	4.339%	-6.60%
6	8	▲	Python	3.999%	+1.29%
7	10	▲	Visual Basic .NET	3.168%	+1.25%
8	7	▼	PHP	2.868%	+0.02%
9	9		JavaScript	2.295%	+0.30%
10	17	▲	Delphi/Object Pascal	1.869%	+1.04%
11	-	▲	Visual Basic	1.839%	+1.84%
12	12		Perl	1.759%	+0.28%
13	23	▲	R	1.524%	+0.85%
14	-	▲	Swift	1.440%	+1.44%
15	19	▲	MATLAB	1.436%	+0.66%
16	13	▼	Ruby	1.359%	-0.03%
17	26	▲	PL/SQL	1.229%	+0.74%
18	31	▲	COBOL	0.948%	+0.54%
19	34	▲	ABAP	0.849%	+0.49%
20	18	▼	Pascal	0.846%	+0.04%

1.1.3 Java 的特点

- 面向对象

面向对象是指将程序中的所有事物视为对象。这与我们平时生活中的思考方式完全相同。一个事物有两个部分：属性和行为。这两个事物都被封装在类当中。类是 Java 中的基本组成单元。与面向过程思想不同之处在于“注重于对象”还是“注重于过程”。

- 跨平台性

跨平台性是 Java 中十分重要的一个特性。只要装有 JRE 的平台都可以运行 Java 应用程序。实现了“Write once, run anywhere”（一次编写，到处运行）。

- 易学

Java 中没有 C++ 的多继承、指针。增加了自动的垃圾回收，大大减少了程序员的工作量。而且会 C++ 的程序员很快可以掌握 Java。

这里只列举三个最重要的。其他的可以自己去看资料。

本章小结：

- Java 的历史充满戏剧性
- Java 十分受欢迎，一直与 C 争第一
- Java 面向对象、跨平台、易学

1.2 如何学好 Java 语言

1.2.1 要有编程的兴趣

编程的兴趣对于学好一个编程语言有多大的影响我不需要多说。不仅仅是编程语言，学任何东西的前提就是要自己有兴趣。

许多人看着 Java 程序员能够领高额的工资就想要放弃自己的爱好，马上开始学编程，希望一展身手，马上找到工作。这种做法是完全不可取的。在你真正翻开教程之前，最好先问问自己：我是否真的喜欢编程？我能否坚持下去？我做了程序员会快乐吗？

兴趣是最好的老师。

1.2.2 要有良好的英文能力

Java 是用英语编写的。如果英语实力好的话，更可以理解代码的本质，不需要死记硬背。Java 中的关键字的词的意思往往都和它的作用有很大的关系。如果英语好，甚至看到这个词就能大概摸出来这个关键字的意义。例如 if、while、void 等。

不过其实英语不太好也不是说真的学不了了。这是要花费的功夫可能要多一点。

1.2.3 要会电脑的基本操作

这个我真的不用多说，连打字都不会编什么程。。

1.2.4 劳逸结合、充沛的精神

编写很大型的项目是需要投入进很多精力和时间的。所以说如果准备冲刺考试这类情况的话建议先把学习编程的任务放在一边，把自己的学业看的最重要。假期的时候再来学效率肯定更高。

编程时遇到困难千万不要在电脑面前唉声叹气。遇到问题倒不如出去散个步，听会歌。先放松一下自己的心情，然后再回来思考问题。如果不换换脑筋不仅不能解决问题，还会影响心情，产生厌恶感。

1.2.5 关于初学是否用 IDE（有争议，请理智辩论）

简单的来说 IDE 就像是你的小助手一样，可以帮你生成代码，补全拼写等。目前比较流行的 IDE 软件有 Eclipse、NetBeans 等。

这些软件很方便，但是本人认为，初学者不要使用。

虽然都说“工欲善其事，必先利其器”，但是是在你会做这个事情的情况下。就像是还没有会加法，就要用计算器了吗？要记住，你们学的不是怎么简化工作量，不是提高效率，而是 Java 语言。如果刚开始就直接用 IDE，会产生依赖，最本质的事情却没有明白。

建议用一些比较简单的编写软件，例如 notepad++、editplus、ultraedit 等。

本章小结：

- 要有兴趣
- 要有良好的英文能力
- 要会电脑的基本操作
- 要有时间
- 不需要用 IDE

1.3 命令提示符

1.3.1 命令提示符概述

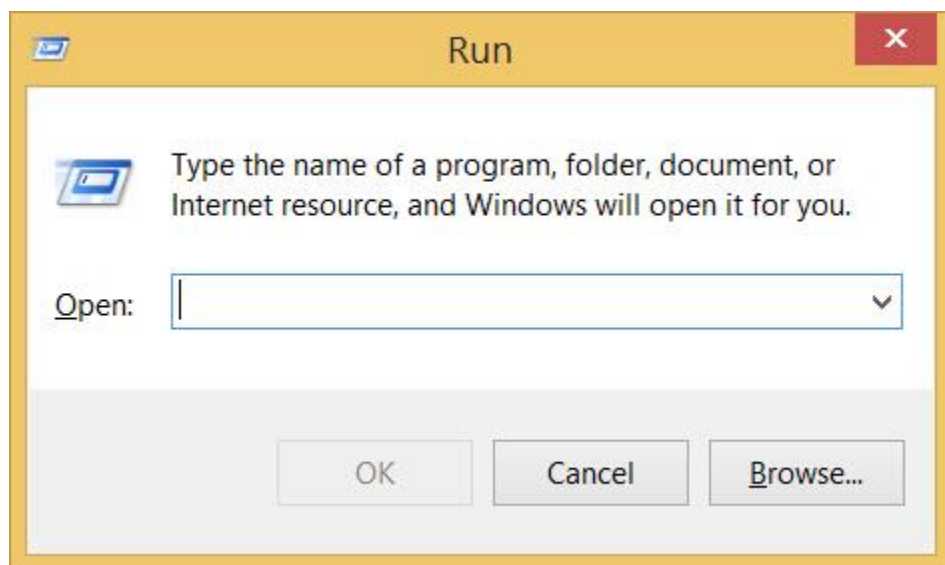
早期的电脑是通过命令提示符来操作磁盘或文件的。这种操作方式是通过繁琐的指令来实现的，与图形化界面相对。图形化界面就是我们现在用的操作方式。这种操作方式不需要记住繁琐的命令，只需要用鼠标点点，一目了然。

1.3.2 为什么要学习命令提示符呢？

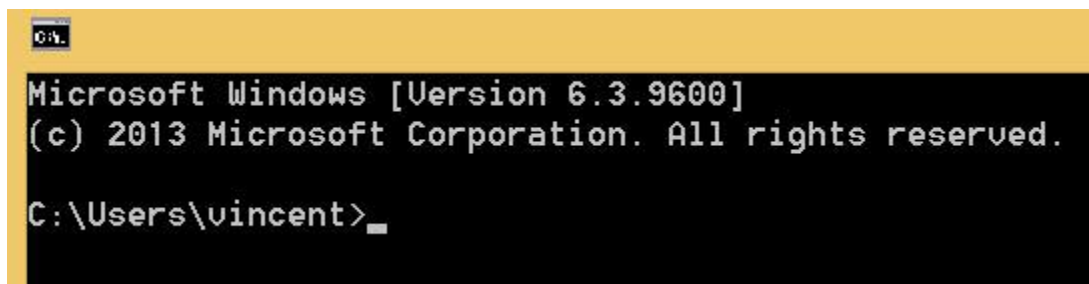
虽说我们现在有了图形化界面，但是 Java 程序的编译和运行我们还是要通过命令提示符来实现。虽然说 IDE 具有自动编译和运行的功能，但是如上一章所讲的，不推荐使用。

1.3.3 如何打开命令提示符（Windows 系统）？

- 1、按住 Windows 键+R。此时会弹出“运行”窗口；



- 2、在“打开”中输入 `cmd`，然后点击 OK。命令提示符被打开了。



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\vincent>
```

1.3.4 常见命令

命令	作用
盘符:	进入该盘根目录
cd <路径>	进入该路径
cd..	进入上级路径
md <名称>	在该目录下建立文件夹
rd <名称>	在该目录下删除指定的文件夹
del <文件名>	删除文件
cls	清屏

示范：盘符：



```
C:\Users\vincent>d:
D:\>g:
G:\>
```

在 C 盘下输入“d:”则切换到了 D 盘。后面同理。

示范：cd <路径> 和 cd..

```
D:\>cd 资料  
D:\资料>cd..  
D:\>_
```

在 D 盘根目录下输入“cd 资料”则进入了“D:\资料”该路径。

在 D:\资料路径下，输入“cd..”则回到了 D:根目录。

示范：**md** <名称> 和 **rd** <文件名>

```
D:\>md demo
```

输入“md demo”则在其路径下创建了一个文件夹：



输入“rd demo”则将其删除。

示范：**del** <文件名>

此时，我在 D 盘根目录下有一个叫做“demo.txt”的文件：



输入“del demo.txt”,该文件被删除。

```
D:\>del demo.txt
```

示范：**cls**


```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\vincent>d:

D:\>md demo

D:\>rd demo

D:\>del demo.txt

D:\>_
```

此时，我的命令提示符中还记录着我之前写过的命令。
输入“cls”之后，其被清屏：



顺带一提，在命令提示符中可以通过↑或↓自动键入上次输入过得命令。

本章小结：

- cmd 的打开方式是：Windows+R，打开“运行”，输入“cmd”，然后回车
- “盘符:”用于切换命令行的路径的盘符
- “cd <路径>”用于切换到这个目录
- “cd..”用于回到上级目录
- “md <文件夹名>”用于在当前目录下创建一个指定名称的文件夹
- “rd <文件夹名>”用于在当前目录下找到这个文件夹，并将其删除
- “del <文件名>”用于在当前目录下找到这个文件，并删除
- “cls”用于清屏当前的命令行

1.4 安装 jdk，配置环境变量

1.4.1 JRE、JDK 和 JVM

JRE 全称 **Java Runtime Environment**，中文 Java 运行环境。如果要运行 Java 应用程序，必须安装 JRE。

JRE 下面包含 JVM。JVM 全称 **Java Virtual Machine**，中文 Java 虚拟机。简单的来说，JVM 是用来阅读你的.class 字节码文件的虚拟的、抽象的机器。

学生提问：那么 JRE 就只包含 JVM 吗？运行 Java 程序还需要别的东西吗？

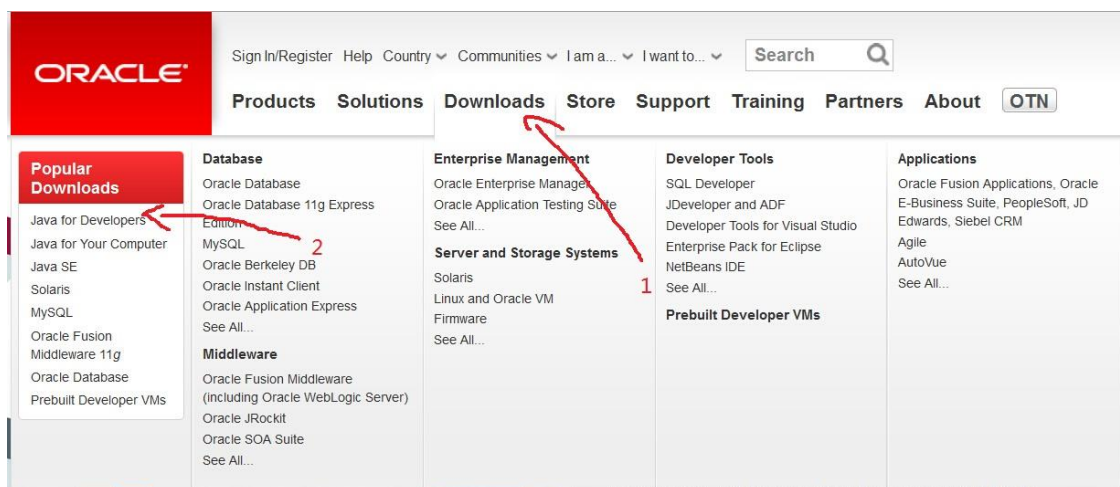
答：JRE 不仅仅只包括 JVM。要运行 Java 程序除了需要虚拟机之外，还需要其他的基础类库、校验器等其他组件。

JDK 全称 **Java Development Kit**，中文 Java 开发工具包。JDK 包含编译器 javac、jar、javadoc 等组件。

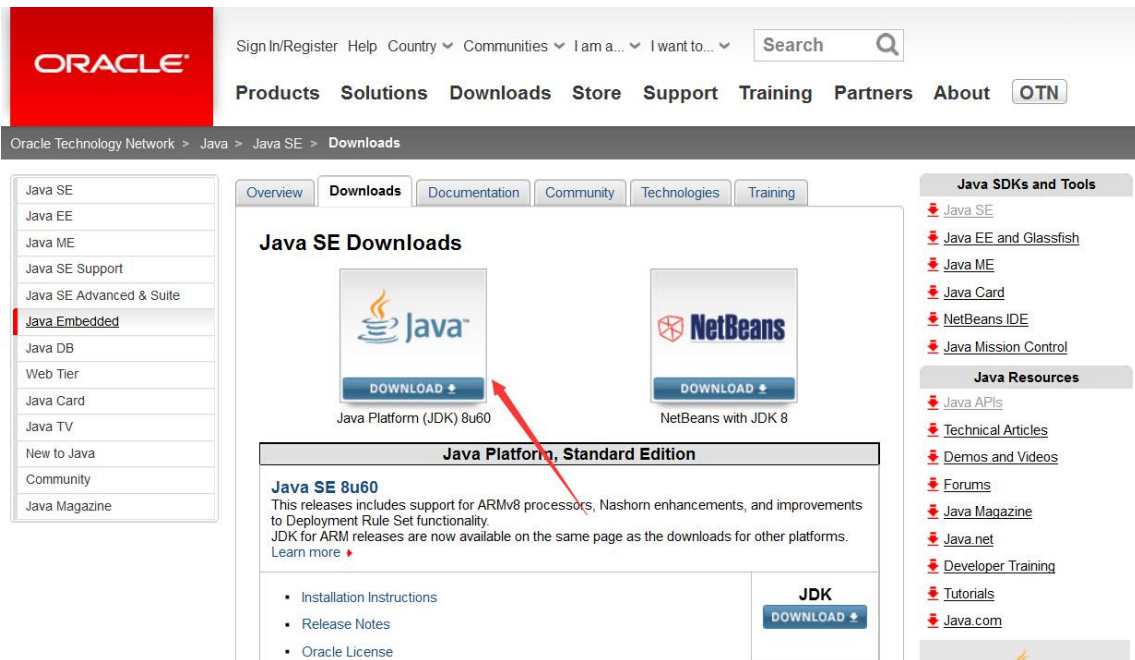
简单来说：光要运行 Java，装 JRE 就好了。如果要开发，两者都得安装。

1.4.2 安装 JDK

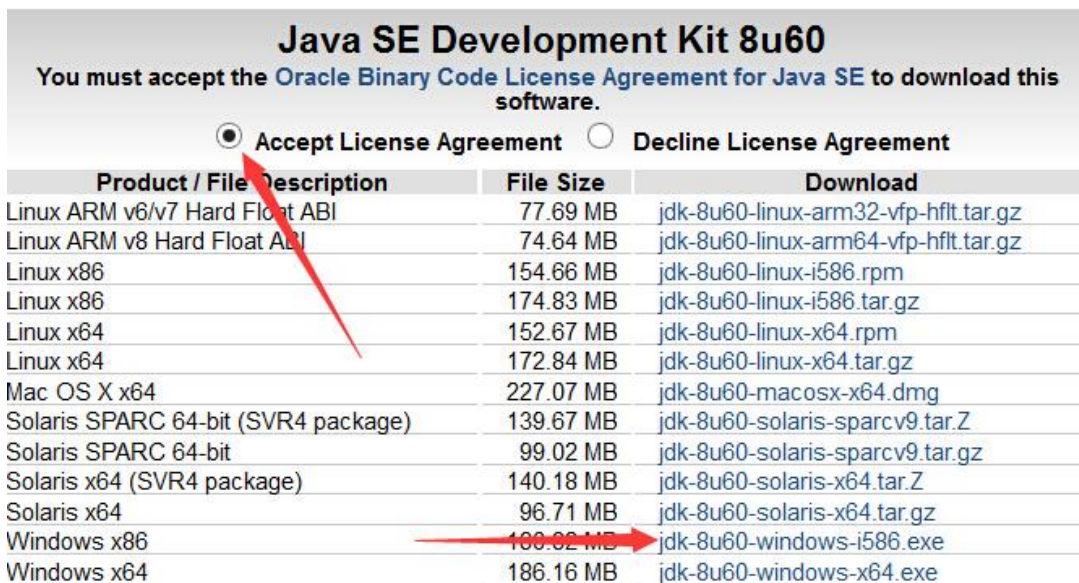
- 1、访问 [Oracle 官网](#)
- 2、让鼠标悬浮在 Downloads 目录下，然后点击左侧的 Java for Developers。



- 3、点击 Java Platform (JDK) 8u60



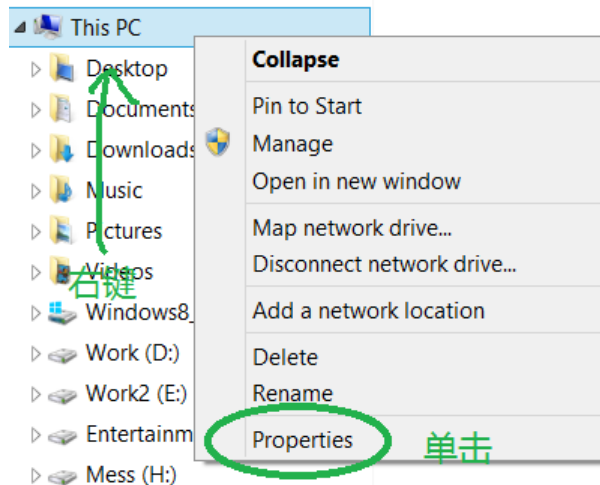
- 4、稍微向下拉，点击 Accept License Agreement，然后下载对应你的平台的 JDK。



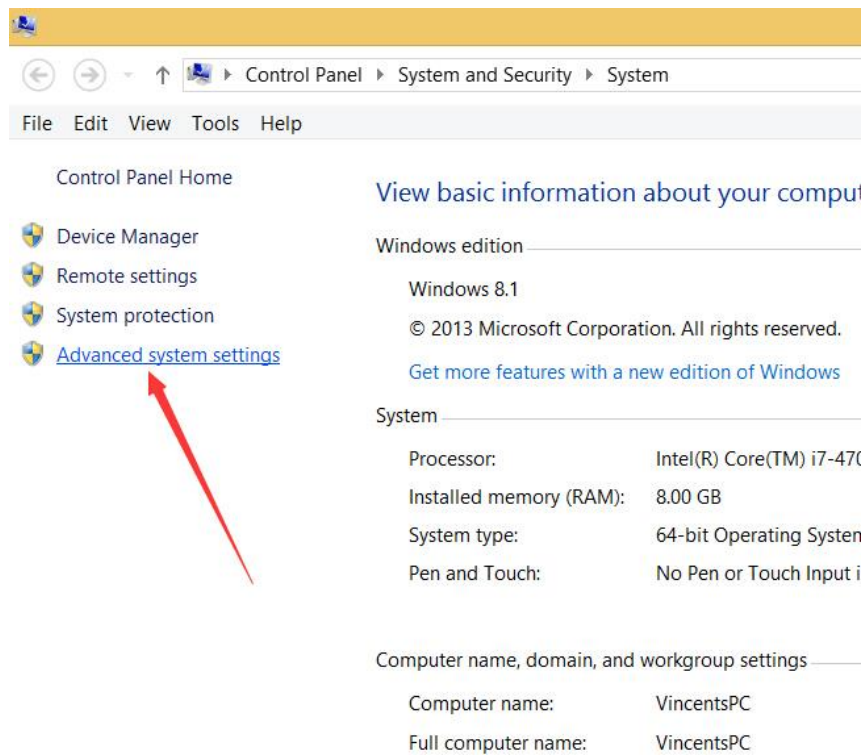
- 5、开始下载
- 6、根据提示，安装 JDK。

1.4.3 配置环境变量（Windows 8 系统）

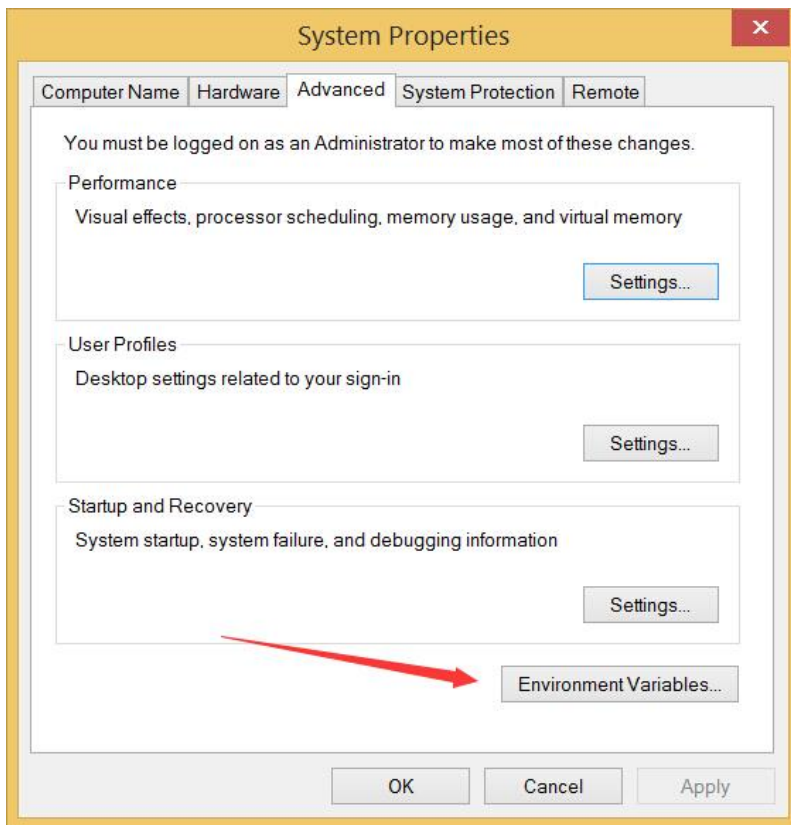
- 1、在“我的电脑”下右键“计算机”，点击“属性”菜单



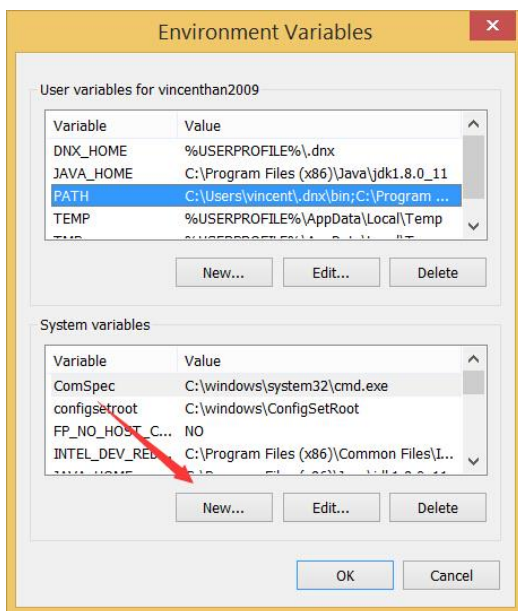
- 2、在弹出的窗口的左侧，有一个“高级系统设置”，单击它



- 3、点击后会弹出“系统属性”窗口。单击“环境变量”



- 4、弹出“环境变量”窗口之后，点击新建。

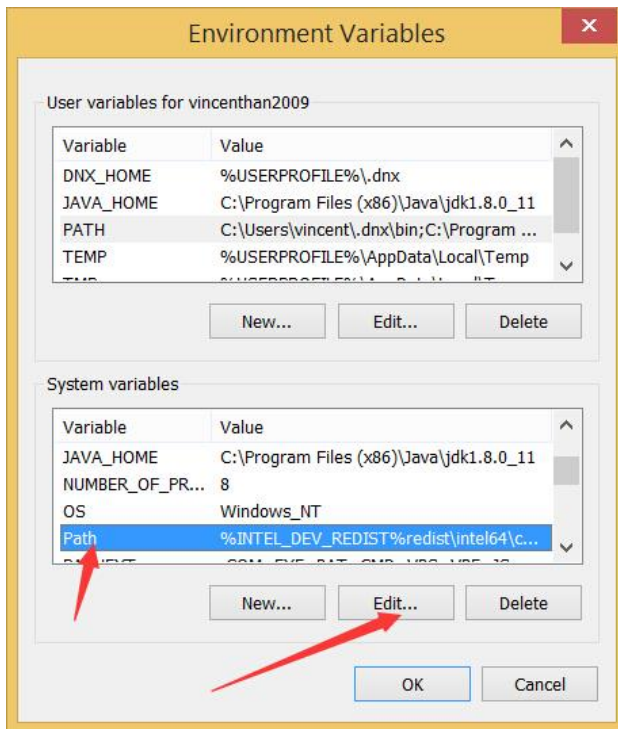


- 5、输入变量名和变量值，然后单击 OK。

变量名：JAVA_HOME

变量值：C:\Program Files (x86)\Java\jdk1.8.0_11（你的 jdk 安装路径，因人而异）

- 6、找到已存在的 Path 路径，单击后点击“编辑”。注意：要选择在系统变量中的 Path，而非上边用户变量的 PATH。



- 7、在整个变量值最前端加入一下代码，然后单击 OK。

.;%JAVA_HOME%\bin;

- 8、打开命令提示符，输入 javac。如果出现了以下提示，恭喜您配置成功。


```

C:\Users\vincent>javac
Usage: javac <options> <source files>
where possible options include:
  -g               Generate all debugging info
  -g:none          Generate no debugging info
  -g:{lines,vars,source} Generate only some debugging info
  -nowarn          Generate no warnings
  -verbose         Output messages about what the compiler is doing
  -deprecation     Output source locations where deprecated APIs are used
  -classpath <path> Specify where to find user class files and annotation processors
  -cp <path>       Specify where to find user class files and annotation processors
  -sourcepath <path> Specify where to find input source files
  -bootclasspath <path> Override location of bootstrap class files
  -extdirs <dirs>   Override location of installed extensions
  -endorseddirs <dirs> Override location of endorsed standards path
  -proc:{none,only} Control whether annotation processing and/or compilation is done.
  -processor <class1>[,<class2>,<class3>...] Names of the annotation processors to run; bypasses default discovery process
  -processorpath <path> Specify where to find annotation processors
  -parameters     Generate metadata for reflection on method parameters
  -d <directory>   Specify where to place generated class files
  -s <directory>   Specify where to place generated source files
  -h <directory>   Specify where to place generated native header files
  -implicit:{none,class} Specify whether or not to generate class files for implicitly referenced files
  -encoding <encoding> Specify character encoding used by source files
  -source <release> Provide source compatibility with specified release
  -target <release> Generate class files for specific VM version
  -profile <profile> Check that API used is available in the specified profile
  -version         Version information
  -help            Print a synopsis of standard options
  -Akey[:value]    Options to pass to annotation processors
  -X              Print a synopsis of nonstandard options
  -J<flag>         Pass <flag> directly to the runtime system
  -Werror          Terminate compilation if warnings occur
  @<filename>      Read options and filenames from file

```

本章小结:

- JRE 用于运行、JDK 用于开发
- 安装 JDK
- 在编程之前要配置环境变量
- 通过输入 javac 命令检测是否配置成功

1.5 Hello, world 程序

经过了前四章的讲（废）解（话），我们终于可以写一点代码了，是不是很激动呢？

1.5.1 Hello, world 意义

Hello, world 是世界上第一个程序。这个程序表示真正踏入了某一个编程语言的征途中。

该程序的要求：在命令提示符上打印“hello, world”。

```
C:\Users\vincent\Desktop>java HelloWorld  
hello, world
```

1.5.2 开始编写！

这一章建议使用记事本来编写。

打开记事本，开始写代码吧！

首先我们要干的第一件事就是让计算机知道，这是一个 Java 的程序。Java 中，“类”是一个基本组成单元。我们要干的第一件事是要声明这是一个类。

类需要有一个名字，也需要有一个区间。

请写下：

```
class HelloWorld{  
  
}
```

class 是一个关键字。用于声明类。后面的 HelloWorld 不是关键字，是一个标识符，相当于类的名字，可以自己写。大括号表示类的区间。

接着，我们要干另一件事情，叫做声明主方法。

在一个项目中，里面有成千上万个类。但是其实有特殊一个类，他是程序的入口点，这种类被称为主类。在这个例子当中，我们刚才定义的类肯定是主类，毕竟我们要直接运行它。


```
class HelloWorld{
    public static void main(String[] args) {

    }
}
```

我刚开始学的时候也蒙了。。这么长，究竟什么意思？这个我们只能先背下来，以后我们会一个一个解释。

定义了主方法之后，现在的类成为了主类，它具备独立运行的资格了。现在我们要加上一个语句，用于打印字符串。请写下：

```
class HelloWorld{
    public static void main(String[] args) {
        System.out.println("hello, world");
    }
}
```

`println` 是一个方法，是指打印一段字符串之后换行。这段代码大家也都要记住。

注意：后面的分号不能忘掉。这是一个语句，语句通过分号来结束。

代码就是这么多，写下来我们来讲一下如何运行。

学生提问：print()方法和println()方法有什么区别？

回答：这两个方法的用法是完全相同的。但是区别在于print()方法在打印完之后不会换行，而println()方法会自动换行。具体使用哪种取决于需求。

1.5.3 设置 classpath

在编译和运行之前，需要做一件事情。那就是告诉 Java 虚拟机到哪里去找程序的字节码文件。其是通过 `classpath` 环境变量来找到的。如果设置不当，虚拟机就会找不到字节码文件的位置，自然运行不了。`classpath` 应当在命令行中修改。语法如下：

```
set classpath=路径 1;路径 2;路径 3.....路径 n
```

虚拟机会先在路径 1 找字节码文件，如果没有找到再去路径 2，以此类推。路径可以输入绝对路径，也可以用点“.”代表当前目录。所以说我们先暂时这样设置：

```
set classpath=.
```

这样就可以了。

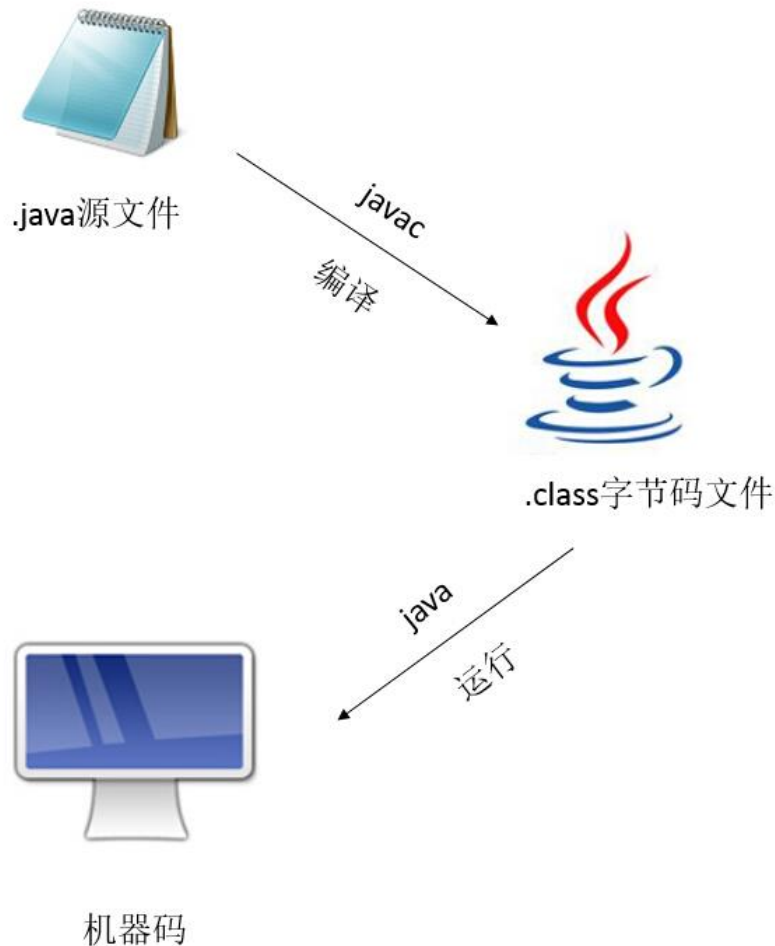
1.5.4 编译与运行

首先我们把刚才写的记事本文件的后缀名改成.java，让计算机知道这是一个 Java 源文件。

但是计算机不能够直接认出来我们所写的代码，我们要把这个代码“翻译”成电脑读得懂的语言。

“翻译”这个过程叫做“编译”。编译之后会生成一个.class 的文件，这个文件是计算机读得懂得，叫做字节码文件。

编译之后就可以运行了。那么具体怎么做呢？先上渣图：

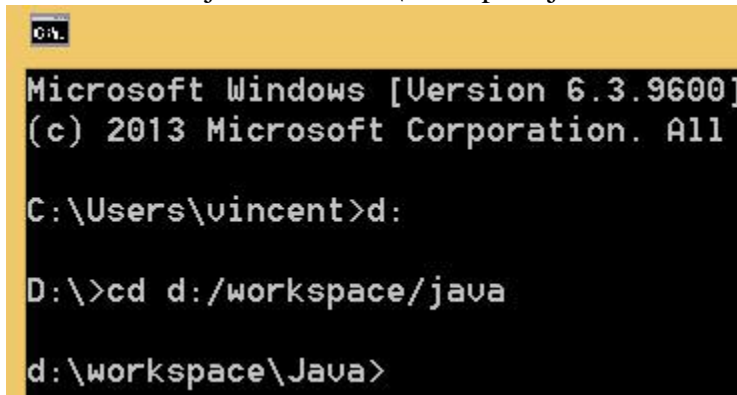


编译和运行都是在命令提示符中通过命令的形式来实现的。编译用到的命令为 `javac <源文件名>`，运行时 `java <字节码文件名>`。

学生提问：我把源代码改了，再一次运行，为什么结果没有变？

回答：源文件更改之后，需要重新编译，不然看不到新的运行结果。比如说我有一本书已经翻译完了，我对书的内容进行了调整，如果不再次翻译，读译文的读者会读到不同吗？所以说如果代码改变了，一定要重新编译。

打开命令提示符，通过切换盘符和 cd 命令将路径改到.java 文件所在的目录，在我的电脑里我的.java 路径在 d:\workspace\java 下。



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All
C:\Users\vincent>d:

D:\>cd d:/workspace/java

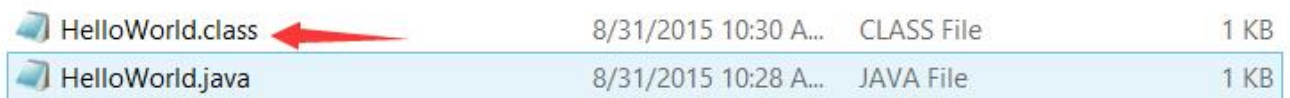
d:\workspace\Java>
```

路径对好了之后，就可以用到 javac 命令了。

javac 后面要跟上.java 的文件名，需要声明后缀名。
请输入：

```
javac HelloWorld.java
```

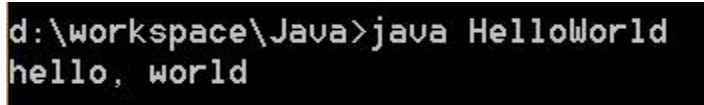
输入之后回车。请看.java 所在的目录，多了一个叫做.class 的文件！



HelloWorld.class	8/31/2015 10:30 A...	CLASS File	1 KB
HelloWorld.java	8/31/2015 10:28 A...	JAVA File	1 KB

编译过程完成了，现在运行。
请输入（不需要.class）：

```
java HelloWorld
```



```
d:\workspace\Java>java HelloWorld
hello, world
```

回车之后，"hello, world"打印在了命令提示符上了！恭喜你，你的第一个 Java 程序

完成了！

本章小结：

- `hello, world` 代表着正式开始学习一个计算机语言
- 学习了定义类和方法
- 学习了打印输出语句
- 学会了编译和运行

1.6 注释

1.6.1 注释概述

在编写规模较大的程序时，有可能要对代码进行一些解释，以增强阅读性。如果是由一个团队一起开发的话留下注释也是很好的习惯，别人更容易理解你的代码。

注释部分被虚拟机忽略，不读。

在 Java 中，有三种注释：

- 单行注释
- 多行注释
- 文档注释

文档注释暂时先不讲。

1.6.2 单行注释

在 Java 中，单行注释通过“//”标记。引用上一章 Hello world 的例子：

```
class HelloWorld{                                //定义类
    public static void main(String[] args) {      //主方法
        System.out.println("hello, world");      //打印
    }
}
```

在这个程序当中，一共有三个注释。从//开始所有字符直到行的结束都会变成注释，被虚拟机忽略。

1.6.3 多行注释

顾名思义，多行注释可以使其区间内的字符都变为注释。标记方法为/*开始，*/结束。例如：

```
/*
这是一个多行注释
该程序用于输出hello world
*/
class HelloWorld{                                //定义类
    public static void main(String[] args){      //定义主方法
        System.out.println("Hello, world");      //该语句打印了Hello, world
    }
}
```

1.6.4 多行注释的嵌套

一定要注意：多行注释中可以嵌套单行注释，但是不能嵌套多行注释。

```
/*
    /*
    */
*/
```

该代码为非法。原因是一个注释开始标记被第一个注释关闭标记关闭，第二个注释开始标记因为在注释中，被忽略了，所以最后一个注释关闭标记没有其所对应的开始标记。

本章小结：

- 注释可以提高阅读性
- Java 有单行注释、多行注释、文档注释三种注释。