

# 第1章

# Java语言概述



讲师：宋红康  
新浪微博：尚硅谷-宋红康



**Java基础是学习JavaEE、大数据、Android开发的基石！**



## 举例：Spring – Rest(Spring MVC)



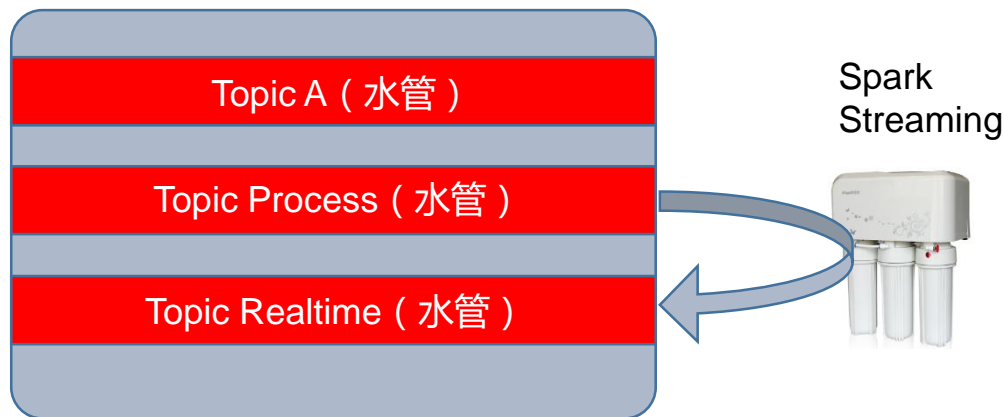
### 核心代码：

```
public Model getPieData(Model model) {  
    List<Count> counts = countRepository.findAll();  
    JSONArray result = new JSONArray();  
    for (Count count:counts) {  
        JSONObject o = new JSONObject();  
        o.put("name", count.getProvince());  
        o.put("value", count.getSum());  
        result.add(o);  
    }  
    model.addAttribute("data", result);  
    return model;  
}
```

```
client.onMessageArrived = function(message) {  
    var data = eval('(' + message.payloadString + ')');  
    if(data.type == "map") {  
        var newData = data.data;  
        for (var i = 0; i < newData.length; i++) {  
            mapData.push(newData[i]);  
        }  
        map.setOption(option = {  
            series: [{  
                data: mapData  
            }]  
        });  
    } else if(data.type == "pie") {  
        //reload  
        $.get('rest/pieData', function (data) {  
            pieData = data.data;  
            pie.setOption(option = {  
                series: [{  
                    data: pieData  
                }]  
            });  
        });  
    }  
};
```



## 举例：Spark – Spark Streaming



核心代码：

```
public void call(Iterator<ConsumerRecord<String, String>> consumerRecords) {  
    KafkaProducer<String, String> kafkaproducer = new KafkaProducer<>(props);  
    while(consumerRecords.hasNext()) {  
        ConsumerRecord<String, String> record = consumerRecords.next();  
        String[] params = record.value().split(regex: ";");  
        String geo = "["+params[params.length-2] +","+params[params.length-1]+"]";  
        kafkaproducer.send(new ProducerRecord<String, String>(topicBroadcast.getValue(), geo));  
        System.out.println(geo);  
    }  
    kafkaproducer.close();  
}
```



```
@Bind(R.id.scrollView),
GradationScrollView scrollView;
@Bind(R.id.tv_usercenter)
TextView tvUsercenter;
private Context mContext;
private int height;
```

```

@Nullable
@Override
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup
    mContext = getActivity();

    View view = View.inflate(mContext, R.layout.fragment_person, null);
    ButterKnife.bind(this, view);

    return view;
}

vto.addOnGlobalLayoutListener(() -> {
    rlHeader.getViewTreeObserver().removeGlobalOnLayoutListener(this);

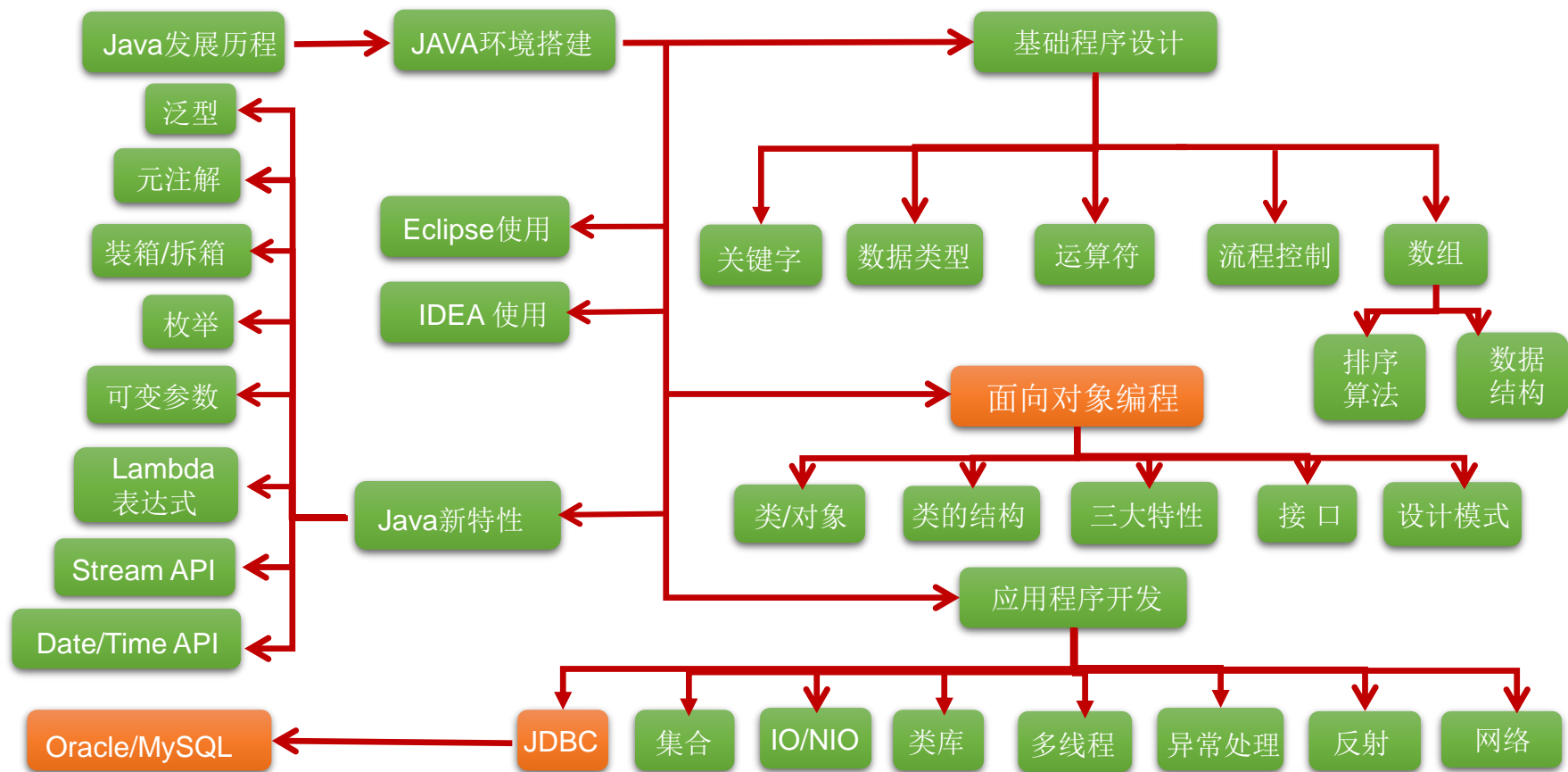
    height = rlHeader.getHeight();

    scrollView.setScrollListener((scrollView, x, y, oldx, oldy) -> {

        if (y <= 0) {
            tvUsercenter.setBackgroundColor(Color.argb(0, 255, 0, 0));
        } else if (y > 0 && y <= height) {
            float scale = (float) y / height;
            float alpha = 255 * scale;

            tvUsercenter.setTextColor(Color.argb((int) alpha, 255, 255, 255));
            tvUsercenter.setBackgroundColor(Color.argb((int) alpha, 255, 0, 0));
        } else {
            tvUsercenter.setBackgroundColor(Color.argb(255, 255, 0, 0));
        }

    });
});
});
}
```





# Java基础课程概述

第一部分：编程语言核心结构

主要知识点：变量、基本语法、分支、循环、数组、...

第二部分：Java面向对象的核心逻辑

主要知识点：OOP、封装、继承、多态、接口、...

第三部分：开发Java SE高级应用程序

主要知识点：异常、集合、I/O、多线程、反射机制、网络编程、.....

第四部分：实训项目

项目一：家庭收支记账软件

项目二：客户信息管理软件

项目三：开发团队人员调度软件

附加项目一：银行业务管理软件

附件项目二：单机考试管理软件



- 第1章 Java语言概述
- 第2章 基本语法
- 第3章 数组
- 第4章 面向对象编程(上)
- 第5章 面向对象编程(中)
- 第6章 面向对象编程(下)
- 第7章 异常处理
- 第8章 枚举类&注解
- 第9章 Java集合
- 第10章 泛型
- 第11章 IO流
- 第12章 多线程
- 第13章 Java常用类
- 第14章 Java反射机制
- 第15章 网络编程
- 第16章 Lambda表达式与Stream API
- 第17章 Java 9 & 10 & 11新特性



# 目录



1

软件开发介绍

2

计算机编程语言介绍

3

Java语言概述

4

运行机制及运行过程

5

Java的环境搭建

6

开发体验—HelloWorld

# 目录



5

常见问题及解决方法

6

注释(Comment)

7

Java API文档

8

良好的编程风格

9

常用的Java开发工具



# 1-1 软件开发介绍



# 1.1 软件开发介绍

- 软件开发

软件，即一系列按照特定顺序组织的计算机数据和指令的集合。有系统软件和应用软件之分。

- 人机交互方式

- 图形化界面(Graphical User Interface GUI)这种方式简单直观，使用者易于接受，容易上手操作。
- 命令行方式(Command Line Interface CLI): 需要有一个控制台，输入特定的指令，让计算机完成一些操作。较为麻烦，需要记录住一些命令。

Pascal之父Nicklaus Wirth: “Algorithms+Data Structures=Programs”



### ●常用的DOS命令

- **dir** : 列出当前目录下的文件以及文件夹
- **md** : 创建目录
- **rd** : 删除目录
- **cd** : 进入指定目录
- **cd..** : 退回到上一级目录
- **cd\** : 退回到根目录
- **del** : 删除文件
- **exit** : 退出 dos 命令行
  - ✓ 补充: `echo javase>1.doc`

### ● 常用快捷键

- ← →: 移动光标
- ↑ ↓: 调阅历史操作命令
- Delete和Backspace: 删除字符



## 1-2 计算机编程语言介绍



### ●什么是计算机语言

➤ 语言：是人与人之间用于沟通的一种方式。例如：中国人与中国人用普通话沟通。而中国人要和英国人交流，就要学习英语。

➤ **计算机语言：人与计算机交流的方式。**

如果人要与计算机交流，那么就要学习计算机语言。

计算机语言有很多种。如：**C ,C++ ,Java ,PHP , Kotlin, Python, Scala**等。

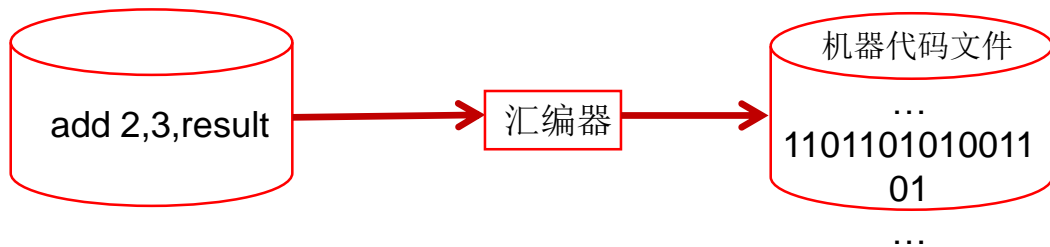


- 第一代语言

- 机器语言。指令以二进制代码形式存在。

- 第二代语言

- 汇编语言。使用助记符表示一条机器指令。







- 第三代语言：高级语言
  - C、Pascal、Fortran面向过程的语言
  - C++面向过程/面向对象
  - **Java**跨平台的纯面向对象的语言
  - .NET跨语言的平台
  - Python、Scala...





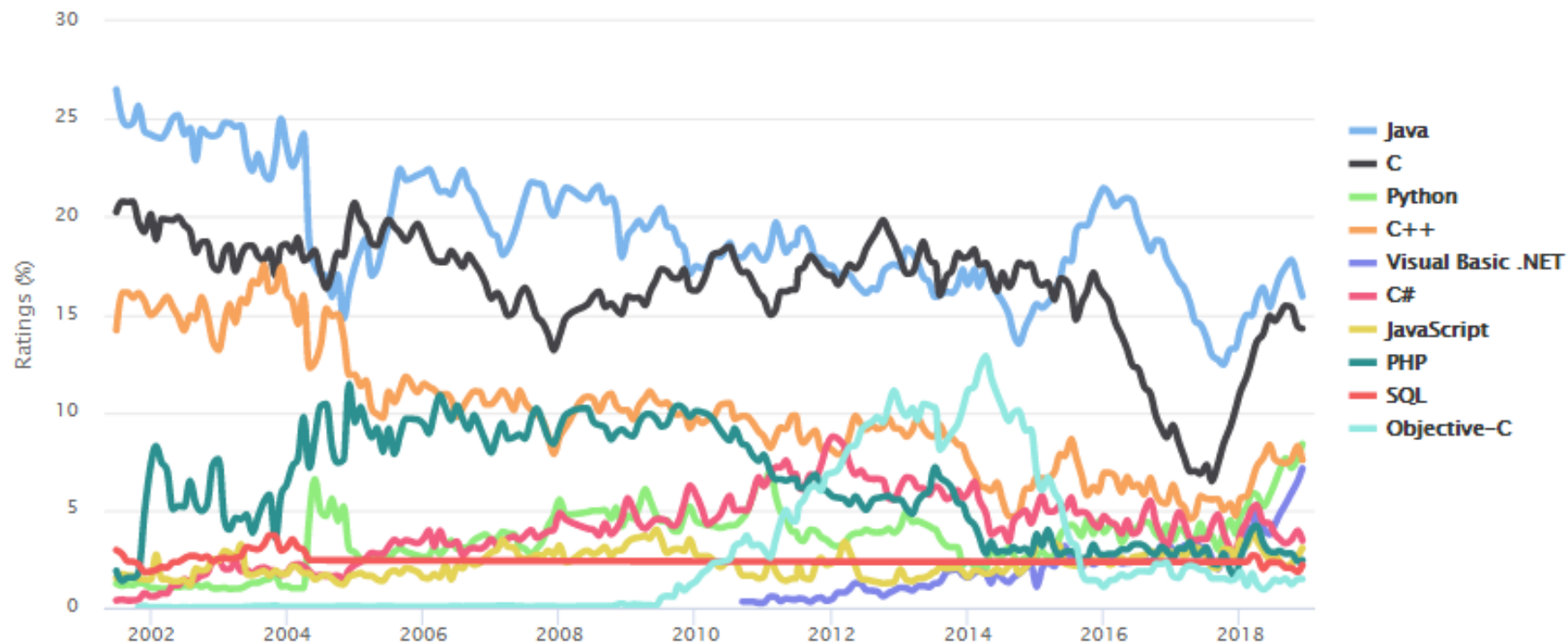
# 1. 从java语言的市场需求来看

Dec 2018	Dec 2017	Change	Programming Language	Ratings	Change
1	1		Java	15.932%	+2.66%
2	2		C	14.282%	+4.12%
3	4	▲	Python	8.376%	+4.60%
4	3	▼	C++	7.562%	+2.84%
5	7	▲	Visual Basic .NET	7.127%	+4.66%
6	5	▼	C#	3.455%	+0.63%
7	6	▼	JavaScript	3.063%	+0.59%
8	9	▲	PHP	2.442%	+0.85%
9	-	▲	SQL	2.184%	+2.18%
10	12	▲	Objective-C	1.477%	-0.02%
11	16	▲	Delphi/Object Pascal	1.396%	+0.00%
12	13	▲	Assembly language	1.371%	-0.10%
13	10	▼	MATLAB	1.283%	-0.29%



## TIOBE Programming Community Index

Source: [www.tiobe.com](http://www.tiobe.com)



数据来源: TIOBE

让天下没有难学的技术



# 1-3 Java语言概述



- 是SUN(**S**tanford **U**niversity **N**etwork, 斯坦福大学网络公司) 1995年推出的一门高级编程语言。
- 是一种面向Internet的编程语言。Java一开始富有吸引力是因为Java程序可以在Web浏览器中运行。这些Java程序被称为Java小程序（applet）。applet使用现代的图形用户界面与Web用户进行交互。 applet内嵌在HTML代码中。
- 随着Java技术在web方面的不断成熟，已经成为Web应用程序的首选开发语言。

后台开发：Java、PHP、Python、Go、Node.js



- 1991年 Green项目，开发语言最初命名为Oak (橡树)
- 1994年，开发组意识到Oak 非常适合于互联网
- 1996年，发布JDK 1.0，约8.3万个网页应用Java技术来制作
- 1997年，发布JDK 1.1，JavaOne会议召开，创当时全球同类会议规模之最
- 1998年，发布JDK 1.2，同年发布企业平台J2EE
- 1999年，Java分成J2SE、J2EE和J2ME，JSP/Servlet技术诞生
- 2004年，发布里程碑式版本：JDK 1.5，为突出此版本的重要性，更名为JDK 5.0
- 2005年，J2SE -> JavaSE，J2EE -> JavaEE，J2ME -> JavaME
- 2009年，Oracle公司收购SUN，交易价格74亿美元
- 2011年，发布JDK 7.0
- 2014年，发布JDK 8.0，是继JDK 5.0以来变化最大的版本
- 2017年，发布JDK 9.0，最大限度实现模块化
- 2018年3月，发布JDK 10.0，版本号也称为18.3
- 2018年9月，发布JDK 11.0，版本号也称为18.9



## 1.3 Java语言概述



让天下没有难学的技术





### ◆Java技术体系平台

#### Java SE(Java Standard Edition)标准版

支持面向桌面级应用（如Windows下的应用程序）的Java平台，提供了完整的Java核心API，此版本以前称为J2SE

#### Java EE(Java Enterprise Edition)企业版

是为开发企业环境下的应用程序提供的一套解决方案。该技术体系中包含的技术如：Servlet、Jsp等，主要针对于Web应用程序开发。版本以前称为J2EE

#### Java ME(Java Micro Edition)小型版

支持Java程序运行在移动终端（手机、PDA）上的平台，对Java API有所精简，并加入了针对移动终端的支持，此版本以前称为J2ME

#### Java Card

支持一些Java小程序（Applets）运行在小内存设备（如智能卡）上的平台



- 从Java的应用领域来分，Java语言的应用方向主要表现在以下几个方面：
  - **企业级应用**：主要指复杂的大企业的软件系统、各种类型的网站。Java的安全机制以及它的跨平台的优势，使它在分布式系统领域开发中有广泛应用。应用领域包括金融、电信、交通、电子商务等。
  - **Android平台应用**：Android应用程序使用Java语言编写。Android开发水平的高低很大程度上取决于Java语言核心能力是否扎实。
  - **大数据平台开发**：各类框架有Hadoop, spark, storm, flink等，就这类技术生态圈来讲，还有各种中间件如flume, kafka, sqoop等等，这些框架以及工具大多数是用Java编写而成，但提供诸如Java, scala, Python, R等各种语言API供编程。
  - **移动领域应用**：主要表现在消费和嵌入式领域，是指在各种小型设备上的应用，包括手机、PDA、机顶盒、汽车通信设备等。



## 1.3 Java语言概述：Java语言的诞生

java之父James Gosling团队在开发“Green”项目时，发现C缺少垃圾回收系统，还有可移植的安全性、分布程序设计和多线程功能。最后，他们想要一种易于移植到各种设备上的平台。

Java确实是从C语言和C++语言继承了许多成份，甚至可以将Java看成是**类C语言**发展和衍生的产物。比如Java语言的变量声明，操作符形式，参数传递，流程控制等方面和C语言、C++语言完全相同。但同时，Java是一个**纯粹的面向对象**的程序设计语言，它继承了C++语言面向对象技术的核心。Java**舍弃了C语言中容易引起错误的指针**（以引用取代）、运算符重载（operator overloading）、多重继承（以接口取代）等特性，**增加了垃圾回收器功能**用于回收不再被引用的对象所占据的内存空间。JDK1.5又引入了泛型编程（Generic Programming）、类型安全的枚举、不定长参数和自动装/拆箱





- **Java语言是易学的。**Java语言的语法与C语言和C++语言很接近，使得大多数程序员很容易学习和使用Java。
- **Java语言是强制面向对象的。**Java语言提供类、接口和继承等原语，为了简单起见，只支持类之间的单继承，但支持接口之间的多继承，并支持类与接口之间的实现机制（关键字为implements）。
- **Java语言是分布式的。**Java语言支持Internet应用的开发，在基本的Java应用编程接口中有一个网络应用编程接口（java net），它提供了用于网络应用编程的类库，包括URL、URLConnection、Socket、ServerSocket等。Java的RMI（远程方法激活）机制也是开发分布式应用的重要手段。
- **Java语言是健壮的。**Java的强类型机制、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语言是安全的。**Java通常被用在网络环境中，为此，Java提供了一个安全机制以防恶意代码的攻击。如：安全防范机制（类ClassLoader），如分配不同的名字空间以防替代本地的同名类、字节代码检查。
- **Java语言是体系结构中立的。**Java程序（后缀为java的文件）在Java平台上被编译为体系结构中立的字节码格式（后缀为class的文件），然后可以在实现这个Java平台的任何系统中运行。
- **Java语言是解释型的。**如前所述，Java程序在Java平台上被编译为字节码格式，然后可以在实现这个Java平台的任何系统的解释器中运行。
- **Java是性能略高的。**与那些解释型的高级脚本语言相比，Java的性能还是较优的。
- **Java语言是原生支持多线程的。**在Java语言中，线程是一种特殊的对象，它必须由Thread类或其子（孙）类来创建。



# 1-4 Java程序运行机制 及运行过程



### ● Java语言的特点

#### ➤ 特点一：面向对象

- ✓ 两个基本概念：类、对象
- ✓ 三大特性：封装、继承、多态

#### ➤ 特点二：健壮性

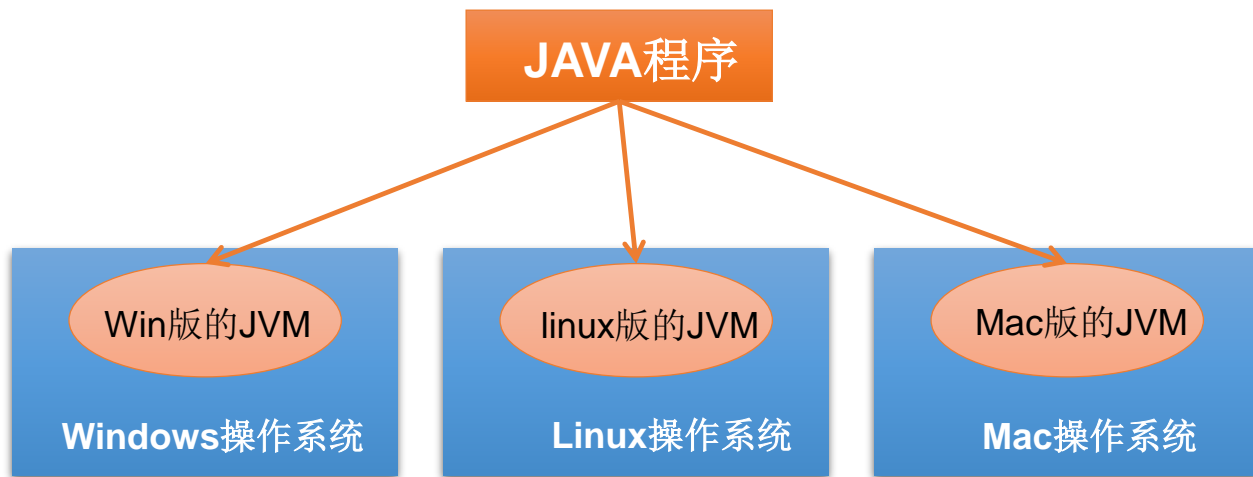
- ✓ 吸收了C/C++语言的优点，但去掉了其影响程序健壮性的部分（如指针、内存的申请与释放等），提供了一个相对安全的内存管理和访问机制

#### ➤ 特点三：跨平台性

- ✓ 跨平台性：通过Java语言编写的应用程序在不同的系统平台上都可以运行。“**Write once , Run Anywhere**”
- ✓ 原理：只要在需要运行 java 应用程序的操作系统上，先安装一个Java虚拟机 (JVM Java Virtual Machine) 即可。由JVM来负责Java程序在该系统中的运行。



- **Java语言的特点：跨平台性**



因为有了JVM，同一个Java 程序在三个不同的操作系统中都可以执行。这样就实现了Java 程序的跨平台性。





### ●Java两种核心机制

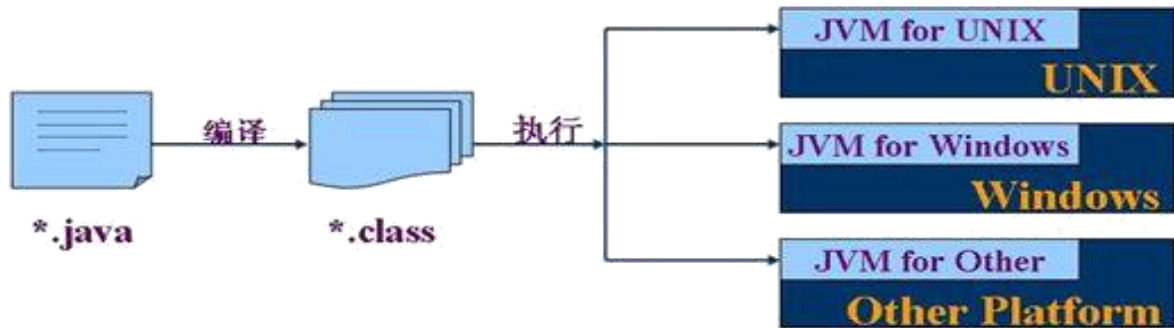
- Java虚拟机 (Java Virtual Machine)
- 垃圾收集机制 (Garbage Collection)



## 1.4 Java语言运行机制及运行过程

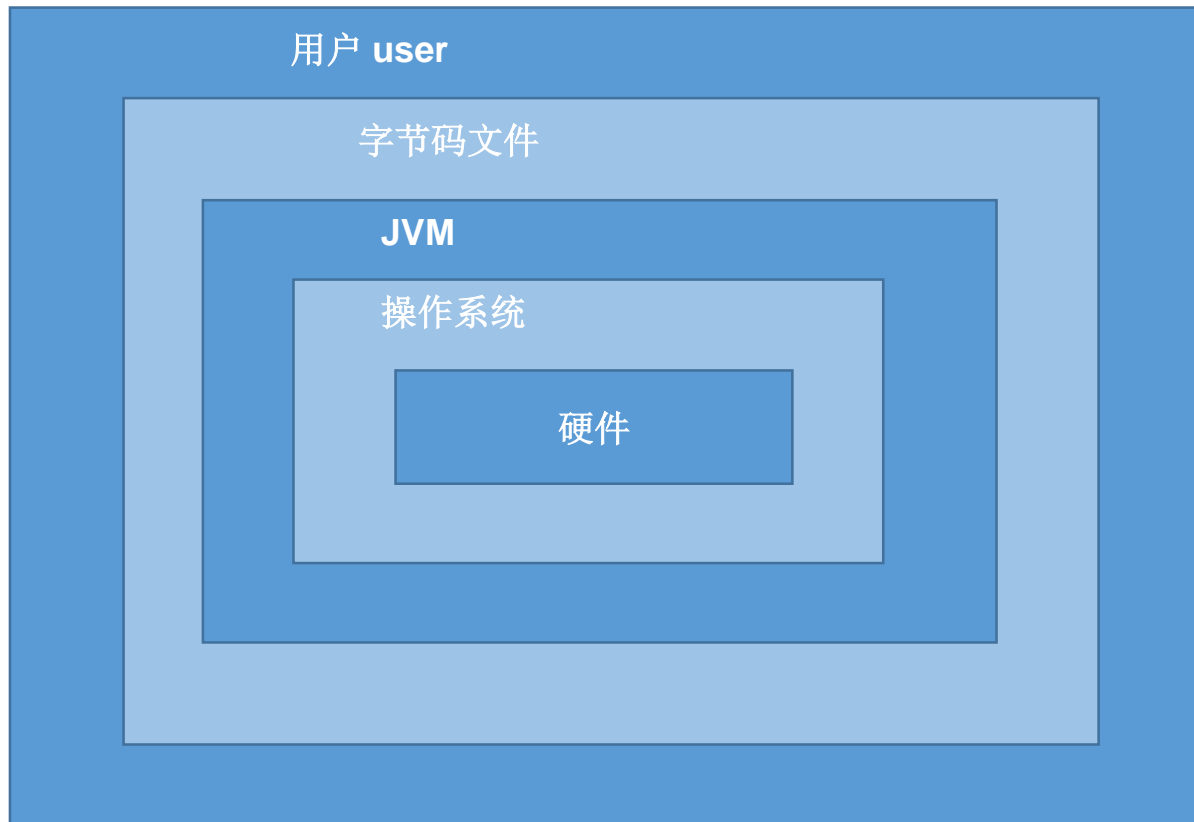
### 核心机制—Java虚拟机

- **JVM**是一个虚拟的计算机，具有指令集并使用不同的存储区域。负责执行指令，管理数据、内存、寄存器。
- 对于不同的平台，有不同的虚拟机。
- 只有某平台提供了对应的java虚拟机，java程序才可在此平台运行
- Java虚拟机机制屏蔽了底层运行平台的差别，实现了“一次编译，到处运行”





## 1.4 Java语言运行机制及运行过程





### 核心机制—垃圾回收

- 不再使用的内存空间应回收—— 垃圾回收。
  - 在C/C++等语言中，由程序员负责回收无用内存。
  - Java 语言消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空闲时，检查并释放那些可被释放的存储空间。
- 垃圾回收在Java程序运行过程中自动进行，程序员无法精确控制和干预。
- Java程序还会出现内存泄漏和内存溢出问题吗？Yes!



# 1-5 Java语言的环境搭建



7? 8? 9? 10? 11?



- 明确什么是JDK, JRE
- 下载 JDK
- 安装 JDK
- 配置环境变量
  - **path**: windows系统执行命令时要搜寻的路径。
- 验证是否成功: `javac java`
- 选择合适的文本编辑器或 IDE 开发



## 什么是JDK, JRE

### JDK(**J**ava **D**evelopment **K**it Java开发工具包)

JDK是提供给Java开发人员使用的, 其中包含了java的开发工具, 也包括了JRE。所以安装了JDK, 就不用在单独安装JRE了。

➤ 其中的开发工具: 编译工具(javac.exe) 打包工具(jar.exe)等

### JRE(**J**ava **R**untime **E**nvironment Java运行环境)

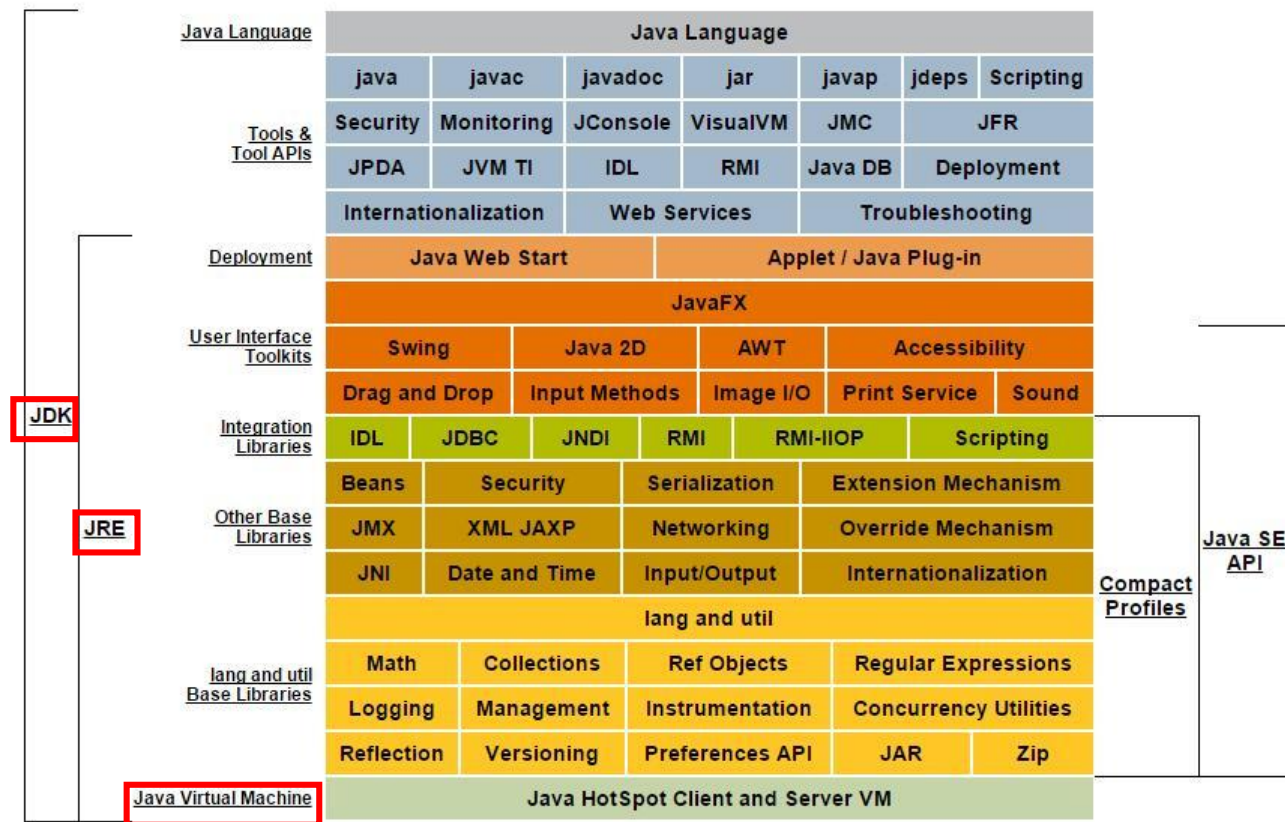
包括Java虚拟机(JVM **J**ava **V**irtual **M**achine)和Java程序所需的核心类库等, 如果想要**运行**一个开发好的Java程序, 计算机中只需要安装JRE即可。

简单而言, 使用**JDK**的开发工具完成的**java**程序, 交给**JRE**去运行。





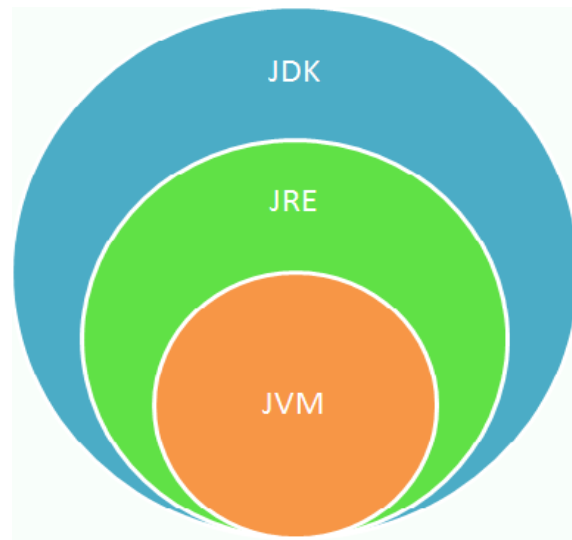
# 1.5 Java语言的环境搭建: JDK、JRE、JVM关系



Java 8.0 Platform



## 1.5 Java语言的环境搭建：JDK、JRE、JVM关系



- $\text{JDK} = \text{JRE} + \text{开发工具集}$ （例如Javac编译工具等）
- $\text{JRE} = \text{JVM} + \text{Java SE标准类库}$



## ●官方网址：

- [www.oracle.com](http://www.oracle.com)
- [java.sun.com](http://java.sun.com)

## ●安装JDK

- 傻瓜式安装，下一步即可。
- 建议：安装路径不要有中文或者空格等特殊符号。
- 如果操作系统是64位的，软件尽量选择支持64位的（除非软件本身不区分）。
- 当提示安装 JRE 时，正常在JDK安装时已经装过了，但是为了后续使用Eclipse等开发工具不报错，建议也根据提示安装JRE。



### 配置环境变量 path

- 在dos命令行中敲入javac，出现错误提示：

```
D:\developer_tools\Java\jdk1.8.0_131>javac  
'javac' 不是内部或外部命令，也不是可运行的程序  
或批处理文件。
```

- 错误原因：当前执行的程序在当前目录下如果不存在，windows系统会在系统中已有的一个名为path的环境变量指定的目录中查找。如果仍未找到，会出现以上的错误提示。所以进入到 jdk安装路径\bin目录下，执行javac，会看到javac参数提示信息。

```
D:\developer_tools\Java\jdk1.8.0_131\bin>javac  
用法: javac <options> <source files>  
其中，可能的选项包括：  
-g 生成所有调试信息
```



## 配置环境变量 path

每次执行 `java` 的工具都要进入到`bin`目录下，是非常麻烦的。可不可以有任何目录下都可以执行`java`的工具呢？

- 根据windows系统在查找可执行程序的原理，可以将`java`工具所在路径定义到 `path` 环境变量中，让系统帮我们去找运行执行的程序。
- 配置方法：
  - 我的电脑--属性--高级系统设置--环境变量
  - 编辑 `path` 环境变量，在变量值开始处加上`java`工具所在目录，后面用 “;”和其他值分隔开即可。
  - 打开DOS命令行，任意目录下敲入`javac`。如果出现`javac` 的参数信息，配置成功。

注： 具体操作流程，参看JDK8下载\_安装\_配置.doc



# 1.5 Java语言的环境搭建

path:window操作系统执行命令时，所要搜寻的路径

目的：希望D:\developer\_tools\Java\jdk1.8.0\_131路径下的命令可以在任何文件路径下执行

path

D:\developer\_tools\Java\jdk1.8.0\_131\bin;

%JAVA\_HOME%\bin;

JAVA\_HOME=

D:\developer\_tools\Java\jdk1.8.0\_131



### 配置完path环境变量以后的验证

```
C:\Users\Administrator>javac
用法: javac <options> <source files>
其中, 可能的选项包括:
    -g                生成所有调试信息
    -g:none           不生成任何调试信息
    -g:<lines,vars,source> 只生成某些调试信息
```

```
C:\Users\Administrator>java
用法: java [options] <主类> [args...]
      <执行类>
或 java [options] -jar <jar 文件> [args...]
      <执行 jar 文件>
或 java [options] -m <模块>[/<主类>] [args...]
```

```
C:\Users\Administrator>java -version
java version "9.0.1"
Java(TM) SE Runtime Environment (build 9.0.1+11)
Java HotSpot(TM) 64-Bit Server VM (build 9.0.1+11, mixed mode)
```



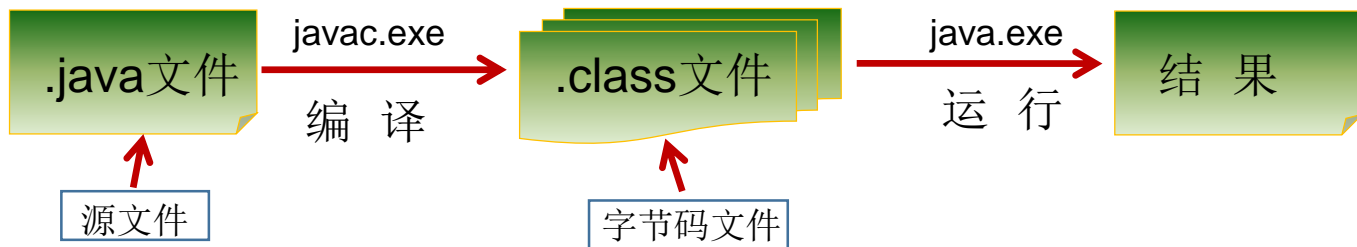
# 1-6 开发体验— HelloWorld





### ● 步骤:

1. 将 Java 代码**编写**到扩展名为 `.java` 的文件中。
2. 通过 `javac` 命令对该 `java` 文件进行**编译**。
3. 通过 `java` 命令对生成的 `class` 文件进行**运行**。





## 1.6 开发体验 — HelloWorld

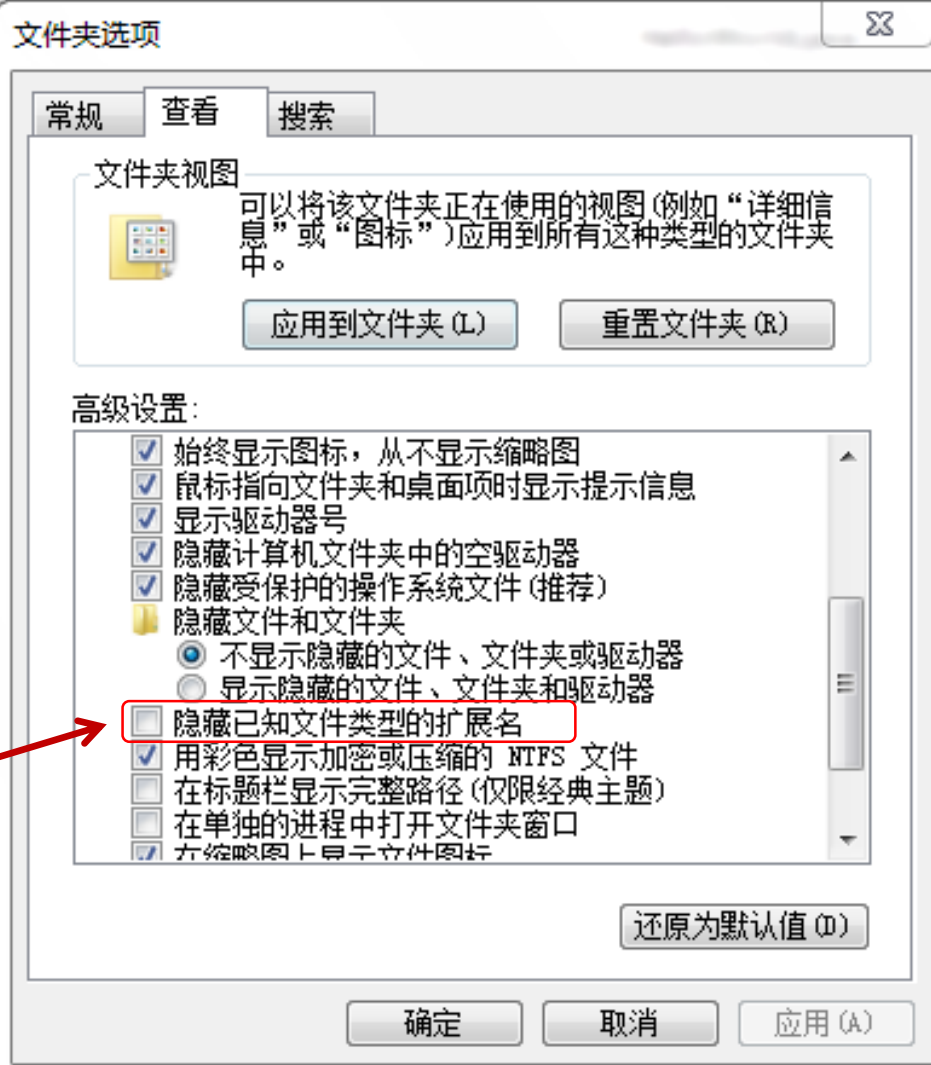
### ● 步骤一：编写

➤ 选择最简单的编辑器：记事本。

➤ 敲入代码 `class Test{ }`

将文件保存成 `Test.java`，这个文件是存放 `java` 代码的文件，称为源文件。

取消勾选





- 第一个Java程序

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



### ● 步骤二：编译

```
D:\>javac Test.java
```

- 有了java源文件，通过编译器将其编译成JVM可以识别的字节码文件。
- 在该源文件目录下，通过javac编译工具对Test.java文件进行编译。
- 如果程序没有错误，没有任何提示，但在当前目录下会出现一个Test.class文件，该文件称为字节码文件，也是可以执行的java的程序。



### ● 步骤三：运行

- 有了可执行的java程序(Test.class字节码文件)
- 通过运行工具java.exe对字节码文件进行执行。
- 出现提示：缺少一个名称为main的方法。

```
D:\>java Test  
错误: 在类 Test 中找不到主方法, 请将主方法定义为:  
public static void main(String[] args)
```

- 因为一个程序的执行需要一个起始点或者入口，所以在Test类中的加入public static void main(String[] args){ }
- 对修改后的Test.java源文件需要重新编译，生成新的class文件后，再进行执行。
- 发现没有编译失败，但也没有任何效果，因为并没有告诉JVM要帮我们做什么事情，也就是没有可以具体执行的语句。
- 想要和JVM来个互动，只要在main方法中加入一句System.out.println("Hello World");因为程序进行改动，所以再重新编译，运行即可。



## 1-7 常见问题及解决方法



## 1.7 常见问题及解决方法

```
D:\>javac Test1.java
javac: 找不到文件: Test1.java
用法: javac <options> <source files>
-help 用于列出可能的选项
```

- 源文件名不存在或者写错
- 当前路径错误
- 后缀名隐藏问题

```
D:\>java Test1
错误: 找不到或无法加载主类 Test1
```

- 类文件名写错，尤其文件名与类名不一致时，要小心
- 类文件不在当前路径下，或者不在classpath指定路径下



## 1.7 常见问题及解决方法

```
D:\>javac Test.java
Test.java:1: 错误: 类Test1是公共的, 应在名为 Test1.java 的文件中声明
public class Test1{
      ^
1 个错误
```

➤ 声明为public的类应与文件名一致，否则编译失败

```
D:\>javac Test.java
Test.java:3: 错误: 需要';'
        System.out.println("hello")
                               ^
1 个错误
```

➤ 编译失败，注意错误出现的行数，再到源代码中指定位置改错





### 总结：

学习编程最容易犯的错是语法错误。Java要求你必须按照语法规则编写代码。如果你的程序违反了语法规则，例如：忘记了分号、大括号、引号，或者拼错了单词，java编译器都会报语法错误。尝试着去看懂编译器会报告的错误信息。



## 1-8 注释(Comment)



- 用于注解说明解释程序的文字就是注释。
- Java中的注释类型：
  - 单行注释
  - 多行注释
  - 文档注释 (java特有)
- 提高了代码的阅读性；调试程序的重要方法。
- 注释是一个程序员必须要具有的良好编程习惯。
- 将自己的思想通过注释先整理出来，再用代码去体现



- 单行注释

- 格式: `//注释文字`

- 多行注释

- 格式: `/* 注释文字 */`

- 注:

- 对于单行和多行注释, 被注释的文字, 不会被JVM (java虚拟机) 解释执行。
  - 多行注释里面不允许有多行注释嵌套。



## 1.8 注 释(comment)

A：嘿 //是什么意思啊？

B：嘿.

A：呃 我问你//是什么意思？

B：问吧.

A：我刚才不是问了么？

B：啊？

A：你再看看记录...

B：看完了.

A：.....所以//是啥？

B：所以什么？

A：你存心耍我呢吧？

B：没有啊 你想问什么？

.....

不断循环之后，A一气之下和B绝交，自己苦学程序。

N年之后，A终于修成正果，回想起B，又把聊天记录翻出来看，这时，他突然发现B没有耍他.....

而他自己也不知道当年他问B的究竟是什么问题.....





- 文档注释（Java特有）

- 格式：**/\*\***

- @author** 指定java程序的作者

- @version** 指定源文件的版本

- \*/**

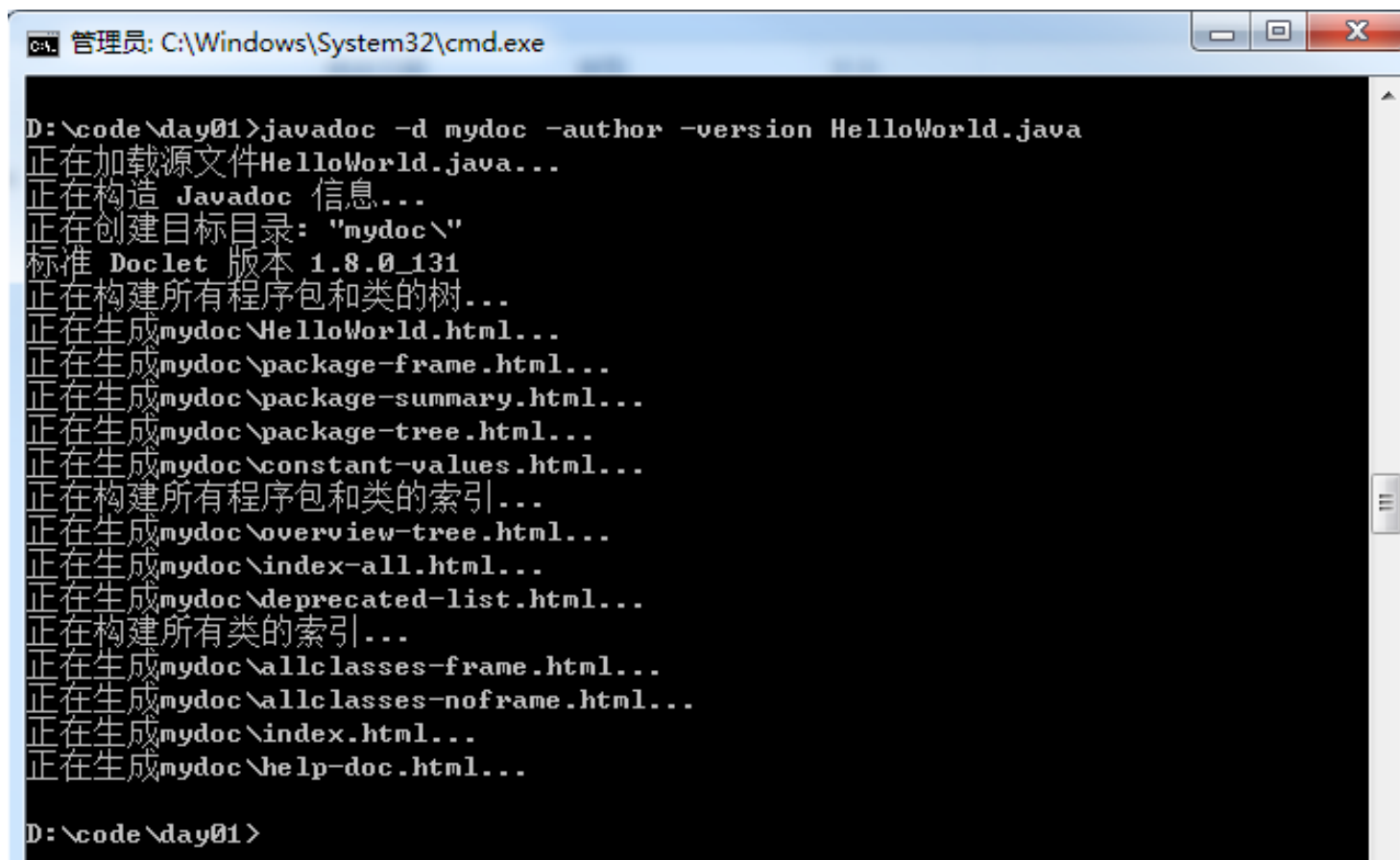
- 注释内容可以被JDK提供的工具 `javadoc` 所解析，生成一套以网页文件形式体现的该程序的说明文档。

- 操作方式

```
D:\javase\code\unit1>javadoc -d mydoc -author -version HelloWorld.java
```



## 1.8 注释(comment)



```
C:\Windows\System32\cmd.exe

D:\code\day01>javadoc -d mydoc -author -version HelloWorld.java
正在加载源文件HelloWorld.java...
正在构造 Javadoc 信息...
正在创建目标目录: "mydoc\"
标准 Doclet 版本 1.8.0_131
正在构建所有程序包和类的树...
正在生成mydoc\HelloWorld.html...
正在生成mydoc\package-frame.html...
正在生成mydoc\package-summary.html...
正在生成mydoc\package-tree.html...
正在生成mydoc\constant-values.html...
正在构建所有程序包和类的索引...
正在生成mydoc\overview-tree.html...
正在生成mydoc\index-all.html...
正在生成mydoc\deprecated-list.html...
正在构建所有类的索引...
正在生成mydoc\allclasses-frame.html...
正在生成mydoc\allclasses-noframe.html...
正在生成mydoc\index.html...
正在生成mydoc\help-doc.html...

D:\code\day01>
```



# 小结第一个程序

- Java源文件以“java”为扩展名。源文件的基本组成部分是类（class），如本例中的HelloWorld类。
- Java应用程序的执行入口是main()方法。它有固定的书写格式：  
**public static void main(String[] args) {...}**
- Java语言严格区分大小写。
- Java方法由一条条语句构成，每个语句以“;”结束。
- 大括号都是成对出现的，缺一不可。
- 一个源文件中最多只能有一个public类。其它类的个数不限，如果源文件包含一个public类，则文件名必须按该类名命名。





# 1-9 Java API文档



- API（Application Programming Interface,应用程序编程接口）是 Java 提供的基本编程接口。
- Java语言提供了大量的基础类，因此 Oracle 也为这些基础类提供了相应的 API文档，用于告诉开发者如何使用这些类，以及这些类里包含的方法。
- 下载API：  
<http://www.oracle.com/technetwork/java/javase/downloads/index.html>  
➤ Additional Resources-Java SE 8 Documentation 下载。
- 详见：**JDK8的下载-安装-配置.doc**



## 1.9 Java API的文档

Additional Resources	
<b>Oracle Java Advanced Management Console</b> Advanced Management Console (AMC) enables desktop administrators to track and manage Java usage across their organization -- understanding which Java versions are used with which applications and managing compatibility/security. AMC is a commercial product available for Java users who license Java SE Advanced or Java SE Advanced Suite. <a href="#">Learn More</a> ▶	<a href="#">DOWNLOAD</a> ↓
<b>Java SE 8 Documentation</b> <ul style="list-style-type: none"><li>▪ <a href="#">Java SE 8 Documentation</a></li><li>▪ <a href="#">Docs Installation Instructions</a></li></ul>	<a href="#">DOWNLOAD</a> ↓
<b>Java Cryptography Extension (JCE) Unlimited Strength</b>	<a href="#">DOWNLOAD</a> ↓



The screenshot shows the Java Platform, Standard Edition 7 API Specification website. The left sidebar contains two sections: 'Packages' and 'All Classes'. The 'Packages' section lists various Java packages like `java.applet`, `java.awt`, `java.awt.color`, `java.awt.datatransfer`, `java.awt.dnd`, `java.awt.event`, `java.awt.font`, and `java.awt.geom`. The 'All Classes' section lists various Java classes like `AbstractAction`, `AbstractAnnotationValueVisitor6`, `AbstractAnnotationValueVisitor7`, `AbstractBorder`, `AbstractButton`, `AbstractCellEditor`, `AbstractCollection`, `AbstractColorChooserPanel`, `AbstractDocument`, `AbstractDocument.AttributeConst`, `AbstractDocument.Content`, `AbstractDocument.ElementEdit`, `AbstractElementVisitor6`, `AbstractElementVisitor7`, `AbstractExecutorService`, `AbstractInterruptibleChannel`, and `AbstractLayoutCache`. The main content area displays the 'Java Platform, Standard Edition 7 API Specification' title and a table of packages with descriptions. The table has two columns: 'Package' and 'Description'. The packages listed are `java.applet`, `java.awt`, `java.awt.color`, `java.awt.datatransfer`, `java.awt.dnd`, `java.awt.event`, `java.awt.font`, and `java.awt.geom`. The descriptions provide details about the classes and interfaces in each package. Red boxes highlight the '包列表区' (Package List Area) and the '类列表区' (Class List Area).



# 1-10 良好的编程风格



- 正确的注释和注释风格
  - 使用文档注释来注释整个类或整个方法。
  - 如果注释方法中的某一个步骤，使用单行或多行注释。
- 正确的缩进和空白
  - 使用一次tab操作，实现缩进
  - 运算符两边习惯性各加一个空格。比如：2 + 4 \* 5。
- 块的风格
  - Java API 源代码选择了行尾风格

```
public class Test {  
    public static void main(String[] args){  
        System.out.println("Block Style!");  
    }  
}
```

行尾风格

```
public class Test  
{  
    public static void main(String[] args)  
    {  
        System.out.println("Block Style!");  
    }  
}
```

次行风格



# 1-11 常用的Java开发工具

(Integrated Development Environment)



- 文本编辑工具:

- 记事本
- UltraEdit
- EditPlus
- TextPad
- NotePad





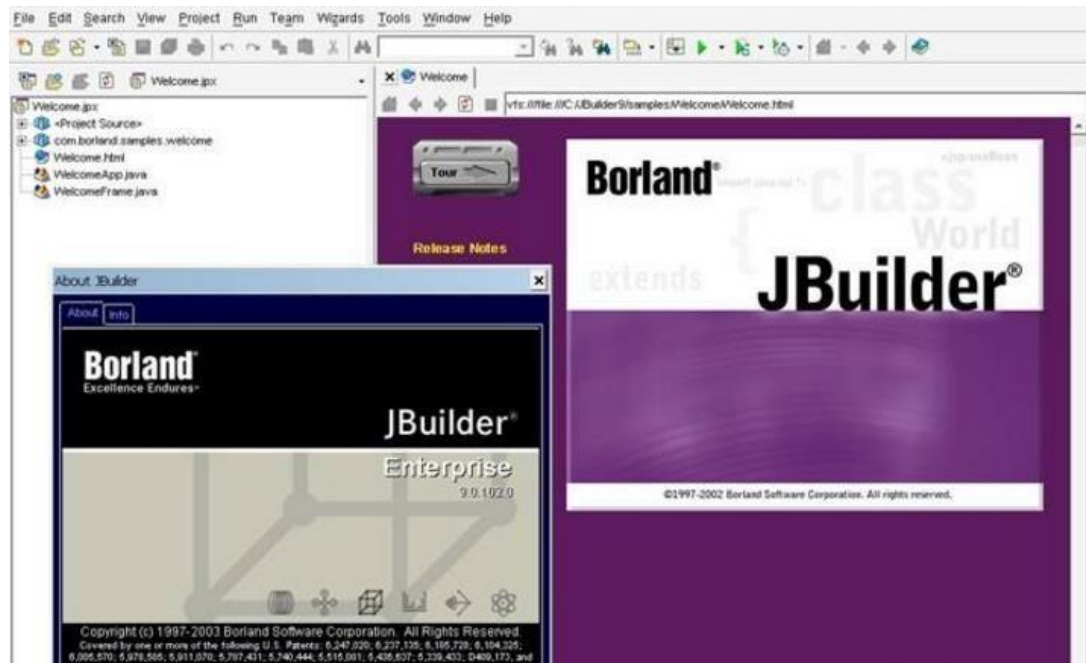
- Java集成开发环境（IDE）：

- JBuilder
- NetBeans
- Eclipse
- MyEclipse
- IntelliJ IDEA



## 1.11 常用的Java开发工具

JBuilder是Borland公司开发的针对java的开发工具，使用JBuilder将可以快速，有效的开发各类java应用。Jbuilder支持各种应用服务器。Jbuilder与Inprise Application Server紧密集成，同时支持WebLogic Server，支持EJB 1.1和EJB 2.0，可以快速开发J2EE的电子商务应用。支持远程调试和多线程调试，调试器支持各种JDK版本



<http://edn.embarcadero.com/cn/jbuilder>



## 1.11 常用的Java开发工具

NetBeans是一款用Java编写的开源IDE。既可用于Java开发，也支持其他语言，特别是PHP、C/C++，和HTML5。NetBeans开发环境提供了丰富的产品文档和培训资源以及大量的第三方插件。



<https://netbeans.org/features/index.html>



Eclipse应该是大多数Java程序员使用的第一个IDE。众所周知的、最流行、也最受欢迎的Java开发工具。优点很多：免费、更新快、代码智能化、ANT构建等，拥有众多插件，完全免费、有中文版、上手比较快。缺点也非常明显，安装插件麻烦、插件对版本要求比较严格。

<https://eclipse.org/>



The logo for MyEclipse, featuring the word "myeclipse" in a blue, lowercase, sans-serif font. The "my" is in a lighter blue color and has a slightly different, more rounded font style compared to the "eclipse" part.

MyEclipse也是一款功能强大的J2EE集成开发环境，由Genuitec公司发布，提供免费版和收费版。但免费版，只能满足基本开发需求

<http://www.myeclipsecn.com/>



## 1.11 常用的Java开发工具

IntelliJ IDEA被认为是目前Java开发效率最快的IDE工具。是JetBrains公司的产品，这家公司总部位于捷克共和国的首都布拉格。它整合了开发过程中实用的众多功能，智能提示错误，强大的调试工具，Ant，JavaEE支持，CVS整合，最大程度的加快开发的速度。简单而又功能强大。与其他的一些繁冗而复杂的IDE工具有鲜明的对比。

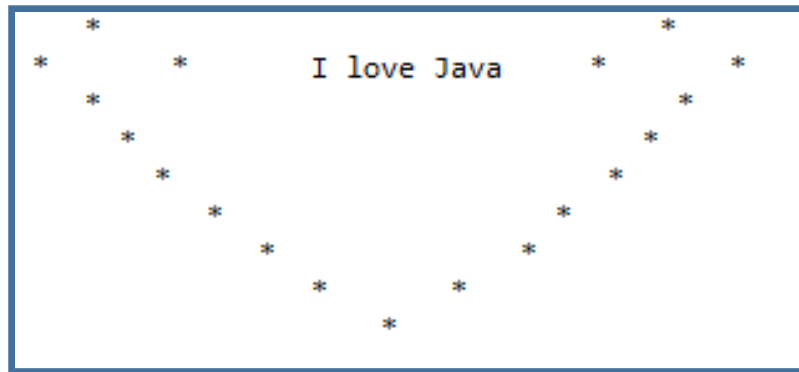
<https://www.jetbrains.com/idea/>





# 作业

1. 独立编写HelloJava程序，并配上必要的注释。
2. 将个人的基本信息（姓名、性别、籍贯、住址）打印到控制台上输出。各条信息分别占一行。
3. 结合\n(换行), \t(制表符), 空格等在控制台打印出如下图所示的效果。





# 知识回顾

- JDK,JRE,JVM的关系。
- 环境变量path配置及其作用。
- Java程序的编写、编译、运行步骤：

```
E:\_Work\Java\sample>javac HelloWorld.java  
  
E:\_Work\Java\sample>java HelloWorld  
Hello World
```

- Java程序编写的规则。
- 在配置环境、编译、运行各个步骤中常见的错误以及解决方法。



让天下没有难学的技术



尚硅谷