

# 第一章 前言

## 1.1 Java语言概述

Java语言是美国Sun公司（Stanford University Network），在1995年推出的高级的编程语言。所谓编程语言，是计算机的语言，人们可以使用编程语言对计算机下达命令，让计算机完成人们需要的功能。

## 1.2 计算机基础知识

### 1.2.1 进制

在计算机语言中常用的进制有二进制、八进制、十进制和十六进制，十进制是最主要的表达形式。

**基数：**基数是指一种进制中组成的基本数字，也就是不能再进行拆分的数字。二进制是0和1；八进制是0-7；十进制是0-9；十六进制是0-9+A-F（大小写均可）。

- 二、八、十、十六进制基数对照表

二进制 (B)	八进制 (O)	十进制 (D)	十六进制 (H)
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F

$2^0=1$ 、 $2^1=2$ 、 $2^2=4$ 、 $2^3=8$ 、 $2^4=16$ 、 $2^5=32$ 、 $2^6=64$ 、 $2^7=128$ 、 $2^8=256$ 、 $2^9=512$ 、 $2^{10}=1024$

- 十进制数据转成其他进制数据：整数部分用**除基取余法**，小数部分用**乘基取整法**
  - 例：(6)<sub>10</sub>=(110)<sub>2</sub>、(12)<sub>10</sub>=(1100)<sub>2</sub>
- 其他进制数据转成十进制数据：**按权相加法**
  - 例：(1101)<sub>2</sub>=1×2<sup>0</sup>+0×2<sup>1</sup>+1×2<sup>2</sup>+1×2<sup>3</sup>=1+4+8=(13)<sub>10</sub>
- 八进制数据转成二进制数据：将八进制的每一位用3位二进制数表示
  - (745)<sub>8</sub>=(111 100 101)<sub>2</sub>
- 二进制数据转成八进制数据：三位一组法（给定的二进制数据从低位到高位每三位划为一组，每组用其对应的八进制数表示）
  - (001 010 011)<sub>2</sub>=(1 2 3)<sub>8</sub>
- 十六进制数据转成二进制数据：将十六进制的每一位用4位二进制数表示
  - (53)<sub>16</sub>=(0101 0011)<sub>2</sub>
- 二进制数据转成十六进制数据：四位一组法（给定的二进制数据从低位到高位每四位划为一组，每组用其对应的十六进制数表示）
  - (0110 1010 1101)<sub>2</sub>=(6AD)<sub>16</sub>

## 1.2.2 字节

字节是我们常见的计算机中最小存储单元。计算机存储任何的数据，都是以字节的形式存储，右键点击文件属性，我们可以查看文件的字节大小。

- 8个bit（二进制位）0000-0000表示为一个字节，写成1byte或者1B
  - 8 bit=1 B
  - 1024 B=1 KB
  - 1024 KB=1 MB
  - 1024 MB=1 GB
  - 1024 GB=1 TB

## 1.2.3 常用DOS命令

- 进入DOS操作窗口
  - 按下 **Windows+R**，打开运行窗口，输入 **cmd** 回车，进入DOS的操作窗口

命令	操作符号
盘符切换命令	盘符名：
查看当前文件夹	dir
进入文件夹命令	cd 文件夹名
退出文件夹命令	cd ..
退出到磁盘根目录	cd \
清屏	cls

# 第二章 开发环境搭建

## 2.1 Java虚拟机——JVM

- JVM (Java Virtual Machine) : Java虚拟机, 简称VM, 是运行所有Java程序的假想计算机, 是Java程序的运行环境, 是Java 最具吸引力的特性之一。我们编写的Java代码, 都运行在JVM 之上。
- 跨平台: 任何软件的运行, 都必须运行在操作系统之上, 而我们用Java编写的软件可以运行在任意的操作系统上, 这个特性称为Java语言的跨平台特性。该特性是由JVM实现的, 我们编写的程序运行在JVM上, 而JVM运行在操作系统上。

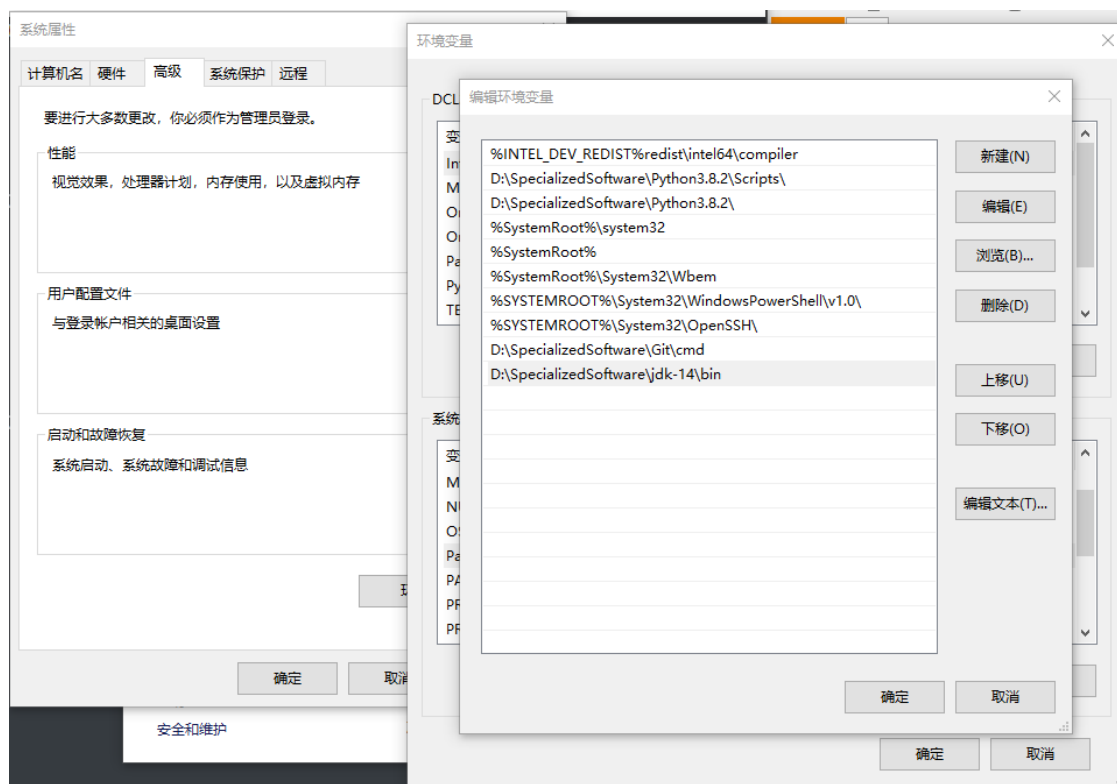


## 2.2 JRE和JDK

- JRE (Java Runtime Environment) : 是Java程序的运行时环境, 包含JVM和运行时所需要的的核心类库
- JDK (Java Development Kit) : 是Java程序开发工具包, 包含JRE和开发人员使用的工具

## 2.3 JDK14的安装及环境变量配置

- 登录<https://www.oracle.com/java/technologies/javase-downloads.html> 下载对应操作系统的安装文件及API文档
- JDK坚果云链接: <https://www.jianguoyun.com/p/DVjHjzsQtrqnCBiCol4D> ; API文档坚果云链接: [https://www.jianguoyun.com/p/DS\\_gE7wQtrqnCBjxn44D](https://www.jianguoyun.com/p/DS_gE7wQtrqnCBjxn44D)
- 百度云分享链接: [https://pan.baidu.com/s/1v7Qf41x2xcN\\_hV9eonfmoQ](https://pan.baidu.com/s/1v7Qf41x2xcN_hV9eonfmoQ) ; 提取码: yckt
- JDK14的环境变量配置不同于前几代版本, 只需要在path的最后一项加上jdk-14\bin的路径即可, 如图



- 配置完成后, 在DOS命令行中分别输入 java、javac 的命令, 出现以下界面则说明配置完成

```
C:\Windows\System32\cmd.exe
C:\Users\DCL25>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...]
      (执行模块中的主类)
或 java [options] --module <模块>[/<主类>] [args...]
      (执行模块中的主类)
或 java [options] <源文件> [args]
      (执行单个源文件程序)

将主类、源文件、-jar <jar 文件>、-m 或
--module <模块>[/<主类>] 后的参数作为参数
传递到主类。

其中，选项包括：

-cp <目录和 zip/jar 文件的类搜索路径>
-classpath <目录和 zip/jar 文件的类搜索路径>
--classpath <目录和 zip/jar 文件的类搜索路径>
      使用 ; 分隔的，用于搜索类文件的目录，JAR 档案
      和 ZIP 档案列表。

-p <模块路径>
--module-path <模块路径>
      用 ; 分隔的目录列表，每个目录
      都是一个包含模块的目录。
--upgrade-module-path <模块路径>...
      用 ; 分隔的目录列表，每个目录
      都是一个包含模块的目录，这些模块
      用于替换运行时映像中的可升级模块
--add-modules <模块名称>[/<模块名称>... ]
      除了初始模块之外要解析的根模块。
      <模块名称> 还可以为 ALL-DEFAULT, ALL-SYSTEM,
      ALL-MODULE-PATH.
--list-modules
      列出可观察模块并退出
-d <module name>
--describe-module <模块名称>
      描述模块并退出
--dry-run
      创建 VM 并加载主类，但不执行 main 方法。
      此 --dry-run 选项对于验证诸如
      模块系统配置这样的命令行选项可能非常有用。
--validate-modules
      验证所有模块并退出
```

```
C:\Windows\System32\cmd.exe
C:\Users\DCL25>javac
用法: javac <options> <source files>
其中，可能的选项包括：
@<filename>
      从文件读取选项和文件名
-Akey[=value]
      传递给注释处理程序的选项
--add-modules <模块>[/<模块>]*
      除了初始模块之外要解析的根模块。如果 <module>
      为 ALL-MODULE-PATH，则为模块路径中的所有模块。
--boot-class-path <path> -bootclasspath <path>
      覆盖引导类文件的位置
--class-path <path> -classpath <path> -cp <path>
      指定查找用户类文件和注释处理程序的位置
-d <directory>
      指定放置生成的类文件的位置
-deprecation
      输出使用已过时的 API 的源位置
--enable-preview
      启用预览语言功能。要与 -source 或 --release 一起使用。
-encoding <encoding>
      指定源文件使用的字符编码
-endorseddirs <dirs>
      覆盖签名标准路径的位置
-extdirs <dirs>
      覆盖所安装扩展的位置
-g
      生成所有调试信息
-g: {lines, vars, source}
      只生成某些调试信息
-g:none
      不生成任何调试信息
-h <directory>
      指定放置生成的本机标头文件的位置
--help, -help, -?
      输出此帮助消息
--help-extra, -X
      输出额外选项的帮助
-implicit: {none, class}
      指定是否为隐式引用文件生成类文件
-J<flag>
      直接将 <标记> 传递给运行时系统
--limit-modules <模块>[/<模块>]*
      限制可观察模块的领域
--module <模块>[/<模块>]* -m <模块>[/<模块>]*
      只编译指定的模块，请检查时间戳
--module-path <path> -p <path>
      指定查找应用程序模块的位置
--module-source-path <module-source-path>
      指定查找多个模块的输入源文件的位置
--module-version <版本>
      指定正在编译的模块版本
-nowarn
      不生成任何警告
-parameters
      生成元数据以用于方法参数的反射
-proc: {none, only}
      控制是否执行注释处理和/或编译。
-processor <class1>[,<class2>,<class3>,...]
      要运行的注释处理程序的名称；绕过默认的搜索进程
--processor-module-path <path>
      指定查找注释处理程序的模块路径
--processor-path <path> -processorpath <path>
      指定查找注释处理程序的位置
```

## 第三章 HelloWorld入门程序

程序开发的三步骤：编写、编译、运行

### 3.1 编写Java源程序

1. 在选定目录下新建文本文档，完整文件名修改为 `HelloWorld.java`，其中文件名为 `HelloWorld`，后缀名必须为 `.java`
2. 使用记事本或者Sublime Text 3打开，键入代码如下：

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

这就是第一个 `HelloWorld` 源程序，但是这个程序是给程序员看的，JVM是看不懂的，也就不能运行，所以必须将**Java源文件**编译成JVM可以看的懂的**字节码文件**

## 3.2 编译Java源文件

在DOS命令行中，进入**Java源文件的目录**，使用 `javac` 命令进行编译

- 命令： `javc Java源文件名.java`
- 例： `javac HelloWorld.java`

编译成功后，命令行不会有任何提示，在目录中会产生一个以 `.class` 结尾的新文件，这个文件就是编译后的文件，是Java的可运行文件，称为**字节码文件**

## 3.3 运行Java程序

在DOS命令行中，进入**Java源文件的目录**，使用 `java` 命令进行运行

- 命令： `java 类名字`
- 例： `java HelloWorld`

```
D:\>javac HelloWorld.java  
  
D:\>java HelloWorld  
Hello World!  
  
D:\>
```

## 3.4 关于入门程序的一些说明

### 3.4.1 编译与运行

- **编译**：是指将面向程序员的java源文件翻译成面向计算机的class字节码文件
- **运行**：是指将class字节码文件交给JVM去运行

### 3.4.2 main方法

- 主方法，写法固定不变，是程序的起始点

### 3.4.3 添加注释

- **注释**：就是对代码的解释说明，目的是便于了解及维护代码
- 注释方法：
  - 单行注释： `// 注释内容`
  - 多行注释： `/* 注释内容 */`

### 3.4.4 关键字

- 是指在程序中，Java官方已经定义好的有特殊含义的单词
- 以上程序中，出现的关键字有 `public`、`class`、`static`、`void` 等
- 关键字不需要死记硬背，出现一个记一个即可

### 3.4.5 标识符

- 是指在程序中，我们自己定义的内容，比如类的名字、方法的名字和变量的名字
- 以上程序中，出现的标识符有类名字 `HelloWorld`
- **命令规则**：硬性要求
  - 标识符可以包含：`A-Z`、`a-z`、`0-9`、`$`、`_`
  - 标识符不能以数字开头
  - 标识符不能是关键字
- **命名规范**：软性建议
  - 类名规范：首字母大写，后面每个单词首字母大写（大驼峰式）
  - 方法名规范：首字母小写，后面每个单词首字母大写（小驼峰式）
  - 变量名规范：全部小写

## 第四章 常量

是指在Java程序中固定不变的数据

- 分类

类型	含义	数据举例
整数常量	所有的整数	0, 1, 345, -3
小数常量	所有的小数	0.0, 12.66, -5.2
字符常量	单引号引起来，只能写一个字符，必须有内容	'a', '', '我'
字符串常量	双引号引起来，可以写多个字符，也可以不写	"A", "Hello", "好的",
布尔常量	只有两个值（流程控制中讲解）	true, false
空常量	只有一个值（引用数据类型中讲解）	null

- 练习

```
public class Demo {
    public static void main(String[] args) {
        //输出整数常量
        System.out.println(123);
        //输出小数常量
        System.out.println(1.25);
        //输出字符常量
        System.out.println('A');
        //输出字符串常量
        System.out.println("Hello");
        //输出布尔常量
        System.out.println(true);
    }
}
```

```
}  
}
```

# 第五章 变量

对应于常量，在Java程序中可以变化的量

Java要求一个变量每次只能保存一个数据，必须要明确保存的数据类型

## 5.1 数据类型

### 5.1.1 数据类型分类

- **基本数据类型**：整数、浮点数、字符、布尔
- **引用数据类型**：类、数组、接口

### 5.1.2 基本数据类型

- 四类八种基本数据类型

数据类型	关键字	内存占用	取值范围
字节型	byte	1个字节	-128~127
短整型	short	2个字节	-32768~32767
整型	int	4个字节	-2 <sup>31</sup> ~2 <sup>31</sup> -1
长整型	long	8个字节	-2 <sup>63</sup> ~2 <sup>63</sup> -1
单精度浮点数	float	4个字节	1.4013E-45~3.4028E+38
双精度浮点数	double	8个字节	4.8E-324~1.7977E+308
字符型	char	2个字节	0~65535
布尔类型	boolean	1个字节	true, false

Java中的默认类型，整数类型是 `int`，浮点类型是 `double`

## 5.2 变量的定义

变量定义的格式包括：数据类型、变量名、数据值

- **格式**：`数据类型 变量名 = 数据值;`
- 在同一个大括号范围内，变量的名字不可以相同
- 定义的变量，不赋值不能使用
- 练习

```
public class Demo {  
    public static void main(String[] args) {  
        //字节型变量  
        byte b = 100;  
    }  
}
```

```
System.out.println(b);
//短整形变量
short s = 1000;
System.out.println(s);
//整形变量
int i = 10000;
System.out.println(i);
//长整形变量,建议数据后加L表示
long l = 123456789L;
System.out.println(l);
//单精度浮点型变量,建议数据后加F表示
float f = 1.2F;
System.out.println(f);
//双精度浮点型变量
double d = 5.6;
System.out.println(b);
//布尔型变量
boolean bool = true;
System.out.println(bool);
//字符型变量
char c = 'A';
System.out.println(c);
}
}
```

---