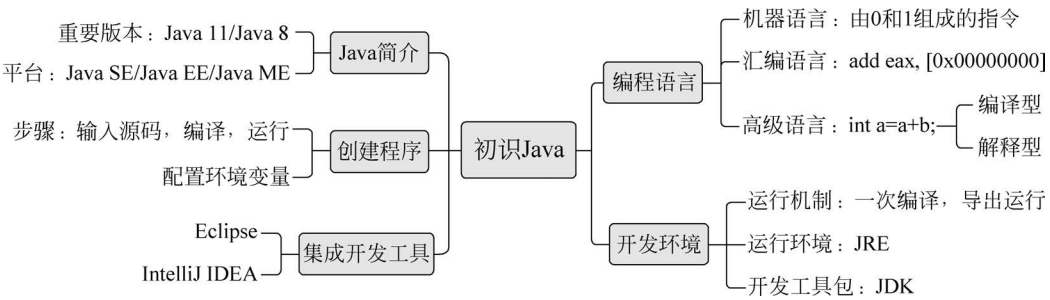
项目 1　打印软件功能菜单

技能目标

- 了解 Java 语言的发展历史和主要特点。
- 理解 Java 语言的运行机制。
- 学会 JDK 的安装与配置方法。
- 编写并运行第一个 Java 程序。
- 了解 Eclipse/IDEA 的使用方法。

知识图谱



教学重难点

教学重点：

- Java 环境的安装；
- Java 程序的结构；
- 运行 Java 程序的方法和步骤；
- 集成开发工具的基本功能；
- Eclipse 的基本用法。

教学难点：

- Java 语言的运行机制；
- 配置环境变量；
- Java 代码常见的要素；
- Java 项目的创建和导入；
- Java 语言的优缺点。

1.1 项目任务

在计算机世界中,菜单指的是计算机操作系统或者软件的操作命令目录,也可称为选项列表。因为类似于点菜的菜单,故大家习惯上将这些操作命令目录称为菜单。菜单也是软件系统的门面,是界面设计中最重要的元素,菜单的设计是否合理直接影响软件的使用效果。

如何设计并打印一个符合要求的菜单呢?一般要求从软件系统的功能需求出发,可以参考以下设计原则:

(1) 菜单通常采用"常用→主要→次要→工具→帮助"的位置排列;

(2) 菜单的使用有先后次序要求或有向导作用时,应该按先后次序排列;

(3) 没有顺序要求的菜单项按使用频率和重要性排列,重要的放在开头,次要的放在后边。

在本项目中,我们将初步接触 Java,理解 Java 的运行机制和编译过程,并使用最基本的输出功能实现"成绩管理系统"功能菜单的打印。

1.2 需求分析

根据项目任务描述,本项目需要的技术包括:搭建 Java 运行环境,配置环境变量,在记事本或集成开发工具中编写 Java 源代码文件并进行编译和运行,具体可参照以下过程:

(1) 先安装 JDK,配置环境变量;

(2) 在记事本中编写 Java 源程序;

(3) 使用 JDK 编译和运行此程序,理解 Java 的运行机制;

(4) 在 Eclipse/IDEA 中编写系统菜单打印的程序,设计系统菜单并进行输出,熟悉在集成开发环境中的程序运行步骤。

如何与计算机进行对话

1.3 技术储备

1.3.1 如何与计算机对话

如何与计算机进行对话呢?这肯定要通过语言。就像中国人用中文进行交流、美国人用英语进行交流一样,与计算机进行交流的语言叫作编程语言。换句话说,编程语言是一种向计算机发起命令且控制它行为的语言。编程语言经历了以下三个阶段。

1. 机器语言

计算机的世界是 0 与 1 的世界,从根本上说,计算机只能识别由 0 和 1 组成的指令。由于机器指令与 CPU 紧密相关,所以不同种类的 CPU 所对应的机器指令也就不同,而且它们的指令系统往往相差很大。机器语言难写难记,难以推广使用,在计算机发展初期只有极少数的专业人员才会编写计算机程序。

例如,在 x86 架构中,一个基本的加法指令可能看起来像这样:

```
81 C0 00 00 00 00
```

这里，81 是操作码，表示 32 位立即数的加法操作；C0 是 MODRM 字节，表示目标操作数是 EAX 寄存器；00 00 00 00 是立即数 0 的 32 位表示。

2. 汇编语言

程序员如果每天不停写 10101001010……很容易造成混乱。于是有人就想把这些指令用英语单词表示出来，之前 81 代表加法指令，直接写 add 岂不是更好？于是汇编语言出现了，它的可读性比机器语言增强了很多，例如，上面的加法操作在汇编语言中可能看起来像以下指令：

```
add eax, 0     ;将立即数 0 加到 eax 寄存器
```

或者

```
add eax, [0x00000000]     ;将地址 0x00000000 处的值加到 eax 寄存器
```

汇编语言中的 add、mov、loop、sub 等都是见名思义，每条指令几乎和机器指令一一对应，这样只要再拥有一个类似“翻译器”的东西，把它翻译成机器语言就可以了。但是，还是存在 0x00000000 这种内存地址信息，可见仍然是面对硬件编程，不同型号的计算机使用的汇编语言不能通用。汇编语言仍然对程序员不友好，只有懂了硬件才能编写程序，于是，面向程序员的高级语言开始诞生了。

图 1-1 演示了汇编语言的执行过程。

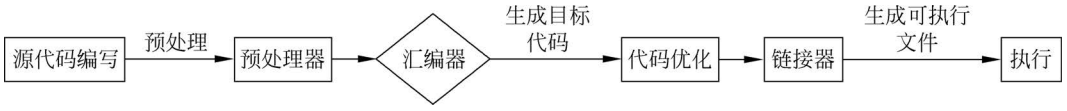


图 1-1　汇编语言执行过程

3. 高级语言

高级语言是绝大多数编程者的选择。与汇编语言相比，它不但将许多相关的机器指令合成为单条指令，并且去掉了与具体操作有关但与完成工作无关的细节，例如使用堆栈、寄存器等，这样就大大简化了程序中的指令。同时，由于省略了很多细节，编程者也就不需要有太多的专业知识。

将上述汇编语言中的加法操作改写为高级语言，如 Java，可以非常简单。

```
int value = 2;                  // 定义并初始化变量 value 为 2
int result;                     // 定义变量 result，用于存储结果
result = result + value;        // 将 value 的值累加到 result 上
```

高级语言所编制的程序不能直接被计算机识别，必须经过转换才能被执行，按转换方式可将它们分为编译型和解释型两类。

（1）编译型语言：把做好的源程序全部编译成二进制代码的可运行程序，然后直接运行这个程序。

（2）解释型语言：把做好的源程序翻译一句，然后执行一句，直至结束。

那它们之间有什么区别呢？举个例子，我们与外国人沟通时，需将中文转为外语。解释型语言的运行方式就像在进行同声翻译，我们说一句，它翻译一句；而编译型语言则先获取我们的发言稿，并将其翻译成目标外语文件，再整体进行输出，但要转为第三种外语则需要

重新进行翻译。

因此不难得出,编译型语言的执行速度快、效率高,但它依靠编译器、跨平台性差些;而解释型语言的执行速度慢、效率低,但依靠解释器、跨平台性好。

1.3.2 Java 语言发展历史和版本

Java 是 1995 年由 Sun 公司推出的一种极富创造力的高级语言,由有“Java 之父”之称的 Sun 研究院院士 James Gosling(詹姆斯·高斯林,图 1-2)创立的。Java 最初的名字是 Oak,1995 年被重命名为 Java 后正式发布。

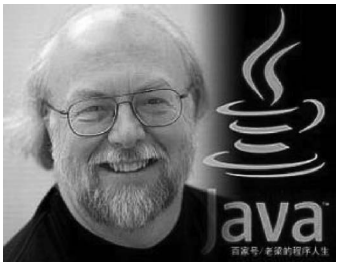


图 1-2 James Gosling

Java 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)。随着 Java 技术在 Web 方面的不断成熟,特别是其“一次编译,到处运行”的跨平台特性,已经成为 Web 应用程序的首选开发语言。

1. Java 语言的发展历程

Java 语言的发展历程如图 1-3 所示。

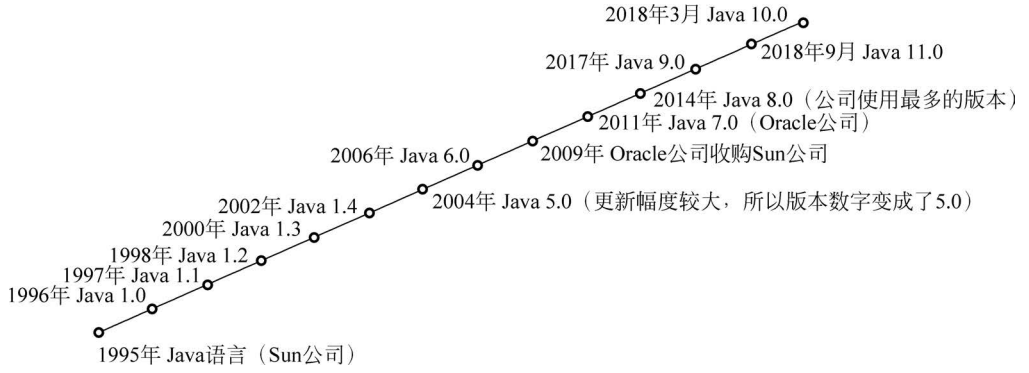


图 1-3 Java 发展历程

对应时间节点的关键事件如下。

Java 语言发展历史和版本

(1) 1996 年,发布 JDK 1.0,约 8.3 万个网页应用 Java 技术来制作。

(2) 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。

(3) 1998 年,发布 JDK 1.2,同年发布企业平台 J2EE。

(4) 1999 年,Java 分成 J2SE、J2EE 和 J2ME,JSP/Servlet 技术诞生。

(5) 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。

(6) 2005 年,J2SE 改名为 JavaSE,J2EE 改名为 JavaEE,J2ME 改名为 JavaME。

(7) 2009 年,Oracle 公司收购 Sun 公司,交易价格为 74 亿美元。

(8) 2011 年,发布 JDK 7.0。

(9) 2014 年,发布 JDK 8.0,是继 JDK 5.0 以来变化最大的版本,也是目前使用最广泛

的版本。

(10) 2017 年,发布 JDK 9.0,最大限度实现模块化。

(11) 2018 年 3 月,发布 JDK 10.0,版本号也称为 18.3。

(12) 2018 年 9 月,发布 JDK 11.0,版本号也称为 18.9。

(13) 2019 年 3 月,发布 JDK 12.0。

(14) 2019 年 9 月,发布 JDK 13.0。

……

2. Java 的三个技术平台

针对不同的开发市场,Sun 公司将 Java 划分为三个技术平台,分别是 Java SE、Java EE 和 Java ME。

(1) Java SE(Java platform standard edition,Java 平台标准版)。该版本是为开发普通桌面和商务应用程序提供的解决方案。JavaSE 是三个平台中最核心的部分,JavaEE 和 JavaME 都是从 JavaSE 的基础上发展而来的,JavaSE 平台中包括了 Java 最核心的类库,如集合、I/O、数据库连接以及网络编程等。

(2) Java EE(Java platform enterprise edition,Java 平台企业版)。该版本是为开发企业级应用程序提供的解决方案。Java EE 可以被看作一个技术平台,该平台用于开发、装配以及部署企业级应用程序,其中主要包括 Servlet、JSP、JavaBean、EJB、Web Service 等。

(3) Java ME(Java platform micro edition,Java 平台微型版)。该版本是为开发电子消费产品和嵌入式设备提供的解决方案。Java ME 主要用于微型数字子设备上软件程序的开发。例如,为家用电器增加智能化控制和联网功能,为手机增加游戏和通讯录管理功能。

1.3.3　安装 Java 开发环境

Java 是以工具包的形式进行发布的,简称 JDK(Java development kit,Java 开发工具包),在 JDK 中包含了 Java 开发工具和 JRE(Java runtime environment,Java 运行环境),而 JRE 中又包含了 Java 的基础类库和 JVM(Java virtual machine,Java 虚拟机),图 1-4 展示了这种包含关系。

JDK 是提供给 Java 开发人员使用的,如果安装了 JDK,就不用再单独安装 JRE 了。而如果不需要开发而只需要运行 Java,则只需单独下载 JRE 进行安装即可。

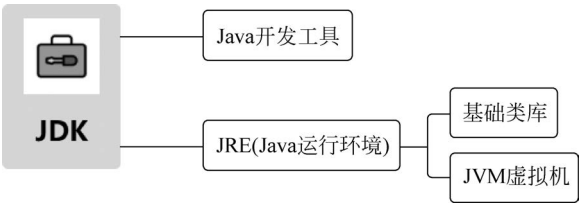


图 1-4　JDK 组成

安装 Java 开发环境

从图 1-4 可以看到,JRE 中包含了 JVM,正是有了不同操作平台的 JVM,Java 才能够实现“一次编译,到处运行”的特性。那它是怎么做到的呢? 请看图 1-5。

图 1-5 演示了 Java 语言的编译过程:Java 源代码程序首先被编译成为一种中间文件,也叫 Java 字节码文件,然后再由目标机器上的 JVM 解释成为能在该操作系统上运行的机

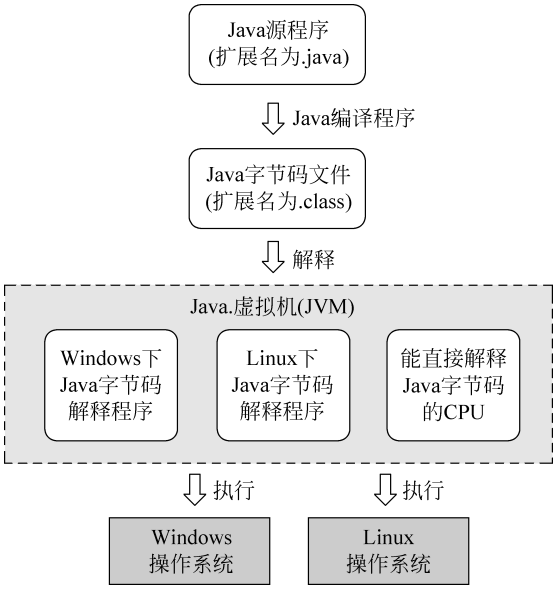


图 1-5　Java 语言编译过程

器指令。简单地说,JVM 就是一个操作系统平台的翻译机:Windows 版本的 JVM 把 Java 字节码翻译成 Windows 系统可以运行的指令,Linux 版本的 JVM 可以把 Java 字节码翻译成 Linux 系统可以运行的指令。原则上不同的平台只要有不同的 JVM 虚拟机,就可以实现 Java 的跨平台运行。

那 Java 语言到底是解释型语言还是编译型语言呢?同学们可以通过查找资料,说说自己的看法。

1. 下载 JDK

接下来,让我们登录 Oracle 公司官网提供的 JDK 下载界面(图 1-6),根据不同的操作系统,选择页面中对应的下载链接。

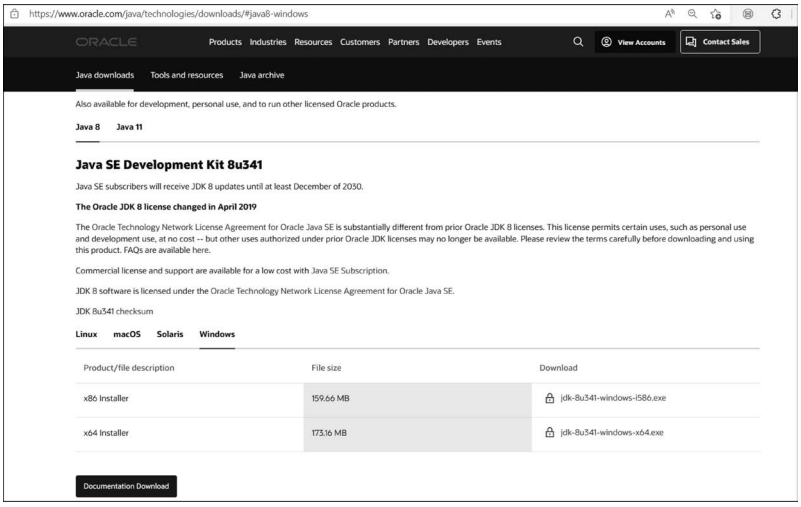


图 1-6　JDK 官方下载页面

目前的最新版本是 Java SE 18，也可选择最常用的 Java SE 8 版本的 JDK，该版本在图 1-6 中的更新版本为 8u341，文件全称为 jdk-8u431-windows-x64.exe，表示在 Windows 64 位操作系统下运行的 JDK。

2. 安装 JDK

运行下载的 JDK 安装包，按照图 1-7 的提示完成安装，默认安装位置为 Program Files\Java\。我们也可自定义目录，目录名称最好不要出现中文信息。

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 等开发工具时不报错，建议也根据提示安装 JRE。

(a) 安装首页　(b) 选择安装路径

(c) 提取安装程序　(d) 安装成功

图 1-7　JDK 安装过程

安装完成后进入安装路径，可以看到 jdk 文件夹下有如图 1-8 所示的文件结构。

- bin：该路径下存放了 JDK 的各种工具命令。
- db：该路径是安装 Java DB 的路径。
- include：一些平台特定的头文件。
- jre：该路径下安装的是运行 Java 程序所必需的 JRE 环境。
- lib：该路径下存放的是 JDK 工具命令的实际执行程序。

图 1-8　文件结构

- src.zip：该压缩文件里存放的是 Java 所有核心类库的源代码。
- LICENSE 和 README.html：说明性文档。

值得一提的是，在 JDK 的 bin 目录下放着很多重要的可执行程序，其中最重要的就是 javac.exe 和 java.exe。exe 是一种文件的后缀名，表示可执行程序的意思。后面会专门对这两个命令进行讲解。

1.3.4 创建和运行第一个 Java 程序

Hello World 的中文意思是“你好世界”。该程序的效果就是让程序展示一段文字，内容为 Hello World。程序员在学习任何一门编程语言时，第一个入门案例经常是 Hello World。

1. 创建步骤

(1) 使用最简单的记事本编写程序。打开记事本，输入以下代码，注意代码要区分大小写，具体的代码作用在本项目后面再进行介绍。

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

(2) 代码编写完成后，将代码保存为 Welcome.java 文件，并保存在图 1-8 所示的 JDK 安装目录所在的 bin 目录下。这里需要注意，代码中的 Welcome 类名必须和 Welcome.java 的文件名部分(不包括扩展名)完全一致(包括大小写)，否则后面的编译会失败。

(3) 在 Windows 桌面选择“开始”→“Windows 系统”→“命令提示符”命令，打开命令执行窗口，此窗口的当前路径和 JDK 的安装路径不一致，需要使用 DOS 指令切换到当前目录，可以按照图 1-9 的方式通过 DOS 命令进入到当前的执行目录，可看到当前的目录为 D:\Java\jdk1.8.0_291\bin，这是 JDK 的安装目录。

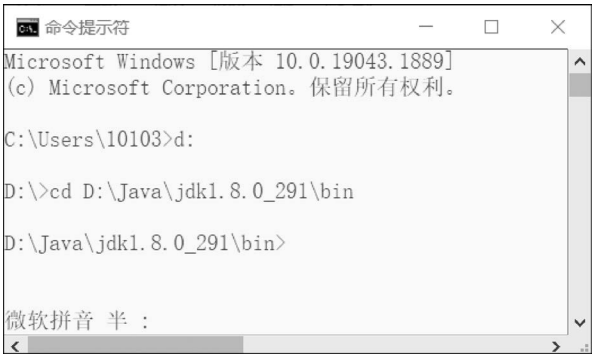


图 1-9 进入安装目录

第一个 Java 程序

小贴士 认识 DOS

DOS(disk operating system，磁盘操作系统)是早期个人计算机上的操作系统，其特点是可以直接操纵管理硬盘的文件，并以 DOS 窗口的形式运行。

DOS 的常见命令举例如下。

- dir：显示指定路径上所有文件或目录的信息，格式为“dir[盘符：][路径][文件名][参数]”
- cd：进入指定目录，格式为“cd [路径]”
- md：建立目录，格式为“md [盘符：] [路径]”
- rd：删除目录，格式为“rd [盘符][路径]”

(4) 在当前窗口输入 javac Welcome.java，对 Java 程序进行编译，得到 Welcome.class 字节码文件。

(5) 在当前窗口输入 java Welcome，运行 Java 程序，控制台出现“Hello World!”的运行结果。

整个过程如图 1-10 所示。

图 1-10　运行 Java 程序

2. 程序运行机制

上述代码能够运行，主要依靠两个非常重要的命令，即 java.exe 和 javac.exe。这两个命令都在 JDK 安装目录的 bin 文件夹里。其中 javac.exe 命令用于编译 Java 源代码，形成后缀名为 class 的字节码；java.exe 命令用于执行编译好的 Java 字节码。最终将运行结果呈现在控制台上，具体过程如图 1-11 所示。

那是不是所有的 Java 源代码都必须保存在 JDK 安装目录的 bin 子目录里呢？肯定不是的。之所以将源代码放在 bin 子目录下，是因为 java.exe 和 javac.exe 两个命令也在这个子目录里。比如，如果把 Welcome.java 放在其他目录里，如 D:\test，运行时会出现图 1-12 中第一条命令执行时显示的错误，意味着系统不认识 javac 命令。如果在 D:\test 目录下要成功运行 Welcome.java，就必须在 javac 命令前面加上完整的路径，如图 1-12 中的第二条和第三条命令执行的效果，此时 javac 和 java 命令后面都加了完整路径。

3. 配置环境变量

新编写的代码如果存放在 bin 目录下，可以直接使用 javac 和 java 命令执行；如果代码存放在其他目录下，就需要指定这两个命令的完整路径才能执行。如果想把代码存放在任意目录下并直接使用 javac 和 java 命令执行，该怎么设置才能不需要输入完整路径呢？

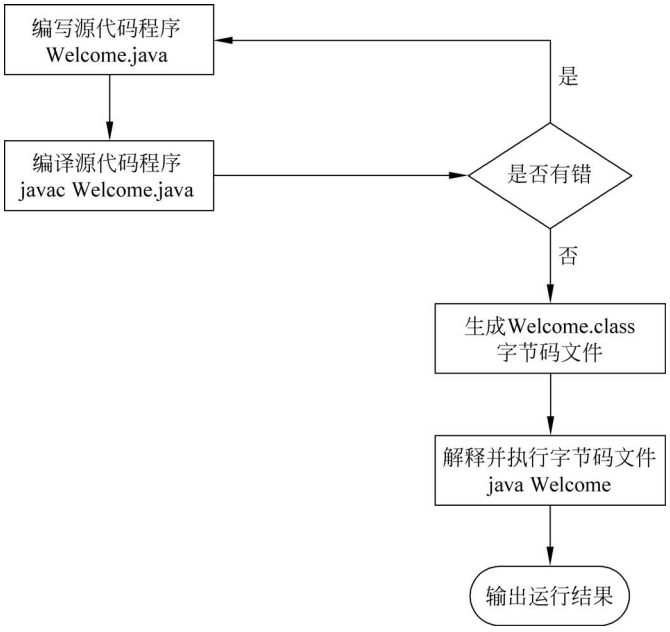


图 1-11　Java 程序运行机制

图 1-12　完整路径指令

可以把 javac 和 java 命令配置到系统的环境变量当中，使得在任何的控制台窗口下都可以直接编译和执行 Java 程序而不需要输入完整路径。

那么什么是环境变量呢？环境变量一般是指在操作系统中用来指定操作系统运行环境的一些参数，如临时文件夹位置和系统文件夹位置等。环境变量有很多种，其中系统的 Path 环境变量用来指定可运行程序的路径。

除了需要配置 Path 环境变量，通常还需要配置 JAVA_HOME 的环境变量，这是因为 Eclipse、Tomcat 等一些软件就是通过搜索 JAVA_HOME 变量来找到并使用安装好的 JDK。至于为什么一定是 JAVA_HOME 这个名字呢？那就是约定俗成的事情了。

配置环境变量的步骤如下：