

河南中医药大学信息技术学院（智能医疗行业学院）智能医学工程专业《计算机程序设计》课程

第12章：开发控制台软件

王猛

河南中医药大学信息技术学院（智能医疗行业学院）与河南方和信息科技有限公司 联合建设
河南中医药大学信息技术学院智能医疗教研室
<https://aitcm.hactcm.edu.cn>
2025/2/16

1

本章概要

- 12.1 DOS命令
 - 12.1.1 DOS系统的基本原理
 - 12.1.2 绝对路径和相对路径
 - 12.1.3 DOS窗口如何打开
 - 12.1.4 常用DOS命令
- 12.2 JDK JRE和JVM间的关系
 - 12.2.1 JDK
 - 12.2.2 JRE
 - 12.2.3 JVM
 - 12.2.4 Java API
- 12.3 通过DOS命令编译并运行.java文件
 - 12.3.1 输入DOS命令前需要注意的问题
 - 12.3.2 简单编写一个.java源文件
 - 12.3.3 使用javac命令生成.class字节码文件
 - 12.3.4 生成.class文件后通过DOS命令运行java程序
 - 12.3.5 实践java程序的解释---HelloWorld
 - 12.3.6 输入与交互



2

12.1 DOS命令

12.1.1 DOS系统的基本原理

- 命令控制台

在window(可视化,图形化操作)系统出来之前,主流的操作系统是DOS(Disk Operation System磁盘操作系统的缩写)是个人计算机上的一类操作系统。它直接操纵管理硬盘的文件，一般都是黑底白色文字的界面。



12.1 DOS命令

12.1.1 DOS系统的基本原理

- 从1981年直到1995年的15年间，磁盘操作系统在IBM PC 兼容机市场中占有举足轻重的地位。而且，若是把部分以DOS为基础的Microsoft Windows版本，如Windows 95、Windows 98和Windows Me等都算进去的话，那么其商业寿命至少可以算到2000年。微软的所有后续版本中，磁盘操作系统仍然被保留着。

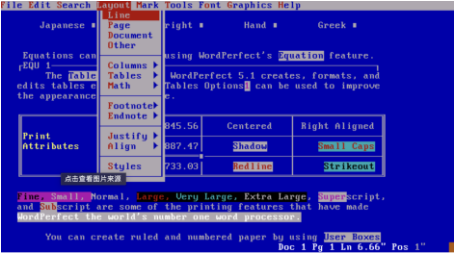


12.1 DOS命令

12.1.1 DOS系统的基本原理

- 经典DOS应用程序

- 金山WPS
- Norton
- 电子表格处理软件
- 单机游戏



12.1 DOS命令

12.1.2 绝对路径和相对路径

- 什么是路径

- 在计算机科学中，路径是文件系统中的一串字符，用于指定文件或目录的位置。 路由相对路径和绝对路径两种形式组成，分别用于不同的情境和目的。例如：

Windows:

C:\Users\YourName\Documents\file.txt

Linux:

/usr/local/java

12.1 DOS命令

12.1.2 绝对路径和相对路径

- 相对路径

- 相对路径是指从当前工作目录到目标文件或目录的路径，它不包含根目录的信息。相对路径的优点是简洁易用，用户不需要知道整个文件系统的结构，只需指定当前目录和目标目录之间的相对位置即可。

相对路径的缺点是在某些情况下可能不够准确，因为它依赖于当前工作目录的状态。例如：

`\Documents\file.txt`

12.1 DOS命令

12.1.2 绝对路径和相对路径

- 绝对路径的应用场景

- 系统文件和配置文件的定位

操作系统中的系统文件和配置文件通常使用绝对路径来定位，以确保文件的正确性和稳定性。

- 跨程序的数据共享

在一些情况下，不同程序需要访问相同的文件，此时可以使用绝对路径来确保文件的位置不会因为当前工作目录的不同而改变。

12.1 DOS命令

12.1.2 绝对路径和相对路径

- 相对路径的应用场景

- 项目内部文件引用

在同一个项目内部，不同文件之间通常使用相对路径来引用，以简化文件的路径表示和管理工作。

- 用户自定义的目录结构

用户可以根据自己的需要和习惯来设置工作目录，使用相对路径来访问文件或目录，以提高工作效率和便捷性。

12.1 DOS命令

12.1.3 DOS窗口如何打开

- 在Windows系统中打开DOS窗口

- 启动命令提示符
- 用户可以通过在开始菜单中搜索"命令提示符"或"cmd"，然后点击打开来启动DOS窗口。另外，用户
- 还可以通过按Win+R键，输入"cmd"命令，然后回车来快速打开命令提示符。
- 启动PowerShell
 - PowerShell是Windows系统中另一个功能强大的命令行工具，用户可以通过在开始菜单中搜索"Powershell"，然后点击打开来启动DOS窗口。与命令提示符相比，PowerShell提供了更多的功能和命令，可以满足更高级的系统管理和开发需求。

12.1 DOS命令

12.1.3 DOS窗口如何打开

- 在其他系统中打开命令行窗口
 - 在Linux系统中打开终端
 - 在Linux系统中，用户可以通过打开终端来使用命令行界面。终端是Linux系统中进行命令行操作的主要工具，用户可以通过终端执行各种命令，进行系统管理和软件安装等操作。
 - 在macOS系统中打开终端
 - 在macOS系统中，用户也可以通过打开终端来使用命令行界面。与Linux系统类似，macOS系统的终端也是用户进行命令行操作的主要工具，可以满足各种系统管理和开发需求。

12.1 DOS命令

12.1.4 DOS常用命令

- DOS命令的分类
 - 文件操作命令
 - 创建文件: **CREATE** <文件名>
 - 删除文件: **ERASE** <文件名>
 - 打开文件: **OPEN** <文件名>
 - 关闭文件: **CLOSE** <文件名>
 -

12.1 DOS命令

12.1.4 DOS常用命令

- DOS命令的分类

- 目录操作命令

- 创建目录: **MKDIR** <目录名>
 - 删除目录: **RMDIR** <目录名>
 - 切换目录: **CHDIR** <目录名>
 - 列出目录内容: **DIR**
 -

12.1 DOS命令

12.1.4 DOS常用命令

- DOS命令的分类

- 系统操作命令

- 关机: **SHUTDOWN**
 - 重启计算机: **REBOOT**
 - 显示系统信息: **SYSTEMINFO**
 - 清屏: **CLS**
 -

12.1 DOS命令

12.1.4 DOS常用命令

- DOS命令格式

- 基本格式

- 命令+ 参数+ 选项

- 参数格式

- 必选参数: 在使用命令时必须提供的参数 可选参数:在使用命令时可以根据需要提供的参数 参数值:参数的具体值, 用于指定命令的操作对象

12.1 DOS命令

12.1.4 DOS常用命令

- DOS命令示例

- 文件操作

- 创建文件: **CREATE test.txt**
 - 删除文件: **ERASE test.txt**

- 目录操作

- 创建目录: **MKDIR testDir**
 - 删除目录: **RMDIR testDir**

12.1 DOS命令

12.1.4 DOS常用命令

- DOS命令示例

- 系统操作

- 关机: **SHUTDOWN s t 0** (**s** 表示关机, **t** 表示延迟时间, **0** 表示立即关机)
 - 重启计算机: **REBOOT f** (**f** 表示强制重启)

12.2 JDK JRE 和 JVM间的关系

12.2.1 JDK

- JDK的概念

- JDK的定义

- JDK是**Java Development Kit**的缩写, 中文名称为Java开发工具包。JDK是Java语言的开发环境, 它提供了编译Java代码、运行Java应用程序所需的库和其他工具。JDK由Sun Microsystems公司(现已被甲骨文公司收购)开发, 是Java语言的标准开发工具。

12.2 JDK JRE 和 JVM间的关系

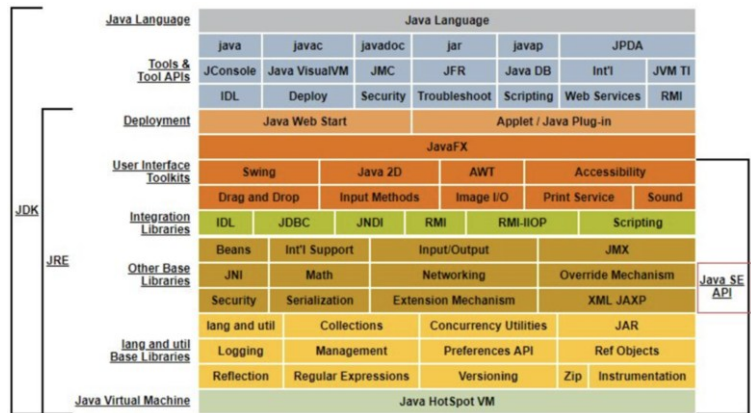
12.2.1 JDK

- JDK的概念
 - JDK的版本
 - JDK从1.0版本开始，到目前为止已经发布了多个版本，每个版本都带来了新的特性和改进。常见的JDK版本有JDK 8。
 - JDK 7、JDK 8、JDK9、JDK 10、JDK 11、JDK 17、JDK 21、JDK 22等。每个版本的JDK都在语言特性、性能、安全性等方面进行了改进和优化。

12.2 JDK JRE 和 JVM间的关系

12.2.1 JDK

- JDK结构体系



12.2 JDK JRE 和 JVM间的关系

12.2.1 JDK

- Java编译器(javac)

- Java编译器(javac)

- Java编译器是JDK中的一个重要组件，用于将Java源代码文件(.java)编译成字节码文件(.class)。编译器检查源代码中的语法错误，并将源代码转换为平台无关的字节码，以便在Java虚拟机(JVM)上运行。

- 编译器的使用

- 用户可以通过命令行输入javac命令后跟源文件名来编译Java源代码。编译成功后，会生成对应的字节码文件，可以在JVM上运行。

12.2 JDK JRE 和 JVM间的关系

12.2.1 JDK

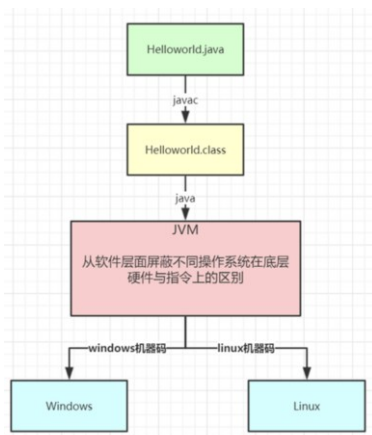
- Java编译器(javac)

- JVM是Java Virtual Machine（Java虚拟机）的缩写。是运行在计算机内存中的一个解析Java class文件的核心虚拟机。它首先通过类ClassLoader类的加载机制把JRE的核心初始启动的类加载到JVM中。保证正常启动运动。
 - Java的跨平台特性正式由于JVM在不同的操作系统上底层调用的不同硬件指令所实现。

12.2 JDK JRE 和 JVM间的关系

12.2.1 JDK

● Java编译器(javac)



12.2 JDK JRE 和 JVM间的关系

12.2.2 JRE

● Java运行时环境(JRE)

- JRE是Java程序运行的基础环境，提供了Java虚拟机(JVM)和Java标准库(Java API)。JRE负责加载字节码文件，并在虚拟机上执行，同时提供Java API供Java程序调用。
- JVM: Java虚拟机，负责加载和执行字节码文件。
- Java API :一组预定义的函数和接口、供Java程序调用。

- <https://docs.oracle.com/javase/8/docs/api/index.html>

12.2 JDK JRE 和 JVM间的关系

12.2.3 JVM

- JVM的定义与发展

- JVM的诞生背景

- JVM (Java虚拟机)的诞生源于Java语言的可移植性需求。Java语言在1995年由Sun Microsystems公司推出，其最大的特点就是"一次编写，到处运行"。JVM的推出使得Java语言能够在不同的操作系统和硬件平台上运行，极大地提高了开发效率。

12.2 JDK JRE 和 JVM间的关系

12.2.3 JVM

- JVM的定义与发展

- JVM的发展历程

- 1995年，JVM1.0版本发布，支持Java语言的基本运行机制。2004年，JVM2.0版本发布，引入了JT(JustIn Time)编译器、大大提高了Java程序的运行效率。2014年，JVM9.0版本发布，正式引入了GraalVM，进一步提高了Java程序的性能和可移植性。

12.2 JDK JRE 和 JVM间的关系

12.2.3 JVM

- JVM的核心组件

- 类加载器(Class Loader)

- 类加载器是JVM的核心组件之一，负责将Java字节码文件加载到JVM中。JVM有三种类加载器:Bootstrap Class Loader、ExtensionClass Loader和Application Class Loader.

- 运行时数据区(Runtime Data Area)

- 运行时数据区是JVM在运行Java程序时所需要的一系列数据结构，包括方法区、堆、栈、本地方法栈和程序计数器。这些数据结构存储了Java程序的运行状态和数据信息。

12.2 JDK JRE 和 JVM间的关系

12.2.3 JVM

- JVM的核心组件

- 执行引擎(Execution Engine)

- 执行引擎是JVM的核心组件之一，负责执行Java字节码。JVM的执行引擎包括解释器、即时编译器(JIT)和垃圾回收器等。

- 本地方法接口(Native Interface)

- 本地方法接口是JVM与本地操作系统和硬件进行交互的接口。通过本地方法接口，Java程序可以调用本地交互。

12.2 JDK JRE 和 JVM间的关系

12.2.4 Java API

- JAVA API的定义和作用

- Java SE API 的定义和作用
- Java SE API (**Java Standard Edition Application Programming Interface**)是一组用于开发 Java 应用程序的预定义的接口和类。
- 通过使用 Java SE API、开发者可以实现各种功能、如输入输出操作、网络编程、数据结构操作等。
- Java SE API提供了 Java 编程语言的核心功能，是 Java 开发的基础。

12.2 JDK JRE 和 JVM间的关系

12.2.4 Java API

- Java SE API 的版本和更新

- Java SE API 随着 Java 版本的更新而不断发展和完善，
- 每个版本的 Java SE API都可能包含新增的类和接口，以及废弃或修改的类和接口。
- 开发者需要关注 Java 版本的更新，以便及时了解和掌握最新的 API变化。

12.2 JDK JRE 和 JVM间的关系

12.2.4 Java API

- 常用类和接口概述

- Java SE API 中包含许多常用的类和接口，用于实现各种功能。以下是一些常用的类和接口：
- 集合框架
 - List 接口:实现有序集合，如 ArrayList、LinkedList 等。Set 接口:实现无序且不重复的集合，如 HashSet、TreeSet 等。Map 接口:实现键值对的映射，如 HashMap、TreeMap 等。
- 输入输出
 - InputStream 类:用于读取数据，如 FileInputStream、ByteArrayInputStream等。OutputStream 类:用于写入数据，如FileOutputStream等。Reader 类:用于读取字符数据,如 FileReader、BufferedReader 等。Writer 类:用于写入字符数据，如 FileWriter、BufferedWriter 等。

12.3 通过DOS命令编译并运行.java文件

12.2.4 Java API

- 常用类和接口概述

- 网络编程
 - InetAddress 类: 用于表示网络地址，如 Socket 类: 用于网络通信，ServerSocket 类: 用于创建服务器端Socket,
- 数据结构
 - 如数组array; 字符串String: 用于表示字符序列; 日期和时间: 如Date 类、Calendar 类等。

12.3 通过DOS命令编译并运行.java文件

12.3.1 输入DOS命令前需要注意的问题

- 输入DOS命令前的注意事项

- 命令格式与参数

- 命令格式

- DOS命令一般由命令名称和参数组成，参数可以指定命令的执行方式和对象。命令格式要正确，否则可能导致命令无法执行或产生错误结果。

- 参数输入

- 参数输入要准确，包括文件路径、文件名、参数值等，否则可能导致命令执行不正确或失败。

12.3 通过DOS命令编译并运行.java文件

12.3.1 输入DOS命令前需要注意的问题

- 输入DOS命令前的注意事项

- 命令提示符

- 命令提示符是指计算机操作系统中用户输入命令的地方。命令提示符有不同类型，如命令提示符(CMD)、PowerShell等，要根据不同的提示符进行相应的操作。

- 命令执行目录

- 命令执行目录是指DOS命令执行时所在的目录。确保命令执行目录正确，否则可能导致命令无法找到指定的文件或目录。

12.3 通过DOS命令编译并运行.java文件

12.3.1 输入DOS命令前需要注意的问题

- 输入DOS命令前的注意事项

- 命令权限

- 命令权限是指用户执行DOS命令所需的权限。确保用户具有执行命令所需的权限，否则可能导致命令无法执行或产生错误结果。

- 命令安全性

- 命令安全性是指执行DOS命令可能带来的风险。注意避免执行来路不明的命令，防止计算机受到恶意攻击。

12.3 通过DOS命令编译并运行.java文件

12.3.1 输入DOS命令前需要注意的问题

- 输入DOS命令前的注意事项

- 命令帮助

- 命令帮助是指通过帮助命令(如help)查看命令的使用方法和参数说明。在使用不熟悉的命令时，先查看命令帮助，了解命令的使用方式和参数含义。

- 命令文档

- 命令文档是指详细介绍命令使用方法和参数说明的文档。在需要深入了解命令的使用时，可以查阅相关的命令文档。

12.3 通过DOS命令编译并运行.java文件

12.3.2 简单编写一个.java源文件

- 创建一个Java源文件
 - 使用文本编辑器或IDE创建一个名为HelloWorld.java的文件。输入Java代码并保存。

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

12.3 通过DOS命令编译并运行.java文件

12.3.3 使用javac命令生成.class字节码文件

- 编译Java源文件
 - 在命令行输入javac HelloWorld.java进行编译。成功编译后，会在同一目录下生成HelloWorld.class文件。



```
管理员: C:\WINDOWS\system... X + v  
Microsoft Windows [版本 10.0.22621.2283]  
(c) Microsoft Corporation。保留所有权利。  
C:\Users\Administrator>e:  
E:\>javac HelloWorld.java  
E:\>
```

12.3 通过DOS命令编译并运行.java文件

12.3.4 生成.class文件后通过DOS命令运行java程序

- 运行Java程序

- 在命令行输入java HelloWorld运行程序。输出"Hello, World!"表示程序运行成功。

```

管理员: C:\WINDOWS\system...
Microsoft Windows [版本 10.0.22621.2283]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Administrator>e:

E:\>javac HelloWorld.java

E:\>java HelloWorld
Hello, World!
  
```

12.3 通过DOS命令编译并运行.java文件

12.3.5 实践java程序的解释---HelloWorld

- 实践java程序的解释---HelloWorld

类修饰符
公共类

类名

Java 中的主运行方法，它和 C/C++ 中的 main() 作用是一样的，就是所有的程序都从 "main()" 中开始执行。要执行 Java 程序，必须有一个包括主运行方法的类。

```

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

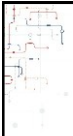
"System.out.println()" 是 Java.lang 包的一个方法，用来将字符串 "Hello world" 输出到命令行窗口。



12.3 通过DOS命令编译并运行.java文件

12.3.6 输入与交互

- Java输入与交互的基本概念
 - 输入输出流
 - Java中的输入输出流是实现程序输入与交互的重要方式。输入流负责从外部读取数据到程序中，输出流负责将程序中的数据写入到外部。Java提供了丰富的输入输出流类库，如InputStream、OutputStream、Reader、Writer等。
 - 交互概念
 - 程序交互指的是程序与用户之间的数据交换。Java程序可以通过命令行输入、图形用户界面(GUI)、网络通信等方式与用户进行交互。



12.3 通过DOS命令编译并运行.java文件

12.3.6 输入与交互

- Java程序输入与交互的方法
 - 命令行输入
 - Java程序可以通过Scanner类实现命令行输入。使用Scanner类可以方便地从键盘读取用户的输入数据。
 - 示例: `Scanner scanner = newScanner(System.in);int num =scanner.nextInt();`
 - 图形用户界面(GUI)
 - Java Swing和JavaFX是实现GUI的常用库。通过GUI可以创建按钮、文本框、菜单等控件，实现与用户的交互。示例:使用Swing创建一个简单的输入界面，包括一个文本框和一个按钮，当用户点击按钮时，程序可以获取文本框中的输入数据。
 - 网络通信
 - Java中的网络通信丰要通过Socket编程实现。客户端和服务端通过Socket实现数据传输和交互。示例:创建一个简单的网络聊天程序客户端发送消息到服务器，服务器将消息返回给客户端。

12.3 通过DOS命令编译并运行.java文件

12.3.6 输入与交互

- 文件输入输出

- 文件读取与写入

- Java中使用FileInputStream和FileOutputStream类实现文件读写。示例:使用FileInputStream读取文件内容,使用FileOutputStream将数据写入文件。

- 文本文件处理

- Java中使用BufferedReader和BufferedWriter类处理文本文件。示例:使用BufferedReader从文本文件中读取数据,使用BufferedWriter向文本文件中写入数据。

12.3 通过DOS命令编译并运行.java文件

12.3.6 输入与交互

- 数据库交互

- JDBC技术

- JDBC (Java Database Connectivity)是Java实现数据库连接的标准规范。示例:使用JDBC连接MySQL数据库,执行SQL查询并处理结果。

- 数据库操作

- Java程序可以通过JDBC实现对数据库的增删改查操作。示例:创建一个Java程序,实现用户注册功能、将用户信息存储到数据库中。

12.3 通过DOS命令编译并运行.java文件

12.3.6 输入与交互

- 多线程处理输入与交互
 - 线程的概念
 - Java中线程是程序执行的基本单位，多线程可以实现程序的并发执行。示例:创建一个简单的多线程程序，实现两个线程交替输出数字。
 - 线程在输入与交互中的应用
 - 在Java程序中，可以使用多线程实现同时处理多个用户的输入与交互。示例:创建一个服务器程序，使用多线程处理多个客户端的连接请求和数据交互。

12.3 通过DOS命令编译并运行.java文件

12.3.6 输入与交互

命令行交互程序示例

导入Scanner类

创建Scanner对象

使用Scanner对象读取输入

Scanner类提供了多种方法用于读取不同类型的输入，例如nextInt(), nextDouble(), nextLine()等。

```
import java.util.Scanner;

public class CommandLineInputExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("请输入一个整数: ");
        int number = scanner.nextInt();
        scanner.nextLine(); // 消耗掉nextInt()后的换行符

        System.out.println("请输入一个字符串: ");
        String line = scanner.nextLine();

        System.out.println("你输入的整数是: " + number);
        System.out.println("你输入的字符串是: " + line);

        scanner.close();
    }
}
```

注意：如果在读取int、double等之后紧接着读取String，nextLine()可能会跳过一次输入，因为它会读取之前的nextInt()或nextDouble()之后的换行符。一种常见的解决方案是在两个nextLine()之间添加一个额外的nextLine()来消耗掉那个换行符。

关闭Scanner对象

尽管在许多简单或一次性的应用程序中，可能不需要显式关闭Scanner对象，但在需要精确控制资源使用的场景下，应该在使用完毕后关闭它。



本章重点

- DOS窗口的基本操作
- DOS窗口常用命令以及使用场景
- JDK, JRE, JVM的概念以及区别
- 能够编写简单的Hello World程序，并进行编译输出正确结果
- 能够编写简单的命令行交互程序，并进行编译输出正确结果



本章作业

- 常用命令练习
 - 在windows(或linux)环境中，练习使用命令行创建文件夹、创建文件，删除文件夹、删除文件、移动文件夹、移动文件、复制文件夹、复制文件等常用操作；
- 编译练习
 - 将前面练习的任意一段项目代码重新用纯文本编辑器编辑录入保存，
 - 通过命令行定位到该程序位置
 - 在命令行中通过**javac**命令编译改程序
 - 在命令行中通过**java**命令运行编译过的java程序文件



本章作业

- 程序练习
 - 使用Scanner类在命令行中模拟用户录入，要求能够录入例如患者姓名，患者身份证号码，患者手机号码，患者地址等信息；
 - 使用BufferedWriter类在本地文本文件中写入数据，例如写入患者姓名，患者身份证号码，患者手机号码，患者地址等信息；
 - 使用BufferedReader类在控制台中打印输出上一步骤中写入的患者数据；

信创智能医疗系统研发课程体系
河南中医药大学信息技术学院（智能医疗行业学院）



河南中医药大学信息技术学院（智能医疗行业学院）智能医疗教研室

河南中医药大学医疗健康信息工程技术研究所