



山东大学

信息科学与工程学院

2025—2026 学年第一学期

实验报告

课程名称: Java 编程技术

实验名称: 一个简单的控制台应用程序

专业班级 通信一班

学生学号 202300120317

学生姓名 陈都阳

实验时间 2025年9月16日

【实验目的】

1. 掌握安装 SDK 软件包、Eclipse 软件、EditPlus 编辑软件的方法。
2. 掌握设置程序运行环境的方法。
3. 掌握编写与运行程序的方法。
4. 理解面向对象的编程思想。

【实验要求】

1. 编写一个简单的控制台应用程序，该程序在命令行窗口输出两行文字：“Hello World!” 和 “We are students.”。
2. 使用 Eclipse 编译运行，并截图实验结果。
3. 使用命令行方式编译运行，并截图结果。
4. 实验后回答相关思考问题。

【实验具体内容】

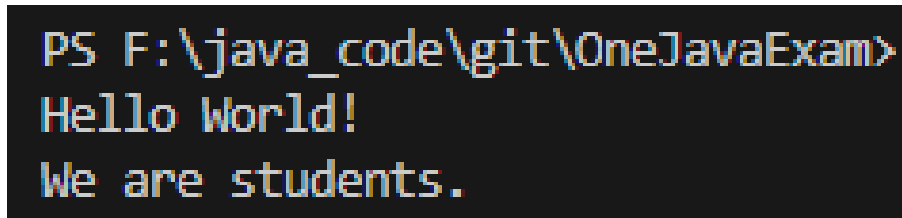
实验源代码

```
1 // HelloWorld.java
2 public class HelloWorld {
3     public static void main(String[] args) {
4         System.out.println("Hello World!");
5         System.out.println("We are students.");
6     }
7 }
8 class Test {
9     public static void main(String[] args) {
10         System.out.println("This is a test class.");
11     }
12 }
```

图 1: HelloWorld 源代码

实验过程与结果

1. 在合适IDE里面编译代码，在vscode的IDE中运行HelloWorld.java，编译运行成功。



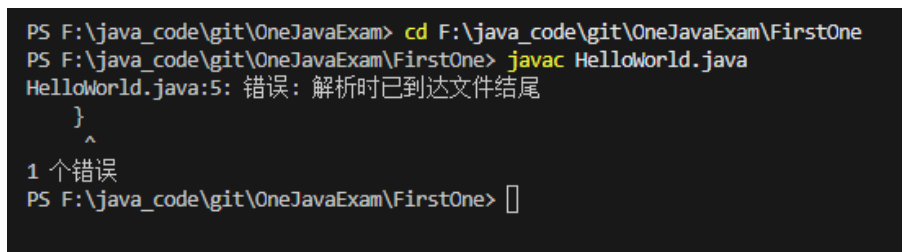
```
PS F:\java_code\git\OneJavaExam>
Hello World!
We are students.
```

图 2: 运行结果

2. 思考问题

- (a) 编译器怎样提示丢失大括号的错误？

编译器会提示解析已经到达文件结尾的错误。

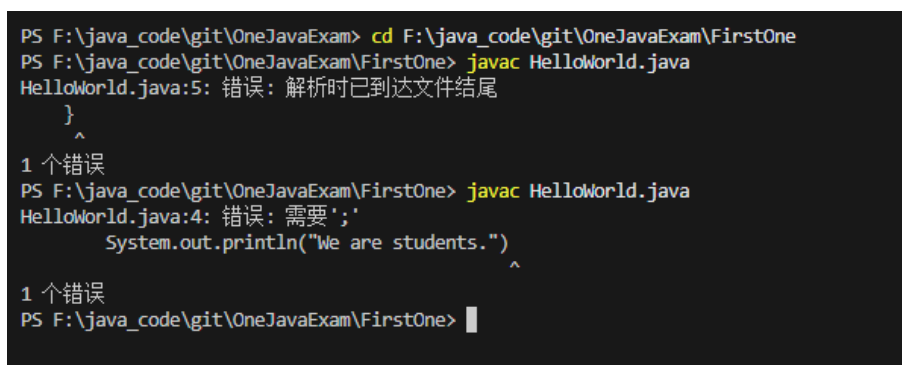


```
PS F:\java_code\git\OneJavaExam> cd F:\java_code\git\OneJavaExam\FirstOne
PS F:\java_code\git\OneJavaExam\FirstOne> javac HelloWorld.java
HelloWorld.java:5: 错误: 解析时已到达文件结尾
    }
    ^
1 个错误
PS F:\java_code\git\OneJavaExam\FirstOne> 
```

图 3: 缺少大括号编译器提示

- (b) 编译器怎样提示语句丢失分号的错误？

编译器会提示需要；



```
PS F:\java_code\git\OneJavaExam> cd F:\java_code\git\OneJavaExam\FirstOne
PS F:\java_code\git\OneJavaExam\FirstOne> javac HelloWorld.java
HelloWorld.java:5: 错误: 解析时已到达文件结尾
    }
    ^
1 个错误
PS F:\java_code\git\OneJavaExam\FirstOne> javac HelloWorld.java
HelloWorld.java:4: 错误: 需要';'
    System.out.println("We are students.")
                        ^
1 个错误
PS F:\java_code\git\OneJavaExam\FirstOne> 
```

图 4: 缺少分号编译器提示

- (c) 编译器怎样提示将System写成system这一错误？

编译器会提示程序包system不存在。

```

PS F:\java_code\git\OneJavaExam> cd F:\java_code\git\OneJavaExam\FirstOne
PS F:\java_code\git\OneJavaExam\FirstOne> javac HelloWorld.java
HelloWorld.java:5: 错误: 解析时已到达文件结尾
    }
    ^
1 个错误
PS F:\java_code\git\OneJavaExam\FirstOne> javac HelloWorld.java
HelloWorld.java:4: 错误: 需要';'
    System.out.println("We are students.")
    ^
1 个错误
PS F:\java_code\git\OneJavaExam\FirstOne> javac HelloWorld.java
HelloWorld.java:4: 错误: 程序包system不存在
    system.out.println("We are students.");
    ^
1 个错误
PS F:\java_code\git\OneJavaExam\FirstOne>

```

图 5: System写成system编译器提示

(d) 其他问题

i. 面向对象与面向过程的区别？

面向对象强调对象之间的关系与交互，面向过程强调操作流程。前者针对问题建模拔出属性与行为，后者针对问题分步骤实现。前者把数据与操作塞到一块，而后者分开。

面向对象的编程更贴向与抽象，可以更好的处理一类问题。面向过程的编程更贴向与具体，可以更好的处理单一具体问题。

个人觉得面向对象写出成品更快，客户更容易接受，比如C#和C，c#写的很快，可以扒拉一些现有的库，C写的很慢，但是可以更底层控制各种数据单元。

ii. 举例说明面向对象与面向过程的区别？

面向对象：把食材塞入自动炒饭机，按下按钮，等待成品。如果需要做其他的菜就另搓机器完成，或者改装机器，至于机器咋做，就不用管了。但不能自己更好的控制每一个步骤。

面向过程：洗菜、切菜、炒菜、盛盘，每一步都要自己动手。每一道菜都得自己做，至于怎么做，就得自己研究了。

iii. 简单说说面向对象的基本概念和面向对象程序设计的特点和优点

表 1: 面向过程与面向对象的优缺点对比

对比维度	面向过程	面向对象
优点	1. 思路清晰，流程直观，容易上手 2. 执行效率高，适合底层开发 3. 程序规模小，开发成本低	1. 强调封装、继承、多态，代码复用性强 2. 模块化程度高，便于维护和扩展 3. 适合大型复杂系统，利于团队协作
缺点	1. 数据与操作分离，代码冗余度高 2. 模块间耦合度高，维护和扩展困难 3. 不适合大型复杂系统	1. 思维方式较复杂，上手难度高 2. 对性能有一定损耗 3. 小型程序中可能显得过于复杂

iv. 搓一个app

之前利用HbuilderX搓了一个wife端口查看的app，主要功能是链接单片机的wifi端口，查看wifi端口输出信息，查看代码运行的情况，具体app在附件中。

【实验心得】

本次实验，我虽然使用了java配套的ide，但感觉怪怪的，毕竟重点是Java版本，而不是IDE。感觉还是vscode更好一些，毕竟可以配合git使用。这样随便扒拉个地都可以写，而且也不怕出错，可以随时回滚。

而且，java版本之前我用的是24，虽然之前没咋用，只是扒别人项目配的环境，但同时使用双版本的配置环境变量还是比较麻烦的，好在单独把java8抽出来写代码还是比较简单的

在这次实验中，我好的看了看java的垃圾回收的具体实现过程，逐渐拔出了java的内存管理的具体过程，感觉java的内存管理还是比较复杂的，毕竟要考虑到各种情况，比如内存泄漏，内存溢出等等。每次运行完后会到exit（）函数，进行栈的释放（感觉是线程结束就嘎了，新版本的是ZGC去把垃圾回收了），释放内存。

```

754 /**
755  * This method is called by the system to give a Thread
756  * a chance to clean up before it actually exits.
757  */
758 private void exit() {
759     if (threadLocals != null && TerminatingThreadLocal.REGISTER.isPresent()) { threadLocals = threadLocals.threadLocalMap#12
760     }
761     TerminatingThreadLocal.threadTerminated();
762     if (group != null) {
763         group.threadTerminated(this);
764         group = null;
765     }
766     /* Aggressively null out all reference fields: see bug 4006245 */
767     target = null;
768     /* speed the release of some of these resources */
769     threadLocals = null;
770     inheritableThreadLocals = null;
771     inheritedAccessControlContext = null;
772     blockers = null;
773     uncaughtExceptionHandler = null;
774 }

```

图 6: 线程结束的exit()函数释放栈内存

之后还进行了反编译，vscode的相关的插件提供反编译的功能，可以看到java的源码。至于反编译之后出现了一个实例化方法则是自动生成的一个构造函数代码。初步推断是为了确保可以去实例化而自动添加的

```

1 // Source code is decompiled from a .class file using FernFlower decompiler (from IntelliJ)
2 public class HelloWorld {
3     public HelloWorld() {
4     }
5
6     public static void main(String[] var0) {
7         System.out.println("Hello World!");
8         System.out.println("We are students.");
9     }
10 }
11

```

图 7: 反编译后的java源码

这次实验报告尝试使用latex排版系统进行排版，前期花了大量的时间去还原封面了，感觉还原的还不错。如果老师需要world，我后期会进行world的版本的实验报告。

由于在linux等其他系统上时间戳是exe文件需要调整wine的兼容层，我附上git的提交记录，来证明我是分时间完成的。最后版本同步时，魔法有些不灵了

```

1 // Source code is decompiled from a .class file using FernFlower decompiler (from IntelliJ)
2 public class HelloWorld {
3     public HelloWorld() {
4     }
5
6     public static void main(String[] var0) {
7         System.out.println("Hello World!");
8         System.out.println("We are students.");
9     }
10 }
11

```

图 8: 反编译后的java源码