

# 深圳大学实验报告

课程名称：Java 程序设计

实验项目名称：课程实验 1：基础知识、基本类型和类的初级应用

学院：计算机与软件学院

专业：数计

指导教师：潘微科

报告人：詹耿羽 学号：2023193026 班级：数计班

实验时间：2024 年 9 月 6 日（周五）-2024 年 9 月 25 日（周三）

实验报告提交时间：

教务部制

### 实验目的与要求:

**实验目的:** 掌握 Java 程序设计开发环境的搭建, 编写简单 Java Project, 掌握编译、运行等基本步骤和命令; 在掌握 Java 数组基本概念及应用的基础上, 变换数组的内容, 完成主类创建, 查找等功能的实现; 熟练掌握数据类型、运算符、表达式和语句; 初步掌握面向对象编程中类的编写。

### 实验要求:

#### **Part 1 (25 分)**

(1.1). 下载、安装"Java SE Development Kit 22.0.2"最新的版本, 进行系统环境变量的设置(如需要), 之后进行简单的测试以示安装成功。每一步操作请在报告中附上截图, 应至少包含一个全屏截图(其他截图可以不用全屏)和详细的文字说明。(5 分)

(1.2). 下载、安装"Eclipse IDE for Java Developers" (2024-08 版本), 并进行 JRE/JDK 的设置(如需要)。每一步操作请在报告中附上截图, 应至少包含一个全屏截图(其他截图可以不用全屏)和详细的文字说明。(5 分)

(1.3). 将第一章讲义 (JavaPD-Ch01) 中的三个应用程序在 Eclipse 中运行。每一步操作(例如, 新建类、编写代码、运行程序等)请在报告中附上截图, 应至少包含一个全屏截图(其他截图可以不用全屏)和详细的文字说明。(5 分)

(1.4). 浏览 <https://docs.oracle.com/en/java/javase/22/>, 阅读 "Security" 板块的内容, 并用自己的话进行介绍 (500-800 字), 要求重点突出、条理清楚, 可读性强。(10 分)

#### **Part 2 (25 分)**

(2.1) 编写 Java 程序: 创建一个  $1000 \times 1000 \times 100$  三维的 float 数组, 对数组中的元素进行随机赋值(要求使用 Math.random()生成 0-1 之间的数)。通过算法找到该数组中最小的 15 个数, 要求从小到大输出, 同时计算整个程序所耗费的时间, 并分析算法的复杂度。对每一行语句加上注释。要求不能使用 PriorityQueue, 可以使用 Stack 或 Array。时间复杂度  $O(nk)$  即可, 其中  $n$  是  $1000 \times 1000 \times 100$ ,  $k$  是 15。在报告中附上程序截图、运行结果截图和详细的文字说明。(5 分)

(2.2) 编写 Java 程序: 从键盘输入 21 个浮点数, 放入一个一维数组, 然后将前 5 个元素与后 5 个元素对换, 即将第 1 个元素与第 21 个元素互换, 将第 2 个元素与第 20 个元素互换, 依次类推。对每一行语句加上注释。在报告中附上程序截图、运行结果截图和详细的文字说明。(5 分)

(2.3) 编写 Java 程序: 计算 10-10000 之间有多少个素数, 并输出所有素数。在报告中附上程序截图、运行结果截图和详细的文字说明。(5 分)

(2.4) 编写 Java 程序: 随机生成 5 个 21 位数(整数), 并判断它是不是回文。要求对每个生成的随机数输出三个信息: 随机数、逆序数、是否是回文。所谓 "回文" 是指一种从前向后读和从后向前读都一样的数字, 例如, 1234321、322223。对每一行语句加上注释。在报告中附上程序截图、运行结果截图和详细的文字说明。(10 分)

#### **Part 3 (30 分)**

(3.1). 运行第 4 章课件中第 4 页、第 24 页、第 32 页和第 34 页中的四个程序, 并对每一行语句加上注释。对每一行语句加上注释。在报告中附上程序截图、运行结果截图和简要的文字说明。(5 分)

(3.2). 设计并测试一个长方体类 Box。(i) 数据成员包括 length、width 和 height, 分别表示长方体的长、宽和高; (ii) 定义 setInfo(int,int,int)方法设置这 3 个数据成员的值; (iii) 定义 volume()方法求长方体的体积; (iv) 定义 area()方法求长方体的表面积; (v) 定义 toString()方法把长方体的长、宽、高以及长方体的体积和表面积转化为字符串并返回。对每一行语句加上注释。在报告中附上程序截图、运行结果截图和简要的文字说明。

(5 分)

(3.3).参照题(2)设计并测试一个圆锥体 Cone。对每一行语句加上注释。在报告中附上程序截图、运行结果截图和简要的文字说明。(5 分)

(3.4).设计并测试一个研究生类 PostGraduateStudent。(i) 数据成员包括 ID (学号)、name (姓名) 以及 3 门课程 math、programming、english; (ii) 定义 comSum()、comAvg()、comMax() 计算 3 门课程的总分、平均分和最高分; (iii) 在该类中实现对两个学生进行比较的方法 (根据总分)。对每一行语句加上注释。在报告中附上程序截图、运行结果截图和简要的文字说明。(5 分)

(3.5) 编写一个 Teacher 类。类中包含以下成员变量: name (姓名)、title (职位)、course (主讲的课程)、research (研究方向) 和 office (办公室)。定义对应的方法对这几个成员变量的值进行设置和读取。(i) 在 Teacher 类外的 main 方法里面, 创建该类的一个对象, 并调用各个方法, 展示相应的效果。(ii) 在 Teacher 类内的 main 方法里面, 创建该类的一个对象, 并调用各个方法, 展示相应的效果。在报告中附上程序截图、运行结果截图和简要的文字说明。(5 分)

(3.6).当设计一个类的时候, 有哪些注意事项? 请用自己的话进行阐述 (300-500 字), 要求重点突出、条理清楚, 可读性强。(5 分)

报告写作。要求: 主要思路有明确的说明, 重点代码有详细的注释, 行文逻辑清晰可读性强, 报告整体写作较为专业。(20 分)

说明:

(1) 本次实验课作业满分为 100 分, 占总成绩的比例 7%。

(2) 本次实验课作业截至时间 2024 年 9 月 25 日 (周三) 21:59。

(3) 报告正文: 请在**指定位置填写**, 本次实验**不需要单独提交源程序文件**。

(4) 个人信息: WORD 文件名中的“姓名”、“学号”, 请改为你的**姓名和学号**; 实验报告的首页, 请**准确填写“学院”、“专业”、“报告人”、“学号”、“班级”、“实验报告提交时间”**等信息。

(5) 提交方式: 截至时间前, 请在 Blackboard 平台中提交。

(6) 发现抄袭 (包括复制&粘贴整句话、整张图), **抄袭者和被抄袭者的成绩记零分**。

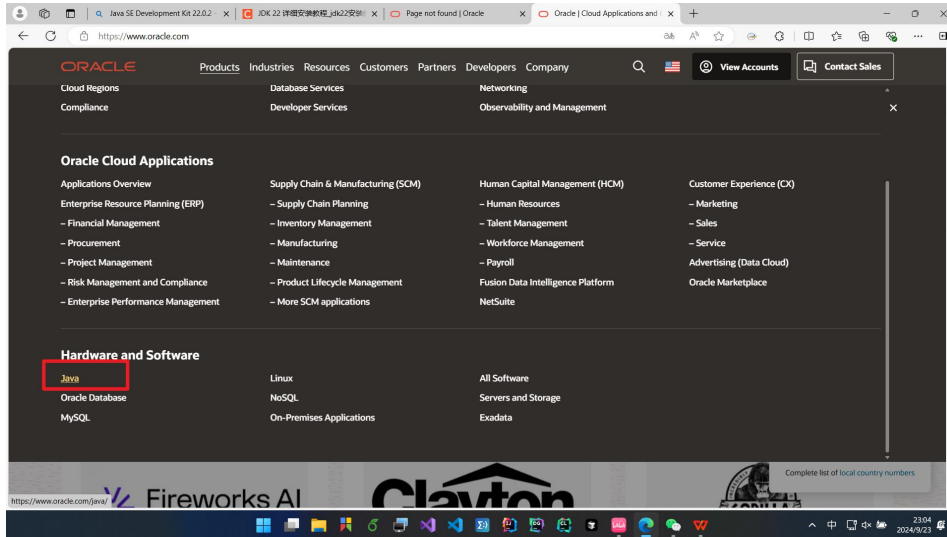
(7) 延迟提交, 不得分; 如有特殊情况, 请于截至日期之后的 **48 小时内**发邮件到 panweike@szu.edu.cn, 并在邮件中注明课程名称、作业名称、姓名、学号等信息, 以及特殊情况的说明, 我收到后会及时回复。

(8) 期末考试阶段补交无效。

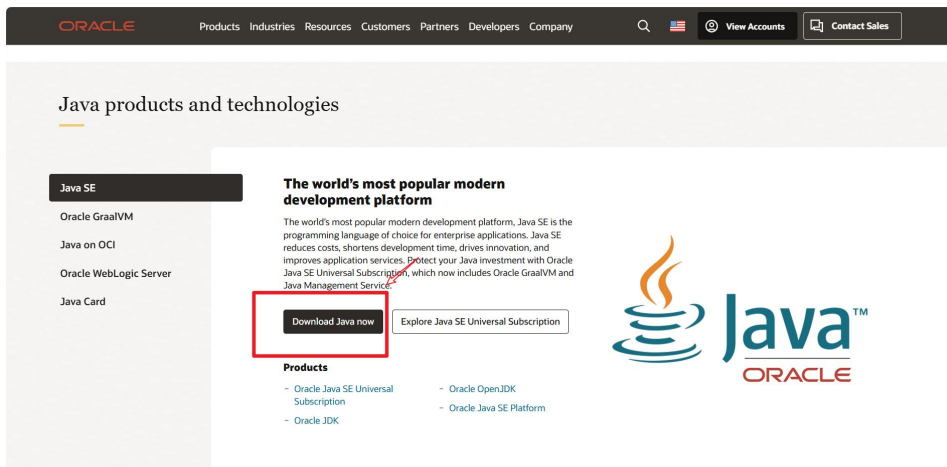
## Part 1 (25 分)

(1.1).下载、安装"Java SE Development Kit 22.0.2"最新的版本，进行系统环境变量的设置（如需要），之后进行简单的测试以示安装成功。每一步操作请在报告中附上截图，应至少包含一个全屏截图（其他截图可以不用全屏）和详细的文字说明。（5 分）

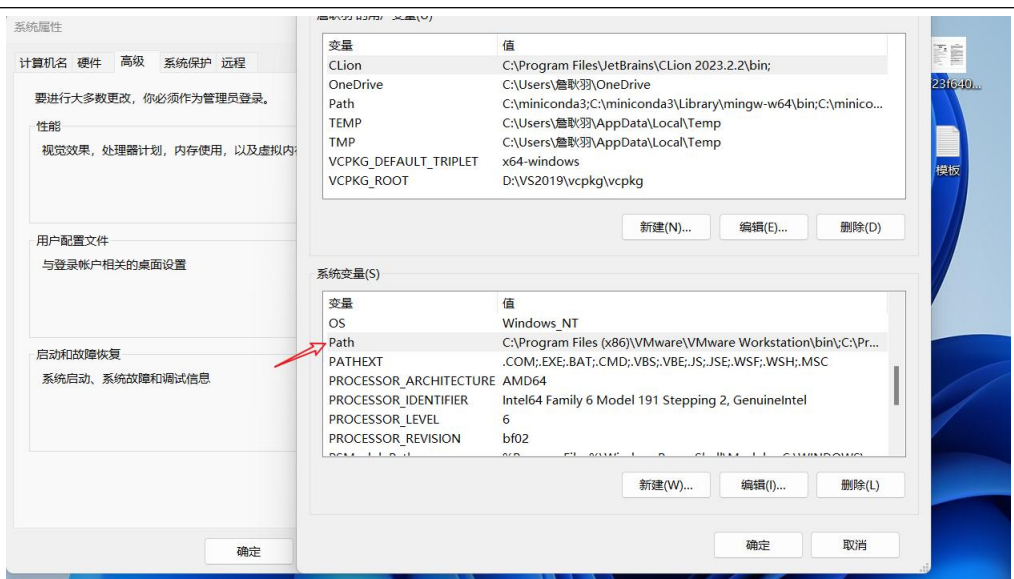
- 在网上搜索 Oracle 官网，在产品部分选择 Java。



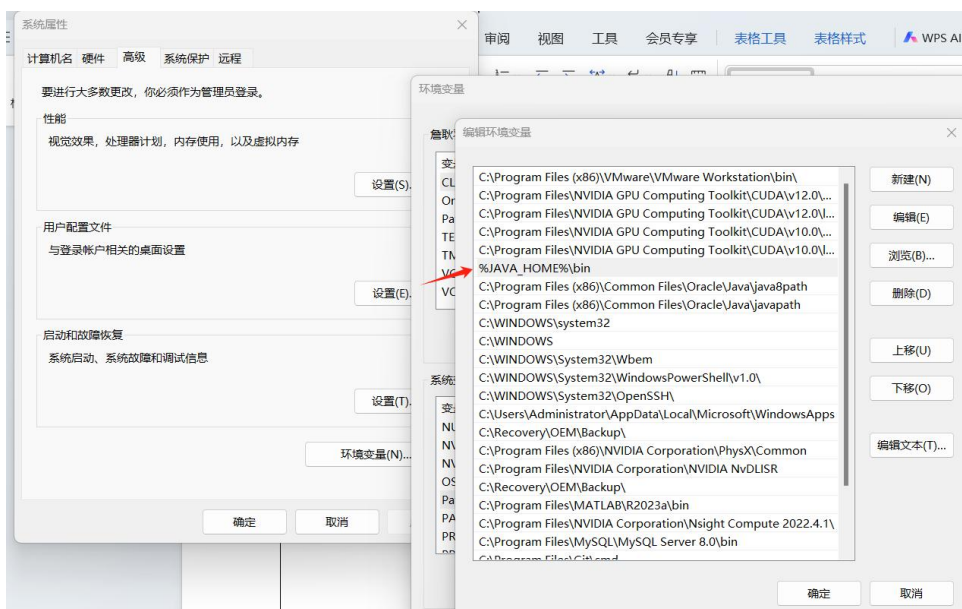
- 点击 download 选项。



- 打开电脑环境变量。



- 配置环境。



(1.2).下载、安装"Eclipse IDE for Java Developers"（2024-08 版本），并进行 JRE/JDK 的设置（如需要）。每一步操作请在报告中附上截图，应至少包含一个全屏截图（其他截图可以不用全屏）和详细的文字说明。（5 分）

- 在官网搜索 Eclipse IDE for Java Developers。

国内版国际版

Microsoft Bing

Eclipse IDE for Java Developers

网页 图片 视频 学术 词典 地图 更多 工具

检测到您输入了英文，试试切换到国际版？搜英文结果更丰富更准确

约 727,000 个结果

The Eclipse Foundation

<https://www.eclipse.org/downloads/packages/release/...>

Eclipse IDE for Java Developers | Eclipse Packages - The Eclipse

网页 2022年6月15日 · Eclipse IDE for Java Developers. Package Description. The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Maven and Gradle ...

Download

The Eclipse Temurin™ project provides high-quality, TCK certified OpenJDK ...

Packages

Eclipse IDE for Java and DSL Developers. 502 MB ; 921 DOWNLOADS; The ...

仅显示来自 eclipse.org 的搜索结果

CSDN博客

[https://blog.csdn.net/m0\\_62627216/article/details/135540237](https://blog.csdn.net/m0_62627216/article/details/135540237)

JAVA安装下载、Eclipse下载安装及配置JAVA项目（超详细）

本文介绍了如何下载并配置Eclipse IDE，包括单版本JDK的安装，多版本共存设置，以及如何创建JavaSE项目并进行测试。还提供了Eclipse的优化配置建议，如 ...

• 点击 download。

ECLIPSE FOUNDATION

Projects Supporters Collaborations Resources The Foundation

Home » Downloads » Eclipse downloads - Select a mirror

All downloads are provided under the terms and conditions of the Eclipse Foundation Software User Agreement unless otherwise specified.

Download

Download from: United States - Oms.dev Team (https)

File: [eclipse-inst-jre-win64.exe](#) | SHA-512

>> Select Another Mirror

Sponsored Ad

list

an open source strategy

Advertise Here

Other options for this file

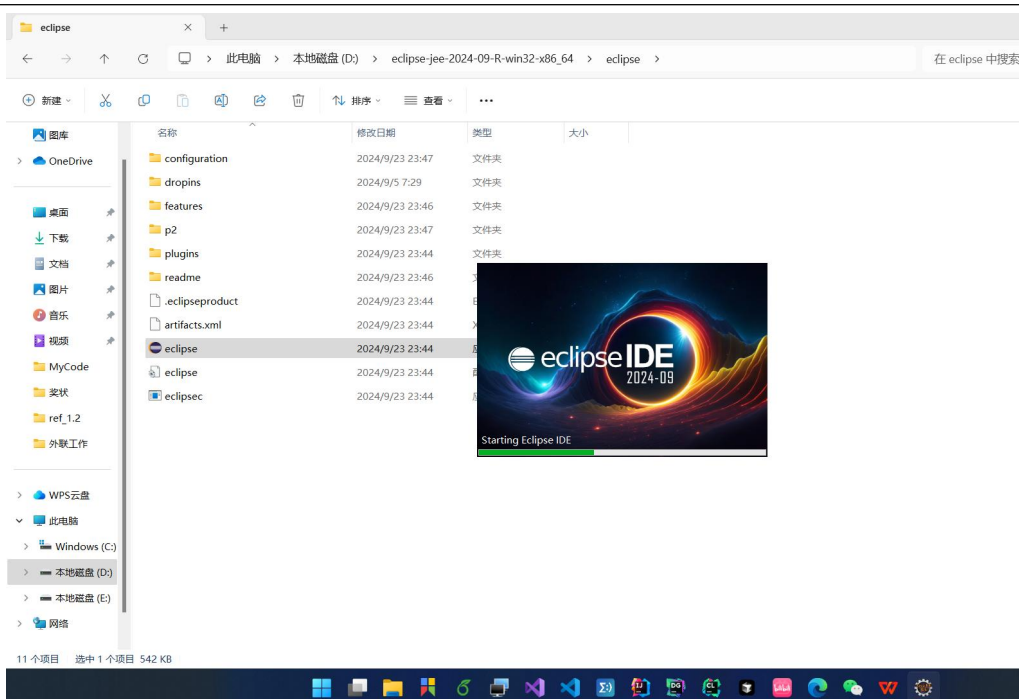
• 等待下载。

eclipse-jee-2024-09-R-win32-x86\_64

3.0 MB/s - 32.7 MB/530 MB, 剩余 3 分钟

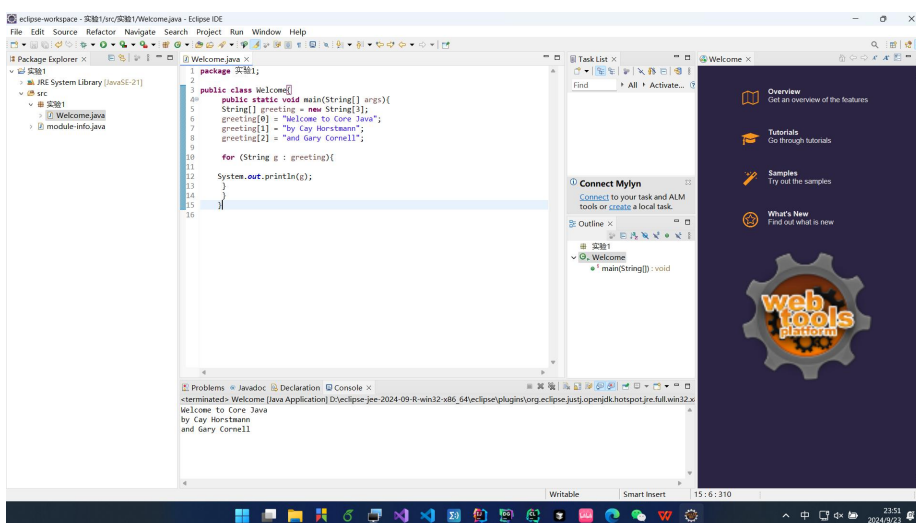
• 下载完成。





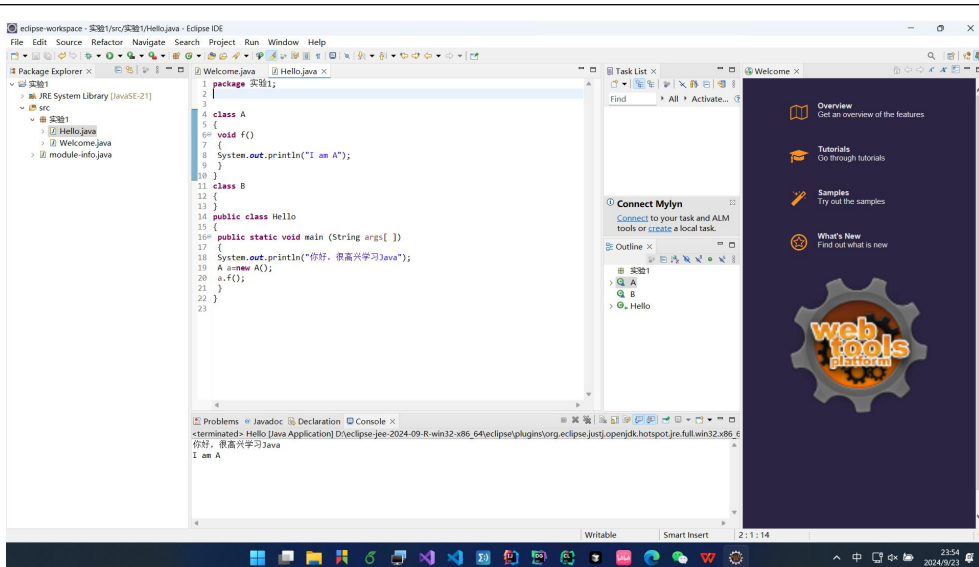
(1.3).将第一章讲义（JavaPD-Ch01）中的三个应用程序在 Eclipse 中运行。每一步操作（例如，新建类、编写代码、运行程序等）请在报告中附上截图，应至少包含一个全屏截图（其他截图可以不用全屏）和详细的文字说明。（5 分）

程序 1:



这段 Java 代码定义了一个名为 `Welcome` 的公共类，其中包含一个 `main` 方法。`main` 方法是程序的入口点。代码创建了一个字符串数组 `greeting`，大小为 3，存储了三条欢迎信息。随后，使用增强的 `for` 循环遍历数组中的每个字符串，并将其打印到控制台。最终的输出是三行欢迎信息。

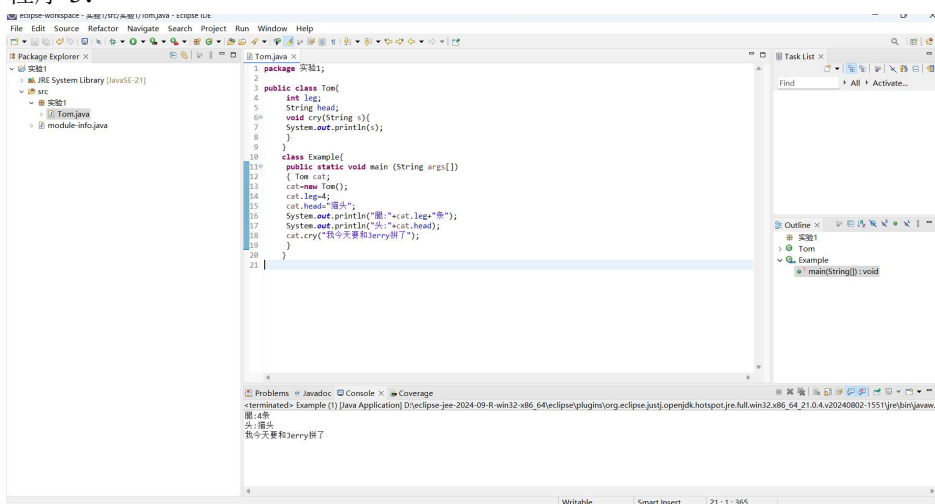
程序 2:



- A 类
- 方法：
  - `void f()`：一个无返回值的方法，打印出“I am A”。
- B 类
- 这个类是空的，没有任何字段或方法。
- Hello 类
- 主方法：
  - `System.out.println("你好，很高兴学习 Java");`：输出一条欢迎信息。
  - `A a = new A();`：创建一个`A`类的实例，并将其赋值给变量`a`。
  - `a.f();`：调用`a`的`f`方法，这将打印“I am A”。

这段代码的功能是打印欢迎信息和类`A`的方法输出，展示了如何创建类的实例并调用其方法。最终输出会是两行文字。

### 程序3：



- Tom 类
- 字段：
  - `int leg;`：表示腿的数量。
  - `String head;`：表示头的描述（例如“猫头”）。
- 方法：



```
- `void cry(String s)`：接受一个字符串参数`s`并将其打印到控制台。
• Example 类
- 主方法：
  - `Tom cat`：声明一个`Tom`类型的变量`cat`。
  - `cat = new Tom()`：创建一个`Tom`类的实例并赋值给`cat`。
  - `cat.leg = 4`：设置`cat`的`leg`字段为 4。
  - `cat.head = "猫头"`：设置`cat`的`head`字段为“猫头”。
- 输出：
  - `System.out.println("腿:" + cat.leg + "条");`：打印出`cat`的腿的数量。
  - `System.out.println("头:" + cat.head);`：打印出`cat`的头的描述。
  - `cat.cry("我今天要和 Jerry 拼了");`：调用`cry`方法，打印出指定的字符串。
```

这段代码的功能是创建一个`Tom`对象（代表一只猫），设置它的腿和头的属性，并打印出相关信息和一条信息。最终的输出会展示猫的腿的数量、头的描述，以及它的叫声。

(1.4).浏览 <https://docs.oracle.com/en/java/javase/22/>，阅读“Security”板块的内容，并用自己的话进行介绍（500-800 字），要求重点突出、条理清楚，可读性强。（10 分）

Java SE 22 的“安全性”模块提供了多个关键功能，旨在确保应用程序的安全性和数据保护。该安全架构涉及密码学、身份验证、安全通信等多个领域，并提供了强大的 API 和工具来帮助开发者实现安全功能。

#### 1. 密码学和密钥管理

Java 提供了多种内置的加密算法和密钥管理功能。Java 的 `java.security` 和 `javax.crypto` 包支持常见的加密算法，如 AES、RSA、ChaCha20 等。通过这些 API，开发者可以执行对称加密、非对称加密和哈希运算。Java 还支持密钥存储，通过 `KeyStore` 和 `CertStore` 类来安全地保存密钥和证书。

Java 提供对 PKCS#11 和 PKCS#12 等行业标准的支持，确保开发者能够轻松与硬件安全模块（如智能卡）进行集成。Java 的 SunPKCS11 提供程序能够与 PKCS#11 设备进行无缝通信，允许在 Java 程序中使用硬件存储的密钥进行加密操作。

#### 2. 公钥基础设施 (PKI)

Java 的公钥基础设施 (PKI) 支持数字证书的管理和验证。通过 `java.security.cert` 包中的 API，开发者可以管理 X.509 证书、证书吊销列表 (CRL)，并使用 PKIX 兼容的证书路径验证机制。此类功能广泛应用于确保通信双方的身份验证和数据完整性【6†source】。

#### 3. 安全通信

在网络通信中，Java 支持多种安全通信协议，如 TLS（传输层安全协议）和 DTLS（数据报传输层安全协议），通过加密保障数据的机密性、完整性和通信双方的身份验证。Java 提供了 `javax.net.ssl` 包来实现 SSL/TLS 协议的支持，开发者可以使用这些 API 实现安全的客户端-服务器通信，防止数据在传输过程中被窃取或篡改【6†source】【8†source】。

#### 4. 身份验证

身份验证是安全系统的重要组成部分，Java 提供了可插拔的身份验证模块（Pluggable Authentication Modules, PAM），通过 `LoginContext` 类，开发者可以指定不同的登录模块来验证用户身份。Java 内置了对 Kerberos 协议和 LDAP 的支持，帮助开发者在不同

的环境中实现灵活的身份验证机制【6†source】。

## 5. 安全编码指南

Java 的安全模块还包括一系列安全编码指南，帮助开发者避免常见的安全漏洞。这些指南涵盖了减少权限检查、正确管理资源、使用可靠的第三方库等多个方面。通过遵循这些最佳实践，开发者可以有效地减少代码中的安全风险。例如，使用 `try-with-resources` 模式来确保资源的正确释放，避免因资源泄漏导致的系统崩溃。

## 6. 总结

Java SE 22 的安全架构提供了全面的解决方案来应对密码学、身份验证和安全通信等方面的挑战。通过提供多种标准算法、密钥管理和证书管理 API，Java 确保了应用程序能够安全地处理敏感信息，并防止潜在的安全威胁。配合强大的安全编码指南，开发者可以构建出更加健壮和安全的 Java 应用。

## Part 2 (25 分)

(2.1) 编写 Java 程序：创建一个  $1000 \times 1000 \times 100$  三维的 float 数组，对数组中的元素进行随机赋值（要求使用 `Math.random()` 生成 0-1 之间的数）。通过算法找到该数组中最小的 15 个数，要求从小到大输出，同时计算整个程序所耗费的时间，并分析算法的复杂度。对每一行语句加上注释。要求不能使用 `PriorityQueue`，可以使用 `Stack` 或 `Array`。时间复杂度  $O(nk)$  即可，其中  $n$  是  $1000 \times 1000 \times 100$ ， $k$  是 15。在报告中附上程序截图、运行结果截图和详细的文字说明。（5 分）

✓ 代码呈现：

```
1.  public class FindMinValues {
2.      public static void main(String[] args) {
3.          int dim1 = 1000;
4.          int dim2 = 1000;
5.          int dim3 = 100;
6.
7.          // 创建并填充三维数组
8.          float[][][] array = new float[dim1][dim2][dim3];
9.          for (int i = 0; i < dim1; i++) {
10.             for (int j = 0; j < dim2; j++) {
11.                 for (int k = 0; k < dim3; k++) {
12.                     array[i][j][k] = (float) Math.random(); //
生成 0 到 1 之间的随机数
13.                 }
14.             }
15.         }
16.
17.         // 记录开始时间
18.         long startTime = System.currentTimeMillis();
19.
```

```
20.          // 将三维数组中的元素提取到一维数组中
21.          float[] allElements = new float[dim1 * dim2 * dim3];
22.          int index = 0;
23.          for (int i = 0; i < dim1; i++) {
24.              for (int j = 0; j < dim2; j++) {
25.                  for (int k = 0; k < dim3; k++) {
26.                      allElements[index++] = array[i][j][k];
27.                  }
28.              }
29.          }
30.
31.          // 找到最小的15个数
32.          findMinValues(allElements, 15);
33.
34.          // 记录结束时间
35.          long endTime = System.currentTimeMillis();
36.          long elapsedTime = endTime - startTime;
37.
38.          // 输出程序执行时间
39.          System.out.println("程序执行时间: " + elapsedTime + " 毫
秒");
40.      }
41.
42.      public static void findMinValues(float[] array, int k) {
43.          if (k <= 0) {
44.              System.out.println("无效的 k 值");
45.              return;
46.          }
47.
48.          // 使用快速排序对数组进行排序
49.          quickSort(array, 0, array.length - 1);
50.
51.          // 输出最小的 k 个数
52.          System.out.println("最小的 " + k + " 个数是: ");
53.          for (int i = 0; i < k && i < array.length; i++) {
54.              System.out.println(array[i]);
55.          }
56.      }
57.
58.      // 快速排序的实现
59.      private static void quickSort(float[] array, int low, int h
igh) {
60.          if (low < high) {
61.              int pi = partition(array, low, high);
```

```

62.         quickSort(array, low, pi - 1);
63.         quickSort(array, pi + 1, high);
64.     }
65. }
66.
67.     private static int partition(float[] array, int low, int high) {
68.         float pivot = array[high];
69.         int i = low - 1;
70.         for (int j = low; j < high; j++) {
71.             if (array[j] < pivot) {
72.                 i++;
73.                 swap(array, i, j);
74.             }
75.         }
76.         swap(array, i + 1, high);
77.         return i + 1;
78.     }
79.
80.     private static void swap(float[] array, int i, int j) {
81.         float temp = array[i];
82.         array[i] = array[j];
83.         array[j] = temp;
84.     }
85. }

```

✓ 程序截图：

The screenshot shows an IDE window titled 'FindMinValues.java'. The code is as follows:

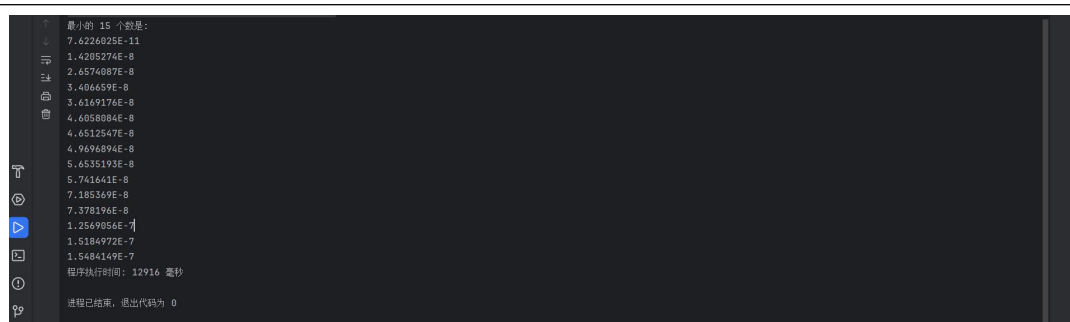
```

1 public class FindMinValues {
2     public static void FindMinValues(float[] array, int k) { 1个用法
3         if (k <= 0) {
4             System.out.println("无效的k值");
5             return;
6         }
7
8         // 使用快速排序对数组进行排序
9         quickSort(array, 0, array.length - 1);
10
11         // 输出最小的k个数
12         System.out.println("最小的 " + k + " 个数是: ");
13         for (int i = 0; i < k && i < array.length; i++) {
14             System.out.println(array[i]);
15         }
16
17         // 快速排序的实现
18         private static void quickSort(float[] array, int low, int high) { 3个用法
19             if (low < high) {
20                 int pi = partition(array, low, high);
21                 quickSort(array, low, pi - 1);
22                 quickSort(array, pi + 1, high);
23             }
24         }
25     }
26 }

```

A red annotation '篇幅有限, 详细代码已在前面呈现' (Limited space, detailed code is presented in the front) points to the call to `quickSort(array, 0, array.length - 1);` on line 9.

✓ 运行结果：



```
最小的 15 个数是:
7.6226025E-11
1.4205274E-8
2.6574087E-8
3.406659E-8
3.6189174E-8
4.6058004E-8
4.6512547E-8
4.9896894E-8
5.6535193E-8
5.741641E-8
7.185369E-8
7.378196E-8
1.2569054E-7
1.5184972E-7
1.5484149E-7
程序执行时间: 12916 毫秒
进程已结束, 退出代码为 0
```

✓ 详细的文字说明:

#### 1. 数组初始化:

- 1) `'dim1'`, `'dim2'`, 和 `'dim3'` 定义了三维数组的维度。此处三维数组的尺寸为 `'1000 x 1000 x 100'`。
- 2) `'array'` 是一个 `'float'` 类型的三维数组, 初始化为指定的维度。
- 3) 使用三重循环填充 `'array'` 中的每个元素, 赋值为 `'0'` 到 `'1'` 之间的随机浮点数。

#### 2. 提取元素到一维数组:

- 1) 创建一个一维数组 `'allElements'`, 其大小是三维数组所有元素的总数。
- 2) 通过三重循环将三维数组的所有元素依次存储到 `'allElements'` 中。

#### 3. 寻找最小的 15 个数:

- 1) 记录程序开始时间 `'startTime'` 和结束时间 `'endTime'`, 计算程序执行时间 `'elapsedTime'`。
- 2) 调用 `'findMinValues'` 方法找到并输出最小的 15 个数。

#### 4. `'findMinValues'` 方法:

- 1) 这个方法首先检查 `'k'` 是否有效 (大于 0)。如果无效, 输出错误信息。
- 2) 使用 `'quickSort'` 对 `'allElements'` 数组进行排序。`'quickSort'` 方法是一个经典的排序算法, 采用分治策略。
- 3) 排序完成后, 输出数组中前 `'k'` 个元素, 即最小的 15 个数。

#### 5. `'quickSort'` 实现:

- 1) `'quickSort'` 是递归的排序方法, 通过 `'partition'` 方法将数组分为两部分, 并对这两部分进行排序。
- 2) `'partition'` 方法选择数组的最后一个元素作为基准 (pivot), 并重新排列数组, 使得基准左边的元素都小于基准, 右边的元素都大于基准。
- 3) `'swap'` 方法用于交换数组中的两个元素的位置。

(2.2) 编写 Java 程序: 从键盘输入 21 个浮点数, 放入一个一维数组, 然后将前 5 个元素与后 5 个元素对换, 即将第 1 个元素与第 21 个元素互换, 将第 2 个元素与第 20 个元素互换, 依次类推。对每一行语句加上注释。在报告中附上程序截图、运行结果截图和详细的文字说明。(5 分)

✓ 代码:

```
1. import java.util.Scanner;
2.
3. public class SwapArrayElements {
```

```
4.     public static void main(String[] args) {
5.         // 创建一个 Scanner 对象用于从键盘读取输入
6.         Scanner scanner = new Scanner(System.in);
7.
8.         // 创建一个长度为21 的浮点型数组
9.         float[] numbers = new float[21];
10.
11.        // 从键盘读取 21 个浮点数，并存储到数组中
12.        System.out.println("请输入 21 个浮点数: ");
13.        for (int i = 0; i < 21; i++) {
14.            numbers[i] = scanner.nextFloat(); // 读取浮点数并存储到数组中
15.        }
16.
17.        // 输出原始数组
18.        System.out.println("原始数组: ");
19.        printArray(numbers);
20.
21.        // 交换前5 个元素与后5 个元素
22.        for (int i = 0; i < 5; i++) {
23.            // 计算对应的对换位置
24.            int frontIndex = i; // 前5 个元素的索引
25.            int backIndex = 20 - i; // 后5 个元素的索引
26.
27.            // 交换前5 个元素与后5 个元素
28.            float temp = numbers[frontIndex]; // 临时变量存储前5 个元素的值
29.            numbers[frontIndex] = numbers[backIndex]; // 后5 个元素的值赋给前5 个元素
30.            numbers[backIndex] = temp; // 临时变量的值赋给后5 个元素
31.        }
32.
33.        // 输出交换后的数组
34.        System.out.println("交换后的数组: ");
35.        printArray(numbers);
36.
37.        // 关闭 Scanner 对象
38.        scanner.close();
39.    }
40.
41.    // 辅助方法：打印数组
42.    public static void printArray(float[] array) {
43.        for (float num : array) {
```



```

44.         System.out.print(num + " "); // 打印数组中的每个元素
45.     }
46.         System.out.println(); // 换行
47.     }
48. }

```

✓ 运行截图：

```

1  import java.util.Scanner;
2
3  public class SwapArrayElements {
4      public static void main(String[] args) {
5          // 创建一个 Scanner 对象用于从键盘读取输入
6          Scanner scanner = new Scanner(System.in);
7
8          // 创建一个长度为21的浮点型数组
9          float[] numbers = new float[21];
10
11         // 从键盘读取21个浮点数，并存储到数组中
12         System.out.println("请输入21个浮点数。");
13         for (int i = 0; i < 21; i++) {
14             numbers[i] = scanner.nextFloat(); // 读取浮点数并存储到数组中
15         }
16
17         // 输出原始数组
18         System.out.println("原始数组。");
19         printArray(numbers);
20
21         // 交换前5个元素与后5个元素
22         for (int i = 0; i < 5; i++) {
23             // 计算对应的对称位置
24             int frontIndex = i; // 前5个元素的索引
25             int backIndex = 20 - i; // 后5个元素的索引

```

✓ 结果：

```

请输入21个浮点数:
0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0 2.1
原始数组:
0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0 2.1
交换后的数组:
2.1 2.0 1.9 1.8 1.7 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.3 1.4 1.5 1.6 0.5 0.4 0.3 0.2 0.1
进程已结束，退出代码为 0

```

✓ 文字说明：

1.导入 Scanner 类：

import java.util.Scanner; 用于导入 Java 的 Scanner 类，用于从控制台读取用户输入。

2.创建 Scanner 对象：

Scanner scanner = new Scanner(System.in); 创建一个 Scanner 对象以从标准输入读取数据。

3.定义和初始化数组：

float[] numbers = new float[21]; 创建一个长度为 21 的浮点型数组。

4.读取用户输入：

- System.out.println("请输入 21 个浮点数。"); 提示用户输入浮点数。

使用 for 循环从键盘读取 21 个浮点数，并将其存储到 numbers 数组中。

5.输出原始数组：

printArray(numbers); 调用 printArray 方法输出数组的当前状态。

6.交换前 5 个和后 5 个元素：

- for (int i = 0; i < 5; i++) 循环处理前 5 个元素。
- int frontIndex = i; 和 int backIndex = 20 - i; 确定要交换的元素索引。
- 使用临时变量 temp 交换 frontIndex 和 backIndex 位置的元素。

7.输出交换后的数组:

再次调用 printArray(numbers); 输出经过交换后的数组状态。

8.关闭 Scanner 对象:

scanner.close(); 关闭 Scanner 对象以释放资源。

9.辅助方法 printArray:

printArray 方法用于打印数组中的所有元素。

(2.3) 编写 Java 程序: 计算 10-10000 之间有多少个素数, 并输出所有素数。在报告中附上程序截图、运行结果截图和详细的文字说明。(5 分)

✓ 代码

```

1.  public class PrimeNumbers {
2.      public static void main(String[] args) {
3.          // 设置范围的起始值和结束值
4.          int start = 10;
5.          int end = 10000;
6.
7.          // 计算范围内的素数并输出
8.          System.out.println("10 到 10000 之间的素数有: ");
9.          int count = 0;
10.         for (int num = start; num <= end; num++) {
11.             if (isPrime(num)) {
12.                 System.out.print(num + " ");
13.                 count++;
14.             }
15.         }
16.
17.         // 输出素数的总数
18.         System.out.println("\n 总共找到 " + count + " 个素数。");
19.     }
20.
21.     // 判断一个数是否为素数的辅助方法
22.     public static boolean isPrime(int number) {
23.         if (number <= 1) {
24.             return false;
25.         }
26.         if (number == 2) {
27.             return true; // 2 是唯一的偶数素数
28.         }
29.         if (number % 2 == 0) {
30.             return false; // 排除其他偶数
31.         }
32.         // 只检查到平方根即可

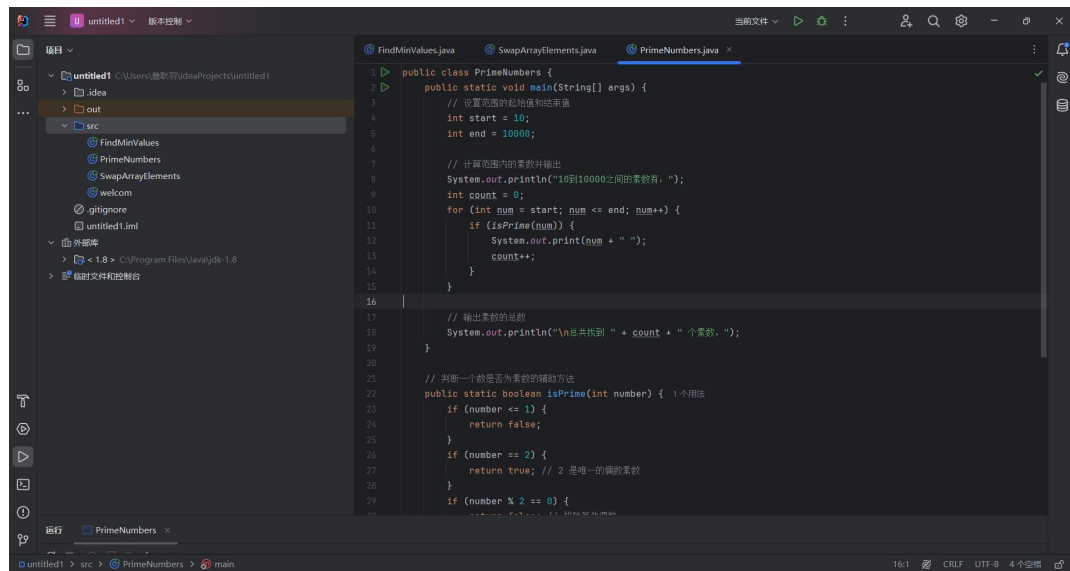
```

```

33.         for (int i = 3; i <= Math.sqrt(number); i += 2) {
34.             if (number % i == 0) {
35.                 return false;
36.             }
37.         }
38.         return true;
39.     }
40. }

```

#### ✓ 程序截图

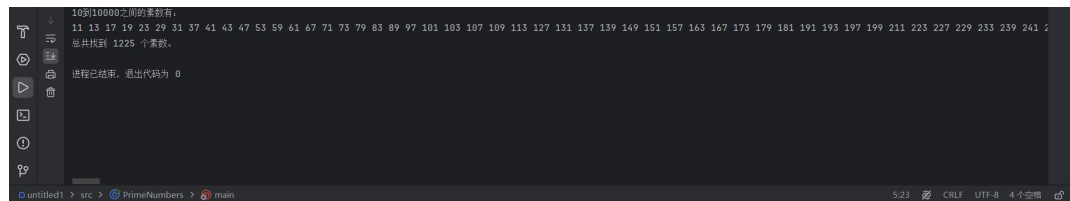


```

1 public class PrimeNumbers {
2     public static void main(String[] args) {
3         // 设置范围的起始值和结束值
4         int start = 10;
5         int end = 10000;
6
7         // 计算范围内的素数并输出
8         System.out.println("10到10000之间的素数有:");
9         int count = 0;
10        for (int num = start; num <= end; num++) {
11            if (isPrime(num)) {
12                System.out.print(num + " ");
13                count++;
14            }
15        }
16
17        // 输出素数的数量
18        System.out.println("\n总共找到 " + count + " 个素数。");
19    }
20
21    // 判断一个数是否为素数的辅助方法
22    public static boolean isPrime(int number) { 1个用法
23        if (number <= 1) {
24            return false;
25        }
26        if (number == 2) {
27            return true; // 2 是唯一的偶数素数
28        }
29        if (number % 2 == 0) {
30            return false;
31        }
32        for (int i = 3; i <= Math.sqrt(number); i += 2) {
33            if (number % i == 0) {
34                return false;
35            }
36        }
37        return true;
38    }
39 }

```

#### ✓ 结果



```

10到10000之间的素数有:
11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193 197 199 211 223 227 229 233 239 241 251 257 263 269 271 277 281 283 293 307 311 313 317 331 337 347 349 353 359 367 373 379 383 389 397 401 409 419 421 431 433 439 443 449 457 461 463 467 473 479 487 491 499 503 509 517 521 523 527 533 539 547 557 563 569 571 577 587 593 599 601 607 613 617 619 623 629 631 637 641 643 647 653 659 661 667 671 673 677 683 687 691 697 701 703 709 713 719 727 733 737 743 749 757 761 763 769 773 779 787 793 797 803 809 811 817 821 823 827 829 833 837 839 843 847 853 857 859 863 869 877 881 883 887 893 897 907 911 913 917 919 923 929 931 937 941 943 947 953 959 967 971 973 977 983 989 993 997
总共找到 1228 个素数。

进程已结束，退出代码为 0

```

#### ✓ 文字说明

##### 1.定义范围:

int start = 10; 和 int end = 10000; 设置素数搜索的范围。

##### 2.计算素数:

- 使用 for 循环遍历范围内的所有整数。
- 调用 isPrime(num) 方法判断每个整数是否为素数。

##### 3.判断素数:

isPrime 方法判断一个数是否为素数:

- 小于等于 1 的数不是素数。
- 2 是唯一的偶数素数。
- 排除其他偶数。
- 只需检查到平方根即可提高效率。

##### 4.输出结果:

打印所有找到的素数，并统计素数的数量。

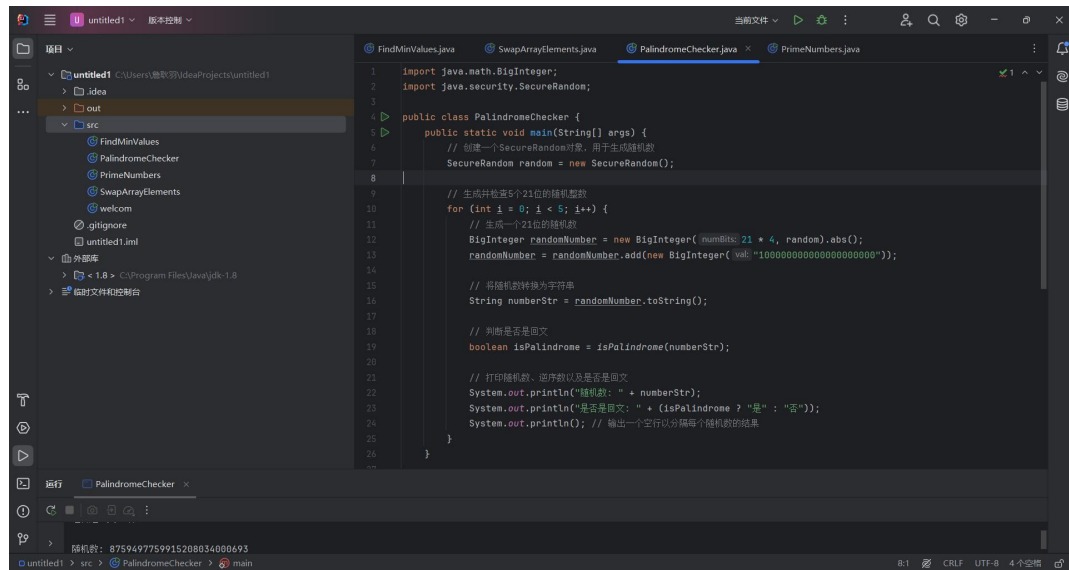
(2.4) 编写 Java 程序：随机生成 5 个 21 位数（整数），并判断它是不是回文。要求对每个生成的随机数输出三个信息：随机数、逆序数、是否是回文。所谓“回文”是指一种从前向后读和从后向前读都一样的数字，例如，1234321、322223。对每一行语句加上注释。在报告中附上程序截图、运行结果截图和详细的文字说明。（10 分）

✓ 代码

```
1. import java.math.BigInteger;
2. import java.security.SecureRandom;
3.
4. public class PalindromeChecker {
5.     public static void main(String[] args) {
6.         // 创建一个 SecureRandom 对象，用于生成随机数
7.         SecureRandom random = new SecureRandom();
8.
9.         // 生成并检查 5 个 21 位的随机整数
10.        for (int i = 0; i < 5; i++) {
11.            // 生成一个 21 位的随机数
12.            BigInteger randomNumber = new BigInteger(21 * 4, random).abs();
13.            randomNumber = randomNumber.add(new BigInteger("100000000000000000000"));
14.
15.            // 将随机数转换为字符串
16.            String numberStr = randomNumber.toString();
17.
18.            // 判断是否是回文
19.            boolean isPalindrome = isPalindrome(numberStr);
20.
21.            // 打印随机数、逆序数以及是否是回文
22.            System.out.println("随机数: " + numberStr);
23.            System.out.println("是否是回文: " + (isPalindrome ? "是" : "否"));
24.            System.out.println(); // 输出一个空行以分隔每个随机数的结果
25.        }
26.    }
27.
28.    // 自定义判断字符串是否是回文的函数
29.    private static boolean isPalindrome(String str) {
30.        int left = 0;
31.        int right = str.length() - 1;
32.
33.        // 比较左右字符
34.        while (left < right) {
35.            if (str.charAt(left) != str.charAt(right)) {
```

```
36.                return false; // 发现不同字符, 返回false
37.            }
38.            left++;
39.            right--;
40.        }
41.        return true; // 所有字符匹配, 返回true
42.    }
43. }
```

#### ✓ 程序截图



```
1  import java.math.BigInteger;
2  import java.security.SecureRandom;
3
4  public class PalindromeChecker {
5      public static void main(String[] args) {
6          // 创建一个SecureRandom对象, 用于生成随机数
7          SecureRandom random = new SecureRandom();
8
9          // 生成并输出5个21位的随机整数
10         for (int i = 0; i < 5; i++) {
11             // 生成一个21位的随机数
12             BigInteger randomNumber = new BigInteger( numBits: 21 * 4, random).abs();
13             randomNumber = randomNumber.add(new BigInteger( val: "10800000000000000000"));
14
15             // 将随机数转换为字符串
16             String numberStr = randomNumber.toString();
17
18             // 判断是否是回文
19             boolean isPalindrome = isPalindrome(numberStr);
20
21             // 打印随机数、逆序数以及是否是回文
22             System.out.println("随机数: " + numberStr);
23             System.out.println("是否是回文: " + (isPalindrome ? "是" : "否"));
24             System.out.println(); // 输出一个空行以分隔每个随机数的结果
25         }
26     }
```

#### ✓ 结果

```
随机数: 239230616763372098329348
逆序数: 843923890273367616032932
是否是回文: 否

随机数: 5637008511788790580562833
逆序数: 3382650850978871158007365
是否是回文: 否

随机数: 17980817052670634988495381
逆序数: 18359488943607625071808971
是否是回文: 否

随机数: 214236213523015526534594
逆序数: 495435625510325312632412
是否是回文: 否

随机数: 3290857897756873872231109
逆序数: 9011322783786577987580923
是否是回文: 否
```

```
进程已结束, 退出代码为 0
```

✓ 文字说明

1. 导入必要的类:

```
import java.math.BigInteger;
import java.security.SecureRandom;
- 'BigInteger' 类用于处理大整数。
- 'SecureRandom' 类用于生成安全的随机数。
```

2. 主方法:

```
public static void main(String[] args) {
- 程序的入口点。
```

3. 创建 'SecureRandom' 对象:

```
SecureRandom random = new SecureRandom();
- 用于生成高质量的随机数。
```

4. 生成并检查 5 个 21 位的随机整数:

```
for (int i = 0; i < 5; i++) {
- 循环 5 次, 每次生成一个 21 位的随机整数并检查是否是回文。
```

5. 生成 21 位的随机整数:

```
BigInteger randomNumber = new BigInteger(21 * 4, random).abs();
randomNumber = randomNumber.add(new BigInteger("100000000000000000000"));
```



- `new BigInteger(21 * 4, random)`: 生成一个具有 84 位二进制的随机数。21 位十进制数大约需要 63 位二进制，因此生成更大的位数确保随机数足够大。

- `.abs()`: 取绝对值以确保随机数为非负。

- `randomNumber.add(new BigInteger("10000000000000000000"))`: 确保随机数至少是 21 位。通过加上一个 21 位的最小值来实现。

6. 将随机数转换为字符串:

```
String numberStr = randomNumber.toString();
```

7. 判断是否是回文:

```
boolean isPalindrome = isPalindrome(numberStr);
```

- 调用自定义的 `isPalindrome` 方法来判断字符串是否是回文。

8. 打印结果:

```
System.out.println("随机数: " + numberStr);
```

```
System.out.println("是否是回文: " + (isPalindrome ? "是" : "否"));
```

```
System.out.println();
```

- 打印每个随机数和它是否是回文的结果。

- 通过条件运算符 `?:` 来简洁地输出 "是" 或 "否"。

9. 自定义的回文判断方法:

```
private static boolean isPalindrome(String str) {  
    int left = 0;  
    int right = str.length() - 1;  
  
    // 比较左右字符  
    while (left < right) {  
        if (str.charAt(left) != str.charAt(right)) {  
            return false; // 发现不同字符，返回 false  
        }  
        left++;  
        right--;  
    }  
    return true; // 所有字符匹配，返回 true  
}
```

- `left` 和 `right` 分别指向字符串的开头和结尾。

- 在 `while` 循环中逐一比较字符:

- 如果发现左右字符不相等，则不是回文，返回 `false`。

- 否则，继续向中间移动。

- 如果所有字符都匹配，则返回 `true`，表示是回文。

### **Part 3 (30 分)**

(3.1).运行第 4 章课件中第 4 页、第 24 页、第 32 页和第 34 页中的四个程序，并对每一行语句加上注释。对每一行语句加上注释。在报告中附上程序截图、运行结果截图和简要的文字说明。(5 分)

程序 1:

✓ 程序截图

```
// 定义一个 Circle 类
class Circle 2个用法
{
    // 声明一个 double 类型的变量 radius, 用于存储圆的半径
    double radius; 3个用法

    // 定义一个方法 getArea, 用于计算圆的面积
    double getArea() 1个用法
    {
        // 计算面积, 使用公式  $\pi * r * r$ , 这里用 3.14 作为  $\pi$  的近似值
        double area = 3.14 * radius * radius;
        // 返回计算得到的面积
        return area;
    }
}

// 定义一个公共类 Example4_2
public class Example4_2
{
    // 主方法, 程序的入口
    public static void main(String args[])
    {
        // 声明一个 Circle 类型的变量 circle
        Circle circle;
        // 实例化 Circle 对象
        circle = new Circle();
        // 设置圆的半径为 1
    }
}
```

#### ✓ 运行结果



```
"C:\Program Files\Java\jdk-1.8\bin\java.exe" ...
20
100
进程已结束, 退出代码为 0
|

untitled1 > src > Example4_2.java > Example4_2 > main
```

#### ✓ 完整代码

1. // 定义一个 Circle 类
2. class Circle
3. {
4. // 声明一个 double 类型的变量 radius, 用于存储圆的半径
5. double radius;

```

6.
7.      // 定义一个方法 getArea, 用于计算圆的面积
8.      double getArea()
9.      {
10.         // 计算面积, 使用公式  $\pi * r * r$ , 这里用 3.14 作为  $\pi$  的近似
            值
11.         double area = 3.14 * radius * radius;
12.         // 返回计算得到的面积
13.         return area;
14.     }
15. }
16.
17. // 定义一个公共类 Example4_2
18. public class Example4_2
19. {
20.     // 主方法, 程序的入口
21.     public static void main(String args[])
22.     {
23.         // 声明一个 Circle 类型的变量 circle
24.         Circle circle;
25.         // 实例化 Circle 对象
26.         circle = new Circle();
27.         // 设置圆的半径为 1
28.         circle.radius = 1;
29.         // 调用 getArea 方法计算面积, 并将结果存储在 area 变量中
30.         double area = circle.getArea();
31.         // 输出计算得到的面积
32.         System.out.println(area);
33.     }
34. }

```

✓ 文字说明

先定义一个圆的类, 里面含有半径的变量和求得圆面积的方法。然后在公共类 Example4\_2 中声明一个 Circle 类型的变量 circle, 设置圆的半径为 1, 再调用圆的求面积函数来求得该圆的面积。

程序 2:

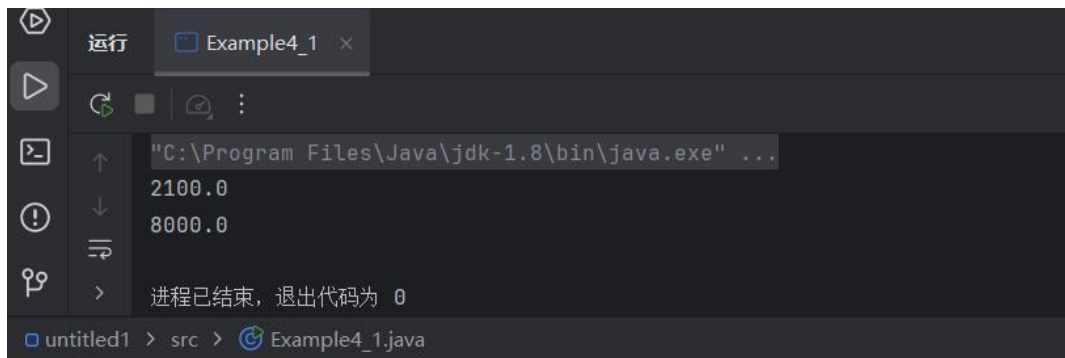
✓ 程序截图

```

1 // 定义一个 Ladder 类
2 class Ladder 3个用法
3 {
4     // 声明三个 double 类型的变量，分别表示梯子的上边长、下边长和高度
5     double above, bottom, height; 3个用法
6
7     // 默认构造函数
8     Ladder() {} 1个用法
9
10    // 带参数的构造函数，用于初始化梯子的上边长、下边长和高度
11    Ladder(double a, double b, double h) 1个用法
12    {
13        above = a; // 设置上边长
14        bottom = b; // 设置下边长
15        height = h; // 设置高度
16    }
17
18    // 设置上边长的方法
19    public void setAbove(double a) 2个用法
20    {
21        above = a; // 将参数 a 赋值给上边长
22    }
23
24    // 设置下边长的方法

```

#### ✓ 运行结果



```

运行 Example4_1 x
C:\Program Files\Java\jdk-1.8\bin\java.exe" ...
2100.0
8000.0
进程已结束，退出代码为 0
untitled1 > src > Example4_1.java

```

#### ✓ 完整代码

```

1. // 定义一个 Ladder 类
2. class Ladder
3. {
4.     // 声明三个 double 类型的变量，分别表示梯子的上边长、下边长和高
    度
5.     double above, bottom, height;
6.
7.     // 默认构造函数
8.     Ladder() {}
9.
10.    // 带参数的构造函数，用于初始化梯子的上边长、下边长和高度
11.    Ladder(double a, double b, double h)
12.    {

```

```
13.         above = a;    // 设置上边长
14.         bottom = b;   // 设置下边长
15.         height = h;   // 设置高度
16.     }
17.
18.     // 设置上边长的方法
19.     public void setAbove(double a)
20.     {
21.         above = a;    // 将参数 a 赋值给上边长
22.     }
23.
24.     // 设置下边长的方法
25.     public void setBottom(double b)
26.     {
27.         bottom = b;   // 将参数 b 赋值给下边长
28.     }
29.
30.     // 设置高度的方法
31.     public void setHeight(double h)
32.     {
33.         height = h;   // 将参数 h 赋值给高度
34.     }
35.
36.     // 计算梯子面积的方法
37.     double computeArea()
38.     {
39.         // 使用梯形面积公式 (上边长 + 下边长) * 高 / 2
40.         return (above + bottom) * height / 2.0;
41.     }
42. }
43.
44. // 定义一个公共类 Example4_1
45. public class Example4_1
46. {
47.     // 主方法，程序的入口
48.     public static void main(String args[])
49.     {
50.         // 声明两个 double 类型的变量，用于存储梯子的面积
51.         double area1 = 0, area2 = 0;
52.         // 声明两个 Ladder 类型的变量
53.         Ladder ladderOne, ladderTwo;
54.
55.         // 实例化第一个 Ladder 对象，使用默认构造函数
56.         ladderOne = new Ladder();
```

```

57.          // 实例化第二个 Ladder 对象，使用带参数的构造函数
58.          ladderTwo = new Ladder(10, 88, 20);
59.
60.          // 设置第一个梯子的上边长
61.          ladderOne.setAbove(16);
62.          // 设置第一个梯子的下边长
63.          ladderOne.setBottom(26);
64.          // 设置第一个梯子的高度
65.          ladderOne.setHeight(100);
66.
67.          // 设置第二个梯子的上边长
68.          ladderTwo.setAbove(300);
69.          // 设置第二个梯子的下边长
70.          ladderTwo.setBottom(500);
71.
72.          // 计算第一个梯子的面积
73.          area1 = ladderOne.computeArea();
74.          // 计算第二个梯子的面积
75.          area2 = ladderTwo.computeArea();
76.
77.          // 输出第一个梯子的面积
78.          System.out.println(area1);
79.          // 输出第二个梯子的面积
80.          System.out.println(area2);
81.      }
82.  }

```

✓ 文字说明

首先定义了一个梯形类，里面含有上边长、下边长、高度，里面的方法有默认构造函数 `ladder()` {}，带参构造函数，还有设置上、下边长，高度的方法以及计算梯形面积的方法。

接着定义了一个公共类 `Example4_1`，初始化构造了第一个梯子，带参构造了第二个梯子。然后设置第一个梯子的上下边长和高度，以及第二个梯子的上下边长，在分别计算他们的面积后打印。

程序 3:

✓ 程序截图



```

// 定义一个 Ladder1 类
class Ladder1 6个用法
{
    double above, height; // 实例变量：上边长和高度 2个用法
    static double bottom; // 静态变量：下边长 5个用法

    // 设置上边长的方法
    void setAbove(double a) 2个用法
    {
        above = a; // 将参数 a 赋值给上边长
    }

    // 设置下边长的方法
    void setBottom(double b) 1个用法
    {
        bottom = b; // 将参数 b 赋值给静态下边长
    }

    // 获取上边长的方法
    double getAbove() 2个用法
    {
        return above; // 返回上边长
    }
}

```

#### ✓ 运行结果

```

运行 Example4_21 x
"C:\Program Files\Java\jdk-1.8\bin\java.exe" ...
60.0
60.0
60.0
100.0
11.0
22.0

进程已结束，退出代码为 0

```

untitled1 > src > Example4\_21.java > Example4\_21 > main

#### ✓ 完整代码

1. // 定义一个 Ladder1 类
2. class Ladder1
3. {

```
4.     double above, height; // 实例变量: 上边长和高度
5.     static double bottom; // 静态变量: 下边长
6.
7.     // 设置上边长的方法
8.     void setAbove(double a)
9.     {
10.         above = a; // 将参数 a 赋值给上边长
11.     }
12.
13.    // 设置下边长的方法
14.    void setBottom(double b)
15.    {
16.        bottom = b; // 将参数 b 赋值给静态下边长
17.    }
18.
19.    // 获取上边长的方法
20.    double getAbove()
21.    {
22.        return above; // 返回上边长
23.    }
24.
25.    // 获取下边长的方法
26.    double getBottom()
27.    {
28.        return bottom; // 返回静态下边长
29.    }
30. }
31.
32. // 定义一个公共类 Example4_21
33. public class Example4_21
34. {
35.     // 主方法, 程序的入口
36.     public static void main(String args[])
37.     {
38.         Ladder1.bottom = 60; // 设置静态下边长
39.         Ladder1 ladderOne, ladderTwo; // 声明两个 Ladder1 对象
40.         System.out.println(Ladder1.bottom); // 输出静态下边长
41.
42.         ladderOne = new Ladder1(); // 实例化第一个 Ladder1 对象
43.         ladderTwo = new Ladder1(); // 实例化第二个 Ladder1 对象
44.
45.         // 输出第一个和第二个对象的静态下边长
46.         System.out.println(ladderOne.getBottom());
47.         System.out.println(ladderTwo.getBottom());
```

```

48.
49.         ladderOne.setAbove(11); // 设置第一个对象的上边长
50.         ladderTwo.setAbove(22); // 设置第二个对象的上边长
51.         ladderTwo.setBottom(100); // 设置静态下边长
52.
53.         // 输出更新后的静态下边长
54.         System.out.println(Ladder1.bottom);
55.         // 输出两个对象的上边长
56.         System.out.println(ladderOne.getAbove());
57.         System.out.println(ladderTwo.getAbove());
58.     }
59. }

```

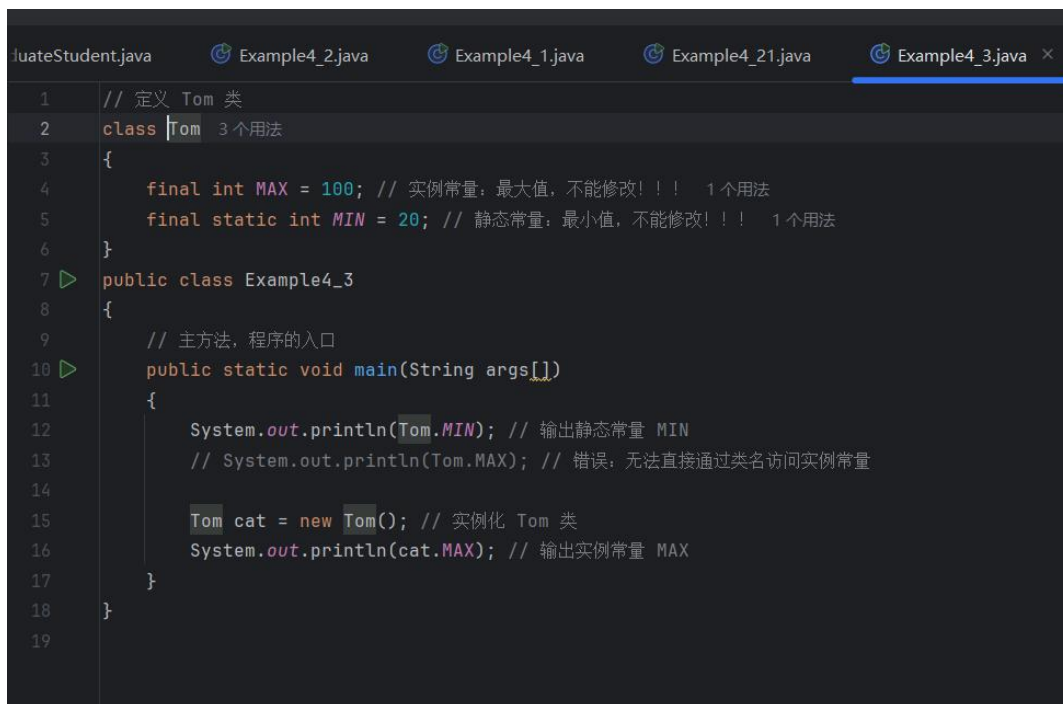
#### ✓ 文字说明

首先定义了一个梯形类，与程序 3 不同的是，里面含有上边长、下边长（静态变量）、高度，里面的方法有有设置上、下边长，高度的方法以及得到上下边长的方法。

接着定义了一个公共类 Example4\_21，在主方法中，设置了 Ladder1 的下边长（静态），然后声明 2 个 Ladder1 对象，再输出 Ladder1 的下边长。接着将声明的两个对象进行实例化，从而输出两个对象的下边长，再设置他们的上边长，以及设置类的下边长，最后输出更新后的下边长，以及两个对象的上边长。

程序 4:

#### ✓ 程序截图

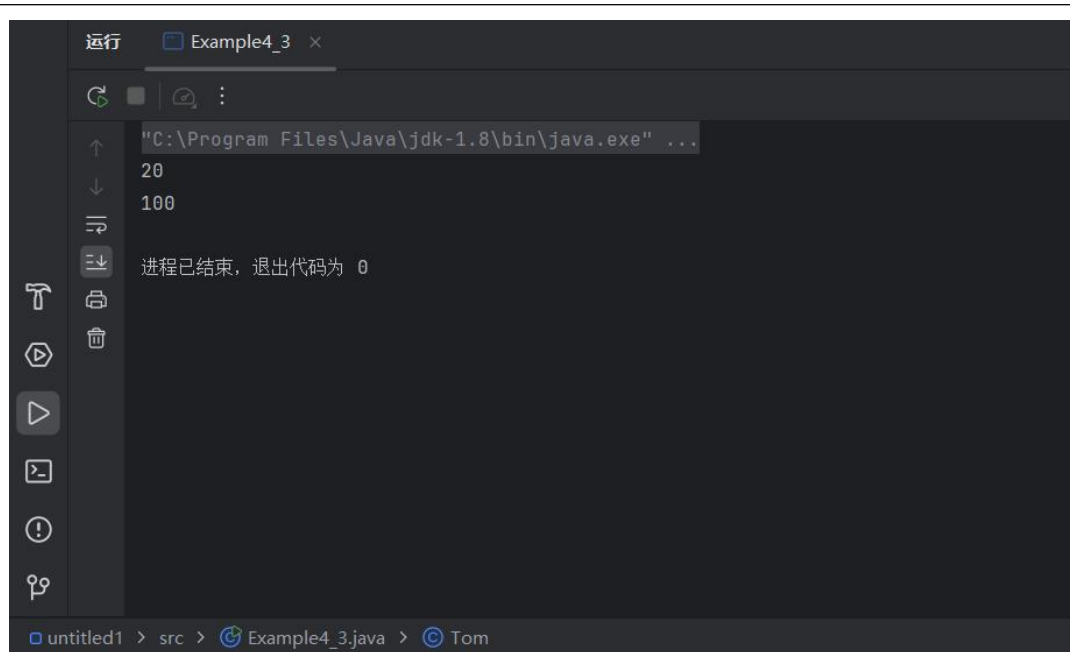


```

1 // 定义 Tom 类
2 class Tom 3 个用法
3 {
4     final int MAX = 100; // 实例常量：最大值，不能修改!!! 1 个用法
5     final static int MIN = 20; // 静态常量：最小值，不能修改!!! 1 个用法
6 }
7 public class Example4_3
8 {
9     // 主方法，程序的入口
10    public static void main(String args[])
11    {
12        System.out.println(Tom.MIN); // 输出静态常量 MIN
13        // System.out.println(Tom.MAX); // 错误：无法直接通过类名访问实例常量
14
15        Tom cat = new Tom(); // 实例化 Tom 类
16        System.out.println(cat.MAX); // 输出实例常量 MAX
17    }
18 }
19

```

#### ✓ 运行结果



#### ✓ 完整代码

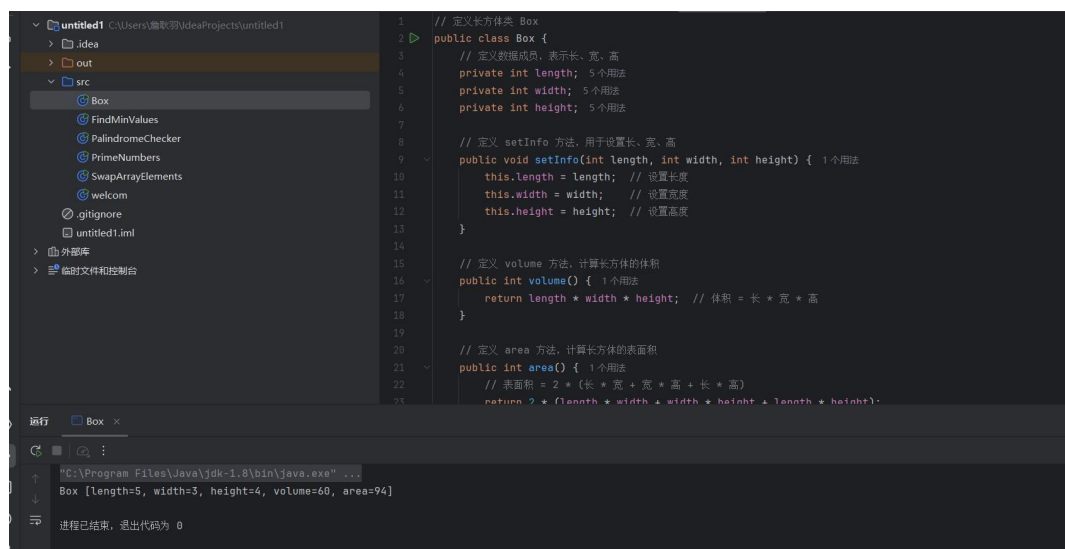
由于截图完整，这里为了节省空间不呈现完整代码。

#### ✓ 文字说明

首先定义了 Tom 类型，Tom 类内有 MAX 实例常量（最大值）和静态常量（最小值），定义公共类 Eample4\_3，在主方法中，输出 Tom 的静态常量 MIN，然后再定义一个 Tom 类，名为 cat，再输出 cat 的最大值（MAX）。

(3.2).设计并测试一个长方体类 Box。(i) 数据成员包括 length、width 和 height，分别表示长方体的长、宽和高；(ii) 定义 setInfo(int,int,int)方法设置这 3 个数据成员的值；(iii) 定义 volume()方法求长方体的体积；(iv) 定义 area()方法求长方体的表面积；(v) 定义 toString()方法把长方体的长、宽、高以及长方体的体积和表面积转化为字符串并返回。对每一行语句加上注释。在报告中附上程序截图、运行结果截图和简要的文字说明。（5 分）

#### ✓ 程序截图



## ✓ 运行结果

```
Box [length=5, width=3, height=4, volume=60, area=94]
进程已结束，退出代码为 0
```

## ✓ 完整代码

```
1. // 定义长方体类 Box
2. public class Box {
3.     // 定义数据成员，表示长、宽、高
4.     private int length;
5.     private int width;
6.     private int height;
7.
8.     // 定义 setInfo 方法，用于设置长、宽、高
9.     public void setInfo(int length, int width, int height) {
10.         this.length = length; // 设置长度
11.         this.width = width; // 设置宽度
12.         this.height = height; // 设置高度
13.     }
14.
15.     // 定义 volume 方法，计算长方体的体积
16.     public int volume() {
17.         return length * width * height; // 体积 = 长 * 宽 * 高
18.     }
19.
20.     // 定义 area 方法，计算长方体的表面积
21.     public int area() {
22.         // 表面积 = 2 * (长 * 宽 + 宽 * 高 + 长 * 高)
23.         return 2 * (length * width + width * height + length *
24.             height);
25.     }
26.     // 定义 toString 方法，返回长方体的基本信息和计算结果
27.     @Override
28.     public String toString() {
29.         // 返回包含长、宽、高、体积、表面积的字符串
30.         return "Box [length=" + length + ", width=" + width + "
31.             , height=" + height
32.             + ", volume=" + volume() + ", area=" + area() +
33.             "]";
34.     }
35.
36.     // 主方法，用于测试 Box 类
```

```

35.     public static void main(String[] args) {
36.         // 创建一个 Box 对象
37.         Box box = new Box();
38.
39.         // 设置长方体的长、宽、高
40.         box.setInfo(5, 3, 4);
41.
42.         // 打印长方体的基本信息
43.         System.out.println(box.toString());
44.     }
45. }

```

#### ✓ 文字说明

##### 1. 类和成员变量的定义：

- 定义了类 `Box`，其中包含三个私有数据成员 `length`、`width` 和 `height`，分别表示长方体的长、宽、高。

##### 2. 方法 `setInfo(int, int, int)`：

- 该方法用于设置长方体的长、宽和高，通过参数传入三个整数，分别赋值给 `length`、`width` 和 `height`。

##### 3. 方法 `volume()`：

- 该方法用于计算长方体的体积，体积的计算公式为：长  $\times$  宽  $\times$  高。

##### 4. 方法 `area()`：

- 该方法用于计算长方体的表面积。

##### 5. 方法 `toString()`：

- 该方法返回一个描述长方体的字符串，包括长、宽、高以及体积和表面积。

##### 6. 测试代码：

- 在 `main()` 方法中创建了一个 `Box` 对象，并通过 `setInfo()` 设置长、宽、高的值。然后调用 `toString()` 打印长方体的相关信息。

(3.3)参照题(2)设计并测试一个圆锥体 Cone。对每一行语句加上注释。在报告中附上程序截图、运行结果截图和简要的文字说明。（5分）

#### ✓ 程序截图

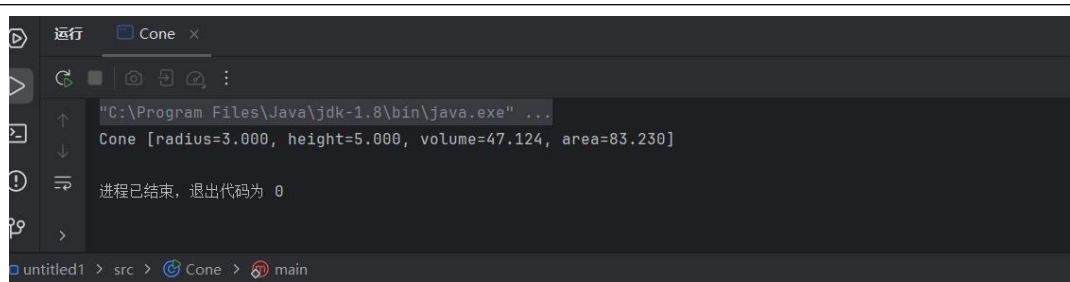
```

1 // 定义圆锥体类 Cone
2 public class Cone {
3     // 定义数据成员，表示圆锥体的半径和高
4     private double radius; 6个用法
5     private double height; 4个用法
6
7     // 定义 setInfo 方法，用于设置半径和高
8     public void setInfo(double radius, double height) { 1个用法
9         this.radius = radius; // 设置圆锥体的半径
10        this.height = height; // 设置圆锥体的高
11    }
12
13    // 定义 volume 方法，计算圆锥体的体积
14    public double volume() { 1个用法
15        // 体积公式: (1/3) * π * r^2 * h
16        return (1.0 / 3) * Math.PI * Math.pow(radius, 2) * height;
17    }
18
19    // 定义 slantHeight 方法，计算圆锥体的斜高
20    public double slantHeight() { 1个用法
21        // 斜高公式: sqrt(r^2 + h^2)
22        return Math.sqrt(Math.pow(radius, 2) + Math.pow(height, 2));
23    }
24 }

```

#### ✓ 运行结果





```
运行 Cone x
"C:\Program Files\Java\jdk-1.8\bin\java.exe" ...
Cone [radius=3.000, height=5.000, volume=47.124, area=83.230]
进程已结束, 退出代码为 0
untitled1 > src > Cone > main
```

✓ 完整代码

```
1. // 定义圆锥体类 Cone
2. public class Cone {
3.     // 定义数据成员, 表示圆锥体的半径和高
4.     private double radius;
5.     private double height;
6.
7.     // 定义 setInfo 方法, 用于设置半径和高
8.     public void setInfo(double radius, double height) {
9.         this.radius = radius; // 设置圆锥体的半径
10.        this.height = height; // 设置圆锥体的高
11.    }
12.
13.    // 定义 volume 方法, 计算圆锥体的体积
14.    public double volume() {
15.        // 体积公式:  $(1/3) * \pi * r^2 * h$ 
16.        return (1.0 / 3) * Math.PI * Math.pow(radius, 2) * height;
17.    }
18.
19.    // 定义 slantHeight 方法, 计算圆锥体的斜高
20.    public double slantHeight() {
21.        // 斜高公式:  $\sqrt{r^2 + h^2}$ 
22.        return Math.sqrt(Math.pow(radius, 2) + Math.pow(height, 2));
23.    }
24.
25.    // 定义 area 方法, 计算圆锥体的表面积
26.    public double area() {
27.        // 表面积公式:  $\pi * r * (r + \text{斜高})$ 
28.        return Math.PI * radius * (radius + slantHeight());
29.    }
30.
31.    // 定义 toString 方法, 返回圆锥体的基本信息和计算结果
32.    @Override
33.    public String toString() {
34.        // 返回包含半径、高、体积、表面积的字符串, 保留三位小数
```

```

35.         return String.format("Cone [radius=%.3f, height=%.3f, v
           olume=%.3f, area=%.3f]",
36.             radius, height, volume(), area());
37.     }
38.
39.     // 主方法, 用于测试 Cone 类
40.     public static void main(String[] args) {
41.         // 创建一个 Cone 对象
42.         Cone cone = new Cone();
43.
44.         // 设置圆锥体的半径和高度
45.         cone.setInfo(3, 5);
46.
47.         // 打印圆锥体的基本信息
48.         System.out.println(cone.toString());
49.     }
50. }

```

✓ 文字说明

1. 类和成员变量的定义:

- 定义了类 `Cone`, 其中包含两个私有数据成员 `radius` 和 `height`, 分别表示圆锥体的半径和高。

2. 方法 `setInfo(double, double)`:

- 该方法用于设置圆锥体的半径和高度, 通过参数传入两个浮点数, 分别赋值给 `radius` 和 `height`。

3. 方法 `volume()`:

- 该方法用于计算圆锥体的体积。

4. 方法 `slantHeight()`:

- 该方法用于计算圆锥体的斜高。

5. 方法 `area()`:

- 该方法用于计算圆锥体的表面积。

6. 方法 `toString()`:

- 该方法返回一个描述圆锥体的字符串, 包括半径、高、体积和表面积。

7. 测试代码:

- 在 `main()` 方法中创建了一个 `Cone` 对象, 并通过 `setInfo()` 设置半径和高度的值。然后调用 `toString()` 打印圆锥体的相关信息。

(3.4).设计并测试一个研究生类 PostGraduateStudent。(i) 数据成员包括 ID (学号)、name (姓名) 以及 3 门课程 math、programming、english; (ii) 定义 comSum()、comAvg()、comMax() 计算 3 门课程的总分、平均分和最高分; (iii) 在该类中实现对两个学生进行比较的方法 (根据总分)。对每一行语句加上注释。在报告中附上程序截图、运行结果截图和简要的文字说明。(5 分)

✓ 程序截图

```
// 定义研究生类 PostGraduateStudent
public class PostGraduateStudent {
    // 定义数据成员：学号、姓名和3门课程的成绩
    private String ID; 2个用法
    private String name; 3个用法
    private double math; 3个用法
    private double programming; 3个用法
    private double english; 3个用法

    // 定义构造方法，用于初始化学生的ID、姓名和3门课程成绩
    public PostGraduateStudent(String ID, String name, double math, double programming, double english) {
        this.ID = ID; // 设置学号
        this.name = name; // 设置姓名
        this.math = math; // 设置数学成绩
        this.programming = programming; // 设置编程成绩
        this.english = english; // 设置英语成绩
    }

    // 定义 comSum 方法，计算3门课程的总分
    public double comSum() { 4个用法
        // 总分 = 数学成绩 + 编程成绩 + 英语成绩
        return math + programming + english;
    }
}
```

#### ✓ 运行结果

```
"C:\Program Files\Java\jdk-1.8\bin\java.exe" ...
PostGraduateStudent [ID=1001, name=james, sum=255.00, avg=85.00, max=90.00]
PostGraduateStudent [ID=1002, name=lebron, sum=258.00, avg=86.00, max=92.00]
The student with the higher score is: lebron

进程已结束，退出代码为 0
```

#### ✓ 完整代码

```
1. // 定义研究生类 PostGraduateStudent
2. public class PostGraduateStudent {
3.     // 定义数据成员：学号、姓名和3门课程的成绩
4.     private String ID;
5.     private String name;
6.     private double math;
7.     private double programming;
8.     private double english;
9.
10.    // 定义构造方法，用于初始化学生的ID、姓名和3门课程成绩
11.    public PostGraduateStudent(String ID, String name, double math, double programming, double english) {
12.        this.ID = ID; // 设置学号
13.        this.name = name; // 设置姓名
14.        this.math = math; // 设置数学成绩
15.        this.programming = programming; // 设置编程成绩
16.        this.english = english; // 设置英语成绩
17.    }
```

```
18.
19.    // 定义 comSum 方法, 计算 3 门课程的总分
20.    public double comSum() {
21.        // 总分 = 数学成绩 + 编程成绩 + 英语成绩
22.        return math + programming + english;
23.    }
24.
25.    // 定义 comAvg 方法, 计算 3 门课程的平均分
26.    public double comAvg() {
27.        // 平均分 = 总分 / 3
28.        return comSum() / 3;
29.    }
30.
31.    // 定义 comMax 方法, 计算 3 门课程中的最高分
32.    public double comMax() {
33.        // 最高分 = 取数学、编程、英语中的最大值
34.        return Math.max(math, Math.max(programming, english));
35.    }
36.
37.    // 定义 compare 方法, 比较两个学生的总分
38.    public static PostGraduateStudent compare(PostGraduateStudent s1, PostGraduateStudent s2) {
39.        // 比较两个学生的总分, 返回总分较高的学生
40.        if (s1.comSum() > s2.comSum()) {
41.            return s1; // 如果学生 1 的总分高, 返回学生 1
42.        } else {
43.            return s2; // 否则返回学生 2
44.        }
45.    }
46.
47.    // 定义 toString 方法, 返回学生的基本信息和成绩信息
48.    @Override
49.    public String toString() {
50.        // 返回学生的 ID、姓名、总分、平均分和最高分的信息
51.        return String.format("PostGraduateStudent [ID=%s, name=%s, sum=%.2f, avg=%.2f, max=%.2f]",
52.                               ID, name, comSum(), comAvg(), comMax());
53.    }
54.
55.    // 主方法, 用于测试 PostGraduateStudent 类
56.    public static void main(String[] args) {
57.        // 创建两个 PostGraduateStudent 对象
58.        PostGraduateStudent student1 = new PostGraduateStudent(
            "1001", "james", 85, 90, 80);
```

```

59.         PostGraduateStudent student2 = new PostGraduateStudent(
        "1002", "lebron" +
60.             "", 78, 88, 92);
61.
62.         // 打印两个学生的成绩信息
63.         System.out.println(student1.toString());
64.         System.out.println(student2.toString());
65.
66.         // 比较两个学生的总分，输出成绩较高的学生
67.         PostGraduateStudent topStudent = PostGraduateStudent.co
        mpare(student1, student2);
68.         System.out.println("The student with the higher score i
        s: " + topStudent.name);
69.     }
70. }

```

✓ 文字说明

1. 类和成员变量的定义：

- 定义了类 `PostGraduateStudent`，包含数据成员 `ID`（学号）、`name`（姓名）和 3 门课程的成绩 `math`（数学）、`programming`（编程）和 `english`（英语）。

2. 构造方法：

- 用于初始化学生的学号、姓名和 3 门课程的成绩。

3. 方法 `comSum()`：

- 该方法用于计算 3 门课程的总分。

4. 方法 `comAvg()`：

- 该方法用于计算 3 门课程的平均分。

5. 方法 `comMax()`：

- 该方法用于计算 3 门课程中的最高分。

6. 静态方法 `compare()`：

- 该方法用于比较两个学生的总分，返回总分较高的学生对象。

7. 方法 `toString()`：

- 该方法返回包含学生基本信息和成绩的字符串。

8. 测试代码：

- 在 `main()` 方法中，创建了两个学生对象，打印每个学生的基本信息，并比较两个学生的总分，输出总分较高的学生。

（3.5）编写一个 `Teacher` 类。类中包含以下成员变量：`name`（姓名）、`title`（职位）、`course`（主讲的课程）、`research`（研究方向）和 `office`（办公室）。定义对应的方法对这几个成员变量的值进行设置和读取。（i）在 `Teacher` 类外的 `main` 方法里面，创建该类的一个对象，并调用各个方法，展示相应的效果。（ii）在 `Teacher` 类内的 `main` 方法里面，创建该类的一个对象，并调用各个方法，展示相应的效果。在报告中附上程序截图、运行结果截图和简要的文字说明。（5 分）

✓ 程序截图

```
ts\untitled1 1 // 定义教师类 Teacher
2 public class Teacher {
3     // 定义数据成员：姓名、职位、主讲课程、研究方向、办公室
4     private String name; 3个用法
5     private String title; 3个用法
6     private String course; 3个用法
7     private String research; 3个用法
8     private String office; 3个用法
9
10    // 定义 setName 方法，用于设置姓名
11    public void setName(String name) { 1个用法
12        this.name = name;
13    }
14
15    // 定义 getName 方法，用于获取姓名
16    public String getName() { 0个用法
17        return name;
18    }
19
20    // 定义 setTitle 方法，用于设置职位
21    public void setTitle(String title) { 1个用法
22        this.title = title;
23    }
```

```
1 public class Main {
2     public static void main(String[] args) {
3         // 创建一个 Teacher 对象
4         Teacher teacher = new Teacher();
5
6         // 设置教师的姓名、职位、主讲课程、研究方向和办公室
7         teacher.setName("Dr. Alice");
8         teacher.setTitle("Associate Professor");
9         teacher.setCourse("Data Science");
10        teacher.setResearch("Machine Learning");
11        teacher.setOffice("Room 405");
12
13        // 读取并打印各个属性
14        System.out.println("Name: " + teacher.getName());
15        System.out.println("Title: " + teacher.getTitle());
16        System.out.println("Course: " + teacher.getCourse());
17        System.out.println("Research: " + teacher.getResearch());
18        System.out.println("Office: " + teacher.getOffice());
19    }
20 }
```

✓ 运行结果

类内：

```
C:\Program Files\Java\jdk-1.8\bin\java.exe ...
Teacher [name=Dr. John, title=Professor, course=Computer Science, research=Artificial Intelligence, office=Room 203]
进程已结束，退出代码为 0
```

类外：

✓ 完整代码

由于类外已给出截图，这里呈现类内的。

```
1. // 定义教师类 Teacher
2. public class Teacher {
3.     // 定义数据成员：姓名、职位、主讲课程、研究方向、办公室
4.     private String name;
5.     private String title;
6.     private String course;
7.     private String research;
8.     private String office;
9.
10.    // 定义 setName 方法，用于设置姓名
11.    public void setName(String name) {
12.        this.name = name;
13.    }
14.
15.    // 定义 getName 方法，用于获取姓名
16.    public String getName() {
17.        return name;
18.    }
19.
20.    // 定义 setTitle 方法，用于设置职位
21.    public void setTitle(String title) {
22.        this.title = title;
23.    }
24.
25.    // 定义 getTitle 方法，用于获取职位
26.    public String getTitle() {
27.        return title;
28.    }
29.
30.    // 定义 setCourse 方法，用于设置主讲课程
```

```
31.     public void setCourse(String course) {
32.         this.course = course;
33.     }
34.
35.     // 定义 getCourse 方法, 用于获取主讲课程
36.     public String getCourse() {
37.         return course;
38.     }
39.
40.     // 定义 setResearch 方法, 用于设置研究方向
41.     public void setResearch(String research) {
42.         this.research = research;
43.     }
44.
45.     // 定义 getResearch 方法, 用于获取研究方向
46.     public String getResearch() {
47.         return research;
48.     }
49.
50.     // 定义 setOffice 方法, 用于设置办公室
51.     public void setOffice(String office) {
52.         this.office = office;
53.     }
54.
55.     // 定义 getOffice 方法, 用于获取办公室
56.     public String getOffice() {
57.         return office;
58.     }
59.
60.     // 定义 toString 方法, 返回教师的基本信息
61.     @Override
62.     public String toString() {
63.         return String.format("Teacher [name=%s, title=%s, course=%s, research=%s, office=%s]",
64.                                name, title, course, research, office);
65.     }
66.
67.     // 在 Teacher 类内定义的 main 方法
68.     public static void main(String[] args) {
69.         // 在 Teacher 类内创建一个 Teacher 对象
70.         Teacher teacher = new Teacher();
71.
72.         // 设置教师的姓名、职位、主讲课程、研究方向和办公室
73.         teacher.setName("Dr. John");
```



```

74.         teacher.setTitle("Professor");
75.         teacher.setCourse("Computer Science");
76.         teacher.setResearch("Artificial Intelligence");
77.         teacher.setOffice("Room 203");
78.
79.         // 打印教师的信息
80.         System.out.println(teacher.toString());
81.     }
82. }

```

✓ 文字说明

1. 类和成员变量的定义：

- `Teacher` 类包含 5 个成员变量：`name`（姓名）、`title`（职位）、`course`（主讲课程）、`research`（研究方向）和 `office`（办公室）。

2. `set` 和 `get` 方法：

- 为每个成员变量定义了 `set` 方法用于设置值，`get` 方法用于获取值。

3. `toString()` 方法：

- 该方法返回包含教师基本信息的字符串格式，用于展示所有属性。

4. 类内 `main()` 方法：

- 在 `Teacher` 类内实现了 `main()` 方法，用于创建 `Teacher` 对象并调用相应的 `set` 和 `get` 方法展示效果。

5. 类外 `main()` 方法：

- 在 `Main` 类中定义了 `main()` 方法，用于在 `Teacher` 类外创建 `Teacher` 对象并调用相关方法展示效果。

(3.6).当设计一个类的时候, 有哪些注意事项? 请用自己的话进行阐述 (300-500 字), 要求重点突出、条理清楚, 可读性强。(5 分)

1. 明确职责：每个类应该有一个清晰的职责（Single Responsibility Principle）。避免将过多的功能聚集在一个类中，这样不仅使类的理解变得困难，还会增加后期修改的风险。遵循单一职责原则，可以提高代码的可读性和可维护性。

2. 使用合适的命名：类名应清晰、简洁，并能够准确描述类的功能。通常使用名词或名词短语，例如 Customer 或 OrderProcessor。良好的命名有助于他人快速理解类的目的。

3. 封装：合理使用访问修饰符（如 private, protected, public）来隐藏类的内部实现细节。通过提供公共方法（getter 和 setter）来访问私有属性，增强数据的安全性和类的可维护性。

4. 设计构造函数：根据需要设计适当的构造函数，确保对象在创建时处于有效状态。可以考虑提供多个构造函数以支持不同的初始化方式，或者使用建造者模式（Builder Pattern）来处理复杂对象的创建。

5. 考虑可扩展性：设计时要考虑将来可能的扩展。使用接口和抽象类可以帮助实现多态性，方便后续功能的扩展。此外，避免使用硬编码的值，使用常量或配置文件来提高灵活性。

6. 代码复用：通过继承和组合来实现代码复用。合理使用继承可以减少重复代码，但要避免过度使用，导致类层次结构复杂化。组合通常更灵活，可以更好地实现功能的组合。

7. 编写文档：为类和方法添加适当的注释，描述它们的功能、参数和返回值。这不仅有助于自己在未来维护代码，也方便其他开发者理解你的代码。

8. 测试：设计时考虑单元测试，确保类的功能易于测试。编写清晰的接口和方法，使得

测试变得简单。使用测试驱动开发（TDD）方法，可以帮助确保代码的质量和可靠性。

+++++

其他（例如感想、建议等等）。

Java 的多线程机制非常强大，能够有效地利用系统资源，提高程序的并发性。通过 **Thread** 类和 **Runnable** 接口，可以轻松创建和管理多个线程。线程间的同步和通信机制使得在共享资源时能避免数据不一致的问题。然而，多线程编程也带来了复杂性，需要谨慎处理死锁和竞争条件等问题。因此，掌握 **Java** 多线程是提高应用性能和响应能力的关键。

指导教师批阅意见：
成绩评定：
指导教师签字：
2024 年    月    日
备注：

- 注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。
- 2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。