**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4：I/O、GUI和网络编程**

**学院： 计算机与软件学院**

**专业： 数计班**

**指导教师：潘微科**

**报告人： 詹耿羽 学号： 2023193026 班级： 数计**

**实验时间：2024年11月29日（周五）-2024年12月18日（周三）**

**实验报告提交时间： 2024/12/13**

**教务部制**

|  |
| --- |
| **实验目的与要求：**  **实验目的：**掌握I/O程序设计，能够读写本地文件等，初步掌握图形界面程序设计，掌握网络通信协议及相关程序设计。  **实验要求：**  **Part 1（25分）**  (1.1). 数据解析和统计。<https://snap.stanford.edu/data/web-Amazon.html> 网站上有很多Amazon的数据集供研究人员下载使用。本次实验使用Watches.txt.gz数据集，请下载后解压。格式说明请看网页上的“Data Format”部分。在报告中附上程序截图、运行结果（如每个输出文件前10行的截图等）和简要文字说明。  （i）使用Java语言读取解压后的文件（Watches.txt），并得到以下文件（10分）：  **review.txt**：每行2列，以分号作为分隔符，第1列是userID，第2列是productID，表示(user, product)二元组。该文件中不同行之间的顺序，按照userID从小到大排列，当userID相同时按照productID从小到大排列。  注：删除ID为unknown的记录；排序时按照字符串顺序。  （ii）使用Java语言根据review.txt进行计算，并得到以下文件（15分）：  **productNeighborhood.txt**：每行4列，以分号作为分隔符，第1列是productID，第2-4列是与该product最相似（根据相似度值）的3个product的productID，按相似度值从大到小排列（当相似度相同时，按照productID的字符串顺序从小到大排序），其中相似度是通过review.txt中的两列的信息计算得到的Jaccard index值。该文件中不同行之间的顺序，按照第1列的productID从小到大排列。关于相似度的计算，要求使用多线程实现（5分）。  注：计算两个商品（product）之间的相似度（即Jaccard index）时，根据这两个商品所关联的用户的集合来计算——集合的交集的大小除以集合的并集的大小。  **Part 2（25分）**  (2.1).使用JLabel、JTextArea、JButton等控件实现句子中英互译的demo，该demo包含两个文本框，第一个文本框用于输入中文句子或显示第二个文本框中的英文句子的中文翻译，第二个文本框用于输入英文句子或显示第一个文本框中的中文句子的英文翻译。每个文本框下方各有一个按钮，第一个按钮的名称是“中译英”，第二个按钮的名称是“英译中”，点击按钮表示将该文本框中的内容翻译成另一种语言。要求使用以下三种方式：  A、使用自己事先准备好的中英文翻译（此部分占5分）；  B、使用百度翻译API、有道翻译API或其他API中的一个API（此部分占5分）；  C、使用腾讯混元大模型、百度文心一言大模型API、阿里通义千问大模型API或其他大模型API中的两个API（此部分占10分）；  要求使用以下两个例句：  建校41年，深圳大学秉承“自立、自律、自强”的校训，紧随特区，锐意改革、快速发展，为特区发展和国家现代化建设做出了重要贡献。  Sticking to the motto of “self-reliance, self-discipline, self-improvement”, the University is dedicated to serving the Shenzhen Special Economic Zone (SEZ), demonstrating China’s reform and opening up and pioneering change in higher education.  要求使用图形用户界面，界面美观、交互友好。在报告中附上程序截图、运行结果和详细的文字说明。（5分）  **Part 3（30分）**  (3.1). 利用套接字连接（TCP）编写程序，该程序包括三个客户端（ClientA、ClientB、ClientC）和一个服务端（ServerS），三个客户端通过服务端作为桥梁实现相互间的文字交流，例如，ClientA先发信息给ServerS，然后ServerC再将收到的信息转发给ClientB和ClientC。在报告中附上示意图（三个客户端+一个服务端）、程序截图、完整的运行结果和简要文字说明。（20分）  (3.2). 利用数据报通信（UDP）实现题(1)中的要求。（10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12月18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). 数据解析和统计。<https://snap.stanford.edu/data/web-Amazon.html> 网站上有很多Amazon的数据集供研究人员下载使用。本次实验使用Watches.txt.gz数据集，请下载后解压。格式说明请看网页上的“Data Format”部分。在报告中附上程序截图、运行结果（如每个输出文件前10行的截图等）和简要文字说明。  （i）使用Java语言读取解压后的文件（Watches.txt），并得到以下文件（10分）：  **review.txt**：每行2列，以分号作为分隔符，第1列是userID，第2列是productID，表示(user, product)二元组。该文件中不同行之间的顺序，按照userID从小到大排列，当userID相同时按照productID从小到大排列。  注：删除ID为unknown的记录；排序时按照字符串顺序。  （ii）使用Java语言根据review.txt进行计算，并得到以下文件（15分）：  **productNeighborhood.txt**：每行4列，以分号作为分隔符，第1列是productID，第2-4列是与该product最相似（根据相似度值）的3个product的productID，按相似度值从大到小排列（当相似度相同时，按照productID的字符串顺序从小到大排序），其中相似度是通过review.txt中的两列的信息计算得到的Jaccard index值。该文件中不同行之间的顺序，按照第1列的productID从小到大排列。关于相似度的计算，要求使用多线程实现（5分）。  注：计算两个商品（product）之间的相似度（即Jaccard index）时，根据这两个商品所关联的用户的集合来计算——集合的交集的大小除以集合的并集的大小。  （i）  ·格式说明      ·程序截图    ·完整代码   1. package 课程作业; 2. import java.io.\*; 3. import java.util.\*; 4. import java.util.stream.Collectors; 5. public class WatchesDataProcessor { 6. public static void main(String[] args) { 7. String inputFilePath = "C:\\Users\\詹耿羽\\Desktop\\Watches.txt\\Watches.txt"; 8. String outputFilePath = "C:\\Users\\詹耿羽\\Desktop\\Watches.txt\\review.txt"; 9. try (BufferedReader reader = new BufferedReader(new FileReader(inputFilePath)); 10. BufferedWriter writer = new BufferedWriter(new FileWriter(outputFilePath))) { 11. *// 用于存储 (userID, productID) 的列表* 12. List<String[]> dataList = new ArrayList<>(); 13. *// 每个review块的临时存储变量* 14. String line; 15. String userId = null; 16. String productId = null; 17. *// 按行读取文件* 18. while ((line = reader.readLine()) != null) { 19. line = line.trim(); *// 去除行首尾空格* 20. *// 如果是productId行* 21. if (line.startsWith("product/productId:")) { 22. productId = line.split("productId:")[1].trim(); *// 提取productId* 23. } 24. *// 如果是userId行* 25. else if (line.startsWith("review/userId:")) { 26. userId = line.split("userId:")[1].trim(); *// 提取userId* 27. } 28. *// 如果是空行且userId和productId已经读取完成* 29. else if (line.isEmpty() && userId != null && productId != null) { 30. *// 过滤掉unknown的记录* 31. if (!"unknown".equals(userId) && !"unknown".equals(productId)) { 32. dataList.add(new String[]{userId, productId}); *// 加入到列表中* 33. } 34. *// 重置临时变量以处理下一个块* 35. userId = null; 36. productId = null; 37. } 38. } 39. *// 按userID和productID排序* 40. dataList = dataList.stream() 41. .sorted((a, b) -> { 42. int userCompare = a[0].compareTo(b[0]); *// 比较userID* 43. return userCompare != 0 ? userCompare : a[1].compareTo(b[1]); *// 若相等比较productID* 44. }) 45. .collect(Collectors.toList()); 46. *// 写入review.txt文件* 47. for (String[] data : dataList) { 48. writer.write(data[0] + ";" + data[1]); *// 以分号分隔* 49. writer.newLine(); *// 换行* 50. } 51. System.out.println("数据处理完成，结果已写入 " + outputFilePath); 52. } catch (IOException e) { 53. e.printStackTrace(); *// 捕获并打印IO异常* 54. } 55. } 56. }   ·文字说明  1.导入必要的库  java.io.\*: 用于文件读写操作。  java.util.\*: 提供数据结构如 List 和 ArrayList。  java.util.stream.Collectors: 用于处理数据流（排序和收集）。  2.变量初始化  inputFilePath 和 outputFilePath 分别指定输入和输出文件路径。  BufferedReader 用于高效读取文本文件，BufferedWriter 用于高效写入。  3.读取和解析文件  使用 while 循环逐行读取文件。  根据行的前缀判断字段类型：  product/productId: 提取 productId。  review/userId: 提取 userId。  遇到空行时，判断一个记录是否完整，并排除 unknown 记录。  4.排序逻辑  使用 stream 对 dataList 进行排序：  首先按 userId 排序。  若 userId 相等，则按 productId 排序。  5.写入输出文件  遍历排序后的 dataList，按指定格式（用分号分隔）写入文件。  6.异常处理  捕获 IOException 异常以确保程序健壮性，并打印错误信息。  注：运行结果已经在附件提交。  （ii）  ·程序截图    ·完整代码   1. package 课程作业; 2. import java.io.\*; 3. import java.util.\*; 4. import java.util.concurrent.\*; 5. import java.util.stream.Collectors; 6. public class ProductNeighborhood { 7. public static void main(String[] args) { 8. String inputFilePath = "C:\\Users\\詹耿羽\\Desktop\\Watches.txt\\Review.txt"; 9. String outputFilePath = "C:\\Users\\詹耿羽\\Desktop\\Watches.txt\\productNeighborhood.txt"; 10. try { 11. *// 读取review.txt并创建商品-用户映射* 12. Map<String, Set<String>> productToUsers = new HashMap<>(); 13. try (BufferedReader reader = new BufferedReader(new FileReader(inputFilePath))) { 14. String line; 15. while ((line = reader.readLine()) != null) { 16. String[] parts = line.split(";"); 17. if (parts.length == 2) { 18. String userId = parts[0].trim(); *// 提取用户ID* 19. String productId = parts[1].trim(); *// 提取商品ID* 20. productToUsers.computeIfAbsent(productId, k -> new HashSet<>()).add(userId); *// 构建商品-用户映射* 21. } 22. } 23. } 24. *// 多线程计算相似度* 25. List<String> productIds = new ArrayList<>(productToUsers.keySet()); 26. Map<String, List<String>> productNeighbors = computeSimilarities(productIds, productToUsers); 27. *// 写入productNeighborhood.txt文件* 28. try (BufferedWriter writer = new BufferedWriter(new FileWriter(outputFilePath))) { 29. for (Map.Entry<String, List<String>> entry : productNeighbors.entrySet()) { 30. String productId = entry.getKey(); 31. List<String> neighbors = entry.getValue(); 32. writer.write(productId + ";" + String.join(";", neighbors)); *// 按要求格式写入* 33. writer.newLine(); 34. } 35. } 36. System.out.println("数据处理完成，结果已写入 " + outputFilePath); 37. } catch (IOException e) { 38. e.printStackTrace(); *// 捕获IO异常并打印* 39. } 40. } 41. */\*\** 42. \* 多线程计算所有商品的相似度并获取最相似的3个商品 43. \*/ 44. private static Map<String, List<String>> computeSimilarities(List<String> productIds, Map<String, Set<String>> productToUsers) { 45. Map<String, List<String>> result = new ConcurrentHashMap<>(); 46. ExecutorService executor = Executors.newFixedThreadPool(Runtime.getRuntime().availableProcessors()); *// 创建线程池* 47. CountDownLatch latch = new CountDownLatch(productIds.size()); *// 用于线程同步* 48. for (String productId : productIds) { 49. executor.submit(() -> { *// 每个商品提交一个任务* 50. try { 51. *// 当前商品与所有其他商品的相似度计算* 52. List<String[]> similarities = new ArrayList<>(); 53. Set<String> usersA = productToUsers.get(productId); 54. for (String otherProductId : productIds) { 55. if (!productId.equals(otherProductId)) { 56. Set<String> usersB = productToUsers.get(otherProductId); 57. *// 计算Jaccard相似度* 58. double jaccard = calculateJaccard(usersA, usersB); 59. similarities.add(new String[]{otherProductId, String.valueOf(jaccard)}); 60. } 61. } 62. *// 排序并选取相似度最高的3个商品* 63. List<String> top3 = similarities.stream() 64. .sorted((a, b) -> { 65. double simA = Double.parseDouble(a[1]); 66. double simB = Double.parseDouble(b[1]); 67. if (simA != simB) { 68. return Double.compare(simB, simA); *// 按相似度从大到小排序* 69. } else { 70. return a[0].compareTo(b[0]); *// 按商品ID的字典序* 71. } 72. }) 73. .limit(3) 74. .map(a -> a[0]) 75. .collect(Collectors.toList()); 76. result.put(productId, top3); *// 将计算结果存入结果映射* 77. } finally { 78. latch.countDown(); *// 任务完成，计数器减1* 79. } 80. }); 81. } 82. try { 83. latch.await(); *// 等待所有任务完成* 84. } catch (InterruptedException e) { 85. Thread.currentThread().interrupt(); *// 恢复线程中断状态* 86. System.err.println("线程被中断：" + e.getMessage()); 87. } 88. executor.shutdown(); *// 关闭线程池* 89. *// 按商品ID排序结果* 90. return result.entrySet().stream() 91. .sorted(Map.Entry.comparingByKey()) 92. .collect(Collectors.toMap( 93. Map.Entry::getKey, 94. Map.Entry::getValue, 95. (a, b) -> a, 96. LinkedHashMap::new 97. )); 98. } 99. */\*\** 100. \* 计算两个集合之间的Jaccard相似度 101. \*/ 102. private static double calculateJaccard(Set<String> setA, Set<String> setB) { 103. Set<String> intersection = new HashSet<>(setA); *// 交集* 104. intersection.retainAll(setB); 105. Set<String> union = new HashSet<>(setA); *// 并集* 106. union.addAll(setB); 107. return union.isEmpty() ? 0.0 : (double) intersection.size() / union.size(); 108. } 109. }   ·文字说明  1. 主方法  -文件读取  使用 BufferedReader 按行读取 review.txt 文件。  提取 userId 和 productId 信息，建立商品与用户的映射 productToUsers。  -计算相似度  提取所有商品的 ID，调用 computeSimilarities 方法，计算每个商品与其他商品的相似度。  -写入输出  将计算结果按照指定格式写入 productNeighborhood.txt 文件。  2. computeSimilarities 方法  -多线程实现  使用 ExecutorService 创建线程池，分配任务。  每个线程计算一个商品与其他商品的相似度。  -相似度计算逻辑  对每对商品，计算 Jaccard 相似度。  排序规则：  优先按相似度从高到低排序。  若相似度相同，按商品 ID 的字典序排序。  每个商品选取相似度最高的 3 个商品。  -线程同步  使用 CountDownLatch 确保所有线程任务完成后才返回结果。  3. calculateJaccard 方法  -计算公式  使用 retainAll 方法求交集。  使用 addAll 方法求并集。  返回交集大小除以并集大小。  ·运行结果  运行结果已经在附件给出。  **Part 2（25分）**  (2.1).使用JLabel、JTextArea、JButton等控件实现句子中英互译的demo，该demo包含两个文本框，第一个文本框用于输入中文句子或显示第二个文本框中的英文句子的中文翻译，第二个文本框用于输入英文句子或显示第一个文本框中的中文句子的英文翻译。每个文本框下方各有一个按钮，第一个按钮的名称是“中译英”，第二个按钮的名称是“英译中”，点击按钮表示将该文本框中的内容翻译成另一种语言。要求使用以下三种方式：  A、使用自己事先准备好的中英文翻译（此部分占5分）；  B、使用百度翻译API、有道翻译API或其他API中的一个API（此部分占5分）；  C、使用腾讯混元大模型、百度文心一言大模型API、阿里通义千问大模型API或其他大模型API中的两个API（此部分占10分）；  要求使用以下两个例句：  建校41年，深圳大学秉承“自立、自律、自强”的校训，紧随特区，锐意改革、快速发展，为特区发展和国家现代化建设做出了重要贡献。  Sticking to the motto of “self-reliance, self-discipline, self-improvement”, the University is dedicated to serving the Shenzhen Special Economic Zone (SEZ), demonstrating China’s reform and opening up and pioneering change in higher education.  要求使用图形用户界面，界面美观、交互友好。在报告中附上程序截图、运行结果和详细的文字说明。（5分）  A题：  ·程序截图    ·完整代码   1. package 课程作业; 2. import javax.swing.\*; 3. import java.awt.\*; 4. import java.awt.event.ActionEvent; 5. import java.awt.event.ActionListener; 6. import java.util.HashMap; 7. import java.util.Map; 8. public class TranslationDemo2 { 9. public static void main(String[] args) { 10. *// 创建主窗口* 11. JFrame frame = new JFrame("中英互译 Demo"); 12. frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); 13. frame.setSize(600, 400); 14. frame.setLayout(new GridLayout(2, 1)); 15. *// 定义中英文翻译映射* 16. Map<String, String> translations = new HashMap<>(); 17. translations.put("建校41年，深圳大学秉承“自立、自律、自强”的校训，紧随特区，锐意改革、快速发展，为特区发展和国家现代化建设做出了重要贡献。", 18. "Sticking to the motto of “self-reliance, self-discipline, self-improvement”, the University is dedicated to serving the Shenzhen Special Economic Zone (SEZ), demonstrating China’s reform and opening up and pioneering change in higher education."); 19. translations.put("Sticking to the motto of “self-reliance, self-discipline, self-improvement”, the University is dedicated to serving the Shenzhen Special Economic Zone (SEZ), demonstrating China’s reform and opening up and pioneering change in higher education.", 20. "建校41年，深圳大学秉承“自立、自律、自强”的校训，紧随特区，锐意改革、快速发展，为特区发展和国家现代化建设做出了重要贡献。"); 21. *// 第一部分：中文文本框和按钮* 22. JPanel chinesePanel = new JPanel(new BorderLayout()); 23. JLabel chineseLabel = new JLabel("中文"); 24. JTextArea chineseTextArea = new JTextArea(); 25. JButton translateToEnglishButton = new JButton("中译英"); 26. chinesePanel.add(chineseLabel, BorderLayout.NORTH); 27. chinesePanel.add(new JScrollPane(chineseTextArea), BorderLayout.CENTER); 28. chinesePanel.add(translateToEnglishButton, BorderLayout.SOUTH); 29. *// 第二部分：英文文本框和按钮* 30. JPanel englishPanel = new JPanel(new BorderLayout()); 31. JLabel englishLabel = new JLabel("English"); 32. JTextArea englishTextArea = new JTextArea(); 33. JButton translateToChineseButton = new JButton("英译中"); 34. englishPanel.add(englishLabel, BorderLayout.NORTH); 35. englishPanel.add(new JScrollPane(englishTextArea), BorderLayout.CENTER); 36. englishPanel.add(translateToChineseButton, BorderLayout.SOUTH); 37. *// 将两部分添加到主窗口* 38. frame.add(chinesePanel); 39. frame.add(englishPanel); 40. *// 按钮事件：中译英* 41. translateToEnglishButton.addActionListener(new ActionListener() { 42. @Override 43. public void actionPerformed(ActionEvent e) { 44. String chineseText = chineseTextArea.getText().trim(); 45. if (translations.containsKey(chineseText)) { 46. englishTextArea.setText(translations.get(chineseText)); 47. } else { 48. englishTextArea.setText("没有找到对应的翻译。"); 49. } 50. } 51. }); 52. *// 按钮事件：英译中* 53. translateToChineseButton.addActionListener(new ActionListener() { 54. @Override 55. public void actionPerformed(ActionEvent e) { 56. String englishText = englishTextArea.getText().trim(); 57. if (translations.containsKey(englishText)) { 58. chineseTextArea.setText(translations.get(englishText)); 59. } else { 60. chineseTextArea.setText("没有找到对应的翻译。"); 61. } 62. } 63. }); 64. *// 显示窗口* 65. frame.setVisible(true); 66. } 67. }   ·运行结果    英译中：      中译英：      ·文字说明  1. 主程序入口  public static void main(String[] args) {  这是程序的入口，所有的逻辑都在 main 方法中。  通过创建窗口、布局组件和绑定事件，完成整个应用的初始化。  2. 创建主窗口  JFrame frame = new JFrame("中英互译 Demo");  frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  frame.setSize(600, 400);  frame.setLayout(new GridLayout(2, 1));  JFrame 是主窗口，标题为“中英互译 Demo”。  设置了窗口关闭操作为 EXIT\_ON\_CLOSE，即点击关闭按钮时程序退出。  使用了 GridLayout(2, 1) 布局，将窗口分为上下两部分。  3. 翻译映射定义  Map<String, String> translations = new HashMap<>();  使用 HashMap 存储中英文对照的翻译数据，模拟一个简单的翻译字典。  示例中包含两个翻译：  中文到英文。  英文到中文。  4. 第一部分：中文区域  JPanel chinesePanel = new JPanel(new BorderLayout());  JLabel chineseLabel = new JLabel("中文");  JTextArea chineseTextArea = new JTextArea();  JButton translateToEnglishButton = new JButton("中译英");  JPanel 使用 BorderLayout 布局。  添加组件：  JLabel：显示“中文”标签。  JTextArea：供用户输入中文文本。  JButton：一个按钮，点击后会触发“中译英”操作。  5. 第二部分：英文区域  JPanel englishPanel = new JPanel(new BorderLayout());  JLabel englishLabel = new JLabel("English");  JTextArea englishTextArea = new JTextArea();  JButton translateToChineseButton = new JButton("英译中");  同样使用了 BorderLayout 布局。  添加组件：  JLabel：显示“English”标签。  JTextArea：供用户输入英文文本。  JButton：一个按钮，点击后会触发“英译中”操作。  6. 主窗口组件添加  frame.add(chinesePanel);  frame.add(englishPanel);  将中文区域和英文区域的面板添加到主窗口中，分别占据上下部分。  7. 按钮事件：中译英  translateToEnglishButton.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  String chineseText = chineseTextArea.getText().trim();  if (translations.containsKey(chineseText)) {  englishTextArea.setText(translations.get(chineseText));  } else {  englishTextArea.setText("没有找到对应的翻译。");  }  }  });  触发方式：点击“中译英”按钮。  逻辑：  获取中文输入框的内容并去掉多余空格。  在 translations 字典中查找对应的英文翻译：  如果找到，显示在英文文本框中。  如果找不到，显示提示“没有找到对应的翻译”。  8. 按钮事件：英译中  translateToChineseButton.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  String englishText = englishTextArea.getText().trim();  if (translations.containsKey(englishText)) {  chineseTextArea.setText(translations.get(englishText));  } else {  chineseTextArea.setText("没有找到对应的翻译。");  }  }  });  触发方式：点击“英译中”按钮。  逻辑：  获取英文输入框的内容并去掉多余空格。  在 translations 字典中查找对应的中文翻译：  如果找到，显示在中文文本框中。  如果找不到，显示提示“没有找到对应的翻译”。  9. 显示窗口  frame.setVisible(true);  调用 setVisible(true) 将窗口显示给用户。  B题：  ·api申请：  首先需要上百度翻译，申请一个api。    ·环境配置  需要先配好maven环境。（相关视频在b站）        导入相关的依赖：    ·程序截图    ·完整代码   1. package 课程作业; 2. import javax.swing.\*; 3. import java.awt.\*; 4. import java.awt.event.ActionEvent; 5. import java.awt.event.ActionListener; 6. import java.io.BufferedReader; 7. import java.io.InputStreamReader; 8. import java.net.HttpURLConnection; 9. import java.net.URL; 10. import java.security.MessageDigest; 11. import java.util.UUID; 12. public class TranslationDemo { 13. private static final String APP\_ID = "20241204002219316"; *// 百度翻译APP ID* 14. private static final String SECURITY\_KEY = "rFvBWS5oHudr9677kddV"; *// 百度翻译密钥* 15. private static final String API\_URL = "https://fanyi-api.baidu.com/api/trans/vip/translate"; 16. public static void main(String[] args) { 17. *// 创建 JFrame 窗口* 18. JFrame frame = new JFrame("中英互译 Demo"); 19. frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); 20. frame.setSize(600, 400); 21. frame.setLayout(new GridLayout(3, 1)); 22. *// 第一个文本框和按钮* 23. JTextArea textArea1 = new JTextArea(5, 20); 24. JButton button1 = new JButton("中译英"); 25. JPanel panel1 = new JPanel(new BorderLayout()); 26. panel1.add(new JLabel("中文句子或翻译结果："), BorderLayout.NORTH); 27. panel1.add(new JScrollPane(textArea1), BorderLayout.CENTER); 28. panel1.add(button1, BorderLayout.SOUTH); 29. *// 第二个文本框和按钮* 30. JTextArea textArea2 = new JTextArea(5, 20); 31. JButton button2 = new JButton("英译中"); 32. JPanel panel2 = new JPanel(new BorderLayout()); 33. panel2.add(new JLabel("英文句子或翻译结果："), BorderLayout.NORTH); 34. panel2.add(new JScrollPane(textArea2), BorderLayout.CENTER); 35. panel2.add(button2, BorderLayout.SOUTH); 36. *// 添加组件到 JFrame* 37. frame.add(panel1); 38. frame.add(panel2); 39. *// 提示区域* 40. JTextArea infoArea = new JTextArea(3, 50); 41. infoArea.setEditable(false); 42. infoArea.setLineWrap(true); 43. infoArea.setWrapStyleWord(true); 44. infoArea.setBackground(new Color(240, 240, 240)); 45. frame.add(new JScrollPane(infoArea)); 46. *// 按钮监听事件* 47. button1.addActionListener(new ActionListener() { 48. @Override 49. public void actionPerformed(ActionEvent e) { 50. String chineseText = textArea1.getText().trim(); 51. if (chineseText.isEmpty()) { 52. infoArea.setText("请输入中文内容进行翻译！"); 53. return; 54. } 55. String translatedText = translate(chineseText, "zh", "en"); 56. textArea2.setText(translatedText); 57. } 58. }); 59. button2.addActionListener(new ActionListener() { 60. @Override 61. public void actionPerformed(ActionEvent e) { 62. String englishText = textArea2.getText().trim(); 63. if (englishText.isEmpty()) { 64. infoArea.setText("请输入英文内容进行翻译！"); 65. return; 66. } 67. String translatedText = translate(englishText, "en", "zh"); 68. textArea1.setText(translatedText); 69. } 70. }); 71. *// 显示窗口* 72. frame.setVisible(true); 73. } 74. */\*\** 75. \* 调用百度翻译API进行翻译 76. \* 77. \* @param query    待翻译文本 78. \* @param fromLang 源语言 79. \* @param toLang   目标语言 80. \* @return 翻译结果 81. \*/ 82. public static String translate(String query, String fromLang, String toLang) { 83. try { 84. String salt = UUID.randomUUID().toString().replaceAll("-", ""); 85. String sign = md5(APP\_ID + query + salt + SECURITY\_KEY); 86. String urlStr = API\_URL + "?q=" + java.net.URLEncoder.encode(query, "UTF-8") + 87. "&from=" + fromLang + "&to=" + toLang + 88. "&appid=" + APP\_ID + "&salt=" + salt + "&sign=" + sign; 89. URL url = new URL(urlStr); 90. HttpURLConnection connection = (HttpURLConnection) url.openConnection(); 91. connection.setRequestMethod("GET"); 92. connection.setConnectTimeout(5000); 93. connection.setReadTimeout(5000); 94. BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream(), "UTF-8")); 95. StringBuilder result = new StringBuilder(); 96. String line; 97. while ((line = in.readLine()) != null) { 98. result.append(line); 99. } 100. in.close(); 101. *// 解析返回的 JSON，获取翻译结果* 102. String json = result.toString(); 103. int start = json.indexOf("\"dst\":\"") + 7; 104. int end = json.indexOf("\"", start); 105. String translatedText = json.substring(start, end); 106. *// 将 Unicode 转义字符解码为可读文本* 107. return decodeUnicode(translatedText); 108. } catch (Exception e) { 109. e.printStackTrace(); 110. return "翻译失败，请检查网络连接或输入内容！"; 111. } 112. } 113. */\*\** 114. \* 解码 Unicode 转义字符 115. \* 116. \* @param unicodeStr Unicode 转义字符串 117. \* @return 解码后的字符串 118. \*/ 119. private static String decodeUnicode(String unicodeStr) { 120. StringBuilder out = new StringBuilder(); 121. int len = unicodeStr.length(); 122. for (int i = 0; i < len; i++) { 123. char c = unicodeStr.charAt(i); 124. if (c == '\\' && i + 1 < len && unicodeStr.charAt(i + 1) == 'u') { 125. String hex = unicodeStr.substring(i + 2, i + 6); 126. out.append((char) Integer.parseInt(hex, 16)); 127. i += 5; *// 跳过 \\uXXXX* 128. } else { 129. out.append(c); 130. } 131. } 132. return out.toString(); 133. } 134. */\*\** 135. \* 生成 MD5 加密 136. \* 137. \* @param text 输入文本 138. \* @return 加密后的字符串 139. \*/ 140. public static String md5(String text) { 141. try { 142. MessageDigest md = MessageDigest.getInstance("MD5"); 143. byte[] array = md.digest(text.getBytes("UTF-8")); 144. StringBuilder sb = new StringBuilder(); 145. for (byte b : array) { 146. sb.append(String.format("%02x", b)); 147. } 148. return sb.toString(); 149. } catch (Exception e) { 150. throw new RuntimeException(e); 151. } 152. } 153. }   ·运行结果    中译英      英译中      题目例子翻译：  中译英    翻译为：  For 41 years since its establishment, Shenzhen University has adhered to the motto of self-reliance, self-discipline, and self-improvement, closely following the special zone, and striving for reform and rapid development, making important contributions to the development of the special zone and the modernization of the country.  英译中    翻译为  学校秉承“自力更生、自律自强”的校训，致力于服务深圳经济特区，展示中国改革开放和高等教育的开创性变革。  ·文字说明  1. 主程序入口  public static void main(String[] args) {  这是程序的入口，负责初始化 GUI 窗口，添加组件，并为按钮设置监听事件。  程序启动后，用户可以通过界面进行翻译操作。  2. GUI 界面设计  使用 JFrame 作为主窗口，设置标题、关闭操作和布局。  JFrame frame = new JFrame("中英互译 Demo");  frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  frame.setSize(600, 400);  frame.setLayout(new GridLayout(3, 1));  窗口被分为三个部分，每部分分别对应：  中文输入区。  英文输入区。  信息提示区。  (1) 中文输入区  JTextArea textArea1 = new JTextArea(5, 20);  JButton button1 = new JButton("中译英");  JPanel panel1 = new JPanel(new BorderLayout());  panel1.add(new JLabel("中文句子或翻译结果："), BorderLayout.NORTH);  panel1.add(new JScrollPane(textArea1), BorderLayout.CENTER);  panel1.add(button1, BorderLayout.SOUTH);  包含一个 JTextArea 文本框，用于用户输入中文。  按钮 button1 用于触发“中译英”翻译。  (2) 英文输入区  JTextArea textArea2 = new JTextArea(5, 20);  JButton button2 = new JButton("英译中");  JPanel panel2 = new JPanel(new BorderLayout());  panel2.add(new JLabel("英文句子或翻译结果："), BorderLayout.NORTH);  panel2.add(new JScrollPane(textArea2), BorderLayout.CENTER);  panel2.add(button2, BorderLayout.SOUTH);  包含一个 JTextArea 文本框，用于用户输入英文。  按钮 button2 用于触发“英译中”翻译。  (3) 信息提示区  JTextArea infoArea = new JTextArea(3, 50);  infoArea.setEditable(false);  infoArea.setLineWrap(true);  infoArea.setWrapStyleWord(true);  infoArea.setBackground(new Color(240, 240, 240));  frame.add(new JScrollPane(infoArea));  一个只读的文本框，用于显示提示信息（如输入错误、网络异常等）。  3. 按钮监听事件  **中译英按钮事件**  button1.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  String chineseText = textArea1.getText().trim();  if (chineseText.isEmpty()) {  infoArea.setText("请输入中文内容进行翻译！");  return;  }  String translatedText = translate(chineseText, "zh", "en");  textArea2.setText(translatedText);  }  });  获取用户输入的中文文本。  如果文本为空，显示错误提示。  调用 translate 方法完成翻译，并将结果显示在英文文本框中。  **英译中按钮事件**  button2.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  String englishText = textArea2.getText().trim();  if (englishText.isEmpty()) {  infoArea.setText("请输入英文内容进行翻译！");  return;  }  String translatedText = translate(englishText, "en", "zh");  textArea1.setText(translatedText);  }  });  类似逻辑：获取英文输入内容，调用 translate 方法翻译为中文，并显示结果。  4. 百度翻译 API 调用  (1) 翻译主函数  public static String translate(String query, String fromLang, String toLang) {  输入参数：  query：待翻译文本。  fromLang：源语言（如 "zh"）。  toLang：目标语言（如 "en"）。  返回值：翻译结果字符串。  (2) 翻译请求构造  String urlStr = API\_URL + "?q=" + java.net.URLEncoder.encode(query, "UTF-8") +  "&from=" + fromLang + "&to=" + toLang +  "&appid=" + APP\_ID + "&salt=" + salt + "&sign=" + sign;  拼接百度翻译 API 的 URL，包括以下参数：  q：待翻译文本（需 UTF-8 编码）。  from 和 to：源语言和目标语言。  appid：应用 ID。  salt：随机字符串。  sign：通过 MD5 生成的签名，确保安全性。  (3) HTTP 请求  HttpURLConnection connection = (HttpURLConnection) url.openConnection();  connection.setRequestMethod("GET");  connection.setConnectTimeout(5000);  connection.setReadTimeout(5000);  使用 HTTP GET 方法发送请求。  设置超时时间，防止卡死。  (4) 返回结果解析  int start = json.indexOf("\"dst\":\"") + 7;  int end = json.indexOf("\"", start);  String translatedText = json.substring(start, end);  从返回的 JSON 数据中提取翻译结果（"dst" 字段）。  5. 工具函数  (1) MD5 签名生成  public static String md5(String text) {  MessageDigest md = MessageDigest.getInstance("MD5");  byte[] array = md.digest(text.getBytes("UTF-8"));  ...  return sb.toString();  }  将 APP\_ID + query + salt + SECURITY\_KEY 生成 MD5 签名，用于 API 安全认证。  (2) Unicode 转义解码  private static String decodeUnicode(String unicodeStr) {  ...  return out.toString();  }  将翻译结果中可能出现的 Unicode 转义字符（如 \u4F60\u597D）解码为可读文本。  C题：  ·api的申请  硅基api的申请      百度api的申请      ·程序截图    ·完整代码   1. package 课程作业; 2. import javax.swing.\*; 3. import java.awt.\*; 4. import java.awt.event.ActionEvent; 5. import java.awt.event.ActionListener; 6. import java.io.\*; 7. import java.net.HttpURLConnection; 8. import java.net.URL; 9. import org.json.JSONArray; 10. import org.json.JSONObject; 11. public class TranslationDemo1 { 12. *// 百度 API 信息* 13. private static final String BAIDU\_ACCESS\_TOKEN = "24.6b0025976054ebc04bd8b3c628503d98.2592000.1736142581.282335-116571818"; 14. private static final String BAIDU\_CHAT\_URL = "https://aip.baidubce.com/rpc/2.0/ai\_custom/v1/wenxinworkshop/chat/yi\_34b\_chat"; 15. *// 硅基 API 信息* 16. private static final String SILICON\_API\_KEY = "sk-qlpswutpukmnzabgilaetvhjzmuapvzxpflynyauuncwgklt"; 17. private static final String SILICON\_CHAT\_URL = "https://api.siliconflow.cn/v1/chat/completions"; 18. public static void main(String[] args) { 19. *// 创建主窗口* 20. JFrame frame = new JFrame("中英互译 Demo"); 21. frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); 22. frame.setSize(600, 500); 23. frame.setLayout(new BorderLayout(10, 10)); 24. *// 创建主面板，使用FlowLayout来优化控件位置* 25. JPanel panel = new JPanel(); 26. panel.setLayout(new FlowLayout(FlowLayout.LEFT, 10, 10)); 27. *// API选择* 28. JLabel apiLabel = new JLabel("选择API："); 29. String[] apiOptions = {"百度 API", "硅基 API"}; 30. JComboBox<String> apiComboBox = new JComboBox<>(apiOptions); 31. apiComboBox.setPreferredSize(new Dimension(150, 30)); 32. *// 中文输入/翻译文本框及按钮* 33. JLabel chineseLabel = new JLabel("中英文输入/翻译结果："); 34. JTextArea chineseTextArea = new JTextArea(5, 20); 35. chineseTextArea.setLineWrap(true); 36. chineseTextArea.setWrapStyleWord(true); 37. JScrollPane chineseScrollPane = new JScrollPane(chineseTextArea); 38. chineseScrollPane.setPreferredSize(new Dimension(300, 100)); 39. JButton translateToEnglishButton = new JButton("中译英"); 40. translateToEnglishButton.setPreferredSize(new Dimension(100, 40)); 41. translateToEnglishButton.setBackground(new Color(121, 169, 253)); 42. translateToEnglishButton.setForeground(Color.WHITE); 43. translateToEnglishButton.setFocusPainted(false); 44. *// 英文输入/翻译文本框及按钮* 45. JTextArea englishTextArea = new JTextArea(5, 20); 46. englishTextArea.setLineWrap(true); 47. englishTextArea.setWrapStyleWord(true); 48. JScrollPane englishScrollPane = new JScrollPane(englishTextArea); 49. englishScrollPane.setPreferredSize(new Dimension(300, 100)); 50. JButton translateToChineseButton = new JButton("英译中"); 51. translateToChineseButton.setPreferredSize(new Dimension(100, 40)); 52. translateToChineseButton.setBackground(new Color(56, 128, 255)); 53. translateToChineseButton.setForeground(Color.WHITE); 54. translateToChineseButton.setFocusPainted(false); 55. *// 设置按钮行为* 56. translateToEnglishButton.addActionListener(new ActionListener() { 57. @Override 58. public void actionPerformed(ActionEvent e) { 59. String chineseText = chineseTextArea.getText().trim(); 60. if (!chineseText.isEmpty()) { 61. String translatedText = ""; 62. *// 判断使用哪个API* 63. String selectedAPI = (String) apiComboBox.getSelectedItem(); 64. if (selectedAPI.equals("百度 API")) { 65. translatedText = translateWithBaidu(chineseText); 66. } else if (selectedAPI.equals("硅基 API")) { 67. translatedText = translateWithSilicon(chineseText, "中译英"); 68. } 69. englishTextArea.setText(translatedText); 70. } else { 71. JOptionPane.showMessageDialog(frame, "请输入中文句子！"); 72. } 73. } 74. }); 75. translateToChineseButton.addActionListener(new ActionListener() { 76. @Override 77. public void actionPerformed(ActionEvent e) { 78. String englishText = englishTextArea.getText().trim(); 79. if (!englishText.isEmpty()) { 80. String translatedText = ""; 81. *// 判断使用哪个API* 82. String selectedAPI = (String) apiComboBox.getSelectedItem(); 83. if (selectedAPI.equals("百度 API")) { 84. translatedText = translateWithBaidu(englishText); 85. } else if (selectedAPI.equals("硅基 API")) { 86. translatedText = translateWithSilicon(englishText, "英译中"); 87. } 88. chineseTextArea.setText(translatedText); 89. } else { 90. JOptionPane.showMessageDialog(frame, "请输入英文句子！"); 91. } 92. } 93. }); 94. *// 设置面板* 95. panel.add(apiLabel); 96. panel.add(apiComboBox); 97. panel.add(chineseLabel); 98. panel.add(chineseScrollPane); 99. panel.add(translateToEnglishButton); 100. panel.add(englishScrollPane); 101. panel.add(translateToChineseButton); 102. *// 设置面板的边框* 103. panel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10)); 104. *// 添加面板到窗口* 105. frame.add(panel, BorderLayout.CENTER); 106. *// 创建底部面板来放置一些提示或说明文字* 107. JPanel bottomPanel = new JPanel(); 108. JLabel footerLabel = new JLabel("选择翻译API，输入中文或英文以进行翻译。"); 109. footerLabel.setFont(new Font("Arial", Font.ITALIC, 12)); 110. bottomPanel.add(footerLabel); 111. *// 添加底部面板到窗口* 112. frame.add(bottomPanel, BorderLayout.SOUTH); 113. *// 设置窗口居中显示* 114. frame.setLocationRelativeTo(null); 115. *// 显示窗口* 116. frame.setVisible(true); 117. } 118. *// 百度文心一言翻译（中译英 / 英译中）* 119. public static String translateWithBaidu(String text) { 120. try { 121. URL url = new URL(BAIDU\_CHAT\_URL + "?access\_token=" + BAIDU\_ACCESS\_TOKEN); 122. HttpURLConnection connection = (HttpURLConnection) url.openConnection(); 123. connection.setRequestMethod("POST"); 124. connection.setDoOutput(true); 125. connection.setRequestProperty("Content-Type", "application/json"); 126. *// 构造请求 JSON* 127. JSONObject messageObject = new JSONObject(); 128. messageObject.put("role", "user"); 129. messageObject.put("content", text); 130. JSONArray messages = new JSONArray(); 131. messages.put(messageObject); 132. JSONObject requestBody = new JSONObject(); 133. requestBody.put("messages", messages); 134. *// 发送请求* 135. OutputStream os = connection.getOutputStream(); 136. os.write(requestBody.toString().getBytes()); 137. os.close(); 138. *// 读取响应* 139. BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream())); 140. StringBuilder response = new StringBuilder(); 141. String line; 142. while ((line = in.readLine()) != null) { 143. response.append(line); 144. } 145. in.close(); 146. *// 解析响应* 147. JSONObject jsonResponse = new JSONObject(response.toString()); 148. return jsonResponse.optString("result", "翻译失败"); 149. } catch (Exception e) { 150. e.printStackTrace(); 151. return "调用百度文心一言失败！"; 152. } 153. } 154. *// 硅基翻译（中译英 / 英译中）* 155. public static String translateWithSilicon(String text, String direction) { 156. try { 157. URL url = new URL("https://api.siliconflow.cn/v1/chat/completions"); 158. HttpURLConnection connection = (HttpURLConnection) url.openConnection(); 159. connection.setRequestMethod("POST"); 160. connection.setDoOutput(true); 161. connection.setRequestProperty("Content-Type", "application/json"); 162. connection.setRequestProperty("Authorization", "Bearer " + SILICON\_API\_KEY); 163. *// 构造请求 JSON* 164. JSONObject userMessage = new JSONObject(); 165. userMessage.put("role", "user"); 166. *// 添加翻译方向到请求内容* 167. userMessage.put("content", direction + ": " + text); 168. JSONArray messages = new JSONArray(); 169. messages.put(userMessage); 170. JSONObject requestBody = new JSONObject(); 171. requestBody.put("model", "Qwen/Qwen2-7B-Instruct"); 172. requestBody.put("messages", messages); 173. requestBody.put("max\_tokens", 4096); 174. requestBody.put("stream", false); 175. requestBody.put("temperature", 0.5); 176. *// 发送请求* 177. OutputStream os = connection.getOutputStream(); 178. os.write(requestBody.toString().getBytes()); 179. os.close(); 180. *// 读取响应* 181. int responseCode = connection.getResponseCode(); 182. InputStream inputStream; 183. if (responseCode == HttpURLConnection.HTTP\_OK) { 184. inputStream = connection.getInputStream(); 185. } else { 186. inputStream = connection.getErrorStream(); 187. } 188. BufferedReader in = new BufferedReader(new InputStreamReader(inputStream)); 189. StringBuilder response = new StringBuilder(); 190. String line; 191. while ((line = in.readLine()) != null) { 192. response.append(line); 193. } 194. in.close(); 195. *// 打印响应内容* 196. *//  System.out.println("API响应: " + response.toString());* 197. *// 解析响应* 198. JSONObject jsonResponse = new JSONObject(response.toString()); 199. JSONArray results = jsonResponse.optJSONArray("choices"); 200. if (results != null && results.length() > 0) { 201. return results.getJSONObject(0).optJSONObject("message").optString("content", "翻译失败"); 202. } else { 203. return "翻译失败"; 204. } 205. } catch (Exception e) { 206. e.printStackTrace(); 207. return "调用硅基API失败！"; 208. } 209. } 210. }   ·运行结果    以硅基API为例：  中译英      英译中      题目例子的翻译：  中译英    翻译为  Established 41 years ago, Shenzhen University, adhering to its motto of "self-reliance, self-discipline, and self-improvement", has followed the special economic zone, pioneered reforms, and achieved rapid development. It has made significant contributions to the development of the special economic zone and the modernization of the country.  英译中：    翻译为  遵循“自立、自严、自强”的校训，该大学致力于服务于深圳经济特区（SEZ），展现中国的改革开放，并在高等教育领域引领变革。  以百度API为例：  中译英    英译中：    题目例子翻译：  中译英：    翻译为：  Founded 41 years ago, Shenzhen University has adhered to the school motto of "self-reliance, self-discipline, and self-improvement," closely following the pace of the special economic zone, and has been committed to reform and rapid development. The university has made important contributions to the development of the special zone and the modernization of the country.  英译中：    翻译为：  坚守“自力更生、自我约束、自我提升”的校训，这所大学致力于为深圳经济特区(SEZ)服务，展示中国的改革开放和高等教育领域的开拓创新。  ·文字说明  1. 主程序入口  public static void main(String[] args) {  程序的入口方法，负责初始化 GUI 界面，设置布局和组件，以及绑定用户交互事件。  采用了 Swing 进行 GUI 开发，主窗口使用 JFrame 作为容器。  2. GUI 界面设计  (1) 主窗口设计  JFrame frame = new JFrame("中英互译 Demo");  frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  frame.setSize(600, 500);  frame.setLayout(new BorderLayout(10, 10));  使用 JFrame 创建主窗口。  布局为 BorderLayout，主界面分为中间内容区和底部提示区。  (2) 主面板布局  JPanel panel = new JPanel();  panel.setLayout(new FlowLayout(FlowLayout.LEFT, 10, 10));  使用 FlowLayout 布局方式，使控件从左到右排列，带有适当的间距。  (3) API 选择控件  JLabel apiLabel = new JLabel("选择API：");  String[] apiOptions = {"百度 API", "硅基 API"};  JComboBox<String> apiComboBox = new JComboBox<>(apiOptions);  apiComboBox.setPreferredSize(new Dimension(150, 30));  使用 JComboBox 提供 API 选择功能。  默认有两个选项：百度 API 和 硅基 API。  (4) 中文输入与翻译结果区域  JTextArea chineseTextArea = new JTextArea(5, 20);  JScrollPane chineseScrollPane = new JScrollPane(chineseTextArea);  JButton translateToEnglishButton = new JButton("中译英");  使用 JTextArea 提供中文输入框，支持多行文本输入。  添加一个按钮 translateToEnglishButton，用于触发“中译英”翻译。  (5) 英文输入与翻译结果区域  JTextArea englishTextArea = new JTextArea(5, 20);  JScrollPane englishScrollPane = new JScrollPane(englishTextArea);  JButton translateToChineseButton = new JButton("英译中");  类似中文输入区，提供英文输入框和按钮 translateToChineseButton。  (6) 底部提示区  JPanel bottomPanel = new JPanel();  JLabel footerLabel = new JLabel("Select Translation API and enter Chinese or English for translation.");  bottomPanel.add(footerLabel);  底部区域用于展示说明文字，提示用户如何操作。  3. 按钮事件绑定  (1) 中译英按钮事件  translateToEnglishButton.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  String chineseText = chineseTextArea.getText().trim();  if (!chineseText.isEmpty()) {  String translatedText = "";  String selectedAPI = (String) apiComboBox.getSelectedItem();  if (selectedAPI.equals("百度 API")) {  translatedText = translateWithBaidu(chineseText);  } else if (selectedAPI.equals("硅基 API")) {  translatedText = translateWithSilicon(chineseText, "中译英");  }  englishTextArea.setText(translatedText);  } else {  JOptionPane.showMessageDialog(frame, "请输入中文句子！");  }  }  });  获取用户输入的中文文本，如果为空则弹出提示。  根据用户选择的 API，调用 translateWithBaidu 或 translateWithSilicon 进行翻译。  将翻译结果显示在英文文本框中。  (2) 英译中按钮事件  translateToChineseButton.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  String englishText = englishTextArea.getText().trim();  if (!englishText.isEmpty()) {  String translatedText = "";  String selectedAPI = (String) apiComboBox.getSelectedItem();  if (selectedAPI.equals("百度 API")) {  translatedText = translateWithBaidu(englishText);  } else if (selectedAPI.equals("硅基 API")) {  translatedText = translateWithSilicon(englishText, "英译中");  }  chineseTextArea.setText(translatedText);  } else {  JOptionPane.showMessageDialog(frame, "请输入英文句子！");  }  }  });  英文翻译按钮的逻辑与中文翻译类似。  4. 百度 API 翻译实现  public static String translateWithBaidu(String text) {  try {  URL url = new URL(BAIDU\_CHAT\_URL + "?access\_token=" + BAIDU\_ACCESS\_TOKEN);  HttpURLConnection connection = (HttpURLConnection) url.openConnection();  connection.setRequestMethod("POST");  connection.setDoOutput(true);  connection.setRequestProperty("Content-Type", "application/json");    JSONObject messageObject = new JSONObject();  messageObject.put("role", "user");  messageObject.put("content", text);  JSONArray messages = new JSONArray();  messages.put(messageObject);  JSONObject requestBody = new JSONObject();  requestBody.put("messages", messages);    OutputStream os = connection.getOutputStream();  os.write(requestBody.toString().getBytes());  os.close();    BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()));  StringBuilder response = new StringBuilder();  String line;  while ((line = in.readLine()) != null) {  response.append(line);  }  in.close();    JSONObject jsonResponse = new JSONObject(response.toString());  return jsonResponse.optString("result", "翻译失败");  } catch (Exception e) {  e.printStackTrace();  return "调用百度文心一言失败！";  }  }  使用 POST 方法调用百度 API。  构造 JSON 请求体并发送。  解析响应 JSON，提取翻译结果。  5. 硅基 API 翻译实现  public static String translateWithSilicon(String text, String direction) {  try {  URL url = new URL(SILICON\_CHAT\_URL);  HttpURLConnection connection = (HttpURLConnection) url.openConnection();  connection.setRequestMethod("POST");  connection.setDoOutput(true);  connection.setRequestProperty("Content-Type", "application/json");  connection.setRequestProperty("Authorization", "Bearer " + SILICON\_API\_KEY);    JSONObject userMessage = new JSONObject();  userMessage.put("role", "user");  userMessage.put("content", direction + ": " + text);  JSONArray messages = new JSONArray();  messages.put(userMessage);    JSONObject requestBody = new JSONObject();  requestBody.put("model", "Qwen/Qwen2-7B-Instruct");  requestBody.put("messages", messages);  requestBody.put("max\_tokens", 4096);  requestBody.put("stream", false);  requestBody.put("temperature", 0.5);    OutputStream os = connection.getOutputStream();  os.write(requestBody.toString().getBytes());  os.close();    BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()));  StringBuilder response = new StringBuilder();  String line;  while ((line = in.readLine()) != null) {  response.append(line);  }  in.close();    JSONObject jsonResponse = new JSONObject(response.toString());  JSONArray results = jsonResponse.optJSONArray("choices");  if (results != null && results.length() > 0) {  return results.getJSONObject(0).optJSONObject("message").optString("content", "翻译失败");  } else {  return "翻译失败";  }  } catch (Exception e) {  e.printStackTrace();  return "调用硅基API失败！";  }  }  构造 POST 请求调用硅基 API。  提供翻译方向（中译英或英译中）。  解析响应 JSON，提取翻译结果。  **Part 3（30分）**  (3.1). 利用套接字连接（TCP）编写程序，该程序包括三个客户端（ClientA、ClientB、ClientC）和一个服务端（ServerS），三个客户端通过服务端作为桥梁实现相互间的文字交流，例如，ClientA先发信息给ServerS，然后ServerC再将收到的信息转发给ClientB和ClientC。在报告中附上示意图（三个客户端+一个服务端）、程序截图、完整的运行结果和简要文字说明。（20分）  **·说明**  以下是刚开始没有仔细思考，导致片面的结果，下面的代码是片面的，因为我的理解是被下面红字误导了，A发信息给服务端S，S再中转给B、C，但是这样的话，A仅仅只有发送信息功能，B、C也只有接收的功能。  在咨询老师和思考后，应该是A、B、C均具有发送、接收功能。这个S就是中转站。  这里我打个比方：类似于**微信群聊**，A、B、C就是每个用户，他们可以发送信息，也能收到彼此间发送的信息，而这一切的完成需要服务端S来进行构建桥梁。  17472ec817ce155640423ac0d1e20e1  ·示意图    ·程序截图（含完整代码）  由于Server.java截图不全，我会展示全部代码，其余已有完整截图，这里节省空间不作展示。          （完整代码）   1. package 课程作业; 2. import java.io.\*; 3. import java.net.\*; 4. import java.util.\*; 5. public class ServerS { 6. private static final int PORT = 12345; 7. private static List<PrintWriter> clients = new ArrayList<>(); 8. public static void main(String[] args) { 9. try (ServerSocket serverSocket = new ServerSocket(PORT)) { 10. System.out.println("Server started..."); 11. while (true) { 12. Socket clientSocket = serverSocket.accept(); 13. synchronized (clients) { 14. clients.add(new PrintWriter(clientSocket.getOutputStream(), true)); 15. } 16. new ClientHandler(clientSocket).start(); 17. } 18. } catch (IOException e) { 19. e.printStackTrace(); 20. } 21. } 22. private static class ClientHandler extends Thread { 23. private Socket clientSocket; 24. private BufferedReader in; 25. public ClientHandler(Socket socket) { 26. this.clientSocket = socket; 27. } 28. @Override 29. public void run() { 30. try { 31. in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream())); 32. String message; 33. while ((message = in.readLine()) != null) { 34. System.out.println("Received: " + message); 35. synchronized (clients) { 36. for (PrintWriter client : clients) { 37. client.println(message); 38. } 39. } 40. } 41. } catch (IOException e) { 42. e.printStackTrace(); 43. } finally { 44. try { 45. in.close(); 46. clientSocket.close(); 47. } catch (IOException e) { 48. e.printStackTrace(); 49. } 50. } 51. } 52. } 53. }   ·运行结果  启动：          现在从ClientA的终端窗口输入：123;    ServerS的反应：    ClientB/C的反应：      接着继续从ClientA终端窗口输入“I love JAVA!”，观察反应：      ServerS的反应：    CilentB/C的反应：      ·文字说明：  **第一段代码（ServerS）**  服务器端代码，用于监听客户端连接并将接收到的消息广播给所有连接的客户端。  **关键点**  服务器启动：  通过 ServerSocket 监听端口 12345。  每当有新的客户端连接时，接受该连接并启动一个新的线程 ClientHandler 来处理该客户端的消息。  客户端管理：  clients 是一个 List<PrintWriter>，存储所有客户端的输出流，以便实现消息广播。  消息广播：  每当服务器接收到一条来自某客户端的消息时，它会将该消息发送给所有连接的客户端。  **主要代码解释**  ServerSocket serverSocket = new ServerSocket(PORT)： 启动服务器，监听指定端口。  clients.add(new PrintWriter(clientSocket.getOutputStream(), true))： 将新连接客户端的输出流存储到 clients 列表中。  new ClientHandler(clientSocket).start()： 为每个客户端启动单独的线程，异步处理消息。  for (PrintWriter client : clients)： 遍历所有连接的客户端，将接收到的消息广播出去。  **运行效果**  服务器可以处理多个客户端连接，并将某一客户端发送的消息广播给所有客户端。  **第二段代码（ClientA）**  客户端 A 的代码，用于向服务器发送消息。  **关键点**  连接服务器：  客户端通过指定的 IP 地址和端口号连接到服务器。  发送消息：  用户可以在控制台输入消息，客户端会将消息发送给服务器。  退出机制：  当用户输入 "exit" 时，客户端会断开与服务器的连接。  **主要代码解释**  Socket socket = new Socket(SERVER\_ADDRESS, SERVER\_PORT)： 建立与服务器的连接。  PrintWriter out = new PrintWriter(socket.getOutputStream(), true)： 用于向服务器发送消息。  BufferedReader in = new BufferedReader(new InputStreamReader(System.in))： 从控制台读取用户输入。  out.println(message)： 将用户输入的消息发送到服务器。  **运行效果**  通过客户端 A 向服务器发送消息，服务器会将消息转发给所有已连接的客户端。  **第三段代码（ClientB）**  客户端 B 的代码，用于接收服务器广播的消息。  **关键点**  连接服务器：  客户端通过指定的 IP 地址和端口号连接到服务器。  接收消息：  客户端持续监听服务器发送的消息，并在控制台打印收到的内容。  主要代码解释  BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()))： 从服务器的输入流读取消息。  while ((message = in.readLine()) != null)： 持续接收服务器发送的消息。  **运行效果**  客户端 B 能够实时接收服务器广播的消息，并在控制台输出。  **第四段代码（ClientC）**  客户端 C 的代码，与 ClientB 基本相同，也是用于接收服务器广播的消息。  **关键点**  连接服务器：  客户端通过指定的 IP 地址和端口号连接到服务器。  接收消息：  客户端持续监听服务器发送的消息，并在控制台打印收到的内容。  **运行效果**  客户端 C 能够实时接收服务器广播的消息，并在控制台输出。  **整体运行流程**  服务器启动：  ServerS 启动后等待客户端连接。  客户端连接：  ClientA 连接到服务器并发送消息。  ClientB 和 ClientC 连接到服务器并接收消息。  消息广播：  当 ClientA 向服务器发送消息时，服务器会广播给所有连接的客户端（包括 ClientB 和 ClientC）。  退出机制：  客户端可以通过输入 "exit" 断开连接。  正确部分如下（更新后）：  ·完整代码  Server类：   1. import java.io.\*; 2. import java.net.\*; 3. import java.util.\*; 4. public class ServerS { 5. private static final int PORT = 12345; 6. private static Set<ClientHandler> clientHandlers = new HashSet<>(); 7. public static void main(String[] args) { 8. System.out.println("Server is running..."); 9. try (ServerSocket serverSocket = new ServerSocket(PORT)) { 10. while (true) { 11. Socket clientSocket = serverSocket.accept(); 12. ClientHandler clientHandler = new ClientHandler(clientSocket); 13. clientHandlers.add(clientHandler); 14. new Thread(clientHandler).start(); 15. } 16. } catch (IOException e) { 17. e.printStackTrace(); 18. } 19. } 20. public static void broadcastMessage(String message, ClientHandler sender) { 21. for (ClientHandler client : clientHandlers) { 22. if (client != sender) { 23. client.sendMessage(message); 24. } 25. } 26. } 27. public static void removeClient(ClientHandler clientHandler) { 28. clientHandlers.remove(clientHandler); 29. } 30. private static class ClientHandler implements Runnable { 31. private Socket socket; 32. private PrintWriter out; 33. private String clientName; 34. public ClientHandler(Socket socket) { 35. this.socket = socket; 36. } 37. @Override 38. public void run() { 39. try (InputStream input = socket.getInputStream(); 40. BufferedReader reader = new BufferedReader(new InputStreamReader(input))) { 41. out = new PrintWriter(socket.getOutputStream(), true); 42. clientName = reader.readLine(); 43. System.out.println(clientName + " has connected."); 44. String message; 45. while ((message = reader.readLine()) != null) { 46. System.out.println(clientName + ": " + message); 47. ServerS.broadcastMessage(clientName + ": " + message, this); 48. } 49. } catch (IOException e) { 50. System.out.println(clientName + " has disconnected."); 51. } finally { 52. ServerS.removeClient(this); 53. try { 54. socket.close(); 55. } catch (IOException e) { 56. e.printStackTrace(); 57. } 58. } 59. } 60. public void sendMessage(String message) { 61. if (out != null) { 62. out.println(message); 63. } 64. } 65. } 66. }   Client类（A、B、C相似，仅名字不一样）   1. import java.io.\*; 2. import java.net.\*; 3. import java.util.Scanner; 4. public class Client { 5. private static final String SERVER\_ADDRESS = "127.0.0.1"; 6. private static final int SERVER\_PORT = 12345; 7. public static void main(String[] args) { 8. System.out.print("Enter your name: "); 9. Scanner scanner = new Scanner(System.in); 10. String clientName = scanner.nextLine(); 11. try (Socket socket = new Socket(SERVER\_ADDRESS, SERVER\_PORT); 12. PrintWriter out = new PrintWriter(socket.getOutputStream(), true); 13. BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()))) { 15. out.println(clientName); 17. new Thread(() -> { 18. String message; 19. try { 20. while ((message = in.readLine()) != null) { 21. System.out.println(message); 22. } 23. } catch (IOException e) { 24. System.out.println("Connection closed."); 25. } 26. }).start(); 28. System.out.println("Start chatting..."); 29. while (true) { 30. String message = scanner.nextLine(); 31. out.println(message); 32. } 33. } catch (IOException e) { 34. e.printStackTrace(); 35. } 36. } 37. }   ·运行结果  打开服务端S：    客户端A输入名字，进行连接：      客户端B、C输入名字进行连接。      客户端A首先发送信息：    客户端B、C收到信息。      服务端S为中转站：    客户端C发送信息：    客户端B收到信息，并发送信息给A、C。        可以在服务端查看聊天记录。    ·示意图    ·文字说明  **服务端代码（ServerS）**  核心功能  服务端监听一个特定端口（12345），接收多个客户端的连接请求。  客户端连接后，服务端通过多线程处理每个客户端的消息。  将某一客户端的消息广播到所有其他已连接的客户端，实现群聊功能。  主要组成部分  main 方法  功能：启动服务端，监听指定端口。  接收客户端连接，启动新的线程处理每个客户端的通信。  关键代码：  try (ServerSocket serverSocket = new ServerSocket(PORT)) {  while (true) {  Socket clientSocket = serverSocket.accept();  ClientHandler clientHandler = new ClientHandler(clientSocket);  clientHandlers.add(clientHandler);  new Thread(clientHandler).start();  }  }  创建 ServerSocket 实例，绑定到端口 12345。  使用 accept() 方法阻塞，等待客户端连接。  每当接收到客户端连接时，创建一个新的 ClientHandler 实例，并将其加入到 clientHandlers 集合中，方便广播消息。  启动新线程，运行客户端处理逻辑。  broadcastMessage 方法  功能：将某个客户端的消息发送给所有其他已连接的客户端。  关键代码：  public static void broadcastMessage(String message, ClientHandler sender) {  for (ClientHandler client : clientHandlers) {  if (client != sender) {  client.sendMessage(message);  }  }  }  遍历 clientHandlers 集合。  如果目标客户端不是消息发送者（sender），则调用其 sendMessage 方法转发消息。  ClientHandler 内部类  功能：每个客户端连接都由一个 ClientHandler 线程处理。  关键方法和逻辑：构造函数： 保存客户端的 Socket，初始化输入输出流。  run 方法：从客户端接收消息并广播。  处理客户端断开连接。  sendMessage 方法： 将消息发送到客户端的输出流。  **客户端代码（Client）**  核心功能  客户端连接服务端，通过输入输出流发送和接收消息。  客户端既能发送消息到服务端，又能接收服务端广播的消息。  主要组成部分  main 方法  功能：启动客户端，连接到指定地址和端口的服务端。  创建发送和接收消息的线程。  关键代码：  try (Socket socket = new Socket(SERVER\_ADDRESS, SERVER\_PORT);  PrintWriter out = new PrintWriter(socket.getOutputStream(), true);  BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()))) {  使用 Socket 连接服务端。  初始化输出流（out）和输入流（in）：out 用于向服务端发送消息。  in 用于从服务端接收广播消息。  消息发送线程  功能：从用户输入中读取消息，并发送给服务端。  关键代码：  while (true) {  String message = scanner.nextLine();  out.println(message);  }  持续读取用户输入，并通过输出流发送到服务端。  消息接收线程  功能：监听服务端的消息广播，并显示到控制台。  关键代码：  new Thread(() -> {  String message;  try {  while ((message = in.readLine()) != null) {  System.out.println(message);  }  } catch (IOException e) {  System.out.println("Connection closed.");  }  }).start();  启动一个新线程持续读取输入流。  如果接收到消息，打印到控制台。  (3.2). 利用数据报通信（UDP）实现题(1)中的要求。（10分）  下面的也是片面的错误部分。  ·示意图    ·程序截图（含完整代码）  ClientB    ClientC    ClientA    ServerS（代码过长，我会展示完整代码）    完整代码   1. package 实验4; 2. import java.net.\*; 3. import java.util.\*; 4. public class ServerS { 5. private static final int PORT = 12345; 6. private static DatagramSocket socket; 7. private static Map<String, Integer> clients = new HashMap<>(); *// 记录客户端地址和端口* 8. public static void main(String[] args) { 9. try { 10. socket = new DatagramSocket(PORT); 11. System.out.println("Server started..."); 12. byte[] receiveData = new byte[1024]; 13. while (true) { 14. DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length); 15. socket.receive(receivePacket); *// 接收客户端消息* 16. String message = new String(receivePacket.getData(), 0, receivePacket.getLength()); 17. InetAddress clientAddress = receivePacket.getAddress(); 18. int clientPort = receivePacket.getPort(); 19. *// 记录新客户端* 20. String clientKey = clientAddress.getHostAddress() + ":" + clientPort; 21. if (!clients.containsKey(clientKey)) { 22. clients.put(clientKey, clientPort); 23. System.out.println("New client connected: " + clientKey); 24. } 25. System.out.println("Received from " + clientKey + ": " + message); 26. *// 判断是否为普通消息，而非控制消息* 27. if (!message.equals("REGISTER")) { 28. *// 转发消息给其他客户端* 29. for (Map.Entry<String, Integer> entry : clients.entrySet()) { 30. String[] clientInfo = entry.getKey().split(":"); 31. InetAddress address = InetAddress.getByName(clientInfo[0]); 32. int port = entry.getValue(); 33. if (!(address.equals(clientAddress) && port == clientPort)) { 34. DatagramPacket sendPacket = new DatagramPacket( 35. message.getBytes(), 36. message.length(), 37. address, 38. port 39. ); 40. socket.send(sendPacket); 41. } 42. } 43. } 44. } 45. } catch (Exception e) { 46. e.printStackTrace(); 47. } 48. } 49. }   ·运行结果  启动：          下面实验的时候我试了另一种方式，就是把ClientC先关了，输完Lakers再打开，接着输入Champion！，观察各个窗口的反应来比对。  在ClientA窗口输入：“Lakers！”    ServerS的反应    ClientB的反应：    这个时候我才启动了ClientC，可以看到：（可以看到没有收到Lakers！）    接着在ClientA窗口输入Champion！：    ServerS的反应：    ClientB的反应    这个时候我们可以看一下ClientC的反应：    显然，他由于刚才没有启动，并没有接收到Lakers！，但是启动后接收到了Champion！。  ·文字说明  **第一段代码：ServerS**  **功能**  服务器端代码，负责接收客户端发送的消息，并将消息转发给其他已连接的客户端。  监听一个固定的端口（12345），通过UDP协议与多个客户端通信。  维护一个客户端列表，记录所有已连接的客户端地址。  **关键逻辑**  创建服务器端Socket：  socket = new DatagramSocket(PORT);  服务器绑定到端口12345，监听所有发送到该端口的消息。  接收消息：  socket.receive(receivePacket);  使用UDP的DatagramSocket接收消息，保存到receivePacket中。  消息内容通过receivePacket.getData()获取。  记录客户端：  if (!clients.containsValue(clientAddress)) {  clients.put("Client" + (clients.size() + 1), clientAddress);  }  如果当前客户端地址尚未记录，则将其加入clients映射表。  消息转发：  for (Map.Entry<String, InetAddress> entry : clients.entrySet()) {  if (!entry.getValue().equals(clientAddress)) {  socket.send(sendPacket);  }  }  将收到的消息发送给所有其他客户端（广播消息）。  **注意事项**  客户端列表只记录地址（InetAddress），未考虑端口信息，可能会导致问题。  转发逻辑中未处理重复发送的可能性。  没有明确区分控制消息（如REGISTER）和普通聊天消息。  **第二段代码：ClientA**  **功能**  一个UDP客户端，负责向服务器发送消息。  提供命令行界面，允许用户输入消息并发送到服务器。  **关键逻辑**  创建客户端Socket：  try (DatagramSocket socket = new DatagramSocket()) {  创建一个未绑定特定端口的DatagramSocket，让操作系统自动分配端口。  用户输入并发送消息：  message = scanner.nextLine();  DatagramPacket sendPacket = new DatagramPacket(  message.getBytes(),  message.length(),  serverAddress,  SERVER\_PORT  );  socket.send(sendPacket);  用户输入消息，通过DatagramPacket封装后发送到服务器地址和端口。  退出机制：  if (message.equalsIgnoreCase("exit")) {  break;  }  用户输入exit后，退出客户端程序。  **注意事项**  没有接收消息的逻辑，仅负责发送。  服务器地址硬编码为localhost，只能在同一台主机上测试。  **第三段代码：ClientB**  **功能**  一个UDP客户端，负责接收服务器转发的消息。  启动后自动监听服务器的广播消息。  **关键逻辑**  创建客户端Socket：  try (DatagramSocket socket = new DatagramSocket()) {  同样使用未绑定特定端口的DatagramSocket。  接收消息：  DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);  socket.receive(receivePacket);  String message = new String(receivePacket.getData(), 0, receivePacket.getLength());  使用socket.receive()接收消息，并将消息转换为字符串。  输出接收到的消息：  System.out.println("ClientB received: " + message);  将接收到的消息打印到控制台。  **注意事项**  receive()方法是阻塞的，会等待服务器的消息。  未实现发送消息的功能，客户端仅作为被动接收者。  **第四段代码：ClientC**  **功能**  与ClientB类似，也是一个被动接收消息的UDP客户端。  **关键逻辑**  代码逻辑与ClientB几乎完全相同，区别仅在于类名。  下面的是正确部分（更新后）：  ·完整代码  服务端S类   1. import java.net.\*; 2. import java.util.\*; 3. public class ServerS { 4. private static final int SERVER\_PORT = 12345; 5. private static final int BUFFER\_SIZE = 1024; 6. private static final Map<String, InetSocketAddress> clients = new HashMap<>(); 7. public static void main(String[] args) { 8. System.out.println("UDP Server is running..."); 9. try (DatagramSocket serverSocket = new DatagramSocket(SERVER\_PORT)) { 10. byte[] buffer = new byte[BUFFER\_SIZE]; 11. while (true) { 12. DatagramPacket packet = new DatagramPacket(buffer, buffer.length); 13. serverSocket.receive(packet); 14. String message = new String(packet.getData(), 0, packet.getLength()); 15. InetSocketAddress clientAddress = new InetSocketAddress(packet.getAddress(), packet.getPort()); 17. if (!clients.containsValue(clientAddress)) { 18. clients.put(message.split(":", 2)[0], clientAddress); 19. System.out.println("New client registered: " + clientAddress); 20. continue; 21. } 22. System.out.println("Received: " + message + " from " + clientAddress); 24. for (InetSocketAddress client : clients.values()) { 25. if (!client.equals(clientAddress)) { 26. byte[] data = message.getBytes(); 27. DatagramPacket sendPacket = new DatagramPacket(data, data.length, client.getAddress(), client.getPort()); 28. serverSocket.send(sendPacket); 29. } 30. } 31. } 32. } catch (Exception e) { 33. e.printStackTrace(); 34. } 35. } 36. }   客户端A、B、C：   1. import java.io.\*; 2. import java.net.\*; 3. import java.util.Scanner; 4. public class Client { 5. private static final String SERVER\_ADDRESS = "127.0.0.1"; 6. private static final int SERVER\_PORT = 12345; 7. public static void main(String[] args) { 8. System.out.print("Enter your name: "); 9. Scanner scanner = new Scanner(System.in); 10. String clientName = scanner.nextLine(); 11. try (Socket socket = new Socket(SERVER\_ADDRESS, SERVER\_PORT); 12. PrintWriter out = new PrintWriter(socket.getOutputStream(), true); 13. BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()))) { 15. out.println(clientName); 17. new Thread(() -> { 18. String message; 19. try { 20. while ((message = in.readLine()) != null) { 21. System.out.println(message); 22. } 23. } catch (IOException e) { 24. System.out.println("Connection closed."); 25. } 26. }).start(); 28. System.out.println("Start chatting..."); 29. while (true) { 30. String message = scanner.nextLine(); 31. out.println(message); 32. } 33. } catch (IOException e) { 34. e.printStackTrace(); 35. } 36. } 37. }   ·运行结果  启动服务端S：    客户A输入名字，开始聊天：    服务端收到客户端A启动聊天信息：    同理，启动B、C。    客户端A发送信息：    客户端B、C收到信息：    服务端反应：    客户端B、C分别发送信息：    客户端A收到：    服务端反应：    ·示意图  示意图与上题一致。    ·文字说明  **客户端代码 (Client)**  客户端的作用是连接到服务器，与其他客户端进行聊天。  主要功能  连接服务器  使用 Socket 连接到指定的服务器地址和端口 (127.0.0.1:12345)。  接收消息  客户端开启一个线程来监听来自服务器的消息，并打印到控制台。  发送消息  客户端主线程负责从用户输入读取消息并发送到服务器。  关键代码讲解  1. 连接服务器  Socket socket = new Socket(SERVER\_ADDRESS, SERVER\_PORT);  创建 Socket 并连接到服务器。  SERVER\_ADDRESS 为 127.0.0.1（本地地址），SERVER\_PORT 为 12345。  2. 发送客户端名称  out.println(clientName);  连接后，客户端将用户输入的名称发送给服务器，用于标识发送者。  3. 接收服务器消息  new Thread(() -> {  String message;  try {  while ((message = in.readLine()) != null) {  System.out.println(message);  }  } catch (IOException e) {  System.out.println("Connection closed.");  }  }).start();  独立线程监听来自服务器的消息。  当有新消息时，打印到控制台。  4. 发送消息到服务器  while (true) {  String message = scanner.nextLine(); // 读取用户输入  out.println(message); // 发送到服务器  }  主线程持续读取用户输入，并通过 PrintWriter 将消息发送到服务器。  **服务器端代码 (ServerS)**  服务器的作用是接受客户端连接，管理客户端的消息通信，并将一个客户端的消息广播到所有其他客户端。  主要功能  监听客户端连接  使用 ServerSocket 来监听指定端口 (PORT = 12345) 上的客户端连接请求。  管理客户端  使用 Set<ClientHandler> 存储所有连接的客户端，便于消息广播和管理。  消息广播  当一个客户端发送消息时，服务器将消息转发给其他所有客户端。  处理客户端断开连接  如果某个客户端断开连接，服务器从客户端列表中移除对应的 ClientHandler。  关键代码讲解  1. 主线程监听客户端连接  try (ServerSocket serverSocket = new ServerSocket(PORT)) {  while (true) {  Socket clientSocket = serverSocket.accept(); // 接受新客户端  ClientHandler clientHandler = new ClientHandler(clientSocket);  clientHandlers.add(clientHandler); // 添加到客户端集合  new Thread(clientHandler).start(); // 为每个客户端开启新线程  }  }  ServerSocket 监听端口 12345，当检测到客户端连接时，accept() 方法返回一个 Socket 对象。  为每个新连接创建一个 ClientHandler 实例，并用新线程来处理该客户端的消息收发。  2. 客户端消息广播  public static void broadcastMessage(String message, ClientHandler sender) {  for (ClientHandler client : clientHandlers) {  if (client != sender) { // 避免将消息发回给发送者  client.sendMessage(message);  }  }  }  遍历所有连接的客户端，并调用 sendMessage 方法将消息发送给它们。  sender 参数用来避免将消息发送回原始发送者。  3. 客户端消息处理器 (ClientHandler)  ClientHandler 是服务器端用于处理每个客户端的独立任务。  功能：  接收并解析来自客户端的消息。  将消息广播给其他客户端。  处理客户端断开连接的情况。  @Override  public void run() {  try (InputStream input = socket.getInputStream();  BufferedReader reader = new BufferedReader(new InputStreamReader(input))) {  out = new PrintWriter(socket.getOutputStream(), true);  clientName = reader.readLine(); // 接收客户端名称  System.out.println(clientName + " has connected.");  String message;  while ((message = reader.readLine()) != null) {  System.out.println(clientName + ": " + message);  ServerS.broadcastMessage(clientName + ": " + message, this); // 广播消息  }  } catch (IOException e) {  System.out.println(clientName + " has disconnected."); // 处理断开连接  } finally {  ServerS.removeClient(this); // 从集合中移除  }  }  接收到的客户端消息通过 ServerS.broadcastMessage 方法广播。  在客户端断开时，触发 finally 块清理连接资源。  4. 发送消息  public void sendMessage(String message) {  if (out != null) {  out.println(message);  }  }  通过 PrintWriter 将消息发送到客户端。  ++++++++++++++++++++++++++++++++++++++++++++++++++++++  **其他（例如感想、建议等等）。**  通过分析代码，我认识到对消息类型的区分、异常处理和客户端管理是提高系统稳定性和扩展性的关键。  总体而言，本实验让我不仅掌握了很多编程的基本技巧，还增强了对网络通信设计的思考能力，为今后的开发实践打下了良好基础。 |

深圳大学学生实验报告用纸

|  |
| --- |
| 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。