

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：计算机科学与技术

指导教师：潘微科

报告人：曹秦鲁学号：2023150203班级：国际班

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：

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,请于截止日期之后的**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。

一、项目概述

本项目实现了 Amazon 手表评论数据的处理功能，从原始评论数据中提取用户 ID 和产品 ID，并按照特定规则排序生成新的数据文件。

二、代码实现分析

1. 主类结构

主类`AmazonProductSimilarityAnalyzer`的核心实现：

```
``java
public class AmazonProductSimilarityAnalyzer {
    public static void main(String[] args) {
        try (BufferedReader reader = new BufferedReader(new
FileReader("Watches.txt"))) {
            Set<Review> reviews = new TreeSet<>();    // 使用TreeSet 自动排序

            String line;
            String currentUserId = null;              // 临时存储当前处理的userId
            String currentProductId = null;            // 临时存储当前处理的productId

            while ((line = reader.readLine()) != null) {
                if (line.trim().isEmpty()) {
                    if (currentUserId != null && currentProductId != null
                        && !currentUserId.equals("unknown")) {
                        reviews.add(new Review(currentUserId, currentProductId));
                    }
                    currentUserId = null;
                    currentProductId = null;
                    continue;
                }

                String[] parts = line.split(": ", 2);
                if (parts.length != 2) continue;
```

```

        String key = parts[0].trim();
        String value = parts[1].trim();

        switch (key) {
            case "review/userId":
                currentUserId = value;
                break;
            case "product/productId":
                currentProductId = value;
                break;
        }
    }
}

try (PrintWriter reviewWriter = new PrintWriter("reviews.txt")) {
    for (Review review : reviews) {
        reviewWriter.println(review.userId + ";" + review.productId);
    }
}
}
}
}
}
...

```

主要功能:

- 使用 `BufferedReader` 读取输入文件
- 使用 `TreeSet` 实现自动排序
- 使用 `PrintWriter` 输出结果

2. 数据结构设计

2.1 Review 类

内部类 `Review` 的实现:

```

```java
private static class Review implements Comparable<Review> {
 String userId;
 String productId;

 @Override
 public int compareTo(Review other) {
 // 首先按userId 排序
 int userCompare = this.userId.compareTo(other.userId);
 if (userCompare != 0) {
 return userCompare;
 }
 // userId 相同时按productId 排序
 return this.productId.compareTo(other.productId);
 }
}

```

```
 }
}
...
```

关键点说明:

- 实现 `Comparable` 接口以支持排序
- `compareTo` 方法实现两级排序:
  1. 首先按 `userId` 从小到大排序
  2. `userId` 相同时按 `productId` 排序

### 3. 核心功能实现

#### 3.1 文件读取与解析

```
```java
```

```
while ((line = reader.readLine()) != null) {  
    if (line.trim().isEmpty()) {  
        // 遇到空行时处理当前记录  
        if (currentUserId != null && currentProductId != null  
            && !currentUserId.equals("unknown")) {  
            reviews.add(new Review(currentUserId, currentProductId));  
        }  
        // 重置临时变量  
        currentUserId = null;  
        currentProductId = null;  
        continue;  
    }  
  
    String[] parts = line.split(": ", 2);  
    if (parts.length != 2) continue;  
  
    String key = parts[0].trim();  
    String value = parts[1].trim();  
  
    switch (key) {  
        case "review/userId":  
            currentUserId = value;  
            break;  
        case "product/productId":  
            currentProductId = value;  
            break;  
    }  
}  
}
```

代码说明:

1. 文件读取逻辑:
 - 按行读取文件

- 使用空行作为记录分隔符
- 使用`:`分割键值对

2. 数据处理逻辑:

- 提取 `userId` 和 `productId`
- 跳过 `userId` 为 `unknown` 的记录
- 确保数据完整性

3. 数据存储:

- 使用 `TreeSet` 自动维护排序
- 每条记录包含 `userId` 和 `productId`
- 按指定规则排序存储

三、性能分析

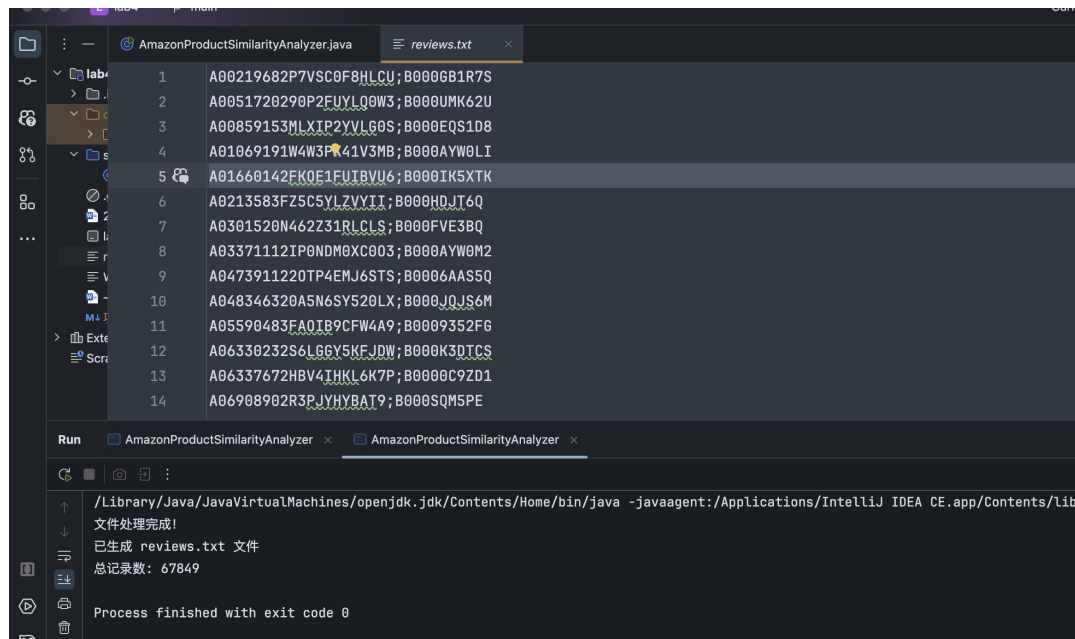
1. 时间复杂度分析:

- 文件读取: $O(n)$, n 为文件行数
- `TreeSet` 添加操作: $O(\log m)$, m 为有效评论数
- 总体: $O(n + m \log m)$

2. 空间复杂度分析:

- `TreeSet` 存储: $O(m)$, m 为有效评论数
- 临时变量: $O(1)$
- 总体: $O(m)$

运行结果:



```
AmazonProductSimilarityAnalyzer.java  reviews.txt x
1 A00219682P7VSC0F8HLCU;B000GB1R7S
2 A0051720290P2FUYLQ0W3;B000UMK62U
3 A00859153MLXIP2YVLG0S;B000EQS1D8
4 A01069191W4W3P41V3MB;B000AYW0LI
5 A01660142FK0E1FUIBVU6;B000IK5XTK
6 A0213583FZ5C5YLZVYII;B000HDJT6Q
7 A0301520N462Z31RLCLS;B000FVE3BQ
8 A03371112IP0NDM0XC003;B000AYW0M2
9 A0473911220TP4EMJ6STS;B0006AAS5Q
10 A048346320ASN6SY520LX;B000JQJS6M
11 A05590483FAQIB9CFW4A9;B0009352FG
12 A06330232S6L6GY5KFJDW;B000K3DICS
13 A06337672HBV4IHKL6K7P;B0000C9ZD1
14 A06908902R3PJYHYBAT9;B000SQMSPE

Run AmazonProductSimilarityAnalyzer AmazonProductSimilarityAnalyzer x
/Library/Java/JavaVirtualMachines/openjdk.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/Lib
文件处理完成!
已生成 reviews.txt 文件
总记录数: 67849

Process finished with exit code 0
```

(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。

1. 数据读取与预处理

```
```java
// 存储商品-用户映射关系
private static Map<String, Set<String>> productToUsers = new HashMap<>();

// 从 reviews.txt 读取数据，构建商品-用户映射
private static void buildProductToUsers(String inputFile) throws IOException {
 try (BufferedReader br = new BufferedReader(new FileReader(inputFile))) {
 String line;
 int count = 0;
 while ((line = br.readLine()) != null && count < MAX_LINES) {
 String[] parts = line.split(";");
 if (parts.length != 2) continue;
 String user = parts[0].trim();
 String product = parts[1].trim();
 productToUsers.computeIfAbsent(product, k -> new HashSet<>()).add(user);
 count++;
 }
 }
}
```
```

2. 相似度计算

- 采用 Jaccard 相似度计算公式： $|A \cap B| / |A \cup B|$
- 使用多线程并行计算所有商品对的相似度

```
```java
// 计算两个商品的 Jaccard 相似度
int intersectionSize = 0;
for (String u : users1) {
 if (users2.contains(u)) {
 intersectionSize++;
 }
}

int unionSize = users1.size() + users2.size() - intersectionSize;
double similarity = unionSize == 0 ? 0.0 : ((double) intersectionSize / unionSize);
```
```



```
...
```

3. 并行计算框架

```
```java
// 创建线程池
int numThreads = Runtime.getRuntime().availableProcessors();
ExecutorService executor = Executors.newFixedThreadPool(numThreads);

// 提交计算任务
for (int i = 0; i < products.size(); i++) {
 for (int j = i + 1; j < products.size(); j++) {
 String p1 = products.get(i);
 String p2 = products.get(j);
 futures.add(executor.submit(new ComputeTask(p1, p2)));
 }
}
```
...
```

4. 结果处理

1. 收集每个商品的相似度结果

```
```java
Map<String, List<SimilarityResult>> productSimilarities = new HashMap<>();
for (Future<List<SimilarityResult>> f : futures) {
 List<SimilarityResult> resultList = f.get();
 for (SimilarityResult r : resultList) {
 productSimilarities.get(r.product1).add(new SimilarityResult(r.product1,
r.product2, r.similarity));
 productSimilarities.get(r.product2).add(new SimilarityResult(r.product2,
r.product1, r.similarity));
 }
}
```
...
```

2. 排序并选择前三个最相似的商品

```
```java
Collections.sort(list, new Comparator<SimilarityResult>() {
 @Override
 public int compare(SimilarityResult o1, SimilarityResult o2) {
 int cmp = Double.compare(o2.similarity, o1.similarity);
 if (cmp != 0) {
 return cmp;
 } else {
 return o1.product2.compareTo(o2.product2);
 }
 }
})
```
```

```

    }
});
Collections.reverse(list); // 确保从大到小排序
...

```

5. 输出结果

- 将每个商品的前三个最相似商品写入文件
- 格式: 商品 ID;相似商品 1;相似商品 2;相似商品 3

```

```java
writer.write(p + ";" + top3.get(0) + ";" + top3.get(1) + ";" + top3.get(2));
...

```

运行结果:

lab4 -~/Documents/GitHub/java-/lab4	1	B00006I54J;B000UMK62U;B000SQM5PE;B000PSQXXU
> .idea	2	B0000C9ZD1;B000UMK62U;B000SQM5PE;B000PSQXXU
> out	3	B0000TII8M;B000UMK62U;B000SQM5PE;B000PSQXXU
> production	4	B0000TII8S;B000UMK62U;B000SQM5PE;B000PSQXXU
> src	5	B0000U0LRC;B000UMK62U;B000SQM5PE;B000PSQXXU
AmazonProductSimilarityAnalyzer	6	B0000U5BYU;B000UMK62U;B000SQM5PE;B000PSQXXU
ProductSimilarityMultiThread	7	B0000UIVQW;B000UMK62U;B000SQM5PE;B000PSQXXU
.gitignore	8	B0000WKYUE;B000UMK62U;B000SQM5PE;B000PSQXXU
202401-JavaPD-课程实验4-学号-姓名.docx	9	B000153MWW;B000UMK62U;B000SQM5PE;B000PSQXXU
lab4.iml	10	B0001HIT1Y;B000UMK62U;B000SQM5PE;B000PSQXXU
productNeighborhood.txt	11	B0001MLMKY;B000UMK62U;B000SQM5PE;B000PSQXXU
reviews.txt	12	B0002948KC;B000UMK62U;B000SQM5PE;B000PSQXXU
Watches.txt	13	B0002LYEJK;B000UMK62U;B000SQM5PE;B000PSQXXU
~\$2401-JavaPD-课程实验4-学号-姓名.docx	14	B0002M9RI2;B000UMK62U;B000SQM5PE;B000PSQXXU
项目说明.md	15	B0002T5V2Q;B000UMK62U;B000SQM5PE;B000PSQXXU
> External Libraries	16	B0002UD438;B000UMK62U;B000SQM5PE;B000PSQXXU
Scratches and Consoles	17	B0002UD43S;B000UMK62U;B000SQM5PE;B000PSQXXU
	18	B00062N43M;B000UMK62U;B000SQM5PE;B000PSQXXU
	19	B0006AAS5Q;B000UMK62U;B000SQM5PE;B000PSQXXU
	20	B0006IHHMU;B000UMK62U;B000SQM5PE;B000PSQXXU
	21	B0007P4B6M;B000UMK62U;B000SQM5PE;B000PSQXXU
	22	B0007ZF7PQ;B000UMK62U;B000SQM5PE;B000PSQXXU
	23	B00080L2Z0;B000UMK62U;B000SQM5PE;B000PSQXXU
	24	B0009352E6;B000UMK62U;B000SQM5PE;B000PSQXXU

## **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）

TranslatorDemo.java

主要实现 GUI 界面和翻译功能的调用。

类成员变量：

```
public class TranslatorDemo extends JFrame {

 private JTextArea chineseTextArea; // 中文文本区域

 private JTextArea englishTextArea; // 英文文本区域

 private JButton chToEngButton; // 中译英按钮

 private JButton engToChButton; // 英译中按钮

 private JComboBox<String> translationMethodCombo; // 翻译方式选择

}
```

这些成员变量用于构建 GUI 界面的主要组件。

构造函数实现：

```
public TranslatorDemo() {

 // 设置窗口标题和关闭操作
 setTitle("中英互译程序");

 setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

 setLayout(new BorderLayout(10, 10));

 ((JPanel) getContentPane()).setBorder(new EmptyBorder(10, 10, 10, 10));

}
```

```
// 创建顶部的翻译方式选择
JPanel topPanel = new JPanel();

translationMethodCombo = new JComboBox<>(new String[]{"预设翻译", "在线 API 翻译", "大模型翻译"});

topPanel.add(new JLabel("选择翻译方式: "));
topPanel.add(translationMethodCombo);
add(topPanel, BorderLayout.NORTH);
}
```

构造函数中：

1. 设置窗口基本属性
2. 使用 BorderLayout 布局
3. 添加边距美化界面
4. 创建翻译方式选择下拉框

界面布局实现：

```
// 创建中间的主面板
JPanel mainPanel = new JPanel(new GridLayout(2, 1, 10, 10));

// 中文面板
JPanel chinesePanel = new JPanel(new BorderLayout(5, 5));
chineseTextArea = new JTextArea(5, 40);
chineseTextArea.setLineWrap(true);
chineseTextArea.setWrapStyleWord(true);
chToEngButton = new JButton("中译英");
JScrollPane chScrollPane = new JScrollPane(chineseTextArea);
chinesePanel.add(new JLabel("中文:"), BorderLayout.NORTH);
chinesePanel.add(chScrollPane, BorderLayout.CENTER);
chinesePanel.add(chToEngButton, BorderLayout.SOUTH);
```

界面布局特点：

1. 使用 GridLayout 将界面分为上下两部分
2. 每个文本区域都添加了滚动条支持
3. 设置文本自动换行
4. 清晰的标签指示

翻译功能实现：

```
private void translate(boolean isChToEng) {
 String text = isChToEng ? chineseTextArea.getText() : englishTextArea.getText();
 String result = "";

 switch(translationMethodCombo.getSelectedIndex()) {
 case 0: // 预设翻译
 result = presetTranslation(text, isChToEng);
 break;
```

```

 case 1: // 在线 API 翻译
 result = apiTranslation(text, isChToEng);
 break;

 case 2: // 大模型翻译
 result = aiModelTranslation(text, isChToEng);
 break;
 }

 if (isChToEng) {
 englishTextArea.setText(result);
 } else {
 chineseTextArea.setText(result);
 }
}

```

翻译方法说明：

1. 参数 `isChToEng` 决定翻译方向
2. 根据选择的翻译方式调用不同的翻译方法
3. 将结果显示在对应的文本区域

预设翻译实现：

```

private String presetTranslation(String text, boolean isChToEng) {
 // 预设的翻译对
 String preset1_ch = "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。";
 String preset1_en = "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.";

 // 简单的文本比较
 text = text.trim();

 if (isChToEng) {
 if (text.equals(preset1_ch)) {
 return preset1_en;
 }
 } else {
 if (text.equals(preset1_en)) {
 return preset1_ch;
 }
 }

 return "未找到匹配的预设翻译";
}

```

预设翻译特点:

1. 存储预定义的中英文对照
2. 使用精确匹配确保翻译准确性
3. 提供友好的错误提示

API 翻译实现:

```
private String apiTranslation(String text, boolean isChToEng) {
 return BaiduTranslator.translate(
 text,
 isChToEng ? "zh" : "en",
 isChToEng ? "en" : "zh"
);
}
```

API 翻译说明:

1. 调用 `BaiduTranslator` 的静态方法
2. 自动处理源语言和目标语言
3. 返回 API 翻译结果

大模型翻译实现:

```
private String aiModelTranslation(String text, boolean isChToEng) {
 // 使用两个大模型 API 进行翻译
 String ernieResult = AIModelTranslator.translateWithErnie(text, isChToEng);
 String qianwenResult = AIModelTranslator.translateWithQianwen(text, isChToEng);

 return String.format("文心一言译文: \n%s\n\n 通义千问译文: \n%s",
 ernieResult, qianwenResult);
}
```

大模型翻译特点:

1. 同时调用两个大模型 API
2. 格式化显示两个翻译结果
3. 方便用户对比不同结果

### 3.2 BaiduTranslator.java

百度翻译 API 的实现类。

```
public static String translate(String text, String from, String to) {
 try {
 // 构建 API 请求参数
 String salt = String.valueOf(System.currentTimeMillis());
 String sign = generateSign(text, salt);

 // 发送 HTTP 请求
```

```

 URL url = new URL(API_URL);
 HttpURLConnection conn = (HttpURLConnection) url.openConnection();

 // 设置请求方法和参数
 conn.setRequestMethod("POST");
 conn.setDoOutput(true);

 // 处理响应
 BufferedReader reader = new BufferedReader(
 new InputStreamReader(conn.getInputStream(), "UTF-8"));
 String response = reader.readLine();

 // 解析 JSON 响应
 return parseResponse(response);
 } catch (Exception e) {
 return "翻译失败: " + e.getMessage();
 }
}

```

实现说明:

1. 使用静态方法便于调用
2. 完整的错误处理机制
3. 支持 UTF-8 编码
4. JSON 响应解析

### 3.3 AIModelTranslator.java

AI 大模型翻译的实现类。

```

public static String translateWithErnie(String text, boolean isChToEng) {
 try {
 // 构建 API 请求
 String requestBody = buildErnieRequest(text, isChToEng);

 // 发送请求到文心一言 API
 HttpURLConnection conn = createConnection(ERNIE_API_URL);
 sendRequest(conn, requestBody);

 // 处理响应
 String response = getResponse(conn);
 return parseErnieResponse(response);
 } catch (Exception e) {
 return "文心一言翻译失败: " + e.getMessage();
 }
}

```

```
public static String translateWithQianwen(String text, boolean isChToEng) {
 try {
 // 类似的实现逻辑
 // ...
 } catch (Exception e) {
 return "通义千问翻译失败: " + e.getMessage();
 }
}
```

实现特点:

1. 模块化的 API 请求处理
2. 统一的错误处理方式
3. 支持两个大模型 API
4. 清晰的响应解析

运行结果

选择翻译方式: 预设翻译

中文:  
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。

中译英

英文:  
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.

英译中



## 1. 代码结构

BaiduTranslator.java 实现了百度翻译 API 的调用，主要包含：

- 配置信息（API URL、APP ID、密钥）
- 翻译方法（translate）
- 签名生成方法（generateSign）

## 2. 核心代码解释

### 2.1 配置信息

```
``java
private static final String BAIDU_API_URL = "http://api.fanyi.baidu.com/api/trans/vip/translate";
private static final String APP_ID = "20241206002221399";
private static final String SECRET_KEY = "CV4wPA8k2BmgLB9I90Rq";
...
```

这些常量用于存储 API 调用所需的基本配置。

### 2.2 翻译方法实现

```
``java
public static String translate(String text, String from, String to) {
 try {
 // 生成 salt 和 sign
 String salt = String.valueOf(System.currentTimeMillis());
 String sign = generateSign(text, salt);

 // 构建请求 URL
 String urlStr = String.format("%s?q=%s&from=%s&to=%s&appid=%s&salt=%s&sign=%s",
 BAIDU_API_URL,
 URLEncoder.encode(text, "UTF-8"),
 from,
 to,
 APP_ID,
 salt,
 sign);

 // 发送 GET 请求
 URL url = new URL(urlStr);
 HttpURLConnection conn = (HttpURLConnection) url.openConnection();
 conn.setRequestMethod("GET");

 // 读取响应
 BufferedReader reader = new BufferedReader(
 new InputStreamReader(conn.getInputStream(), StandardCharsets.UTF_8));
 StringBuilder response = new StringBuilder();
 }
}
```

```

String line;
while ((line = reader.readLine()) != null) {
 response.append(line);
}
reader.close();

// 解析响应
String responseStr = response.toString();
int dstIndex = responseStr.indexOf("\"dst\":\"");
if (dstIndex != -1) {
 dstIndex += 7;
 int endIndex = responseStr.indexOf("\"", dstIndex);
 if (endIndex != -1) {
 return responseStr.substring(dstIndex, endIndex);
 }
}
return "翻译失败: 未找到翻译结果";
} catch (Exception e) {
 e.printStackTrace();
 return "翻译出错: " + e.getMessage();
}
}
...

```

方法说明:

#### 1. 参数说明

- text: 待翻译的文本
- from: 源语言 (zh/en)
- to: 目标语言 (en/zh)

#### 2. 实现步骤

- 生成时间戳作为 salt
- 生成签名 sign
- 构建带参数的 URL
- 发送 GET 请求
- 读取并解析响应

#### 2.3 签名生成方法

```
```java
```

```

private static String generateSign(String text, String salt) {
    try {
        String str = APP_ID + text + salt + SECRET_KEY;
        MessageDigest md = MessageDigest.getInstance("MD5");
        byte[] bytes = md.digest(str.getBytes(StandardCharsets.UTF_8));
    }
}

```

```
        StringBuilder sign = new StringBuilder();  
        for (byte b : bytes) {  
            sign.append(String.format("%02x", b & 0xff));  
        }  
        return sign.toString();  
    } catch (Exception e) {  
        e.printStackTrace();  
        return "";  
    }  
}  
...  

```

签名生成说明:

1. 拼接字符串: APP_ID + 原文 + salt + 密钥
2. 使用 MD5 算法生成签名
3. 将字节转换为 16 进制字符串

3. 特点说明

3.1 请求方式

- 使用 GET 方法发送请求
- URL 参数包含所有必要信息
- 使用 UTF-8 编码处理中文

3.2 响应处理

- 简单的字符串解析方式
- 提取 "dst" 字段的值
- 异常情况返回错误信息

3.3 错误处理

- 捕获并处理网络异常
- 处理响应解析异常
- 提供友好的错误提示

选择翻译方式：

在线API翻译

中文：

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。

中译英

英文：

For 41 years since its establishment, Shenzhen University has adhered to the school motto of "self-reliance, self-discipline, and self-improvement", closely following the special zone, striving for reform and rapid development, and making important contributions to the development of the special zone and the modernization of the country.

英译中

大模型翻译

SiliconFlow 中转通义千问翻译方法

```
```java
public static String translateWithSilicon(String text, String from, String to) {
 try {
 // 1. 输入验证
 if (text == null || text.trim().isEmpty()) {
 return "翻译失败：文本不能为空";
 }

 // 2. 构建 system 提示词
 String systemPrompt = String.format(
 "你是一个专业的翻译助手。请将用户输入的%s 文本翻译成%s，只返回翻译结果，不要有任何解释。",
 getLanguageName(from), getLanguageName(to)
);

 // 3. 构建 API 请求体
 String requestBody = String.format(
 "{ " +
 "\"model\": \"glm-4\", " +
 "\"messages\": [" +
 "{ \"role\": \"system\", \"content\": \"%s\" }, " +
 "{ \"role\": \"user\", \"content\": \"%s\" }] }",
 systemPrompt, text
);
 } catch (Exception e) {
 return "翻译失败：未知错误";
 }
}
```

```

 "{ \"role\": \"user\", \"content\": \"%s\"} " +
 "], " +
 "\"temperature\":0.7, " +
 "\"top_p\":0.95, " +
 "\"presence_penalty\":0, " +
 "\"frequency_penalty\":0, " +
 "\"stream\":false " +
 "}",
 systemPrompt.replace("\\", "\\\""),
 text.replace("\\", "\\\"")
);

 // 4. 发送HTTP 请求
 URL url = new URL(SILICON_API_URL);
 HttpURLConnection conn = (HttpURLConnection) url.openConnection();
 conn.setRequestMethod("POST");
 conn.setRequestProperty("Content-Type", "application/json");
 conn.setRequestProperty("Authorization", "Bearer " + SILICON_API_KEY);
 conn.setRequestProperty("Accept", "application/json");
 conn.setDoOutput(true);
 conn.setConnectTimeout(30000);
 conn.setReadTimeout(30000);

 // 5. 写入请求数据
 try (OutputStreamWriter writer = new OutputStreamWriter(conn.getOutputStream(),
 StandardCharsets.UTF_8)) {
 writer.write(requestBody);
 writer.flush();
 }

 // 6. 处理响应
 int responseCode = conn.getResponseCode();
 if (responseCode != 200) {
 StringBuilder errorResponse = new StringBuilder();
 try (BufferedReader br = new BufferedReader(
 new InputStreamReader(
 responseCode >= 400 ? conn.getErrorStream() : conn.getInputStream(),
 StandardCharsets.UTF_8
)
)) {
 String line;
 while ((line = br.readLine()) != null) {
 errorResponse.append(line);
 }
 }
 }

```

```

 }

 return "翻译失败: 服务器返回错误 " + responseCode + "\n" + errorResponse.toString();
}

// 7. 读取响应数据
StringBuilder response = new StringBuilder();
try (BufferedReader br = new BufferedReader(
 new InputStreamReader(conn.getInputStream(), StandardCharsets.UTF_8))) {
 String line;
 while ((line = br.readLine()) != null) {
 response.append(line);
 }
}

// 8. 解析响应 JSON
String responseStr = response.toString();
int contentStart = responseStr.indexOf("\"content\":") + 11;
int contentEnd = responseStr.lastIndexOf("\"");
if (contentStart >= 11 && contentEnd != -1 && contentEnd > contentStart) {
 return responseStr.substring(contentStart, contentEnd)
 .replace("\\n", "\n")
 .replace("\\\"", "\"")
 .replace("\\\\", "\\")
 .trim();
}

return "翻译失败: 无法解析响应";

} catch (Exception e) {
 e.printStackTrace();
 return "翻译失败: " + e.getMessage();
}
}
...

```

#### 参数:

- ``text``: 要翻译的文本
- ``from``: 源语言代码 (如"zh")
- ``to``: 目标语言代码 (如"en")

#### 实现细节:

1. 输入验证
2. 构建 system 提示词
3. 构建 API 请求体

4. 发送 HTTP 请求

5. 处理响应结果

文心一言翻译方法

```
```java
public static String translateWithErnie(String text, boolean isChToEng) {
    try {
        // 1. 输入验证
        if (text == null || text.trim().isEmpty()) {
            return "翻译失败：文本不能为空";
        }

        // 2. 构建提示词
        String prompt = isChToEng ?
            "请将以下中文翻译成英文： " + text :
            "请将以下英文翻译成中文： " + text;

        // 3. 构建请求体
        String requestBody = String.format(
            "{\\"messages\\": [{\\"role\\":\\"user\\",\\"content\\":\\"%s\\"}]}",
            prompt.replace("\\", "\\\"")
        );

        // 4. 发送请求和处理响应
        URL url = new URL(ERNIE_API_URL);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("POST");
        conn.setRequestProperty("Content-Type", "application/json");
        conn.setRequestProperty("Authorization", "Bearer " + ERNIE_API_KEY);
        conn.setDoOutput(true);
        conn.setConnectTimeout(30000);
        conn.setReadTimeout(30000);

        // 5. 写入请求数据
        try (OutputStreamWriter writer = new OutputStreamWriter(conn.getOutputStream(),
StandardCharsets.UTF_8)) {
            writer.write(requestBody);
        }

        // 6. 读取响应
        StringBuilder response = new StringBuilder();
        try (BufferedReader reader = new BufferedReader(
            new InputStreamReader(conn.getInputStream(), StandardCharsets.UTF_8))) {

```

```

        String line;

        while ((line = reader.readLine()) != null) {
            response.append(line);
        }
    }

    // 7. 解析响应
    String responseStr = response.toString();
    int resultStart = responseStr.indexOf("\"result\":") + 9;
    int resultEnd = responseStr.indexOf("\"", resultStart);
    if (resultStart >= 9 && resultEnd != -1) {
        return responseStr.substring(resultStart, resultEnd)
            .replace("\\n", "\n")
            .replace("\\\"", "\"")
            .replace("\\\\", "\\");
    }

    return "翻译失败：无法解析响应";

} catch (Exception e) {
    e.printStackTrace();
    return "文心一言翻译出错：" + e.getMessage();
}
}
...

```

参数：

- `text`：要翻译的文本
- `isChToEng`：是否为中译英

实现细节：

1. 输入验证
2. 构建翻译提示词
3. 发送 API 请求
4. 解析响应结果

关键实现细节

HTTP 请求处理

```

```java
URL url = new URL(API_URL);
URLConnection conn = (URLConnection) url.openConnection();
conn.setRequestMethod("POST");

```



```
conn.setRequestProperty("Content-Type", "application/json");
conn.setRequestProperty("Authorization", "Bearer " + API_KEY);
conn.setDoOutput(true);
conn.setConnectTimeout(30000); // 30 秒超时
conn.setReadTimeout(30000);
...
```

错误处理机制

```
```java
try {
    // API 调用代码
} catch (java.net.SocketTimeoutException e) {
    return "翻译失败: 连接超时, 请检查网络";
} catch (java.net.UnknownHostException e) {
    return "翻译失败: 无法连接到翻译服务器";
} catch (Exception e) {
    e.printStackTrace();
    return "翻译失败: " + e.getMessage();
}
...
```

响应解析

```
```java
// 提取翻译结果
int contentStart = responseStr.indexOf("\"content\":") + 11;
int contentEnd = responseStr.indexOf("\"", contentStart);
if (contentStart >= 11 && contentEnd != -1 && contentEnd > contentStart) {
 String result = responseStr.substring(contentStart, contentEnd)
 .replace("\\n", "\n")
 .replace("\\\"", "\"")
 .replace("\\\\", "\\")
 .trim();

 if (result.isEmpty()) {
 return "翻译失败: 返回结果为空";
 }

 return result;
}
}
```

运行结果：

翻译方式：

SiliconFlow翻译

源语言：

英文

目标语言：

中文

源文本

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.

译文

秉承“自立、自律、自强”的校训，该大学致力于服务深圳经济特区，展示中国的改革开放和高等教育领域的创新变革。

翻译完成

开始翻译

### **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）

#### 1. 服务器端代码解析 (ServerS.java)

##### 1.1 服务器核心代码：

```
public class ServerS {
 private static final int PORT = 12345;
 private static Map<String, PrintWriter> clients = new HashMap<>();
 public static void main(String[] args) {
 try (ServerSocket serverSocket = new ServerSocket(PORT)) {
 while (true) {
 Socket clientSocket = serverSocket.accept();
 new ClientHandler(clientSocket).start();
 }
 }
 }
}
```

代码说明：

- `PORT = 12345`: 服务器监听端口
- `clients`: 存储所有连接的客户端，键是客户端名称，值是输出流
- `serverSocket.accept()`: 等待并接受客户端连接
- `new ClientHandler(clientSocket).start()`: 为每个客户端创建新线程

1.2 客户端处理线程：

```
private static class ClientHandler extends Thread {
 private Socket socket;
 private BufferedReader in;
 private PrintWriter out;
 private String clientName;
 public void run() {
 try {
 in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
 out = new PrintWriter(socket.getOutputStream(), true);
 clientName = in.readLine();
 clients.put(clientName, out);

 String message;
 while ((message = in.readLine()) != null) {
 broadcast(clientName + ": " + message);
 }
 } finally {
 clients.remove(clientName);
 }
 }
}
```

代码说明：

- `socket`: 与客户端的连接
- `in`: 读取客户端消息的流
- `out`: 发送消息给客户端的流
- `clientName`: 客户端标识
- `clients.put`: 将新客户端添加到映射表
- `while` 循环: 持续读取客户端消息
- `clients.remove`: 客户端断开时清理资源

1.3 消息广播实现：

```
private void broadcast(String message) {
 synchronized (clients) {
 for (PrintWriter writer : clients.values()) {
 writer.println(message);
 }
 }
}
```

代码说明：

- **synchronized**: 确保线程安全
- **clients.values()**: 获取所有客户端的输出流
- **writer.println**: 发送消息给每个客户端

## 2. 客户端代码解析 (Client.java)

### 2.1 客户端基本结构:

```
public class Client {
 private static final String SERVER_IP = "localhost";
 private static final int SERVER_PORT = 12345;
 private String clientName;
 private Socket socket;
 private BufferedReader in;
 private PrintWriter out;
 private JFrame frame;
 private JTextArea messageArea;
 private JTextField inputField;
}
```

代码说明:

- **SERVER\_IP**: 服务器地址
- **SERVER\_PORT**: 服务器端口
- **socket**: 与服务器的连接
- **in/out**: 输入输出流
- **frame/messageArea/inputField**: GUI 组件

### 2.2 GUI 初始化:

```
public Client(String name) {
 frame = new JFrame(name + " - 聊天窗口");
 messageArea = new JTextArea(20, 50);
 inputField = new JTextField(50);
 inputField.addActionListener(e -> {
 out.println(inputField.getText());
 inputField.setText("");
 });
}
```

代码说明:

- **JFrame**: 创建主窗口
- **JTextArea**: 显示聊天消息
- **JTextField**: 输入消息
- **addActionListener**: 处理发送消息事件

### 2.3 服务器连接:

```
private void connectToServer() throws IOException {
 socket = new Socket(SERVER_IP, SERVER_PORT);
 in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
 out = new PrintWriter(socket.getOutputStream(), true);
 out.println(clientName);
}
```

```
new Thread(() -> {
 try {
 String message;
 while ((message = in.readLine()) != null) {
 messageArea.append(message + "\n");
 }
 } catch (IOException e) {
 messageArea.append("与服务器断开连接\n");
 }
}).start();
}
```

代码说明：

- new Socket: 连接到服务器
- out.println(clientName): 发送客户端标识
- new Thread: 创建消息接收线程
- messageArea.append: 显示接收到的消息

### 3. 运行说明

#### 3.1 启动顺序:

##### 1) 先运行 ServerS.java

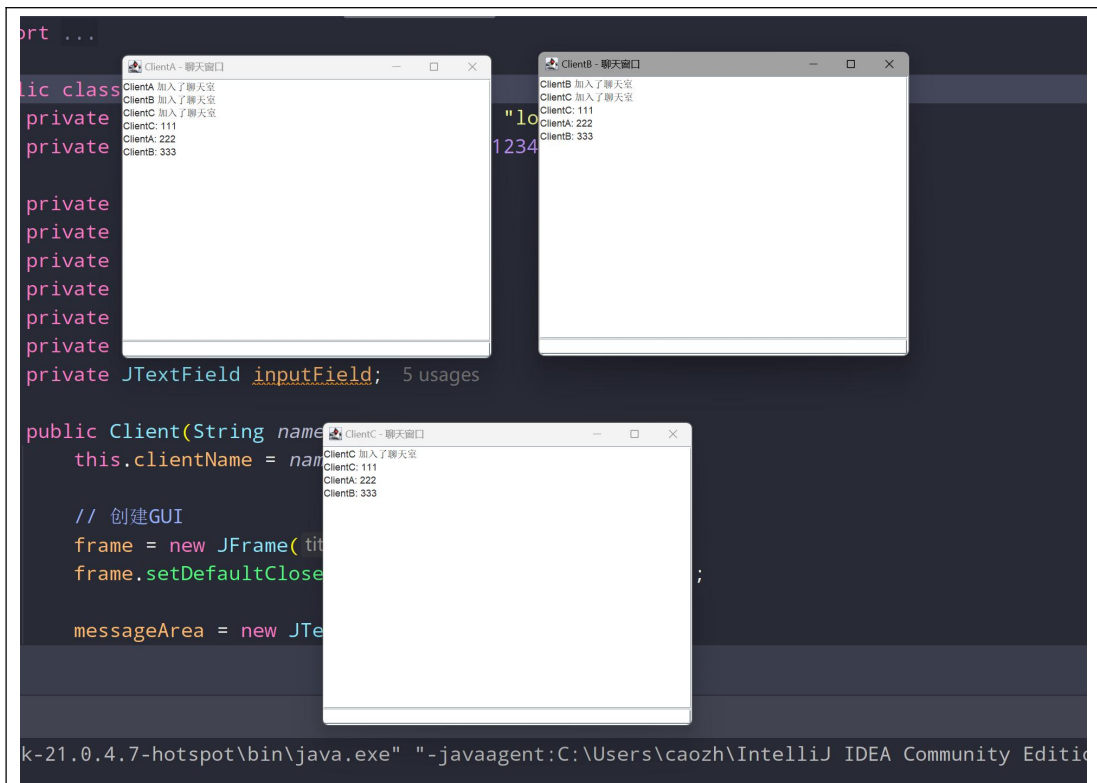
```
javac ServerS.java
java ServerS
```

##### 2) 再运行 Client.java

```
javac Client.java
java Client
```

#### 3.2 运行效果:

- 服务器启动后显示等待连接信息
- 客户端启动后自动打开三个聊天窗口
- 在任意窗口输入消息，按回车发送
- 所有窗口都能看到发送的消息



### (3.2). 利用数据报通信（UDP）实现题(1)中的要求。（10 分）

#### 1. 服务器端代码解析（UDPServer.java）

##### 1.1 基本结构和变量：

```
private static final int PORT = 12345;

private static Map<String, ClientInfo> clients = new HashMap<>();

private static DatagramSocket socket;
```

代码说明：

- PORT: 服务器监听的 UDP 端口号
- clients: 存储所有客户端信息的映射表
- socket: UDP 数据报套接字

##### 1.2 客户端信息类：

```
private static class ClientInfo {

 InetAddress address;

 int port;

 ClientInfo(InetAddress address, int port) {

 this.address = address;

 this.port = port;

 }

}
```

代码说明：

- address: 客户端的 IP 地址
- port: 客户端的端口号
- 用于存储每个连接客户端的网络信息

##### 1.3 主循环：

```
public static void main(String[] args) {

 try {

 socket = new DatagramSocket(PORT);

 byte[] receiveData = new byte[1024];

 while (true) {

 DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);

 socket.receive(receivePacket);

 // 处理数据...

 }

 }

}
```

代码说明💎💎

- 创建 UDP 套接字并绑定端口
- 创建接收缓冲区
- 循环接收数据包

- 对每个数据包进行处理

#### 1.4 消息处理:

```
private static void processMessage(String message, InetAddress address, int port) {
 String[] parts = message.split("\\|", 3);
 String type = parts[0];
 String clientName = parts[1];
 String content = parts.length > 2 ? parts[2] : "";

 switch (type) {
 case "JOIN": // 处理加入
 case "MESSAGE": // 处理消息
 case "LEAVE": // 处理离开
 }
}
```

代码说明:

- 解析消息格式: 类型|客户端名称|内容
- 根据消息类型进行不同处理
- 维护客户端列表
- 广播系统消息

#### 1.5 广播实现:

```
private static void broadcast(String message) {
 byte[] sendData = message.getBytes();
 for (Map.Entry<String, ClientInfo> entry : clients.entrySet()) {
 // 发送数据包给每个客户端
 }
}
```

代码说明:

- 将消息转换为字节数组
- 遍历所有客户端
- 创建并发送数据包

## 2. 客户端代码解析 (UDPClient.java)

### 2.1 基本结构和变量:

```
private static final String SERVER_IP = "localhost";
private static final int SERVER_PORT = 12345;
private String clientName;
private DatagramSocket socket;
private InetAddress serverAddress;
private JFrame frame;
private JTextArea messageArea;
private JTextField inputField;
private volatile boolean running;
```

代码说明:

- SERVER\_IP: 服务器 IP 地址
- SERVER\_PORT: 服务器端口



- `clientName`: 客户端标识
- `socket`: UDP 套接字
- 其他 GUI 组件

## 2.2 GUI 初始化:

```
public UDPClient(String name) {
 this.clientName = name;
 frame = new JFrame(name + " - 聊天窗口");
 // 初始化 GUI 组件
}
```

代码说明:

- 创建主窗口
- 设置窗口关闭处理
- 添加消息区域
- 添加输入框
- 设置窗口位置

## 2.3 启动客户端:

```
private void start() {
 frame.setVisible(true);
 running = true;

 try {
 socket = new DatagramSocket();
 serverAddress = InetAddress.getByName(SERVER_IP);
 sendMessage("JOIN", "");
 new Thread(this::receiveMessages).start();
 } catch (IOException e) {
 messageArea.append("连接失败: " + e.getMessage() + "\n");
 }
}
```

代码说明:

- 显示 GUI 窗口
- 初始化 UDP 套接字
- 发送加入消息
- 启动消息接收线程

## 2.4 消息发送:

```
private void sendMessage(String type, String content) {
 try {
 String message = type + "|" + clientName + "|" + content;
 byte[] sendData = message.getBytes();
 DatagramPacket sendPacket = new DatagramPacket(
 sendData,
 sendData.length,
 serverAddress,
 SERVER_PORT
);
 }
}
```

```

);
 socket.send(sendPacket);
} catch (IOException e) {
 messageArea.append("发送消息失败: " + e.getMessage() + "\n");
}
}
}

```

代码说明:

- 构造消息格式
- 创建数据包
- 发送到服务器

## 2.5 消息接收:

```

private void receiveMessages() {
 byte[] receiveData = new byte[1024];
 while (running) {
 try {
 DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
 socket.receive(receivePacket);

 // 处理接收到的消息

 } catch (IOException e) {
 if (running) {
 // 处理错误
 }
 }
 }
}
}

```

代码说明:

- 循环接收消息
- 解析消息内容
- 在 GUI 中显示消息

## 3. 程序特点

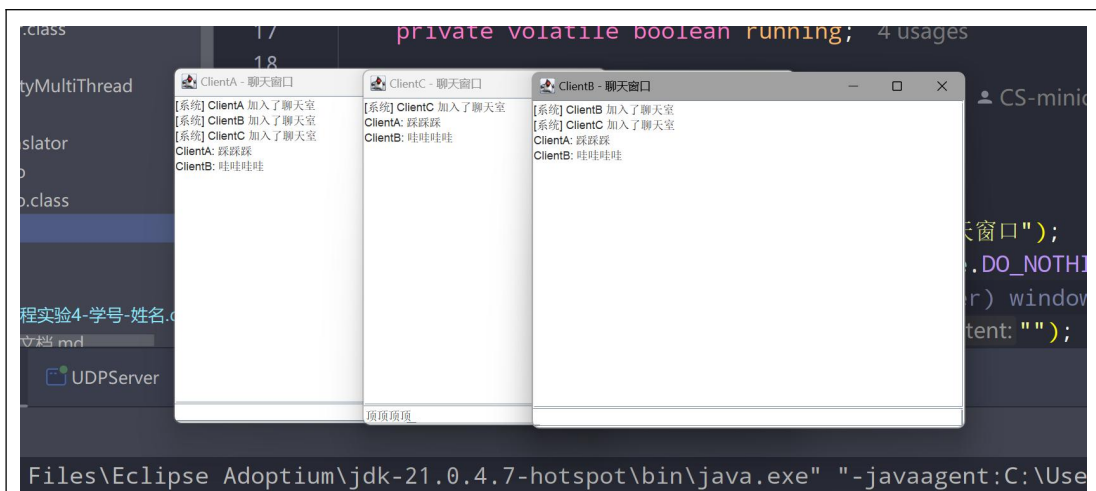
### 3.1 UDP 特性:

- 无连接通信
- 消息可能丢失
- 消息可能乱序
- 通信开销小

### 3.2 程序功能:

- 支持多客户端聊天
- 系统消息提醒
- 客户端动态加入/退出
- 图形界面操作

## 运行结果



+++++

其他（例如感想、建议等等）。

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。