

深圳大学实验报告

课程名称: Java 程序设计

实验项目名称: 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: 毛斐巧

报告人: 吴嘉楷 学号: 2022150168 班级: 国际班

实验时间: 2024 年 11 月 26、12 月 3、10、17、24 日 (周二)

实验报告提交时间: 2024 年 12 月 6 日

教务部制

一、实验目的

1. 掌握 I/O 程序设计，能够读写本地文件等。
2. 熟练掌握图形界面程序设计。
3. 掌握网络通信协议及相关程序设计。

二、实验内容与要求

1. 数据解析和统计（40 分）

- <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。
- 在报告中应体现出程序截图、运行结果（如每个输出文件前 8 行的截图等）和简要文字说明。

（1.1）使用 Java 语言读取解压后的文件（Watches.txt），并得到以下文件（20 分）：

- **review.txt:** 每行 3 列，以分号作为分隔符，第 1 列是 userID，第 2 列是 productID，第 3 列是 summary；表示(user, product, summary)三元组。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：如果没有按照 userID 或 productID 从小到大排序扣 5 分。

（1.2）使用 Java 语言根据 review.txt 进行计算，并得到以下文件（20 分）：

- **userNeighborhood.txt:** 每行 11 列，以分号作为分隔符，第 1 列是 userID，第 2-10 列是与该用户最相似的 10 个用户的 userID，按相似度值从大到小排列，其中相似度是通过 review.txt 中的前两列计算得到的 Jaccard index 值。该文件中不同行之间的顺序，按照第 1 列的 userID 从小到大排列。注：如果没有使用多线程实现扣 5 分，如果没有按相似度从大到小排列扣 5 分。

2.编写 Java 应用程序，实现“Java 机考”的功能（“单机版 Java 简易机考程序”），包含单选题、多选题和判断题三种题型。（20 分）

在主线程中创建一个 Frame 类型的窗口，在该窗口中再创建一个线程 giveQuestion。

线程 giveQuestion 每隔 20 秒钟输出一个选择题（含 A,B,C,D 共 4 个选项，要求支持单选题和多选题，单选题用 radio button，多选题用 check box）或一个判断题（用 radio button），选择题和判断题混合着给出；用户输入答案并按提交按钮提交结果（达到 20 秒自动提交结果）；程序判断用户输入的答案是否正确（如果错选或漏选均得零分），并实时显示当前题目的正确答案、已经给出的题目的数量（分别给出单选题数量、多选题数量和判断题数量）、用户答对的数量（分别给出单选题数量、多选题数量和判断题数量）、用户的成绩和用户答题所花的总的时间。

如此循环 15 次，包括随机选择的 5 个单选题（每题 1 分）、随机选择的 5 个多选题（每题 2 分）和随机选择的 5 个判断题（每题 1 分），结束测试时给出最终成绩。

题库应至少包含 10 个单选题、10 个多选题和 10 个判断题。要求使用图形用户界面。

3.网络通信编程：（20 分）

（3.1）利用数据报通信方式编写一个程序，该程序生成两个客户端，一个服务器端，

两个客户端可以相互进行简短的文字交流。在报告中应有程序截图、完整的运行结果和简要文字说明。（15 分：数据报通信 3 分（如用套接字连接扣 2 分），两个客户端 2 分，一个服务器端 2 分，实现文字交流 2 分，程序注释和截图 2 分，运行结果截图 2 分，文字说明 2 分）

（3.2）编写 Java 应用程序，根据第 2 题“单机版 Java 简易机考程序”的要求，将之改为网络版。客户端和服务端建立套接字连接后，服务器端向客户端发送一个题目；客户端将答案发送给服务器端；服务器端判断客户端的答案是否正确。要求使用图形界面，其他要求同“单机版 Java 简易机考程序”。在报告中应有程序截图、完整的运行结果和简要文字说明。（5 分：套接字连接 2 分，客户端和服务端实现双向通信 3 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。

（2）报告正文：请在指定位置填写。本次实验需要单独提交源程序文件，请将源程序文件压缩成为一个 code.zip 包提交。共提交两个文件：实验报告.doc 和 code.zip。

（3）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。

（4）提交方式：请在 Blackboard 平台中提交。

（5）发现雷同，所有雷同者该次作业记零分。

三、实验过程及结果

1. 数据解析和统计编程（共 40 分）

（1.1）使用 Java 语言读取解压后的文件（Watches.txt），并得到以下文件（20 分）：

- **review.txt**: 每行 3 列，以分号作为分隔符，第 1 列是 userID，第 2 列是 productID，第 3 列是 summary；表示(user, product, summary)三元组。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：如果没有按照 userID 或 productID 从小到大排序扣 5 分。

A. 首先定义一个名为 Review 的类

它用于存储用户对商品的评论信息。类中包含三个私有字段，分别代表用户 ID、商品 ID 和评论概要。通过提供的构造函数，可以在创建 Review 对象时设置这些字段的值。此外，类还提供了三个公共的访问器方法，允许外部代码获取私有字段的值，从而访问评论的用户 ID、商品 ID 和评论概要。

代码如下：

```
class Review {  
    private String userId;    // 用户 ID  
    private String productId; // 商品 ID  
    private String summary;   // 评论  
    // 构造函数  
    public Review(String userId, String productId, String summary) {
```

```

        this.userId = userId;
        this.productId = productId;
        this.summary = summary;
    }

    public String getUserId() {
        return userId;
    } // 获取用户 ID

    public String getProductId() {
        return productId;
    } // 获取商品 ID

    public String getSummary() {
        return summary;
    } // 获取评论
}

```

B. 编写测试类主函数

`main` 方法首先定义了输入和输出文件的路径，并初始化了一个列表来存储评论对象。使用 `ArrayList` 来存储 `Review` 对象，因为它提供了灵活的添加和删除操作，适合在读取数据时动态地构建数据集合。

接着，使用 `BufferedReader` 逐行读取输入文件，根据每行的开头识别出用户 ID、商品 ID 和评论概要，并在读取到完整的评论信息后创建一个 `Review` 对象，将其添加到列表中。

读取完所有数据后，程序使用 `Collections.sort` 对列表中的评论对象按照用户 ID 和商品 ID 进行排序。首先按照用户 ID 排序，如果用户 ID 相同，则按照商品 ID 排序。

最后，程序使用 `BufferedWriter` 将排序后的评论数据写入到输出文件 `review.txt` 中，每个评论对象的数据以用户 ID、商品 ID 和评论概要的格式，用分号分隔，每条评论占一行。

在整个过程中，程序通过 `try-catch` 块来处理可能出现的 IO 异常。这是因为文件操作是出错率较高的操作，需要妥善处理可能的异常，以避免程序崩溃。

代码如下：

```

public class Demo1_1 {
    public static void main(String[] args) throws IOException {
        String inputFilePath = "Watches.txt"; // 输入文件路径
        String outputFilePath = "result_data/review.txt"; // 输出文件路径
        List<Review> reviews = new ArrayList<>(); // 使用 ArrayList 容器存储，便于插入、排序
        try (BufferedReader br = new BufferedReader(new FileReader(inputFilePath))) {
            String line;
            Review currentReview = null;
            String productId = "", userId = "", summary = "";
            while ((line = br.readLine()) != null) {
                if (line.startsWith("product/productId")) {
                    productId = line.split(":")[1];
                } else if (line.startsWith("review/userId")) {

```

```

        userId = line.split(":")[1];
    } else if (line.startsWith("review/summary")) {
        summary = line.split(":")[1];
        currentReview = new Review(userId, productId, summary); // 此时已经拿到完整的三元组
        reviews.add(currentReview); // 将该三元组添加到 reviews 容器
    }
}

} catch (IOException e) {
    e.printStackTrace();
}

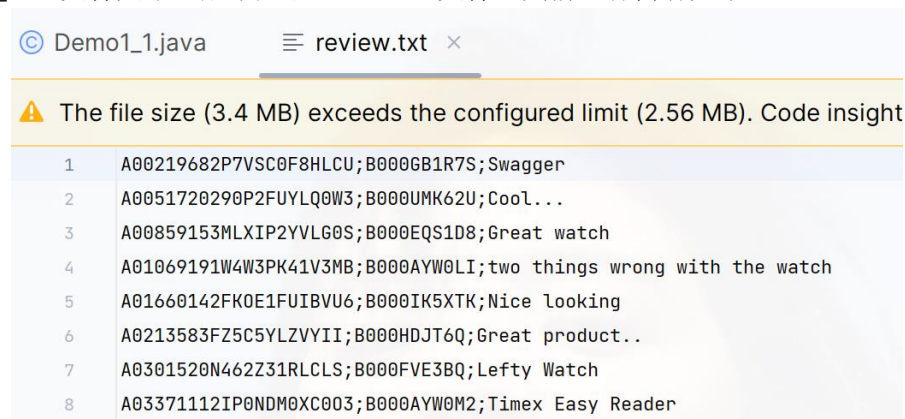
// 对 reviews 进行排序
Collections.sort(reviews, Comparator.comparing(Review::getUserId).thenComparing(Review::getProductId));

// 输出到 review.txt 文件中
try (BufferedWriter bw = new BufferedWriter(new FileWriter(outputFilePath))) {
    for (Review review : reviews) {
        bw.write(review.getUserId() + ";" + review.getProductId() + ";" + review.getSummary());
        bw.newLine();
    }
} catch (IOException e) {
    e.printStackTrace();
}
}
}

```

运行结果：

result_data 文件夹中生成了应该 review.txt 文件，其前 8 行内容如下：



Line	Content
1	A00219682P7VSC0F8HLCU;B000GB1R7S;Swagger
2	A0051720290P2FUYLQ0W3;B000UMK62U;Cool...
3	A00859153MLXIP2YVLG0S;B000EQS1D8;Great watch
4	A01069191W4W3PK41V3MB;B000AYW0LI;two things wrong with the watch
5	A01660142FK0E1FUIBVU6;B000IK5XTK;Nice looking
6	A0213583FZ5C5YLZVYII;B000HDJT6Q;Great product..
7	A0301520N462Z31RLCLS;B000FVE3BQ;Lefty Watch
8	A03371112IP0NDM0XC003;B000AYW0M2;Timex Easy Reader

图 1.1 review.txt 文件的部分内容

(1.2) 使用 Java 语言根据 review.txt 进行计算，并得到以下文件（20 分）：

- **userNeighborhood.txt:** 每行 11 列，以分号作为分隔符，第 1 列是 userID，第 2-11 列是与该用户最相似的 10 个用户的 userID，按相似度值从大到小排列，其中相似度是通过 review.txt 中的前两列计算得到的 Jaccard index 值。该文件中不同行之间的顺序，按照第 1 列的 userID 从小到大排列。注：如果没有使用多线程实现扣 5 分，如果没有按相似度从大到小排列扣 5 分。

A. 定义 UserSimilarity 内部类

UserSimilarity 的静态内部类用于表示两个用户之间的相似度信息，包含两个私有且不可变的成员变量：userId 用于存储另一个用户的 ID，similarity 用于存储这两个用户之间的相似度值。

类中提供了一个构造函数，允许在创建 UserSimilarity 对象时初始化这两个变量。此外，还提供了两个公共的 getter 方法，分别用于获取 userId 和 similarity 的值。

代码如下：

```
static class UserSimilarity {  
    private final String userId;    // 用户 ID  
    private final double similarity; // 相似度  
    // 构造函数  
    public UserSimilarity(String userId, double similarity) {  
        this.userId = userId;  
        this.similarity = similarity;  
    }  
    // getter 方法  
    public String getUserId() {  
        return userId;  
    }  
    public double getSimilarity() {  
        return similarity;  
    }  
}
```

B. 从文件中读取数据，并存储到 Map 中

定义一个 readUserReviews 方法，用于从 review.txt 文件中读取数据，并将每条评论与其对应的用户 ID 关联起来，最后返回一个 Map，其中键是用户 ID，值是该用户评论过的所有产品 ID 的集合。

代码如下：

```
private static Map<String, Set<String>> readUserReviews(String filePath) throws IOException {  
    Map<String, Set<String>> userReviews = new HashMap<>(); // 使用 HashMap 存储用户评论  
    try (BufferedReader br = new BufferedReader(new FileReader(filePath))) { // 使用缓冲读取器读取文件  
        String line;  
        while ((line = br.readLine()) != null) { // 逐行读取文件  
            String[] parts = line.split(";");  
            if (parts.length >= 2) {  
                String userId = parts[0];  
                String productId = parts[1];  
                userReviews.computeIfAbsent(userId, k -> new HashSet<>()).add(productId); // 将评论添加到对应的用户集合  
            }  
        }  
    }  
}
```

```

    }
    return userReviews;
}

```

C. 定义 calculateJaccardIndex 方法：计算 Jaccard 指数

calculateJaccardIndex 方法通过比较两个用户评论过的产品集合的交集和并集来计算得出所有用户对之间的 Jaccard 指数。

这个方法使用多线程来提高计算效率，因为 Jaccard 指数的计算是独立且可以并行处理的。使用 ExecutorService 来管理线程池，这样可以有效地利用系统资源，并行处理任务。

代码如下：

```

private static Map<String, List<UserSimilarity>> calculateJaccardIndex(Map<String, Set<String>> userReviews) {
    Map<String, List<UserSimilarity>> userSimilarities = new ConcurrentHashMap<>(); // 使用ConcurrentHashMap 存储结果
    ExecutorService executor = Executors.newFixedThreadPool(Runtime.getRuntime().availableProcessors()); // 创建线程池
    List<String> userIds = new ArrayList<>(userReviews.keySet()); // 获取所有用户的 ID
    // 遍历所有用户，计算每个用户的相似度
    for (int i = 0; i < userIds.size(); i++) {
        String userId1 = userIds.get(i);
        executor.submit(() -> { // 使用 Lambda 表达式创建任务
            List<UserSimilarity> similarities = new ArrayList<>();
            for (String userId2 : userIds) {
                if (!userId1.equals(userId2)) {
                    double jaccardIndex = computeJaccardIndex(userReviews.get(userId1), userReviews.get(userId2));
                    similarities.add(new UserSimilarity(userId2, jaccardIndex));
                }
            }
            userSimilarities.put(userId1, similarities);
        });
    }
    executor.shutdown(); // 等待所有任务完成
    // 等待所有任务完成或超时
    try {
        if (!executor.awaitTermination(1, TimeUnit.MINUTES)) {
            executor.shutdownNow();
        }
    } catch (InterruptedException e) {
        executor.shutdownNow();
        Thread.currentThread().interrupt();
    }
    return userSimilarities;
}

```

D. 定义 sortAndSelectTopUsers 方法：每个用户的相似度列表进行排序，并选择前 10 个

首先，使用 List 内置排序 API 对每个用户的相似度列表按相似度值进行降序排序，然

后如果列表长度超过 10，则截断列表，只保留前 10 个最相似的用户。

代码如下：

```
private static void sortAndSelectTopUsers(Map<String, List<UserSimilarity>> userSimilarities) {
    for (List<UserSimilarity> similarities : userSimilarities.values()) {
        similarities.sort(Comparator.comparingDouble(UserSimilarity::getSimilarity).reversed());
        if (similarities.size() > 10) { // 如果相似度列表的长度超过 10，则删除后面的元素
            similarities.subList(10, similarities.size()).clear();
        }
    }
}
```

E. 实现 writeResultsToFile 方法：将结果写入输出文件

首先，使用 `BufferedWriter` 缓存流打开指定路径的文件，准备写入数据。然后，从 `userSimilarities` 地图中获取所有 `userID`，并按字母顺序排序。接着，遍历排序后的 `userID` 列表，对于每个 `userID`，创建一个 `StringBuilder` 对象，初始值为当前用户 ID。获取当前用户 ID 对应的 `UserSimilarity` 列表，并将每个 `UserSimilarity` 的 `userId` 用分号分隔追加到 `StringBuilder` 中。最后，将构建好的字符串逐行写入文件。

代码如下：

```
private static void writeResultsToFile(Map<String, List<UserSimilarity>> userSimilarities, String filePath) throws IOException {
    try (BufferedWriter bw = new BufferedWriter(new FileWriter(filePath))) {
        List<String> sortedUserIds = new ArrayList<>(userSimilarities.keySet());
        Collections.sort(sortedUserIds); // 按 userID 排序
        // 将每行结果 (11 列) 写入文件
        for (String userId : sortedUserIds) {
            StringBuilder sb = new StringBuilder(userId);
            for (UserSimilarity similarity : userSimilarities.get(userId)) {
                sb.append(";").append(similarity.getUserId());
            }
            bw.write(sb.toString());
            bw.newLine();
        }
    }
}
```

F. 编写主函数

`main` 方法首先定义了输入和输出文件的路径，然后从输入文件中读取用户评论数据。接着，它计算每对用户的 Jaccard 指数，并对每个用户的相似度列表进行排序，选择最相似的 10 个用户。最后，将这些结果写入到输出文件中。

代码如下：


```

public static void main(String[] args) throws IOException {
    String inputFilePath = "result_data/review.txt"; // 输入文件路径
    String outputFilePath = "result_data/userNeighborhood.txt"; // 输出文件路径
    // 从文件中读取数据
    Map<String, Set<String>> userReviews = readUserReviews(inputFilePath);
    // 计算每对用户之间的 Jaccard 指数
    Map<String, List<UserSimilarity>> userSimilarities = calculateJaccardIndex(userReviews);
    // 排序并选择每个用户最相似的 10 个用户
    sortAndSelectTopUsers(userSimilarities);
    // 将结果写入输出文件
    writeResultsToFile(userSimilarities, outputFilePath);
}

```

运行结果：

在 result_data 文件夹中，生成了一个 userNeighborhood 文件，前 8 行的内容如下：



```

1 A05590483FA0IB9CFW4A9;A0MKKURM43HP;A2H748632P895H;A3835II91Z6PS3;AHR6FUT6ST4I;A3U2C9C0535M7Y;A3HW1XJ01YW6G;A3FR40ZLHH9XZ0;A2809ZSRZZ2035;A310174KNT55U6;ALKB
2 A100E1UHI8C5H3;A339VVX6UL49QH;A1ZT2WPN5G9BEQ;AHM3PMY06F2ZV;A3BNT0DV3037X2;ADJX4LQQ0QRDI;A28YR4WTY2UMAO;A114V9ZNM0889;A3E91PMEIKY4HJ;AFABRIUD792QT;A2IVF7Q4V2LL7e
3 A100Q0HHFA907A;A3GUCSN92RH42;A203GKW5ARJU66;A0C9KSNBBBVKW;A36FYKFION78D6;A1MWRDXK1A7MZ;AQJ10DMLUK16;A3RND4HV3EU9FW;AP55R0KQ5WAKM;A18F2FDTYH0GR0;AM355RQKR6IYW
4 A1013QWPRURXVR;A2W0F0UWAMOE97;A40CW08VZB9HP;A1A0T3FAZUSP9C;AV283AT9V1KTU;A203AZBR6TY5FL;A3SYWBBUZHSPQ;A1004R1PPN40VF;A3J2MTI0J088FV;A2M3T2YRXIJN08;A1016U4PXD143K
5 A102SLZLSHAXRE;AY8I43HJ98AXI;A1KHM1P3K351W;ASJ252T1X37NQ;A13TWHOMD5F7NP;A203GKW5ARJU66;A0C9KSNBBBVKW;A36FYKFION78D6;A1MWRDXK1A7MZ;AQJ10DMLUK16;A3RND4HV3EU9FW
6 A103MBXNOQLK2H;AACHNK8BRN98F;A34BLQ9LAP72X;A6Z04UVXWV06;A7ZP5JRQ1WJ38;A14QB0BUDWGCYY;A2DCA2MH0JHBKH;A1C9SXUSEXE6QU;A3YX504SKQLM7;A2YR630VTWE2M0;AQDE16C6FFW7U
7 A1007DMJ3LFI0;A1IPDM55MNU189;AYHE5F3EA4AM0;A88YBS4BBBHQ6;ANAAWXCQPARI;A203GKW5ARJU66;A0C9KSNBBBVKW;A36FYKFION78D6;A1MWRDXK1A7MZ;AQJ10DMLUK16;A3RND4HV3EU9FW
8 A10FEPQ46P6XI7;A39V3CX3IUIWV;A2PWQ6F6KXFXA;A142J3N77BV0XT;A3U0THQZSX6U6S;A2WQ55CH818JFA;A1Y4166YY2I7KC;A3JFUBRSAG8UIT;ABKK8267EVFF;A1TVL6J5020AIY;A3B3Z3NQACVWPB

```

图 1.2 userNeighborhood 文件的部分内容

2.编写 Java 应用程序，实现“Java 机考”的功能（“单机版 Java 简易机考程序”），包含单选题、多选题和判断题三种题型。（20 分）

在主线程中创建一个 Frame 类型的窗口，在该窗口中再创建一个线程 giveQuestion。

线程 giveQuestion 每隔 20 秒钟输出一个选择题（含 A,B,C,D 共 4 个选项，要求支持单选题和多选题，单选题用 radio button，多选题用 check box）或一个判断题（用 radio button），选择题和判断题混合着给出；用户输入答案并按提交按钮提交结果（达到 20 秒自动提交结果）；程序判断用户输入的答案是否正确（如果错选或漏选均得零分），并实时显示当前题目的正确答案、已经给出的题目的数量（分别给出单选题数量、多选题数量和判断题数量）、用户答对的数量（分别给出单选题数量、多选题数量和判断题数量）、用户的成绩和用户答题所花的总的时间。

如此循环 15 次，包括随机选择的 5 个单选题（每题 1 分）、随机选择的 5 个多选题（每题 2 分）和随机选择的 5 个判断题（每题 1 分），结束测试时给出最终成绩。

题库应至少包含 10 个单选题、10 个多选题和 10 个判断题。要求使用图形用户界面。

在报告中应有程序截图、完整的运行结果截图和详细的文字说明。

- 1) JavaTest 类，在主线程中创建的窗口类，继承 JFrame 类，包含分数，总时间，答题时间，单选、多选、判断题数量，单选、多选、判断题正确数量和答题状态 flag 等 int 类型变量；分数，总时间，答题时间，单选、多选、判断题数量，单选、多选、判断题正确数量标签等 JLabel 类对象；单选 A\B\C\D 选项、判断题 T、F 等 JRadioButton 类对象；多选 A\B\C\D 选项 JCheckBox 类对象；提交按钮 JButton 类对象；题目文本 print、答题结果 result 等 JTextArea 类对象；出题 giveQuestion 等 GiveQuestion 类对象；计时器 timer 等 Timer 类对象

```

1  import javax.swing.*;
2  import java.awt.*;
3  import java.awt.event.ActionEvent;
4  import java.awt.event.ActionListener;
5  import java.util.Random;
6
7  class JavaTest extends JFrame{//继承JFrame类,实现Runnable接口
8      int score,Total_time,time;//分数、总时间、答题时间
9      int sg,multi,tf;//单选,多选,判断数量
10     int T_sg,T_multi,T_tf;//单选,多选,判断正确数量
11     JLabel scoreLabel,Total_timeLabel,timeLabel;//分数、总时间、答题时间标签
12     JLabel singleLabel,multiLabel,tfLabel;//单选,多选,判断数量标签
13     JLabel T_singleLabel,T_multiLabel,T_tfLabel;//单选,多选,判断正确数量标签
14     JRadioButton ARadio,BRadio,CRadio,DRadio;//单选A,B,C,D选项
15     JCheckBox ABox,BBox,CBox,DBox;//多选A,B,C,D选项
16     JRadioButton T,F;//判断题按钮
17     JButton submitButton;//提交按钮
18     JTextArea print,result;//题目文本组件、答题结果
19     GiveQuestion giveQuestion;//出题类对象
20     Timer timer;//计时器
21     int flag;/*作为是否是显示答案时间的标志,
22     为0时,是答题时间,总时间正常++,
23     为1时,是显示答案时间,总时间不计数*/

```

JavaTest 类的构造方法:

首先先创建窗口界面,设置窗口大小、设置窗口标题、让窗口居中,在关闭窗口时自动停止程序,并为了后面设置组件坐标,取消默认居中布局。

然后设置开始考试界面,先将 JavaTest 创建的窗口设为不可见,然后用 new 方法调起开始考试界面类 StartFrame (StartFrame 类将在介绍完 JavaTest 类后介绍)。

再将分数、时间、单选、多选、判断等整数类型初始化为 0,随后创建一个出题线程,先将 GiveQuestion 类对象 giveQuestion 用 new 方法创建,然后创建一个 Thread 类对象 giveQuestionThread,参数为 giveQuestion,将线程启动。

然后创建提交按钮,用 new 方法设置按钮显示的文字,调用 addActionListener 方法,重写 actionPerformed 方法,设置监听,当提交按钮被按下时,调用 giveQuestion 的 Check 方法检查该题是否做对,并唤醒线程,给出下一题,将按钮设置大小和坐标,添加进窗口。

最后设置时间器 timer,每一秒触发一次,重写 actionPerformed 方法,每次计时器被触发,先将计时器暂停,然后将时间+1,如果 flag 为 0,说明是答题时间,将总时间+1,否则为显示答案时间,总时长不变,然后调用 Loading 方法加载时间,并刷新界面 repaint,将更新完的时间显示再窗口里,最后将计时器重启。

```

JavaTest(){//无参构造
    //创建窗口界面
    setSize( width: 680, height: 680); //设置窗口大小
    setTitle("Java机考"); //设置界面标题
    setLocationRelativeTo(null); //窗口居中
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //关闭窗口时结束程序运行
    setLayout(null); //取消默认居中布局
    //开始考试界面
    setVisible(false); //开始考试前不显示机考窗口
    new StartFrame( javaTest: this);
    //分数、时间、单选，多选，判断数量初始化
    score=Total_time=sg=multi=tf=0;
    time=T_sg=T_multi=T_tf=0;
    //创建giveQuestion线程
    giveQuestion=new GiveQuestion( javaTest: this);
    Thread giveQuestionThread=new Thread(giveQuestion);
    giveQuestionThread.start();
    //提交
    submitButton=new JButton( text: "提交");
    submitButton.addActionListener(new ActionListener() {
        //按下按钮后，判断是否做对题，然后继续下一题
        @Override
        public void actionPerformed(ActionEvent e) {
            giveQuestion.Check(); //判断是否做对题
            giveQuestionThread.interrupt(); //唤醒线程
        }
    });
    submitButton.setBounds( x: 290, y: 500, width: 100, height: 50); //给这个标签设置位置
    getContentPane().add(submitButton); //存入容器
    //flag初始化为0
    flag=0;
    //设置Timer，每1s触发一次
    timer = new Timer( delay: 1000, e -> {
        timer.stop();
        time++;
        if (flag == 0) Total_time++;
        Loading();
        repaint();
        timer.start();
    });
}

```

Loading 方法用于加载分数、时间、单选、多选、判断数量等组件：

它先调用 RemoveLoading 方法，将原本的组件移除；

然后创建分数标签，用 new 方法填入“Score:”文本和分数 score，然后用 setBounds 方法设置坐标和宽高，最后用 getContentPane().add()方法存入容器；

最后用同样的方式创建总时间、答题时间、单选、多选、判断、单选正确、多选正确、判断正确标签，不再赘述，详细见代码注释。

```

void Loading(){//加载分数,时间,单选,多选,判断数量
// 移除所有加载组件
RemoveLoading();
// 分数
scoreLabel=new JLabel( text: "Score:"+score);//将文本和分数填入标签
scoreLabel.setBounds( x: 10, y: 10, width: 100, height: 20);//给这个标签设置位置
getContentPane().add(scoreLabel);//存入容器
// 总时间
Total_timeLabel=new JLabel( text: "TotalTime:"+Total_time);//将文本和时间填入标签
Total_timeLabel.setBounds( x: 580, y: 10, width: 100, height: 20);///给这个标签设置位置
getContentPane().add(Total_timeLabel);//存入容器
// 答题时间
timeLabel=new JLabel( text: "Time:"+time);//将文本和时间填入标签
timeLabel.setBounds( x: 310, y: 10, width: 50, height: 20);///给这个标签设置位置
getContentPane().add(timeLabel);//存入容器
// 单选
singleLabel=new JLabel( text: "当前单选题数:"+sg);//将文本和单选题数填入标签
singleLabel.setBounds( x: 10, y: 580, width: 100, height: 20);//给这个标签设置位置
getContentPane().add(singleLabel);//存入容器
// 多选
multiLabel=new JLabel( text: "当前多选题数:"+multi);//将文本和多选题数填入标签
multiLabel.setBounds( x: 295, y: 580, width: 100, height: 20);//给这个标签设置位置
getContentPane().add(multiLabel);//存入容器
// 判断
tfLabel=new JLabel( text: "当前判断题数:"+tf);//将文本和判断题数填入标签
tfLabel.setBounds( x: 570, y: 580, width: 100, height: 20);//给这个标签设置位置
getContentPane().add(tfLabel);//存入容器
// 正确单选
T_singleLabel=new JLabel( text: "正确单选题数:"+T_sg);//将文本和正确单选题数填入标签
T_singleLabel.setBounds( x: 10, y: 610, width: 100, height: 20);//给这个标签设置位置
getContentPane().add(T_singleLabel);//存入容器
// 正确多选
T_multiLabel=new JLabel( text: "正确多选题数:"+T_multi);//将文本和正确多选题数填入标签
T_multiLabel.setBounds( x: 295, y: 610, width: 100, height: 20);//给这个标签设置位置
getContentPane().add(T_multiLabel);//存入容器
// 正确判断
T_tfLabel=new JLabel( text: "正确判断题数:"+T_tf);//将文本和正确判断题数填入标签
T_tfLabel.setBounds( x: 570, y: 610, width: 100, height: 20);//给这个标签设置位置
getContentPane().add(T_tfLabel);//存入容器
}

```

RemoveLoading 方法，用于移除上述 Loading 方法中的组件，更新界面：

先将 Loading 方法中的所有组件 remove 移除，由于第一次调用该方法时这些组件为空，为避免异常，所有 remove 方法用 try-catch 方法包围。然后 repaint 刷新界面。

```

void RemoveLoading(){// 移除加载组件
try{
    remove(scoreLabel);
    remove(Total_timeLabel);
    remove(timeLabel);
    remove(singleLabel);
    remove(multiLabel);
    remove(tfLabel);
    remove(T_singleLabel);
    remove(T_multiLabel);
    remove(T_tfLabel);
}catch (Exception e){}
repaint();
}

```


Single 方法，用于加载单选题的选项按钮组件：

首先调用 RemoveSingle 方法，移除单选组件，然后给 JRadioButton 类对象 new 方法，填入 A、B、C、D，然后创建 ButtonGroup 类对象，将四个 radiobutton 加入 group，将他们控制为单选。

最后于 Loading 中组件设置坐标、宽高，填入容器的方法一致，不再赘述。

```
void Single(){//加载单选按钮组件
    // 移除单选组件
    RemoveSingle();
    // 单选方法
    // 设置选项按钮
    ARadio=new JRadioButton( text: "A");
    BRadio=new JRadioButton( text: "B");
    CRadio=new JRadioButton( text: "C");
    DRadio=new JRadioButton( text: "D");
    // 设置坐标、大小
    ARadio.setBounds( x: 100, y: 400, width: 40, height: 20);
    BRadio.setBounds( x: 250, y: 400, width: 40, height: 20);
    CRadio.setBounds( x: 400, y: 400, width: 40, height: 20);
    DRadio.setBounds( x: 550, y: 400, width: 40, height: 20);
    getContentPane().add(ARadio);
    getContentPane().add(BRadio);
    getContentPane().add(CRadio);
    getContentPane().add(DRadio);
    //buttonGroup控制为单选
    ButtonGroup optionGroup = new ButtonGroup();
    optionGroup.add(ARadio);
    optionGroup.add(BRadio);
    optionGroup.add(CRadio);
    optionGroup.add(DRadio);
}
```

RemoveSingle 方法，用于移除单选组件，实现代码与 RemoveLoading 方法一致，不再赘述。

```
void RemoveSingle(){//移除单选组件
    try{
        remove(ARadio);
        remove(BRadio);
        remove(CRadio);
        remove(DRadio);
    }catch (Exception e){}
    repaint();
}
```

Multiple 方法，用于加载多选题按钮组件：

先调用 RemoveMultiple 方法，移除多选组件，然后设置选项按钮，设置坐标大小，填入容器，与上述 Single 一致，不再赘述。

```

void Multiple(){//加载多选按钮组件
    //移除多选组件
    RemoveMultiple();
    //设置选项按钮
    ABox=new JCheckBox( text: "A");
    BBox=new JCheckBox( text: "B");
    CBox=new JCheckBox( text: "C");
    DBox=new JCheckBox( text: "D");
    //设置坐标、大小
    ABox.setBounds( x: 100, y: 400, width: 40, height: 20);
    BBox.setBounds( x: 250, y: 400, width: 40, height: 20);
    CBox.setBounds( x: 400, y: 400, width: 40, height: 20);
    DBox.setBounds( x: 550, y: 400, width: 40, height: 20);
    getContentPane().add(ABox);
    getContentPane().add(BBox);
    getContentPane().add(CBox);
    getContentPane().add(DBox);
}

```

RemoveMultiple 方法，用于移除多选按钮组件，实现代码与 RemoveSingle 方法一致，不再赘述。

```

void RemoveMultiple(){//移除多选组件
    try{
        remove(ABox);
        remove(BBox);
        remove(CBox);
        remove(DBox);
    }catch (Exception e){}
    repaint();
}

```

TrueFalse 方法，用于加载判断题组件，只比 Single 方法少了两个按钮，其他完全一致，不再赘述。

```

void TrueFalse(){//加载判断按钮组件
    //移除判断组件
    RemoveTrueFalse();
    //设置选项按钮
    T=new JRadioButton( text: "A");
    F=new JRadioButton( text: "B");
    //buttonGroup控制为单选
    ButtonGroup optionGroup = new ButtonGroup();
    optionGroup.add(T);
    optionGroup.add(F);
    //设置坐标、大小
    T.setBounds( x: 250, y: 400, width: 40, height: 20);
    F.setBounds( x: 400, y: 400, width: 40, height: 20);
    getContentPane().add(T);
    getContentPane().add(F);
}

```

RemoveTrueFalse 方法，用于移除判断选项按钮组件，实现代码与 RemoveSingle 方法一致，不再赘述。

```
void RemoveTrueFalse(){// 移除判断组件
    try{
        remove(T);
        remove(F);
    }catch (Exception e){}
    repaint();
}
```

RemovePrint 方法，用于移除题目文本组件，实现代码与 RemoveSingle 方法一致，不再赘述，而文本的加载代码在 giveQuestion 中，稍后介绍。

```
void RemovePrint(){// 移除题目文本
    try{
        remove(print);
    }catch (Exception e){}
    repaint();
}
```

ShowAnswer 方法，用于显示当前题目的答题结果和答案，传入一个布尔类型变量 b，表示是否回答正确，一个字符串 ans，表示答案。

首先用 new 方法创建答案文本；

然后如果 b 为 true，说明回答正确，将 “Perfect! Answer:”和答案字符串 ans 存入文本，并设置为红色，告诉用户回答正确，如果 b 为 false，说明回答错误，将 “Next Try! Answer:”和答案字符串 ans 存入文本，并设置为红色，告诉用户回答错误；

接着设置文本坐标和大小，存入容器；

最后调用 Loading 方法重新加载组件，并 repaint 刷新界面。

```
void ShowAnswer(Boolean b,String ans){// 每答完一题，显示一次结果
    result=new JTextArea();
    if(b){//b==true, 回答正确
        result.setText("Perfect! Answer:"+ans);
        result.setForeground(Color.red);// 字体设为红色
    }
    else{//b==false, 回答错误
        result.setText("Next Try! Answer:"+ans);
        result.setForeground(Color.red);// 字体设为红色
    }
    result.setBounds( x: 0, y: 360, width: 680, height: 20);
    getContentPane().add(result);
    Loading();// 加载组件
    repaint();
}
```

RemoveShowAnswer 方法，用于移除上述题目答案组件，实现代码与 RemoveSingle 方法一致，不再赘述。

```
void RemoveShowAnswer(){// 移除题目答案组件
    try{
        remove(result);
    }catch (Exception e){}
    repaint();
}
```

- 2) StartFrame 类，作为开始考试界面窗口，包含一个 JButton 类对象 start 和有参构造，参数为 JavaTest 类对象 javaTest

有参构造：

先创建窗口，与 JavaTest 的构造方法一致；

然后设置开始按钮，先用 new 方法设置按钮显示的文字，调用 addActionListener 方法，重写 actionPerformed 方法，设置监听，当按钮被按下时，关闭开始考试窗口，并显示出机考窗口，开始答题；

最后设置为显示开始窗口。

```
class StartFrame extends JFrame{//开始界面
    5 usages
    JButton start;//开始考试按钮
    1 usage
    StartFrame(JavaTest javaTest){
        //创建窗口
        setSize( width: 680, height: 680); //设置窗口大小
        setTitle("Java机考");//设置界面标题
        setLocationRelativeTo(null); //窗口居中
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //关闭窗口时结束程序运行
        setLayout(null); //取消默认居中布局
        //设置背景图
        JLabel background = new JLabel(new ImageIcon( filename: "javaBgI.jpg"));
        background.setBounds( x: 0, y: -50, getWidth(), getHeight());
        getContentPane().add(background);
        //开始按钮
        start=new JButton( text: "开始考试");
        start.addActionListener(new ActionListener() {
            //按下开始考试后，关闭开始窗口，打开机考窗口
            @Override
            public void actionPerformed(ActionEvent e) {
                setVisible(false); //关闭开始窗口
                javaTest.setVisible(true); //打开机考窗口
            }
        });
        //设置按钮样式
        start.setFocusPainted(false);
        //设置按钮位置和大小，使其居中
        int x = (getWidth() - 100) / 2;
        int y = getHeight() - 120;
        start.setBounds(x, y, width: 100, height: 50);
        getContentPane().add(start); //存入容器
        //显示开始窗口
        setVisible(true);
    }
}
```

- 3) GiveQuestion 类，实现 Runnable 接口，用于出题，包含一个大小为 30 的字符串数组 question，表示题库，其中 0-9 为 10 道单选题题目，10-19 为 10 道多选题题目，20-29 为 10 道判断题题目，一个大小同样为 30 的字符串数组 answer，表示上述 30 道题对应的答案，两个 int 类型变量 p 和 lp，分别表示本次抽到的题目下标和上一次抽到的题目下标，一个布尔类型数组 visited，表示该题是否被访问过，一个 JavaTest 类对象 javaTest。


```
class GiveQuestion implements Runnable {
    3 usages
    String[] questions = { //题库
        //8-9, 单选
        "Java嵌入式应用开发平台名称为:\n" + "A. 3DK\n" + "B. 32ME\n" + "C. 32SE\n" + "D. 32EE\n",
        "JAVA语言使用的字符集是:\n" + "A. ASCII\n" + "B. EBCDIC\n" + "C. Unicode\n" + "D. BCD",
        "JAVA语言中控制结构包括:\n" + "A. 顺序结构、选择结构、循环结构\n" + "B. 顺序结构、循环结构\n" + "C. 顺序结构、选择结构\n" + "D. 选择结构、循环结构",
        "若a是int型变量, 表达式a=(int)(25.8/353);执行后a的值为:\n" + "A. 1\n" + "B. 2\n" + "C. 3\n" + "D. 4\n",
        "下列哪个类声明是正确的?\n" + "A. abstract final class H1 {}\n" + "B. abstract private move() {} \n" + "C. protected private number;\n" + "D. public abstract class Car() \n",
        "以下哪项符合对象和类的关系的是:\n" + "A. 人和老虎\n" + "B. 书和汽车\n" + "C. 父亲和儿子\n" + "D. 汽车和交通工具\n",
        "下面关于Java中类的说法哪个是不正确的?\n" + "A. 类中只能有变量定义、常量定义和成员方法的定义, 不能包含" + "x3"; "这样的语句。 \n" + "B. 构造函数是类中的特殊方法\n" + "C. 主类一定要声明为public。 \n" + "D. 一个Java文件中可以有多个class定义。 \n",
        "下列哪项是Java编程所必须的默认引用包?\n" + "A. java.sys包\n" + "B. java.lang包\n" + "C. java.net包\n" + "D. 以上都不是\n",
        "在Java中main()的返回值是:\n" + "A. String\n" + "B. int\n" + "C. char\n" + "D. void\n",
        "构造方法相同时被调用\n" + "A. 类定义时\n" + "B. 创建对象时\n" + "C. 调用对象方法时\n" + "D. 使用对象的变量时\n",
        //10-19, 多选
        "下面哪个单词是Java语言的关键字?\n" + "A. Float\n" + "B. this\n" + "C. string\n" + "D. new\n",
        "Java语言有下面一些特点:\n" + "A. 面向对象\n" + "B. 可移植\n" + "C. 不安全性\n" + "D. 多线程以及动态性\n",
        "3DK包括哪些:\n" + "A. Java程序\n" + "B. Java运行环境\n" + "C. 一是Java工具\n" + "D. Java基础的类库\n",
        "给一个变量都拥有:\n" + "A. 名字\n" + "B. 类型\n" + "C. 大小\n" + "D. 长度\n",
        "若A为已定义的类名, 下列声明A类的对象a的语句中错误的是:\n" + "A. float A a;\n" + "B. public A a=A();\n" + "C. A a=new int();\n" + "D. static A a=new A();\n",
        "用来导入已定义好类名的语句错误的是:\n" + "A. a.main\n" + "B. import\n" + "C. public class\n" + "D. class\n",
        "关于类所包含的内容以下说法错误的是:\n" + "A. 属性和方法\n" + "B. 变量和方法\n" + "C. 变量和常量\n" + "D. 变量和属性\n",
        "protected可以修饰以下哪些:\n" + "A. 变量\n" + "B. 方法\n" + "C. 类\n" + "D. 接口\n",
        "public可以修饰以下哪些:\n" + "A. 变量\n" + "B. 方法\n" + "C. 类\n" + "D. 接口\n",
        "以下属于面向对象的特征的是:\n" + "A. 重载\n" + "B. 重写\n" + "C. 封装\n" + "D. 继承\n",
        //20-29, 判断题
        "静态初始化, 就是在定义数组的同时对数组的每个元素赋值。 \n" + "A. 正确\n" + "B. 错误\n",
        "二进制是由数字0和1组成的数字序列。 \n" + "A. 正确\n" + "B. 错误\n",
        "Java区分大小写的。 \n" + "A. 正确\n" + "B. 错误\n",
        "Java中比较两个数是否相等可以使用" + "=". \n" + "A. 正确\n" + "B. 错误\n",
        "二进制的组成两个索引。 \n" + "A. 正确\n" + "B. 错误\n",
        "用static修饰的变量是类变量。 \n" + "A. 正确\n" + "B. 错误\n",
        "在Java程序中, 可以使用protected来修饰一个类。 \n" + "A. 正确\n" + "B. 错误\n",
        "对于abstract类, 不能创建该类的对象。 \n" + "A. 正确\n" + "B. 错误\n",
        "有的类定义时可以不定义构造函数的, 所以构造函数不是必需的。 \n" + "A. 正确\n" + "B. 错误\n",
        "一个Java类可以有多个父类。 \n" + "A. 正确\n" + "B. 错误\n"
    };
};
```

```
String[] answer={ //答案
    "B", "C", "A", "B", "D", "D", "C", "B", "D", "B",
    "BD", "ABD", "BCD", "AB", "ABC", "ACD", "BCD", "AB", "ABCD", "CD",
    "A", "B", "B", "B", "A", "A", "A", "A", "B", "B"
};
20 usages
int p; //作为抽到的题目的下标
3 usages
int lp; //作为上次抽到的题目的下标
7 usages
Boolean[] visited; //布尔类, 判断该题是否访问过
58 usages
JavaTest javaTest; //机考类对象
```

有参构造, 参数为 JavaTest 类对象:

用 this 关键字将 javaTest 存入, 然后用 new 方法给 visited 数组开 30 个空间, 并用循环赋值为 false, 表示未访问过。

```
GiveQuestion(JavaTest javaTest){
    this.javaTest=javaTest;
    //visited一共30个元素, 并初始化为false, 表示未访问过
    visited=new Boolean[30];
    for(int i=0;i<30;i++) visited[i]=false;
}
```

Check 方法, 用于检查题目是否做对:

首先, 定义一个字符串变量 ans, 初始化为空, 表示用户的答题结果;

然后用 if-else 语句找题;

如果当前题目下标 p 小于 10, 说明是单选题, 分别调用 javaTest 的四个单选题对象的 isSelected 方法, 如果被选中, 就将对应的 A/B/C/D 存入字符串 ans, 然后用 equals 方法判断 ans 是否与答案 answer[p]一致, 如果一致, 说明答对, javaTest 的 score++, T_sg++, 并调用 javaTest 的 ShowAnswer 方法, 传入 true 和 answer[p], 否则, 传入 false 和 answer[p];

同理, 如果 p<20, 说明是多选题, 其余与上述单选一致, 如果 p<30, 说明是判断题,

其余也与上述单选一致。

```
void Check(){
    String ans=""; //用户提交的答案
    if(p<10){ //抽到单选
        if(javaTest.ARadio.isSelected()) ans+="A";
        if(javaTest.BRadio.isSelected()) ans+="B";
        if(javaTest.CRadio.isSelected()) ans+="C";
        if(javaTest.DRadio.isSelected()) ans+="D";
        //判断是否回答正确
        if(ans.equals(answer[p])){//回答正确
            javaTest.score++; //加1分
            javaTest.T_sg++; //单选正确数+1
            javaTest.ShowAnswer( b: true,answer[p]);
        }
        else javaTest.ShowAnswer( b: false,answer[p]); //回答错误
    }
    else if(p<20){ //抽到多选
        if(javaTest.ABox.isSelected()) ans+="A";
        if(javaTest.BBox.isSelected()) ans+="B";
        if(javaTest.CBox.isSelected()) ans+="C";
        if(javaTest.DBox.isSelected()) ans+="D";
        //判断是否回答正确
        if(ans.equals(answer[p])){//回答正确
            javaTest.score+=2; //加2分
            javaTest.T_multi++; //多选正确数+1
            javaTest.ShowAnswer( b: true,answer[p]);
        }
        else javaTest.ShowAnswer( b: false,answer[p]); //回答错误
    }
    else{//抽到判断
        if(javaTest.T.isSelected()) ans+="A";
        if(javaTest.F.isSelected()) ans+="B";
        //判断是否回答正确
        if(ans.equals(answer[p])){//回答正确
            javaTest.score++; //加1分
            javaTest.T_tf++; //判断正确数+1
            javaTest.ShowAnswer( b: true,answer[p]);
        }
        else javaTest.ShowAnswer( b: false,answer[p]); //回答错误
    }
}
```

重写 Runnable 接口的 run 方法，用于出题：

先创建一个 Random 类对象，用于生成随机数抽取题目；

然后循环 15 次，表示抽取 15 道题；

如果不是第一次提交，那么该线程需要先休息 2s，用于显示答案。先将 javaTest 的 flag 赋值为 1，表示此时是显示答案时间；然后将时间 time 归零；然后启动计时器 timer；然后调用 Loading 方法，加载里面的组件；然后 repaint 方法刷新界面；然后线程 sleep2s，用 try-catch 包围。

休息完后，调用 RemoveShowAnswer 方法移除显示答案组件；然后检查是否已经抽到 5 次单选、5 次多选、5 次判断，如果其中任意一种达到 5 次，按照题目要求不能再继续抽取对应的题，因此需要将所有对应的题的 visited 设置为 true，不让再访问；然后 while 循环，循环条件为 true，无限循环，在循环中 p 调用 random 的 nextInt(30)方法，抽取 30 以内的随机数，一直到该 p 对应的题目没被访问过为止。

抽取完题目后，将该题的 visited 设置为 true，表示已访问过；然后移除上一题 lp 对应

的题目的按钮；然后调用 `RemovePrint` 方法移除上一题的文本，并设置新文本；然后将现在抽到的题的题目类型，题号，题目内容存进文本，将该题的数量++，并调用对应的加载方法，加载出按钮；然后设置文本坐标、大小；将 `flag` 设置为 0，表示现在是答题时间，再将计时器 `timer` 用 `start` 方法启动，调用 `Loading` 加载组件，然后 `repaint` 刷新界面，让 `lp` 赋值为 `p`，然后线程 `sleep20s`，20s 后调用 `Check` 方法，表示 20s 后自动提交。

最后循环 15 次结束后，即考试结束，先将考试界面设为不可见，再用 `new` 方法打开结束界面 `EndFrame`，`EndFrame` 类将在稍后介绍。

```
@Override
public void run() {
    Random random=new Random();//创建一个随机数对象，用于抽取题目
    for (int i=1;i<=15;i++) { //一共出15道题
        //每提交一题，就暂停2s显示答案
        if(i!=1){
            javaTest.flag=1;//此时是显示答案时间
            javaTest.time=0;//当前题目时间归零
            javaTest.timer.start();//启动计时器
            javaTest.Loading();//加载分数，时间，单选，多选，判断数量
            javaTest.repaint();
            try {
                Thread.sleep( millis: 2000);
            }catch (Exception e1){}
            javaTest.RemoveShowAnswer();//移除显示答案组件
        }
        //检查同类型题目是否已经抽取5道
        if(javaTest.sg>=5){
            //如果已经出了5题，将所有单选设为已访问，防止再抽到单选
            for(int j=0;j<10;j++) visited[j]=true;
        }
        if(javaTest.multi>=5){
            //如果已经出了5题，将所有多选设为已访问，防止再抽到多选
            for(int j=10;j<20;j++) visited[j]=true;
        }
        if(javaTest.tf>=5){
            //如果已经出了5题，将所有判断设为已访问，防止再抽到判断
            for(int j=20;j<30;j++) visited[j]=true;
        }
        //只有没被选过的题能被选择
        while(true){
            p=random.nextInt( bound: 30);//30以内的随机数
            if(visited[p]==false) break;
        }
        //将当前抽到的题目设置为已访问过
        visited[p]=true;
        //移除上一题的按钮组件
        if(lp<10) javaTest.RemoveSingle();
        else if(lp<20) javaTest.RemoveMultiple();
        else javaTest.RemoveTrueFalse();
        //移除上一题文本，设置新文本
        javaTest.RemovePrint();
        javaTest.print=new JTextArea();
        javaTest.print.setEditable(false);
        javaTest.print.setLineWrap(true);
        //显示题目，并让抽到的题型数+1
        if(p<10){
```

```

        javaTest.print.setText("单选题\n"+i+. "+" +questions[p]);
        javaTest.sg++;
        javaTest.Single();//调用javaTest的单选方法
    }
    else if(p<20){
        javaTest.print.setText("多选题\n"+i+. "+" +questions[p]);
        javaTest.multi++;
        javaTest.Multiple();//调用javaTest的多选方法
    }
    else{
        javaTest.print.setText("判断题\n"+i+. "+" +questions[p]);
        javaTest.tf++;
        javaTest.TrueFalse();//调用javaTest的判断方法
    }
}
javaTest.print.setBounds( x: 150, y: 100, width: 400, height: 250);
javaTest.getContentPane().add(javaTest.print);
javaTest.flag=0;//此时是答题时间
javaTest.time=0;//当前题目时间归零
javaTest.timer.start();//启动计时器
javaTest.Loading();//加载分数,时间,单选,多选,判断数量
javaTest.repaint();
lp=p;
try { //20s自动提交
    Thread.sleep( millis: 20000);
    Check();
} catch (Exception e) {}
}
//15题出完后, 调取结束界面
javaTest.setVisible(false);//关闭考试界面
new EndFrame(javaTest);//打开结束界面
}

```

4) EndFrame 类, 包含 JLabel 类对象最终时间标签 final_time、最终成绩标签 final_score 和一个 JButton 类对象关闭按钮 end。

有参构造, 参数为 JavaTest 类对象 javaTest:

首先创建窗口, 与 StartFrame 类的构造方法一致; 然后创建时间标签, 设置坐标、大小, 存入容器; 然后创建成绩标签, 设置坐标、大小, 存入容器; 然后设置结束考试按钮, 先用 new 方法设置按钮显示的文字, 调用 addActionListener 方法, 重写 actionPerformed 方法, 设置监听, 当按钮被按下时, 关闭窗口, System.exit(0)结束程序。

```

EndFrame(JavaTest javaTest){
    //创建窗口
    setSize( width: 680, height: 680); //设置窗口大小
    setTitle("Java机考");//设置界面标题
    setLocationRelativeTo(null); //窗口居中
    setDefaultCloseOperation(3); //关闭窗口时结束程序运行
    setLayout(null); //取消默认居中布局
    //创建时间标签
    final_time=new JLabel( text: "Total_time:"+javaTest.Total_time);
    final_time.setBounds( x: 290, y: 210, width: 100, height: 20);
    getContentPane().add(final_time);
    //创建成绩标签
    final_score=new JLabel( text: "Score:"+javaTest.score);
    final_score.setBounds( x: 290, y: 230, width: 100, height: 20);
    getContentPane().add(final_score);
    //结束考试按钮
    end=new JButton( text: "结束考试");
    end.addActionListener(new ActionListener() {
        //按下结束考试后关闭界面
        略~
        @Override
        public void actionPerformed(ActionEvent e) {
            System.exit( status: 0); //结束程序
        }
    });
    end.setBounds( x: 290, y: 400, width: 100, height: 50); //给这个标签设置位置
    getContentPane().add(end); //存入容器
    //显示窗口
    setVisible(true);
}

```

5) 主线程，只需创建 JavaTest 类对象 javaTest，即创建机考窗口

```
public class Demo2 {  
    public static void main(String[] args) {  
        JavaTest javaTest=new JavaTest();//创建机考窗口  
    }  
}
```

运行结果：

(1) 开始界面



(2) 机考界面

A. 按下按钮提交



Java机考

Score:1

Time:2

TotalTime:19

判断题

1. 静态初始化,就是在定义数组的同时为数组的每个元素赋值。
A.正确
B.错误

Perfect! Answer:A

☒ A

☐ B

提交

当前单选题数:0

当前多选题数:0

当前判断题数:1

正确单选题数:0

正确多选题数:0

正确判断题数:1

B. 20s 自动提交

Java机考

Score:0

Time:7

TotalTime:27

多选题

2. 每一个变量都拥有:
A.名字
B.类型
C.大小
D.长度

☐ A

☐ B

☐ C

☐ D

提交

当前单选题数:0

当前多选题数:2

当前判断题数:0

正确单选题数:0

正确多选题数:0

正确判断题数:0

C. 单选题

Java机考

Score:0Time:1TotalTime:141

单选题

8. 以下选项符合对象和类的关系的是:
A.人和老虎
B.书和汽车
C.父亲和儿子
D.汽车和交通工具

☐ A

☐ B

☐ C

☐ D

提交

当前单选题数:1
正确单选题数:0

当前多选题数:5
正确多选题数:0

当前判断题数:2
正确判断题数:0

D. 多选题

Java机考

Score:0Time:3TotalTime:3

多选题

1. `protected`可以修饰以下哪项:
A.变量
B.方法
C.类
D.接口

☐ A

☐ B

☐ C

☐ D

提交

当前单选题数:0
正确单选题数:0

当前多选题数:1
正确多选题数:0

当前判断题数:0
正确判断题数:0

E. 判断题

Java机考

Score:0Time:4TotalTime:108

判断题

6. 用static修饰的变量是类变量。
A. 正确
B. 错误

☐ A

☐ B

提交

当前单选题数:0
正确单选题数:0

当前多选题数:5
正确多选题数:0

当前判断题数:1
正确判断题数:0

F. 每种题目随机出现 5 题

Java机考

Score:11Time:3TotalTime:110

单选题

15. 以下选项符合对象和类的关系的是:
A. 人和老虎
B. 书和汽车
C. 父亲和儿子
D. 汽车和交通工具

☐ A

☐ B

☐ C

☐ D

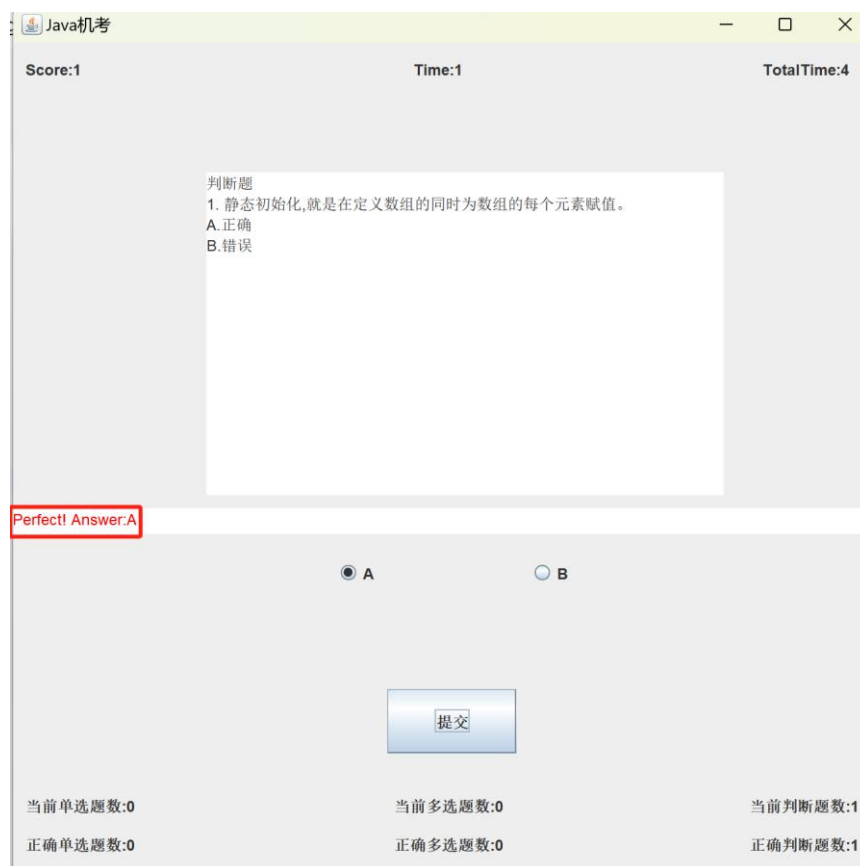
提交

当前单选题数:5
正确单选题数:3

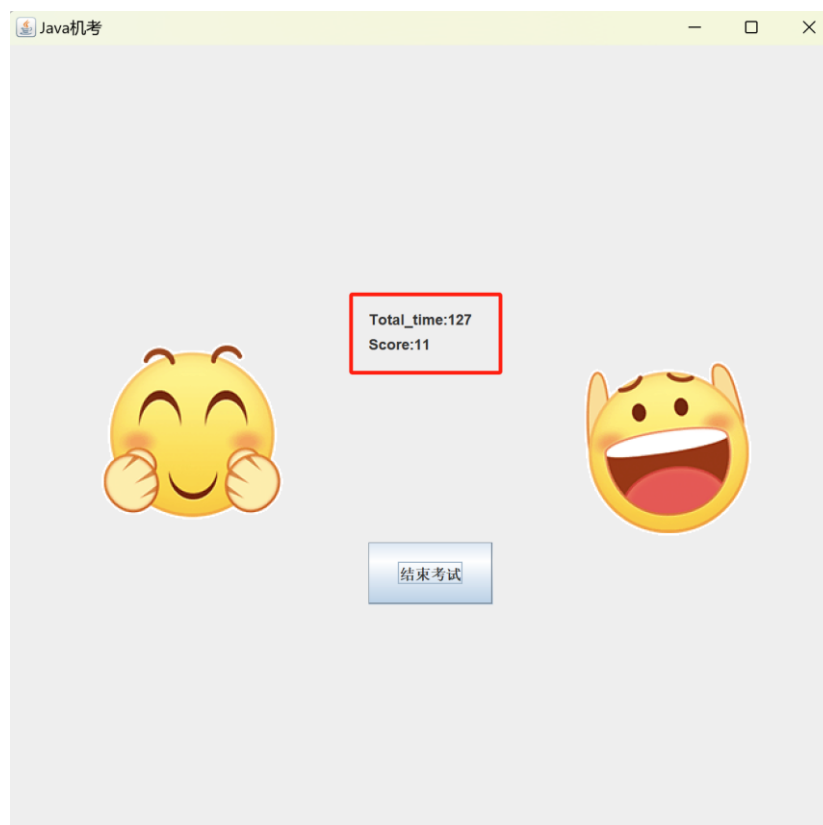
当前多选题数:5
正确多选题数:2

当前判断题数:5
正确判断题数:4

G. 提交作答后暂停 2s，显示正确答案



(3) 结束界面



3.网络通信编程：（20 分）

（3.1）利用数据报通信方式编写一个程序，该程序生成两个客户端，一个服务器端，两个客户端可以相互进行简短的文字交流。在报告中应有程序截图、完整的运行结果和简要文字说明。（15 分：数据报通信 3 分（如用套接字连接扣 2 分），两个客户端 2 分，一个服务器端 2 分，实现文字交流 2 分，程序注释和截图 2 分，运行结果截图 2 分，文字说明 2 分）

代码截图：

服务器端：

A. 编写一个客户端的类，用来存放两个客户的地址和端口，方便服务器可以把一个客户的信息传送给另一个客户。

```
// 客户端类，用于封装服务器地址和端口号
4 usages
class Client {
    4 usages
    InetAddress address; // 服务器地址
    4 usages
    int port; // 服务器端口号
    // 有参构造函数
    2 usages
    Client(InetAddress address, int port){
        this.address = address;
        this.port = port;
    }
}
```

B. 编写服务器类：

首先定义服务器的端口值为 9999，创建一个 DatagramSocket 类的对象监听 9999 端口，并输出提示语句“服务器已经启动”。之后就是一些通讯前的准备工作，需要先接收一次来自客户 1 的信息，创建一个客户类的对象记录客户 1 的地址和端口，然后接受一次来自客户 2 的信息，创建一个客户类的对象记录客户 2 的地址和端口。

```
// 服务器类
public class Servers {
    1 usage
    private static final int serverPort = 9999;
    public static void main(String[] args) {
        try {
            DatagramSocket datagramSocket = new DatagramSocket(serverPort); // 创建数据报套接字
            System.out.println("服务器已启动!!!");
            byte[] receiveData = new byte[1024]; // 创建接收数据缓存
            DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
            // 创建客户端对象 1
            datagramSocket.receive(receivePacket);
            System.out.println("客户1已准备好了!");
            Client client1 = new Client(receivePacket.getAddress(), receivePacket.getPort());
            // 创建客户端对象 2
            datagramSocket.receive(receivePacket);
            System.out.println("客户2已准备好了!");
            Client client2 = new Client(receivePacket.getAddress(), receivePacket.getPort());
        }
    }
}
```

记录好两个客户的地址和端口信息后就可以开始通信了。

新建一个 `DatagramPacket` 的对象记录客户发送过来的数据包。接收到客户发送过来的信息后，输出提示信息。

然后判断是应该把信息发送给客户 1 还是应该发送给客户 2，使用我们之前记录的客户的地址和端口信息就可以判断。判断结束并发送数据包后输出提示信息。

```
while (true){
    // 接收客户端发送的信息
    datagramSocket.receive(receivePacket);
    String clientMessage = new String(receivePacket.getData(), offset: 0, receivePacket.getLength());
    System.out.println("收到来自 " + receivePacket.getAddress() + " 的消息: " + clientMessage);
    // 发送信息给对应的客户端
    if(!client1.address.equals(receivePacket.getAddress()) || client1.port != receivePacket.getPort()){
        DatagramPacket sendPacket = new DatagramPacket(receivePacket.getData(), receivePacket.getLength(), client1.address, client1.port);
        datagramSocket.send(sendPacket);
        System.out.println("已将信息发送给客户1。");
    }
    else {
        DatagramPacket sendPacket = new DatagramPacket(receivePacket.getData(), receivePacket.getLength(), client2.address, client2.port);
        datagramSocket.send(sendPacket);
        System.out.println("已将信息发送给客户2。");
    }
}
```

客户端 1:

首先定义服务端的地址和端口信息。

然后创建一个 `DatagramSocket` 的对象，监听 999 端口，用以发送和接收数据包。

接收信息:

新建一个线程 `receiveThread`。这里使用 `Lambda` 表达式创建线程。

在线程的 `run` 函数中，新建一个 `DatagramPacket` 对象用来接收数据包，当客户 1 接收到信息时，就会输出提示信息并输出接收到的信息。

```
public class Client1 {
    1 usage
    private static final String serverHost = "localhost";
    1 usage
    private static final int serverPort = 9999;
    2 3
    public static void main(String[] args) {
        try {
            // 创建客户端UDP套接字
            DatagramSocket clientSocket = new DatagramSocket( port: 999);
            InetAddress serverAddress = InetAddress.getByName(serverHost);
            // 创建线程用于接受其他客户端的信息
            Thread receiveThread = new Thread(() -> { // 使用Lambda表达式创建线程
                try {
                    byte[] receiveData = new byte[1024]; // 创建接收数据缓冲区
                    DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
                    while (true) {
                        clientSocket.receive(receivePacket); // 接收数据
                        String receiveMessage = new String(receivePacket.getData(), offset: 0, receivePacket.getLength());
                        System.out.println("收到来自服务器的消息: " + receiveMessage);
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            });
            receiveThread.start(); // 启动接收线程
        }
    }
}
```

发送信息:

`BufferedReader` 是 Java 中用于从输入流中读取文本的类。`InputStreamReader(System.in)` 是一个输入流读取器, 它将标准输入 (通常是键盘输入) 转换成字符流。`reader.readLine()` 从控制台读取输入的一行文本并将其存储在 `message` 变量中。

`message.getBytes()` 将字符串 `message` 转换为字节数组 `sendData`。UDP 协议发送的数据需要是字节数组形式。

最后通过服务器的地址和指定的端口号即可将数据包发送给服务器。

```
// 发送消息
while (true) {
    // 从键盘获取输入并发送给服务器
    BufferedReader reader = new BufferedReader(new InputStreamReader(System.in)); // 创建键盘输入流
    String sendMessage = reader.readLine(); // 从键盘读取数据
    byte[] sendData = sendMessage.getBytes(); // 把输入的数据封装成字节数组
    DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, serverAddress, serverPort);
    clientSocket.send(sendPacket);
}
```

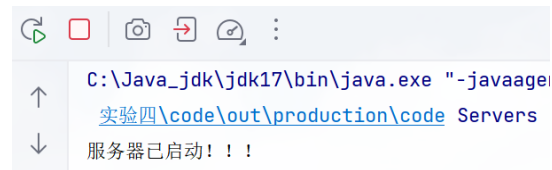
客户端 2:

除了类名和客户端端口号以外, 其它的代码和客户端 1 都是一致的。

```
public class Client2 {
    1 usage
    private static final String serverHost = "localhost";
    1 usage
    private static final int serverPort = 9999;
    2 2
    public static void main(String[] args) {
        try {
            // 创建客户端UDP套接字
            DatagramSocket clientSocket = new DatagramSocket(port: 99);
            InetAddress serverAddress = InetAddress.getByName(serverHost);
            // 创建线程用于接受其他客户端的信息
            Thread receiveThread = new Thread(() -> { // 使用Lambda表达式创建线程
                try {
                    byte[] receiveData = new byte[1024]; // 创建接收数据缓冲区
                    DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
                    while (true) {
                        clientSocket.receive(receivePacket); // 接收数据
                        String receiveMessage = new String(receivePacket.getData(), offset: 0, receivePacket.getLength());
                        System.out.println("收到来自服务器的消息: " + receiveMessage);
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            });
            receiveThread.start(); // 启动接收线程
            // 发送消息
            while (true) {
                // 从键盘获取输入并发送给服务器
                BufferedReader reader = new BufferedReader(new InputStreamReader(System.in)); // 创建键盘输入流
                String sendMessage = reader.readLine(); // 从键盘读取数据
                byte[] sendData = sendMessage.getBytes(); // 把输入的数据封装成字节数组
                DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, serverAddress, serverPort);
                clientSocket.send(sendPacket);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

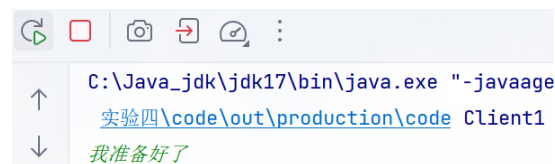
结果截图：

启动服务器和两个客户端；



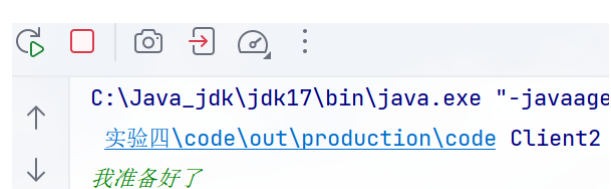
```
C:\Java_jdk\jdk17\bin\java.exe "-javaagent
实验四\code\out\production\code Servers
服务器已启动!!!
```

客户端 1：



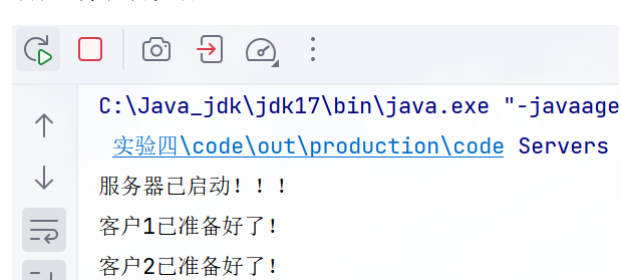
```
C:\Java_jdk\jdk17\bin\java.exe "-javaage
实验四\code\out\production\code Client1
我准备好了
```

客户端 2：



```
C:\Java_jdk\jdk17\bin\java.exe "-javaage
实验四\code\out\production\code Client2
我准备好了
```

客户端准备完成：



```
C:\Java_jdk\jdk17\bin\java.exe "-javaage
实验四\code\out\production\code Servers
服务器已启动!!!
客户1已准备好了!
客户2已准备好了!
```

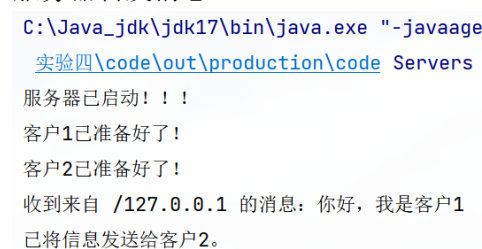
开始通信：

客户 1 发送信息：“你好，我是客户 1”



```
C:\Java_jdk\jdk17\bin\java.exe "-javaage
实验四\code\out\production\code Client1
我准备好了
你好，我是客户1
```

服务器转发消息：



```
C:\Java_jdk\jdk17\bin\java.exe "-javaage
实验四\code\out\production\code Servers
服务器已启动!!!
客户1已准备好了!
客户2已准备好了!
收到来自 /127.0.0.1 的消息：你好，我是客户1
已将信息发送给客户2。
```

客户 2: 成功接受到客户 1 发来的消息

```
C:\Java_jdk\jdk17\bin\java.exe "-javaagent:
实验四\code\out\production\code Client2
我准备好了
收到来自服务器的消息: 你好, 我是客户1
```

客户 2 回信: “你好, 我是客户 2”

```
C:\Java_jdk\jdk17\bin\java.exe "-javaagent:
实验四\code\out\production\code Client2
我准备好了
收到来自服务器的消息: 你好, 我是客户1
你好, 我是客户2
```

服务器转发消息:

```
C:\Java_jdk\jdk17\bin\java.exe "-javaagent:
实验四\code\out\production\code Servers
服务器已启动!!!
客户1已准备好了!
客户2已准备好了!
收到来自 /127.0.0.1 的消息: 你好, 我是客户1
已将信息发送给客户2。
收到来自 /127.0.0.1 的消息: 你好, 我是客户2
已将信息发送给客户1。
```

客户 1: 成功接受到客户 2 发来的回信

```
C:\Java_jdk\jdk17\bin\java.exe "-javaagent:
实验四\code\out\production\code Client1
我准备好了
你好, 我是客户1
收到来自服务器的消息: 你好, 我是客户2
```

客户端之间继续进行通信:

客户 1:

```
C:\Java_jdk\jdk17\bin\java.exe "-javaagent:D:\IJ-idea
实验四\code\out\production\code Client1
我准备好了
你好, 我是客户1
收到来自服务器的消息: 你好, 我是客户2
很高兴认识你
收到来自服务器的消息: 我也是
我们要不出来一起吃个饭吧
收到来自服务器的消息: 可以呀, 就去深圳大学吧, 那里的饭菜还不错
好主意, 明天晚上见
```


客户 2:

```
C:\Java_jdk\jdk17\bin\java.exe "-javaage
```

```
实验四\code\out\production\code Client2
```

我准备好了

收到来自服务器的消息: 你好, 我是客户1

你好, 我是客户2

收到来自服务器的消息: 很高兴认识你

我也是

收到来自服务器的消息: 我们要不出来一起吃饭吧

可以呀, 就去深圳大学吧, 那里的饭菜还不错

收到来自服务器的消息: 好主意, 明天晚上见

服务器日志:

```
C:\Java_jdk\jdk17\bin\java.exe "-javaagent:D:\IJ-idea\Intell
```

```
实验四\code\out\production\code Servers
```

服务器已启动!!!

客户1已准备好了!

客户2已准备好了!

收到来自 /127.0.0.1 的消息: 你好, 我是客户1

已将信息发送给客户2。

收到来自 /127.0.0.1 的消息: 你好, 我是客户2

已将信息发送给客户1。

收到来自 /127.0.0.1 的消息: 很高兴认识你

已将信息发送给客户2。

收到来自 /127.0.0.1 的消息: 我也是

已将信息发送给客户1。

收到来自 /127.0.0.1 的消息: 我们要不出来一起吃饭吧

已将信息发送给客户2。

收到来自 /127.0.0.1 的消息: 可以呀, 就去深圳大学吧, 那里的饭菜还不错

已将信息发送给客户1。

收到来自 /127.0.0.1 的消息: 好主意, 明天晚上见

已将信息发送给客户2。

这样一来, 两个客户之间就可以借助服务器进行简短的交流。

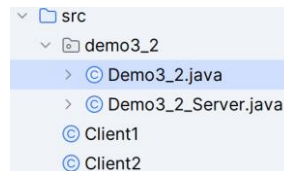
实验使用了数据报通信, 编写了两个客户端, 一个服务器端, 实现了两个客户端的文字交流, 提供了程序注释和截图, 运行结果截图, 并对代码和运行结果有了详细的文字说明文字说明。实验成功。

(3.2) 编写 Java 应用程序, 根据第 2 题“单机版 Java 简易机考程序”的要求, 将之改为网络版。客户端和服务端建立套接字连接后, 服务器端向客户端发送一个题目; 客户端将答案发送给服务器端; 服务器端判断客户端的答案是否正确。要求使用图形界面, 其他要求同“单机版 Java 简易机考程序”。在报告中应有程序截图、完整的运行结果和简要文字说明。(5 分: 套接字连接 2 分, 客户端和服务端实现双向通信 3 分)

代码截图：

此程序的代码是基于任务二的图形界面代码就行修改的，代码基本一致，只是增加了UDP套接字的连接和网络通信。

主要代码从任务二种 copy 而来，为了防止类名重复而报错，我们将代码放到包名为 Demo3_2 下面做隔离。如下图所示：



核心思路：

A. 新增一个服务端，用于发送题目、判题

服务端每次都随机一个题目，在题目内容前面加上标识字符“题目：”（用于告诉客户端这是一个“题目”，而不是“判题结果”），然后发送给客户端，客户端对其进行显示：

```
String question = questions[questionNumber]; // 获取对应题目
question = "题目：" + question;
// 发送题目给客户端
DatagramPacket sendPacket = new DatagramPacket(question.getBytes(), question.getBytes().length, serverAddress, clientPort);
datagramSocket.send(sendPacket);
```

然后，服务器还需要每次接收用户发来的答案，并对答案进行判定正确与否，最后将结果返回给客户端。

B. 在客户端中修改代码，加入 UDP 套接字以收发消息

客户端需要实时接收服务器发送来的题目、判题结果，并渲染到图形化界面中。在此过程中，由于服务器发送来的信息有两种类型，我们需要判断是哪种类型之一。于是，我们在服务器发送信息的时候，会在前面加上几个标识字符，客户端可以根据服务器发送来的信息的前几个标识字符来判断是“题目”还是“判定结果”。

```
byte[] sendData = ByteBuffer.allocate( capacity: 4).putInt(giveQuestion.getP()).array();
DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, InetAddress.getByName( host: "localhost"), port: 9999);
socket.send(sendPacket); // 发送数据

// 创建线程用于接受服务端发来的题目
Thread receiveThread = new Thread(() -> { // 使用Lambda表达式创建线程
    try {
        byte[] receiveData = new byte[1024*64]; // 创建接收数据缓冲区
        DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
        while (true) {
            socket.receive(receivePacket); // 接收数据
            String receiveMessage = new String(receivePacket.getData(), offset: 0, receivePacket.getLength());
            if(receiveMessage.startsWith("题目")){
                System.out.println("收到来自服务器发来的题目: \n" + receiveMessage.substring( beginindex: 3));
            }
            else if(receiveMessage.startsWith("判断结果")){
                System.out.println("服务器的判定结果: \n" + receiveMessage.substring( beginindex: 5));
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
});
receiveThread.start(); // 启动接收线程
```

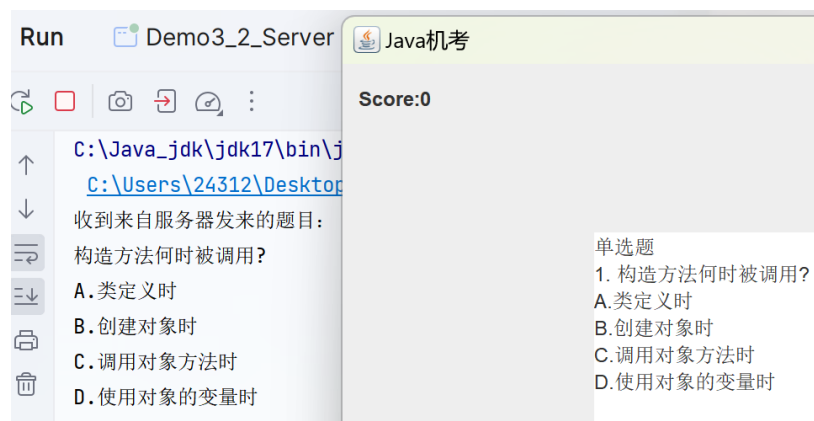

同时，当用户按下“提交”按钮时，客户端需要将“用户回答”发送给服务器进行判定：

```
//提交
submitButton=new JButton( text: "提交");
submitButton.addActionListener(new ActionListener() {
    //按下按钮后，判断是否做对题，然后继续下一题
    @Override
    public void actionPerformed(ActionEvent e) {
        giveQuestion.Check();//判断是否做对题
        giveQuestionThread.interrupt();//唤醒线程

        try {
            byte[] sendData = ans.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, InetAddress.getByName( host: "localhost"),
            socket.send(sendPacket); //发送数据
        } catch (Exception e2){
            e2.printStackTrace();
        }
    }
});
```

运行结果：

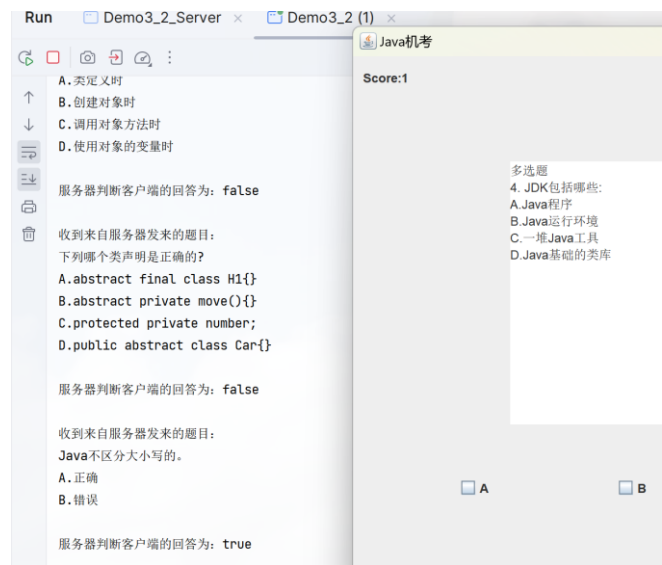
一开始，服务端像客户端发来题目：



当用户提交回答后，服务器判断正确与否，并将结果返回给客户端（这里是 false）：



继续作答，可看到客户端于服务器之间的双向通信：



四、实验总结与体会

(写写感想、建议等)

在完成本次实验 4 任务后，我深刻体会到了 Java 语言在 I/O、GUI 以及网络编程方面的强大能力。通过亲手实践，我对这些领域的理解更加深入，也对 Java 编程有了更全面的认识。

首先，在数据解析和统计编程部分，我通过编写 Java 程序处理 Amazon 的数据集，不仅锻炼了我的文件 I/O 操作能力，还让我对数据结构和算法有了更深入的理解。在实现 review.txt 文件的生成过程中，我学习到了如何使用 Java 的集合框架来存储和排序数据。同时，我也意识到了代码优化的重要性，尤其是在处理大量数据时，高效的算法和数据结构可以显著提高程序的性能。

在 GUI 编程方面，我通过开发“Java 机考”程序，加深了对 Swing 组件和事件驱动编程的理解。设计用户界面时，我学会了如何合理布局组件，以及如何通过多线程来控制程序的流程，确保用户界面的响应性。此外，我也体会到了异常处理的重要性，它不仅能够保证程序的健壮性，还能提供更好的用户体验。

网络编程部分对我来说是一个挑战，尤其是实现客户端和服务端之间的通信。我学习了如何使用 Java 的套接字编程来实现网络通信，这不仅涉及到了网络协议的知识，还让我对分布式系统有了初步的认识。在实现网络版机考程序时，我深刻理解了客户端-服务器模型的工作原理，以及如何在实际应用中处理网络延迟和数据传输的可靠性问题。

实验难点：

1. **对 Jaccard 指数的计算。**这里需要用到 Jaccard 指数的数学计算公式，涉及到集合操作和多线程编程。
2. **Java 机考程序开发。**需要我处理复杂的用户界面和事件驱动逻辑，还要实现定时器和多线程，以保证题目的定时输出和答案的实时提交。
3. **升级 Java 机考程序。**将单机版机考程序改造为网络版，需要处理客户端与服务端之间的实时通信。这里使用数据报通信方式实现客户端之间的文字交流，涉及到网络编程和多线程处理。由于本身程序的耦合度较高，修改起来很困难。

五、成绩评定及评语

1.指导老师批阅意见:

2.成绩评定:

指导教师签字: 毛斐巧
2024 年 月 日