项目设计文档：Offline Judge - 迭代一

1. 背景与目标

1.1 背景

本项目旨在开发一个类似在线评测系统的离线评测系统，允许用户通过提交答案文件来进行自动评分。

1.2 目标

在本次迭代中，将实现题目读取与评分功能。系统应能够从提供的文件夹路径中读取所有的考试文件，并为每份回答文件进行评分，最终将评分结果输出为 CSV 文件。

2. 功能需求与设计思路

2.1 题目读取

- 从给定路径中读取 exams 文件夹内的所有考试文件，这些文件可以是 XML 或 JSON 格式。
- 解析考试文件，提取考试的基本信息（考试编号、名称、开始时间、结束时间）以及题目列表。

设计思路：

- 创建一个 `Exam` 类，负责读取和解析考试文件。

-
  ```java
  package org.example.entity.exam;
  import org.example.entity.question.Question;
  import java.util.List;
  public class Exam {
      private int id;
      private String title;
      private long startTime;
      private long endTime;
      private List<Question> questions;
      // Constructor
      public Exam(int id, String title, long startTime, long endTime,
  List<Question> questions) {
          this.id = id;
          this.title = title;
          this.startTime = startTime;
          this.endTime = endTime;
          this.questions = questions;
      }
  ```

```
    // Getters and setters
}
```

- 通过文件格式判断使用相应的解析器（例如 `XmlExamBuilder` 和 `JsonExamBuilder`）来解析文件。以、`JsonExamBuilder`为例子

```
public class JsonExamBuilder extends ExamBuilder{
    @Override
    public void buildId(int id) {
        exam.setId(id);
    }
    @Override
    public void buildTitle(String title) {
        exam.setTitle(title);
    }
    @Override
    public void buildStartTime(long startTime) {
        exam.setStartTime(startTime);
    }
    @Override
    public void buildEndTime(long endTime) {
        exam.setEndTime(endTime);
    }
    @Override
    public void buildQuestions(List<Question> questions) {}

    public Exam buildExam(File examFile) throws FileNotFoundException {
        Gson gson = new Gson();
        JsonObject examJson = gson.fromJson(new FileReader(examFile),
JsonObject.class);
        //读取id title 起始 终止时间
        buildId(examJson.get("id").getAsInt());
        buildTitle(examJson.get("title").getAsString());
        buildStartTime(Long.parseLong(examJson.get("startTime").getAsString()));
        buildEndTime(Long.parseLong(examJson.get("endTime").getAsString()));
        //读取questions
        List<Question> questions;
        questions = gson.fromJson(examJson.get("questions"), new
TypeToken<List<Question>>() {}.getType());
        buildQuestions(questions);
        return getExam();
    }
}
```

- 将解析得到的考试信息存储在适当的数据结构中，`Exam` 对象列表。

2.2 评分

- 对每份回答文件进行评分，根据题目的类型和要求，计算出相应的得分。
- 对于多选题，需支持多种给分模式，包括多答不得分、错答不得分、漏答不得分等。

设计思路：

- 创建一个 `Answer` 类，负责answers数据实体

-
  ```
  public class Answer {
      private int examId;
      private int stuId;
      private long submitTime;
      private List<AnswerItem> answerItems;
      public Answer(int examId, int stuId, long submitTime, List<AnswerItem>
  answerItems) {
          this.examId = examId;
          this.stuId = stuId;
          this.submitTime = submitTime;
          this.answerItems = answerItems;
      }
      // Getters and setters
      public int getExamId() {
          return examId;
      }
      public void setExamId(int examId) {
          this.examId = examId;
      }
      public int getStuId() {
          return stuId;
      }
      public void setStuId(int stuId) {
          this.stuId = stuId;
      }
      public long getSubmitTime() {
          return submitTime;
      }
      public void setSubmitTime(long submitTime) {
          this.submitTime = submitTime;
      }
      public List<AnswerItem> getAnswerItems() {
          return answerItems;
      }
      public void setAnswerItems(List<AnswerItem> answerItems) {
          this.answerItems = answerItems;
      }
  }
  ```

  answerItem实体存放答题数据

  ```
  public class AnswerItem {
      private int questionId;
      private String answer;
      public AnswerItem(int questionId, String answer) {
          this.questionId = questionId;
          this.answer = answer;
      }
      // Getters and setters
      public int getQuestionId() {
          return questionId;
      }
      public void setQuestionId(int questionId) {
          this.questionId = questionId;
  ```

```
        }
        public String getAnswer() {
            return answer;
        }
        public void setAnswer(String answer) {
            this.answer = answer;
        }
    }
```

- 根据题目类型调用对应类中不同的评分方法，对回答文件进行评分。
- 对于多选题，根据给分模式计算相应的得分，并存储在结果中。

```
private static boolean isValidSubmissionTime(Exam exam, Answer answer) {
    // 获取考试的开始时间和结束时间
    long examStartTime = exam.getStartTime();
    long examEndTime = exam.getEndTime();
    // 获取学生提交答案的时间
    long answerSubmissionTime = answer.getSubmitTime();

    // 检查提交时间是否在考试的开始时间和结束时间之间
    return answerSubmissionTime >= examStartTime && answerSubmissionTime <=
examEndTime;
}

private static int calculateScore(Exam exam, Answer answer) {
    int Score = 0;
    for (Question question : exam.getQuestions()) {
        // 找到与当前问题匹配的答案项
        AnswerItem studentAnswerItem =
findAnswerItemForQuestion(answer.getAnswerItems(), question.getId());
        if (studentAnswerItem != null) {
            // 计算单个问题的得分
            int score = question.calculateScore(studentAnswerItem.getAnswer());
            Score += score;
        }
    }
    return Score;
}

private static AnswerItem findAnswerItemForQuestion(List<AnswerItem>
answerItems, Integer questionId) {
    for (AnswerItem item : answerItems) {
        if (item.getQuestionId() == questionId) {
            return item;
        }
    }
    return null; // 如果没有找到答案项，返回null
}
```

2.3 CSV 输出

- 将评分结果输出为 CSV 文件，包含考试编号、学生编号和得分信息。

设计思路：

```
private static void calculateScores(List<Exam> exams, List<Answer> answers) {
    String csvFilePath = "output.csv"; // CSV 文件路径
    // 准备要写入的数据
    List<String[]> data = new ArrayList<>();
    data.add(new String[]{"examId", "stuId", "score"});

    for (Answer answer : answers) {
        Exam exam = exams.stream().filter(e -> e.getId() ==
answer.getExamId()).findFirst().orElse(null);
        // 假设0分为无效提交的默认分数
        if (exam != null && isValidSubmissionTime(exam, answer)) {
            int score = calculateScore(exam, answer);
            data.add(new String[]{String.valueOf(exam.getId()),
String.valueOf(answer.getStuId()), String.valueOf(score)});
        }
    }
    // 将数据写入 CSV 文件
    try (BufferedWriter writer = new BufferedWriter(new
FileWriter(csvFilePath))) {
        for (String[] rowData : data) {
            String row = String.join(",", rowData);
            writer.write(row);
            writer.newLine();
        }
        System.out.println("CSV 文件写入成功！");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

将计算好的score添加到待输出的data中 输出csv