

# software\_design\_211250201

## 1. 概述

借助 OJ 平台，用户可以提交自己的代码，让系统测试程序能否在给定的测试用例下正常运行，从而检验程序的正确性、空间与时间消耗等信息。学生能够借助 OJ 平台练习编程，提高自己的编程能力，教师也可以借助 OJ 系统对学生的编程能力进行考核。在本次课程项目中，结合在软件系统设计课上学到的面向对象设计原则及设计模式等内容，设计并实现一个类似 OJ 系统的项目。在迭代一中，该系统能够读取 json 或 xml 格式的考试数据和学生答案，然后根据题目对应的评分策略计算得分，并在 csv 文件中输出最终成绩。

## 2. 系统结构

系统分为以下几个主要模块：

**main** ：包含程序主要入口和逻辑。

**test** ：包含测试用例和测试工具类，确保各模块的正确性。

## 3. 模块详细设计

### 3.1 main 模块

#### 3.1.1 java 目录

**org.example** ：包含所有的 Java 源代码文件

**Main.java** ：程序的入口。

**Judge.java** ：核心类，负责协调各个模块的工作流程。

**ExamJudge.java** ：核心类，负责判断计算单个学生考试答案得分。

AnswerReader.java : 负责读取学生的答案数据。

ExamReader.java : 负责读取考试相关数据。

CsvWriter.java : 负责将包含分数信息的数组输出为 csv 文件/

data : 包含数据模型类。

Answer.java : 表示单个问题的答案。

Answers.java : 表示学生提交的一整份答案。

Exam.java : 表示一次考试的信息。

Question.java : 表示考试中一个问题。

IntArrayDeserializer.java : 实现将 Question 类中 answer 属性类型统一为 int[]。

Sample.java : 表示考试中代码题样例数据类型。

strategy : 包含评分策略类。

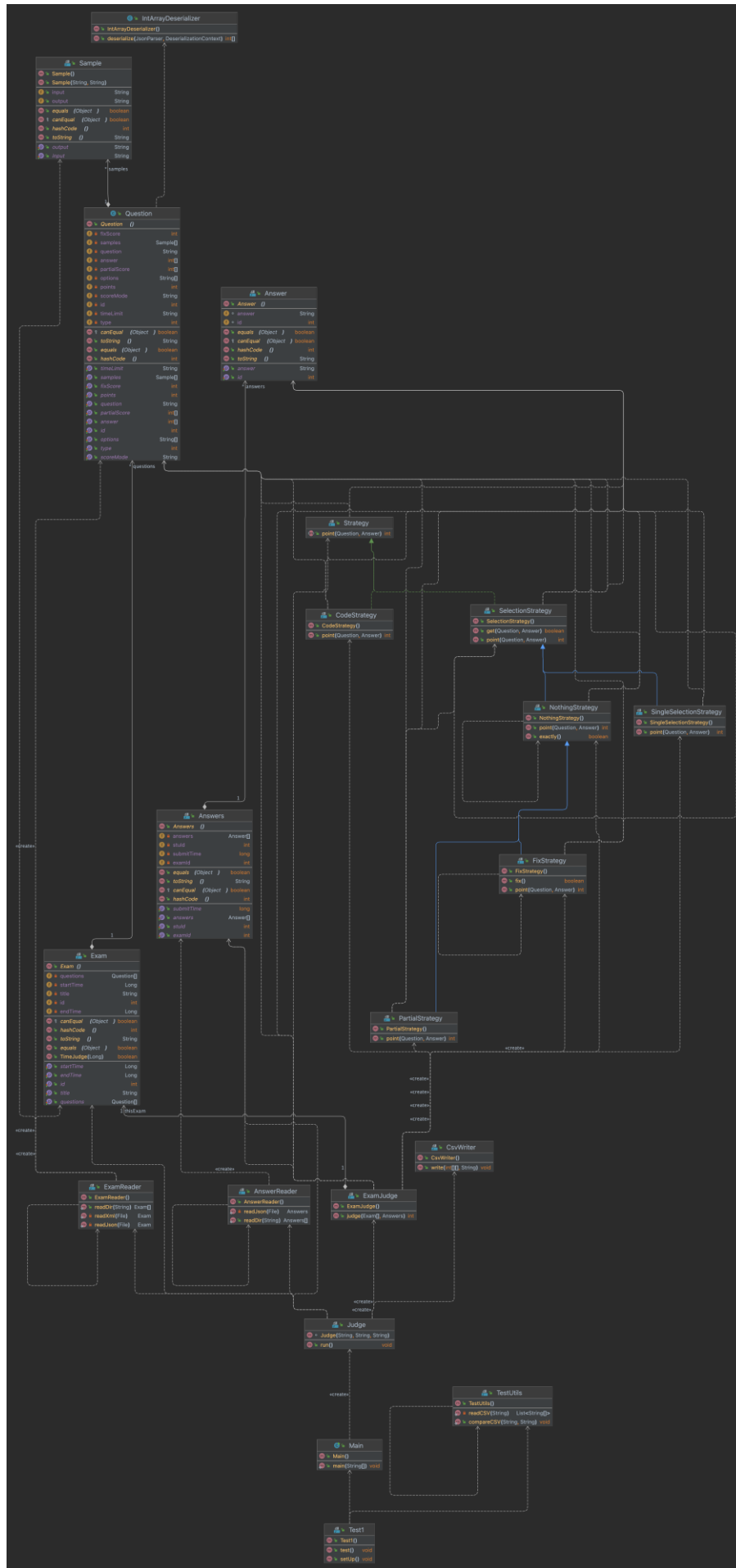
Strategy.java : 评分策略接口。

SingleSelectionStrategy.java, CodeStrategy.java: 实现具体的单选/代码评分策略。

SelectionStrategy : 定义选择题答案转换评分策略接口。

FixStrategy.java, NothingStrategy.java, PartialStrategy.java : 实现具体的多项选择题评分策略

UML 设计类图:



### 3.1.2 resource目录

包含程序运行所需的资源文件，包括 JSON 和 XML 格式的考试和答案数据和预期输出的 CSV文件。

### 3.2 test 模块

包含测试用例和测试工具类。

## 4. 工作流程

1. Main.java 启动程序，调用 ExamReader.java 读取考试题目和答案，AnswerReader.java 读取学生的答题数据。
2. ExamJudge.java 根据考试题目类型及评分方法，选择合适的评分策略。根据读取的考试信息，系统会采用相应的评分策略，如 CodeStrategy.java、SingleSelectionStrategy.java、NothingStrategy.java 等计算得分。
3. 调用 CsvWriter 将最终成绩输出到 CSV文件中。

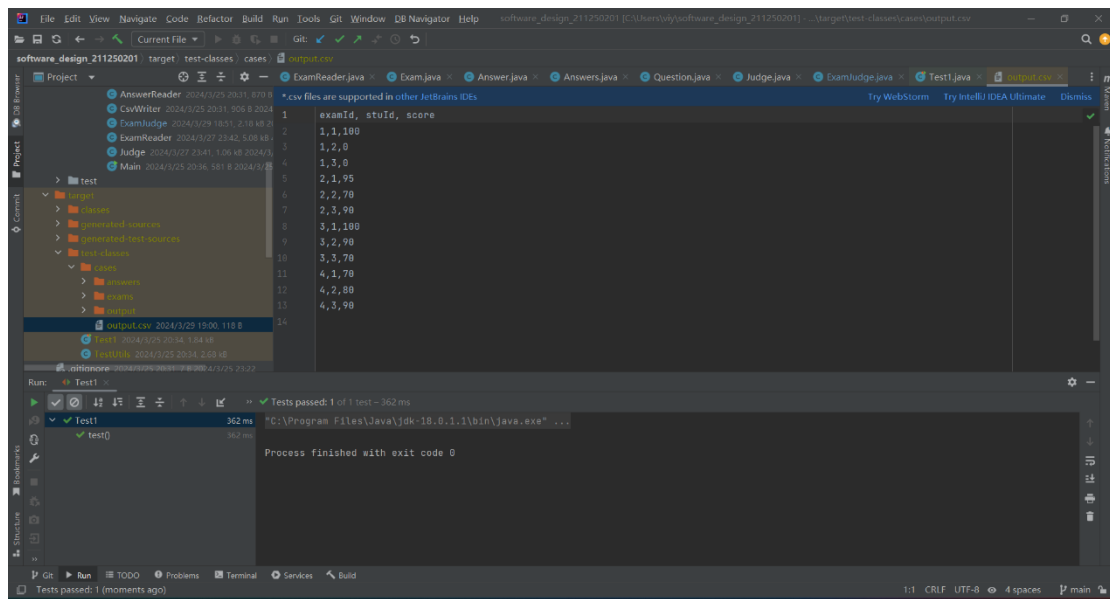
### 5. 设计原则

1. 单一职责原则：每个类或模块只负责一项任务。例如，CsvWriter.java 只处理输出 csv 文件，AnswerReader.java 只负责读取答案，Strategy.java 只复杂定义计算评分策略。类似的设计减少了类或模块之间的耦合，进而提高代码的可重用性和可维护性。
2. 开闭原则：系统的设计应该对扩展开放，对修改关闭。在这个代码结构中，Strategy 接口和其实现类（例如 CodeStrategy.java, SingleSelectionStrategy.java 等）遵循了这一原则。如果需要增加新的评分策略，可以通过实现新的策略类来完成，而无需修改现有的类。
3. 里氏替换原则：子类应该能够替换其基类而不影响程序的正确性。在这个结构中，所有的评分策略类（如 FixStrategy.java, NothingStrategy.java 等）都实现了

Strategy 接口，这意味着它们可以作为 Strategy 类型对象来使用，满足了里氏替换原则。

4. 依赖倒转原则：高层模块不应该依赖于低层模块，它们都应该依赖于抽象。在这个代码结构中，Main.java 作为高层模块，通过使用抽象 Strategy 接口，而不是具体的策略实现，以降低对具体实现的依赖。

## 6. 功能演示



The screenshot shows an IDE interface for a Java project named 'software\_design\_211250201'. The project structure on the left includes 'test' and 'target' directories. The main editor displays a CSV file named 'output.csv' with the following content:

examId	stuId	score
1	1,1	100
2	1,2	0
3	1,3	0
5	2,1	95
6	2,2	78
7	2,3	98
8	3,1	100
9	3,2	98
10	3,3	78
11	4,1	78
12	4,2	88
13	4,3	98

The bottom panel shows a successful test run for 'Test1' with the message 'Tests passed: 1 of 1 test - 362 ms' and 'Process finished with exit code 0'.