Software design设计文档

211250074 左皓升

文档更新记录：

版本号	操作者	修改内容	时间
V1.0	左皓升	迭代一	2024年3月29日

1. 引言

1.1 编制目的

本文档提供software design课程作业的软件架构概览

1.2 对象和范围

本文档的读者是作者本人，参考RUP的《软件架构文档模板》，用于指导下一循环的代码开发和测试工作。

1.3 参考资料

1. IBM RUP (2002) 软件架构文档模板
2. 《软件工程与计算（卷三）：团队与软件开发实践》骆斌、刘嘉等著

1.4 名称和术语

2. 具体设计

2.1 项目结构

2.1.1 树状图

```
├─main
|  ├─java
|  |  └─org
|  |      └─example
|  |          ├─exams
|  |          |  ├─exam
|  |          |  |  └─question
|  |          |  └─examReader
|  |          ├─judge
|  |          |  └─judgeOne
|  |          |      └─multiStrategy
|  |          ├─papers
|  |          |  ├─paper
|  |          |  |  └─answer
|  |          |  └─paperReader
|  |          └─utils
```

```
|   └resources
└test
    ├java
    └resources
        └cases
            ├answers
            ├exams
            └output
```

2.1.2 类图

2.1.3 包名解析

exams

与exams有关的类，负责考试试卷题目信息与正确答案的读取和存储

exam

负责每一份试卷题目信息与正确答案的存储

examReader

负责试卷题目信息与正确答案的读取，包括从xml文件和json文件中读取

papers

与papers有关的类，负责试卷作答的读取和存储

paper

负责每一份试卷作答的存储

paperReader

负责试卷作答的读取

judge

负责对所有作答进行评分

judgeOne

负责对一道题目的一份作答进行评分，包括单选题、多选题、代码题

multiStragegy

负责实现多选题部分正确情况的评分，包括三种策略：fix, nothing, partial

utils

存储一些需要使用的工具函数

2.2 指导原则

2.2.1 单一职责原则

一个类只有一个引起变化的原因。在读取、存储、评分中，对exam和paper的读取存储都被拆分为了读取和存储两个部分，不同文件的读取也被拆分开。在评分部分，不同类型的题目、策略也被拆分开，以便在指责发生变化时减少影响。

2.2.2 接口隔离原则

在服务类中，尽可能减少接口的数量，对服务进行拆分，设计的绝大多数服务类中最多只有一个函数接口。

2.2.3 合成复用原则

在读取exam的时候，使用ReaderFactory来选择使用xmlReader和jsonReader。

2.2.4 依赖倒转原则、开闭原则、里氏替换原则

在读取exam时，使用AbstractReader，并用jsonReader和xmlReader来扩展，方便再需求变化时进行修改。并且jsonReader和xmlReader都可以对AbstractReader进行替换。

2.2.5 迪米特法则

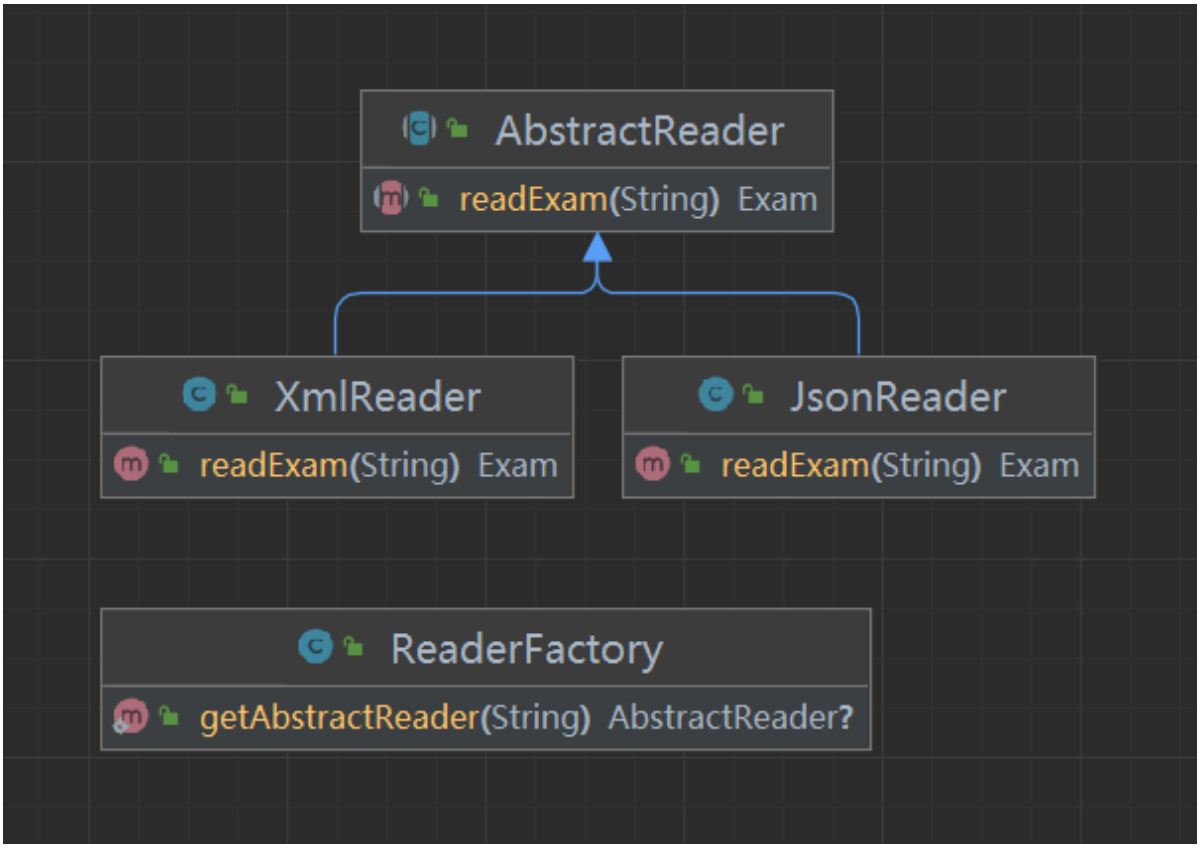
使用模块化设计，exam/paper、judge每个模块只处理自己相关的功能

2.3 设计模式

2.3.1 工厂模式

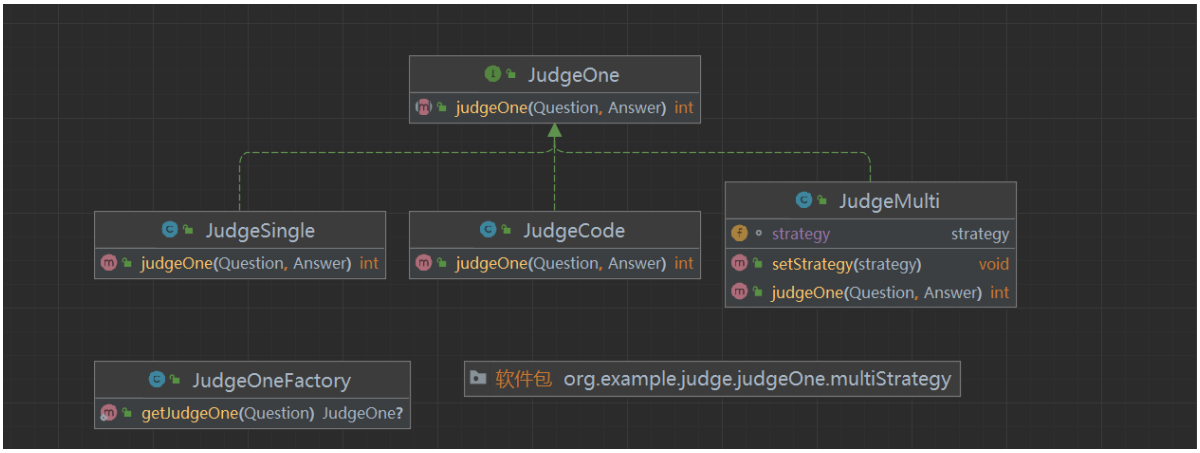
在读取exam，judgeOne时都使用了工厂模式

exam的读取中，工厂模式设计如图：



通过ReaderFactory中的静态方法选择使用哪一个reader，实现了工厂模式

在judgeOne中，工厂模式设计如图：



通过JudgeOneFactory的静态方法来选择使用的JudgeOne，实现了工厂模式。

2.3.2 策略模式

在计算多选题的部分分时，采用了策略模式。由于在设计策略模式之前，JudgeOne部分的工厂模式已经设计好，这里选择通过judgeOneFactory根据策略选择使用不同的实体类，结构如图：

3. 组合视角

3.1 物理包的划分

开发包	依赖的开发包
exams	exam, examReader
exam	question
question	
examReader	
papers	paper, paperReader
paper	answer
answer	
paperReader	
judge	judgeOne, exams, papers, exam, paper, answer, question
judgeOne	exams, papers, exam, paper, answer, question, multiStragegy
multiStragegy	utils
utils	

4. 接口视角

接口名	内容
Exams.readExams	读取所有exam

接口名	内容
AbstractReader.readExam	读取exam
ReaderFactory.getAbstractReader	选择使用的reader
Papers.readPapers	读取所有的paper
PaperReader.readPaper	读取一个paper
Judge.onlineJudge	为所有考试评分
JudgeOne.judgeOne	为一道题目评分
JudgeOneFactory.getJudgeOne	选择使用的judgeOne和多选策略
Convert.convertStringToIntArrayList	将string类型的答案转为数组

5.信息视角

5.1 Exam

```
public class Exam {
    @Getter
    @Setter
    private int id;
    @Getter
    @Setter
    private String title;
    @Getter
    @Setter
    private Long startTime;
    @Getter
    @Setter
    private Long endTime;
    @Getter
    @Setter
    private List<Question> questions;

    public Exam() {
        questions = new ArrayList<>();
    }

    public void addQuestion(Question q) {
        questions.add(q);
    }

}
```

5.2 Question

```
public class Question {
    @Getter
    @Setter
    private int id;
    @Getter
    @Setter
    private int type;
    @Getter
    @Setter
    private List<Integer> question;
    @Getter
    @Setter
    private List<Integer> partialScore;
    @Getter
    @Setter
    private String scoreMode;
    @Getter
    @Setter
    private int points;
    @Getter
    @Setter
    private int fixScore;

    public Question() {
        question = new ArrayList<>();
        partialScore = new ArrayList<>();
    }

    public void addQuestion(int q) {
        question.add(q);
    }

    public void addPartialScore(int s) {
        partialScore.add(s);
    }
}
```

5.3 Paper

```
public class Paper {

    @Getter
    @Setter
    private int examId;
    @Getter
    @Setter
    private int stuId;
    @Getter
    @Setter
    private Long submitTime;
    @Getter
```

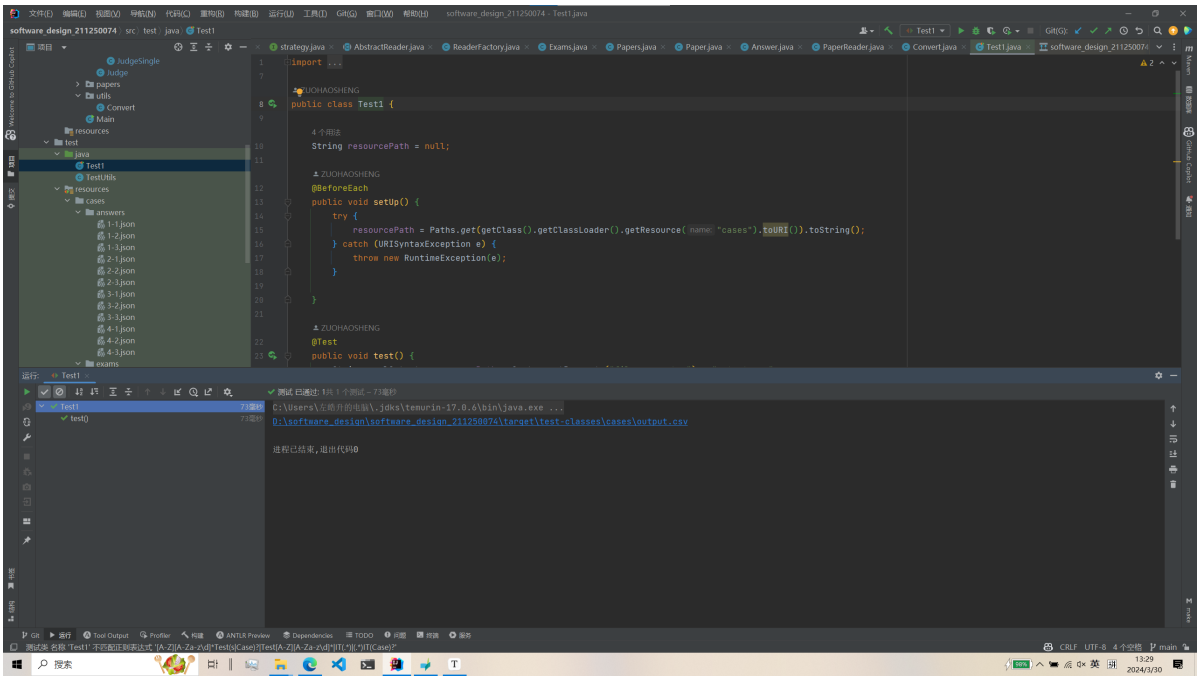
```
    @Setter
    private List<Answer> answersList;

    public Paper() {
        answersList = new ArrayList<>();
    }

    public void addAnswer(Answer a) {
        answersList.add(a);
    }
}
```

5.4 Answer

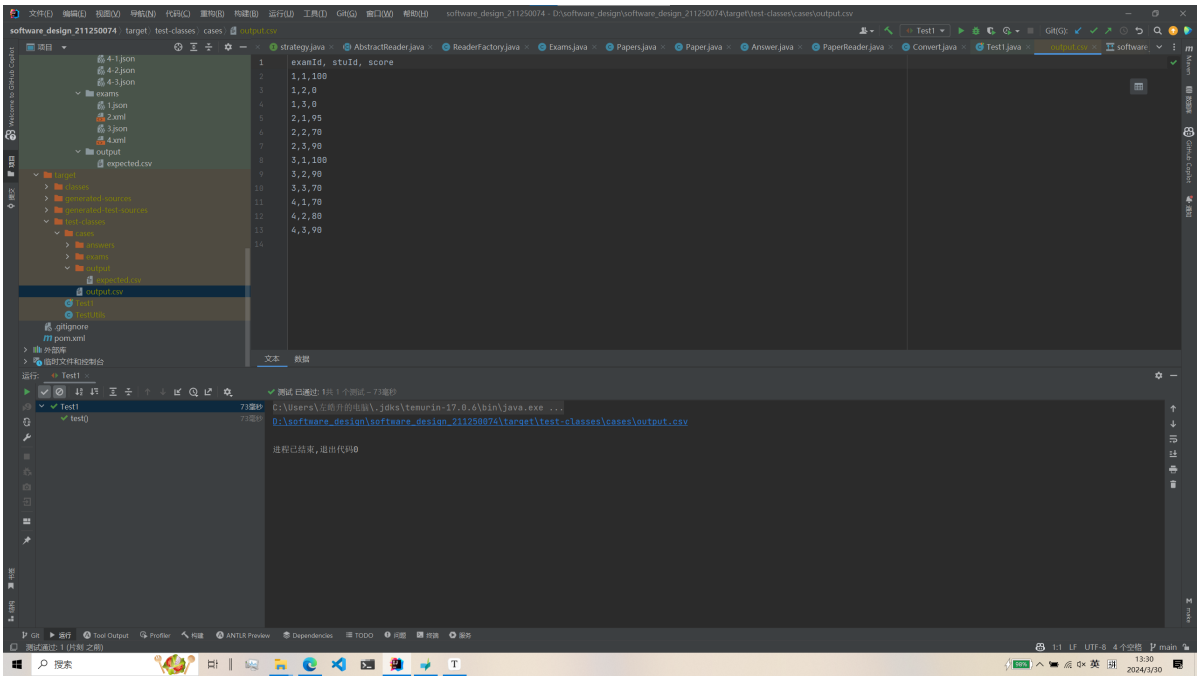
```
public class Answer {
    @Getter
    @Setter
    private int id;
    @Getter
    @Setter
    private String answer;
}
```

6. 功能演示

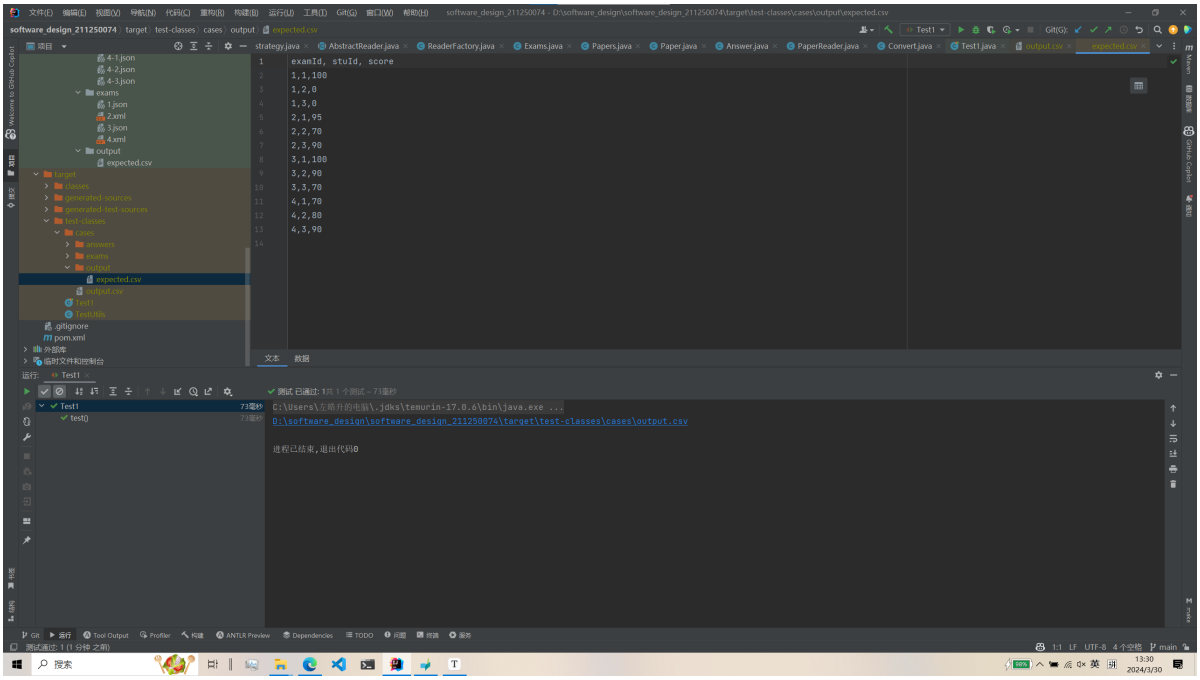
运行测试



输出的结果



正确的答案



可以看到，功能一切正常，输出了正确的结果