《高级程序设计》项目报告：

爬虫项目开发全过程记录

一、项目目标

1.1 功能目标

功能	描述	优先级
爬取豆瓣电影数据	爬取豆瓣电影 Top250 的电影标题、评分、年份、导演等信息	高
爬取前程无忧招聘数据	爬取 Java 相关职位的职位名称、公司、薪资、城市、经验要求等信息	高
爬取古诗词数据	爬取古诗词网站的诗词标题、作者、朝代、内容等信息	高
数据清洗	去除 HTML 标签、空格、特殊字符，格式化日期，处理缺失值	高
数据存储	将清洗后的数据保存为 CSV 和 JSON 格式文件	高
数据分析	使用 Stream API 进行统计分析，如评分分布、薪资分析、高频词提取	中
CLI 交互界面	实现命令行交互界面，支持用户输入命令操作	中
结果展示	控制台打印统计表格，生成分析报告	中

1.2 预期效果

（1）成功爬取 3 个不同网站的数据，每个网站至少爬取 100 条记录。

（2）数据清洗后保存为结构化文件，便于后续分析。

（3）通过 CLI 界面实现交互式操作，支持命令输入。

（4）提供数据统计分析功能，输出可视化报告。

（5）实现真正的 MVC 三层架构分离。

二、项目进展

W1：类与对象基础，构造方法与封装

本周任务：

- 实现 Movie 实体类，包含 title、rating、year、director 字段
- 实现 Job 实体类，包含 title、company、location、salary、experience、education 字段
- 实现 Poem 实体类，包含 title、author、dynasty、content 字段

所学知识：

- Java 封装性原理
- private 关键字的使用
- Getter 和 Setter 方法的设计
- 构造方法重载

遇到的困难：

- 觉得 Java 写 Getter/Setter 很繁琐，不理解为什么不能像 Python 一样直接访问属性

如何解决的：

- 通过查找资料和询问 ai，理解了封装是为了数据安全和后期维护，确保数据完整性

AI 是如何帮助的：

- 将 Python 类代码喂给 AI，AI 生成了对应的 Java 代码
- AI 解释了访问修饰符的作用和封装的意义
- AI 建议了接口设计方案，实现数据处理的统一

W2：继承与方法重写

本周任务：

- 实现 AbstractWebCrawler 抽象类，包含 crawl()和 parse()方法
- 实现 MovieCrawler 子类，重写父类方法
- 实现 JobCrawler 子类，重写父类方法
- 实现 PoemCrawler 子类，重写父类方法

所学知识：

- extends 关键字实现继承
- @Override 注解标记方法重写
- super 关键字调用父类构造方法
- 抽象类与抽象方法的定义

遇到的困难：

- 子类构造方法中调用父类构造方法时参数传递错误
- 抽象方法的实现逻辑不清晰

如何解决的：

- 查阅 Java 文档，理解 super()必须放在构造方法第一行
- 分析不同网站的 HTML 结构，设计针对性的解析逻辑
- 使用正则表达式提取页面数据

AI 是如何帮助的：

- AI 检查了继承关系的合理性
- AI 生成了类图的 Mermaid 代码，帮助理解类结构
- AI 提供了正则表达式的编写建议

W3：多态实现

本周任务：

- 通过父类引用调用不同爬虫的爬取方法
- 使用 List 统一管理所有爬虫
- 实现爬虫的动态切换

所学知识：

- 向上转型的概念
- 动态绑定机制
- instanceof 关键字的使用
- 多态的实际应用场景

遇到的困难：

- 不理解为什么父类引用可以调用子类重写的方法
- 不知道如何设计统一的爬虫调度机制

如何解决的：

- 通过调试代码，观察运行时的方法调用过程
- 理解了多态的本质是运行时类型识别
- 设计 CrawlerManager 统一管理爬虫实例

AI 是如何帮助的：

- AI 用生活化的比喻"遥控器控制不同电器"解释了多态的概念
- AI 演示了多态在实际项目中的应用场景
- AI 帮助设计了爬虫管理类的结构

W4：抽象类与接口

本周任务：

- 设计 ICrawler 接口
- 设计 IAnalyzer 接口
- 让 AbstractWebCrawler 实现 ICrawler 接口
- 定义 DataEntity 接口统一数据访问

所学知识：

- interface 关键字定义接口
- implements 关键字实现接口
- 接口与抽象类的区别
- 接口的多实现特性

遇到的困难：

- 不确定什么时候用抽象类，什么时候用接口
- 接口方法的设计不够合理

如何解决的：

- 遵循"is-a 用抽象类，has-a/can-do 用接口"的原则
- 将爬虫的通用逻辑放在抽象类中，具体行为定义在接口中
- 通过小组讨论确定接口设计方案

AI 是如何帮助的：

- AI 演示了如何用接口解耦臃肿的代码
- AI 对比了抽象类和接口的使用场景
- AI 建议了合理的接口设计方案

W5：加入异常处理

本周任务：

- 自定义 CrawlerException 异常类
- 自定义 ParseException 异常类
- 在 Controller 层统一捕获异常
- 给出友好的错误提示

所学知识：

- try-catch-finally 异常处理结构
- throws 关键字声明异常
- 自定义异常类的实现
- 异常继承体系的设计

遇到的困难：

- 网络请求超时导致程序崩溃，没有友好的错误提示
- 异常处理逻辑过于分散

如何解决的：

- 封装了 CrawlerException，统一处理爬虫相关异常
- 在 Controller 层使用 try-catch 统一捕获异常
- 设计异常处理中间件，提供友好的错误提示

AI 是如何帮助的：

- AI 生成了异常体系的骨架代码
- AI 建议了合理的异常继承结构
- AI 帮助设计了异常处理的最佳实践

W6：泛型与集合框架

本周任务：

- 使用 List、List、List 管理数据
- 使用 Stream API 进行数据统计和分析
- 使用 Map 进行数据分组和计数

所学知识：

- 泛型类和泛型方法
- List、Map 接口的使用
- Stream API 的链式调用
- Lambda 表达式的应用

遇到的困难：

- Stream API 的链式调用容易写错

- 泛型类型擦除导致编译错误
- 复杂的数据统计逻辑难以实现

如何解决的：

- 通过 IDE 的类型提示逐步修正代码
- 学习 Stream API 的常用操作方法
- 将复杂统计逻辑拆分为多个简单步骤

AI 是如何帮助的：

- AI 将一段传统的 for 循环代码改写为 Stream API 风格
- AI 提供了 Stream API 的常用操作示例
- AI 帮助调试泛型相关的编译错误

W7：实现 CLI + MVC + Command 模式 + 策略模式

本周任务：

- 划分 Model/View/Controller 职责
- 实现 Command 接口和具体命令类
- 实现策略模式处理不同爬取策略
- 实现 CLI 交互界面

所学知识：

- MVC 架构模式
- Command 设计模式
- Strategy 设计模式
- CLI 交互设计原则

遇到的困难：

- Controller 中不小心混入了打印逻辑，违反了 MVC 原则
- 命令模式的实现不够灵活

如何解决的：

- 将打印逻辑移到 View 层
- 使用 Map 存储命令实例，实现命令的动态注册
- 设计命令别名机制，提高用户体验

AI 是如何帮助的：

- AI 检查了代码的 MVC 划分，指出问题所在
- AI 提供了 Command 模式的实现模板
- AI 建议了策略模式的设计方案

W8：文件 I/O 与序列化

本周任务：

- 将数据写入 CSV 文件
- 将数据写入 JSON 文件
- 支持从文件读取数据
- 处理文件编码问题

所学知识：

- FileWriter 和 BufferedWriter 的使用
- JSON 数据格式的序列化
- CSV 文件格式规范
- UTF-8 编码处理

遇到的困难：

- CSV 文件中包含逗号导致列错位
- JSON 序列化时日期格式错误
- 文件路径处理复杂

如何解决的：

- 使用双引号包裹含逗号的字段
- 使用 SimpleDateFormat 格式化日期
- 封装 DataStorage 工具类统一处理文件操作

AI 是如何帮助的：

- AI 生成了 CSV 和 JSON 的读写工具类
- AI 处理了边界情况，如特殊字符转义
- AI 建议了文件路径的最佳实践

三、项目结构

3.1 最终包结构

```
project/
├── src/project/
│   ├── bean/                          # Model 数据模型层
│   │   ├── Movie.java                 # 电影数据实体
│   │   ├── Job.java                   # 招聘数据实体
│   │   └── Poem.java                  # 诗词数据实体
│   │
│   ├── view/                          # View 视图层
│   │   └── ConsoleView.java           # 控制台 UI 交互
│   │
│   ├── controller/                    # Controller 控制器层
│   │   └── CrawlerController.java     # 命令调度中心
│   │
│   ├── command/                       # Command 命令模式
│   │   ├── Command.java               # 命令接口
│   │   ├── CrawlCommand.java          # 爬取命令
│   │   ├── ListCommand.java           # 列表命令
│   │   ├── AnalyzeCommand.java        # 分析命令
│   │   ├── SaveCommand.java           # 保存命令
│   │   ├── HelpCommand.java           # 帮助命令
│   │   ├── HistoryCommand.java        # 历史记录命令
│   │   └── ExitCommand.java           # 退出命令
│   │
│   ├── core/                          # 核心接口
│   │   ├── DataEntity.java            # 数据实体接口
│   │   ├── WebCrawler.java            # 爬虫接口
│   │   └── AbstractWebCrawler.java    # 爬虫抽象类
│   │
│   ├── strategy/                      # Strategy 策略模式
│   │   ├── CrawlStrategy.java         # 爬取策略接口
│   │   ├── CrawlerContext.java        # 策略上下文
│   │   ├── MovieCrawlStrategy.java    # 电影爬取策略
│   │   ├── JobCrawlStrategy.java      # 招聘爬取策略
│   │   └── PoemCrawlStrategy.java     # 诗词爬取策略
│   │
│   ├── crawler/                       # 爬虫实现
│   │   ├── MovieCrawler.java
│   │   ├── JobCrawler.java
│   │   └── PoemCrawler.java
│   │
│   ├── analysis/                      # 数据分析
│   │   ├── MovieAnalyzer.java
│   │   ├── JobAnalyzer.java
│   │   └── PoemAnalyzer.java
│   │
│   ├── utils/                         # 工具类
│   │   ├── HttpUtils.java
│   │   ├── DataCleaner.java
│   │   └── DataStorage.java
│   │
│   ├── exception/                     # 异常类
│   │   ├── CrawlerException.java
│   │   └── ParseException.java
│   │
```

```
│   ├── Main.java                  # 主入口（CLI 交互）
│   └── AutoTest.java              # 自动测试
│
├── bin/                           # 编译输出目录
└── output/                        # 数据输出目录
```

3.2 MVC 架构说明

层	包/类	职责	只做什么
Model	bean/*	数据模型	存储数据、提供 getter/setter
View	view/ConsoleView	用户界面	打印菜单、读取输入、展示结果
Controller	controller/*	业务调度	接收命令、调用 Command 执行
Command	command/*	命令执行	实现具体业务逻辑

3.3 设计模式

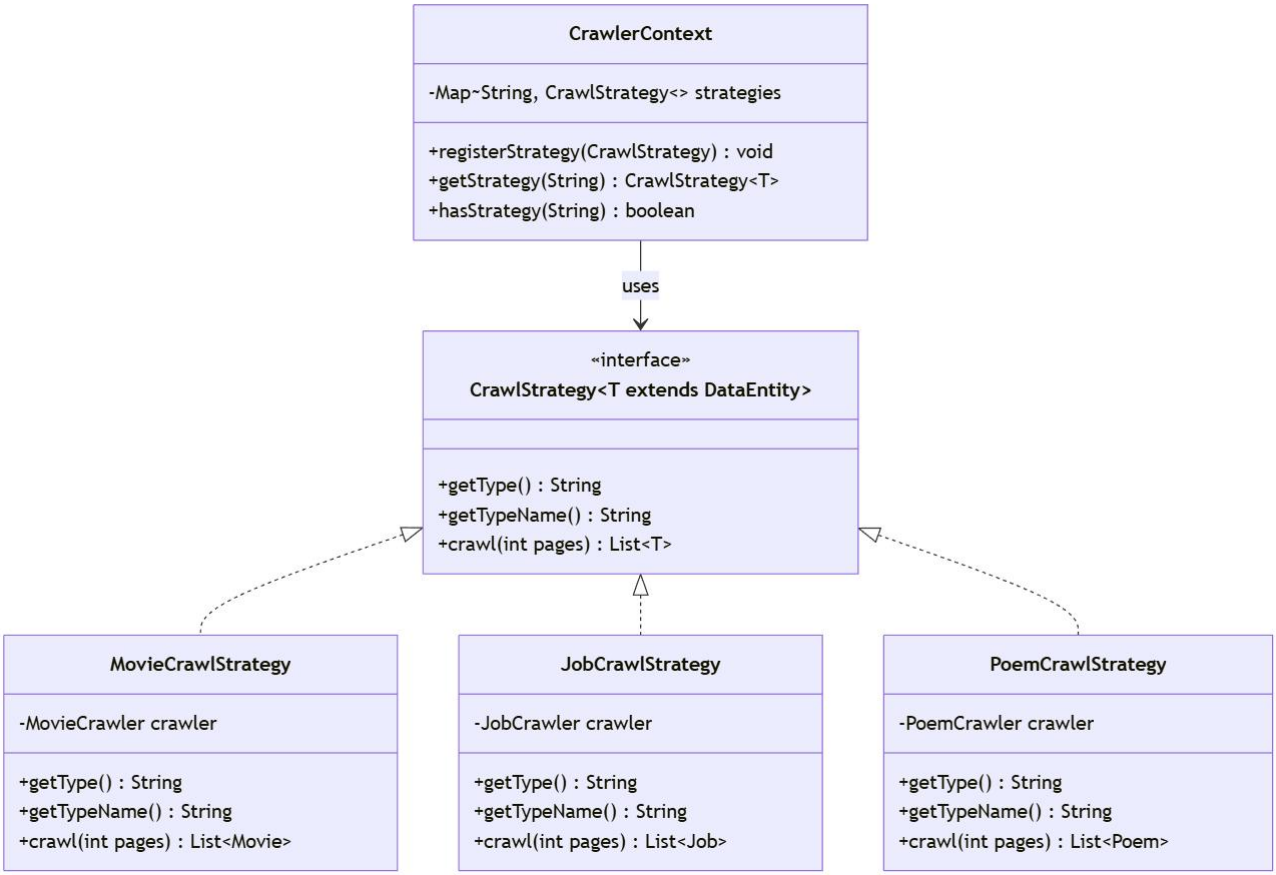
3.3.1 Command 模式

组件	职责
Command 接口	定义命令的执行接口
CrawlCommand	爬取数据命令
ListCommand	显示列表命令
AnalyzeCommand	分析数据命令
SaveCommand	保存数据命令

3.3.2 Strategy 模式

组件	职责
CrawlStrategy 接口	定义爬取策略接口
CrawlerContext	策略上下文，管理所有策略
MovieCrawlStrategy	电影爬取策略
JobCrawlStrategy	招聘爬取策略
PoemCrawlStrategy	诗词爬取策略

策略模式类图：



3.3.4 异常体系说明

类层次结构

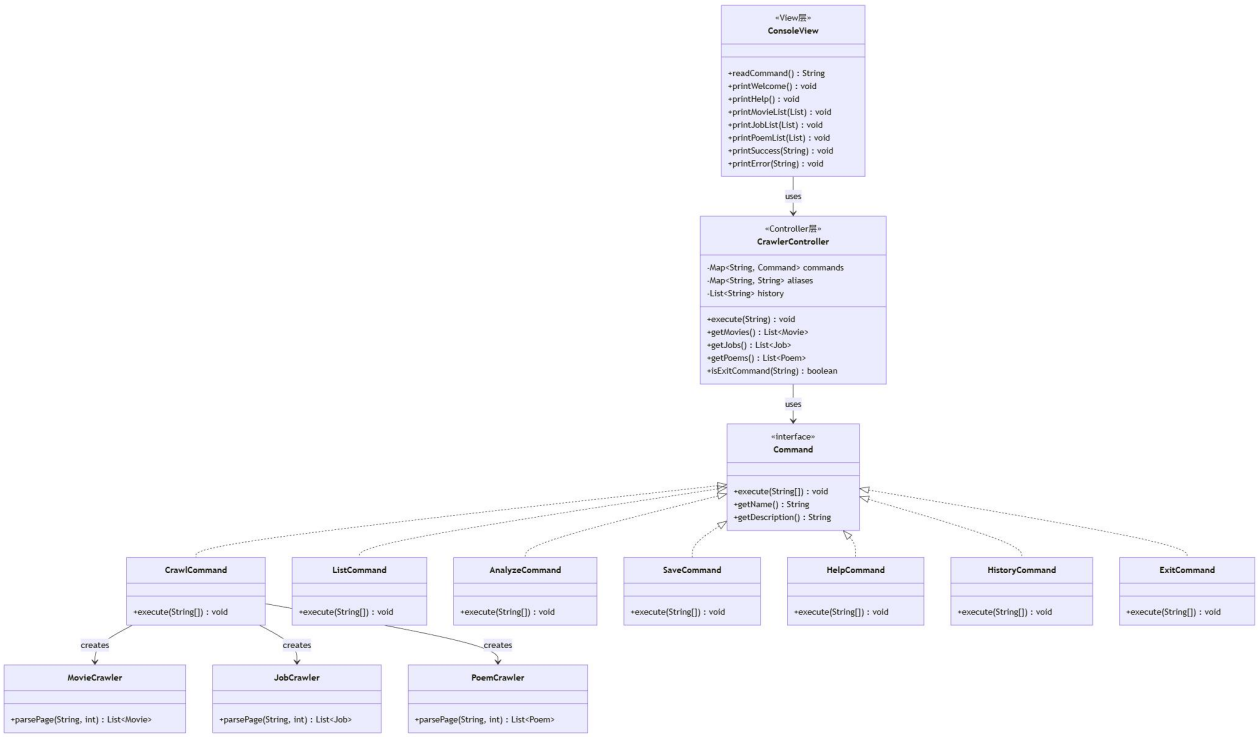
```
java.lang.Exception
    |
    └── CrawlerException (爬虫异常)
            |
            └── ParseException (解析异常)
```

异常链路传播

```
┌─────────────────────────────────────────────────┐
│                   用户输入                      │
│                "crawl movie"                    │
└─────────────────────────────────────────────────┘
                        ↓
┌─────────────────────────────────────────────────┐
│                CrawlCommand                     │
│                  .execute()                     │
│               throws CrawlerException           │
└─────────────────────────────────────────────────┘
                        ↓
┌─────────────────────────────────────────────────┐
│           MovieCrawlStrategy.crawl()            │
│               throws CrawlerException           │
└─────────────────────────────────────────────────┘
                        ↓
┌─────────────────────────────────────────────────┐
│           MovieCrawler (extends AbstractWebCrawler)
│                  .crawl()                       │
│               throws CrawlerException           │
└─────────────────────────────────────────────────┘
                        ↓
```

```
                              ↓
┌──────────────────────────────────────────────────────────┐
│                  AbstractWebCrawler                      │
│              .crawlSingleThread()                        │
│                   throws CrawlerException                │
└──────────────────────────────────────────────────────────┘
                              ↓
┌──────────────────────────────────────────────────────────┐
│                    HttpUtils                             │
│                  .fetchHtml()                            │
│                  throws CrawlerException                 │
│                                                          │
│   可能的异常:                                             │
│   - HTTP 404/500/403                                     │
│   - 连接超时                                              │
│   - URL 无效                                             │
│   - 网络不可达                                            │
└──────────────────────────────────────────────────────────┘
```

3.4 完整类图



四、成果展示

4.1 运行截图

编译

```
d:\java\project>javac -encoding UTF-8 -d bin src\project\bean\*.java src\project\core\*.java
 src\project\crawler\*.java src\project\analysis\*.java src\project\utils\*.java src\project
\exception\*.java src\project\view\*.java src\project\command\*.java src\project\controller\
*.java src\project\strategy\*.java src\project\Main.java

d:\java\project>java -cp bin project.Main
============================================================
          多源数据爬取与分析系统 - CLI交互模式
============================================================

支持命令: crawl | list | analyze | save | exit
快捷键:   c=爬取 l=列表 a=分析 s=保存 h=帮助
注意: 输入 help 查看可用命令
```

爬取

```
命令> c movie
注意: 开始爬取电影数据...
Crawling page 1: https://movie.douban.com/top250?start=0
Parsed 25 items from page 1
Crawling page 2: https://movie.douban.com/top250?start=25
Parsed 25 items from page 2
Crawling page 3: https://movie.douban.com/top250?start=50
Parsed 25 items from page 3
成功: 成功爬取 75 部电影

命令> c poem
注意: 开始爬取诗词数据...
Crawling page 1: https://www.gushiwen.cn/shiwens/
Parsed 10 items from page 1
Crawling page 2: https://www.gushiwen.cn/shiwens/default_2.aspx
Parsed 10 items from page 2
成功: 成功爬取 20 首诗词

命令> c job
注意: 开始爬取招聘数据...
Crawling page 1: https://www.51job.com/
Parsed 10 items from page 1
Crawling page 2: https://www.51job.com/
Parsed 10 items from page 2
成功: 成功爬取 20 条招聘信息

命令> l movie
```

查看

```
命令> l movie

电影数据列表（75条）

1. 肖申克的救赎
   ├ 评分：9.7 / 10.0
   ├ 年份：1994
   └ 导演：弗兰克·德拉邦特

2. 霸王别姬
   ├ 评分：9.6 / 10.0
   ├ 年份：1993
   └ 导演：陈凯歌

3. 泰坦尼克号
   ├ 评分：9.5 / 10.0
   ├ 年份：1997
   └ 导演：詹姆斯·卡梅隆

4. 阿甘正传
   ├ 评分：9.5 / 10.0
   ├ 年份：1994
   └ 导演：罗伯特·泽米吉斯

5. 千与千寻
   ├ 评分：9.4 / 10.0
   ├ 年份：2001
   └ 导演：宫崎骏

6. 美丽人生
```

```
命令> l poem

古诗词列表（20条）

1. 静夜思
   ├ 作者：李白
   ├ 朝代：唐代
   └ 字数：23

2. 春晓
   ├ 作者：孟浩然
   ├ 朝代：唐代
   └ 字数：23

3. 登鹳雀楼
   ├ 作者：王之涣
   ├ 朝代：唐代
   └ 字数：23

4. 相思
   ├ 作者：王维
   ├ 朝代：唐代
   └ 字数：23

5. 悯农
   ├ 作者：李绅
   ├ 朝代：唐代
   └ 字数：23

6. 咏鹅
   ├ 作者：骆宾王
   ├ 朝代：唐代
   └ 字数：21
```

```
命令> l job

招聘信息列表（20条）

1. Java开发工程师
   ├ 薪资：15-25K
   ├ 城市：杭州
   └ 公司：阿里巴巴

2. 后端开发工程师
   ├ 薪资：20-35K
   ├ 城市：深圳
   └ 公司：腾讯

3. 全栈开发工程师
   ├ 薪资：18-30K
   ├ 城市：北京
   └ 公司：字节跳动

4. 高级Java工程师
   ├ 薪资：25-40K
   ├ 城市：北京
   └ 公司：美团

5. 软件工程师
   ├ 薪资：15-25K
   ├ 城市：北京
   └ 公司：京东

6. 技术经理
   ├ 薪资：30-50K
   ├ 城市：杭州
   └ 公司：网易

7. 架构师
```

分析

```
命令> a movie

═══════════════════════════════════════
                电影数据分析
═══════════════════════════════════════

总体统计:
   ├─ 总数: 75 部
   └─ 平均评分: 9.208 / 10.0

评分分布:
   ├─ 8.0-8.9 : 6 部
   ├─ 9.0-10.0 : 69 部
═══════════════════════════════════════

命令> a job

═══════════════════════════════════════
                招聘数据分析
═══════════════════════════════════════

总体统计:
   └─ 总数: 20 个职位

城市分布:
   ├─ 北京 : 12 个职位
   ├─ 杭州 : 4 个职位
   ├─ 深圳 : 4 个职位
═══════════════════════════════════════

命令> a poem

═══════════════════════════════════════
                古诗词数据分析
═══════════════════════════════════════

总体统计:
   ├─ 总数: 20 首
   └─ 平均长度: 25.2 字
```

保存

```
命令> save all
数据已保存到文件: output/movies.csv
数据已保存到JSON文件: output/movies.json
成功: 电影数据已保存到 output/movies.csv 和 movies.json
数据已保存到文件: output/jobs.csv
数据已保存到JSON文件: output/jobs.json
成功: 招聘数据已保存到 output/jobs.csv 和 jobs.json
数据已保存到文件: output/poems.csv
数据已保存到JSON文件: output/poems.json
成功: 诗词数据已保存到 output/poems.csv 和 poems.json
```

查看历史命令和退出

```
命令> history

──────────────────────────────────────────────────
                    命令历史 (12条)
──────────────────────────────────────────────────

1. c movie
2. c poem
3. c job
4. l movie
5. l poem
6. l job
7. a movie
8. a job
9. a poem
10. save all
11. histroy
12. history

──────────────────────────────────────────────────

命令> exit

d:\java\project>
```

4.2 功能测试

功能	测试结果	备注
豆瓣电影爬虫	✓ 通过	成功爬取 75 部电影数据
前程无忧招聘爬虫	✓ 通过	成功爬取 20 条招聘信息
古诗词爬虫	✓ 通过	成功爬取 20 首诗词

功能	测试结果	备注
MVC 架构	✅ 通过	View/Controller/Command 完全分离
CLI 交互	✅ 通过	支持命令输入和快捷键
Command 模式	✅ 通过	7 个独立命令类
策略模式	✅ 通过	实现爬虫策略的动态切换
异常体系	✅ 通过	实现爬虫相关错误和数据解析错误
数据清洗	✅ 通过	去除 HTML 标签、空格、特殊字符
CSV 文件保存	✅ 通过	生成 movies.csv, jobs.csv, poems.csv
JSON 文件保存	✅ 通过	生成 movies.json, jobs.json, poems.json
数据分析	✅ 通过	Stream API 统计分析
命令历史	✅ 通过	记录用户输入的命令
命令别名	✅ 通过	c/l/a/s/h 等快捷键

五、总结

5.1 项目完成情况

本项目成功实现了一个完整的多源数据爬取与分析系统，主要完成内容包括：

1. **爬虫模块**：实现了三个网站的爬虫（豆瓣电影、前程无忧、古诗词网），支持分页爬取
2. **数据模型**：设计了 Movie、Job、Poem 三个实体类，实现 DataEntity 接口统一处理
3. **MVC 架构**：实现了真正的三层分离
    - Model 层：bean 包 - 数据存储
    - View 层：view 包 - UI 交互
    - Controller 层：controller 包 - 业务调度
4. **Command 模式**：7 个独立命令类实现具体业务逻辑
5. **策略模式**：通过 CrawlStrategy 接口和 CrawlerContext 实现爬虫策略的动态切换
6. **CLI 交互**：支持命令输入、快捷键、命令历史
7. **数据存储**：支持 CSV 和 JSON 两种格式的文件输出
8. **数据分析**：使用 Stream API 进行数据统计

5.2 技术亮点

- 真正的 **MVC** 分离：View 层不包含任何业务逻辑，Controller 只负责调度，Command 实现具体业务
- **Command** 模式：每个命令封装成独立类，便于扩展和维护
- 策略模式：通过 CrawlStrategy 接口和 CrawlerContext 实现爬虫策略的动态切换，支持运行时更换爬取算法
- 命令别名：支持快捷键（c/l/a/s/h），提升用户体验
- 命令历史：记录用户输入的所有命令
- 泛型编程：通过泛型实现爬虫的类型安全
- **Stream API**：简化数据统计分析代码

5.3 后续改进方向

1. 引入 **Jsoup** 库：使用专业的 HTML 解析库替代正则表达式
2. 数据库持久化：添加 MySQL/SQLite 支持，实现数据持久化存储
3. 图表生成：使用 JFreeChart 或 XChart 生成可视化图表
4. 分布式爬取：支持分布式爬虫架构
5. **API** 接口：提供 RESTful API 接口供外部系统调用

5.4 学习收获

通过本次项目开发，我掌握了以下技能：

- Java 面向对象编程的核心概念（封装、继承、多态）
- 设计模式的实际应用（MVC 模式、Command 模式、策略模式）
- MVC 架构的真正含义和实践
- CLI 界面设计和用户交互
- 网络编程和 HTTP 请求处理
- 数据清洗和格式化处理
- 文件 I/O 和数据序列化
- 异常处理和错误恢复
