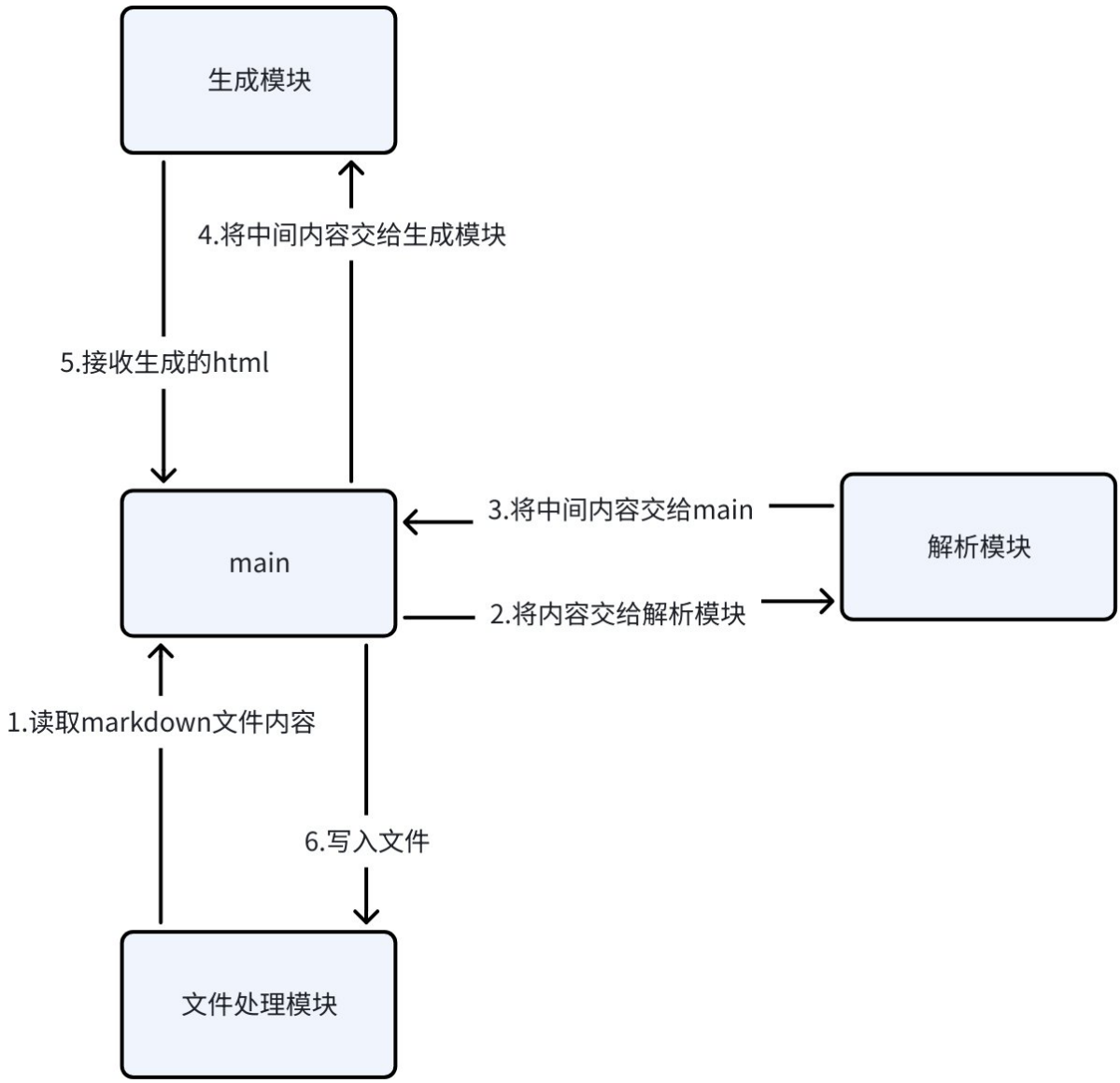
实验1：模块化的Markdown文档转换器

1. 模块划分与设计

- 文件读取和写入应该是一个独立模块吗?
  - 我觉得是的。将I/O独立出来，如果未来需要从其他的地方读取数据，就可以只替换这一个模块，而不用改动其他地方
- 解析Markdown的逻辑和生成HTML的逻辑应该混在一起吗?
  - 我觉得不应该。 解析应该是将输入的文本分解成一种结构化的中间表示，而生成是根据这个中间表示来构建最终的输出字符串。将它们分开，如果之后想要解析其他的数据或转化成新的格式就可以只修改一个模块了
- 如果将来要支持新的格式（如PDF），哪些部分需要改动?
  - 只修改解析的模块就行
- 如果将来要支持新的Markdown语法解析，哪些部分需要改动?
  - 只修改生成模块就行

模块结构图



2. 代码

见文件夹

3. 测试

test.md

D: > Desktop > 软件构造 > 2023214461+吴静+实验1 > test.md

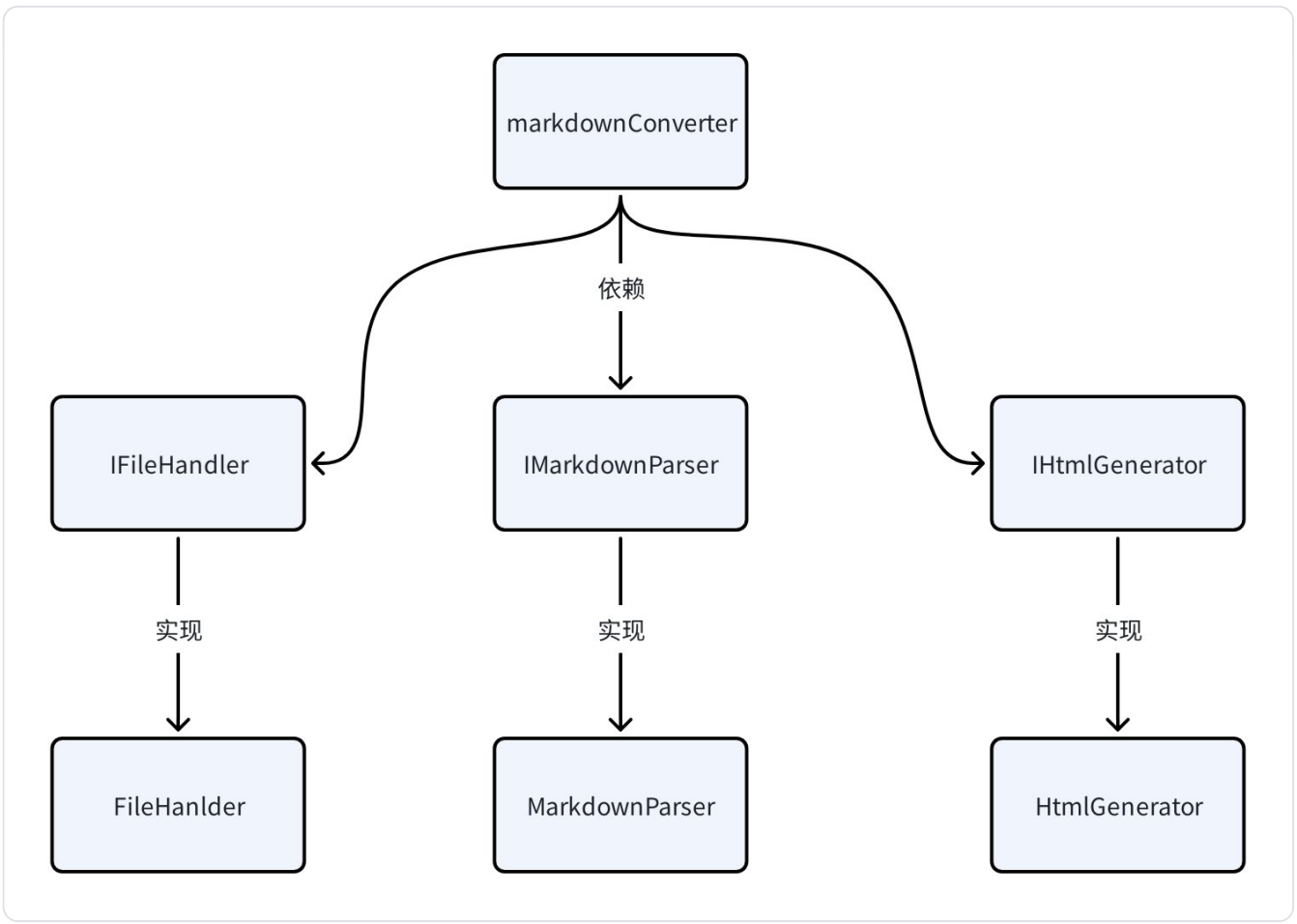
```
1  # 这是一个主标题
2  这是一段文字，里面有**粗体**和*斜体*文字。
3  ## 这是一个副标题
```

这是一个主标题

这是一段文字，里面有**粗体**和*斜体*文字。

这是一个副标题

4. 思考

- 模块依赖图and耦合程度
  - 

  - 模块之间依赖接口，属于低耦合设计
- Markdown解析模块内部

    - 在 `parseLine` 函数中，会按照不同的Markdown语法规则依次进行判断处理。先判断是否是标题，如果是则进行相应的 `<h1><h2><h3>` 等HTML标签转换；接着判断是否有加粗、斜体等标记，并进行对应 `<strong><em>` 等替换；再判断是否满足换行等格式转换的条件并处理。
    - 是高内聚的。整个 `MarkdownParser` 类的核心目的就是解析单行文本转换为对应的HTML格式。其内部的所有代码都是围绕这一单一职责展开的，且内部各功能关联性强。
- 将转换后的HTML内容直接输出到命令行
    - 修改FileHandler。在 `IFileHandler` 接口中定义了 `printToConsole` 抽象方法用于输出到控制台。
- 支持新的Markdown语法
    - 修改 `MarkdowenParser` 。
    - 不困难，只用针对性添加相应的判断逻辑代码就行。
    - 体现了“可维护性与可扩展性”的优点。