软件体系结构2024作业1

学号：2024140793

姓名：李国泽

班级：310

目标系统需求概述

系统概述

本系统为关键词索引生成系统（KWIC，Keyword in Context），旨在根据输入的文本数据生成其所有可能的循环位移版本，并对这些版本进行排序，最终输出排序结果。系统支持两种输入方式：文件输入和网络Socket输入。

核心功能包括对每一行文本进行循环位移处理，即基于指定的位移规则生成所有可能的位移版本。随后，系统按用户定义的排序策略对生成的位移结果进行排序，最终输出以关键字为中心的排序文本。

功能需求

- **输入处理**：系统支持两种方式获取输入数据：文件输入和Socket输入。通过文件输入，用户可以提供指定文件路径，系统将逐行读取文件内容，适用于处理结构化或已保存的文本数据。通过Socket输入，系统可以作为服务器监听指定端口，接收来自客户端的文本数据，并逐行读取，适合处理动态数据传输场景。
- **数据处理**：系统支持灵活的数据处理策略，核心是对输入文本进行必要的预处理和排序。用户可以选择默认排序规则或根据需求扩展自定义处理规则，以满足特定的数据操作需求。
- **循环位移生成**：系统会对每一行文本生成所有可能的循环位移版本。通过字符或单词位置的循环移动形成多种组合，例如对于“apple orange banana”，会生成“orange banana apple”和“banana apple orange”等。这些循环位移可以用于关键词索引或进一步的数据分析需求。
- **排序功能**：系统默认支持按字典顺序对循环位移数据进行排序，确保文本按指定顺序排列。同时，系统允许扩展更复杂的排序规则，例如按特定关键词或其他自定义逻辑进行排序，以满足多样化的需求。
- **输出处理**：排序后的结果会被保存到用户指定的文件中。用户需提供输出文件的路径，系统会将最终的排序结果逐行写入文件。这种方式确保处理后的数据可以持久保存，并便于后续的查看与分析。

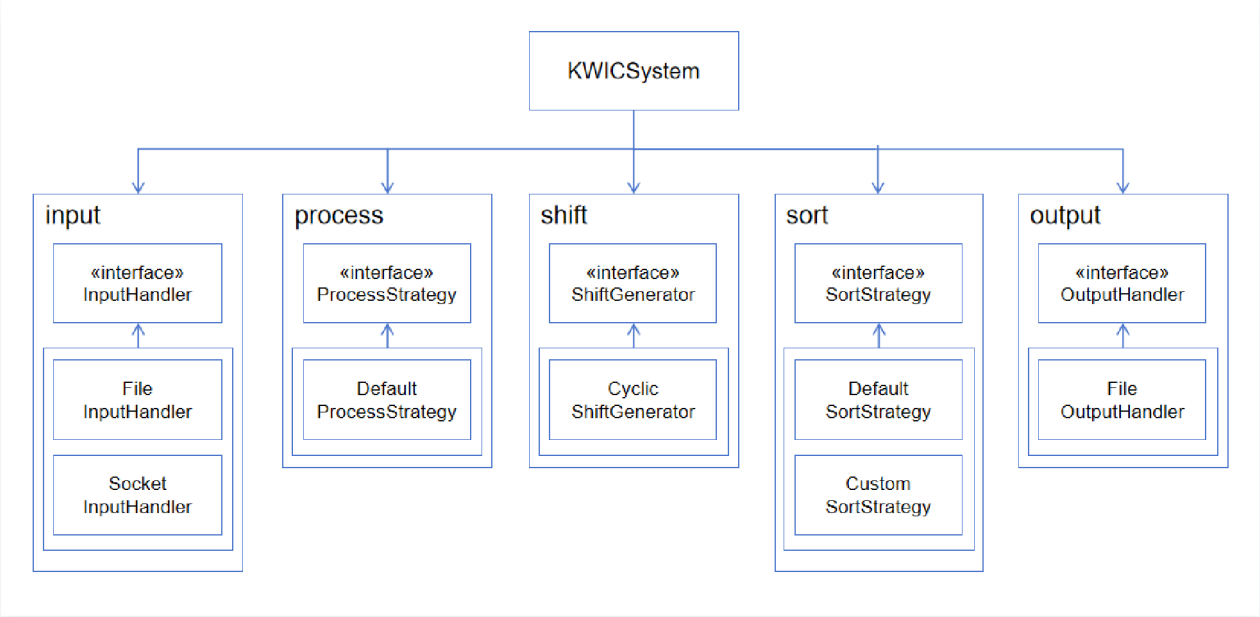
非功能需求

- **性能要求**：系统应能够高效处理大规模输入数据，包括快速完成文本读取、循环位移生成和排序操作。在处理包含数千行文本的数据时，需确保在合理时间内完成所有任务。
- **可扩展性**：系统采用模块化设计，各功能模块（如输入处理、数据处理、位移生成和排序）高度解耦，支持用户扩展功能，例如增加新的输入源、输出方式或自定义排序策略。
- **易用性**：系统提供直观的命令行接口，允许用户通过参数指定输入文件、输出文件和排序策略。命令行选项应简洁明了，附带清晰的使用提示，确保用户能轻松操作。
- **可靠性**：系统应确保数据在处理和输出过程中不丢失，且输出结果准确无误。

系统交互

- 用户通过命令行接口启动系统，指定输入源（文件或Socket）、输出路径和排序策略等参数。
- 系统从指定输入源读取数据，进行循环位移生成与排序处理。
- 最终排序结果通过指定的输出方式保存（如文件）或展示在控制台。

总体架构



如图，是本系统的系统总体架构图，本系统的整体架构由一个主模块 `KWICSystem` 和若干功能模块构成，包括 `Input` 、 `Process` 、 `Shift` 、 `Sort` 和 `Output` 。每个功能模块都通过接口定义其操作，并由具体的实现类来执行。在系统中， `KWICSystem` 作为主模块，协调各个功能模块的工作，依次调用各模块完成数据的读取、处理、位移生成、排序和输出。

系统行为概述

1. **程序启动**
   用户通过命令行传递必要参数（如输入文件路径、输出文件路径等）启动系统。程序根据用户提供的参数初始化各模块，选择适当的输入方式（文件或Socket），并准备开始数据读取。

2. **数据读取**

   - **文件输入模式**：系统根据指定路径读取文件内容，将每一行文本作为独立的数据行处理。
   - **Socket输入模式**：系统监听用户指定的端口，等待客户端连接，并从传输的数据中逐行读取文本内容。

3. **数据预处理**

   系统对读取的数据进行初步处理。当前预处理功能仅为简单的空处理，但预留了接口以支持未来扩展（如数据清理、格式转换等）。

4. **循环位移生成**

   经过预处理的文本数据将被逐行处理，生成其所有可能的循环位移版本。每一行文本会按照循环位移算法产生多个排列版本，这些版本将作为排序模块的输入。

5. **数据排序**

   循环位移生成的所有版本将根据用户指定的排序策略进行排序。系统采用模块化设计，为扩展自定义排序策略预留了接口。

6. **输出结果**

   排序完成后，系统将结果输出至用户指定的文件路径。所有排序后的数据以文本文件形式保存，供用户后续查看和使用。

7. **错误处理与日志**

   程序运行过程中，若发生异常，系统会捕获并将详细的错误信息输出至控制台，以便用户排查问题。

8. **程序结束与退出**

   所有操作完成后，系统提示"程序正常结束，结果已保存至[指定文件路径]"，并释放占用的资源（如文件句柄、网络连接等）。程序随后正常退出。

模块（类）概述

1. **KWICSystem**

   `KWICSystem` 是系统的主类，负责协调整个应用的流程。它通过命令行接受用户输入，根据参数启动对应的输入、处理、排序和输出模块。主类负责初始化各模块，包括输入模块（`InputHandler`）、数据处理策略（`ProcessStrategy`）、排序策略（`SortStrategy`）和输出模块（`OutputHandler`）。其主要方法管理系统的执行流，确保模块依序运行以完成任务。

2. **InputHandler**

   `InputHandler` 是一个接口，定义了从不同数据源读取数据的方法。核心任务是逐行读取数据，并返回一个包含所有行的列表。其实现类分别支持文件输入和Socket输入。

   - **FileInputHandler**

     实现了 `InputHandler` 接口，负责从本地文件中读取数据。通过指定文件路径逐行读取文本，并将结果返回为列表。

   - **SocketInputHandler**

     实现了 `InputHandler` 接口，负责从Socket连接中读取数据。通过监听指定端口接收来自客户端的文本数据，逐行存储至列表，直至接收结束信号。

3. **ProcessStrategy**

   `ProcessStrategy` 是一个接口，定义了对输入数据进行处理的策略。其主要任务是对每行数据进行必要的转换或清洗，以便后续处理使用。

   - **DefaultProcessStrategy**

     默认实现类，对输入数据进行空处理，保留原始格式，不做额外修改。

4. **ShiftGenerator**

   `ShiftGenerator` 接口定义了生成循环位移的逻辑。任务是基于每行文本生成所有可能的循环位移版本，供后续排序使用。

   - **CyclicShiftGenerator**

     实现了 `ShiftGenerator` 接口，通过循环位移算法生成输入文本的所有可能位移版本。生成结果以列表形式返回，用于排序模块处理。

5. **SortStrategy**

   `SortStrategy` 是一个接口，定义了数据排序的策略。其职责是对循环位移生成的文本数据按特定规则进行排序。

   - **DefaultSortStrategy**

     默认实现类，按字典顺序对数据进行排序，将文本行按升序排列。

   - **CustomSortStrategy**

     支持自定义排序规则，如按单词数量排序，或根据特定关键词的优先级对文本数据进行排序。

6. **OutputHandler**

   `OutputHandler` 是一个接口，定义了输出数据的方式。任务是将排序后的文本数据以用户指定的格式保存或展示。

   - **FileOutputHandler**

     实现了 `OutputHandler` 接口，负责将排序结果写入指定的文件。它逐行写入文本，确保数据按用户需求保存。

7. **SocketClient**

   `SocketClient` 是一个独立模块，用于测试Socket输入模式。作为客户端，它连接到指定的Socket服务器接收数据，并在控制台显示接收到的内容。 `SocketClient` 独立运行，不属于主程序的核心模块，仅用于模拟和验证Socket数据输入功能的正常性。