**KWIC索引系统**

**总体设计说明书**

目录

[1. 需求概述 3](#_Toc152981193)

[1.1. 系统简介 3](#_Toc152981194)

[1.2. 需求描述 3](#_Toc152981195)

[2. 总体架构 3](#_Toc152981196)

[2.1. 设计模式 3](#_Toc152981197)

[2.2. 类图 4](#_Toc152981198)

[3. 系统行为概述 4](#_Toc152981199)

[3.1. 工作流程 4](#_Toc152981200)

[3.2. 运行过程 4](#_Toc152981201)

[4. 各模块（类）概述 5](#_Toc152981202)

[4.1. Client类 5](#_Toc152981203)

[4.2. Observer接口 5](#_Toc152981204)

[4.3. Subject类 5](#_Toc152981205)

[4.4. KWICSubject类 5](#_Toc152981206)

[4.5. Input类 5](#_Toc152981207)

[4.6. Shift类 5](#_Toc152981208)

[4.7. Alphabetizer类 5](#_Toc152981209)

[4.8. Output类 5](#_Toc152981210)

# 需求概述

## 系统简介

KWIC（Key Word In Context）索引系统是一种文本处理工具，它能够帮助用户快速检索并查看包含关键字的文本片段。KWIC 索引系统的输入是一组文本行，每个行包含多个单词，系统会将每个单词都旋转并对原始文本进行排序，生成一个包含所有排列的“循环索引”，以便用户可以方便地查找包含特定关键字的文本片段。

## 需求描述

KWIC 索引系统需要满足以下需求：

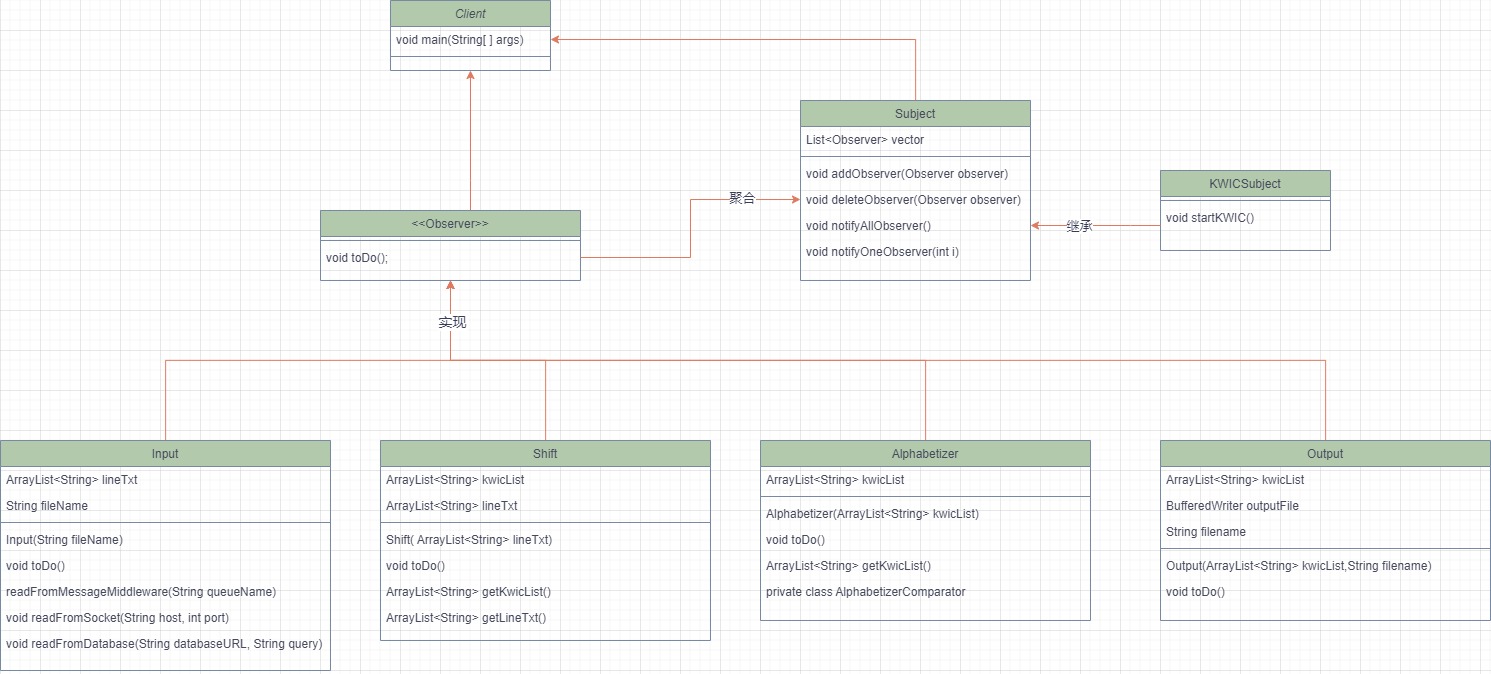
* 接收行的有序集合，输出一个所有行、经过所有循环移位后的排序列表；
* 要求系统输入还可支持Socket输入、消息中间件输入、数据库输入等；
* 系统中各个模块可独立升级；
* 系统达到较高的性能；
* 考虑如果修改排序顺序、增加处理步骤，你的程序如何进行修改；
* 具备异常处理能力，如对于空行等的处理。

# 总体架构

## 设计模式

采用观察者设计模式，使排序组件和移位组件以及IO组件之间实现解耦，排序组件不需要直接知道有哪些组件，只需要将排序结果通知给已注册的观察者即可。这种架构使得系统更加灵活、可扩展，并且符合面向对象设计中的开闭原则。

## 类图



在类图中描述了KWIC系统的详细设计结构，各个类的概述将在第4章阐述。

# 系统行为概述

## 工作流程

KWIC 索引系统的工作流程如下：

用户输入一个或多个文本行；

输入组件将文本行传递给排序组件；

移位组件对每个文本行循环移位；

排序组件对所有文本行按照字典序进行排序；

输出组件将生成的循环索引输出到文件中。

## 运行过程

1. 用户输入文本行，输入组件将其传递给排序组件。
2. 排序组件接收到新的文本行后，通过循环索引的算法对其进行排序，并更新自身状态。
3. 排序完毕后，具体排序组件会通知所有注册的查询组件，即观察者，通知内容可能包含排序结果或仅仅是通知的标记。
4. 接收到通知的观察者（查询组件）会更新自身状态以显示或处理新的排序结果，例如输出到文件、显示在界面上或进行其他的后续处理

# 各模块（类）概述

## Client类

索引系统的客户端类，包含了main入口主函数，使用命令行，以及其他输入方式均通过此类。

## Observer接口

观察者接口声明了一个ToDo方法，每个组件都通过注册自己到观察者列表中，以便在组件有新的结果时被通知。

## Subject类

被观察者类，在这个系统中充当被观察者角色。它维护一个观察者列表，在观察者注册后对观察者进行维护。

## KWICSubject类

继承自Subject类，是KWIC索引系统的具体被观察者类，是维护观察者组件的具体实例，它会在排序完成后通知所有注册的观察者。

## Input类

实现了Observer接口，是观察者对象，也是实现业务的核心类。负责不同类型的输入，包括命令行、文件位置、消息中间件输入、socket输入以及数据库输入。对不同的输入的文本最终处理，维护一个字符串的列表。

## Shift类

同样实现了Observer接口，负责实现循环移位的操作。

## Alphabetizer类

实现Observer接口，实现字典序排序的功能。

## Output类

输出类，也是观察者对象，将生成的循环移位排序后的数据生成文件保存到指定位置。