## 实验2. 软件体系结构风格实践

### 实验内容

采用两种及两种以上的体系结构风格，分别实现KWIC系统。

KWIC（Key Word in Context）的主要功能是：接受一些行，每行有诺干个单词，每个单词由若干个字符组成。其中每一行都可以循环移位，重复地把第一个单词删除，然后拼接到行末尾，然后把所有行的各种位移情况按字母表顺序输出。

### 实验目的

* 理解软件体系结构风格，并实践在应用程序开发中对软件体系结构风格的选择。

### 实验环境

JDK1.8、IDEA。

### 实验步骤

**步骤1：**分析KWIC的功能需求和性能需求，设计系统：

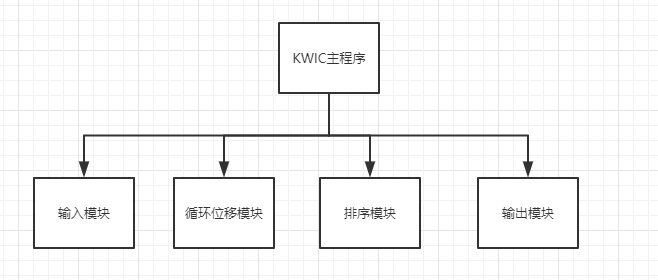
1. 设计构件和连接件；

通过实验内容对KWIC功能的描述，我们可以总结归纳出它应具有4个模块，分别是：输入模块、循环位移模块、排序模块和输出模块。

1. 设计系统的体系结构。包括系统的构件和连接件、所选用的软件体系结构风格。

带有共享数据的主程序/子程序体系结构：

构件就是各类子程序，连接件就是方法的调用。

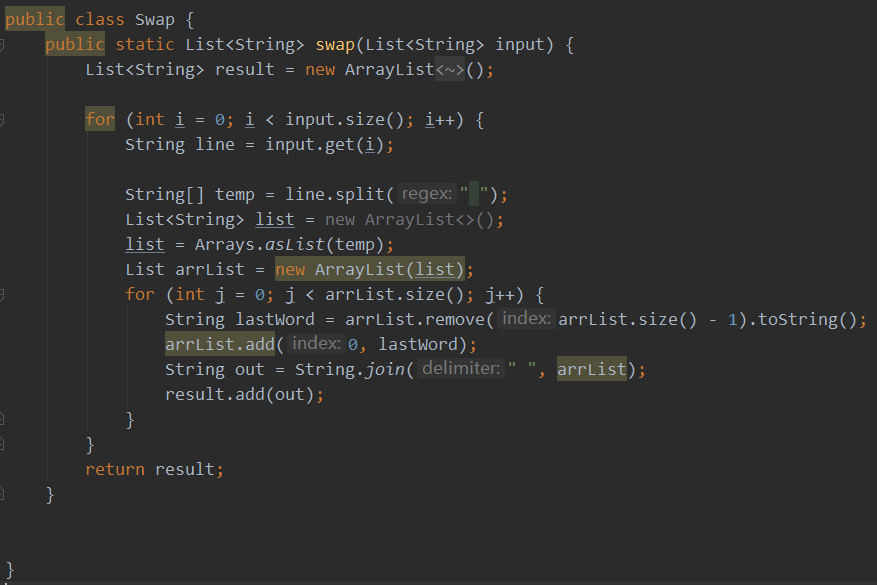


输入模块代码：



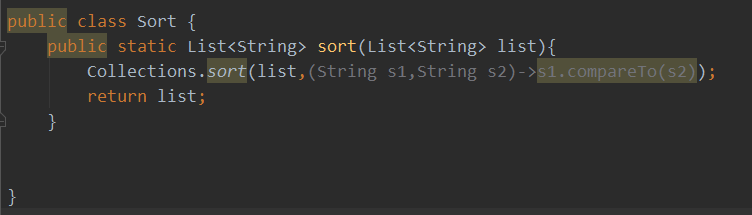
该模块接受一个url地址用来指定输入的文本，返回一个字符串数组。

单词循环位移模块：



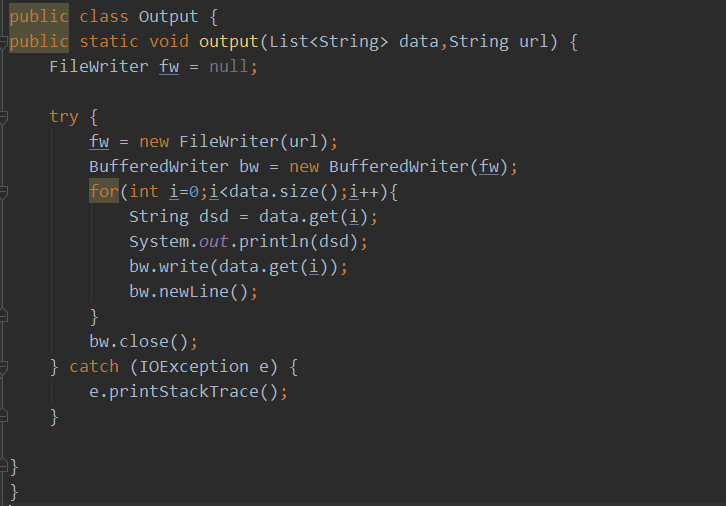
输入一个字符串数组，经过处理后同样也返回一个字符串数组，该模块负责将每个单词循环位移后形成一个新的单词排列。

字典排序模块：



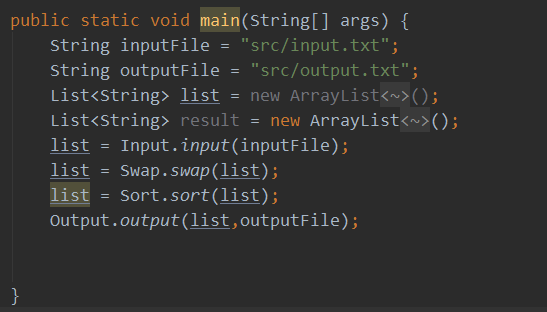
该模块同样也是接收一个字符串数组参数，经过排序后返回。

输出模块：



该模块接受一个字符串数组和输出地址，无返回值。

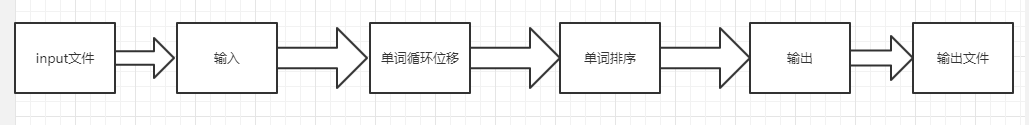
主程序main函数：

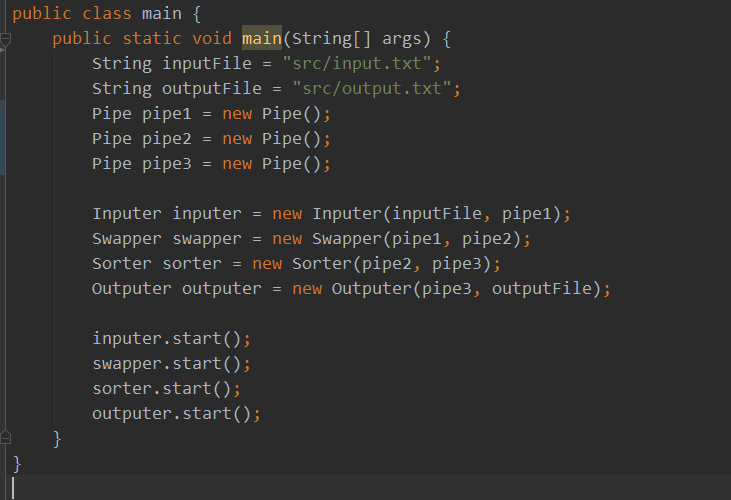


首先指定好输入位置和输出位置，然后分别调用各个模块，完成任务。

管道-过滤器风格：

管道顾名思义就是类似于管道一样来处理数据，每一个组件都有一个输入流和一个输出流，一个组件读入数据然后处理后流出到下一个组件中，就像管道中的水流一样。这个组件一般称为filter。连接器则称为pipes。filter之间不能有数据共享，并且彼此相互独立。最有名的类似的程序就是Linux的Shell了。由于各个部件之间并发执行，所以本次实验中用java的多线程来模拟管道思想，工作流程如图所示：

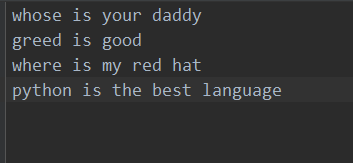




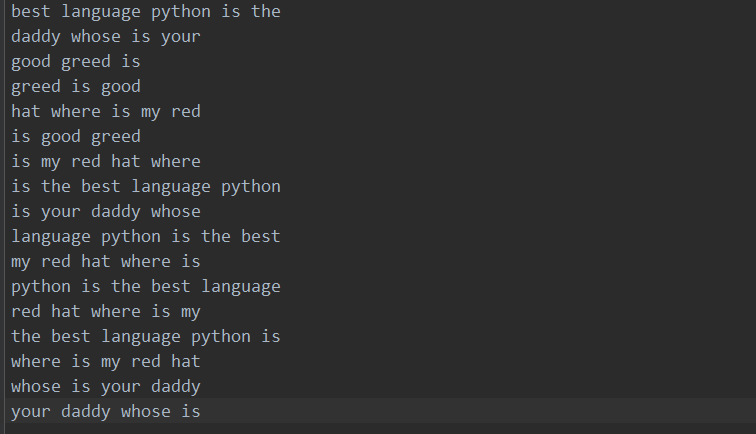
Main函数代码如图，首先声明三个管道，然后将他们通过参数形式连接起来，前一个的输出变成后一个的输入，通过inputer，swapper，sorter，outputer三个过滤器处理完成KWIC任务。

**步骤2：**编写构件和连接件，基于所选择体系结构风格组装构件，并进行测试。

输入文件内容：



输出效果：



完整代码请参考：<https://github.com/jialin43/nwafu_software_architecture>中的“02KWIC”

### 实验总结

通过对KWIC的功能拆分，归纳出共有四个模块，分别是输入模块、输出模块、单词循环位移模块和排序模块。

采用主程序/子程序的设计方式主要是模块功能的实现，将每一个子程序都看做是一个模块，各个模块编写完成之后就可调用，处理流程清晰，增强代码可读性。而且较其他方法更便于控制程序的正确性。

而采用管道和过滤器的方式中，每个部件完成了一个对数据流独立处理的功能，它接受数据流的输入，经过处理后输出给下一个过滤器进行处理。其中连接件主要负责对接适配过滤器的输入和输出，一般不做数据业务处理。各个过滤器可以并发执行，就形成了类似管道一样的流水线。

### 其他参考

功能要求请参考“KWIC.ppt”，而软件体系结构风格/模式的描述请参考“软件体系结构风格（模式）描述示例 - 分层模式.jpg”。