Key Word In Context – Strategy

Ask the user if they would like to input from file or from console.
Ask the user if they would like to output to file or to console.
Don't worry about exception handling or incorrect/invalid user input.

Design requirement:

1. Input should be an interface with the read() method defined. The Input from A1 should be renamed to InputFromFile. The Input from A2 should be renamed to InputFromConsole. Since the method signatures must match, the method may no longer take in the singleton wrappers. Instead, these must now be passed in through a constructor in the InputFromConsole class. Note that InputFromFile does not need a constructor.
2. Output should be an interface with the write() method defined. The Output from A1 should be renamed to OutputToFile. The Output from A2 should be renamed to OutputToConsole. Since the method signatures must match, the method may no longer take in the singleton wrappers. Instead, these must now be passed in through a constructor in the OutputToConsole class. Note that OutputToFile does not need a constructor.
3. MasterControl now uses the singleton wrapper directly to ask the user for their choices. Based on those choices, it creates the appropriate Input and Output object.
4. There are now 4 permutations of how KWIC will run, and you are responsible for manually testing them all for correctness:
    a. Input from file, output to file
    b. Input from file, output to console
    c. Input from console, output to file
    d. Input from console, output to console

Guidelines:

InputFromConsole:

Hint if struggling with the singletons coming in through constructor:

```
public class InputFromConsole implements Input {

    ScannerWrapper scannerWrapper;
    SystemWrapper systemWrapper;

    public InputFromConsole(ScannerWrapper scannerWrapper, SystemWrapper systemWrapper) {
        this.scannerWrapper = scannerWrapper;
        this.systemWrapper = systemWrapper;
    }
```

OutputToConsole:

Hint if struggling with the singleton coming in through constructor:

```
public class OutputToConsole implements Output {

    SystemWrapper systemWrapper;

    public OutputToConsole(SystemWrapper systemWrapper) {
        this.systemWrapper = systemWrapper;
    }
```

MasterControl:

Ask the user for their choices and sets up objects accordingly.

Hint: Look at the last page of the OOP Lab as a refresher on how to handle objects polymorphically!

The program should prompt and work exactly as in the screenshots below:

a)

```
Please enter FILE to input from file or CONSOLE to input from console:
FILE
Please enter FILE to output from file or CONSOLE to output from console:
FILE
```

b)

```
Please enter FILE to input from file or CONSOLE to input from console:
FILE
Please enter FILE to output from file or CONSOLE to output from console:
CONSOLE
and The Sea The Old Man
Ascent of Man The
Descent of Man
Man and The Sea The Old
Man Descent of
Man The Ascent of
of Man Descent
of Man The Ascent
Old Man and The Sea The
Sea The Old Man and The
The Ascent of Man
The Old Man and The Sea
The Sea The Old Man and
```

c)

```
Please enter FILE to input from file or CONSOLE to input from console:
CONSOLE
Please enter FILE to output from file or CONSOLE to output from console:
FILE
Please enter lines to add, then enter -1 to finish:
Sense and Sensibility
Architecture Software
Crouching Tiger Hidden Dragon
-1
```

d)

```
Please enter FILE to input from file or CONSOLE to input from console:
CONSOLE
Please enter FILE to output from file or CONSOLE to output from console:
CONSOLE
Please enter lines to add, then enter -1 to finish:
Sense and Sensibility
Architecture Software
Crouching Tiger Hidden Dragon
-1
and Sensibility Sense
Architecture Software
Crouching Tiger Hidden Dragon
Dragon Crouching Tiger Hidden
Hidden Dragon Crouching Tiger
Sense and Sensibility
Sensibility Sense and
Software Architecture
Tiger Hidden Dragon Crouching
```

Submission

Same as A1.