

Key Word In Context – Singleton Wrappers

Change Input to read input in from the console instead of from a file.

Change Output to output the result to the console instead of to a file.

Don't worry about exception handling or incorrect/invalid user input.

Design requirement:

1. Keep the Java layer abstracted from your code.
2. ScannerWrapper should be a singleton that sets up a Scanner and has a nextLine() method to leverage Scanner's nextLine() method.
3. SystemWrapper should be a singleton that has a println() method to leverage System.out.println()
4. Dependency inversion dictates that the singletons should be passed in to the classes that need them. Input needs both. Output needs only SystemWrapper. For this assignment, pass them in through the method, not setters or constructor (we will change this in the next assignment). Since MasterControl needs both Input and Output, MasterControl also needs both singletons. This means that something needs to pass them in to MasterControl (the main method should).

Guidelines:

Input:

Method signature change:

```
public List<String> read(ScannerWrapper scannerWrapper, SystemWrapper  
                        systemWrapper)
```

Remove any **throws** from the method signature that may be left over from A1.

CircularShifter:

Unchanged

Alphabetizer:

Unchanged

Output:

Method signature change:

```
public void write(List<String> lines, SystemWrapper systemWrapper)
```

Remove any **throws** from the method signature that may be left over from A1.

MasterControl:

Method signature change:

```
public void start(ScannerWrapper scannerWrapper, SystemWrapper systemWrapper)
```

Main method, given to you for free:

```
public static void main(String[] args) throws IOException {  
    MasterControl masterControl = new MasterControl();  
    masterControl.start(ScannerWrapper.getInstance(), SystemWrapper.getInstance());  
}
```

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

```
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
```

For the included tests:

Make sure the test source folder is already created (like in A1). Then:
Right click on your Project in Project View -> Add Framework Support
Find and check Maven (you can type "maven" to find it faster). Click OK.
The pom file will open automatically. Please change:
Group ID: firstname.lastname all lowercase, no spaces
Artifact ID: assignment name all lowercase, no spaces

Before `</project>` paste in:

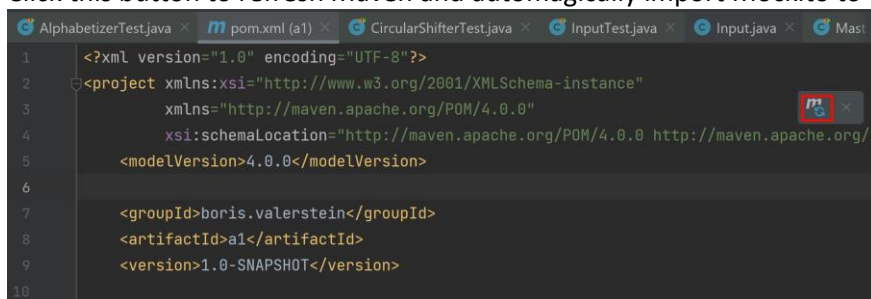
```
<dependencies>
  <dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-core</artifactId>
    <version>3.3.3</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

Should look like this:



```
11 <properties>
12   <maven.compiler.source>11</maven.compiler.source>
13   <maven.compiler.target>11</maven.compiler.target>
14 </properties>
15
16 <dependencies>
17   <dependency>
18     <groupId>org.mockito</groupId>
19     <artifactId>mockito-core</artifactId>
20     <version>3.3.3</version>
21     <scope>test</scope>
22   </dependency>
23 </dependencies>
24
25 </project>
```

Click this button to refresh Maven and automatically import Mockito to your project:



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xmlns="http://maven.apache.org/POM/4.0.0"
4   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/x
5   <modelVersion>4.0.0</modelVersion>
6
7   <groupId>boris.valerstein</groupId>
8   <artifactId>a1</artifactId>
9   <version>1.0-SNAPSHOT</version>
```

If this button does not appear, you can right click anywhere in the pom file -> Maven -> Reload Project

Remember that tests are an indicator that your code works as expected, but always run the code from the main method as well to make sure the program runs correctly for real.

If the Wrappers are incorrect, the JUnit tests *may pass*, but you won't get full credit for the assignment. Your code will now be in src/main/java. You can move your tests to src/test/java.

Submission

Same as A1.