Key Word In Context

WARNING: This assignment is a test of following directions.

This setup is required:

Install IntelliJ Ultimate Edition. It comes with 30 day free trial.

License is free either as part of Github's Student pack or Jetbrains student license:

https://education.github.com/pack

https://www.jetbrains.com/community/education/#students

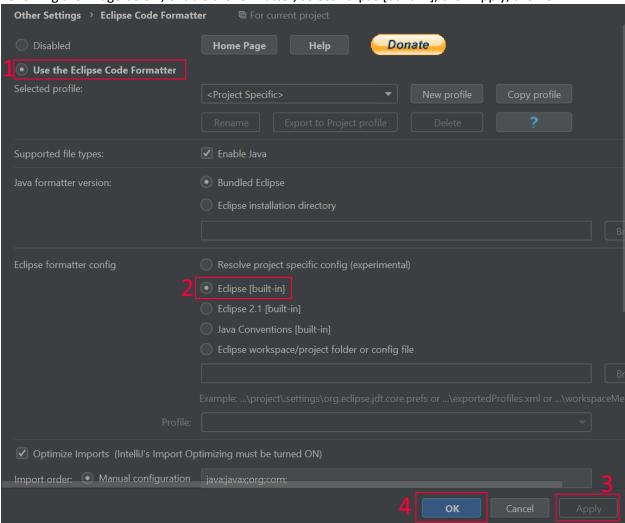
Install the most recent Java JDK/JRE.

Create a new Java Project. File -> New -> Java (left hand nav) -> Next -> Next -> Name it -> Finish Install these 2 plugins:

https://plugins.jetbrains.com/plugin/7642-save-actions/

https://plugins.jetbrains.com/plugin/6546-eclipse-code-formatter/

Restart IntelliJ. Go to File -> Settings. Under "Other Settings" click on "Eclipse Code Formatter" Following the image below, enable the formatter, select Eclipse [built-in], then Apply, then OK.



Under "Other Settings" click on "Save Actions":

Other Settings 〉 Save Actions 恒 For current project		
General		
 Activate save actions on save (before saving each file, performs the configured actions below) Activate save actions on shortcut (default "CTRL + SHIFT + S") 		
Activate save actions on batch ("Code > Save Actions > Execute on multiple files")		
No action if compile errors (applied per file)		
Formatting Actions		
✓ Optimize imports✓ Reformat file		
Reformat only changed code (only if VCS configured)		
✓ Rearrange fields and methods (configured in "File > Settings > Editor > Code Style > () > Arrangement")		
lava Inspection and Quick Fix		

Java Inspection and Quick Fix		
	dd final modifier to field	
	dd final modifier to local variable or parameter	
	dd final modifier to local variable or parameter except if it is implicit	
	dd static modifier to methods	
	dd this to field access	
	dd this to method access	
	dd class qualifier to static member access	
	dd class qualifier to static member access outside declaring class	
✓	dd missing @Override annotations	
✓	dd blocks to if/while/for statements	
	dd a serialVersionUID field for Serializable classes	
	emove unnecessary this to field and method	
	emove final from private method	
	emove unnecessary final to local variable or parameter	
	emove explicit generic type for diamond	
	emove unused suppress warning annotation	
	emove unnecessary semicolon	
	emove blocks from if/while/for statements	
	nange visibility of field or method to lower access	
	OK Cancel Apply	

Select everything as it is above, then click Apply, then OK.

You will have to do all of this for every IntelliJ Project you create. ^

Guidelines:

Follow the tests! (See instructions below)

MasterControl:

Responsible for combining all the other components

Contains a main method, but the main method should just create a new instance of MasterControl and call masterControl.start(); 2 lines max in the main method.

Method: public void start() + public static void main(String[] args)

Input:

Read input file named "kwic.txt" in the default location.

Assume the file will have no punctuation or anything else that requires extra considerations. public List<String> read()

CircularShifter:

Takes a list of Strings and shifts them all, returning the entire list of lines.

Method: public List<String> shiftLines(List<String> lines) {

Alphabetizer:

Takes a list of Strings and alphabetizes them all.

You must use Java's Treeset to do the alphabetizing. Ignore casing when alphabetizing.

Educator notes:

This won't be ideal. Add everything in the list to a TreeSet, then convert the Set back to a List to return. The reason I am requiring using Treeset is for you to have 1 less thing to learn for the next assignment, where TreeSet will actually make more sense to use.

Method: public List<String> sort(List<String> lines)

Output:

Takes a list of Strings and outputs them to a file named "kwic_output.txt" in the default location Method: public void write(List<String> lines)

Included you will find a set of JUnit test cases. You can bring these into your IntelliJ project and run them as a guideline to see if you're on the right track:

Click on "Project" on the left hand nav to open the Project view.

Right click on your project -> New -> Directory -> "test".

Right click on the "test" folder -> Mark Directory As -> Test Sources Root

Copy the files into this new source folder.

Open MasterControlTest. Hover over the @BeforeEach, click to add JUnit 5 to the classpath. DO NOT ADD JUNIT 4. You must use JUnit 5.

```
class MasterControlTest {

MasterControl masterControl;

MasterControl masterControl;

BeforeEach

Void Cannot resolve symbol 'BeforeEach'

Add 'JUnit5.4' to classpath Alt+Shift+Enter More actions... Alt+Enter

void full_kwic() throws IOException {
```

You can run the tests by right clicking on a file -> Run "MasterControlTest" or CTRL+SHIFT+F10 You can run all of the tests at once by right clicking on the project -> Run "All Tests" Throughout the course, I will be using test cases to grade the "functional" part of your grade for assignments. Running them yourself will avoid surprises when you submit your work.

Caveat: These aren't the most beautiful test cases, but such is the sacrifice made for efficient grading.

Use the default package for all files. Please, please, please, no packages for your Java files. Yes, I know, it is bad practice. But again, sacrifices must be made for efficient grading.

Before you submit

You think you have completed the assignment? Not so fast!

Follow these steps:

- 1. Delete the tests from your test source folder
- 2. Download the tests fresh from BB Learn
- 3. Copy the fresh tests into your test source folder
- 4. If anything is red, **stop**. Do **not ever** change the tests. Fix your **code** to make the tests compile. Proceed to step 5 when there are no compilation errors.
- 5. Run all of the tests at once by righting click on the project and running as JUnit tests.
- 6. If any of them are not passing (no green bar), you have not completed the assignment.

If there are any compilation errors, I reserve the right to assign a 0 grade and await a late resubmission. If all tests are not passing, I reserve the right to assign a 0 grade and await a late resubmission.

UML Generation

After you are done, you are required to generate UML for your submission. Right click on src -> Diagrams -> Show Diagram... -> Java Classes Click on each of the 5 buttons at the top-left of the screen:



Make sure NOT to click on the Properties button (X'd out above).

Then click on the top-right export image button to create the diagram image file.

Submission

Submit the UML image file and all Java files in a zip with no folder structure.

Do not submit any of the test files.

The zip file name should include the **Assignment Prefix (A1, A2, etc), your email ID** and **full name** in the file name, exactly as such: A1-bv49-boris-valerstein.zip

When you open your zip file, you should see Java files, not a folder (unless you are on a Mac, see attached instructions).

If you have any submission comments, put them into the submission box in BB Learn when submitting.

If your zip file does not follow these conventions, I reserve the right to assign a 0 grade and await a late resubmission.

Grading:

Compiling code and passing all tests is the bare minimum to receive a grade, far from a guarantee to get 100%.

Checklist:

Components are designed correctly

Small methods

No comments, in favor of more methods with expressive names

Quality variable names

Following the provided naming conventions (Java industry standard)

I should be able to read your code like a book, not struggle

Default package is used by all your classes

Zip file name is correct

Zip contains no directory structure

Zip contains all code + UML, but no test files

Make sure your code is formatted correctly!! (All of the settings above)

This course is not about me seeing if you can code. It is about encouraging you to code better.