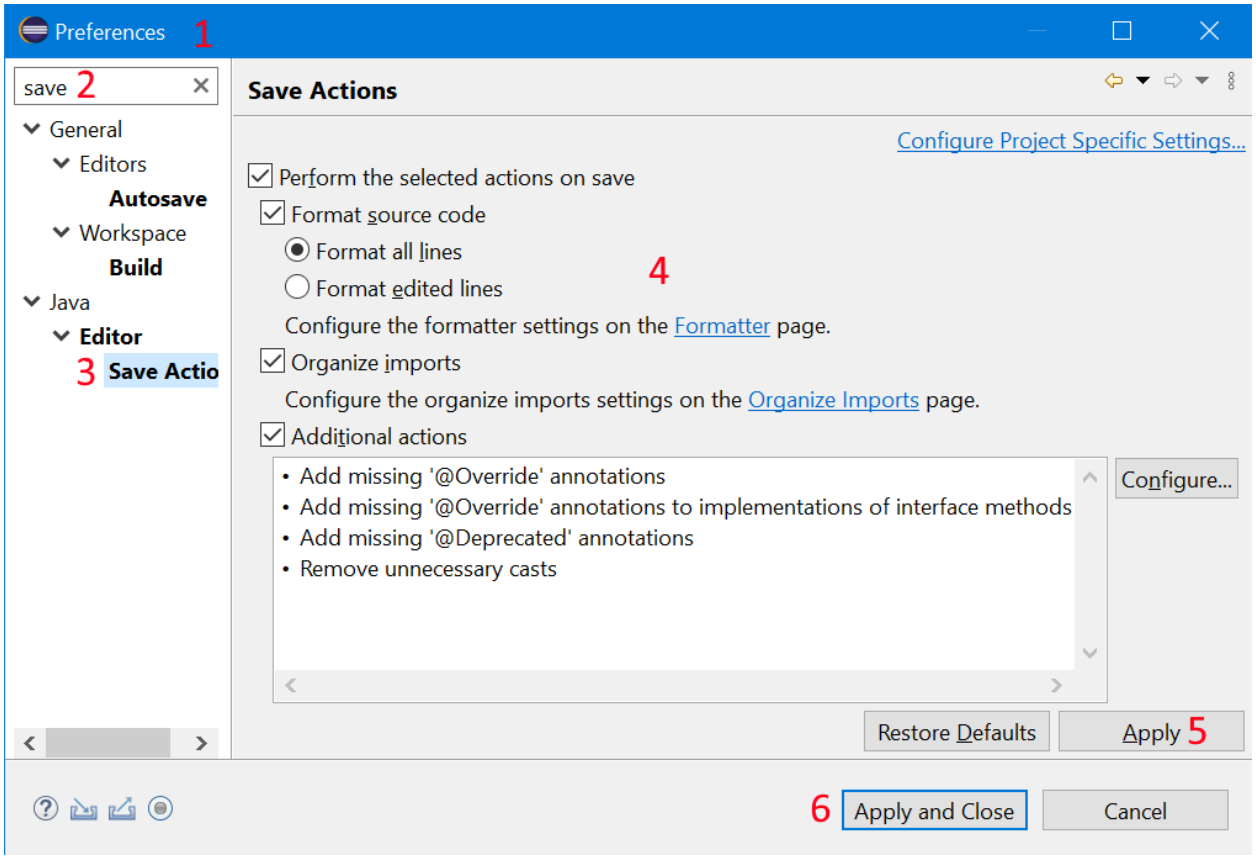
Key Word In Context

WARNING: This assignment is a test of following directions.

This setup is required:

Install Eclipse and the most recent Java JDK/JRE.
Create a new Eclipse Java Project. **Do not create module-info.java if prompted.**
Go to Window -> Preferences
Type “save”
Click on Java -> Editor -> Save Actions
Check everything then click Apply and then click “Apply and Close”



Guidelines:

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.
Ignore casing when alphabetizing.
You must code your own alphabetizing solution. Do not Google, do not use libraries or utils. Must be your own.
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 Eclipse project and run them as a guideline to see if you're on the right track:
Right click on the project -> New -> Source Folder -> "test".
Copy the files into this new source folder.
You can run them by right clicking on a file -> Run As -> JUnit Test.
You can run all of them at once by right clicking on the project -> Run As -> JUnit Test
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. **JUnit 5 is used, please make sure to import the correct JUnit into your build path.**
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.

Submission

UML is required for all assignment submissions.
Turn in a **one page PDF or image file** of your UML generated from a **computer program** (ObjectAid Eclipse plugin is recommended). The UML should include all classes, fields, methods, and relationships between classes. The UML should not include the tests or the package name in the UML (since all will be "default").

Submit the UML 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 **your ID** and **full name** in the file name. For example: A1-bv49-boris-valerstein.zip
If you have any submission comments, put them into a README.txt in the zip. If you have nothing to say or add, please **don't** add a README – it'll just waste my time and I have to grade a lot of these.

Rubric:
Functionality – 50 points
    This includes following directions.
Clean Code – 45 points
    Some examples (but not everything):
        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
UML – 5 points

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