Java Code Formatting Standards

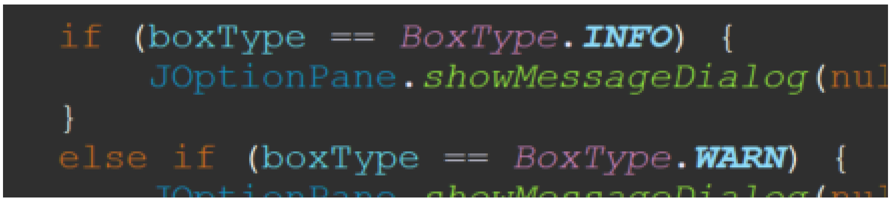
To keep code consistent and formatted the same across everyone's code, we will need to employ a method to format all code to be consistent and the same with one-another.

Below I have listed some of the rules you should follow when programming, however you should also setup your auto formatter in your IDE to format your code as closely as possible to this.

Eclipse users, I have added in screenshots and instructions of the *exact* format to use. For everyone else, find out how to modify it in your editor, and try and replicate it as closely as possible.

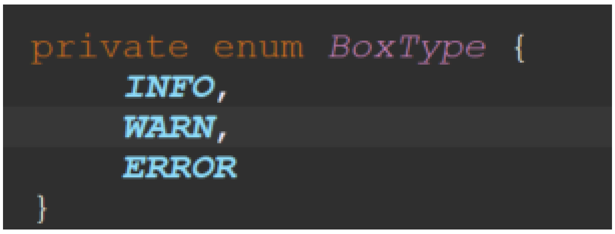
Rules to follow for formatting:

- New Lines
    - New lines before all conditional statements as well as loops
    - New lines between try catch and if else. I.e. the “else if” should be on a new line, not on the same line as the brace above it. e.g.



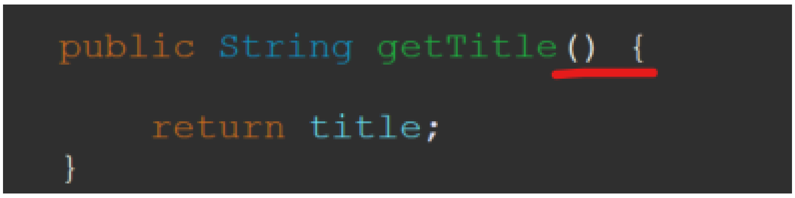
```
if (boxType == BoxType.INFO) {
    JOptionPane.showMessageDialog(nul
}
else if (boxType == BoxType.WARN) {
    JOptionPane.showMessageDialog(nul
```

- All enums and other list declarations should each be on new lines e.g.



```
private enum BoxType {
    INFO,
    WARN,
    ERROR
}
```

- All packages should consist of lowercase letters
- All class names should be upper case for the first letter of each word
- Follow camelCase syntax e.g. All variable names should have a lowercase letter for the first letter
- All opening braces should be on the same line as the statement e.g.



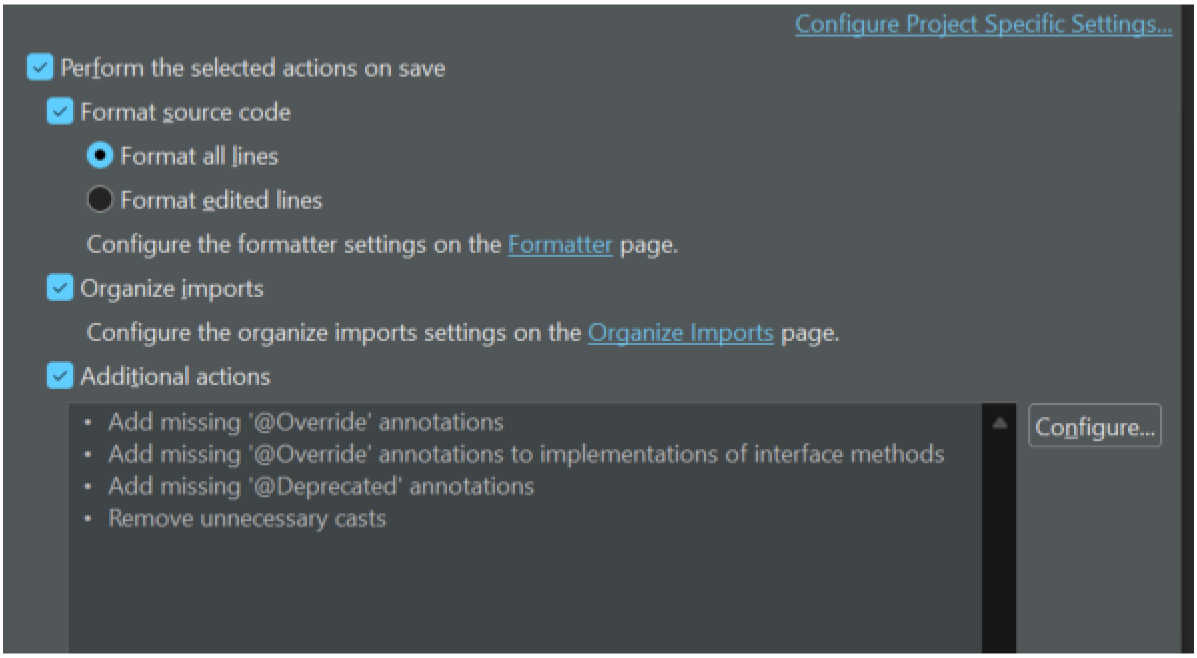
```
public String getTitle() {

    return title;
}
```

- **All tab space indents should be as spaces (not tab characters) and should be 4 spaces wide.** This doesn’t mean you need to press the space key 4 times each.. You can still use the tab key but set it so it indents all occurrences by 4 spaces, rather than 1 tab character.

Instructions for how to setup the auto formatter in eclipse. (Links for reference, please read on below)
https://stackoverflow.com/questions/15655126/how-to-auto-format-code-in-eclipse
https://stackoverflow.com/questions/1392744/how-do-i-configure-autoformatting-in-eclipse

You can either use the autoformatter manually, by using “Ctrl + Shift + F” (or on mac, ⇧ ⌘ F) or you can set it up to automatically apply the formatting when you save the file by doing the following: (Also make sure to select the additional settings I have ticked too!)

- **Window->Preferences->Java->Editor->SaveActions** and check the **Format source code**



Configure Project Specific Settings...

☑ Perform the selected actions on save

☑ Format source code

● Format all lines

○ Format edited lines

Configure the formatter settings on the Formatter page.

☑ Organize imports

Configure the organize imports settings on the Organize Imports page.

☑ Additional actions

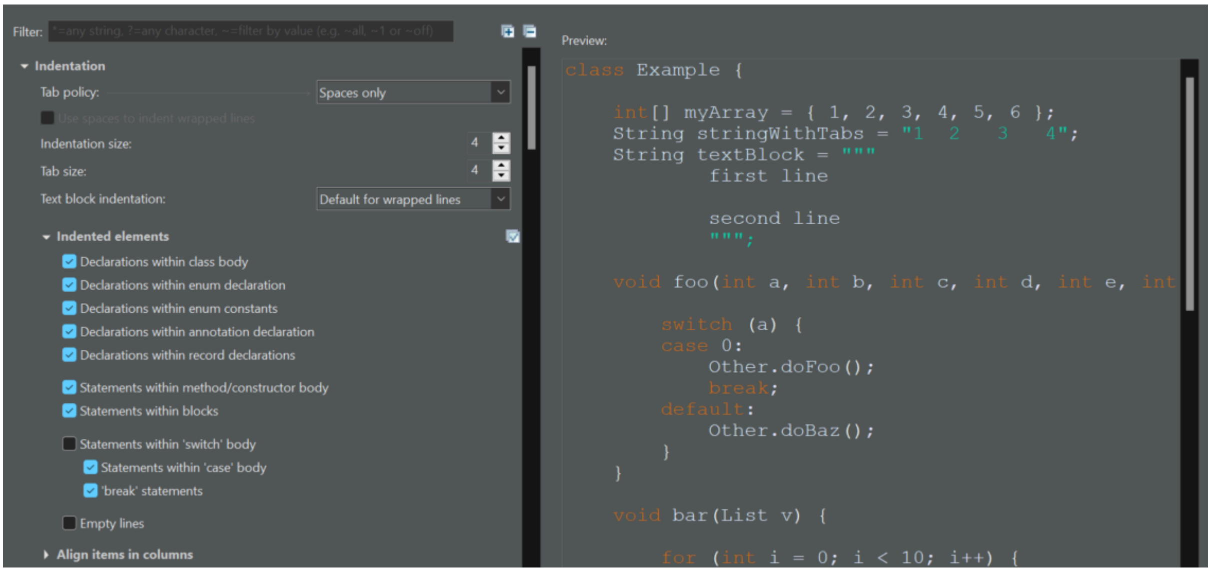
- Add missing '@Override' annotations
- Add missing '@Override' annotations to implementations of interface methods
- Add missing '@Deprecated' annotations
- Remove unnecessary casts

Setting up the auto formatter in eclipse

Follow the following steps to setup the auto formatter for eclipse based on the conventions and setup I listed at the start of this document.

- Go to **Windows -> Preferences**, and in the options table select **Java -> Code Style -> Formatter**
- Select the default eclipse formatter and click new.
- Give the profile a name, e.g. DocDuck format or SWEng format or Custom Format.. Whatever you want
- And initialise it with the built in eclipse profile
- Now, go to the following sections shown in the screenshots below and make sure each checkbox and setting is the same as shown below

1.



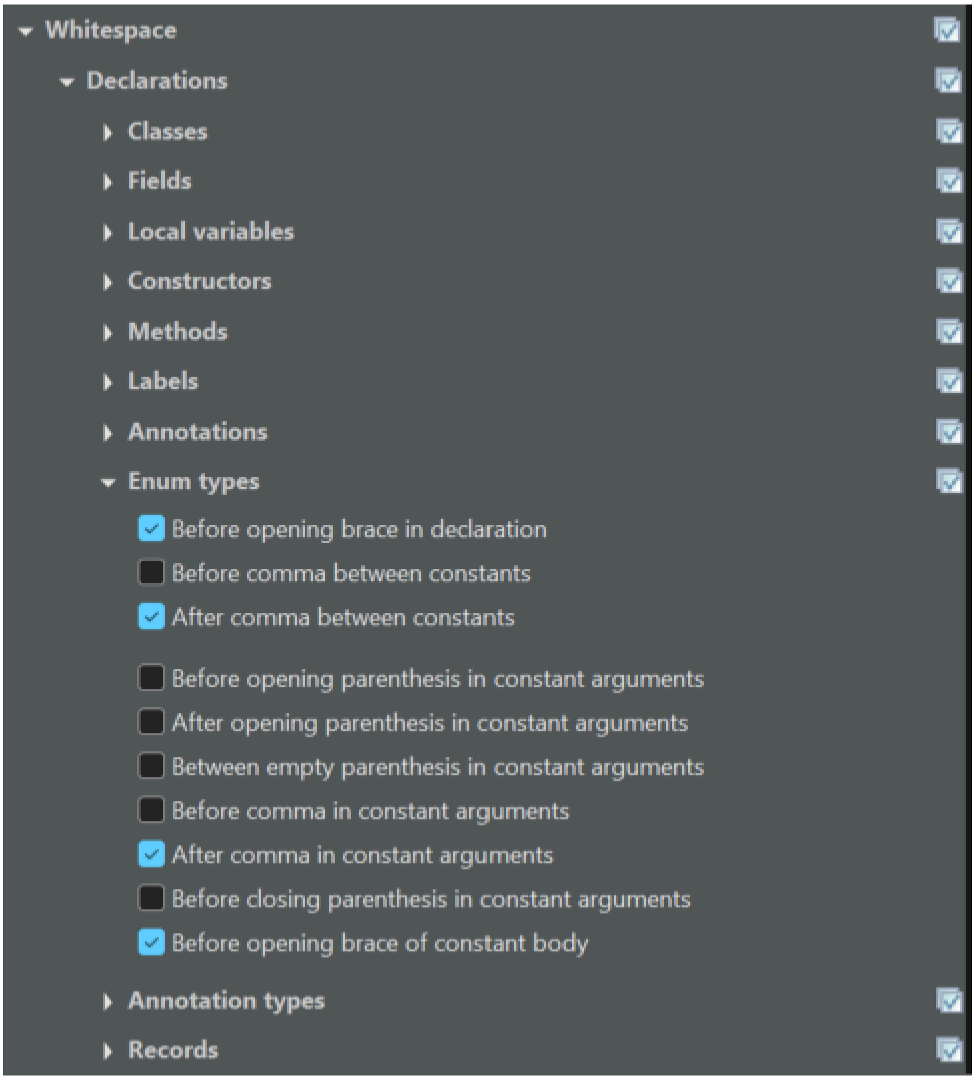
Filter: *=any string, ?=any character, ~=filter by value (e.g. ~all, ~1 or ~off)

- Indentation
  - Tab policy: Spaces only
  - □ Use spaces to indent wrapped lines
  - Indentation size: 4
  - Tab size: 4
  - Text block indentation: Default for wrapped lines
  - Indented elements ☑
    - ☑ Declarations within class body
    - ☑ Declarations within enum declaration
    - ☑ Declarations within enum constants
    - ☑ Declarations within annotation declaration
    - ☑ Declarations within record declarations
    - ☑ Statements within method/constructor body
    - ☑ Statements within blocks
    - □ Statements within 'switch' body
      - ☑ Statements within 'case' body
      - ☑ 'break' statements
    - □ Empty lines
  - Align items in columns

Preview:

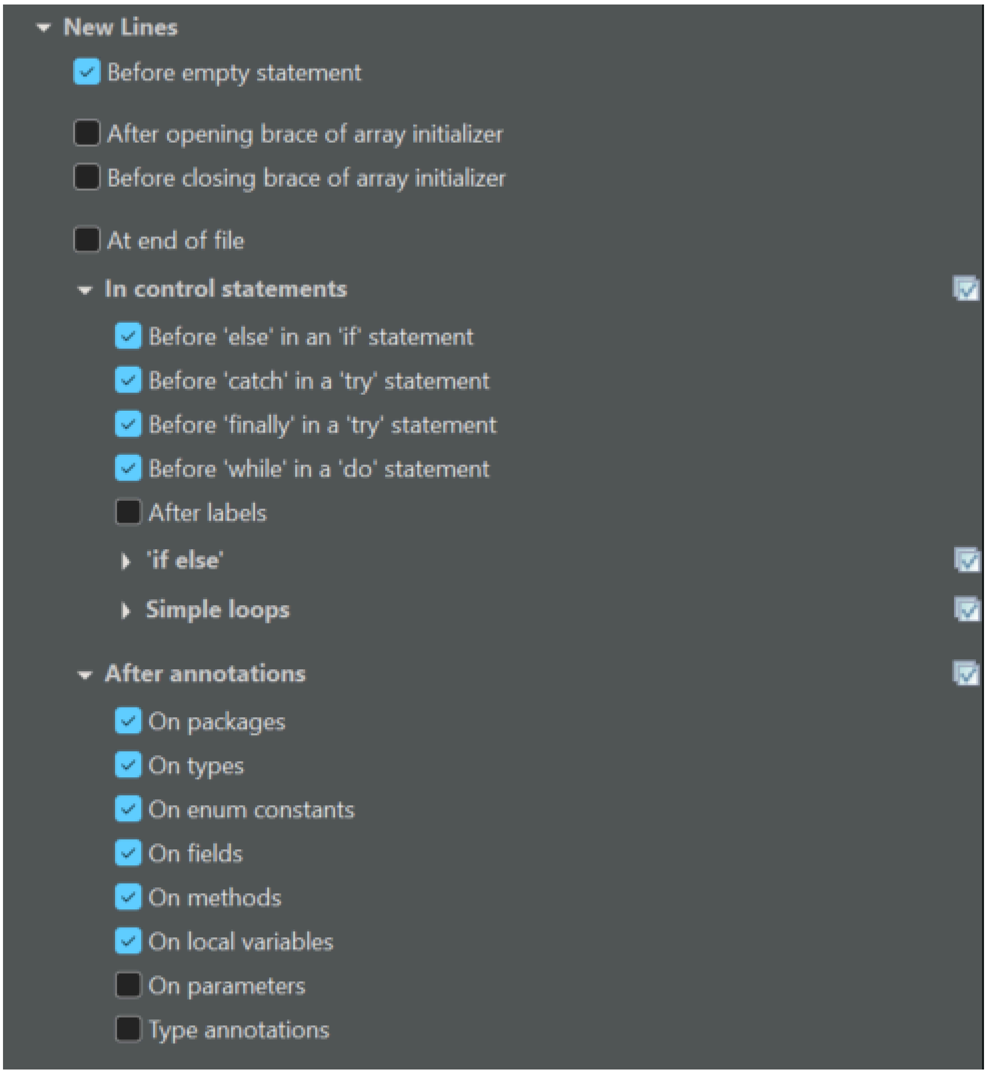
```
class Example {

    int[] myArray = { 1, 2, 3, 4, 5, 6 };
    String stringWithTabs = "1  2   3   4";
    String textBlock = """
            first line

            second line
            """;

    void foo(int a, int b, int c, int d, int e, int

        switch (a) {
        case 0:
            Other.doFoo();
            break;
        default:
            Other.doBaz();
        }
    }

    void bar(List v) {

        for (int i = 0; i < 10; i++) {
```

2.

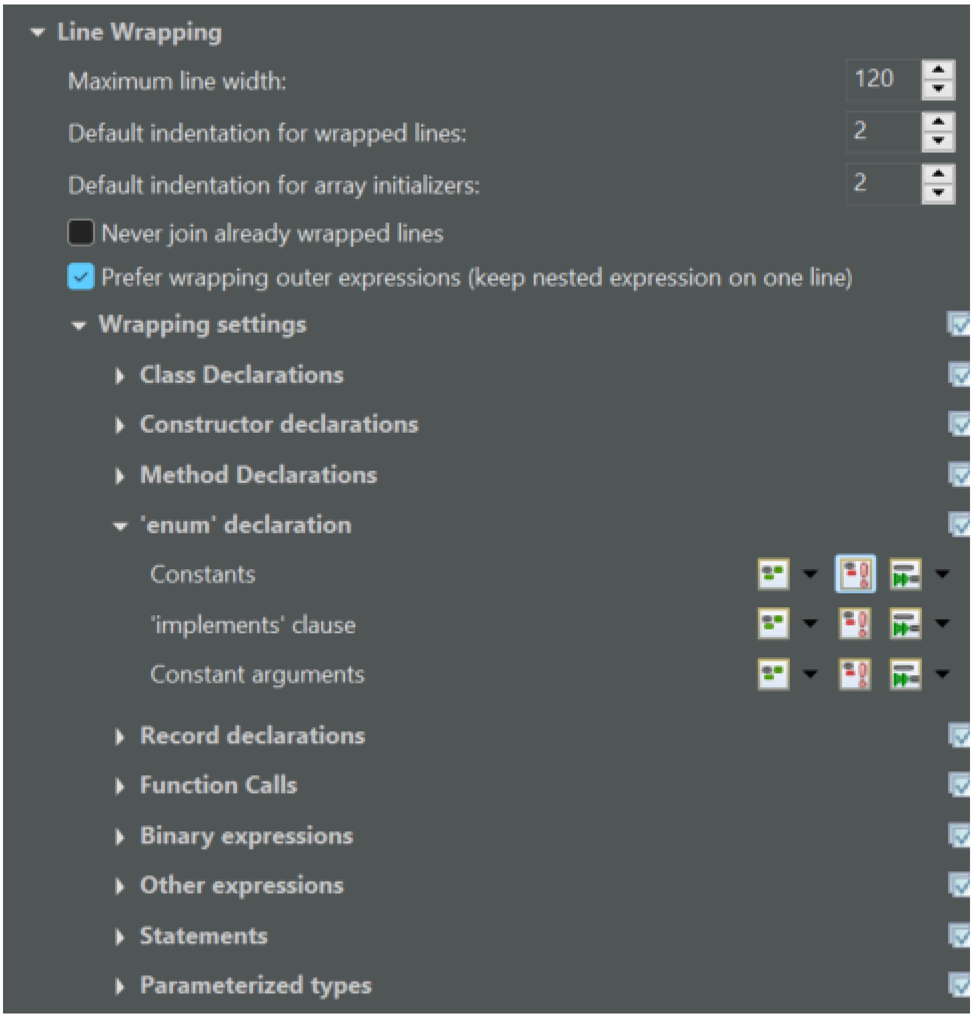


- Whitespace ☑
  - Declarations ☑
    - Classes ☑
    - Fields ☑
    - Local variables ☑
    - Constructors ☑
    - Methods ☑
    - Labels ☑
    - Annotations ☑
    - Enum types ☑
      - ☑ Before opening brace in declaration
      - □ Before comma between constants
      - ☑ After comma between constants
      - □ Before opening parenthesis in constant arguments
      - □ After opening parenthesis in constant arguments
      - □ Between empty parenthesis in constant arguments
      - □ Before comma in constant arguments
      - ☑ After comma in constant arguments
      - □ Before closing parenthesis in constant arguments
      - ☑ Before opening brace of constant body
    - Annotation types ☑
    - Records ☑

3.



- New Lines
  - ☑ Before empty statement
  - ☐ After opening brace of array initializer
  - ☐ Before closing brace of array initializer
  - ☐ At end of file
  - In control statements ☑
    - ☑ Before 'else' in an 'if' statement
    - ☑ Before 'catch' in a 'try' statement
    - ☑ Before 'finally' in a 'try' statement
    - ☑ Before 'while' in a 'do' statement
    - ☐ After labels
    - 'if else' ☑
    - Simple loops ☑
  - After annotations ☑
    - ☑ On packages
    - ☑ On types
    - ☑ On enum constants
    - ☑ On fields
    - ☑ On methods
    - ☑ On local variables
    - ☐ On parameters
    - ☐ Type annotations

4. 

Line Wrapping

Maximum line width: 120

Default indentation for wrapped lines: 2

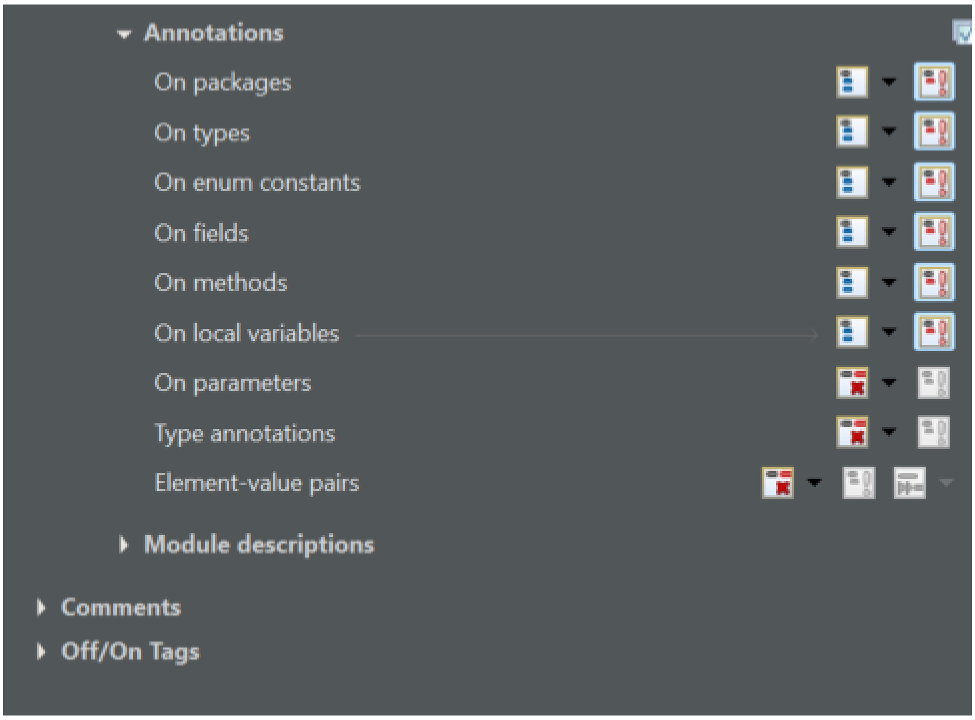
Default indentation for array initializers: 2

☐ Never join already wrapped lines

☑ Prefer wrapping outer expressions (keep nested expression on one line)

- Wrapping settings
  - Class Declarations
  - Constructor declarations
  - Method Declarations
  - 'enum' declaration
    - Constants
    - 'implements' clause
    - Constant arguments
  - Record declarations
  - Function Calls
  - Binary expressions
  - Other expressions
  - Statements
  - Parameterized types

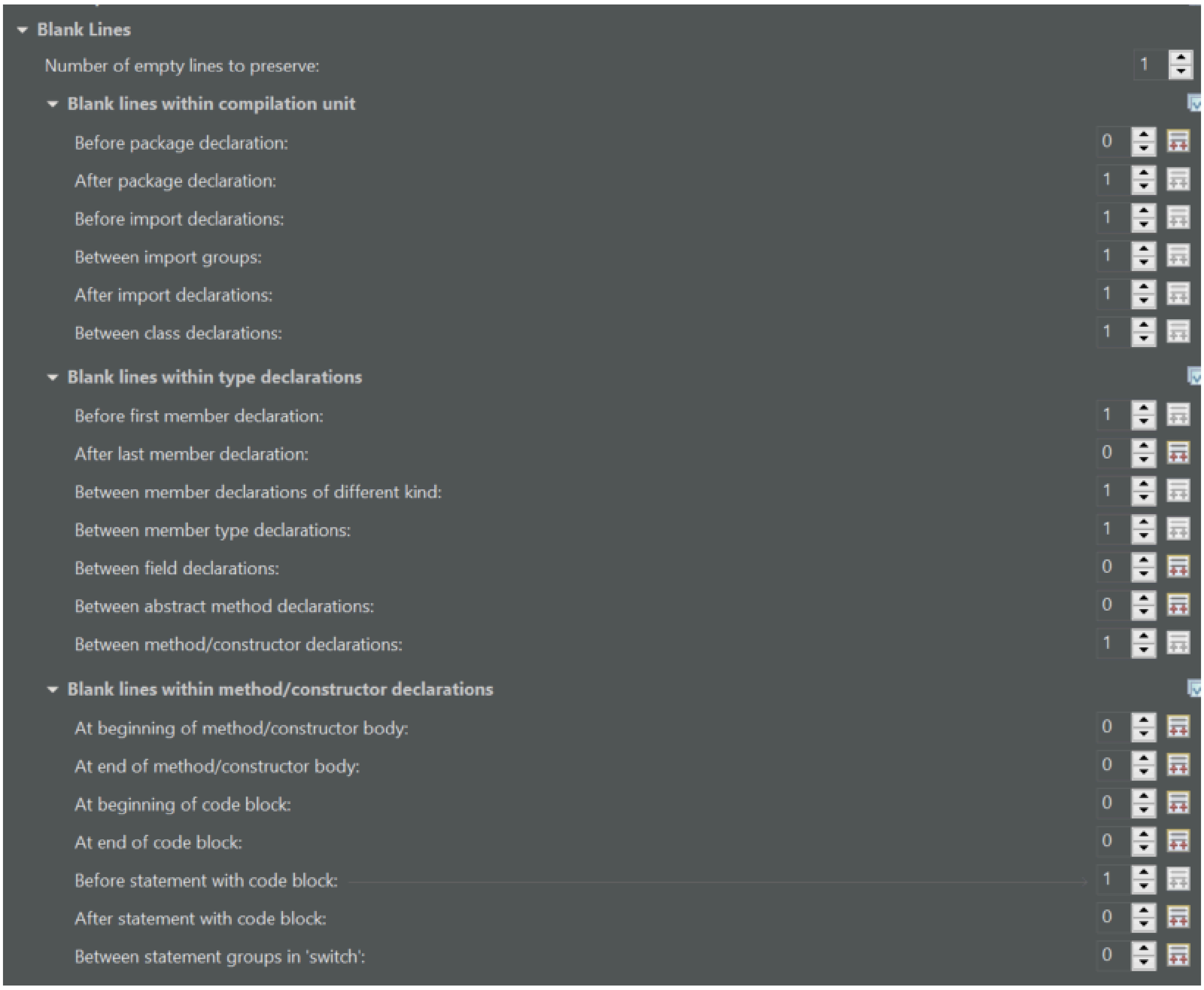
Take note of the enum declaration icons here… The constants middle icon is selected.. Also the column width

5. 

- Annotations
  - On packages
  - On types
  - On enum constants
  - On fields
  - On methods
  - On local variables
  - On parameters
  - Type annotations
  - Element-value pairs
- Module descriptions

- Comments
- Off/On Tags

6.



- Blank Lines
  - Number of empty lines to preserve: 1
  - Blank lines within compilation unit
    - Before package declaration: 0
    - After package declaration: 1
    - Before import declarations: 1
    - Between import groups: 1
    - After import declarations: 1
    - Between class declarations: 1
  - Blank lines within type declarations
    - Before first member declaration: 1
    - After last member declaration: 0
    - Between member declarations of different kind: 1
    - Between member type declarations: 1
    - Between field declarations: 0
    - Between abstract method declarations: 0
    - Between method/constructor declarations: 1
  - Blank lines within method/constructor declarations
    - At beginning of method/constructor body: 0
    - At end of method/constructor body: 0
    - At beginning of code block: 0
    - At end of code block: 0
    - Before statement with code block: 1
    - After statement with code block: 0
    - Between statement groups in 'switch': 0

If you have followed everything correctly you shouldn't need to modify any other values as they will be set by default by eclipse..

Now you can test this out by writing a short class, or opening some existing code and either manually auto formatting, or by modifying and then saving the file to apply the auto save formatter feature (if you set this up)

If it doesn't look quite right for all positions and formatting options mentioned at the start, try adjusting them yourself to make it more accurate (though do not modify any of the ones that you set based on the screenshots for the auto formatter (as I have set the correctly)
If you are still unsure, please send me a message with a screenshot of the code which isn't formatted correctly.

For intellij or vs code users.. Please look up how to set up this yourself. Or alternatively find an auto formatter online that you can copy and paste code into.