COURSE GENERATOR development notes
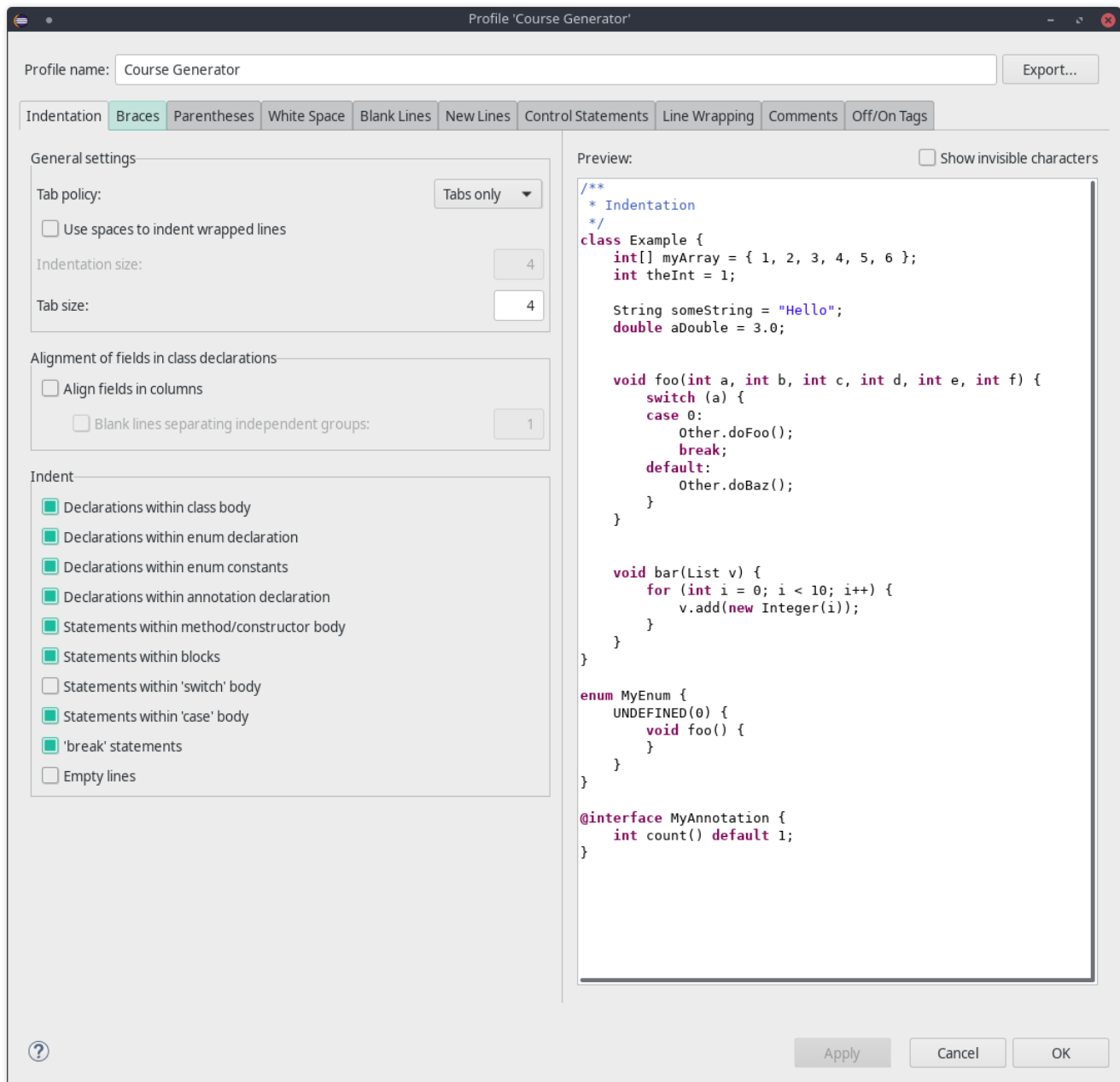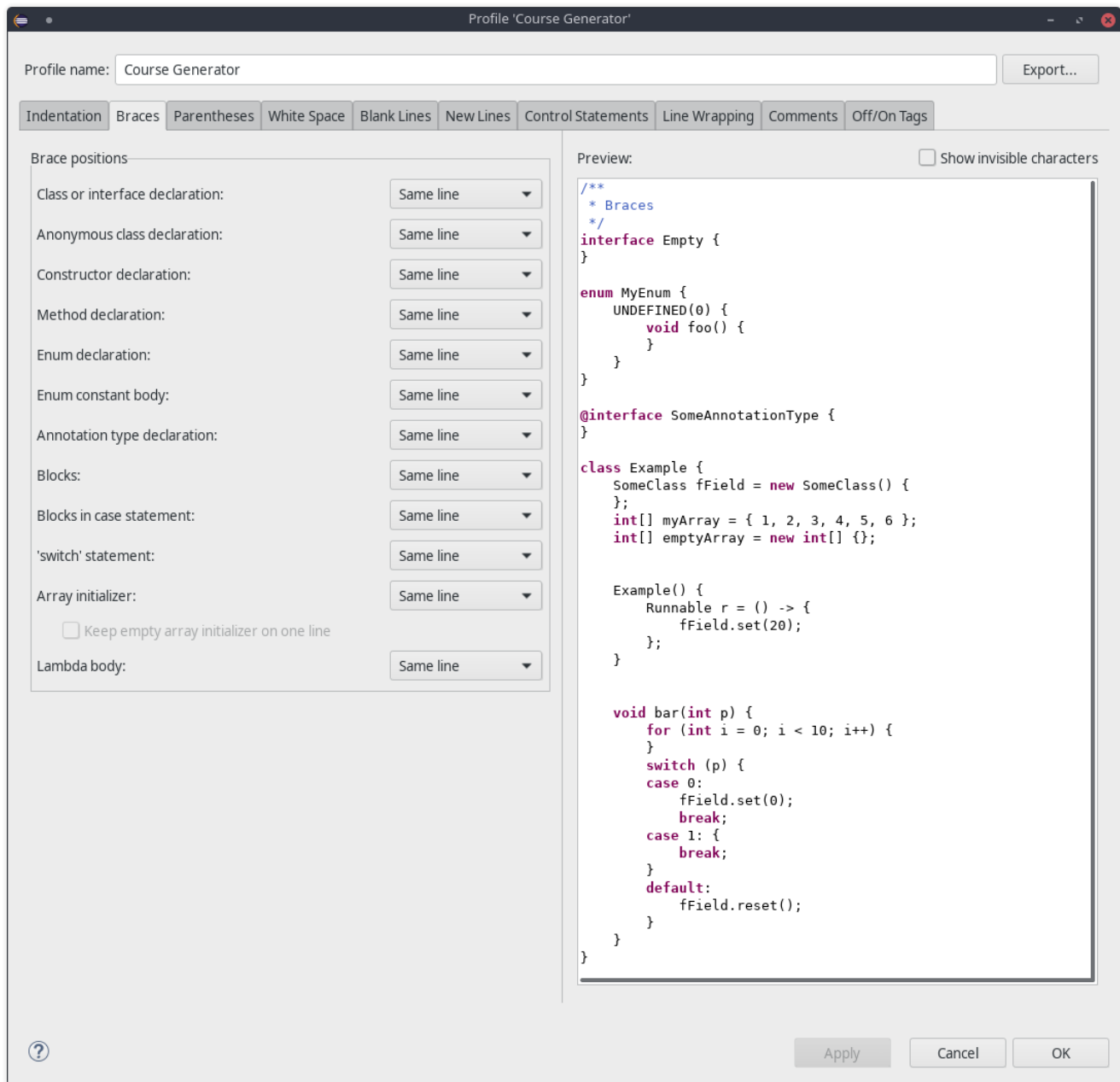
Tools used

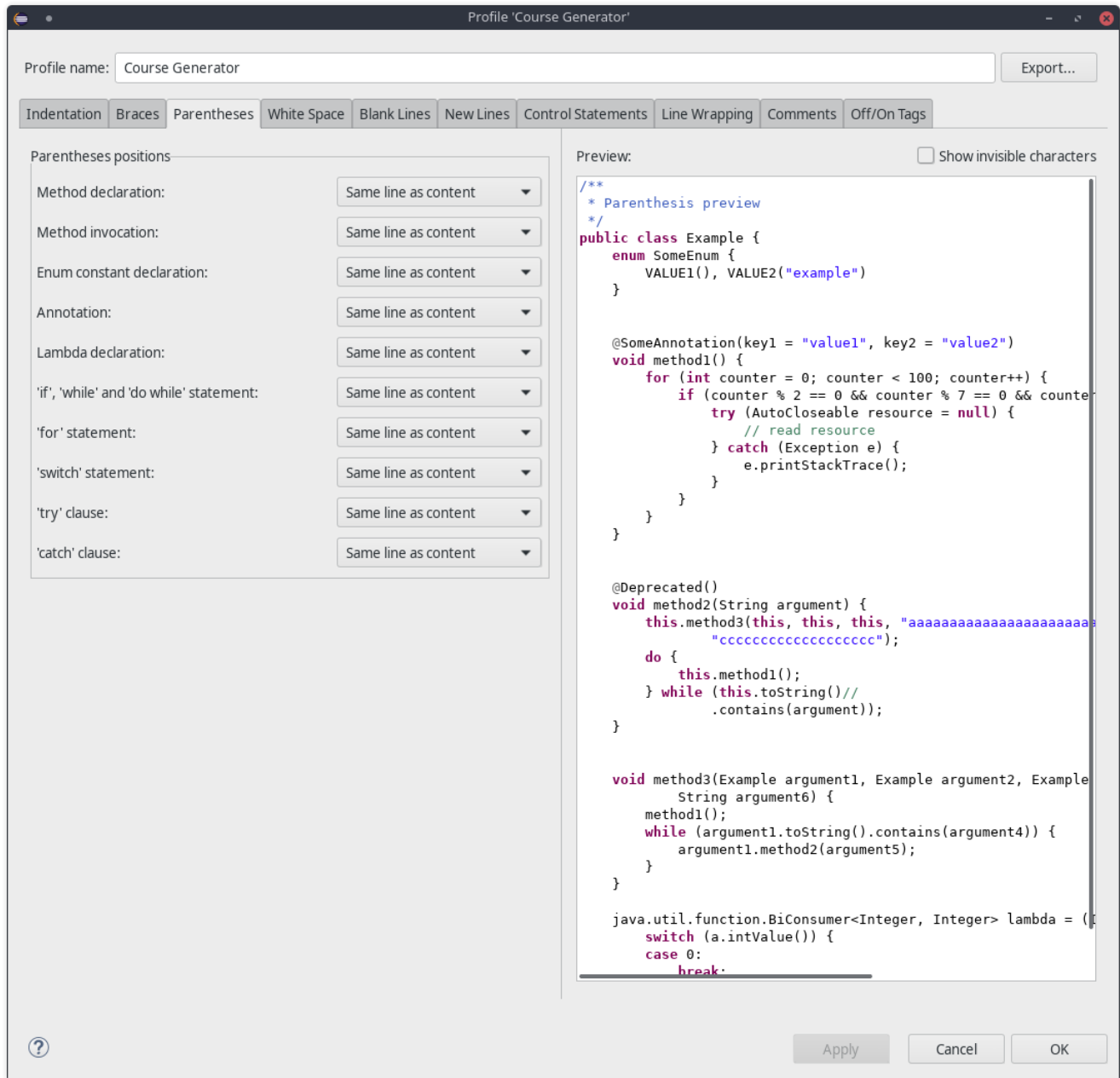- Eclipse version Oxygen.3a Release (4.7.3a)
- Git

Course Generator formatter settings
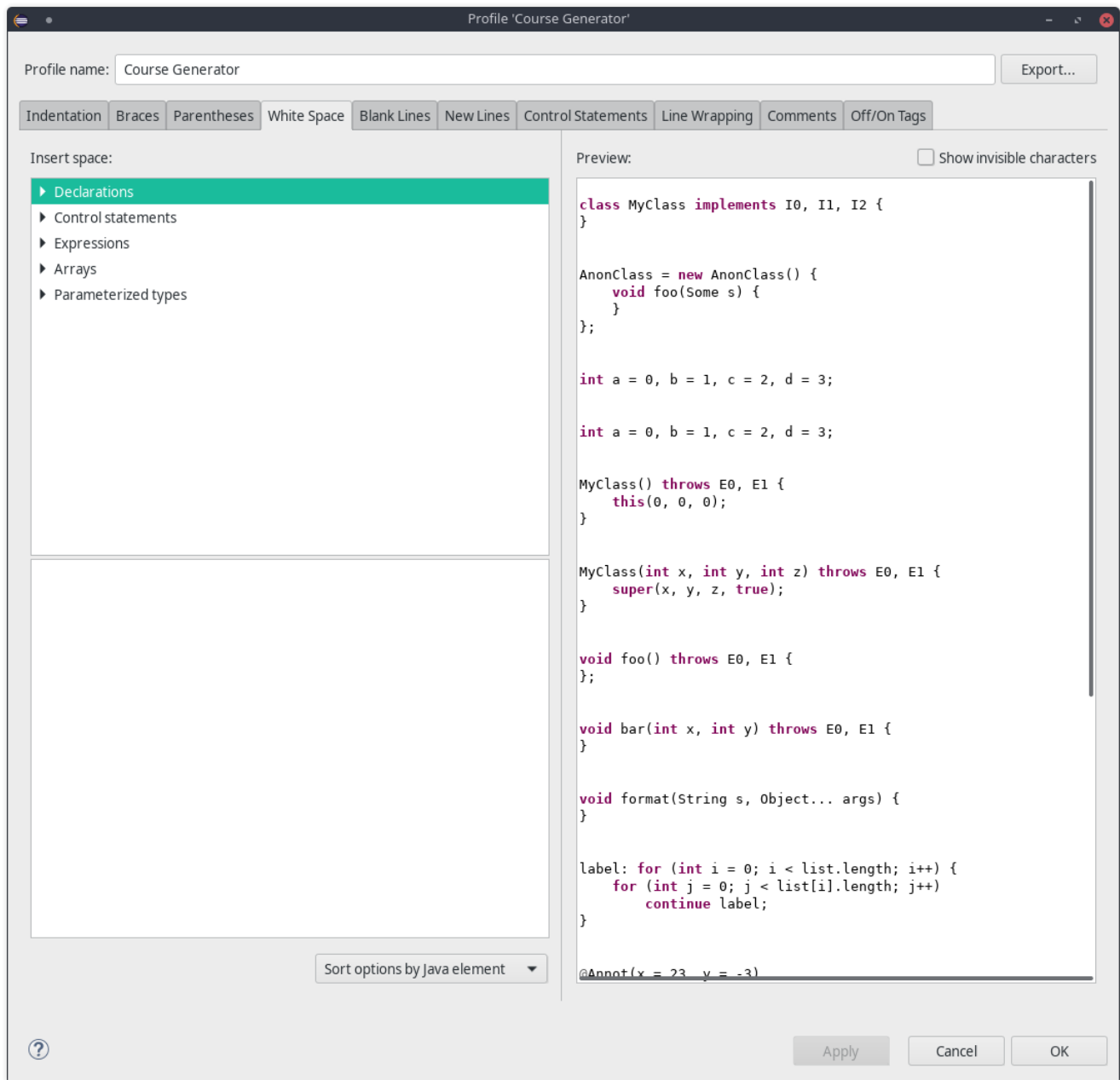
Where to apply the settings : Preference>Java>Code Style>Formatter

Settings :

Profile 'Course Generator'

Profile name: Course Generator   Export...

Indentation | Braces | Parentheses | White Space | Blank Lines | New Lines | Control Statements | Line Wrapping | Comments | Off/On Tags

General settings   Preview: ☐ Show invisible characters

Tab policy: Tabs only ▾
                                                      ```
☐ Use spaces to indent wrapped lines                  /**
                                                       * Indentation
Indentation size:                        4             */
                                                      class Example {
Tab size:                                4                 int[] myArray = { 1, 2, 3, 4, 5, 6 };
                                                          int theInt = 1;
Alignment of fields in class declarations
                                                          String someString = "Hello";
☐ Align fields in columns                                 double aDouble = 3.0;

   ☐ Blank lines separating independent groups:   1        void foo(int a, int b, int c, int d, int e, int f) {
                                                              switch (a) {
Indent                                                        case 0:
                                                                  Other.doFoo();
☑ Declarations within class body                                  break;
                                                              default:
☑ Declarations within enum declaration                            Other.doBaz();
                                                              }
☑ Declarations within enum constants                      }

☑ Declarations within annotation declaration
                                                          void bar(List v) {
☑ Statements within method/constructor body                   for (int i = 0; i < 10; i++) {
                                                                  v.add(new Integer(i));
☑ Statements within blocks                                    }
                                                          }
☐ Statements within 'switch' body                     }

☑ Statements within 'case' body                       enum MyEnum {
                                                          UNDEFINED(0) {
☑ 'break' statements                                          void foo() {
                                                              }
☐ Empty lines                                             }
                                                      }

                                                      @interface MyAnnotation {
                                                          int count() default 1;
                                                      }
                                                      ```

                                                    Apply   Cancel   OK

Profile 'Course Generator'

Profile name: Course Generator | Export...

Indentation | **Braces** | **Parentheses** | **White Space** | **Blank Lines** | **New Lines** | **Control Statements** | **Line Wrapping** | **Comments** | **Off/On Tags**

Brace positions

Class or interface declaration:	Same line ▼
Anonymous class declaration:	Same line ▼
Constructor declaration:	Same line ▼
Method declaration:	Same line ▼
Enum declaration:	Same line ▼
Enum constant body:	Same line ▼
Annotation type declaration:	Same line ▼
Blocks:	Same line ▼
Blocks in case statement:	Same line ▼
'switch' statement:	Same line ▼
Array initializer:	Same line ▼
☐ Keep empty array initializer on one line	
Lambda body:	Same line ▼

Preview: ☐ Show invisible characters

```java
/**
 * Braces
 */
interface Empty {
}

enum MyEnum {
    UNDEFINED(0) {
        void foo() {
        }
    }
}

@interface SomeAnnotationType {
}

class Example {
    SomeClass fField = new SomeClass() {
    };
    int[] myArray = { 1, 2, 3, 4, 5, 6 };
    int[] emptyArray = new int[] {};

    Example() {
        Runnable r = () -> {
            fField.set(20);
        };
    }

    void bar(int p) {
        for (int i = 0; i < 10; i++) {
        }
        switch (p) {
        case 0:
            fField.set(0);
            break;
        case 1: {
            break;
        }
        default:
            fField.reset();
        }
    }
}
```

Apply | Cancel | OK

Profile 'Course Generator'

Profile name: Course Generator

Export...

| Indentation | Braces | Parentheses | White Space | Blank Lines | New Lines | Control Statements | Line Wrapping | Comments | Off/On Tags |

Parentheses positions

Method declaration: Same line as content ▼

Method invocation: Same line as content ▼

Enum constant declaration: Same line as content ▼

Annotation: Same line as content ▼

Lambda declaration: Same line as content ▼

'if', 'while' and 'do while' statement: Same line as content ▼

'for' statement: Same line as content ▼

'switch' statement: Same line as content ▼

'try' clause: Same line as content ▼

'catch' clause: Same line as content ▼

Preview: ☐ Show invisible characters

```java
/**
 * Parenthesis preview
 */
public class Example {
    enum SomeEnum {
        VALUE1(), VALUE2("example")
    }


    @SomeAnnotation(key1 = "value1", key2 = "value2")
    void method1() {
        for (int counter = 0; counter < 100; counter++) {
            if (counter % 2 == 0 && counter % 7 == 0 && counter
                try (AutoCloseable resource = null) {
                    // read resource
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }
    }


    @Deprecated()
    void method2(String argument) {
        this.method3(this, this, this, "aaaaaaaaaaaaaaaaaaaaaa
                "cccccccccccccccccccc");
        do {
            this.method1();
        } while (this.toString()//
                .contains(argument));
    }


    void method3(Example argument1, Example argument2, Example
            String argument6) {
        method1();
        while (argument1.toString().contains(argument4)) {
            argument1.method2(argument5);
        }
    }

    java.util.function.BiConsumer<Integer, Integer> lambda = (
        switch (a.intValue()) {
        case 0:
            break;
```

Apply   Cancel   OK

Profile name: Course Generator

Export...

Indentation | Braces | Parentheses | White Space | Blank Lines | New Lines | Control Statements | Line Wrapping | Comments | Off/On Tags

Insert space:

Preview:

☐ Show invisible characters

▸ **Declarations**
▸ Control statements
▸ Expressions
▸ Arrays
▸ Parameterized types

```
class MyClass implements I0, I1, I2 {
}


AnonClass = new AnonClass() {
    void foo(Some s) {
    }
};


int a = 0, b = 1, c = 2, d = 3;


int a = 0, b = 1, c = 2, d = 3;


MyClass() throws E0, E1 {
    this(0, 0, 0);
}


MyClass(int x, int y, int z) throws E0, E1 {
    super(x, y, z, true);
}


void foo() throws E0, E1 {
};


void bar(int x, int y) throws E0, E1 {
}


void format(String s, Object... args) {
}


label: for (int i = 0; i < list.length; i++) {
    for (int j = 0; j < list[i].length; j++)
        continue label;
}

@Annot(x = 23, y = -3)
```

Sort options by Java element ▾

?

Apply | Cancel | OK

Profile 'Course Generator'

Profile name: Course Generator   Export...

| Indentation | Braces | Parentheses | White Space | Blank Lines | New Lines | Control Statements | Line Wrapping | Comments | Off/On Tags |

Blank lines in compilation unit

Before package declaration: `0`

After package declaration: `1`

Before import declaration: `1`

Between import groups: `1`

After import declaration: `1`

Between class declarations: `1`

Blank lines within class declarations

Before first declaration: `0`

Before declarations of the same kind: `1`

Before member class declarations: `1`

Before field declarations: `0`

Before method declarations: `2`

At beginning of method body: `0`

Existing blank lines

Number of empty lines to preserve: `1`

Preview: ☐ Show invisible characters

```java
/**
 * Blank Lines
 */
package foo.bar.baz;

import java.util.List;
import java.util.Vector;

import java.net.Socket;

public class Another {
}

public class Example {
    public static class Pair {
        public String first;
        public String second;
        // Between here...

        // ...and here are 10 blank lines
    };

    private LinkedList fList;
    public int counter;


    public Example(LinkedList list) {
        fList = list;
        counter = 0;
    }


    public void push(Pair p) {
        fList.add(p);
        ++counter;
    }


    public Object pop() {
        --counter;
        return (Pair) fList.getLast();
    }
}
```

Apply   Cancel   OK

Profile 'Course Generator'

Profile name: Course Generator   [Export...]

| Indentation | Braces | Parentheses | White Space | Blank Lines | New Lines | Control Statements | Line Wrapping | Comments | Off/On Tags |

Insert new line

- ☑ in empty class body
- ☑ in empty anonymous class body
- ☑ in empty method body
- ☑ in empty block
- ☐ after labels
- ☑ in empty enum declaration
- ☑ in empty enum constant body
- ☑ in empty annotation body
- ☐ at end of file

Array initializers

- ☐ Insert new line after opening brace of array initializer
- ☐ Insert new line before closing brace of array initializer

Empty statements

- ☑ Put empty statement on new line

Annotations

- ☑ Insert new line after annotations on packages
- ☑ Insert new line after annotations on types
- ☑ Insert new line after annotations on enum constants
- ☑ Insert new line after annotations on fields
- ☑ Insert new line after annotations on methods
- ☑ Insert new line after annotations on local variables
- ☐ Insert new line after annotations on parameters
- ☐ Insert new line after type annotations

Preview:

☐ Show invisible characters

```
/**
* New Lines
*/
@Deprecated
package com.example; // annotation on package is only allowed

public class Empty {
}

@Deprecated
class Example {
    @Deprecated
    static int[] fArray = { 1, 2, 3, 4, 5 };
    Listener fListener = new Listener() {
    };


    @Deprecated
    @Override
    public void bar(@SuppressWarnings("unused") int i) {
        @SuppressWarnings("unused")
        final @Positive int k;
    }


    void foo() {
        ;
        ;
        label: do {
        } while (false);
        for (;;) {
        }
    }
}

enum MyEnum {
    @Deprecated
    UNDEFINED(0) {
    }
}

enum EmptyEnum {
}

@interface EmptyAnnotation {
}
```

[Apply]   [Cancel]   [OK]

Profile 'Course Generator'

Profile name: Course Generator

Export...

| Indentation | Braces | Parentheses | White Space | Blank Lines | New Lines | Control Statements | Line Wrapping | Comments | Off/On Tags |

General

- [ ] Insert new line before 'else' in an 'if' statement
- [ ] Insert new line before 'catch' in a 'try' statement
- [ ] Insert new line before 'finally' in a 'try' statement
- [ ] Insert new line before 'while' in a 'do' statement

'if else'

- [ ] Keep 'then' statement on same line
  - [ ] Keep simple 'if' on one line
- [ ] Keep 'else' statement on same line
- [x] Keep 'else if' on one line
- [ ] Keep 'return' or 'throw' clause on one line

Preview:

[ ] Show invisible characters

```
/**
 * If...else
 */
class Example {
    void bar() {
        do {
        } while (true);
        try {
        } catch (Exception e) {
        } finally {
        }
    }

    void foo2() {
        if (true) {
            return;
        }
        if (true) {
            return;
        } else if (false) {
            return;
        } else {
            return;
        }
    }

    void foo(int state) {
        if (true)
            return;
        if (true)
            return;
        else if (false)
            return;
        else
            return;
    }
}
```

Apply   Cancel   OK

Profile 'Course Generator'

Profile name: Course Generator | Export...

Indentation | Braces | Parentheses | White Space | Blank Lines | New Lines | Control Statements | Line Wrapping | Comments | Off/On Tags

General settings

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)

- ▼ Annotations
  - **Element-value pairs**
- ▼ Class Declarations
  - 'extends' clause
  - 'implements' clause
- ▼ Constructor declarations
  - Parameters
  - 'throws' clause
- ▼ Method Declarations
  - Declaration
  - Parameters
  - 'throws' clause
- ▼ 'enum' declaration
  - Constants
  - 'implements' clause

Settings for element-value pairs

Line wrapping policy:

Do not wrap ▼

☐ Force split, even if line shorter than maximum line width

Indentation policy:
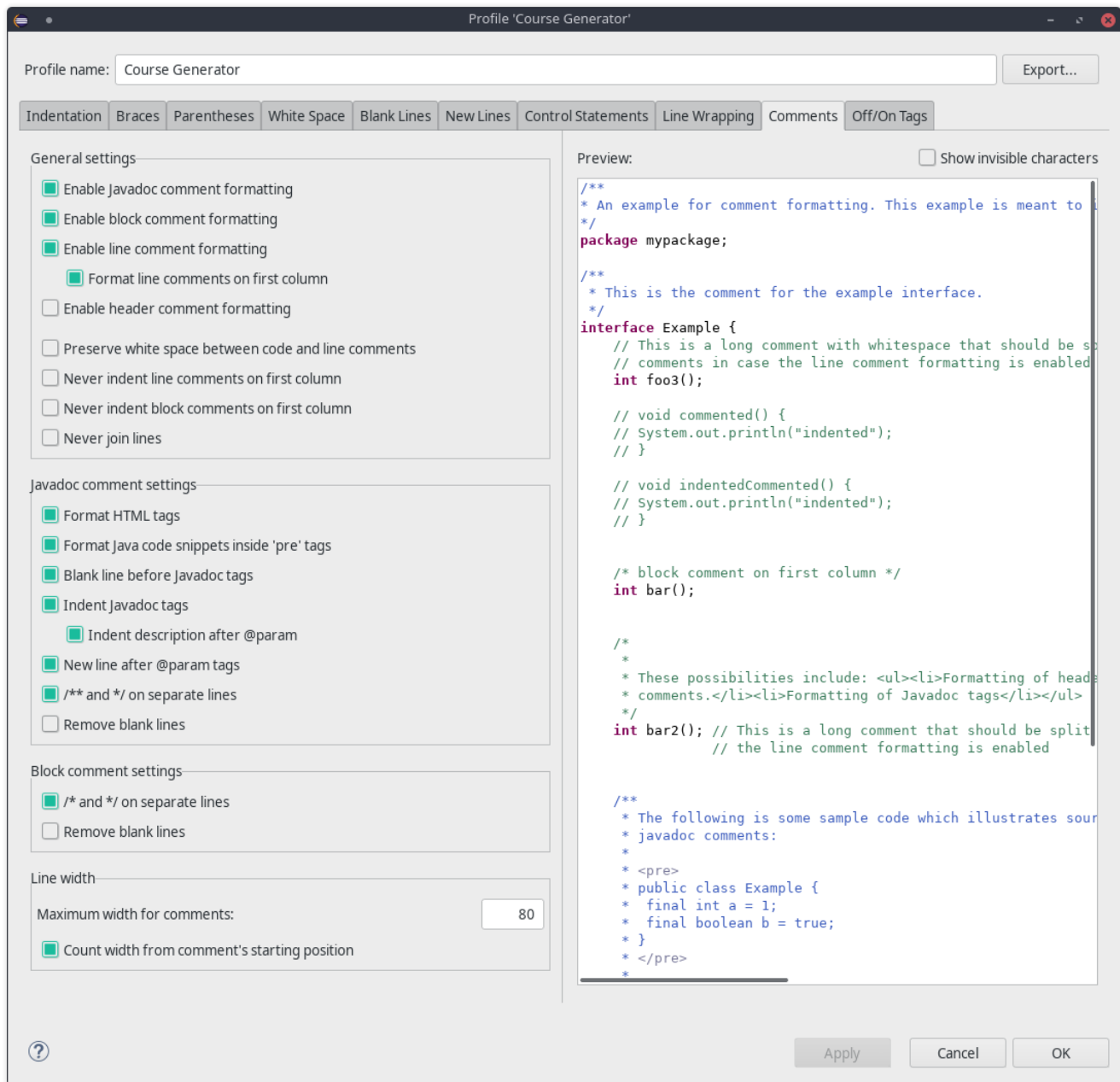
Default indentation ▼

Preview: | ☐ Show invisible characters

```
/**
 * Element-value pairs
 */
@MyAnnotation(value1 = "this is an example", value2 = "of an an
class Example {
}
```

Set line width for preview window: 40

? | Apply | Cancel | OK

Profile 'Course Generator'

Profile name: Course Generator | Export...

| Indentation | Braces | Parentheses | White Space | Blank Lines | New Lines | Control Statements | Line Wrapping | Comments | Off/On Tags |

General settings

- ☑ Enable Javadoc comment formatting
- ☑ Enable block comment formatting
- ☑ Enable line comment formatting
  - ☑ Format line comments on first column
- ☐ Enable header comment formatting

- ☐ Preserve white space between code and line comments
- ☐ Never indent line comments on first column
- ☐ Never indent block comments on first column
- ☐ Never join lines

Javadoc comment settings

- ☑ Format HTML tags
- ☑ Format Java code snippets inside 'pre' tags
- ☑ Blank line before Javadoc tags
- ☑ Indent Javadoc tags
  - ☑ Indent description after @param
- ☑ New line after @param tags
- ☑ /** and */ on separate lines
- ☐ Remove blank lines

Block comment settings

- ☑ /* and */ on separate lines
- ☐ Remove blank lines

Line width

Maximum width for comments: [80]
- ☑ Count width from comment's starting position

Preview: ☐ Show invisible characters

```
/**
* An example for comment formatting. This example is meant to
*/
package mypackage;

/**
 * This is the comment for the example interface.
 */
interface Example {
    // This is a long comment with whitespace that should be s
    // comments in case the line comment formatting is enabled
    int foo3();

    // void commented() {
    // System.out.println("indented");
    // }

    // void indentedCommented() {
    // System.out.println("indented");
    // }


    /* block comment on first column */
    int bar();


    /*
     *
     * These possibilities include: <ul><li>Formatting of head
     * comments.</li><li>Formatting of Javadoc tags</li></ul>
     */
    int bar2(); // This is a long comment that should be split
                // the line comment formatting is enabled


    /**
     * The following is some sample code which illustrates sour
     * javadoc comments:
     *
     * <pre>
     * public class Example {
     *   final int a = 1;
     *   final boolean b = true;
     * }
     * </pre>
     *
```

Apply | Cancel | OK

Profile name: Course Generator   Export...

| Indentation | Braces | Parentheses | White Space | Blank Lines | New Lines | Control Statements | Line Wrapping | Comments | Off/On Tags |

Off/On tags can be used in any comments to turn the formatter off and on in a source file.
- At the beginning of each file, the formatter is enabled.
- Each time the formatter sees an off tag, it disables formatting for that comment and the source after it.
- Each time the formatter sees an on tag, it enables formatting for the source after that comment.

☐ Enable Off/On tags

Off tag: @formatter:off

On tag: @formatter:on

Apply   Cancel   OK