

## CS-259

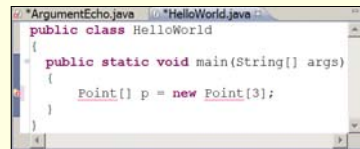
### Data Structures with Java

*CS-259 Code Standards and Setting  
Code Styles in Eclipse*

Instructor: **Joel Castellanos**

e-mail: [joel@unm.edu](mailto:joel@unm.edu)

web: <http://cs.unm.edu/~joel/>



```
public class HelloWorld
{
    public static void main(String[] args)
    {
        Point[] p = new Point(3);
    }
}
```

Course website:

<http://cs.unm.edu/~joel/cs259/>

9/30/2009

## CS-259 Coding Standards

- All *project* and *Labs* must follow the great and hallowed CS-259 coding standards.
- These standards do not necessarily represent the best nor the only good way to write Java code.
- If you have experience programming, then these standards may not be the standards you are used to using.
- However, in this class, these are the standards we will use.

## Primary Reasons for Defined Standard

---

1. A standard makes it easier for the instructors to read your code.
2. A class standard makes it easier for a grader to recognize when a program does not use a *consistent* standard.  
Often when each student is allowed to define his or her own standard, students switch standards multiple times in a single project. It is tedious for a grader to deduce each person's standard and then check for self-consistency.
3. It is good practice to learn to follow a standard.

3

## Coding Standard: Naming

---

- All variable names (fields) shall begin with a lower case letter.
- All instance variables will be given descriptive names.
- All methods will be given descriptive names.
- All class names shall begin with an uppercase letter.
- All variables declared with `final` shall be all uppercase.

4

## Coding Standard: Comments

- Each Java Source file shall begin with a heading in the form:

```
/**
 *@version 1.2 Aug, 24 2009
 *@author Ender Wiggin
 */
```

- Each method shall begin with a comment explaining its inputs, what it does, how it does it, and its return value.

5

## Coding Standard – Open Brackets

Open brackets will be placed at the beginning of a line (not at the end).

ok	<pre>if (x == 5) { y = y+1; }</pre>
Not CS-259 standard	<pre>if (x == 5) {     y = y+1; }</pre>

6

## Coding Standard – Closing Brackets

Closing brackets will be indented on a line with no other commands. The only exception being comments placed on the line with a closing bracket.

```
if (x == 5)
{ y=y+1;
} //Comment here ok
else if (x == 7)
{ y=y+2;
}
```

```
if (x == 5)
{ y=y+1;
} else if (x == 7)
{ y=y+2;
}
```

Bad

7

## Coding Standard – Blocks and { }

- Whenever a structure spans more than one line, brackets must be used. For example:

ok	<pre>if (x == 5) y=y+1;</pre>
ok	<pre>if (x == 5) { y=y+1; }</pre>
Not CS-259 standard	<pre>if (x == 5) y=y+1;</pre>

8

## Coding Standard - Indenting

- Code blocks will be indented to show the block structure with **two spaces** per level.
- Tab characters shall **not** be used for indenting.
- All statements within a block must be indented to the same level.


9

## Coding Standard – 80 Character Line Max

No line shall be more than 80 characters.

The best way to avoid overly long statements is by not doing too much in a single statement.

```
1 if (getVolume(length1, width1, height1) >
  getVolume(length2, width2, height2)) System.out.println
  ("box 1 is bigger"); else System.out.println ("box 2 is
  bigger");
2 int volume1 = getVolume(length1, width1, height1);
3 int volume2 = getVolume(length2, width2, height2);
4 if (volume1 > volume2)
5 { System.out.println("box 1 is bigger");
6 }
7 else
8 { System.out.println("box 2 is bigger");
9 }
```



10

## Fixing Too Long a Line Example 2

- Another case where a temporary variable can shorten a line and improve readability.
- Creating the temporary variable `c` also improves code maintenance:

If the code changes so that the comparison needs to check `stack[topOfStack]` or `stack[topOfStack-2]`, then Line 2 and 3 require only a single change while line 1 requires 4 changes.

```
1 if (stack[topOfStack - 1] == '*' ||
   stack[topOfStack - 1] == '+' || stack[topOfStack -
   1] == '-' || stack[topOfStack - 1] == '/')
2 char c = stack[topOfStack - 1];
3 if (c == '*' || c == '+' || c == '-' || c == '/')
```

11

## Fixing Too Long a Line Example 3

- There are times when breaking a long statement in to multiple statements is more awkward than keeping the long statement.
- In such cases, the statement should be broken in a *logical place* and each line over which the long statement is continued must be indented.
- The indenting must be *at least 2 spaces*, but can be more spaces it that improves readability. Code example 8, indents line 3 so that the comparisons match up.

```
1 if (commandOption == 'f' || commandOption == 'c' ||
   commandOption == 'd' || commandOption == 'g')
2 if (commandOption == 'f' || commandOption == 'c' ||
3   commandOption == 'd' || commandOption == 'g')
```

12

## Quiz 1-4: Coding Standard

Which line does NOT follow the standard?

```
a: for (i=0; i<10; i++)
b: { int c = i*10;
c:   if (c == 30)
d:     c=c+6;

e:   else if (c == 40) c = c-6;
      }
```

13

## Coding Standard: Class Layout

```
import javax.swing.JFrame;

public class GUI_Frame extends JFrame
{ //1st: All class variables
  private static final long serialVersionUID = 1L;
  private GUI_DrawPanel drawPanel;

  //2nd: Constructor(s) - if they exist.
  public GUI_Frame() ☒

  //3rd: Other methods in any order - if they exist.
  public void clear() ☒
  public void drawLines(int r, int g, int b) ☒

  //last: main() - if it exists.
  public static void main(String[] args) ☒
}
```

14

## Coding Standard: Minimal Scope

**Variables** used in only one method shall be local variables.

**Variables** used in more than one method within a class shall be private class variables.

A **data class** (a class with no methods) is the only type of class that may have public variables.

**Constants**, even when used in only one method, may be public static final or private static final.

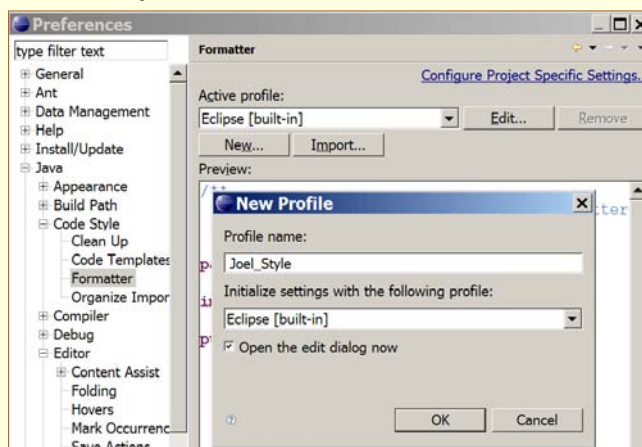
Do not hurt where holding is enough;  
do not wound where hurting is enough;  
do not maim where wounding is enough;  
and kill not where maiming is enough;  
the greatest warrior is one who does not need to kill.  
--Berek's Code, The Illearth War By Stephen R. Donaldson

15

## Setting Code Style in Eclipse

Window → Preferences

- Java → Code Style → Formatter → New...



16

## Code Style → Formatter → Indentation

The screenshot shows the 'Profile 'Joel\_Style'' dialog box with the 'Indentation' tab selected. The 'Profile name' field contains 'Joel\_Style'. The 'General settings' section includes a 'Tab policy' dropdown set to 'Spaces only', an unchecked checkbox for 'Use tabs only for leading indentations', 'Indentation size' and 'Tab size' text boxes both containing the value '2', and an unchecked checkbox for 'Align fields in columns'.

Profile name: Joel\_Style

Indentation Braces White Space Blank Lines New Lines Control Statements

General settings

Tab policy: Spaces only

Use tabs only for leading indentations

Indentation size: 2

Tab size: 2

Alignment of fields in class declarations

Align fields in columns

17

## Code Style → Formatter → Indentation

↑ Top part shown on last slide.

The screenshot shows the 'Indent' section of the dialog box, which contains a list of options with checkboxes. All options are checked except for 'Empty lines'.

Indent

- Declarations within class body
- Declarations within enum declaration
- Declarations within enum constants
- Declarations within annotation declaration
- Statements within method/constructor body
- Statements within blocks
- Statements within 'switch' body
- Statements within 'case' body
- 'break' statements
- Empty lines

18

## Code Style → Formatter → Braces

Profile 'Joel\_Style'

Profile name: Joel\_Style

Indentation | Braces | White Space | Blank Lines | New Lines | Control Statements

Brace positions

Class or interface declaration:	Next line
Anonymous class declaration:	Next line
Constructor declaration:	Next line
Method declaration:	Next line
Enum declaration:	Next line
Enum constant body:	Next line
Annotation type declaration:	Next line
Blocks:	Next line
Blocks in case statement:	Next line
'switch' statement:	Next line
Array initializer:	Next line

Keep empty array initializer on one line

19

## Code Style → Formatter → Control Statements

Profile 'Joel\_Style'

Profile name: Joel\_Style

Indentation | Braces | White Space | Blank Lines | New Lines | Control Statements

General

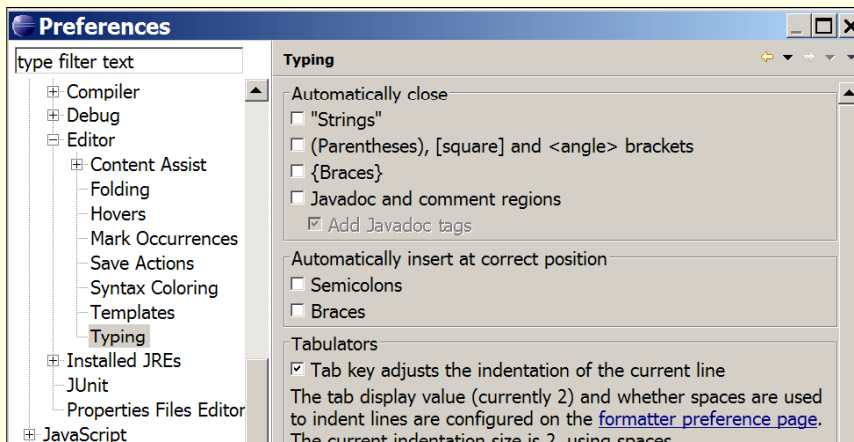
- Insert new line before 'else' in an 'if' statement
- Insert new line before 'catch' in a 'try' statement
- Insert new line before 'finally' in a 'try' statement
- Insert new line before 'while' in a 'do' statement

'if else'

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

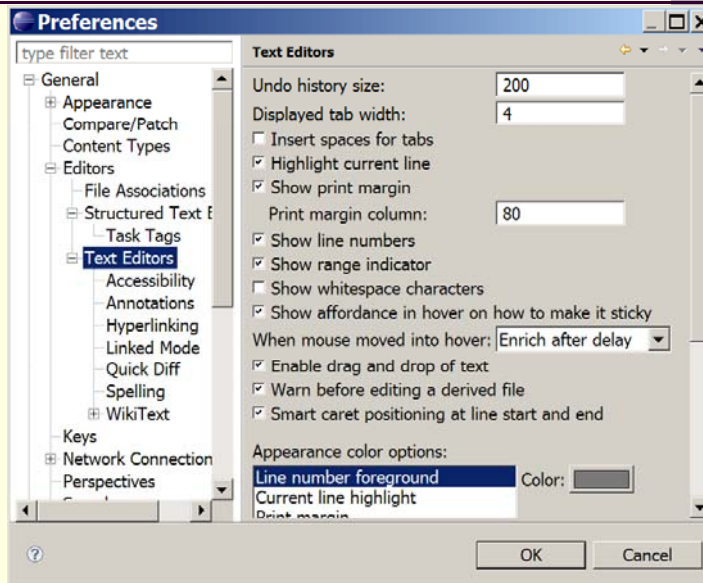
20

## As You type: Automatically Close...



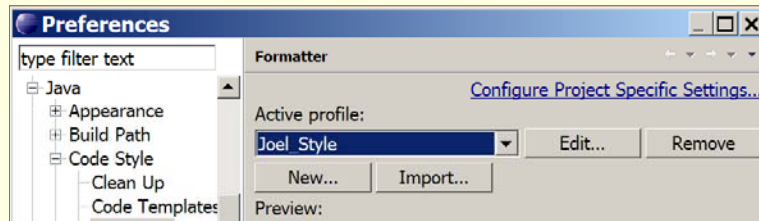
21

## Show 80 Column Print Margin



22

## Applying Style



- Click *OK* to close dialog.
- Then, *Right Click* in source code, and select:  
*Source* → *Format*.

23

## Quiz 1-5: Coding Standard

Which line does NOT follow the standard?

```
for (i=0; i<10; i++)
{ char c = inStr[i];
  if (c == '+') c=a+b;
  else if (c == '*') c = a*b;
a:  else if (c>='0' && c<='9')
b:  { for (j=0; j<c; j++)
c:    { System.out.print("j="+ j);
d:    }
e:  System.out.print("\n");
}
```

24