

# Chapter 4.2 – Coding Conventions

Walter F. Tichy



# Why Coding Conventions?

- A **consistent style** improves readability,
- speeds up **familiarization** if personnel changes, or when re-acquainting with old software
- **saves time** when searching for bugs, extending and maintaining software,
- “A clean style is matter of honor”
- must be applied consistently:
  - Adherence to conventions should be checked during program reviews and inspections.
  - or use Checkstyle
- Careful: Don't overdo it! Apply rules only where sensible.

# Which conventions should I use?

- K&R (Kernighan & Ritchie) Style, or “Kernel style” (Unix Kernel)

```
if (<condition>) {  
    <body>  
}
```

- “Whitesmiths Style”

```
if (<Condition>)  
{  
    <body>  
}
```

- “Allman Style” (Eric Allman), or “BSD Style”

```
if (<condition>)  
{  
    <body>  
}
```

- “GNU Style”, (GNU Emacs)

```
if (<condition>)  
{  
    <body>  
}
```

Answer: pick one and stick to it.

Or better: use the one your organization (everyone else) is using.

# Tool support

- Programming environments have a built-in code formatter
  - Eclipse
  - IntelliJ
  - Visual Studio
  - Emacs
- These can be adjusted to one's needs

# Example: Eclipse-Formatter

Here you can select predefined styles and thus easily switch between them. The key combination <Shift>+<Ctrl>+<F> then formats the current document with the selected style.

Styles are stored in XML files and can be exchanged that way.

You can even define your own style

Preferences

type filter text

- General
- Ant
- Checkstyle
- Help
- Install/Update
- Java
  - Appearance
  - Build Path
  - Code Style
  - Clean Up
  - Code Templates
  - Formatter**
  - Organize Imports
- Compiler
- Debug
- Editor
- Installed JREs
- JUnit
- Properties Files Editor
- Run/Debug
- Tasks
- Team
- Usage Data Collector
- Validation
- XML

**Formatter**

Active profile: Java Conventions [built-in] Edit... Remove

New... Import...

Preview:

```
/**
 * A sample source file for the Code Formatter preview
 */

package mypackage;

import java.util.LinkedList;

public class MyIntStack {
    private final LinkedList fStack;

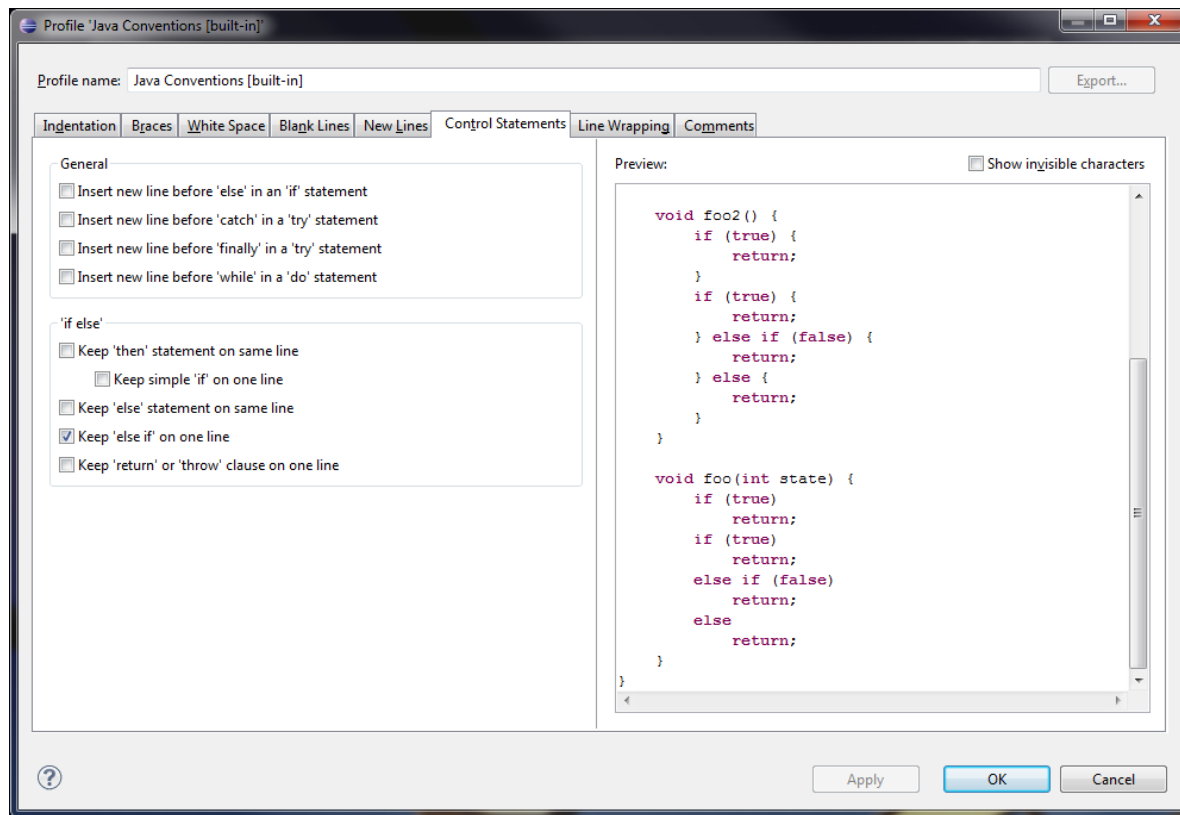
    public MyIntStack() {
        fStack = new LinkedList();
    }

    public int pop() {
        return ((Integer) fStack.removeFirst()).intValue();
    }
}
```

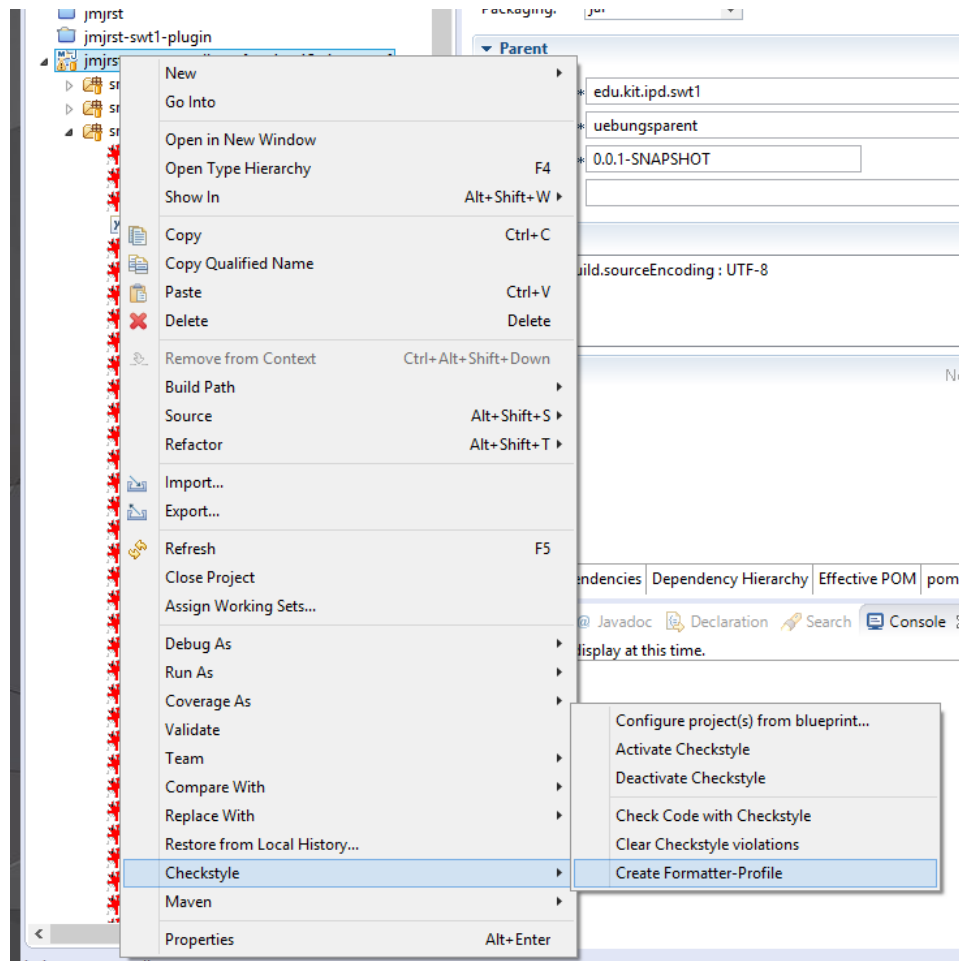
Restore Defaults Apply

OK Cancel

# Example Eclipse-Formatter: Settings



# Example Eclipse

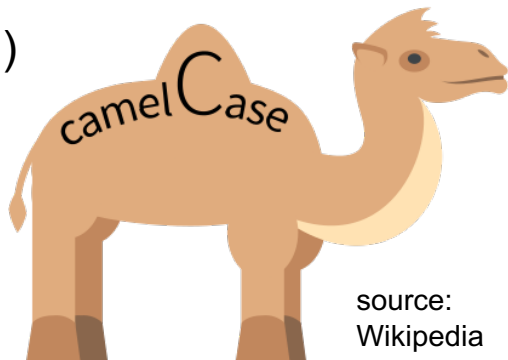


# Naming Conventions

There are conventions for

- class identifiers (noun or noun phrase in singular, starting with upper case letter),
- method identifiers (verb or verb phrase, starting with lower case letter)
- attributes, variables (lower case)
- constant identifiers and enumerations sometimes all capitalized (RED,BLUE, GREEN, const PI=3.14159)
- types sometimes with capital T at the end (ColorT)
- Compound identifiers:
  - words separated by underbar (Widget\_Company)
  - or capitalization (Camel case, or inner majuscule, as in softwareEngineering, eBay, johnSmith, WidgetCompany),
- acronyms are capitalized (IBMComputers or IBMcomputers?)

Stick to the conventions!



source:  
Wikipedia