



# Programmierstil & Coding Conventions

**Michael Inden**

**Freiberuflicher Consultant, Buchautor und Trainer**

---

# Coding Guidelines – Agenda

---



- **Initiale Gedanken**
  - **Schlechte Angewohnheiten und Abhilfen & Refactoring-Beispiel**
  - **Coding Guidelines**
  - **Tooling**
-



# Initiale Gedanken



## DON'T DO THAT

- „Schreibe deinen Code, [...] überlasse die Fehler der Qualitätskontrolle oder Testabteilung.“
  - „In der schlechten alten Zeit bauten Programmierer jede Anwendung von Grund auf neu.“
  - „Ändern Sie einfach so lange, bis es funktioniert.“
-



## DO THAT

- „**Gute Programmierer sind immer auch ein wenig faul: Sie lehnen sich lieber zurück und warten auf eine Erkenntnis, als mit ihrer ersten Idee gleich loszulegen.**“
  - „Warum schreiben Programmierer große Programme, wo auch kleine ausreichen? Ein Grund dafür ist, dass ihnen die [...] wichtige Faulheit fehlt; sie legen direkt los und kodieren ihre erste Idee.“
-



### DO THAT

- „Zu viele Programmierer stürzen sich zu schnell auf >>DIE<< Lösung ihres Problems. Sie denken eine Minute lang nach und schreiben einen Tag lang Code, anstatt eine Stunde lang nachzudenken und dann auch nur eine Stunde lang zu kodieren.“
  - „Wenn eine vom System bereitgestellte [...] Funktion Ihren Bedürfnissen genügt, sollten Sie noch nicht einmal daran denken eine eigene zu schreiben.“
  - „Im Stadium des Entwurfsprozesses ist es unschätzbar, die einschlägige Literatur zu kennen.“
-



## DO THAT

- „Der Zweck des Softwareengineering ist, Kompliziertheit zu kontrollieren und nicht, sie zu erzeugen.“
  - „Eine richtige Sicht der Daten strukturiert tatsächlich die Programme.“
  - „Immer für einfachen Code zu sorgen, ist normalerweise der Schlüssel zur Korrektheit!“
  - „Das wichtigste Prinzip beim Code-Tuning ist: Tun Sie es selten.“
-



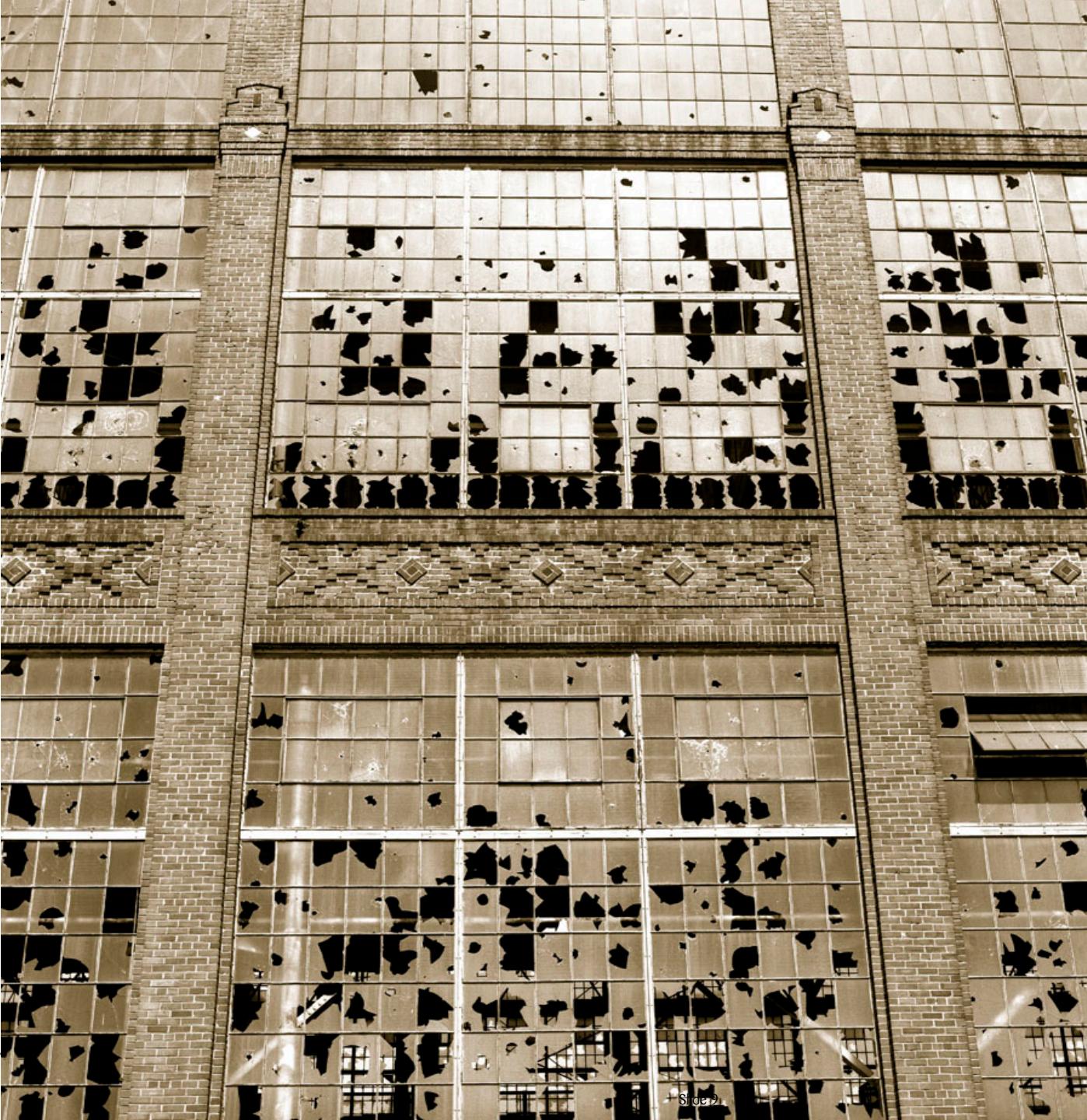
---

# Schlechte Angewohnheiten und Abhilfen & Refactoring-Beispiel



# Broken Window Phenomena

---



# Hotelroom Rule: Somebody is cleaning up for you



# Continuous REFACTORING



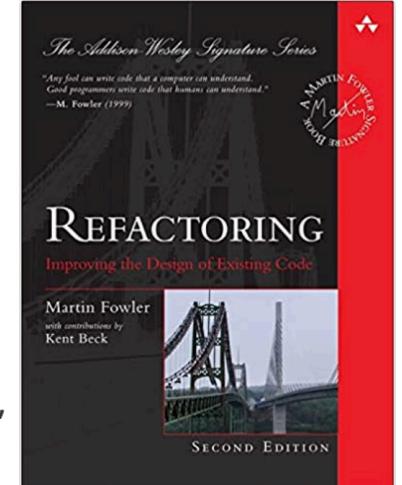
## Definition

In the book, I make the following definition of "refactoring"

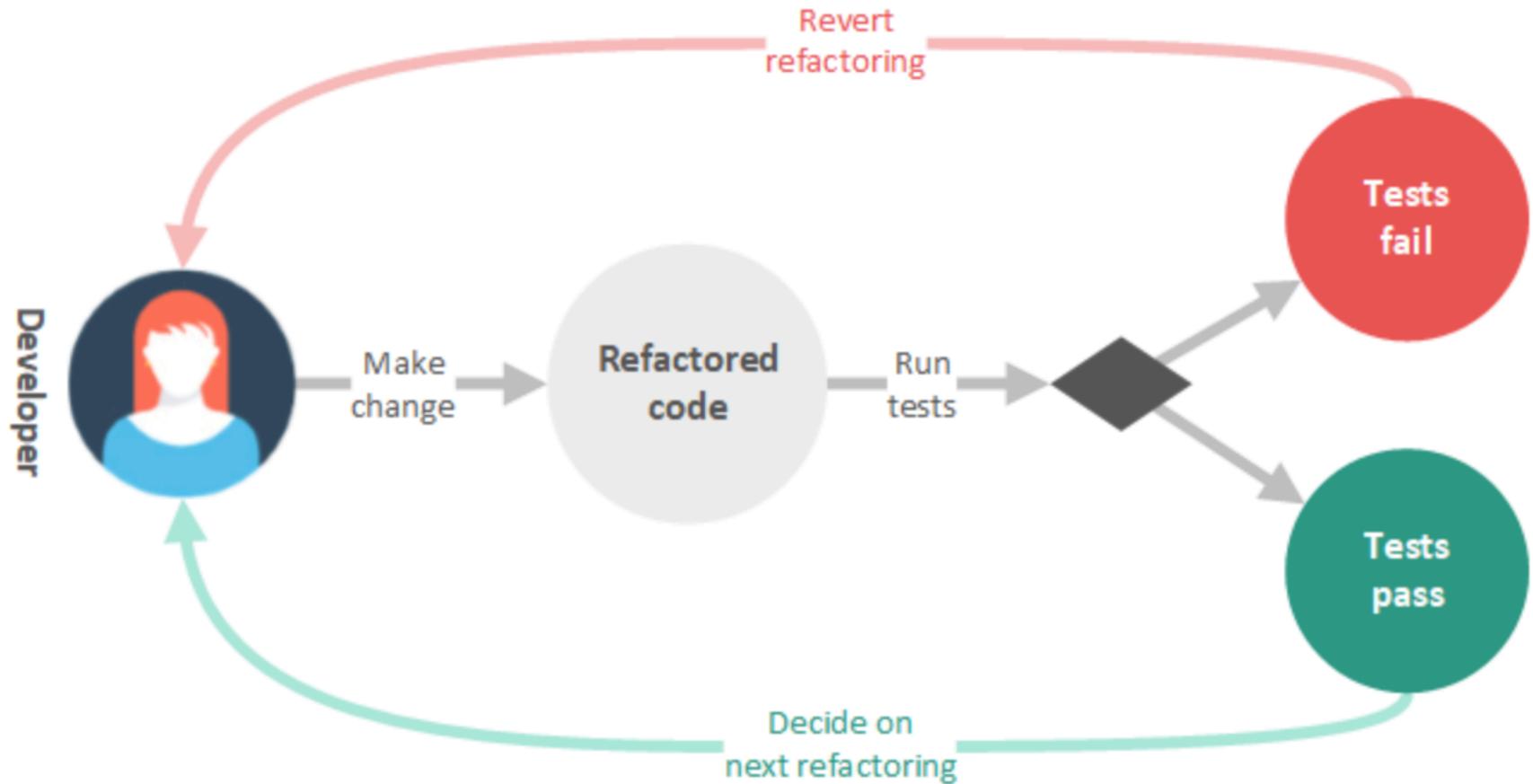
*noun: a change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior*

*verb: to restructure software by applying a series of refactorings without changing its observable behavior.*

Refactoring isn't another word for cleaning up code - it specifically defines *one* technique for improving the health of a code-base. I use "restructuring" as a more general term for reorganizing code that may incorporate other techniques.



# Refactoring in Practice



Source: <https://dzone.com/articles/what-is-refactoring>

# Refactoring in Practice – Ausgangsbasis

---



```
private static void printLetterOrig()
{
    printBanner();

    // print content
    for (int i = 0; i < content.size(); i++)
    {
        ContentLine cl = content.get(i);
        printContentline(cl);
    }

    // add happy christmas if in december and before / on christmas eve
    if (LocalDate.now().getMonth() == Month.DECEMBER &&
        LocalDate.now().getDayOfMonth() < 25)
    {
        printHappyChristmas();
    }

    printFooter();
}
```

# Refactoring in Practice – Beispiel EXTRACT METHOD

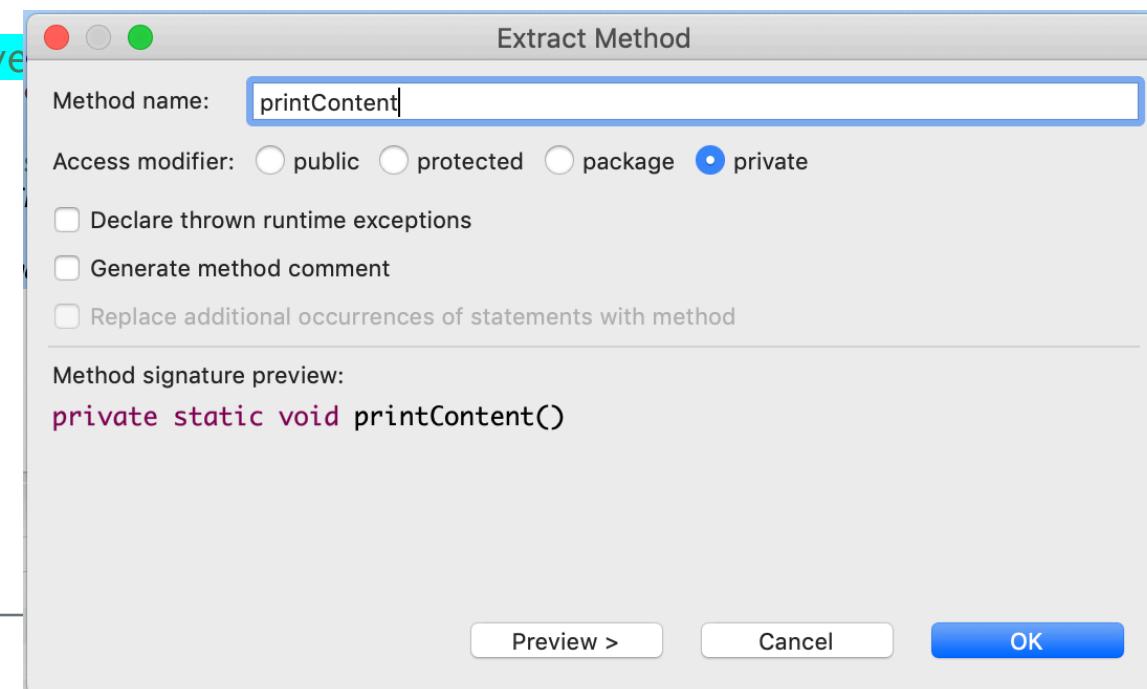


```
private static void printLetterOrig()
{
    printBanner();

    // print content
    for (int i = 0; i < content.size(); i++)
    {
        ContentLine cl = content.get(i);
        printContentline(cl);
    }

    // add happy christmas if ... before / on christmas eve
    if (LocalDate.now().getMonth() == Month.DECEMBER &&
        LocalDate.now().getDayOfMonth() < 25)
    {
        printHappyChristmas();
    }

    printFooter();
}
```



# Refactoring in Practice – Beispiel EXTRACT METHOD



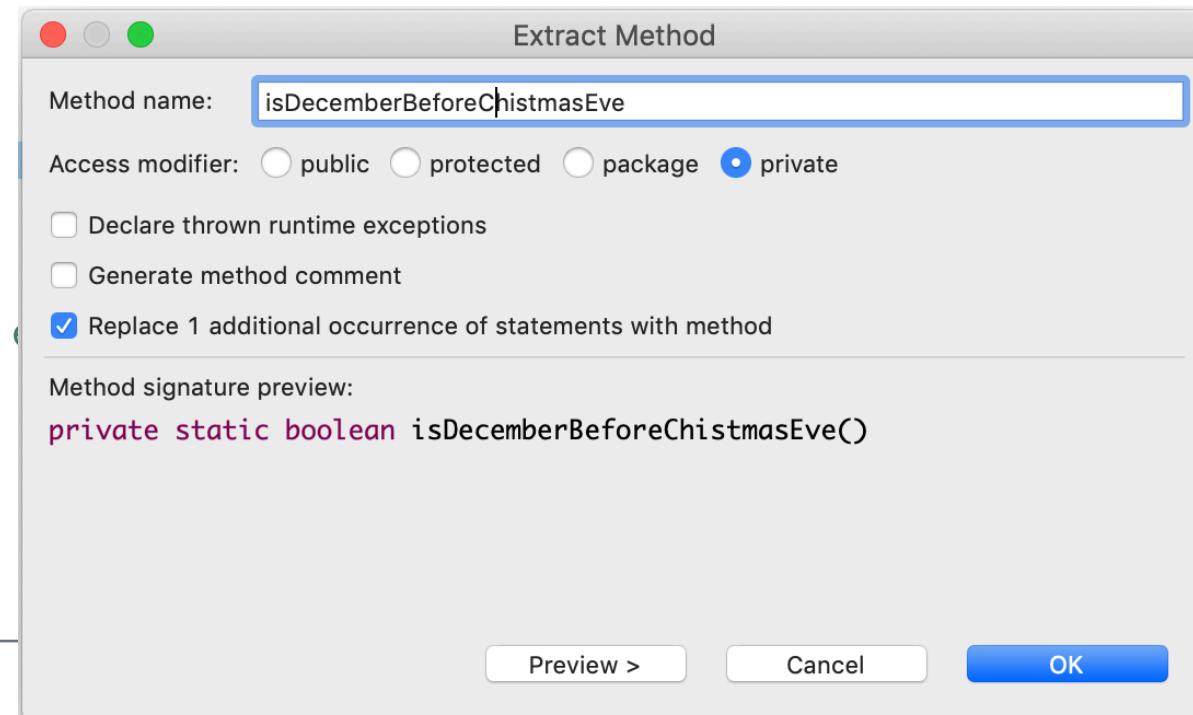
```
private static void printLetterImproved_Step1()
{
    printBanner();

    printContent();

    printFooter();
}

private static void printContent()
{
    for (int i = 0; i < content.size(); i++)
    {
        ContentLine cl = content.get(i);
        printContentline(cl);
    }

    // add happy christmas if ... before / on christmas
    if (LocalDate.now().getMonth() == Month.DECEMBER &&
        LocalDate.now().getDayOfMonth() < 25)
    {
        printHappyChristmas();
    }
}
```



# Refactoring in Practice – Beispiel EXTRACT LOCAL VARIABLE



```
private static void printLetterImproved_Step2()
{
    printBanner();
    printContent();
    printFooter();
}

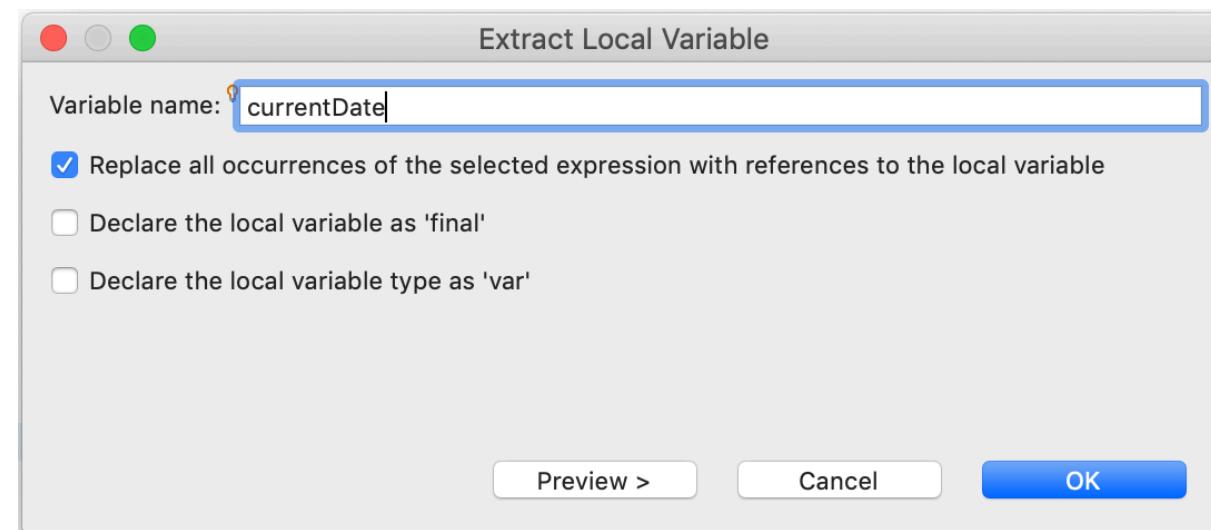
private static void printContent()
{
    for (int i = 0; i < content.size(); i++)
    {
        ContentLine cl = content.get(i);
        printContentline(cl);
    }

    if (inDecemberBeforeChristmasEve())
    {
        printHappyChristmas();
    }
}

private static boolean inDecemberBeforeChristmasEve()
{
    return LocalDate.now().getMonth() == Month.DECEMBER && LocalDate.now().getDayOfMonth() < 25;
}
```



**Short Cut:  
Introduce Parameter**



# Refactoring in Practice – Beispiel INTRODUCE PARAMETER



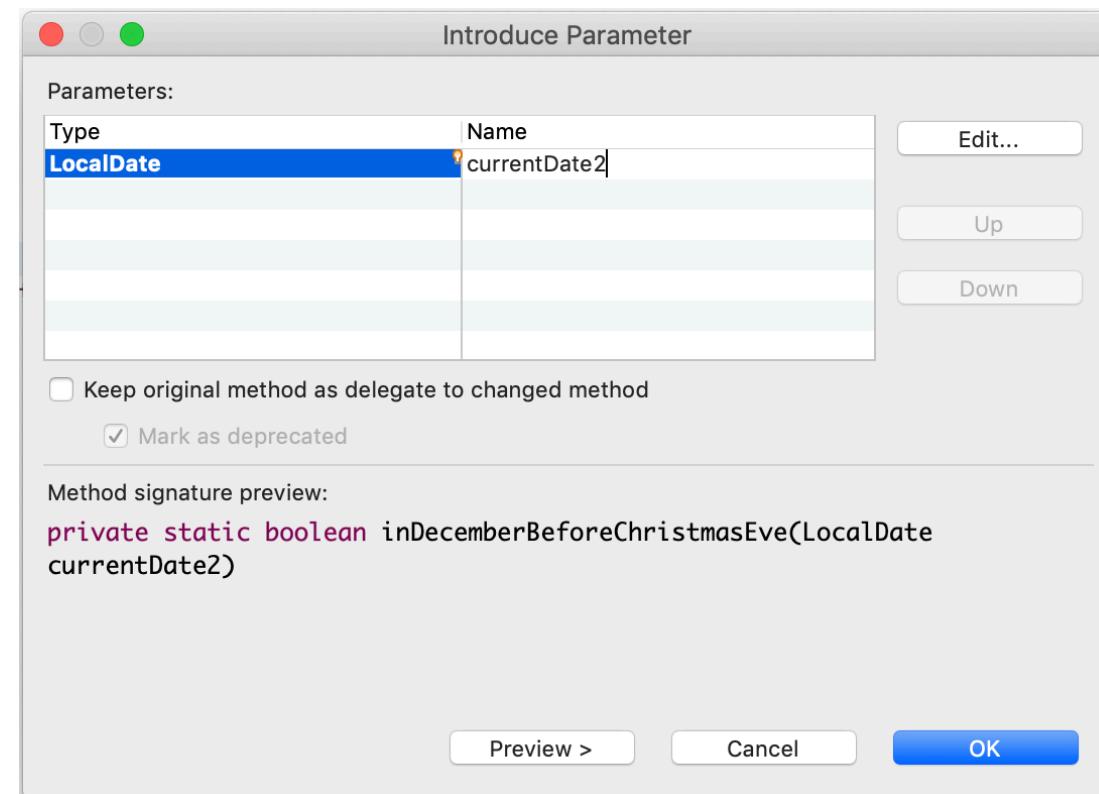
```
private static void printLetterImproved_Step3()
{
    printBanner();
    printContent();

    printFooter();
}

private static void printContent()
{
    for (int i = 0; i < content.size(); i++)
    {
        ContentLine cl = content.get(i);
        printContentline(cl);
    }

    if (inDecemberBeforeChristmasEve())
    {
        printHappyChristmas();
    }
}

private static boolean inDecemberBeforeChristmasEve()
{
    LocalDate currentDate = LocalDate.now();
    return currentDate.getMonth() == Month.DECEMBER && currentDate.getDayOfMonth() < 25;
}
```



# Refactoring in Practice – Beispiel INLINE LOCAL VARIABLE



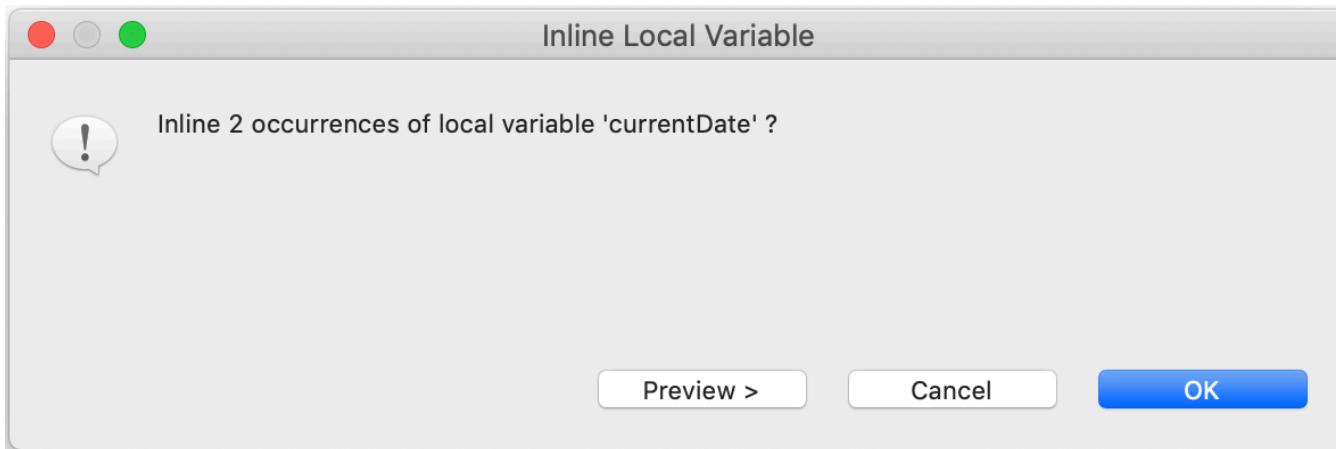
```
private static void printLetterImproved_Step4()
{
    printBanner();
    printContent();

    printFooter();
}

private static void printContent()
{
    for (int i = 0; i < content.size(); i++)
    {
        ContentLine cl = content.get(i);
        printContentline(cl);
    }

    if (inDecemberBeforeChristmasEve(LocalDate.now()))
    {
        printHappyChristmas();
    }
}

private static boolean inDecemberBeforeChristmasEve(LocalDate currentDate2)
{
    LocalDate currentDate = currentDate2;
    return currentDate.getMonth() == Month.DECEMBER && currentDate.getDayOfMonth() < 25;
}
```



## Refactoring in Practice – Beispiel RENAME



```
private static void printLetterImproved_Step5()
{
    printBanner();
    printContent();
    printFooter();
}

private static void printContent()
{
    for (int i = 0; i < content.size(); i++)
    {
        ContentLine cl = content.get(i);
        printContentline(cl);
    }

    if (inDecemberBeforeChristmasEve(LocalDate.now()))
    {
        printHappyChristmas();
    }
}

private static boolean inDecemberBeforeChristmasEve(LocalDate currentDate2)
{
    return currentDate2.getMonth() == Month.DECEMBER && currentDate2.getDayOfMonth() < 25;
```

# Refactoring in Practice – Beispiel FINISH ☺

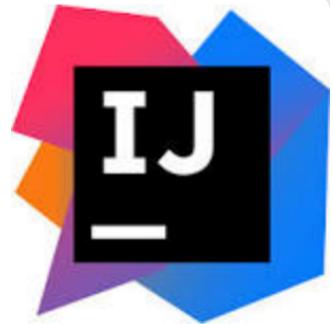


```
private static void printLetterImproved_Step6()
{
    printBanner();
    printContent();
    printFooter();
}

private static void printContent()
{
    for (int i = 0; i < content.size(); i++)
    {
        ContentLine cl = content.get(i);
        printContentline(cl);
    }

    if (inDecemberBeforeChristmasEve(LocalDate.now()))
    {
        printHappyChristmas();
    }
}

private static boolean inDecemberBeforeChristmasEve(LocalDate currentDate)
{
    return currentDate.getMonth() == Month.DECEMBER && currentDate.getDayOfMonth() < 25;
```



Besser in Refactorings:  
Wie erwähnt, kann es die  
letzten Schritte kombinieren

**Boyscout Rule: Leave the campground cleaner than you found it**

---





# Coding Guidelines





**«Jeder Dummkopf kann Code schreiben, der für Computer verständlich ist.»**

**«Gute Programmierer schreiben Code, den Menschen verstehen können.»**

**- Martin Fowler -**

---



**We READ 10x more time than we WRITE**

→ Make it more readable, even if it's harder to write

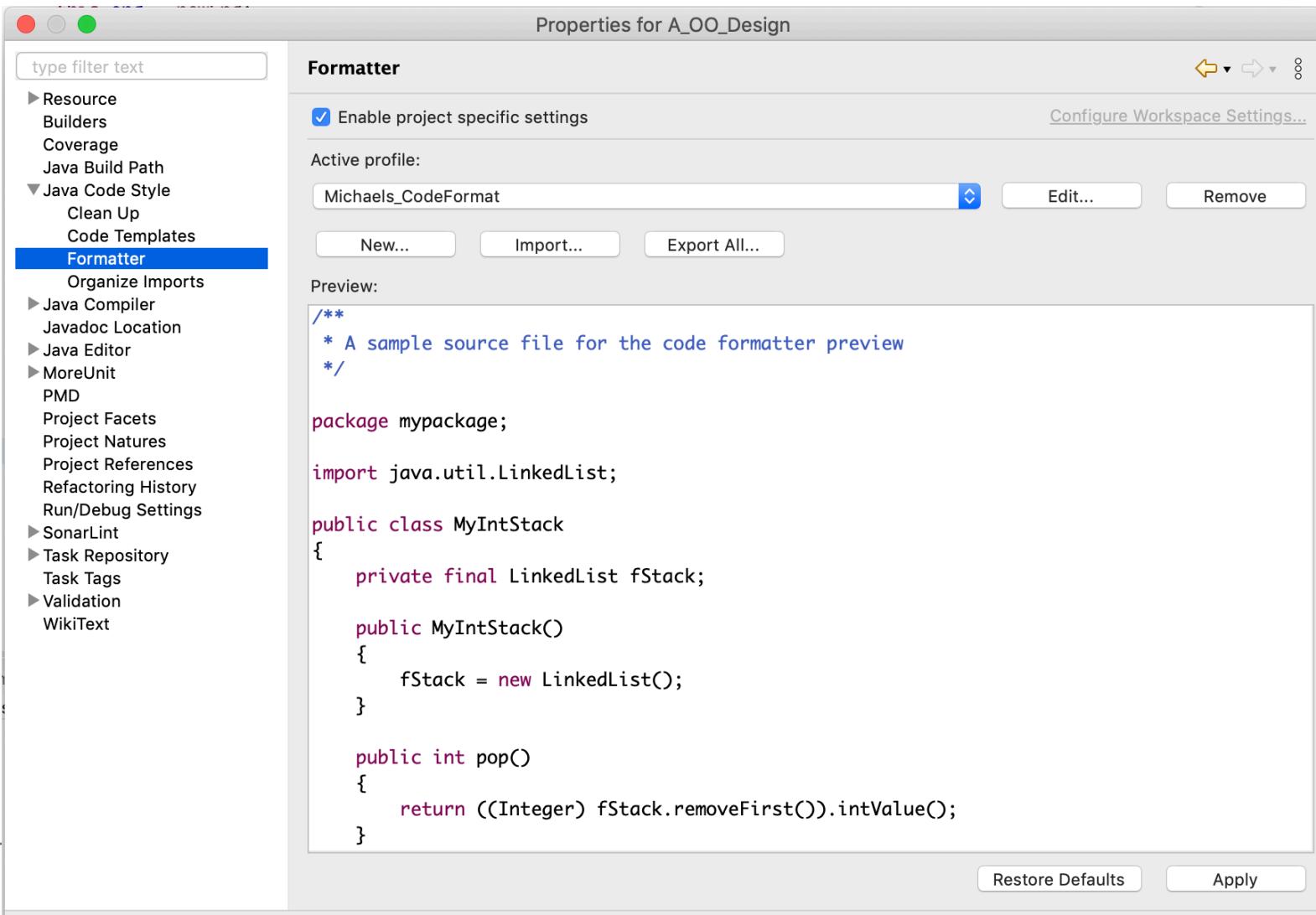


- Es hat es sich immer wieder als Vorteil erwiesen, gewisse Konventionen beim Programmieren einzuhalten.
  - Der Programmierstil und auch das Layout des Sourcecodes haben großen Einfluss auf dessen **Lesbarkeit und Verständlichkeit und spätere Wartbarkeit**
  - **Formatting Conventions** – Wie gestalte ich das **Layout**?
  - **Naming Conventions** – Wie finde ich (sinnvolle), einheitliche **Namen**?
  - **Coding Conventions** – Wie mache ich **Code lesbar**? Wie vermeide ich einfache **Fehler**?
  - **GENERELLES ZIEL:** So viel wie möglich automatisiert durch **Tools erledigen / prüfen lassen**
-

# Formatting Conventions



- **Definition eines Code Formatters**



# Naming Conventions

---



- **Wie finde ich (sinnvolle), einheitliche Namen?**
- **Basics:**

Typ	Präfix	Postfix	Beispiel
Packages	-	-	mypackage.subpackage
Interfaces	I / -	IF / -	IFileInfo, Filterable
Abstrakte Klassen	Base, Abstract	-	AbstractProcessStep
Klassen	-	-	JobAssistant
Testklassen	-	Test	ProcessStepTest
Methoden	-	-	getFileName()
Konstanten	-	-	MAX_VALUE
Lokale Variablen	-	-	imageWidth
Membervariablen	, m_, this.	-	name, m_name, this.name

# Naming Conventions

---



- **Wie finde ich (sinnvolle), einheitliche Namen?**

- **Vermeide Namenskürzel:** `cnt`, `ser`, `ikm`
  - Ausnahmen: `Html`, `Xml`, `Json`, `it`, `sb`, ...
- Vermeide Variablennamen, die nur den Typ wiederholen

```
final Vector<File> vector = new Vector<>();  
  
final List<File> list = new ArrayList<>();
```

- Benenne Klassen / Methoden nach der bereitgestellten Funktionalität
- `*And*`, `*Or*`, `*_*` sind Indikatoren für zu viel Funktionalität => EXTRACT METHOD  
`saveAndPrint()`, `formatAndSendMail()`, `copyOrMove()`, `select_mark_calc_avg()`

# Naming Conventions

---



- **Wie finde ich (sinnvolle), einheitliche Namen?**
- **Weitere Inspirationen:**
  - Kapitel Coding Conventions aus dem Java-Profi
  - <http://www.ambisoft.com/essays/javaCodingStandards.html>
  - [https://wiki.eclipse.org/Naming\\_Conventions](https://wiki.eclipse.org/Naming_Conventions)
  - <https://www.eclipse.org/jetty/documentation/current/coding-standards.html>
  - <https://www.torsten-horn.de/techdocs/java-codingconventions.htm>
  - <https://www.cs.cornell.edu/courses/JavaAndDS/JavaStyle.html>
  - <https://google.github.io/styleguide/javaguide.html>

# Coding Conventions Checklist – coding hints and rules



- Wie mache ich Code lesbar? Wie vermeide ich einfache Fehler?
- Use descriptive Comments but do NOT overcomment  
Document your public methods especially interfaces well

```
public interface IFilterManager
{
    ...
    /**
     * Method getAttributeForName.
     *
     * map a passed attribute name to a filter attribute instance
     *
     * @param strAttributeName
     * @return the stored filter attribute with the passed name or
     *         null if no such attribute exists
     */
    IFilterAttributeRO getAttributeForName ( final String strAttributeName );
    ...
}
```

# Coding Conventions Checklist – coding hints and rules



- Prefer constants instead of Magic Numbers and Strings
- Define constants in classes, not in interfaces

```
public final class MyConstants
{
    public static final String NORDEN = "Nord";
    public static final String SUEDEN = "Süd";

    public static final int GAP = 10;
    public static final int EOF = -1;
}
```

- **Code-Design**
  - Remove duplicate code, use function calls instead
  - Remove unused code
  - Get help from Eclipse: extract method, extract local variable, ...

# Coding Conventions

---



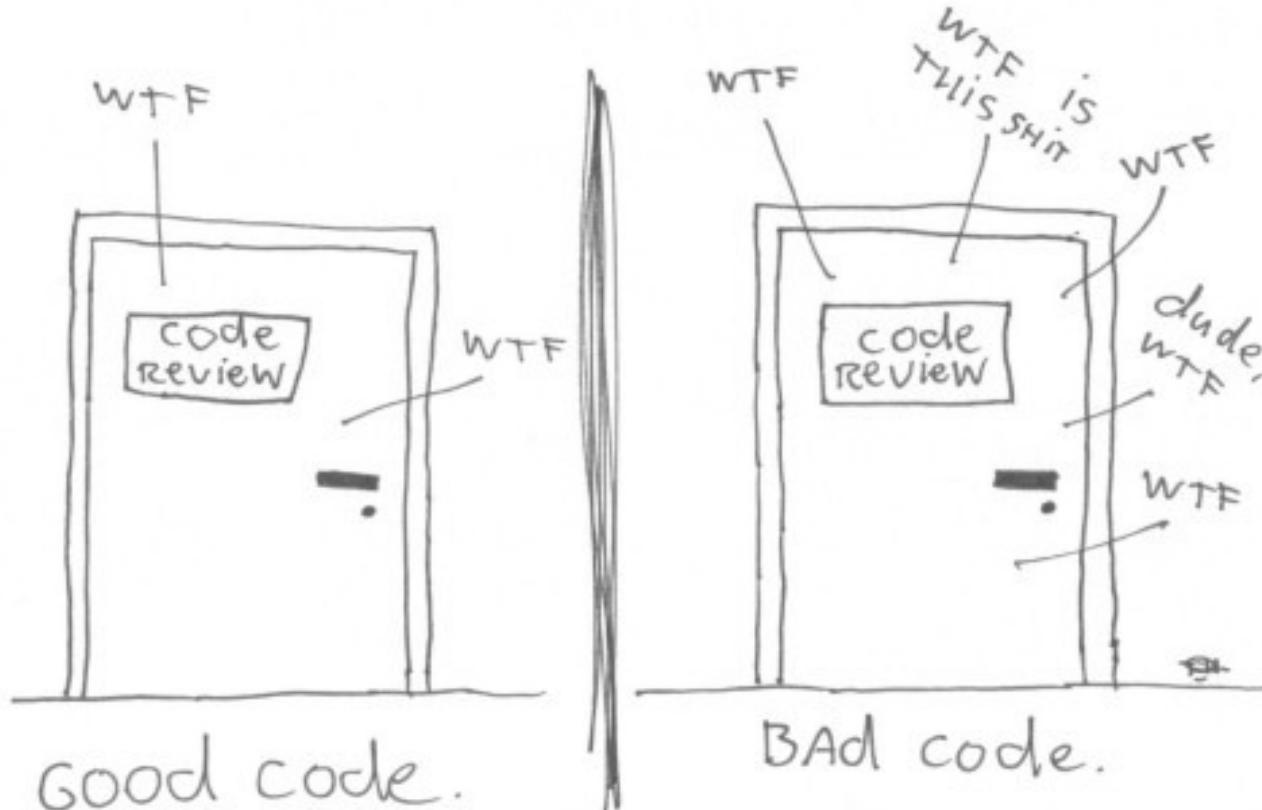
- **Wie mache ich Code lesbar? Wie vermeide ich einfache Fehler?**

- Vermeide Seiteneffekte
- Verstecke Implementierungsdetails
- Vermeide übermässige Debug- und Log-Ausgaben
- Bevorzuge wenige return-Anweisungen
- **Kapitel Coding Conventions aus dem Java-Profi**
- Eclipse Qualitätsprüfungen Built In
  - Vieles schon gut vorkonfiguriert, kleine Anpassungen sinnvoll

# Coding Guidelines Wissenswertes



The ONLY VALID measurement  
OF code QUALITY: WTFs/minute





# Tooling





- **Eclipse Konfiguration Errors/Warnings**
  - **Eclipse Save Actions**
  - **SonarLint**
  - **PMD**
  - **Checkstyle**
-

# Eclipse Konfiguration Errors/Warnings



Properties for AAAAAAA\_QualityCheckExamples

Errors/Warnings

Enable project specific settings [Configure Workspace Settings...](#)

Select the severity level for the following optional problems:

type filter text (use ~ to filter on preference values, e.g. ~ignore or ~off)

**Code style**

Non-static access to static member:	Warning
Indirect access to static member:	Ignore
Unqualified access to instance field:	Ignore
Access to a non-accessible member of an enclosing type:	Ignore
Parameter assignment:	Warning
Non-externalized strings (missing/unused \$NON-NLS\$ tag):	Ignore
Undocumented empty block:	Warning
Resource not managed via try-with-resource (1.7 or higher):	Ignore
Method with a constructor name:	Warning
Method can be static:	Info
Method can potentially be static:	Ignore

**Potential programming problems**

- ▶ Name shadowing and conflicts
- ▶ Deprecated and restricted API
- ▶ Modules
- ▶ Unnecessary code

# Eclipse Konfiguration Errors/Warnings



## Potential programming problems

Comparing identical values ('x == x'): Warning ▾

Assignment has no effect (e.g. 'x = x'): Warning ▾

Possible accidental boolean assignment (e.g. 'if (a = b)'): Ignore ▾

Boxing and unboxing conversions: Ignore ▾

Using a char array in string concatenation: Warning ▾

Inexact type match for vararg arguments: Warning ▾

Unlikely argument type for collection methods using 'Object' Warning ▾

Perform strict analysis against the expected type

Unlikely argument type for method equals(): Info ▾

Empty statement: Ignore ▾

Unused object allocation: Ignore ▾

Incomplete 'switch' cases on enum: Warning ▾

Signal even if 'default' case exists

'switch' is missing 'default' case: Ignore ▾

'switch' case fall-through: Warning ▾

Hidden catch block: Warning ▾

'finally' does not complete normally: Warning ▾

Dead code (e.g. 'if (false)'). Warning ▾

Resource leak: Warning ▾

Potential resource leak: Ignore ▾

Serializable class without serialVersionUID: Warning ▾



# Eclipse Konfiguration Errors/Warnings



▼ Unnecessary code

Value of local variable is not used:	Warning
Value of method parameter is not used:	Warning
<input checked="" type="checkbox"/> Ignore in overriding and implementing methods	
Unused type parameter:	Warning
<input checked="" type="checkbox"/> Ignore unused parameters documented with '@param' tag	
Value of exception parameter is not used:	Ignore
Unused import:	Warning
Unused private member:	Warning
Unnecessary 'else' statement:	Ignore
Unnecessary cast or 'instanceof' operation:	Ignore
Unnecessary declaration of thrown exception:	Ignore
<input checked="" type="checkbox"/> Ignore in overriding and implementing methods	
<input checked="" type="checkbox"/> Ignore exceptions documented with '@throws' or '@exception' tags	
<input checked="" type="checkbox"/> Ignore 'Exception' and 'Throwable'	
Unused 'break' or 'continue' label:	Warning
Redundant super interface:	Ignore

# Eclipse Konfiguration Errors/Warnings

---



## Description

### ▼ ⚠ Warnings (4 items)

💡 Null pointer access: The variable nully can only be null at this location

💡 Switch case may be entered by falling through previous case. If intended, add a new comment //\$/FALL-THROUGH\$ on the line above

💡 The parameter info should not be assigned

💡 The value of the parameter info is not used

# SAVE ACTIONS – Coding Guidelines Hilfestellung



Preferences

Save Actions

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

- Remove unused imports
- Remove unused private methods
- Remove unused private constructors
- Remove unused private types
- Remove unused private fields
- Remove unused local variables
- Add missing '@Override' annotations
- Add missing '@Override' annotations to implementations of interface methods
- Add missing '@Deprecated' annotations
- Remove unnecessary casts
- Remove redundant semicolons
- Remove unnecessary array creation for varargs
- Use Objects.equals() in the equals method implementation
- Remove redundant type arguments
- Use Autoboxing
- Use Unboxing
- Operate on Maps directly

Configure...

Configure Project Specific Settings...

type filter text

- ▶ Help
- ▶ Install/Update
- ▼ Java
  - ▶ Appearance
  - ▶ Build Path
  - Code Coverage
  - ▼ Code Style
    - Clean Up
    - Code Templates
    - Formatter
    - Organize Imports
  - ▶ Compiler
  - ▶ Debug
  - ▼ Editor
    - Code Minings
    - Content Assist
    - Folding
    - Hovers
    - Mark Occurrences
    - Save Actions
    - Syntax Coloring
    - Templates
    - Typing
  - ▶ Installed JREs
  - JUnit
  - Properties Files Editor
  - ▶ Java EE
  - ▶ Java Persistence
  - JSON
  - ▶ Language Servers
  - Maven
  - MoreUnit
  - Mylyn
  - Oomph
  - Pitest

# SAVE ACTIONS – Coding Guidelines Hilfestellung



Additional Save Actions

Code Organizing    Code Style    Member Accesses    Missing Code    Optimization    **Unnecessary Code**

Unused code

- Remove unused imports
- Remove unused private members
  - Types
  - Constructors
  - Fields
  - Methods
- Remove unused local variables

Unnecessary code

- Remove unnecessary casts
- Remove unnecessary '\$NON-NLS\$' tags
- Remove redundant type arguments (1.7 or higher)
- Use Autoboxing (1.5 or higher)
- Use Unboxing (1.5 or higher)
- Push down negation
- Merge conditions of if/else if/else that have the same blocks
- Operate on Maps directly
- Remove redundant modifiers
- Remove redundant semicolons
- Remove unnecessary array creation (1.5 or higher)
- Use Objects.equals() in the equals method implementation (1.7 or higher)

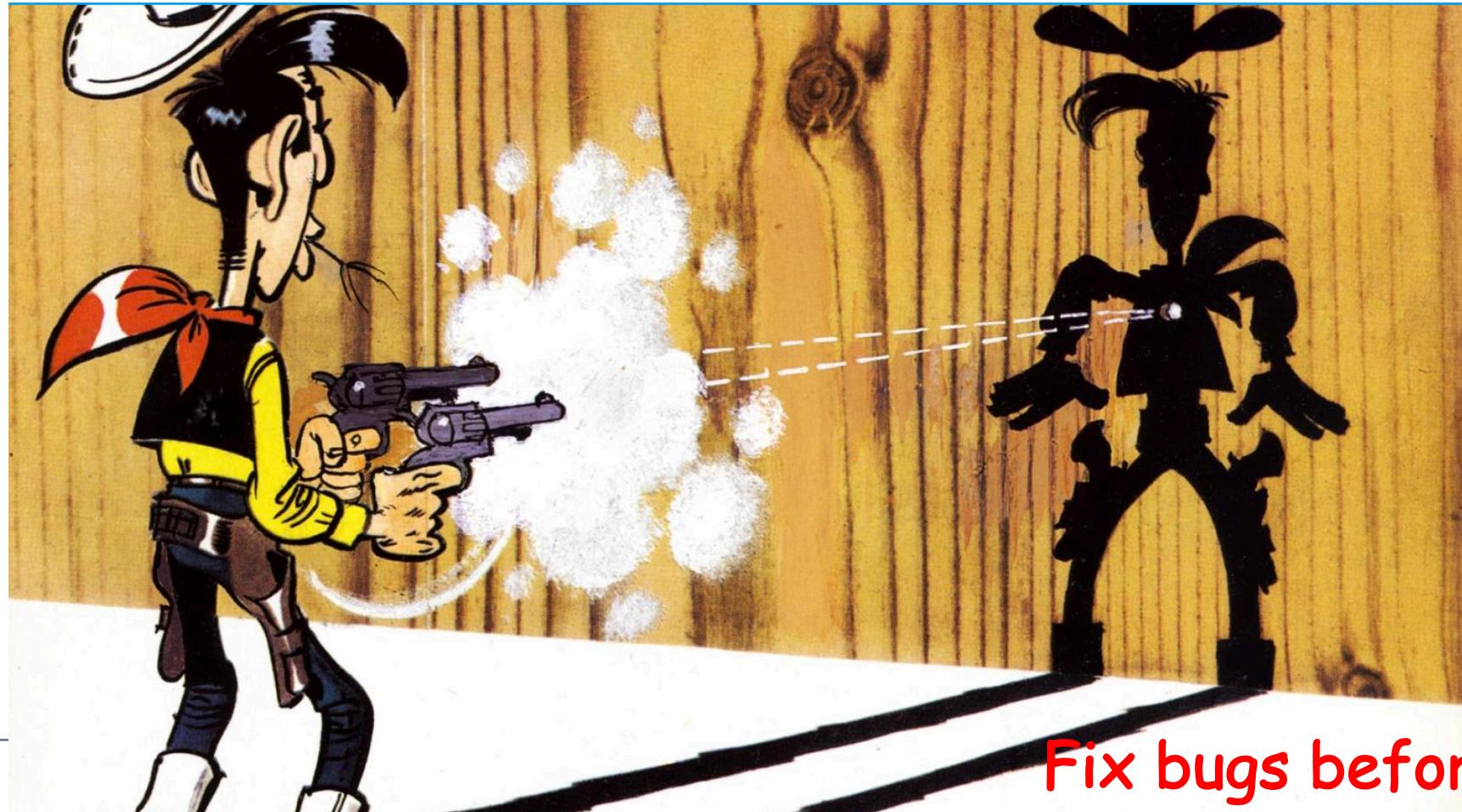
Preview:

```
class Example {  
    public Example(boolean b) {}  
    public void bar() {  
    }  
}  
  
Boolean b= Boolean.TRUE;  
  
public String s; //NON-NLS-1$  
  
Map<Integer, String> map= new HashMap<>();  
Integer i = 0;  
Character c = '*';  
  
Integer integerObject = Integer.MAX_VALUE;  
Character cObject = Character.MAX_VALUE;  
  
int i = integerObject;  
char c = cObject;  
  
boolean b = !(myInt > 0);  
boolean b2 = !(isEnabled || isValid);  
  
if (i == 0) {  
    System.out.println("Duplicate");  
} else if (i == 1) {  
    System.out.println("Duplicate");  
} else {  
    System.out.println("Different");  
}  
  
int x = map.size();  
if (map.containsKey("hello")) {  
    map.remove("hello");  
}
```



---

# SonarLint



Fix bugs before they occur



Eclipse Marketplace

Eclipse Marketplace

Select solutions to install. Press Install Now to proceed with installation.  
Press the "more info" link to learn more about a solution.

Search Recent Popular Favorites Installed Giving IoT an Edge

Find:  All Markets All Categories Go

**SonarLint 5.4.0**

SonarLint is an IDE extension that helps you detect and fix quality issues as you write code in Java, JavaScript, PHP, Python and HTML.  
[more info](#)

sonarlint by [SonarSource S.A.](#), LGPL  
java PHP javascript Python static analysis

204! Installs: **723K** (17'654 last month) Installed

**Marketplaces**

< Back Install Now > Cancel Finish



- **Als Ergänzung oder Stand Alone zu SonarQube Server**
  - **Statische Analyse des Sourcecodes (offline)**
  - **Eclipse Plugin liefert Feedback direkt beim Programmieren**
  - **Findet eine Menge an potenziellen Problemen**
  - **Fehlermeldungen sind (sehr) verständlich, gute Hintergrund-Dokumentation**
  - **Behebung während der aktiven Entwicklung verhindert, dass Bugs oder zumindest Code Smells bis in das Repository gelangen**
  - **Behebung verringert die manuellen Aufwände für Reviews**
  - **Hebt die Qualität in kleinen Schritten und ohne viel Aufwand (ganz einfach nebenbei)**
-



- **Findet eine Menge an potenziellen Problemen**
- **Fehlermeldungen sind (sehr) verständlich**

Violations Overview			
SonarLint Report		X	
99 items			
Resource	Date	Description	
BuilderPattern.java		💡 Rename this local variable to match the regular expression '^[a-z][a-zA-Z0-9]*\$'.	
BuilderPattern.java		💡 Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-zA-Z0-9_])*\$'.	
Circle.java		💡 Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-zA-Z0-9_])*\$'.	
Command.java	5 day...	⚠️ This block of commented-out lines of code should be removed.	
Command.java		💡 Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-zA-Z0-9_])*\$'.	
Command.java		💡 Use "switch" expression to return the value from method.	
CommandExample.java	5 day...	💡 Remove this unused "sub30" local variable.	
CommandExample.java	5 day...	⚠️ Remove this useless assignment to local variable "sub30".	
CommandExample.java		💡 Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-zA-Z0-9_])*\$'.	
ContentLine.java	2 hour...	⚠️ Remove this unused "line" private field.	
ContentLine.java		💡 Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-zA-Z0-9_])*\$'.	
Counter.java		⚠️ Add a nested comment explaining why this method is empty, throw an UnsupportedOperationException or similar.	
CounterV2.java		⚠️ Add a nested comment explaining why this method is empty, throw an UnsupportedOperationException or similar.	
DiscountCalculation.java	5 day...	💡 Remove this unused "easter2" local variable.	
DiscountCalculation.java	5 day...	💡 Remove this unused import 'java.util.function.UnaryOperator'.	
DiscountCalculation.java	5 day...	⚠️ Make this anonymous inner class a lambda	
DiscountCalculation.java	5 day...	⚠️ Remove this useless assignment to local variable "easter2".	
DiscountCalculation.java	5 day...	⚠️ This block of commented-out lines of code should be removed.	



- Fehlermeldungen sind (sehr) verständlich, gute Hintergrund-Dokumentation

A screenshot of the SonarLint interface. At the top, there are tabs: 'Violations Overview' (selected), 'SonarLint Report', and 'SonarLint Rule Description'. On the right side of the header are standard window controls (minimize, maximize, close). Below the tabs, the main content area shows a code smell report for 'Use Java 12 "switch" expression (java:S5194)'. It includes a 'Code smell' icon, a 'Minor' severity indicator, and a detailed description: 'Many existing switch statements are essentially simulations of switch expressions, where each arm either assigns to a common target variable or returns a value. Expressing this as a statement is roundabout, repetitive, and error-prone.' A note below states: 'Java 12 added support for switch expressions, which provide more succinct and less error-prone version of switch.'

## Use Java 12 "switch" expression (java:S5194)

Code smell Minor

Many existing switch statements are essentially simulations of switch expressions, where each arm either assigns to a common target variable or returns a value. Expressing this as a statement is roundabout, repetitive, and error-prone.

Java 12 added support for switch expressions, which provide more succinct and less error-prone version of switch.

### Noncompliant Code Example

```
void day_of_week(Dow day) {  
    int numLetters;  
    switch (day) { // Noncompliant  
        case MONDAY:  
        case FRIDAY:  
        case SUNDAY:  
            numLetters = 6;  
            break;  
        case TUESDAY:  
            numLetters = 7;  
            break;  
        case THURSDAY:  
        case SATURDAY:  
            numLetters = 8;  
    }  
}
```

A screenshot of the SonarLint interface. At the top, there are tabs: 'Problems' (selected), 'Javadoc', 'Declaration', 'SonarLint Report' (selected), and 'SonarLint Rule Description'. On the right side of the header are standard window controls (minimize, maximize, close). Below the tabs, the main content area shows a code smell report for 'Method parameters, caught exceptions and foreach variables' initial values should not be ignored (java:S1226)'. It includes a 'Bug' icon, a 'Minor' severity indicator, and a detailed description: 'While it is technically correct to assign to parameters from within method bodies, doing so before the parameter value is read is likely a bug. Instead, initial values of parameter and foreach parameters should be, if not treated as `final`, then at least read before reassignment.'

### Noncompliant Code Example

```
public void doTheThing(String str, int i, List<String> strings) {  
    str = Integer.toString(i); // Noncompliant  
  
    for (String s : strings) {  
        s = "hello world"; // Noncompliant  
    }  
}
```

# SonarLint – Konfiguration



Preferences

Rules Configuration

Configure rules used for SonarLint analysis. When a project is connected to a SonarQube/SonarCloud server, configuration from the server applies.

All rules

(No rules selected)

type filter text or rule key

HTML

Java

- ".equals()" should not be used to test the values of "Atomic" c
- "=+" should not be used instead of "+="
- "==" and "!=" should not be used when "equals" is overridden
- "@CheckForNull" or "@Nullable" should not be used on primit
- "@Controller" classes that use "@SessionAttributes" must call
- "@Deprecated" code marked for removal should never be use
- "@Deprecated" code should not be used
- "@EnableAutoConfiguration" should be fine-tuned
- "@Import"s should be preferred to "@ComponentScan"s
- "@NotNull" values should not be set to null
- "@Override" should be used on overriding and implementing i
- "@RequestMapping" methods should be "public"
- "@SpringBootApplication" and "@ComponentScan" should no
- "ActiveMQConnectionFactory" should not be vulnerable to ma
- "Arrays.stream" should be used for primitive arrays
- "Bean Validation" (JSR 380) should be properly configured
- "BigDecimal(double)" should not be used
- "Class.forName()" should not load JDBC 4.0+ drivers
- "Cloneables" should implement "clone"
- "Collections.EMPTY\_LIST", "EMPTY\_MAP", and "EMPTY\_SET" s
- "D

No parameters

Restore Defaults

Cancel

type filter text

General  
Ant  
Data Management  
Gradle  
Help  
Install/Update  
Java  
Java EE  
Java Persistence  
JSON  
Language Servers  
Maven  
Mylyn  
Oomph  
Plug-in Development  
Run/Debug  
Server  
SonarLint  
Analyzer Properties  
File Exclusions  
Miscellaneous  
Rules Configuration  
Team  
Terminal  
TextMate  
Validation  
Web  
Web Services  
XML  
XML (Wild Web Developer)  
YAML

?

?

?

?

?



Eclipse Marketplace

Eclipse Marketplace

Select solutions to install. Press Install Now to proceed with installation.  
Press the "more info" link to learn more about a solution.

Search Recent Popular Favorites Installed Giving IoT an Edge

Find:  x All Markets ▼ All Categories ▼ Go

274 Installs: **179K** (939 last month) Install

**pmd-eclipse-plugin 4.17.0**

PMD is a source code analyzer. It finds common programming flaws like unused variables, empty catch blocks, unnecessary object creation, and so forth. It supports... [more info](#)  
by [PMD](#), BSD  
[PMD](#) [linter](#) [Source Code Analyzer](#) [code quality](#) [java](#)

102 Installs: **46.5K** (1'345 last month) Installed

**SWAMP Eclipse Plug-in**

The SWAMP Eclipse Plug-in allows users to easily run static analysis tools available on the Software Assurance Marketplace (<https://www.continuousassurance.org/>)... [more info](#)  
by [Software Assurance Marketplace](#), Apache 2.0

**Marketplaces**



- **Findet eine Menge an potenziellen Problemen**
- **Fehlermeldungen sind doch etwas kryptisch**

Element	# Violations	# Violations/KLOC	# Violations/Methc	Project
examples	74	732.7	8.22	AAAAAAA_QualityCheckExamples
SomeOtherViolationsExample2.java	34	618.2	11.33	AAAAAAA_QualityCheckExamples
SillyNullAccessExample.java	23	821.4	4.60	AAAAAAA_QualityCheckExamples
CommentRequired	5	178.6	1.00	AAAAAAA_QualityCheckExamples
MethodArgumentCouldBeFinal	4	142.9	0.80	AAAAAAA_QualityCheckExamples
DataflowAnomalyAnalysis	3	107.1	0.60	AAAAAAA_QualityCheckExamples
LocalVariableCouldBeFinal	2	71.4	0.40	AAAAAAA_QualityCheckExamples
UnusedAssignment	2	71.4	0.40	AAAAAAA_QualityCheckExamples
ShortVariable	2	71.4	0.40	AAAAAAA_QualityCheckExamples
SwitchStmtsShouldHaveDefault	1	35.7	0.20	AAAAAAA_QualityCheckExamples
LawOfDemeter	1	35.7	0.20	AAAAAAA_QualityCheckExamples
AtLeastOneConstructor	1	35.7	0.20	AAAAAAA_QualityCheckExamples
MissingBreakInSwitch	1	35.7	0.20	AAAAAAA_QualityCheckExamples
AvoidReassigningParameters	1	35.7	0.20	AAAAAAA_QualityCheckExamples

# Checkstyle



Eclipse Marketplace

Eclipse Marketplace

Select solutions to install. Press Install Now to proceed with installation.  
Press the "more info" link to learn more about a solution.

Search Recent Popular Favorites Installed Giving IoT an Edge

Find:  All Markets All Categories Go

18 installs: 20.6K (87 last month) [Install](#)

**Checkstyle Plug-in 8.35.0**  
The Checkstyle Plugin (eclipse-CS) integrates the well-known source code analyzer Checkstyle into the Eclipse IDE. Checkstyle is a development tool to help you... [more info](#)  
by [Lars Ködderitzsch](#), LGPL  
[static analysis](#) [Favorite](#) [validator](#) [coding style](#)

101 installs: 498K (4'638 last month) [Install](#)

**Scalastyle 0.9.0**  
Scalastyle examines your Scala code and indicates potential problems with it. If you have come across Checkstyle for Java, then you'll have a good idea what... [more info](#)

Marketplaces

# Checkstyle



- Checkstyle kann hilfreich sein, aber vieles über Formatter und Save Action lösbar

- <https://www.vogella.com/tutorials/Checkstyle/article.html>
- <https://www.baeldung.com/checkstyle-java>

▼ Checkstyle 86 warnings

▼ Checkstyle real-time scan 86 warnings

▼ C M Main 4 warnings

Checkstyle: 'method def modifier' has incorrect indentation level 4, expected level should be 2.

Checkstyle: 'method def' child has incorrect indentation level 8, expected level should be 4.

Checkstyle: Line contains a tab character.

Checkstyle: 'method def rcurly' has incorrect indentation level 4, expected level should be 2.

▼ I M Validation.java 2 warnings

Checkstyle: Line contains a tab character.

Checkstyle: 'package' should be separated from previous

Overview of Checkstyle violations - 237 markers in 7 categories (filter matched 237 of 237 items)

Checkstyle violation type	Occurrences
⚠ 'X' sollte in einer neuen Zeile stehen.	1
⚠ Der Bezeichner 'X' entspricht nicht dem Muster 'X'.	7
⚠ Das Konstrukt 'X' muss geschweifte Klammern '{}' verwenden.	8
⚠ Es fehlt ein Javadoc-Kommentar.	10
⚠ 'X' an Position X sollte in der vorherigen Zeile stehen.	38
⚠ Das Unterelement von 'X' hat eine unerwartete Einrückungstiefe von X (er...	57
⚠ 'X' hat eine unerwartete Einrückungstiefe von X (erwartet: X).	116



# DEMO

# Quality Check Example



- **SonarLint**
  - Gute Default-Konfiguration
  - Extrem hilfreich, insbesondere Fehlermeldungen und auch Erklärungen
  - Schön in Eclipse eingebunden
  - Verknüpfung mit Server und Konfiguration möglich und sinnvoll
- **PMD**
  - Schlechte Granularität und nicht so gut verständliche Fehlermeldungen
  - GUI eher altbacken
- **Checkstyle**
  - Schlechte Granularität und Fokus auf unwesentliche Dinge
  - Zu viele, wenig relevante und nicht so gut verständliche Fehlermeldungen



# Thank You