Programming Style & Coding Conventions

Michael Inden

Coding Guidelines – Agenda

- Initial thoughts
- Bad habits and remedies
- Coding Guidelines
- Tooling

Initial Thoughts

DON‘T DO THAT

- **" Write your code, [...] leave the bugs to the quality assurance or testing department."**
- "In the bad old days, programmers built every application from scratch."
- **"Just keep changing until it works."**

DO THAT

- **"Good programmers are always a little lazy, too: they'd rather sit back and wait for an insight than get started right away with their first idea."**

- "Why do programmers write large programs when small ones will do? One reason is that they lack the [...] important laziness; they directly start coding their first idea."

Main ideas

DO THAT

- **"Too many programmers rush to >>THE<< solution to their problem too quickly. They think for a minute and write code for a day, instead of thinking for an hour and then coding for even an hour."**

- "If a system-provided [...] function meets your needs, you shouldn't even think about writing your own."

- **"At the stage of the design process, it is invaluable to know the relevant literature."**

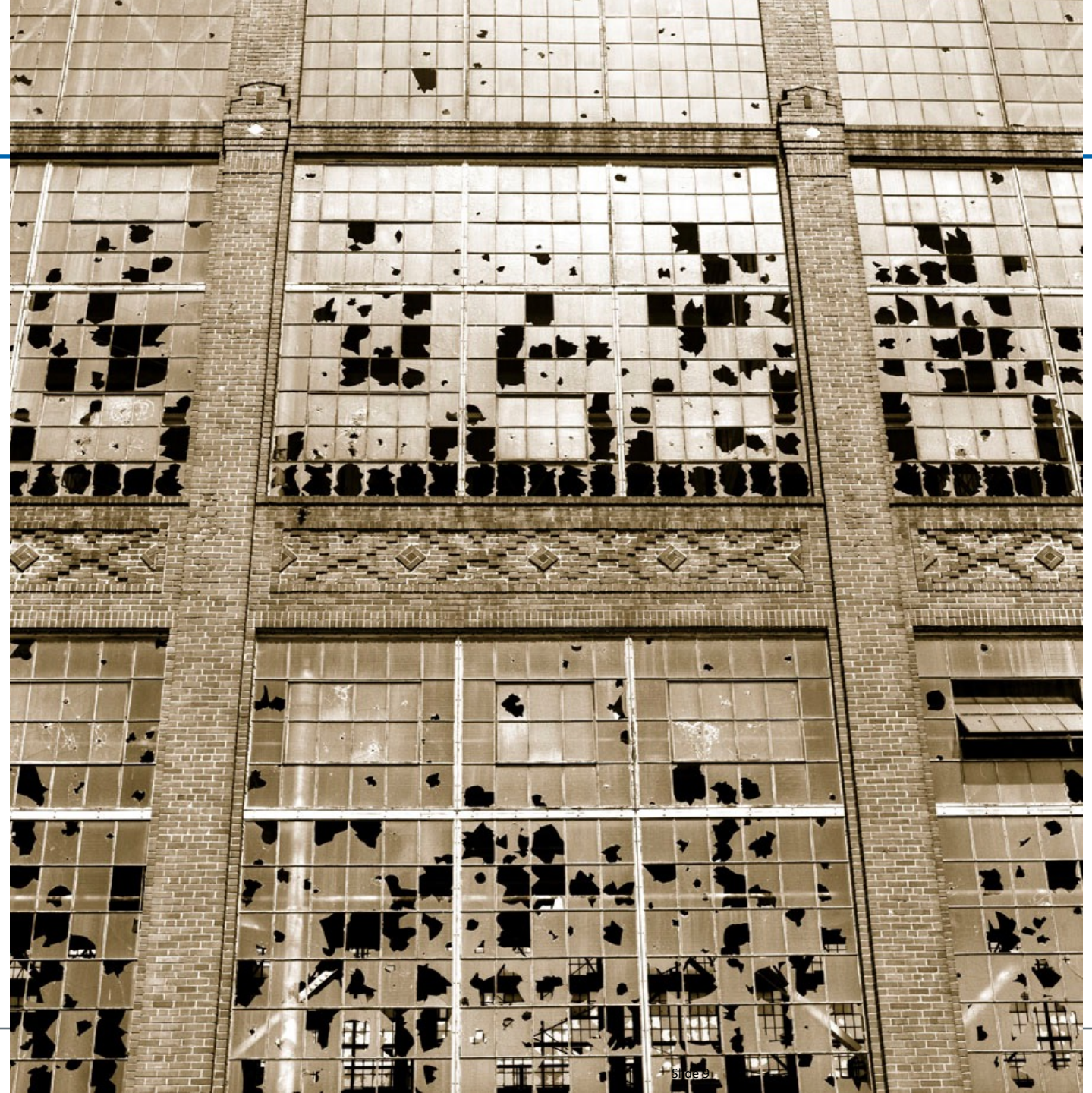
Main ideas

DO THAT

- **"The purpose of software engineering is to control complexity, not to create it."**
- "A correct view of the data actually structures the programs."
- **"Always ensuring simple code is usually the key to correctness!"**
- "The most important principle in code tuning is: do it rarely."

Bad habits and remedies

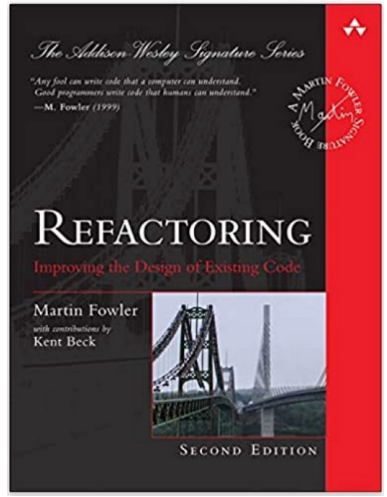
Broken Window Phenomena



Hotelroom Rule: Somebody is cleaning up for you

Boyscout Rule: Leave the campground cleaner than you found it

Remedey Continous Refactroing



Definition

In the book, I make the following definition of “refactoring”

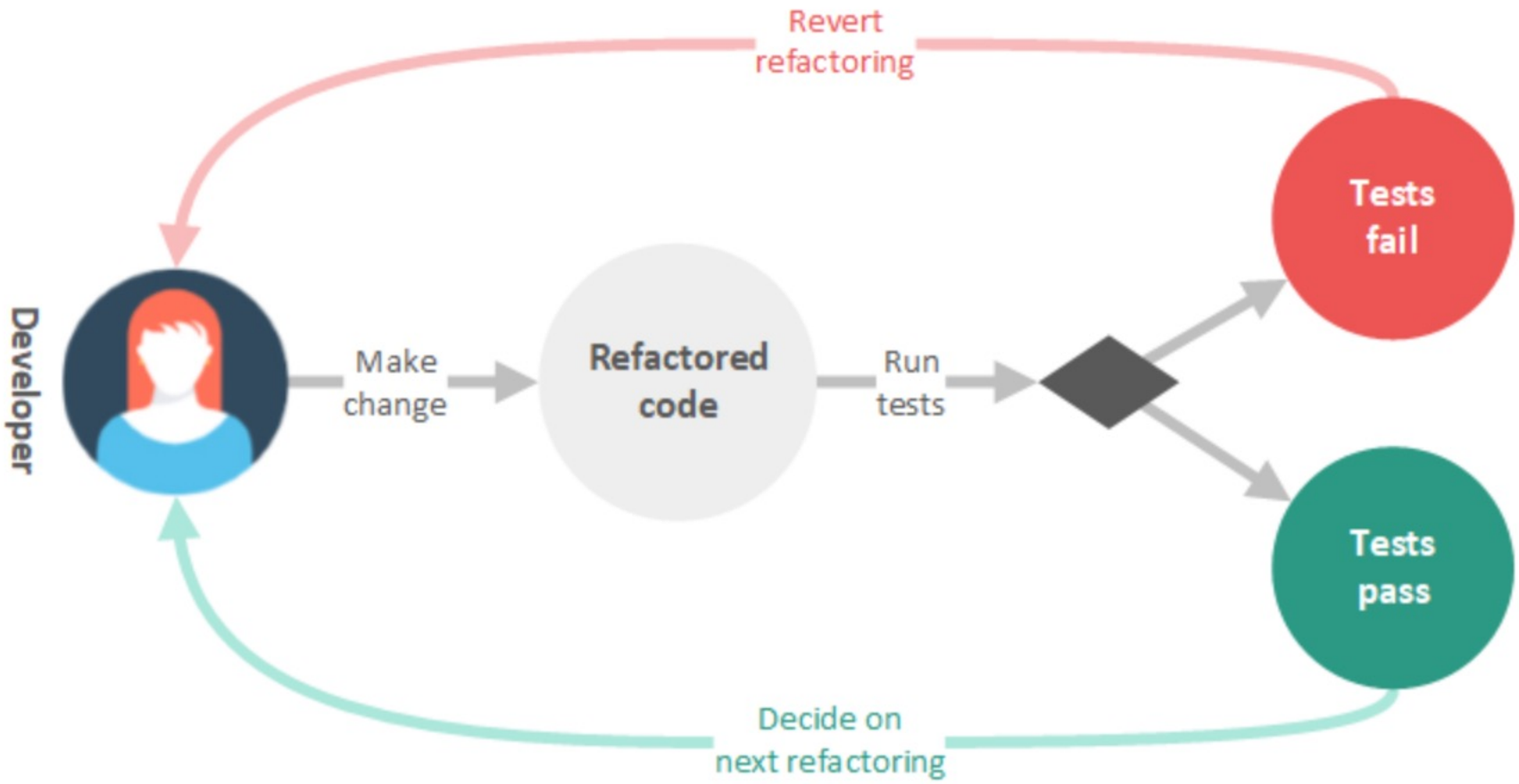
noun: *a change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior*

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.

https://refactoring.com/

Recap: Refactoring in Practice



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

Coding Guidelines

"Any fool can write code that computers can understand."

"Good programmers write code that humans can understand."

- Martin Fowler -

Coding Guidelines Interesting facts

We READ 10x more time than we WRITE

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

Coding Guidelines

- It has proven over and over again to be an advantage to adhere to certain conventions when programming.
- The programming style and also the layout of the source code have a great influence on its readability and comprehensibility and later maintainability.
- Formatting Conventions - How do I arrange the layout?
- Naming Conventions - How do I find (meaningful), uniform names?
- Coding Conventions - How do I make code readable? How do I avoid simple errors?
- GENERAL OBJECTIVE: Automate as much as possible with tools / have them check

Formatting Conventions

Properties for A_OO_Design

type filter text

- Resource
- Builders
- Coverage
- Java Build Path
- Java Code Style
  - Clean Up
  - Code Templates
  - Formatter
  - Organize Imports
- Java Compiler
- Javadoc Location
- Java Editor
- MoreUnit
- PMD
- Project Facets
- Project Natures
- Project References
- Refactoring History
- Run/Debug Settings
- SonarLint
- Task Repository
- Task Tags
- Validation
- WikiText

Formatter

Enable project specific settings — Configure Workspace Settings...

Active profile:

Michaels_CodeFormat — Edit... — Remove

New... — Import... — Export All...

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

Cancel — Apply and Close

Naming Conventions

- How do I find (meaningful), consistent names?
- 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

- **How do I find (meaningful), consistent names?**
  - **Avoid abbreviations:** ~~cnt~~, ~~ser~~, ~~ikm~~
    - Exceptions: Html, Xml, Json, it, sb, ...
  - Vermeide Variablennamen, die nur den Typ wiederholen

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

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

  - Name classes / methods according to the provided functionality
  - *And*, *Or*, *_* are indicators of too much functionality => EXTRACT METHOD
    `saveAndPrint(), formatAndSendMail(), copyOrMove(), select_mark_calc_avg()`

Naming Conventions

- **How do I find (meaningful), consistent names?**
- **Further inspirations:**
  - Chapter «Coding Conventions» from my book «Java-Profi»
  - http://www.ambysoft.com/essays/javaCodingStandards.html
  - 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

- How do I make code readable? How do I avoid simple errors?

- Use descriptive Comments but do NOT overcomment
  Document your public methods espacially 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 exits
     */
    IFilterAttributeRO getAttributeForName ( final String strAttributeName );
   … … …
 )
```

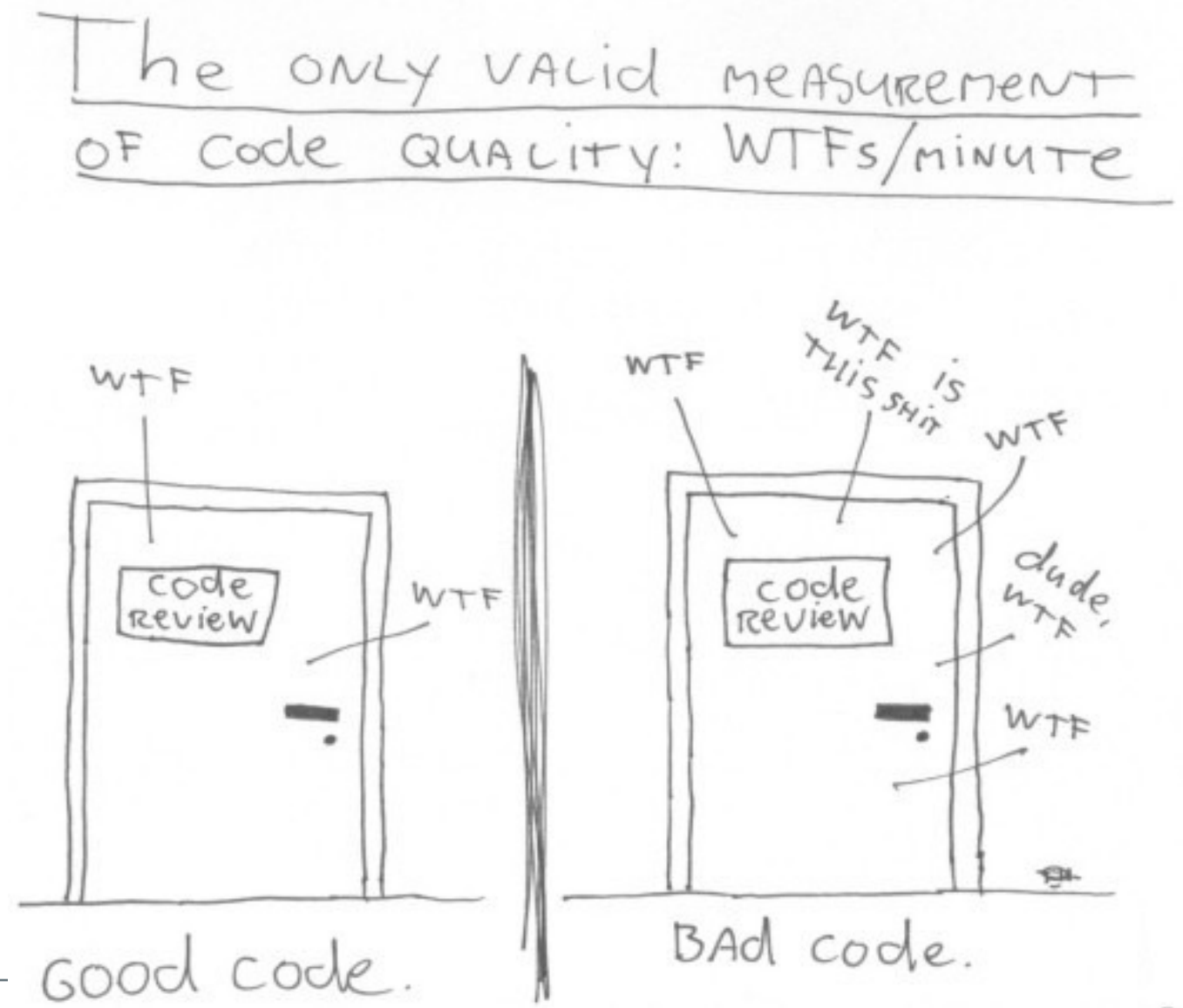
Coding Conventions Checklist – coding hints and rules

- 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 IDE: extract method, extract local variable, ...**

Coding Guidelines – The ONLY Truth



Tooling

Tooling

- Eclipse Configuration Errors/Warnings
- Eclipse Save Actions
- SonarLint
- PMD
- Checkstyle

Eclipse Configuration Errors/Warnings

Properties for AAAAAAA_QualityCheckExamples

type filter text

- Resource
- Builders
- Coverage
- Java Build Path
- Java Code Style
- Java Compiler
  - Annotation Processing
  - Building
  - Errors/Warnings
  - Javadoc
  - Task Tags
- Javadoc Location
- Java Editor
- Project Facets
- Project Natures
- Project References
- Run/Debug Settings
- Task Repository
- Task Tags
- Validation
- WikiText

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 Configuration 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 Configuration Errors/Warnings

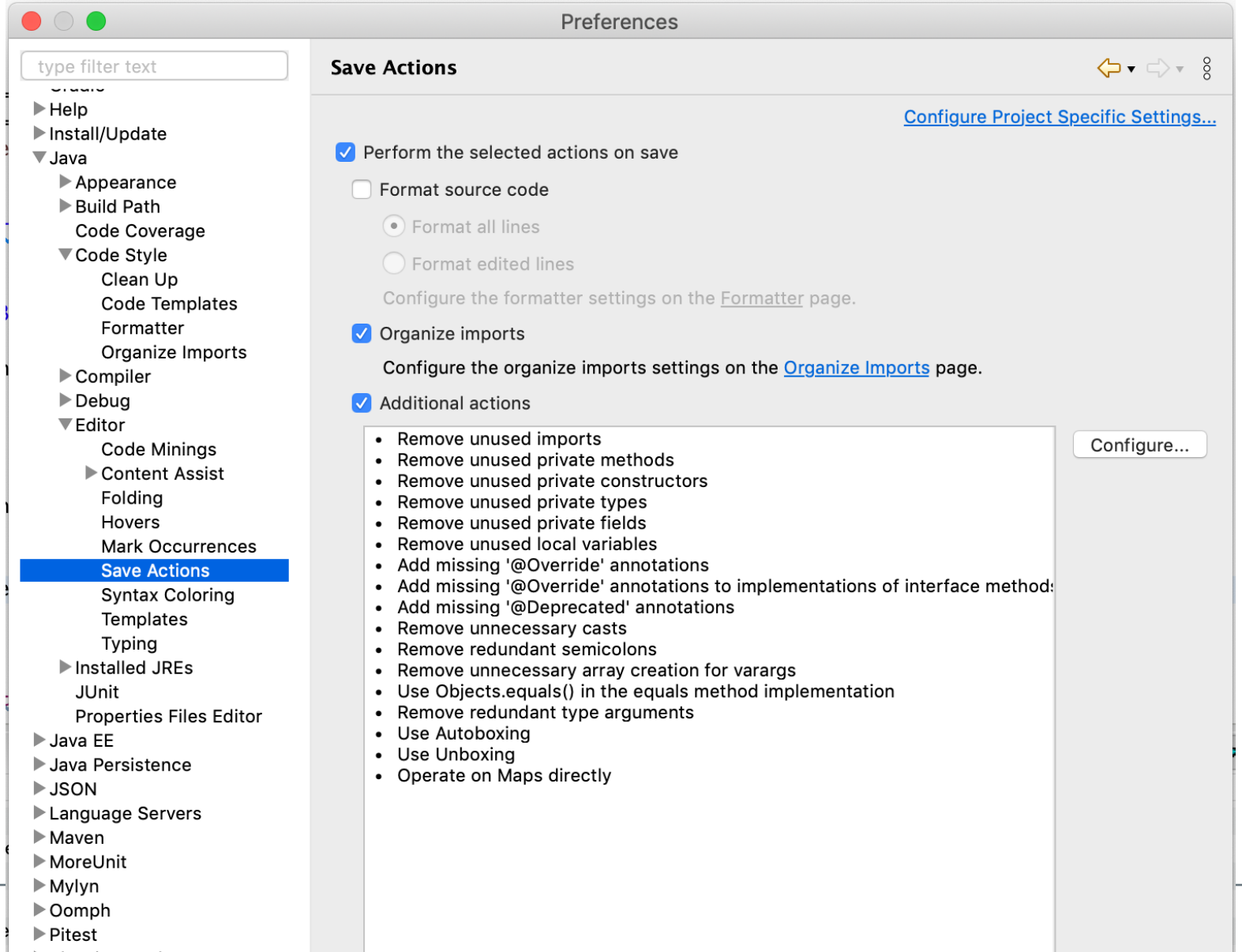
Unnecessary code

Value of local variable is not used:	Warning
Value of method parameter is not used:	Warning
☑ Ignore in overriding and implementing methods	
Unused type parameter:	Warning
☑ 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
☑ Ignore in overriding and implementing methods	
☑ Ignore exceptions documented with '@throws' or '@exception' tags	
☑ Ignore 'Exception' and 'Throwable'	
Unused 'break' or 'continue' label:	Warning
Redundant super interface:	Ignore

Eclipse Configuration 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



SAVE ACTIONS – Coding Guidelines

Additional Save Actions

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

Unused code

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

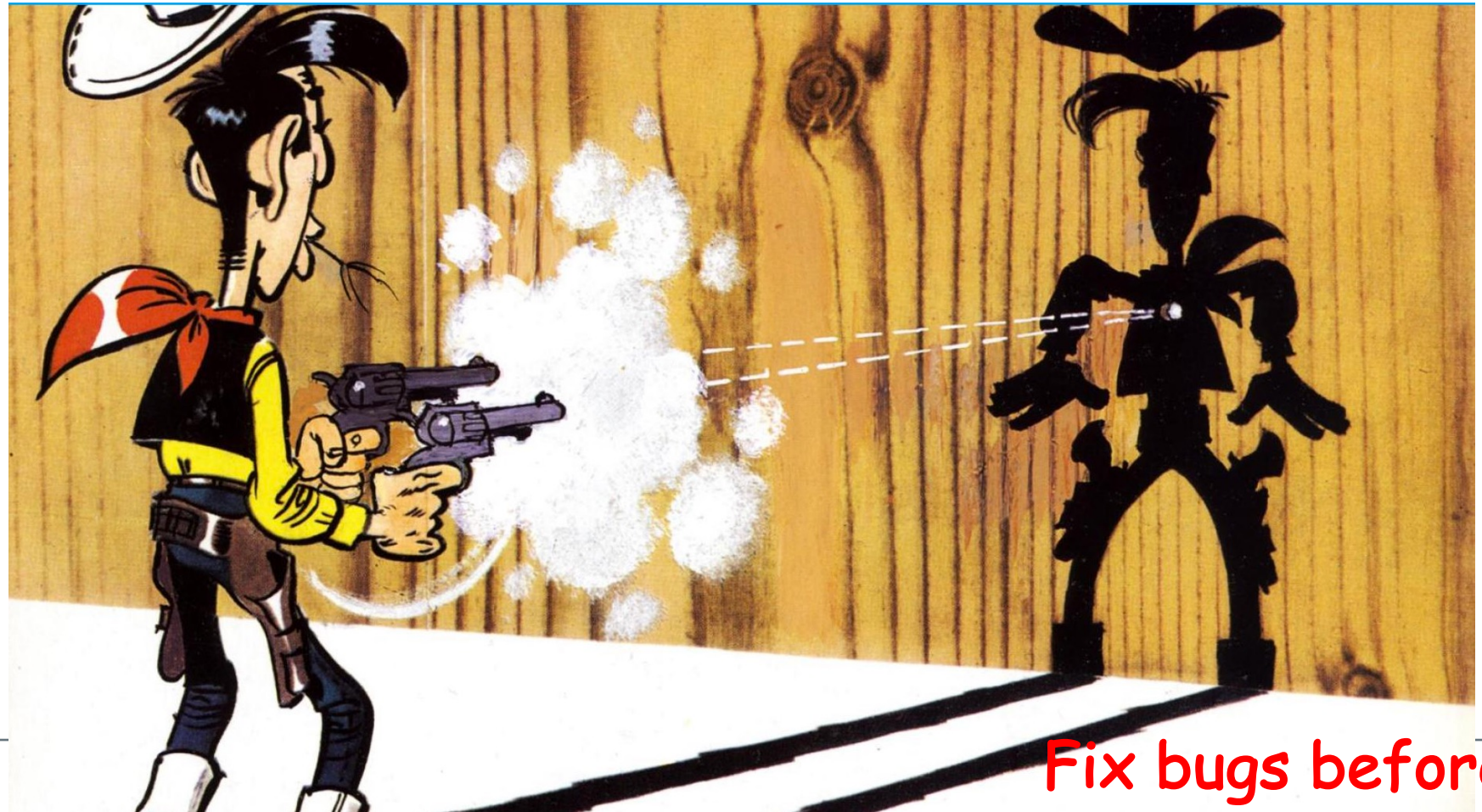
Unnecessary code

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

Preview:

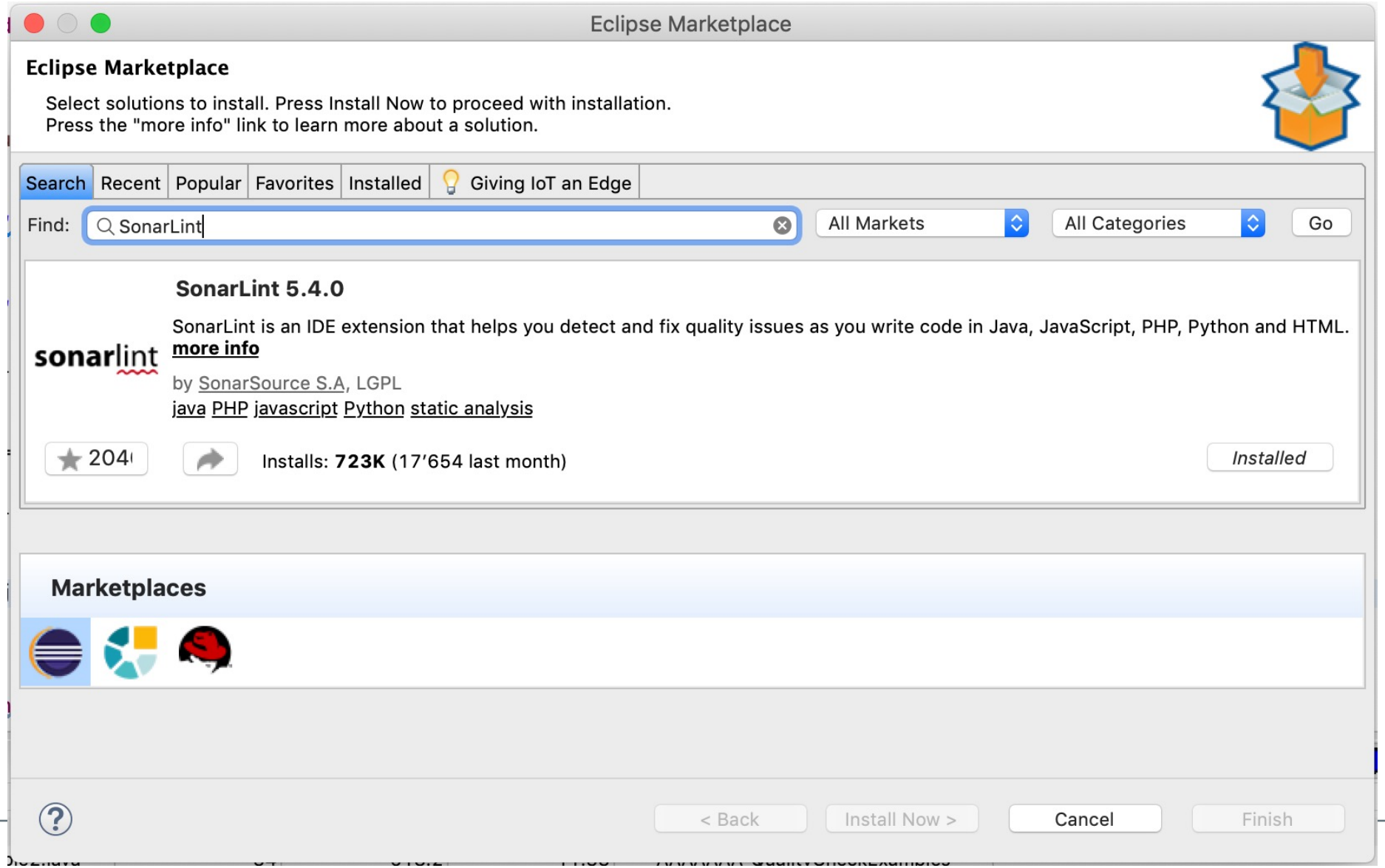
```
class Example {
    public Example(boolean b) {}
    public void bar() {
    }
}

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

SonarLint


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:
SonarLint
All Markets
All Categories
Go
SonarLint 5.4.0
sonarlint
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
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

SonarLint

- As a supplement or stand alone to SonarQube Server

- Static analysis of source code (offline)
- Eclipse / IntelliJ plugin provides feedback directly while programming
- Finds a lot of potential problems
- Error messages are (very) understandable, good background documentation

- Fixing during active development prevents bugs or at least code smells from reaching the repository
- Fixing reduces manual efforts for reviews
- Increases quality in small steps and without much effort (just by the way)

SonarLint

- Finds a lot of potential problems
- Error messages are (very) understandable

Violations Overview | SonarLint Report

99 items

Resource	Date	Description
BuilderPattern.java		Rename this local variable to match the regular expression '^[a-z][a-zA-Z0-9]*$'.
BuilderPattern.java		Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.
Circle.java		Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.
Command.java	5 day...	This block of commented-out lines of code should be removed.
Command.java		Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.
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-z0-9_]*)*$'.
ContentLine.java	2 hou...	Remove this unused "line" private field.
ContentLine.java		Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.
Counter.java		Add a nested comment explaining why this method is empty, throw an UnsupportedOperationException o...
CounterV2.java		Add a nested comment explaining why this method is empty, throw an UnsupportedOperationException o...
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.

SonarLint

- **Error messages are (very) understandable, good background documentation**

Violations Overview | SonarLint Report | SonarLint Rule Description

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) {  // Noncompliar
      case MONDAY:
      case FRIDAY:
      case SUNDAY:
        numLetters = 6;
        break;
      case TUESDAY:
        numLetters = 7;
        break;
      case THURSDAY:
      case SATURDAY:
        numLetters = 8;
```

Problems | Javadoc | Declaration | SonarLint Report | SonarLint Rule Description

Method parameters, caught exceptions and foreach variables' initial values should not be ignored (java:S1226)

Bug | Minor

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 parame
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 – Configuration

Preferences

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

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

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 cal
  - "@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
  - "@NonNull" values should not be set to null
  - "@Override" should be used on overriding and implementing
  - "@RequestMapping" methods should be "public"
  - "@SpringBootApplication" and "@ComponentScan" should nc
  - "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"

(No rules selected)

No parameters

Restore Defaults | Apply

Cancel | Apply and Close

PMD

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: PMD | 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

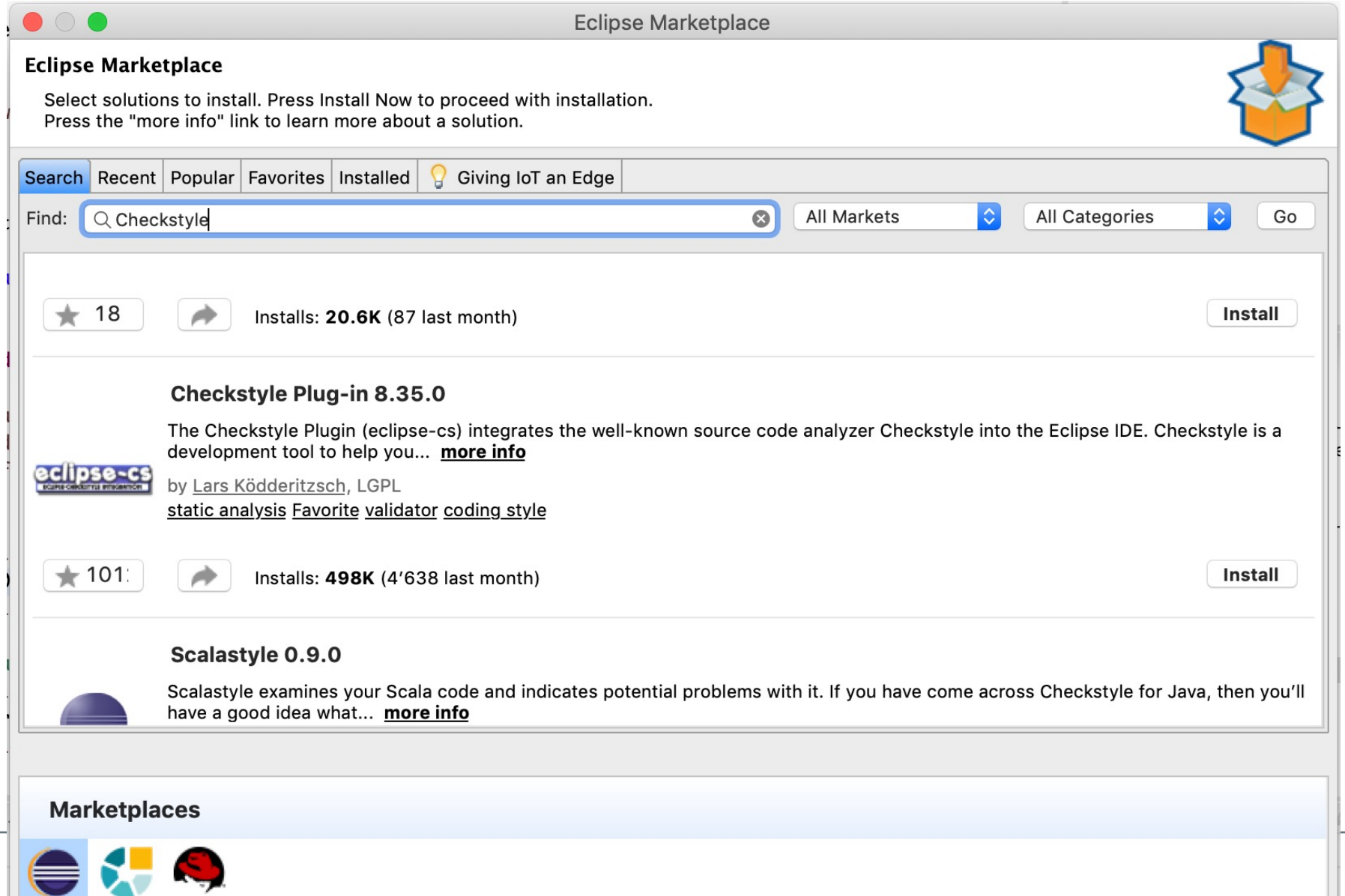
Marketplaces

PMD

- Finds a lot of potential problems
- Error messages are a bit cryptic after all

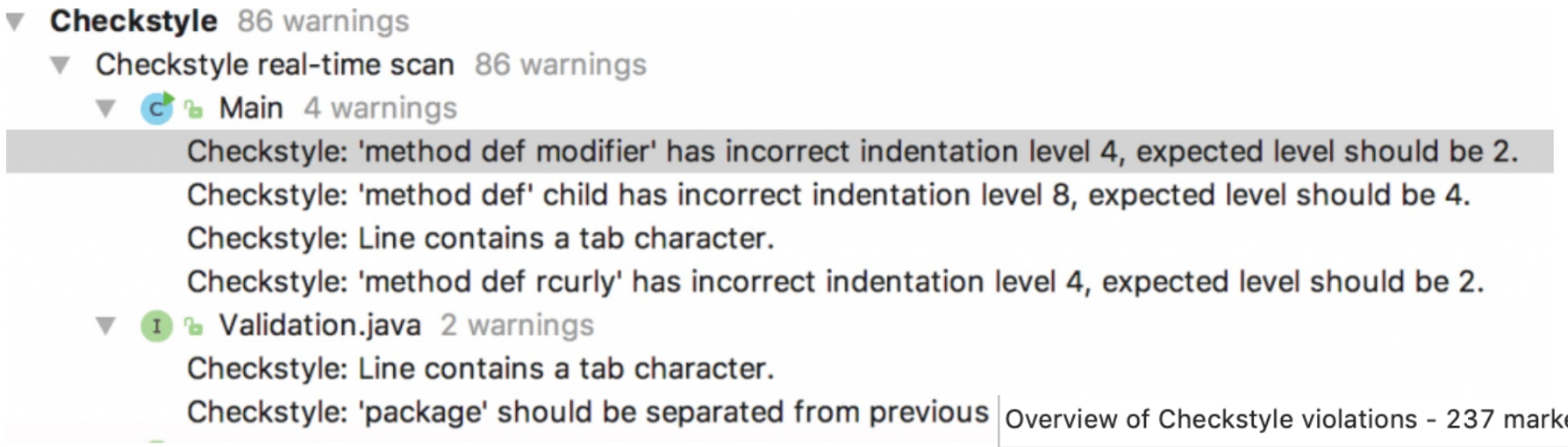
Element	# Violations	# Violations/KLOC	# Violations/Methc	Project
examples	74	732.7	8.22	AAAAAAA_QualityCheckExamples
SomeOtherViolationsExample2.java	34	618.2	11.33	AAAAAAA_QualityCheckExamples
SillyNullAccessExample.java	23	821.4	4.60	AAAAAAA_QualityCheckExamples
CommentRequired	5	178.6	1.00	AAAAAAA_QualityCheckExamples
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:
Checkstyle
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 can be helpful, but much can be solved via formatter and save action**
  - https://www.vogella.com/tutorials/Checkstyle/article.html
  - https://www.baeldung.com/checkstyle-java



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

Tools compared

- **SonarLint**
  - Good default configuration
  - Extremely helpful, especially error messages and also explanations
  - Nicely integrated into Eclipse
  - Link to server and configuration possible and useful
- **PMD**
  - Poor granularity and somewhat unclear error messages
  - GUI rather old-fashioned
- **Checkstyle**
  - Poor granularity and focus on non-essentials
  - Too many, not very relevant and not so well understandable error messages

Thank You