CheckStyle Explanation

Stefan Boodt 4322258
Arthur
Tomas
Daan
Jordy

May 17, 2015

Abstract

This document describes the use of our CheckStyle design decissions.

Contents

Chapter 1

Explanation of differences

This chapter describes our differences with the default CheckStyle configuration of google style.

1.1 General

- The tabsize is set to 4 because 4 is the default indentation for java.
- Disallows the use of whitespaces in generics. List¡ Integer ¿ is disallowed.
- Tabs are allowed in the sourcecode as they are an easy way to indent that is commonly used by editors behind the scenes.

1.2 Code

- CheckStyle should ignore inline conditionals as they are not always bad. The team can decide for themselves if the inline conditional is readable enough to pass.
- Every equals method should have an @Override. This is inspired by FindBugs. The reason is that the code then throws errors on not standard equals methods that are written as if they are equals(Object).
- Default should be the last case in a switch statement because this is more readable.
- It is a CheckStyle error to catch Exception, RunTimeException and Throwable since these catch things that should not be catched, such as OutOfMemoryExceptions and RunTimeExceptions. A RuntimeException is not to be catched.

- All integers are not considered by the magic numbers check. This comes from the high usage of these numbers in the test case and the waste of memory that comes from initializing that many variables simply to use as the expected outcome. Reviews of pull requests should mention magic numbers in non test classes.
- No classes may appear in a default package as that means that they cannot be imported. CheckStyle throws errors on this.
- The override of the method finalize should include a super call because of what finalize does. CheckStyle is configured to do something about this.
- CheckStyle has to throw an error if the cyclomatic complexity of a method is more than 10.
- CheckStyle should throw just a warning for non final parameters in a method or constructor definition.
- Long literals should contain a uppercase L. Not a lowercase one since that resembles an 1. CheckStyle should enforce this.
- CheckStyle should ignore its design for extension argument since this requires methods to be final, while we learned that final methods should be used less during Software Engineering Methods.

1.3 Javadoc

- Missing @Override are seen as errors when @InheritDoc is used because the source cannot be found.
- Annotations should appear between javadoc and method name because of convention. CheckStyle should throw an error if it does not.
- The package annotation is set as well to disallow the javadoc documentation for packages outside the package-info.java file.
- The non empty @ clause is reported as CheckStyle error because we find it not very nice that someone does not write something behind @return or @param.
- The javadoc paragraphs is set so that you need to write a whiteline and a ¡p¿ for every paragraph. This keeps multi paragraph comments readable.

1.4 Import

- We added an import order check to check if the imports are alfabetical and there is a white line between the groups.

1.5 Temporary Checking Commands

This explains how you can manually alter CheckStyle output in the code. Usage of these commands should be explained.

- To stop the CheckStyle tests from running you can write CHECKSTYLE:ON in your comments.
- To restart CheckStyle tests you write CHECKSTYLE:OFF in a comment.
- Alternatively the @SuppressWarnings can be used for this.