Style Checking

Advanced Software Engineering
Fall 2022

Ira Ceka

Why It Matters

“Indeed, the ratio of time spent reading versus writing is well over 10 to 1. We are constantly reading old code as part of the effort to write new code. …[Therefore,] making it easy to read makes it easier to write.”

– **Robert C. Martin**

“It turns out that style matters in programming for the same reason that it matters in writing. It makes for better reading.”

– **Douglas Crockford**

Demo Tools

Java

spring initializr

checkstyle

https://checkstyle.sourceforge.io

C++

Visual Studio Code

cpplint / cpplint Public

forked from google/styleguide

https://github.com/cpplint/cpplint

checkstyle

Some standard checks applied to your source code:

- **Indentation** - Checks correct indentation
- **WhiteSpaceAround** - Checks that a token is surrounded by whitespace
-
- **UnusedImports** - Checks for unused import statements
- **ImportOrder** - Checks the ordering/grouping of imports

checkstyle

```
import java.io.IOException;
import java.net.URL;

import java.io.IOException;

import javax.net.ssl.TrustManager;
import javax.swing.JComponent;
import org.apache.http.conn.ClientConnectionManager;
import java.util.Set;
import com.neurologic.http.HttpClient;
import com.neurologic.http.impl.ApacheHttpClient;
```

https://checkstyle.sourceforge.io/config_imports.html#ImportOrder

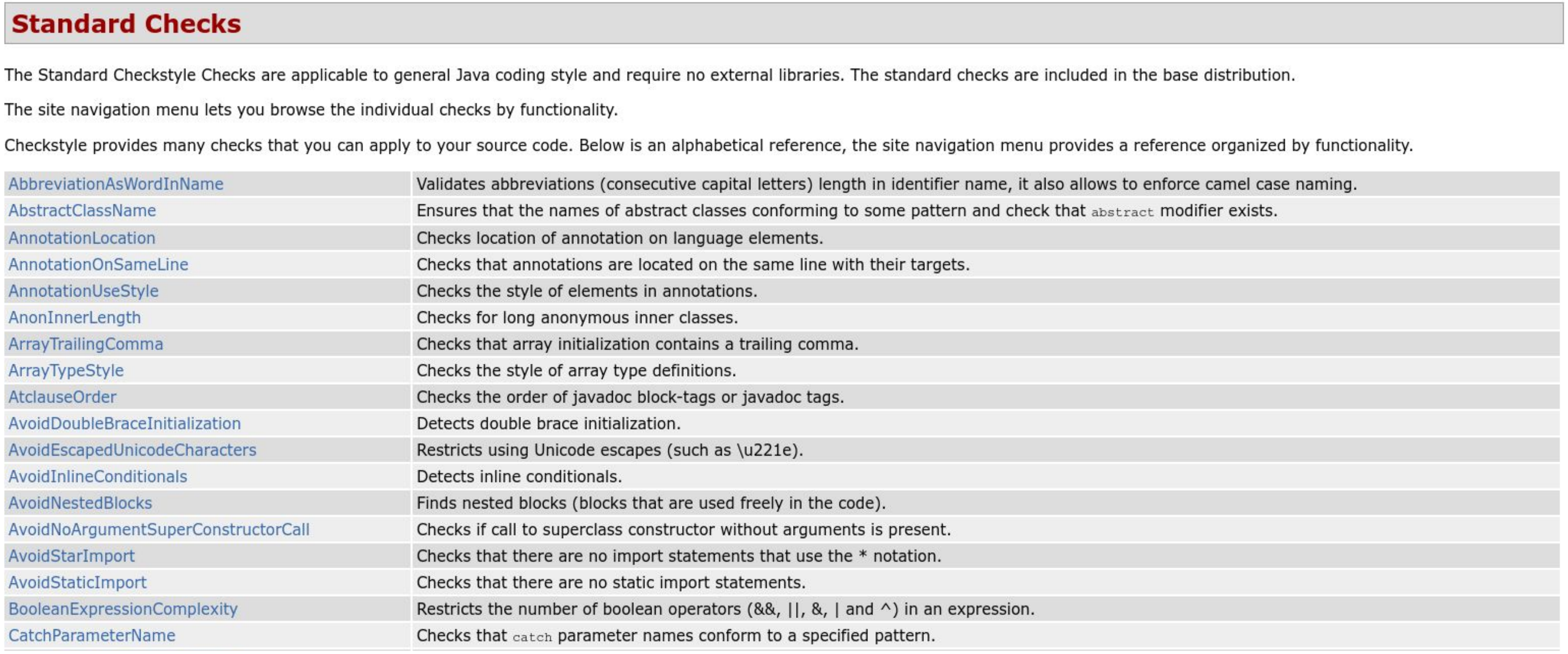
checkstyle

```
import java.io.IOException;
import java.net.URL;

import java.io.IOException; // violation, extra separation before import
                            // and wrong order, comes before 'java.net.URL'.
import javax.net.ssl.TrustManager; // violation, extra separation due to above comment
import javax.swing.JComponent;
import org.apache.http.conn.ClientConnectionManager; // OK
import java.util.Set; // violation, wrong order, 'java' should not come after 'org' imports
import com.neurologic.http.HttpClient; // violation, wrong order, 'com' imports comes at top
import com.neurologic.http.impl.ApacheHttpClient; // OK
```

https://checkstyle.sourceforge.io/config_imports.html#ImportOrder

checkstyle



Standard Checks

The Standard Checkstyle Checks are applicable to general Java coding style and require no external libraries. The standard checks are included in the base distribution.

The site navigation menu lets you browse the individual checks by functionality.

Checkstyle provides many checks that you can apply to your source code. Below is an alphabetical reference, the site navigation menu provides a reference organized by functionality.

AbbreviationAsWordInName	Validates abbreviations (consecutive capital letters) length in identifier name, it also allows to enforce camel case naming.
AbstractClassName	Ensures that the names of abstract classes conforming to some pattern and check that `abstract` modifier exists.
AnnotationLocation	Checks location of annotation on language elements.
AnnotationOnSameLine	Checks that annotations are located on the same line with their targets.
AnnotationUseStyle	Checks the style of elements in annotations.
AnonInnerLength	Checks for long anonymous inner classes.
ArrayTrailingComma	Checks that array initialization contains a trailing comma.
ArrayTypeStyle	Checks the style of array type definitions.
AtclauseOrder	Checks the order of javadoc block-tags or javadoc tags.
AvoidDoubleBraceInitialization	Detects double brace initialization.
AvoidEscapedUnicodeCharacters	Restricts using Unicode escapes (such as \u221e).
AvoidInlineConditionals	Detects inline conditionals.
AvoidNestedBlocks	Finds nested blocks (blocks that are used freely in the code).
AvoidNoArgumentSuperConstructorCall	Checks if call to superclass constructor without arguments is present.
AvoidStarImport	Checks that there are no import statements that use the * notation.
AvoidStaticImport	Checks that there are no static import statements.
BooleanExpressionComplexity	Restricts the number of boolean operators (&&, ||, &, | and ^) in an expression.
CatchParameterName	Checks that `catch` parameter names conform to a specified pattern.

•••

https://checkstyle.sourceforge.io/checks.html

checkstyle

Predefined rulesets:

- Configuration from XML files
- 2 types of checks
    - **sun_checks.xml**
    - **google_checks.xml**
- Checks for Sun coding conventions by default

JAVA DEMO

cpplint

- Style-checker for C++
  - Makes sure your file is following Google's C++ guidelines!
- Written in Python and maintained by Google

  ```
  > $ pip install cpplint
  ```

- Run in command line

  ```
  > $ cpplint myfile.cpp
  ```

https://github.com/cpplint/cpplint

C++
DEMO

Thank you!