CheckStyle

Sang Shin
www.jPassion.com
"Learn with jPassion!"

Topics

- Why coding standard?
- What is & Why CheckStyle?
- Examples
- How to get started?
- Integration with other tools

Why Coding Standard?

Why Coding Standards?

- Coding Standards lead to greater consistency within your code and the code of your teammates.
- Easier to understand
- Easier to develop
- Easier to maintain
- Reduces overall cost of application

General Naming Conventions

- Use full English descriptors
  - *firstName*, *grandTotal*, or *CorporateCustomer*
- Use terminology applicable to the domain
  - Banking domain – *Customer*
  - Software service domain – *Client*
- Use mixed case to make names readable
  - *CorporateCustomer*
- Avoid long names
  - 15 characters or less

Java Naming Conventions

- Classes/Interfaces
  - Class names should be nouns

What is & Why CheckStyle?

What is CheckStyle?

- Checkstyle is a development tool to help programmers write Java code that adheres to a coding standard.
  - It automates the process of checking Java code to spare humans of this boring (but important) task.
  - This makes it ideal for projects that want to enforce a coding standard.
- Checkstyle is highly configurable and can be made to support almost any coding standard.
  - An example configuration file is supplied supporting the Sun Code Conventions.
  - Other sample configuration files are supplied for other well known conventions.

Features

- Checkstyle can check many aspects of your source code.
- Historically it's main functionality has been to check code layout issues, but since the internal architecture was changed in version 3, more and more checks for other purposes have been added.
- Now Checkstyle provides checks that find class design problems, duplicate code, or bug patterns like double checked locking.
  - Some duplication with FindBugs, PMD

Available Checks Examples

Ready-to-use Checks

- Annotations
- Block checks
- Class design
- Coding
- Duplicate code
- Headers
- Imports
- Javadoc comments
- Metrics

Ready-to-use Checks

- Annotations
- Block checks
- Class design
- Coding
- Duplicate code
- Headers
- Imports
- Javadoc comments
- Metrics

Coding Example: ArrayTrailingComma

- Checks that array initialization contains a trailing comma.
  - Rationale: Putting this comma in makes it easier to change the order of the elements or add new elements on the end.

```
// Good practice
int[] a = new int[]
{
    1,
    2,
    3,
};
```

- The check allows to not add a comma if both left and right curlys are on the same line.

```
// Good practice
return new int[] { 0 };
```

Coding Example: EqualsAvoidNull

- Checks that any combination of String literals with optional assignment is on the left side of an equals() comparison.
  - Rationale: Calling the equals() method on String literals will avoid a potential NullPointerException. Also, it is pretty common to see null check right before equals comparisons which is not necessary in the below example.

```
// Bad practice
String nullString = null;
nullString.equals("My_Sweet_String");  // NullPointerException

// Good practice
String nullString = null;
"My_Sweet_String".equals(nullString);
```

Coding Example: ModifiedControlVariable

- Check for ensuring that for loop control variables are not modified inside the for block. An example is:
  - Rationale: If the control variable is modified inside the loop body, the program flow becomes more difficult to follow. An option is to replace the for loop with a while loop.

```
// Bad practice
for (int i = 0; i < 1; i++) {
    i++;
}
```

How to Get Started

Running at the Commandline

- Run checkstyle with configuration file docs/sun_checks.xml on a file

```
// Run checkstyle with configuration file docs/sun_checks.xml on a file
java com.puppycrawl.tools.checkstyle.Main -c docs/sun_checks.xml
Check.java

// Run checkstyle with configuration file docs/sun_checks.xml on
// all java files in a directory
java com.puppycrawl.tools.checkstyle.Main -c docs/sun_checks.xml -r src/
```

Integration with other Tools

Integration with other tools

- Ant
- Maven
- IDE
  - Eclipse, NetBeans, IntelliJ IDEA, etc
- Continuous Integration Tool
  - Hudson, Jenkins
- Continuous Inspection Tool
  - Sonar

Thank you!

Check JavaPassion.com Codecamps!
http://www.javapassion.com/codecamps
“Learn with Passion!”