Part VI: Static Analyzers

Introduction to Static Analysis

What is a static analysis? Usually, static analysis means code review. In companies, especially in one project, colleagues will review your code manually.

- In one word, static analysis is mostly informal manual-human reviews of code.
- Also called "code reviews" or "compile-time analysis".

During our experience in `JSoup` project, we fork our project into our own repository and pull request into the master. One or two of our collaborators will peer view the PR code, give it some advice or directly merge it into the master.

There are best practices for the process of static analyzers.

1. Review small portions of code at a time
2. Record all feedback
3. Review code independently before gathering to discuss
4. Use checklists

Importance of Static Analysis

After talking about the basic concepts of static analysis, we come to the question why we need static analysis. Combined with its basic concepts, we concluded reasons of necessity of static analysis.

- Static analysis can help you find potential bugs early. Using static analysis, your code reviewers can help you find these bugs manually.
- Static analysis can help you stick to the same coding style or coding standard. Since more than one people are involved in one function or one line of code, the clarity of code will increase. In my opinion, to raise the possibility of merging code, you are not the only one who sees the code so that you need to make more comments and try to make the code neat and clear.
- Static analysis can help team collaboration. Your team need to involve in the same project and be responsible for every line of code. To achieve that, your team members must discuss more often and share your ideas on code.

Tools for Static Analysis

We used two different static analyzers on our project `JSoup`.

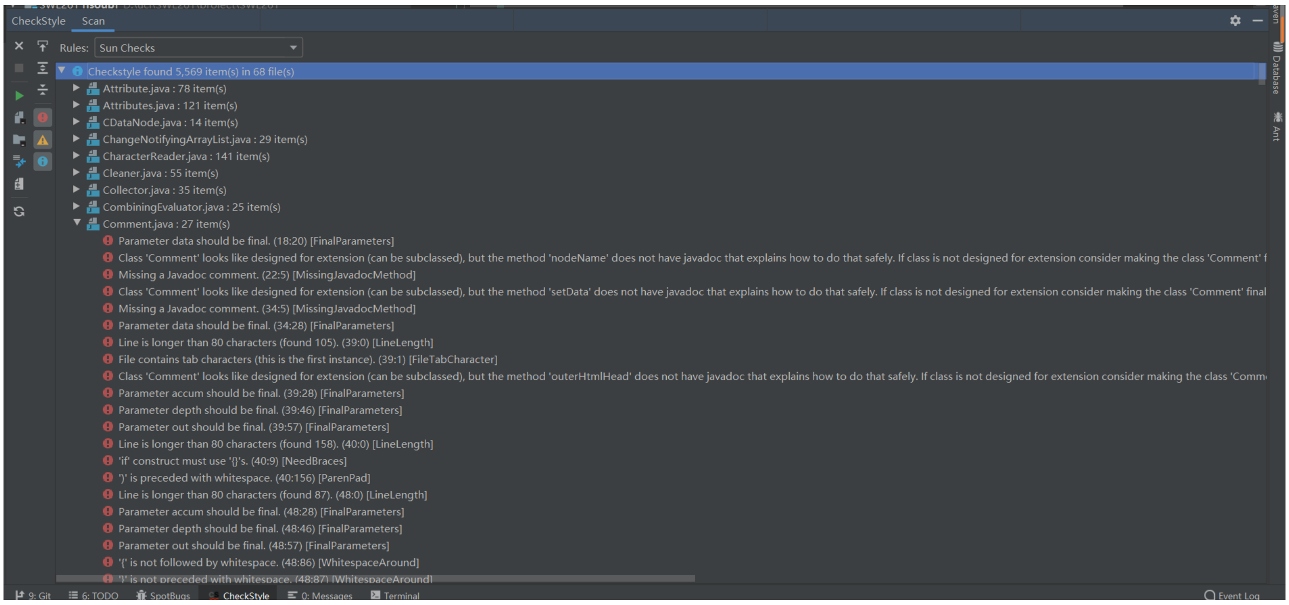
The first one is **SpotBugs**. SpotBugs is a program that uses static analysis to find errors in Java code.

The second one is **Checkstyle**. Checkstyle is a development tool that helps programmers write Java code that meets coding standards. Sample coding standard: google coding style or sun check.

Results of Static Analysis by Spotbugs and Checkstyle

CheckStyle

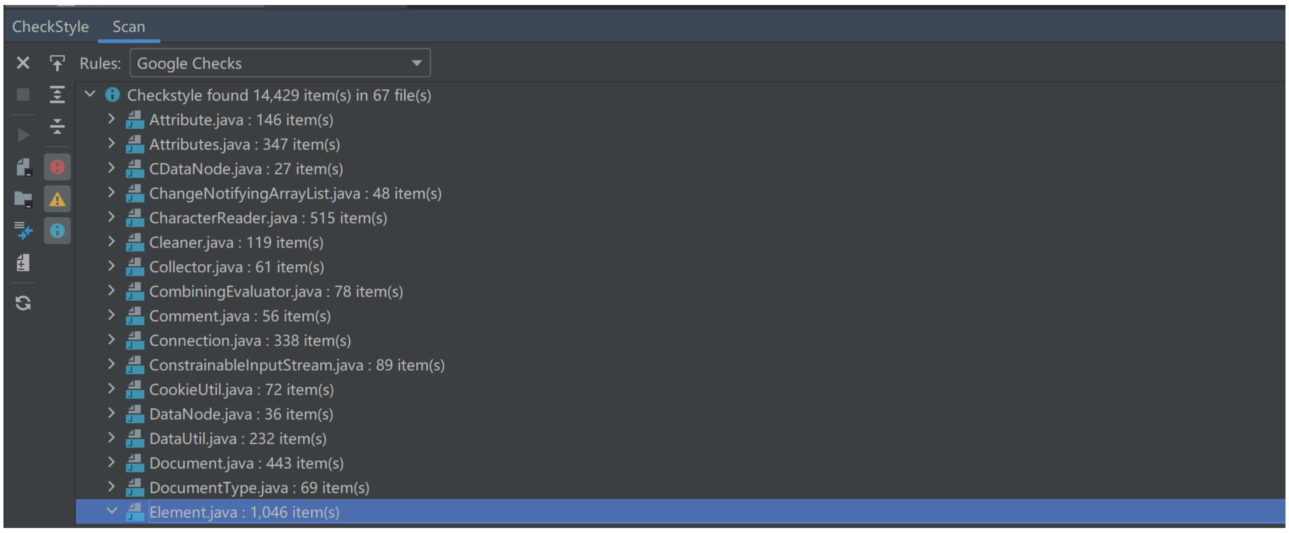
Firstly, we used the checkstyle tool. I pressed the "Check Module" button and the result is as below. This used the sun checks method.



Rules: Sun Checks

```
Checkstyle found 5,569 item(s) in 68 file(s)
  Attribute.java : 78 item(s)
  Attributes.java : 121 item(s)
  CDataNode.java : 14 item(s)
  ChangeNotifyingArrayList.java : 29 item(s)
  CharacterReader.java : 141 item(s)
  Cleaner.java : 55 item(s)
  Collector.java : 35 item(s)
  CombiningEvaluator.java : 25 item(s)
  Comment.java : 27 item(s)
    Parameter data should be final. (18:20) [FinalParameters]
    Class 'Comment' looks like designed for extension (can be subclassed), but the method 'nodeName' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Comment' f
    Missing a Javadoc comment. (22:5) [MissingJavadocMethod]
    Class 'Comment' looks like designed for extension (can be subclassed), but the method 'setData' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Comment' final
    Missing a Javadoc comment. (34:5) [MissingJavadocMethod]
    Parameter data should be final. (34:28) [FinalParameters]
    Line is longer than 80 characters (found 105). (39:0) [LineLength]
    File contains tab characters (this is the first instance). (39:1) [FileTabCharacter]
    Class 'Comment' looks like designed for extension (can be subclassed), but the method 'outerHtmlHead' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Comm
    Parameter accum should be final. (39:28) [FinalParameters]
    Parameter depth should be final. (39:46) [FinalParameters]
    Parameter out should be final. (39:57) [FinalParameters]
    Line is longer than 80 characters (found 158). (40:0) [LineLength]
    'if' construct must use '{}'s. (40:9) [NeedBraces]
    ')' is preceded with whitespace. (40:156) [ParenPad]
    Line is longer than 80 characters (found 87). (48:0) [LineLength]
    Parameter accum should be final. (48:28) [FinalParameters]
    Parameter depth should be final. (48:46) [FinalParameters]
    Parameter out should be final. (48:57) [FinalParameters]
    '{' is not followed by whitespace. (48:86) [WhitespaceAround]
    '}' is not preceded with whitespace. (48:87) [WhitespaceAround]
```

The following picture used the google checks method.



```
Checkstyle found 14,429 item(s) in 67 file(s)
  Attribute.java : 146 item(s)
  Attributes.java : 347 item(s)
  CDataNode.java : 27 item(s)
  ChangeNotifyingArrayList.java : 48 item(s)
  CharacterReader.java : 515 item(s)
  Cleaner.java : 119 item(s)
  Collector.java : 61 item(s)
  CombiningEvaluator.java : 78 item(s)
  Comment.java : 56 item(s)
  Connection.java : 338 item(s)
  ConstrainableInputStream.java : 89 item(s)
  CookieUtil.java : 72 item(s)
  DataNode.java : 36 item(s)
  DataUtil.java : 232 item(s)
  Document.java : 443 item(s)
  DocumentType.java : 69 item(s)
  Element.java : 1,046 item(s)
```

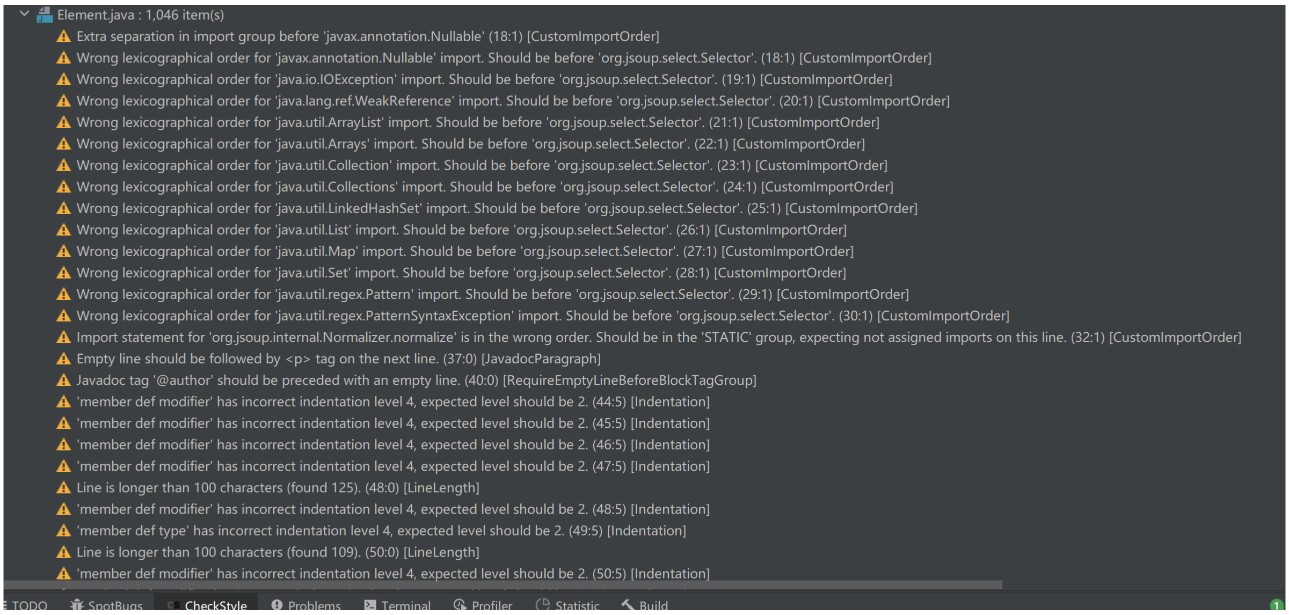
Deep dive into some errors

The first error I met with is `CustomImportError`. Actually it emphasised on the order of import, not an actual bug that you will see in the command line.

The second error I met with is `Indentation`. It used 4 tabs instead of 2 tabs.

The third error I met with is `RequiredEmptyLineBeforeBlockTagGroup`. It requires us to have an empty line before tag `@Return`.

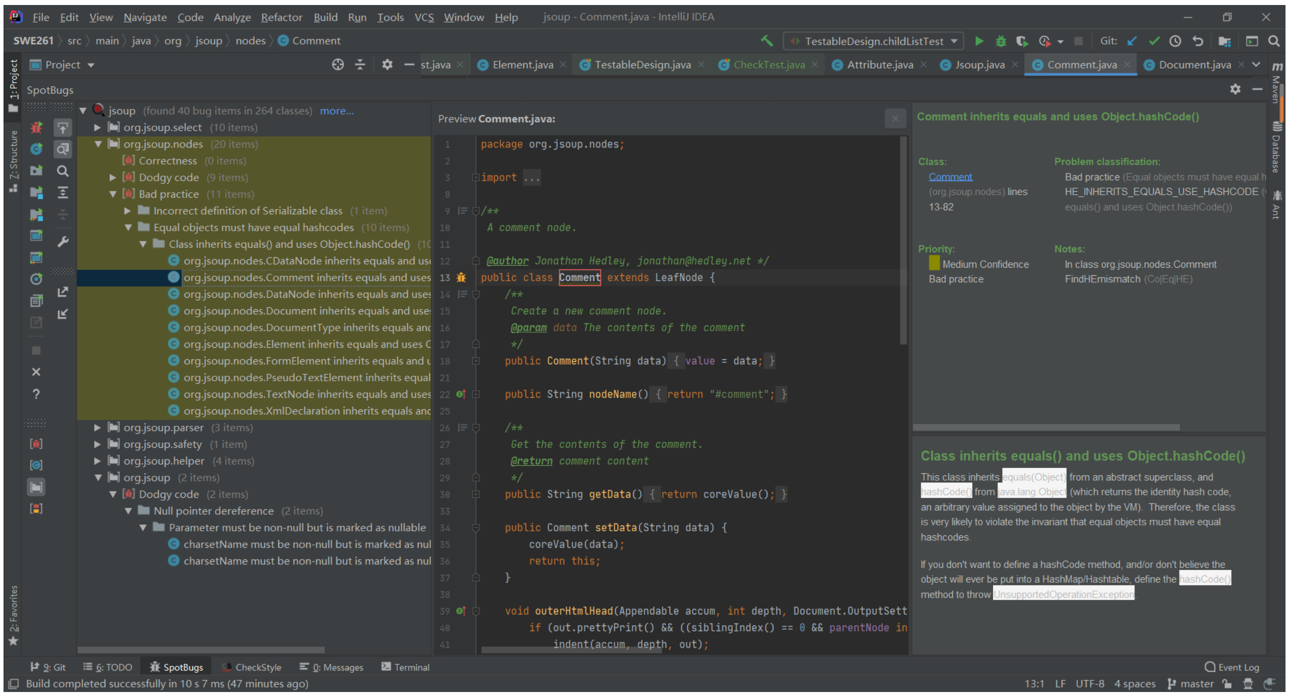
From my perspective, these errors existed, but they aren't actual problems in the code. They are more likely to be the coding style error instead of bugs.



```
Element.java : 1,046 item(s)
  Extra separation in import group before 'javax.annotation.Nullable' (18:1) [CustomImportOrder]
  Wrong lexicographical order for 'javax.annotation.Nullable' import. Should be before 'org.jsoup.select.Selector'. (18:1) [CustomImportOrder]
  Wrong lexicographical order for 'java.io.IOException' import. Should be before 'org.jsoup.select.Selector'. (19:1) [CustomImportOrder]
  Wrong lexicographical order for 'java.lang.ref.WeakReference' import. Should be before 'org.jsoup.select.Selector'. (20:1) [CustomImportOrder]
  Wrong lexicographical order for 'java.util.ArrayList' import. Should be before 'org.jsoup.select.Selector'. (21:1) [CustomImportOrder]
  Wrong lexicographical order for 'java.util.Arrays' import. Should be before 'org.jsoup.select.Selector'. (22:1) [CustomImportOrder]
  Wrong lexicographical order for 'java.util.Collection' import. Should be before 'org.jsoup.select.Selector'. (23:1) [CustomImportOrder]
  Wrong lexicographical order for 'java.util.Collections' import. Should be before 'org.jsoup.select.Selector'. (24:1) [CustomImportOrder]
  Wrong lexicographical order for 'java.util.LinkedHashSet' import. Should be before 'org.jsoup.select.Selector'. (25:1) [CustomImportOrder]
  Wrong lexicographical order for 'java.util.List' import. Should be before 'org.jsoup.select.Selector'. (26:1) [CustomImportOrder]
  Wrong lexicographical order for 'java.util.Map' import. Should be before 'org.jsoup.select.Selector'. (27:1) [CustomImportOrder]
  Wrong lexicographical order for 'java.util.Set' import. Should be before 'org.jsoup.select.Selector'. (28:1) [CustomImportOrder]
  Wrong lexicographical order for 'java.util.regex.Pattern' import. Should be before 'org.jsoup.select.Selector'. (29:1) [CustomImportOrder]
  Wrong lexicographical order for 'java.util.regex.PatternSyntaxException' import. Should be before 'org.jsoup.select.Selector'. (30:1) [CustomImportOrder]
  Import statement for 'org.jsoup.internal.Normalizer.normalize' is in the wrong order. Should be in the 'STATIC' group, expecting not assigned imports on this line. (32:1) [CustomImportOrder]
  Empty line should be followed by <p> tag on the next line. (37:0) [JavadocParagraph]
  Javadoc tag '@author' should be preceded with an empty line. (40:0) [RequireEmptyLineBeforeBlockTagGroup]
  'member def modifier' has incorrect indentation level 4, expected level should be 2. (44:5) [Indentation]
  'member def modifier' has incorrect indentation level 4, expected level should be 2. (45:5) [Indentation]
  'member def modifier' has incorrect indentation level 4, expected level should be 2. (46:5) [Indentation]
  'member def modifier' has incorrect indentation level 4, expected level should be 2. (47:5) [Indentation]
  Line is longer than 100 characters (found 125). (48:0) [LineLength]
  'member def modifier' has incorrect indentation level 4, expected level should be 2. (48:5) [Indentation]
  'member def type' has incorrect indentation level 4, expected level should be 2. (49:5) [Indentation]
  Line is longer than 100 characters (found 109). (50:0) [LineLength]
  'member def modifier' has incorrect indentation level 4, expected level should be 2. (50:5) [Indentation]
```

Spotbugs

Next, we used the spotbugs tool. Different from the checkstyle tool, it is divided into folders and each java file, and it will spotlights the lines and functions or classes that has some type error or bugs.


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
jsoup - Comment.java - IntelliJ IDEA
SWE261 〉 src 〉 main 〉 java 〉 org 〉 jsoup 〉 nodes 〉 Comment
TestableDesign.childListTest
Git:
Project
st.java Element.java TestableDesign.java CheckTest.java Attribute.java Jsoup.java Comment.java Document.java
1: Project
7: Structure
2: Favorites
Maven
Database
Ant
9: Git 6: TODO SpotBugs CheckStyle 0: Messages Terminal
Event Log
13:1 LF UTF-8 4 spaces master

```
SpotBugs
▼ jsoup (found 40 bug items in 264 classes) more...
  ▶ org.jsoup.select (10 items)
  ▼ org.jsoup.nodes (20 items)
      Correctness (0 items)
    ▶ Dodgy code (9 items)
    ▼ Bad practice (11 items)
      ▶ Incorrect definition of Serializable class (1 item)
      ▼ Equal objects must have equal hashcodes (10 items)
        ▼ Class inherits equals() and uses Object.hashCode() (10
            org.jsoup.nodes.CDataNode inherits equals and use
            org.jsoup.nodes.Comment inherits equals and uses
            org.jsoup.nodes.DataNode inherits equals and uses
            org.jsoup.nodes.Document inherits equals and use
            org.jsoup.nodes.DocumentType inherits equals and
            org.jsoup.nodes.Element inherits equals and uses C
            org.jsoup.nodes.FormElement inherits equals and u
            org.jsoup.nodes.PseudoTextElement inherits equal
            org.jsoup.nodes.TextNode inherits equals and uses
            org.jsoup.nodes.XmlDeclaration inherits equals and
  ▶ org.jsoup.parser (3 items)
  ▶ org.jsoup.safety (1 item)
  ▶ org.jsoup.helper (4 items)
  ▼ org.jsoup (2 items)
    ▼ Dodgy code (2 items)
      ▼ Null pointer dereference (2 items)
        ▼ Parameter must be non-null but is marked as nullable
            charsetName must be non-null but is marked as nul
            charsetName must be non-null but is marked as nul
```

Preview **Comment.java:**

```java
package org.jsoup.nodes;

import ...

/**
 A comment node.

 @author Jonathan Hedley, jonathan@hedley.net */
public class Comment extends LeafNode {
    /**
     Create a new comment node.
     @param data The contents of the comment
     */
    public Comment(String data) { value = data; }

    public String nodeName() { return "#comment"; }

    /**
     Get the contents of the comment.
     @return comment content
     */
    public String getData() { return coreValue(); }

    public Comment setData(String data) {
        coreValue(data);
        return this;
    }

    void outerHtmlHead(Appendable accum, int depth, Document.OutputSett
        if (out.prettyPrint() && ((siblingIndex() == 0 && parentNode in
            indent(accum, depth, out);
```

Comment inherits equals and uses Object.hashCode()

Class:
Comment (org.jsoup.nodes) lines 13-82

Problem classification:
Bad practice (Equal objects must have equal h
HE_INHERITS_EQUALS_USE_HASHCODE (
equals() and uses Object.hashCode())

Priority:
Medium Confidence
Bad practice

Notes:
In class org.jsoup.nodes.Comment
FindHEmismatch (Co|Eq|HE)

Class inherits equals() and uses Object.hashCode()

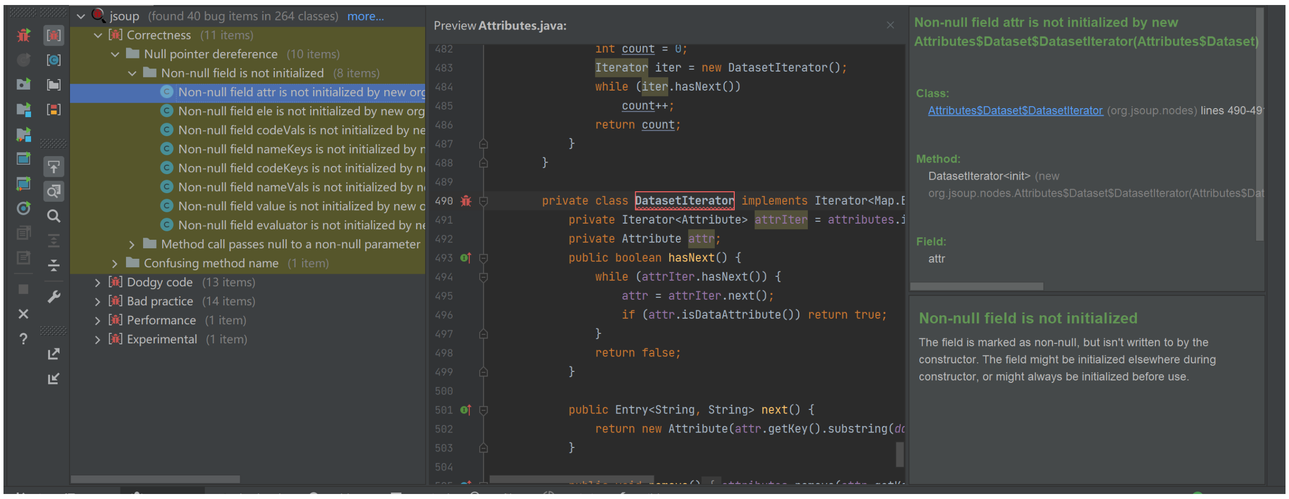
This class inherits equals(Object) from an abstract superclass, and hashCode() from java.lang.Object (which returns the identity hash code, an arbitrary value assigned to the object by the VM). Therefore, the class is very likely to violate the invariant that equal objects must have equal hashcodes.

If you don't want to define a hashCode method, and/or don't believe the object will ever be put into a HashMap/Hashtable, define the hashCode() method to throw UnsupportedOperationException.

Build completed successfully in 10 s 7 ms (47 minutes ago)

Deep dive into some errors

The first error in **correctness** is that non-null field is not initialized. From my perspective, it is mentioning that `JSoup` didn't initialize the parameter for the class `DatasetIterator`



```
▼ jsoup (found 40 bug items in 264 classes) more...
  ▼ Correctness (11 items)
    ▼ Null pointer dereference (10 items)
      ▼ Non-null field is not initialized (8 items)
          Non-null field attr is not initialized by new org
          Non-null field ele is not initialized by new org
          Non-null field codeVals is not initialized by ne
          Non-null field nameKeys is not initialized by n
          Non-null field codeKeys is not initialized by n
          Non-null field nameVals is not initialized by n
          Non-null field value is not initialized by new o
          Non-null field evaluator is not initialized by ne
      ▶ Method call passes null to a non-null parameter
    ▶ Confusing method name (1 item)
  ▶ Dodgy code (13 items)
  ▶ Bad practice (14 items)
  ▶ Performance (1 item)
  ▶ Experimental (1 item)
```

Preview **Attributes.java:**

```java
                int count = 0;
                Iterator iter = new DatasetIterator();
                while (iter.hasNext())
                    count++;
                return count;
            }
        }

        private class DatasetIterator implements Iterator<Map.E
            private Iterator<Attribute> attrIter = attributes.i
            private Attribute attr;
            public boolean hasNext() {
                while (attrIter.hasNext()) {
                    attr = attrIter.next();
                    if (attr.isDataAttribute()) return true;
                }
                return false;
            }

            public Entry<String, String> next() {
                return new Attribute(attr.getKey().substring(dc
            }
```

Non-null field attr is not initialized by new Attributes$Dataset$DatasetIterator(Attributes$Dataset)

Class:
Attributes$Dataset$DatasetIterator (org.jsoup.nodes) lines 490-49

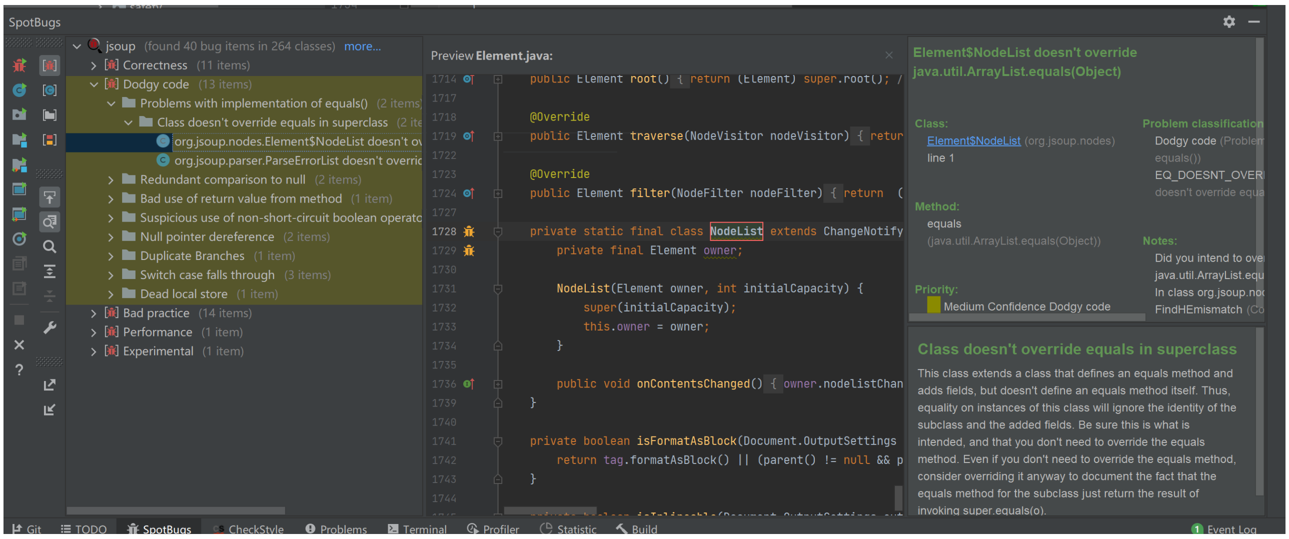
Method:
DatasetIterator<init> (new
org.jsoup.nodes.Attributes$Dataset$DatasetIterator(Attributes$Dat

Field:
attr

Non-null field is not initialized

The field is marked as non-null, but isn't written to by the constructor. The field might be initialized elsewhere during constructor, or might always be initialized before use.

The second error in **dodgy code** is that classes doesn't override equals in superclass. Because it used `extend`, and it should override `equals` function. But as long as it didn't use it, it should be fine that this class doesn't override equals in superclass.



SpotBugs

jsoup (found 40 bug items in 264 classes) more...
- Correctness (11 items)
- Dodgy code (13 items)
  - Problems with implementation of equals() (2 items)
    - Class doesn't override equals in superclass (2 ite
      - org.jsoup.nodes.Element$NodeList doesn't ov
      - org.jsoup.parser.ParseErrorList doesn't overric
  - Redundant comparison to null (2 items)
  - Bad use of return value from method (1 item)
  - Suspicious use of non-short-circuit boolean operato
  - Null pointer dereference (2 items)
  - Duplicate Branches (1 item)
  - Switch case falls through (3 items)
  - Dead local store (1 item)
- Bad practice (14 items)
- Performance (1 item)
- Experimental (1 item)

Preview **Element.java:**

```java
    public Element root() { return (Element) super.root(); /

    @Override
    public Element traverse(NodeVisitor nodeVisitor) { retur

    @Override
    public Element filter(NodeFilter nodeFilter) { return  (

    private static final class NodeList extends ChangeNotify
        private final Element owner;

        NodeList(Element owner, int initialCapacity) {
            super(initialCapacity);
            this.owner = owner;
        }

        public void onContentsChanged() { owner.nodelistChan
    }

    private boolean isFormatAsBlock(Document.OutputSettings
        return tag.formatAsBlock() || (parent() != null && p
    }
```

Element$NodeList doesn't override java.util.ArrayList.equals(Object)

Class:
Element$NodeList (org.jsoup.nodes)
line 1

Method:
equals
(java.util.ArrayList.equals(Object))

Priority:
🟨 Medium Confidence Dodgy code

Problem classification
Dodgy code (Problem
equals())
EQ_DOESNT_OVER
doesn't override equa

Notes:
Did you intend to ove
java.util.ArrayList.equ
In class org.jsoup.no
FindHEmismatch (Co

Class doesn't override equals in superclass

This class extends a class that defines an equals method and adds fields, but doesn't define an equals method itself. Thus, equality on instances of this class will ignore the identity of the subclass and the added fields. Be sure this is what is intended, and that you don't need to override the equals method. Even if you don't need to override the equals method, consider overriding it anyway to document the fact that the equals method for the subclass just return the result of invoking super.equals(o).

Contrast between Spotbugs and Checkstyle

The information for each tool is quite different.

For spotbugs, this one is more neat and clear with description and code. Every error is categorized into five folders.

- Correctness
- Dodgy code
- Bad practice
- Performance
- Experimental

For checkstyle, this one is more messy. Every error message is put under its original file name. Once you press the error information, you will jump into the error line. The error most frequently appeared are

- CustomImportError
- LineLength(too long)
- Indentation
- RequiredEmptyLineBeforeBlockTagGroup

They are more likely to be the coding style error instead of bugs.