Flint

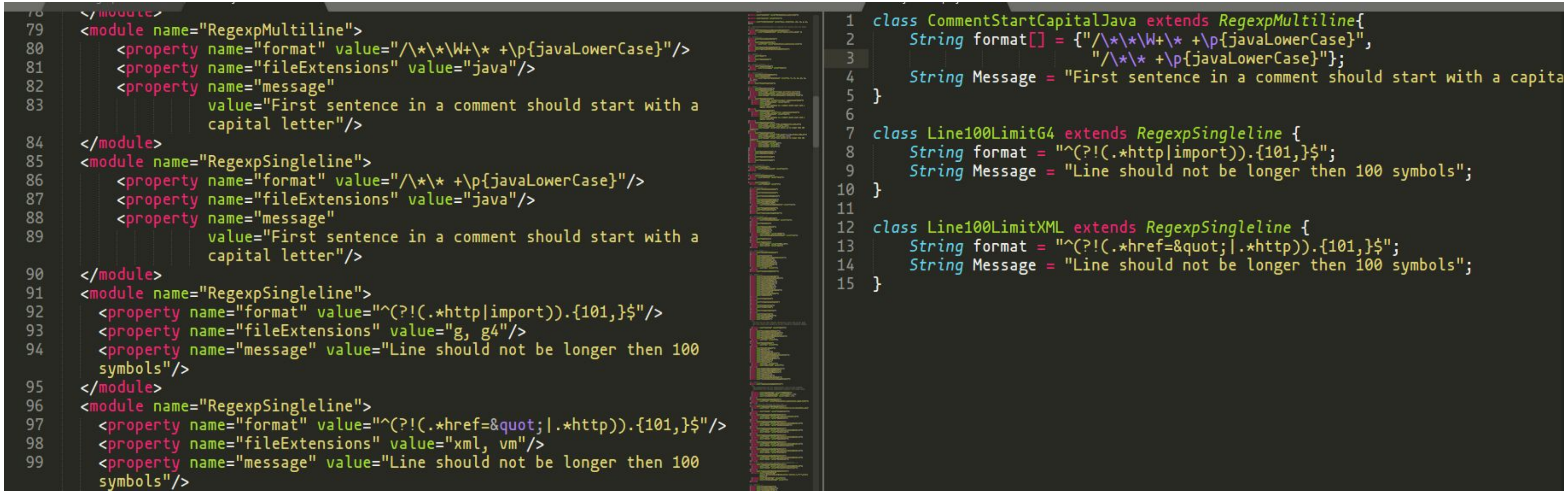
A Programmable Style and Documentation Linter for Java

Ofek Inbar, Daniel Wang, Elliott de Bruin, Jessica Chen

Motivation

- ➔ Developers want tools that are easy to learn and begin using in a short amount of time
- ➔ Using Java for configuration gives the user more control than using XML
  - ◆ Option to run resource intensive checks under defined conditions
- ➔ Developers want to be able to create and develop clean and readable codebases without needing to manually check their code against their style guide

Motivation



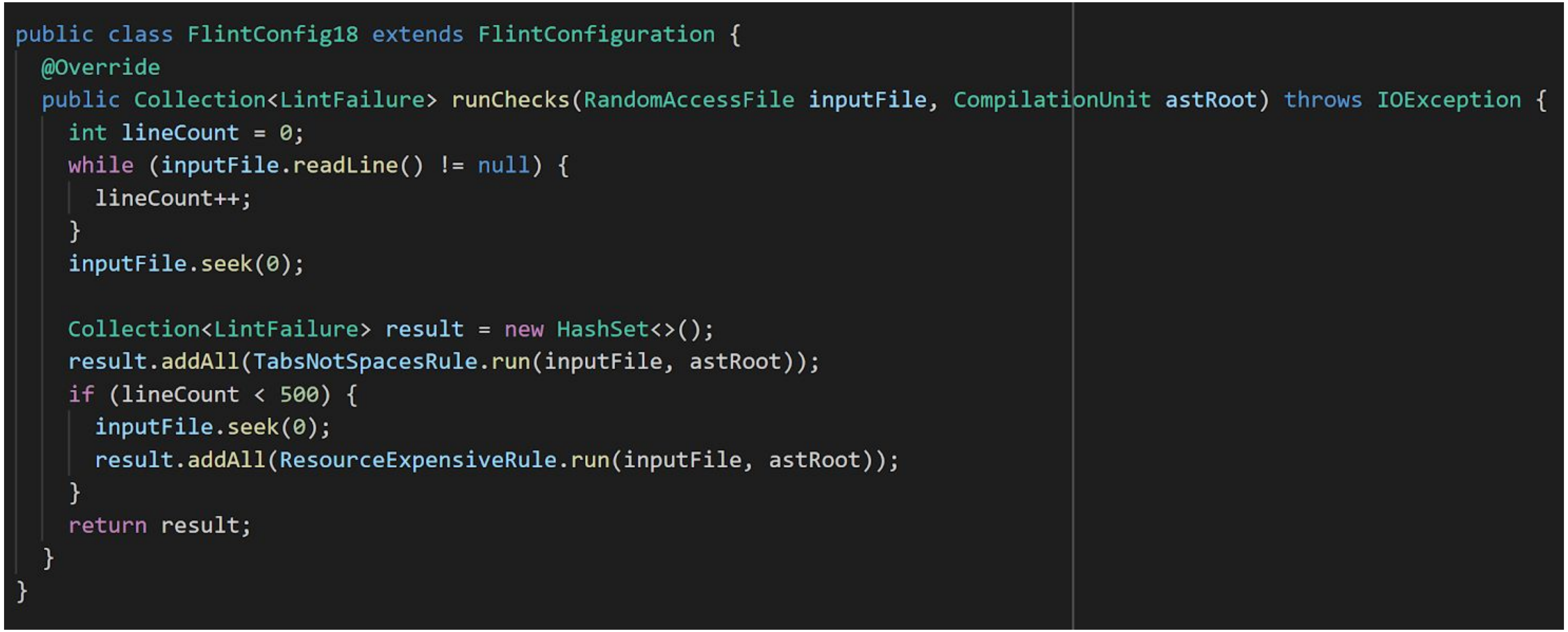
```
</module>
<module name="RegexpMultiline">
    <property name="format" value="/\*\*\W+\* +\p{javaLowerCase}"/>
    <property name="fileExtensions" value="java"/>
    <property name="message"
              value="First sentence in a comment should start with a
              capital letter"/>
</module>
<module name="RegexpSingleline">
    <property name="format" value="/\*\* +\p{javaLowerCase}"/>
    <property name="fileExtensions" value="java"/>
    <property name="message"
              value="First sentence in a comment should start with a
              capital letter"/>
</module>
<module name="RegexpSingleline">
  <property name="format" value="^(?!(.*http|import)).{101,}$"/>
  <property name="fileExtensions" value="g, g4"/>
  <property name="message" value="Line should not be longer then 100
  symbols"/>
</module>
<module name="RegexpSingleline">
  <property name="format" value="^(?!(.*href=&quot;|.*http)).{101,}$"/>
  <property name="fileExtensions" value="xml, vm"/>
  <property name="message" value="Line should not be longer then 100
  symbols"/>
```

```
class CommentStartCapitalJava extends RegexpMultiline{
    String format[] = {"/\*\*\W+\* +\p{javaLowerCase}",
                        "/\*\* +\p{javaLowerCase}"};
    String Message = "First sentence in a comment should start with a capita
}

class Line100LimitG4 extends RegexpSingleline {
    String format = "^(?!(.*http|import)).{101,}$";
    String Message = "Line should not be longer then 100 symbols";
}

class Line100LimitXML extends RegexpSingleline {
    String format = "^(?!(.*href=&quot;|.*http)).{101,}$";
    String Message = "Line should not be longer then 100 symbols";
}
```

Approach

- ➔ One configuration file defines the linter’s behavior
  - ◆ One file is easy for users to understand, makes setup easier
  - ◆ Configuration is written in Java, this gives the user total control over the linter’s behavior, from defining which rules are checked in what order and under what conditions
- ➔ Offer both Command Line Interface and IDE extension
  - ◆ Command line is useful for people who don’t use IDE’s, and provides more control over when to lint
  - ◆ IDE extension speeds up workflow (don’t need to switch applications or contexts to run linter)

Approach

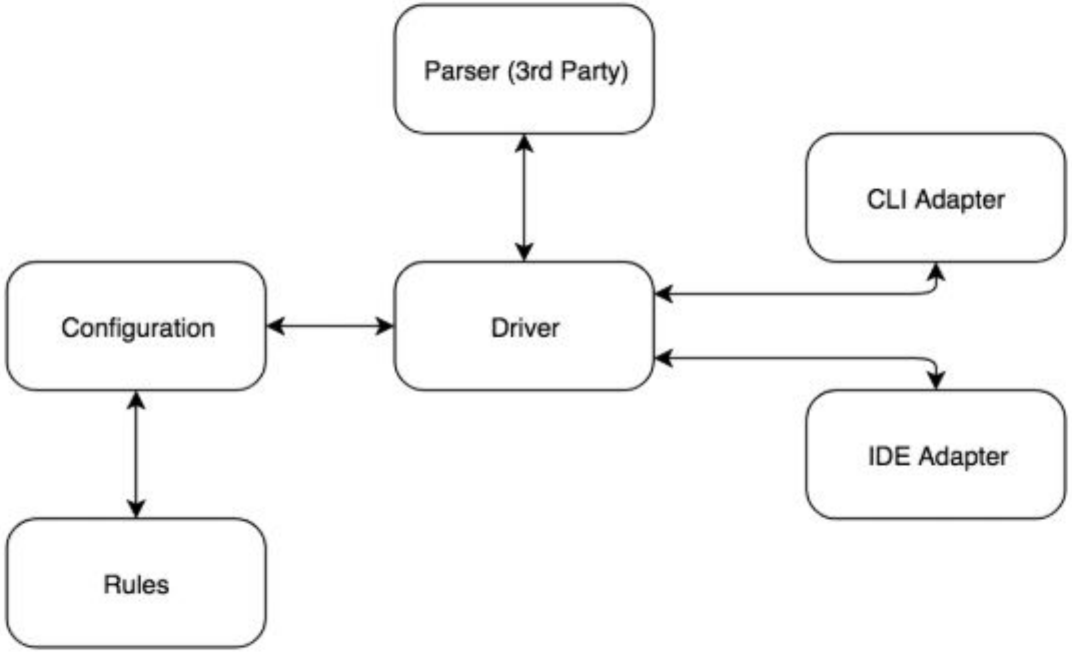


```java
public class FlintConfig18 extends FlintConfiguration {
  @Override
  public Collection<LintFailure> runChecks(RandomAccessFile inputFile, CompilationUnit astRoot) throws IOException {
    int lineCount = 0;
    while (inputFile.readLine() != null) {
      lineCount++;
    }
    inputFile.seek(0);

    Collection<LintFailure> result = new HashSet<>();
    result.addAll(TabsNotSpacesRule.run(inputFile, astRoot));
    if (lineCount < 500) {
      inputFile.seek(0);
      result.addAll(ResourceExpensiveRule.run(inputFile, astRoot));
    }
    return result;
  }
}
```

Research Questions

- ➔ Tools that bring milder learning curve
  - ◆ e.g. Java vs. XML, various linter definitions
- ➔ Functionality of existing tools
  - ◆ How people love and hate about them?
- ➔ What developers would like to get out of program analysis?
- ➔ XML and Java Trade-offs

Preliminary Results



Preliminary Results

- ➔ CLI Version Done
  - ◆ CLI Adapter Implemented and Tested
- ➔ Driver being able to connect rules and CLI Adapter
- ➔ CI tools are powerful and unexpectedly easy to use!

Related Tools

- Checkstyle
  - Able to customize style checks
  - Configuration must be done in XML
  - Comes with pre-written checks
- Google-java-format
  - Style checker for Google’s style guide
  - Not easily customizable
  - Able to automatically fix errors
- Error Prone
  - Hooks into your applications compile step
  - Catches common programming mistakes at runtime and provides suggested fixes
- Uncrustify
  - Code formatter with 655 configurable options
  - Does not have rule customization support

Works Cited

"Checkstyle – Checkstyle 8.17." *Checkstyle*, 27 Jan. 2019, checkstyle.sourceforge.net/.

"google/google-java-format: Reformats Java source code to comply with Google Java style." *Google*, 10 Jan. 2019, github.com/google/google-java-format/.

Maria Christakis , Christian Bird, What developers want and need from program analysis: an empirical study, Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, September 03-07, 2016, Singapore, Singapore [doi>10.1145/2970276.2970347]

"Uncrustify: Source Code Beautifier for C, C++, C#, ObjectiveC, D, Java, Pawn and VALA." *Uncrusify*. Feb. 2019, http://uncrustify.sourceforge.net/

"Error Prone." *Google,* Feb 2019, https://errorprone.info/