Flint

A customizable style linter for Java

●●●

Elliott de Bruin, Ofek Inbar

Motivation

- Developing and maintaining a clean and readable codebase is difficult for teams with many contributors
- Style guides can help keep the codebase readable (and uniform)
- Linters can help enforce style rules, either as command line tools or ide plugins/extensions
- CheckStyle is a good example of a Java linter:
    - Pros:
        - Comes with many pre-written style rules
        - Supports customizing which rules to check for a project
        - Supports writing custom rules
    - Cons:
        - Can only be run as either an Ant task or as a CLI (inferior to the red squiggly lines most IDEs display for errors on every keystroke or file save)
        - Configuration files are written in XML, a Java programmer shouldn't need to learn XML to use a Java linter

Approach

- Build a linter that supports customizing which style rules to apply and also supports writing custom style rules
  - Offer the linter as an IDE plugin/extension that can run on every file-save/keystroke and very quickly informs the user of code locations where the rules are not met
  - Rule configuration/custom rule creation will be defined completely in Java code
- Possible implementation of a configuration system:

```java
package demo;

import AbstractConfiguration;
import CheckRunner;

@Configuration
public class CustomConfiguration extends AbstractConfiguration {
  @Override
  runChecks(CheckRunner runner) {
    TabsNotSpacesRule.runCheck(runner);
    NoTrailingWhitespaceRule.runCheck(runner);
    LineLimitRule.runCheck(runner, 100);
  }
}
```