Proposal for GSoC 2017

Regression Testing Tool and HTML Report Generator for Pull Request

About Me

BASIC INFORMATION

Name: Liangchen Luo
Email: luolc.witty@gmail.com
GitHub Account: Luolc
Website: http://www.luolc.com

EDUCATION

Peking University, Beijing, China
Geographic Information System Candidate, expected graduation July 2019

ISSUES FIXED

* View the merged PRs on GitHub.

The following issues have been fixed (by March 31):

- **#3172,** Regression false-positive FinalLocalVariable PR#4060
- **#4003,** Indentation UTs should not use ROOT locale when they test violation/error message PR#4020
- **#3989,** UTs should not use ROOT locale when they test violation/error message PR#3993
- **#3965,** remove from Input files "Compilable with Java8" PR#3968
- **#3961,** DetailASTTest: 'checkTree' failing on deep AST tree PR#3966
- **#3896,** Test failed due to locale message settings. (with non-English locale settings) PR#3942
- **#3700,** Control Characters are not skipped with google_checks config #PR3894
- **#3731,** expand documentation on METHOD_REF token PR#3884

RELATED EXPERIENCE

- Two-year Android development experience, proficient in Java; familiar with RxJava, MVP structure and other common Android/Java libraries, and Gradle/Maven for package dependencies management.
- Followed the principles of Clean Code and TDD in daily programming.
- Teaching Assistant of course *Algorithm and Data Structure* in 2016.
- Participant in MCM (The Mathematical Contest in Modeling) in 2017.

Project

Project Name

Regression Testing Tool and HTML Report Generator for Pull Request

Project Description

Checkstyle GSoC 2017 Project Ideas#Regression Testing Tool and HTML Report Generator for Pull Request

Restatement

We need to create an automation tool to do regression testing based on proposed patch (Pull Request). To complete that, we are required to achieve the following goals:

1. Fetch PR information on Travis CI.
2. Generate configurations based on Git changes.
3. Generate diff report between the base and the patch.
4. Deploy the report to the public.

Outline

1. Fetch PR information on Travis CI

   When Travis CI is triggered by a PR, a global variable `$TRAVIS_PULL_REQUEST` will be set. With the PR id (`$TRAVIS_PULL_REQUEST`) we could get all the information about the patch by cloning the base/fork repositories and running Git command locally. Then we could get the file changes which are used to generate configurations in the next step.

2. Generate configurations based on Git changes

   After getting the file changes, we need to filter them by the file name at First. Only `*Check.java` should be taken into consideration. It is possible that a utility class that specific checks rely on is changed, which thereby indirectly affects those checks. We will skip such cases for first implementation of the generator and think about it later.

   Firstly, we could generate configurations with no specific options (using default settings), and that is easy. Other way is pre-setting some options for each check in advance or generate configurations with all possible combinations of the options when the amount of properties is less than a certain number. When a new property is introduced, we could make diff report with/without that property; when changed lines are related to a certain property, it would require a regression (it is not an easy task to detect which property to do regression, and we should make it step by step).

   Moreover, from changes of UTs we could take values of properties that are required for regression. As certain UT is changed, it means that behavior for that property values are changed and to do regression on values that UT specify is required. Therefore, we could find the changed lines of corresponding UTs, and then find the config setting code, which could be used to help us generate the configurations. It might be not easy, but I think it is possible and we could have a try.

3. Generate diff report between the base and the patch

   When generating the configurations successfully, we could moving on to the diff report generation. The checkstyle-tester tool should be reuse. We would clone the repository on https://github.com/checkstyle/contribution, also Checkstyle master and the patch repository. Then running the diff report tool to generate the HTML report.

4. Deploy the report to the public
   Finally, the output report should be deployed to the public to let developers and reviewers to browse. Travis CI has a GitHub Pages Deployment service which could help. We could have a try to judge whether it could cover our needs. Last but not least, since the diff report might be space consuming, we need to determine a strategy to remove old/trash reports.

Expected Timeline

Date	Work
Prior - May 4	• Get familiar with Travis CI's function. • Keep diving into Checkstyle's code by fixing issues.
May 4 - 30	• Investigate Git command specification and find a proper way to get the file changes information we need. • Implement the configurations generator preliminarily (using only default options).
June 1 - 30	• Improve the generator (using pre-setting properties and all possible combinations of options). • Improve the generator (generating based on UT changes). • Write a document of evaluation.
July 1 - 28	• Write a shell script to generate diff report (I have already written one and it is being used in my development currently). • Try the GitHub Pages Deployment service of Travis CI. If it is not appropriate, find other deployment solutions. • Determine the strategy of removing old/trash reports. • Write a document of evaluation.
July 29 - August 29	• Update the README of Checkstyle/contribution repository. • Write a summary article throughout the project. • Buffer for unexpected delay.

Extra Information

Working Time

I will be based in Beijing, China during the summer. Therefore, I will be working in GMT +8 timezone. I believe it will take about 10 weeks for me to complete the project. But before student projects will be announced in early May, I can have a head start with some early preparation.

Reason for Participation

I have always wanted to be a great developer since the first day I learnt programming. GSoC provides me a chance to make contributions to open source projects with mentorship from

great developers all over the world. I believe it is really amazing. If I have the chance to participate in GSoC and work with Checkstyle, I will try my best to complete this project.

I have read Martin's *Clean Code* before and know the significance of a good coding style, and I have been using the Checkstyle Gradle plugin to improve the quality of my own code for half a year. In the past few weeks, I have fixed several issues in Checkstyle issue tracker. By now I am familiar with the contribution workflow and how to cooperate with other developers and project maintainers. Besides, since I will continue using Checkstyle in my own projects in the future, I would like to keep maintaining and improving this feature after the program ends.

I am looking forward to working on the project wich Checkstyle!