CheckStyle Structural Metrics Plugin

Shelled Pistachios

About the Plugin

This plugin contains CheckStyle checks that can be used to see the Structural Metrics of a Java program. The Structural Metric checks are:

- CastCounter – Counts the number of casts in each class
- CommentCounter – Counts the number of comments and the number of lines of comments in each class
- ExpressionCounter – Counts the number of expressions, operators, operands, unique operators and unique operands in each class
- HalsteadMetrics – Calculates the Halstead Length, the Halstead Vocabulary, the Halstead Volume, the Halstead Difficulty, and the HalsteadEffort of each class
- LoopingStatementCounter – Counts the number of loop statements in each class
- MethodCounter – Counts the number of methods in each class
- VariableCounter – Counts the number of variables in each class

How to Install and Run the Plugin

Note: Some of these instructions were taken from the Set Up for CheckStyle Development guide posted on Blackboard

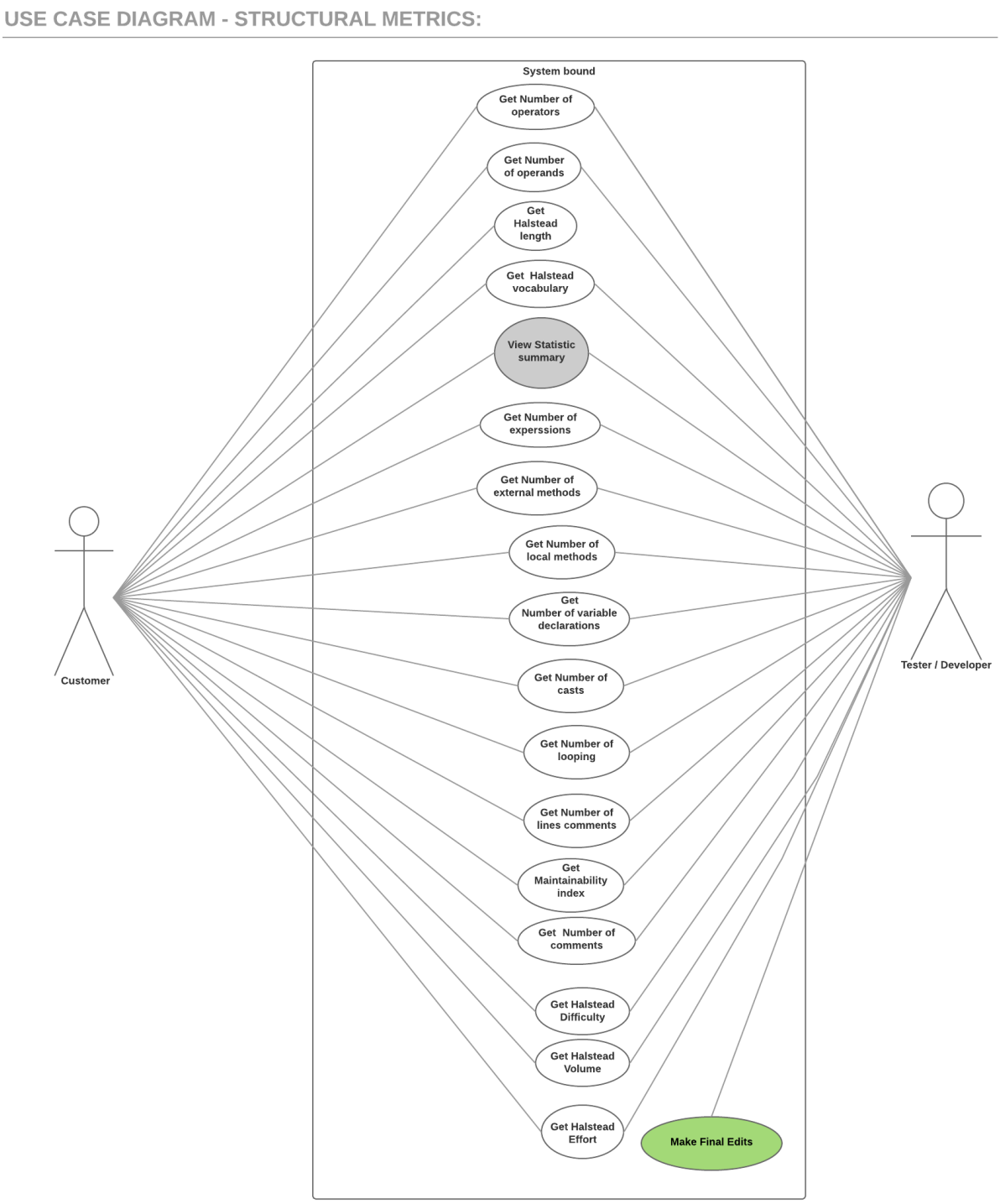
1. Download Eclipse for RCP and RAP developers
2. Install CheckStyle in Eclipse
3. Install Maven
4. Clone the CheckStyle Structural Metrics project
    a. git clone https://github.com/rhodeskl/CPTS422.git
5. Import the Structural Metrics project into Eclipse
    a. Open Eclipse
    b. Click on Import → General → Existing Project into Workspace
    c. Click on Browse, and go to where you cloned the Structural Metrics project
    d. Select the root directory of the Structural Metrics project, the CPTS422 directory
    e. Select all projects
    f. Click on “Copy into Workspace”
    g. Click on “Finish”
6. On the command line, go to the CPTS422 directory and run the following commands:
    a. cd eclipse-cs
    b. mvn clean package
    c. cd ..
    d. cd net.sf.eclipsecs.sample
    e. mvn clean package

    You should get “Build Success” on both of the mvn clean package commands

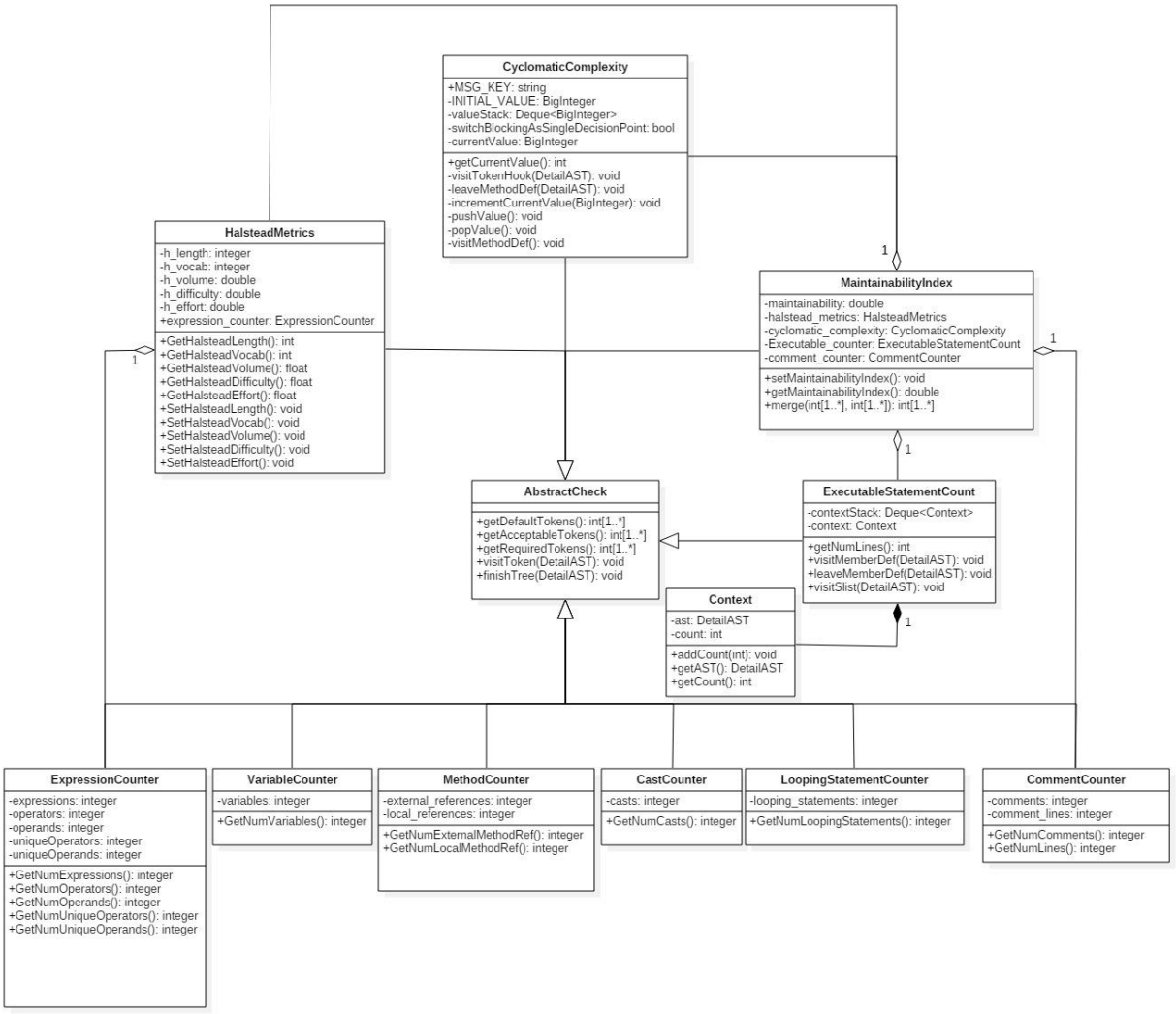
7. Go back to Eclipse, right click on the eclipse-cs directory and click on Run As → Maven Install. This should display a build success once it is done running
8. Do the same for the net.sf.eclipsecs.sample directory
9. Right click on the net.sf.eclipsecs.sample directory and click on Run As → Eclipse Application. A new Eclipse window should come up.
10. In the new Eclipse window, either create a new Java project and add a few classes to run the Structural Metrics CheckStyle checks on or import an existing Java project into the workspace
11. Switch to the Structural Metrics Checkstyle Plugin in the new Eclipse window
    a. Go to Window → Preferences → CheckStyle
    b. Click on Sample Builtin Checks. It should be highlighted in grey after you click on it, then click on "Set as Default"
    c. Double click on Sample Builtin Checks
    d. Click on My custom checks
    e. Click on the checks you want to run to enable them
    f. Click ok
12. In the new Eclipse window, go to Window → Show View → Other
    a. Go to CheckStyle and enable the Checkstyle violations and Checkstyle violations chart
13. Go to a java file or a file containing java files that has been imported into the new Eclipse window, right click on it, and click CheckStyle → Check code with CheckStyle
14. Look at the Checkstyle violations window. The Structural Metrics that are found by the checks you selected in step 11 should be shown. Each check runs for each class in the project that was imported or created in the new Eclipse instance. To view which class a violation is referring to, double click on a violation. To go back to the list of all metrics, click on the back-arrow button in the CheckStyle violations window.

Diagrams

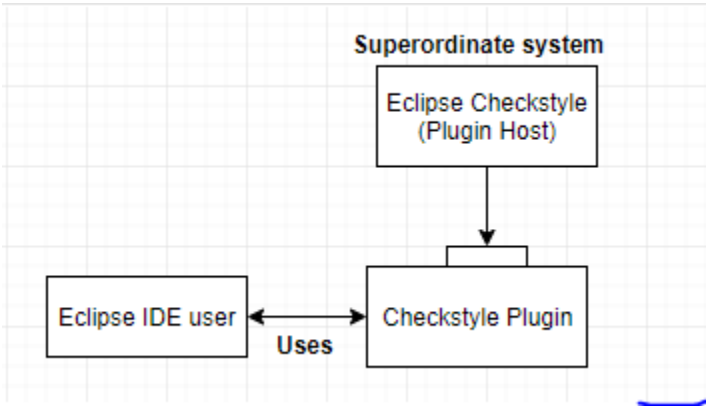
Use Case Diagram



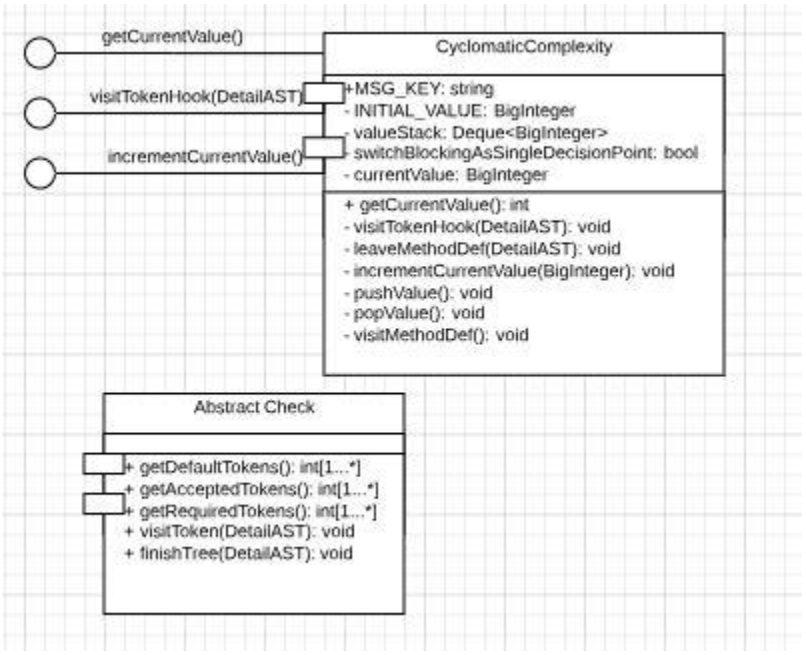
Class Diagram

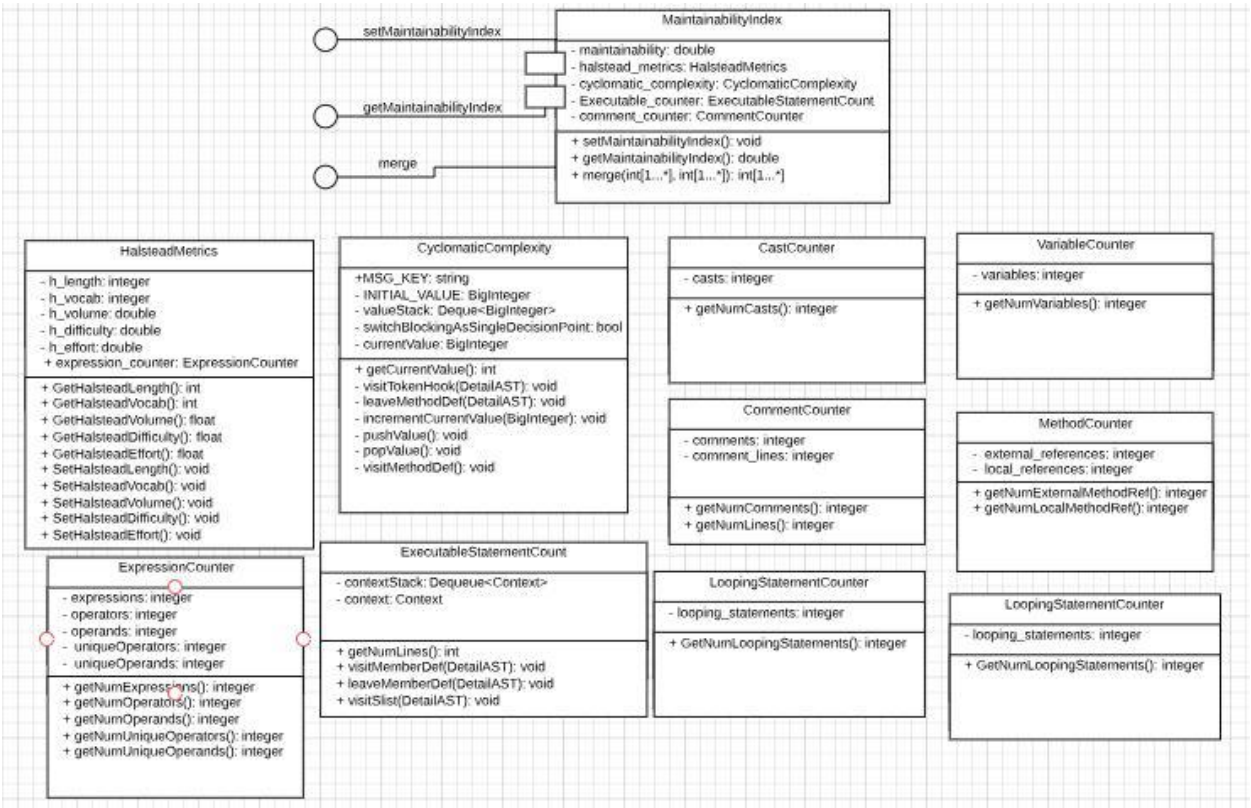


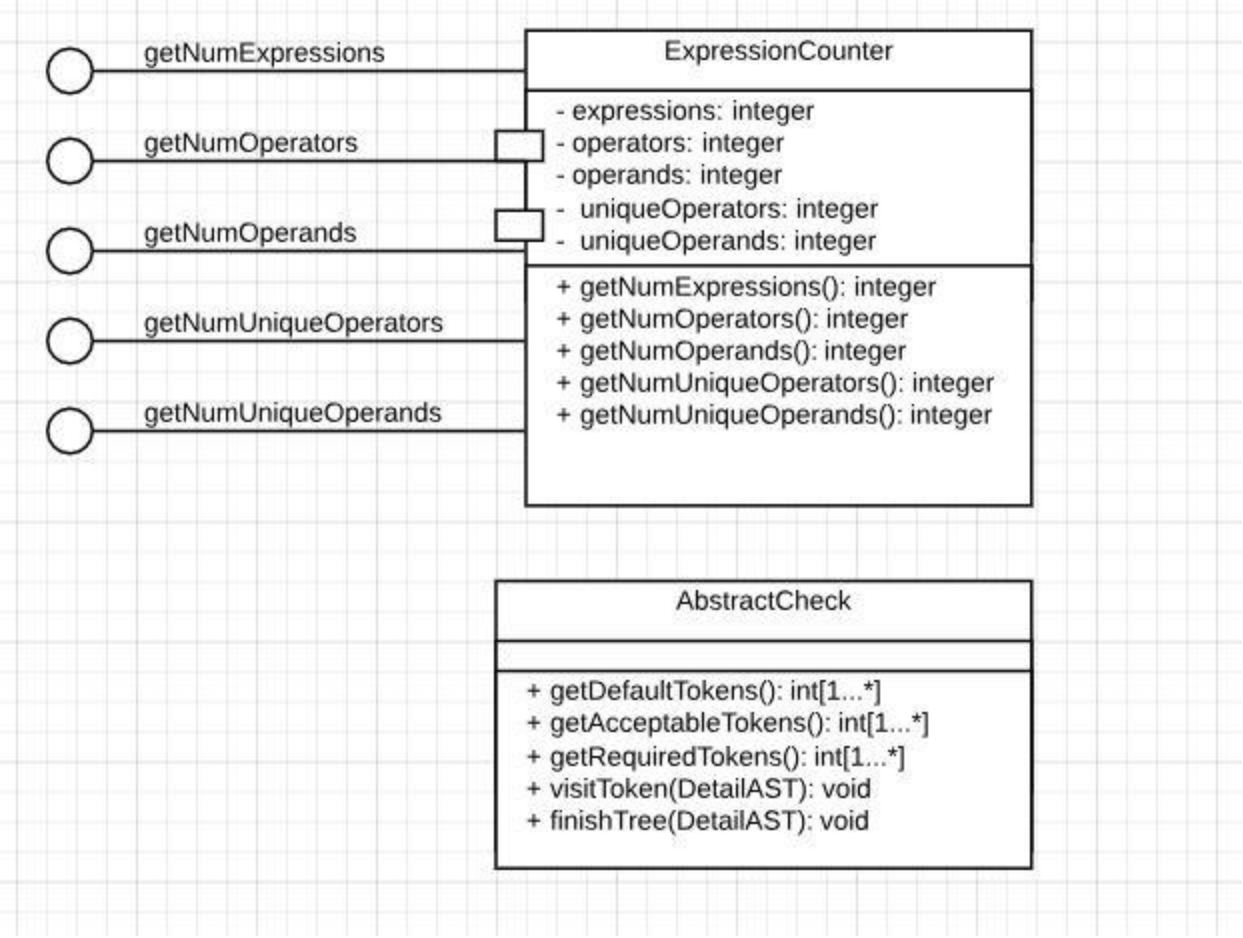
Architectural Diagram

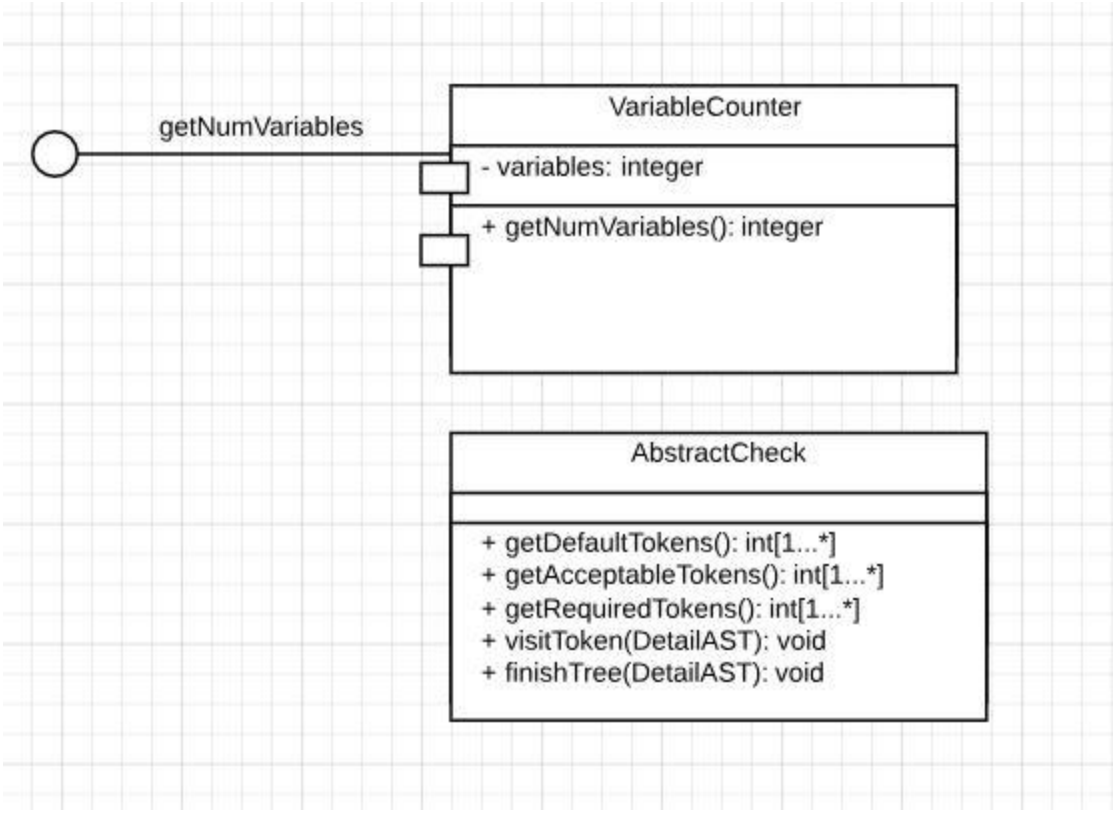


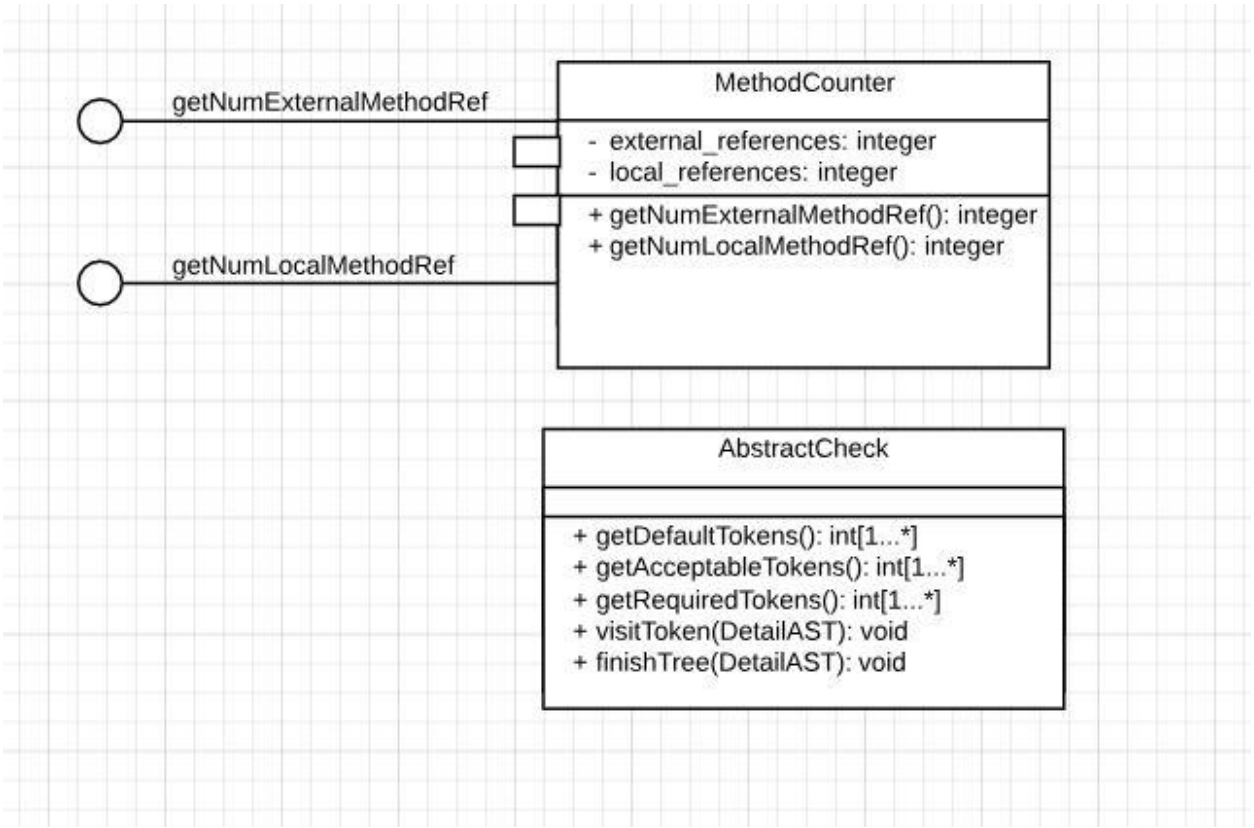
Detailed Component Diagram

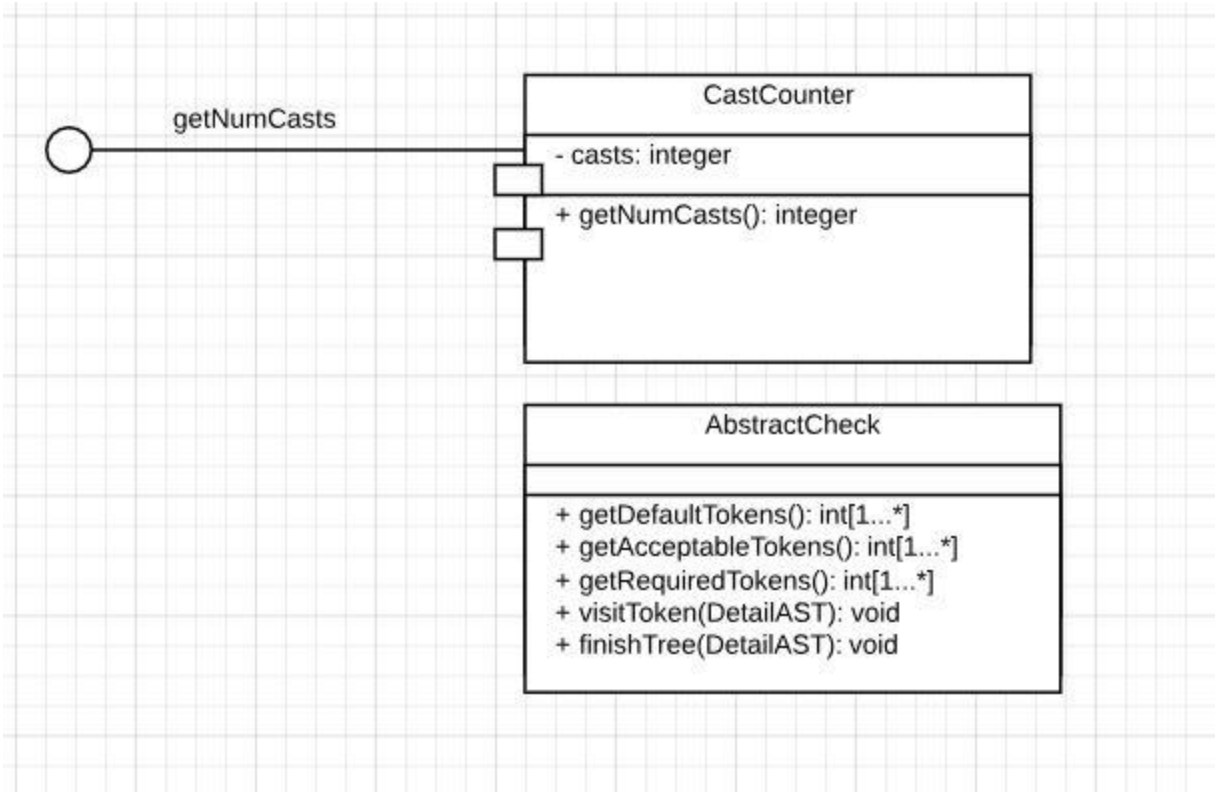


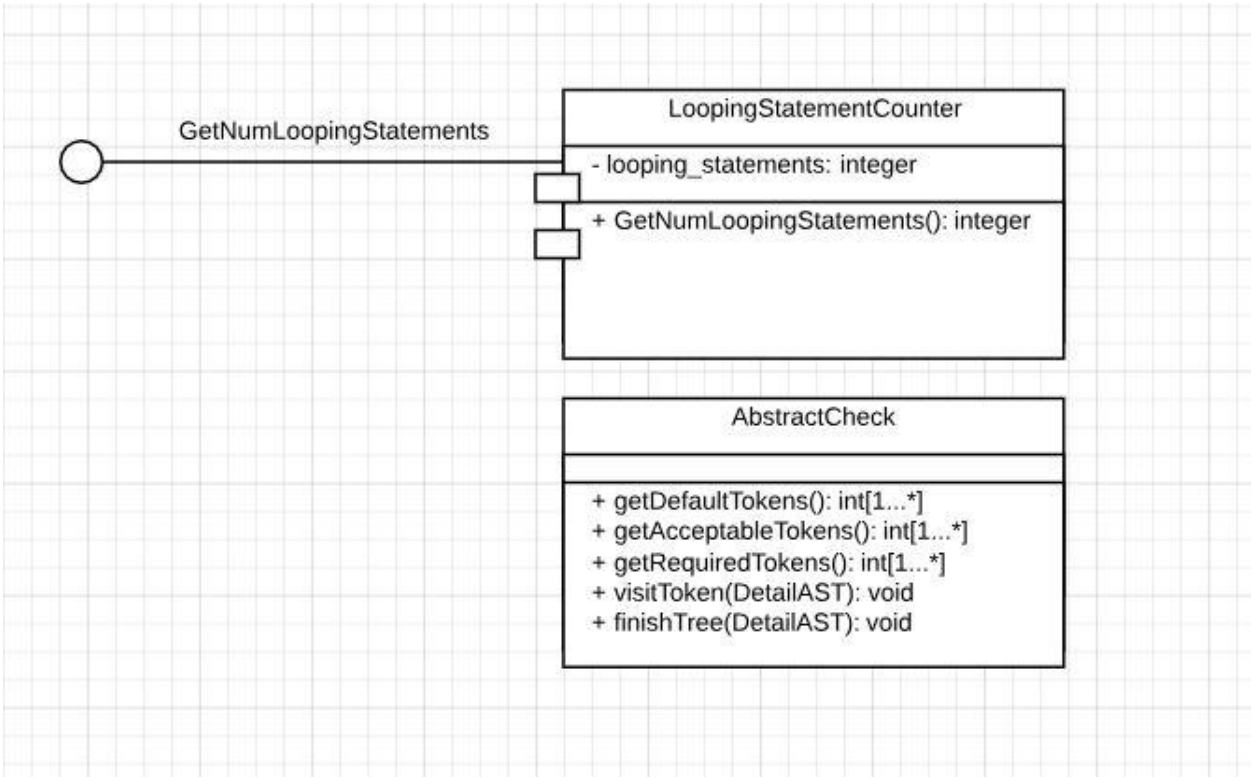


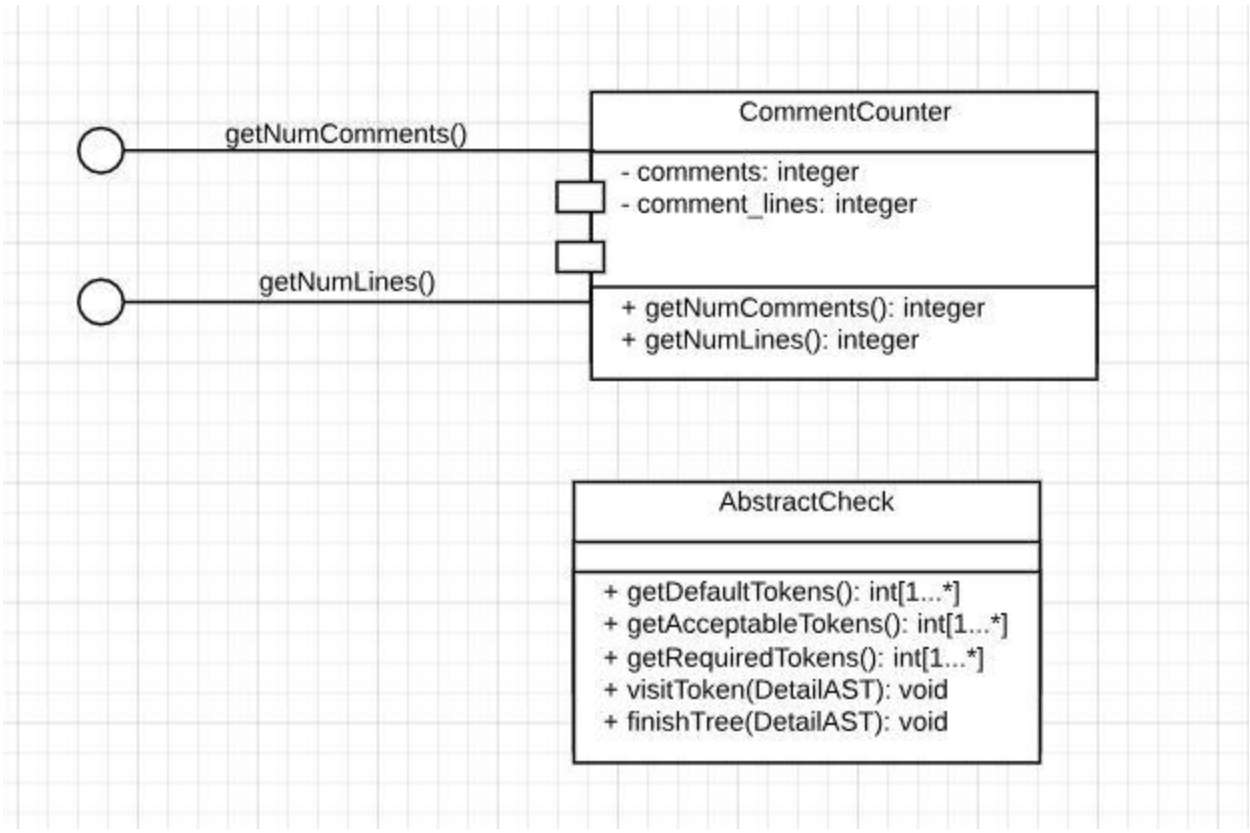

getNumExpressions
getNumOperators
getNumOperands
getNumUniqueOperators
getNumUniqueOperands
ExpressionCounter
- expressions: integer
- operators: integer
- operands: integer
- uniqueOperators: integer
- uniqueOperands: integer
+ getNumExpressions(): integer
+ getNumOperators(): integer
+ getNumOperands(): integer
+ getNumUniqueOperators(): integer
+ getNumUniqueOperands(): integer
AbstractCheck
+ getDefaultTokens(): int[1...*]
+ getAcceptableTokens(): int[1...*]
+ getRequiredTokens(): int[1...*]
+ visitToken(DetailAST): void
+ finishTree(DetailAST): void

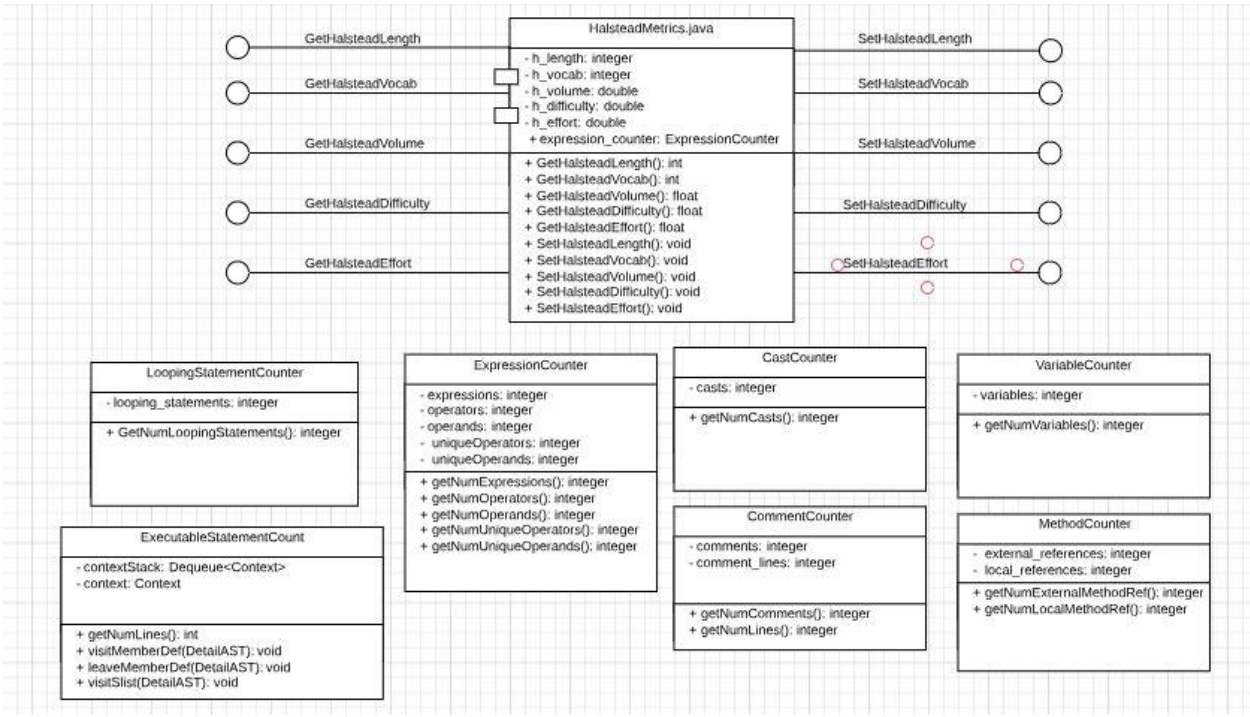

getNumVariables
VariableCounter
- variables: integer
+ getNumVariables(): integer
AbstractCheck
+ getDefaultTokens(): int[1...*]
+ getAcceptableTokens(): int[1...*]
+ getRequiredTokens(): int[1...*]
+ visitToken(DetailAST): void
+ finishTree(DetailAST): void

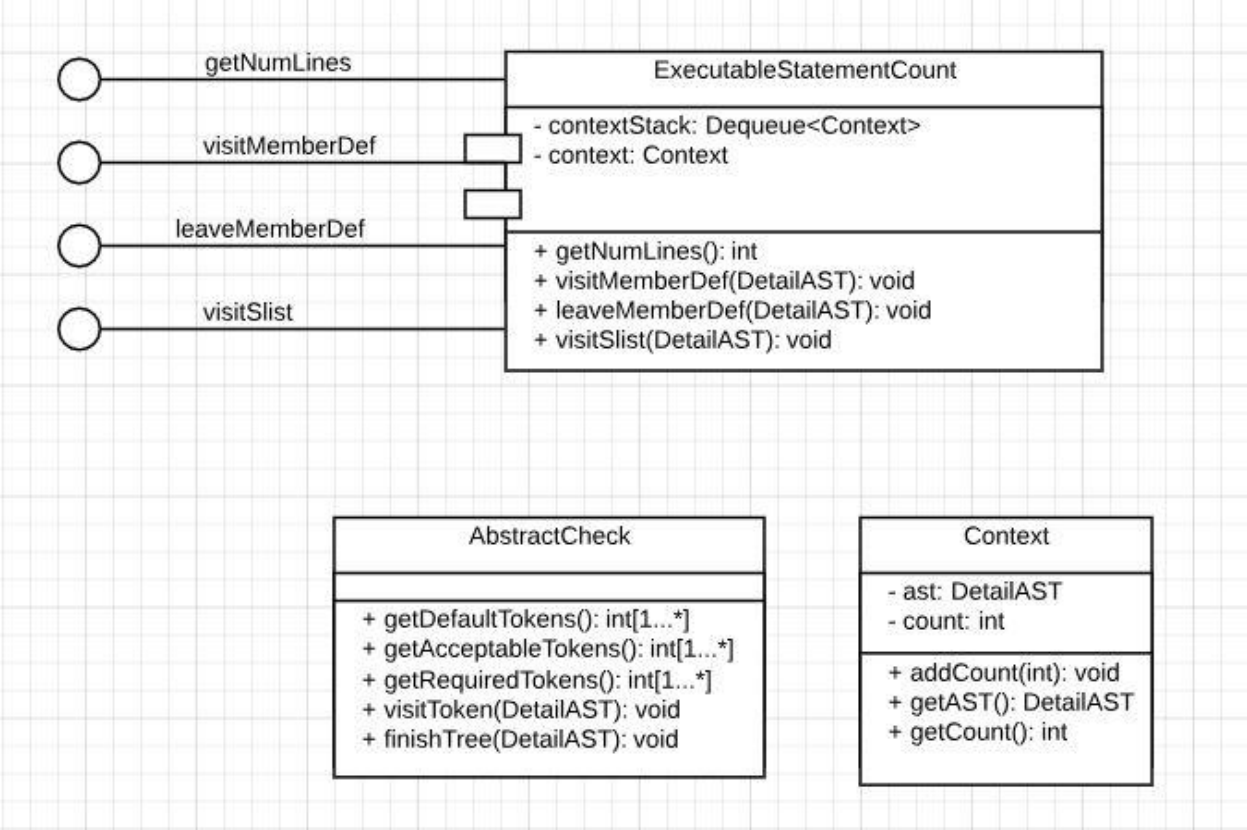

getNumExternalMethodRef
getNumLocalMethodRef
MethodCounter
- external_references: integer
- local_references: integer
+ getNumExternalMethodRef(): integer
+ getNumLocalMethodRef(): integer
AbstractCheck
+ getDefaultTokens(): int[1...*]
+ getAcceptableTokens(): int[1...*]
+ getRequiredTokens(): int[1...*]
+ visitToken(DetailAST): void
+ finishTree(DetailAST): void


getNumCasts
CastCounter
- casts: integer
+ getNumCasts(): integer
AbstractCheck
+ getDefaultTokens(): int[1...*]
+ getAcceptableTokens(): int[1...*]
+ getRequiredTokens(): int[1...*]
+ visitToken(DetailAST): void
+ finishTree(DetailAST): void


GetNumLoopingStatements
LoopingStatementCounter
- looping_statements: integer
+ GetNumLoopingStatements(): integer
AbstractCheck
+ getDefaultTokens(): int[1...*]
+ getAcceptableTokens(): int[1...*]
+ getRequiredTokens(): int[1...*]
+ visitToken(DetailAST): void
+ finishTree(DetailAST): void


getNumComments()
getNumLines()
CommentCounter
- comments: integer
- comment_lines: integer
+ getNumComments(): integer
+ getNumLines(): integer
AbstractCheck
+ getDefaultTokens(): int[1...*]
+ getAcceptableTokens(): int[1...*]
+ getRequiredTokens(): int[1...*]
+ visitToken(DetailAST): void
+ finishTree(DetailAST): void


GetHalsteadLength
GetHalsteadVocab
GetHalsteadVolume
GetHalsteadDifficulty
GetHalsteadEffort
HalsteadMetrics.java
- h_length: integer
- h_vocab: integer
- h_volume: double
- h_difficulty: double
- h_effort: double
+ expression_counter: ExpressionCounter
+ GetHalsteadLength(): int
+ GetHalsteadVocab(): int
+ GetHalsteadVolume(): float
+ GetHalsteadDifficulty(): float
+ GetHalsteadEffort(): float
+ SetHalsteadLength(): void
+ SetHalsteadVocab(): void
+ SetHalsteadVolume(): void
+ SetHalsteadDifficulty(): void
+ SetHalsteadEffort(): void
SetHalsteadLength
SetHalsteadVocab
SetHalsteadVolume
SetHalsteadDifficulty
SetHalsteadEffort
LoopingStatementCounter
- looping_statements: integer
+ GetNumLoopingStatements(): integer
ExpressionCounter
- expressions: integer
- operators: integer
- operands: integer
- uniqueOperators: integer
- uniqueOperands: integer
+ getNumExpressions(): integer
+ getNumOperators(): integer
+ getNumOperands(): integer
+ getNumUniqueOperators(): integer
+ getNumUniqueOperands(): integer
CastCounter
- casts: integer
+ getNumCasts(): integer
VariableCounter
- variables: integer
+ getNumVariables(): integer
ExecutableStatementCount
- contextStack: Dequeue<Context>
- context: Context
+ getNumLines(): int
+ visitMemberDef(DetailAST): void
+ leaveMemberDef(DetailAST): void
+ visitSlist(DetailAST): void
CommentCounter
- comments: integer
- comment_lines: integer
+ getNumComments(): integer
+ getNumLines(): integer
MethodCounter
- external_references: integer
- local_references: integer
+ getNumExternalMethodRef(): integer
+ getNumLocalMethodRef(): integer


getNumLines
visitMemberDef
leaveMemberDef
visitSlist
ExecutableStatementCount
- contextStack: Dequeue<Context>
- context: Context
+ getNumLines(): int
+ visitMemberDef(DetailAST): void
+ leaveMemberDef(DetailAST): void
+ visitSlist(DetailAST): void
AbstractCheck
+ getDefaultTokens(): int[1...*]
+ getAcceptableTokens(): int[1...*]
+ getRequiredTokens(): int[1...*]
+ visitToken(DetailAST): void
+ finishTree(DetailAST): void
Context
- ast: DetailAST
- count: int
+ addCount(int): void
+ getAST(): DetailAST
+ getCount(): int

Milestone Report

Role and Task Assignment of Each Team Member

Wai Fong – Class diagram, HalsteadMetrics, Maintainability Index

Shrunga Malavalli – Architecture diagram, LoopingStatementCounter, and part of Milestone Report

Linh Nguyen – Use case diagram and MethodCounter

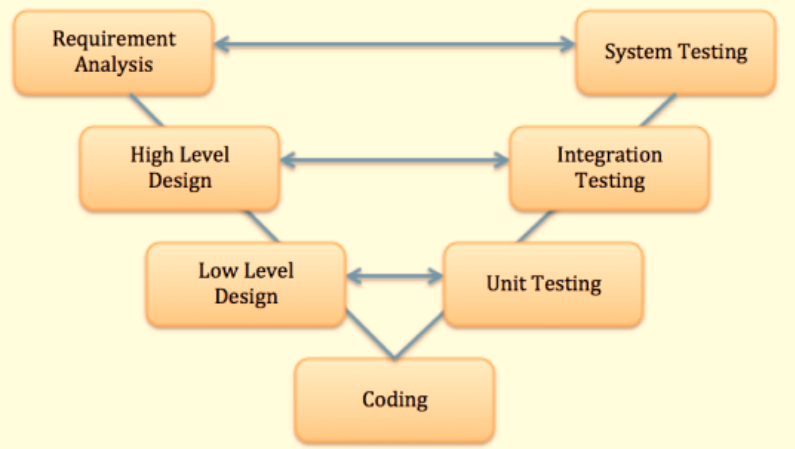
Carly Ott – Component diagram

Kimi Phan – CastCounter, ExpressionCounter, and created the skeleton code to start the project

Kayla Rhodes – CommentCounter, VariableCounter, documentation, and part of the Milestone Report

The Software Process Chosen and Justification

We chose to follow the V-model of software development.



We picked this process as it seemed the most reasonable given the project. In this first deliverable, we traversed down the left side of the V-Model by doing the requirements analysis, drawing our diagrams for the high-level design, and then discussing the implementation details before getting to the coding. This process also gives us a good guideline to follow for when the upcoming deliverables introduce the testing components into the code. We can use this methodology to reiterate over our design elements and test each level accordingly.

The Anti-Patterns Chosen and Realized in the Code

One of the main Anti-Patterns we realized was Cut-And-Paste-Programming. Given the similarity of the checks implemented for the structural metrics, we chose this anti-pattern as it seemed the most likely to appear in a similar context in industry. Similar functionality was cut and paste and therefore has a propensity to propagate bugs through multiple components.

One of the other Anti-Patterns is a variation of Lava Flow. We divided the code amongst ourselves and developed each check in isolation for the most part. However, the code could have been made much less redundant and impervious to refactoring in the future.

Major Project Activities

Team Meeting: August 29th, 3:00:

- Participants: Wai Fong, Shrunga Malavalli, Kimi Phan, Kayla Rhodes
- Created outline of our plan to finish the deliverable

Team Meeting: September 4th, 3:00

- Participants: Wai Fong, Shrunga Malavalli, Linh Nguyen, Kimi Phan, Kayla Rhodes
- Discussed diagrams and figured out who was going to create them

Team Meeting: September 11th, 2:00

- Participants: Wai Fong, Shrunga Malavalli, Linh Nguyen, Carly Ott, Kimi Phan, Kayla Rhodes
- Discussed diagrams again, made plan to meet with professor to get diagrams reviewed the next day

Team Meeting: September 20th, 2:00

- Participants: Wai Fong, Linh Nguyen, Carly Ott, Kimi Phan, Kayla Rhodes
- Updated team members on coding progress
- Tried to get CheckStyle setup and MethodLimitCheck working on everyone's computer

Team Meeting: September 27th, 2:00

- Participants: Wai Fong, Linh Nguyen, Shrunga Malavalli, Kimi Phan, Kayla Rhodes
- Updated team members on coding progress
- Discussed updated deadline and state of deliverable.