Analyse statique avec PMD

Simon Allier(simon.allier@inria.fr)

Last update 9 décembre 2013

1 Présentation de PMD

PMD [1] est un rule checker qui permet de vérifier statiquement qu'un projet ne viole pas un ensemble de règles. PMD fournit plus de 300 règles regroupées en ruleSets. Par exemple, toutes les règles liées à l'utilisation des `String` dans java se trouvent dans le ruleSet java-strings.

L'archive `pmd.zip` contient les sources de PMD, un projet maven pour les compiler et un exemple de règle (`vv.tp3.WhileLoopsMustUseBracesRule`). Pour importer le projet dans Eclipse : File → Import... → Existing Maven Projects.

1.1 Utilisation

Utilisation de PMD à partir du jar créé avec maven (goals `package`) :

```
java -cp .:target/pmd-5.0.5-jar-with-dependencies.jar
net.sourceforge.pmd.PMD  -d src/main/java/
-R java-basic,java-design,java-unusedcode,java-optimizations,
java-empty,java-strictexception,java-strings
-f vbhtml -r report.html
```

Cette commande exécute les ruleSets `java-basic, java-design, java-unusedcode, java-optimizations, java-empty, java-strictexception, java-strings` sur toutes les classes présentes dans le répertoire de source `src/main/java/`. Un rapport au format html est généré.

1.2 Extension de PMD

PMD permet de facilement ajouter de nouvelles règles. Les nouvelles règles doivent hériter de la classe `net.sourceforge.pmd.lang.java.rule.AbstractJavaRule` et redéfinir une ou plusieurs méthodes `visit(ASTStatement node, Object data)`. De plus, tous les objets `node` implémentent l'interface `net.sourceforge.pmd.lang.ast.Node`. Celle-ci fournit des méthodes utiles pour écrire les règles. Enfin, les règles doivent être décrites dans un ruleSet.

Un tutoriel se trouve à l'adresse suivante :
`http://pmd.sourceforge.net/pmd-5.0.5/howtowritearule.html`.

1. http ://pmd.sourceforge.net/

1.3 Exemple de règle « chaque while doit avoir des accolades »

On définit une règle autorisant uniquement les boucles while faisant usage d'accolades. La figure 1 contient la classe définissant cette règle, et la figure 2 contient le fichier de configuration du ruleSet contenant cette règle.

Boucle *while* correcte :

```
while (baz) {
   buz.doSomething();
}
```

Boucle *while* incorrecte :

```
 while (baz)
   buz.doSomething();
```

Commande pour utiliser cette règle : `java -cp .:target/pmd-5.0.5-jar-with-dependencies.jar net.sourceforge.pmd.PMD -d src/main/java/ -R src/main/resources/rulesets/java/VVRulesSet.xml`

2 Questions

Question 1 *Appliquez PMD sur le package* `simpleGame` *du tp2. Pour chaque violation de règle de la classe* `Board`*, vous devez :*

- *proposer un fix ;*
- *ou expliquer pourquoi cette violation est un faux positif.*

Question 2 *Appliquez PMD sur ses propres sources. Qu'observez-vous ?*

Question 3 *Ecrivez les trois règles suivantes :*

1. *Une violation est levée dès que deux boucles* `for` *sont imbriquées.*
2. *Une violation est levée pour chaque* `while(true)` *ou* `while(false)`*.*
3. *Raffinez la règle précédente en prenant en compte les possibilités d'échappement (*`break` *ou* `return`*) dans la boucle* `while`*.*

Ces règles doivent appartenir au package `vv.tp3` *et elles doivent être ajoutées au rulesSet* `src/main/resources/rulesets/java/VVRulesSet.xml`*.*

3 Rapport

Vous devez rendre :

- Un rapport au format PDF contenant les réponses aux questions 1 et 2.
- Un zip contenant :
  - la classe `Board` corrigée à l'aide de PMD ;
  - les 3 classes implémentant les règles de la question 3.

```java
package vv.tp3;

import net.sourceforge.pmd.lang.ast.Node;
import net.sourceforge.pmd.lang.java.ast.*;
import net.sourceforge.pmd.lang.java.rule.AbstractJavaRule;

public class WhileLoopsMustUseBracesRule extends AbstractJavaRule {

    public Object visit(ASTWhileStatement node, Object data) {
        Node firstStmt;
        firstStmt = (Node)node.jjtGetChild(1);
        if (!hasBlockAsFirstChild(firstStmt)) {
            //ajout de la vioaltion
            addViolation(data, node);
        }
        return super.visit(node,data);
    }

    private boolean hasBlockAsFirstChild(Node node) {
        return (node.jjtGetNumChildren() != 0 && (node.jjtGetChild(0)
            instanceof ASTBlock));
    }
}
```

FIGURE 1 – La classe qui implémente la règle while

```
<!-- src/main/resources/rulesets/java/VVRulesSet.xml -->
<?xml version="1.0"?>
<ruleset name="VVRules"
    xmlns="http://pmd.sourceforge.net/ruleset/2.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://pmd.sourceforge.net/ruleset/2.0.0 http://pmd.
        sourceforge.net/ruleset_2_0_0.xsd">
    <rule name="WhileLoopsMustUseBracesRule"
          message="Avoid using 'while' statements without curly braces"
          class="vv.tp3.WhileLoopsMustUseBracesRule">
      <description>
      Avoid using 'while' statements without using curly braces
      </description>
        <priority>3</priority>

      <example>
<![CDATA[
    public void doSomething() {
      while (x < 100)
          x++;
    }
]]>
      </example>
    </rule>
</ruleset>
```

FIGURE 2 – Le fichier de configuration du ruleSet pour la règle while