6ZC-1

Javaセキュアコーディングを促進する不変クラスチェッカの提案と実装

菱田 昂宏† 早川 智一† 疋田 輝雄†
明治大学理工学部情報科学科†

1. はじめに

近年，IT の発達で情報システムが普及する一方で，システムの脆弱性による個人情報の流出などの事故が相次いでいる[3]．システムの脆弱性には様々な原因があるが，その 1 つにコーディングの不備がある．ここで，コーディングに多く使われる言語として Java が挙げられる[6]．Java は C/C++などと比べてセキュアな言語であるが，セキュアなシステムの構築にはプログラマの高い技量が依然として必要である．

この現状を踏まえ，Java のセキュアコーディングのガイドライン[2]（以下，ルール集）を JPCERT/CC が公表しているが，ルール数が 156 と多く内容も難しいため，完全な普及には至っていないようである．また，我々の事前調査では，ルール集への違反の大部分を検出できる静的解析ツールは存在しなかった．

そこで我々は，Java のセキュアコーディングを促進するために不変クラスの利用を提案し，そのチェッカの実装について本論文で報告する．

本論文の構成は次のとおりである．2 節では関連研究を紹介する．3 節では提案手法を概説する．4 節と 5 節では設計と実装を述べる．6 節では評価結果を報告する．7 節では今後の課題を述べる．

2. 関連研究

Kjolstad ら[4]は，不変クラスを用いることはプログラマにとって多くのメリットがあるが，不変クラスを正しく実装することは難しいと述べている．彼らは Java の可変クラスを不変クラスに自動変換する処理系を提案しているが，本研究は不変クラスの学習支援に重点を置いている．

Block[1]は，「可変にすべきかなり正当な理由がない限り，クラスは不変であるべき」と述べており，これは不変クラスの利用を推奨する我々の主張と合致する．

Proposal and Implementation of Immutability Checker for Java Secure Coding
†Takahiro Hishida, Tomokazu Hayakawa and Teruo Hikita, School of Science and Technology, Meiji University

3. 提案手法

3.1. 前提：不変クラスの推奨とその課題

不変クラスとは，初期化が安全に完了した後は内部状態が変わらないクラスのことである．不変クラスのインスタンスは，生存期間中は常に一定の状態を保つ．不変クラスの特徴には，（1）インスタンスの状態空間が最小になる，（2）無条件でインスタンスを共有できる――などがある．不変クラスは，Java の重要なクラス（例：String や Integer など）でも使われている．

不変クラスはマルチスレッド用のパターンとして知られているが[7]，セキュアコーディングの促進にも有用であると我々は考える．なぜならば，我々の予備調査で，ルール集の中に不変クラスの利用で解決できるものが散見されたためである．

一方で我々は，不変クラスを書くこと自体が初学者には容易ではないと懸念する．これは，不変クラスの必要条件の中に難しいものがあるためである（例：コンストラクタ中からの this 参照の逸出など）．また，不変クラスの学習を促進するシステムは存在しないようである．

3.2. 提案概要：不変クラスチェッカ

我々は，3.1 節の課題を踏まえ，不変クラスチェッカを提案する．**図 1** に，提案チェッカの概要を示す．提案チェッカは，（1）学習者のソースコードを解析し，（2）ソースコード中のクラスが可変か不変かを判定し，（3）クラスが可変であれば，不変にするための診断メッセージを表示する．**図 2** に，提案チェッカのスクリーンショットを示す．図より，学習者がソースファイルを GUI から指定すると，下部のテキストエリアにその診断結果が表示されることが分かる．

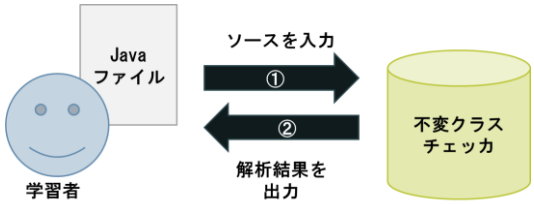

図 1 提案システムの概要

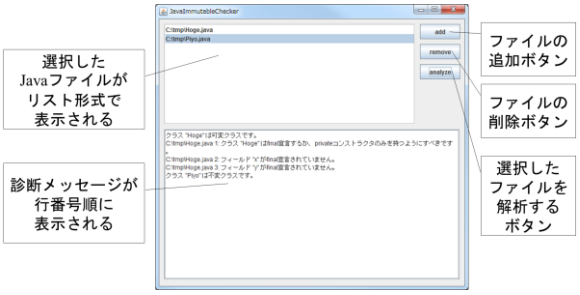

図 2 提案チェッカの GUI

4. 不変クラスチェッカの設計

不変クラスチェッカの設計は次のとおりである：（1）入力の Java ファイルを抽象構文木（AST：Abstract Syntax Tree）に変換する；（2）変換した AST を Visitor パターンで走査し，クラスの不変性の解析を行う；（3）解析結果を元に学習者へ診断メッセージを表示する．

表 1 に，提案チェッカの不変クラスの判定条件を示す．提案チェッカは，表の全条件を対象クラスが満たすと不変と判定する（厳密には，表の一部の条件を満たさずとも不変となる実質不変クラスが存在する．提案チェッカは，実質不変クラスの判定にも部分的に対応している）．なお，提案チェッカは，複雑な参照の追跡やクラスファイルの解析などにはまだ未対応である．

5. 不変クラスチェッカの実装

我々は，Java を用いて不変クラスチェッカを実装した．ソースコードから AST への変換には，JDK7 のコンパイラツリーAPI[5]を使用した．GUI の実装には Swing を使用した．

6. 提案手法の評価

6.1. 不変クラスの有用性の評価

我々は，不変クラスの有用性を評価するために，全 156 のルール集の中から不変クラスのみで対応可能なルール数を算出した．

評価の結果，全数の約 1 割にあたる 14 ルール（ルール番号：OBJ00-J, OBJ01-J, OBJ04-J, OBJ05-J, OBJ06-J, OBJ08-J, OBJ10-J, VNA00-J, VNA01-J, VNA02-J, LCK00-J, TSM01-J, TSM03-J, SER06-J）に不変クラスのみで対応できることが分かった．この値が十分と断言することは難しいが，ルール集の学習に多くの時間がかかることを考慮すると，セキュアコーディングの促進に不変クラスは有用であると我々は考える．

表 1 提案チェッカの不変クラスの判定条件

No	判定条件（抜粋）
1	クラスが final 宣言されている．
2	全フィールドが final 宣言されている．
3	コンストラクタ内で this が逸出していない．
4	可変フィールドに対する独占的アクセスが保証されている．
5	可変フィールドを初期化後に変更していない．

6.2. 不変クラスチェッカの有用性の評価

我々は，不変クラスチェッカの有用性を評価するために，6 名の Java の初学者を被験者に用いた試験を行った．具体的には，（1）可変クラスを不変化する問題を独力で解いてもらい，（2）不正解者には提案チェッカを使って再回答してもらい，（3）提案チェッカで不変化に成功した人数を評価値とした．

評価の結果，独力で正答したのは 2/6 名で，提案チェッカの利用で正答したのは 2/4 名であった．提案チェッカによる正答率は 50%に止まったが，その原因の 1 つとして問題が簡単であった可能性がある．したがって，問題が難しくなるにつれて独力での正答率は下がり，提案チェッカによる正答率が上がると我々は推測する．

7. おわりに

本論文では，Java のセキュアコーディングを促進するために不変クラスの利用を推奨し，その学習を支援するチェッカの実装について報告した．評価の結果は，不変クラスがセキュアコーディングに有用であり，提案チェッカが不変クラスを初学者が作成する際の一助となることを示した．今後は，不変クラスの判定精度の向上や，提案チェッカの Web サービス化を予定している．また，より多くの課題を用いて提案チェッカの追加評価を行うことも検討中である．

参考文献

[1] Block, J.：Effective Java 第 2 版，ピアソンエデュケーション(2008).
[2] JPCERT/CC：Java セキュアコーディングスタンダード CERT/Oracle 版，http://www.jpcert.or.jp/java-rules/.
[3] JPCERT/CC：注意喚起，http://www.jpcert.or.jp/at/2013.html.
[4] Kjolstad, F., Dig, D., Acevedo, G. and Snir, M.: Transformation for Class Immutability, 33rd International Conference on Software Engineering, pp.61–70 (2011).
[5] Oracle: Compiler Tree API, http://docs.oracle.com/javase/7/docs/jdk/api/javac/tree/.
[6] TIOBE SOFTWARE: http://www.tiobe.com/.
[7] 結城浩：増補改訂版 Java 言語で学ぶデザインパターン入門マルチスレッド編，ソフトバンククリエイティブ(2006).