B-004

オブジェクト指向プログラミング初学者のためのソースコード評価ツール
Object Oriented Code Quality Evaluation Tool for Novice Programmer

若林 智徳† 松浦 佐江子‡
Tomoyoshi Wakabayashi Saeko Matsuura

1. はじめに

オブジェクト指向プログラミング初学者のための演習講義で出題された課題に対して，提出された初学者のソースコードを分析すると，あるデータを扱う手続きが複数のクラスに横断的に定義される，あるいは，特定のクラスに手続きのみが集中して定義されるというような，教員が本来意図していたクラス構成とは異なるクラス構成が構築されるという問題が見つかっている．データとそのデータを扱う手続きが分離されることにより，オブジェクト指向の利点であるカプセル化が十分に機能せず，ソースコードの保守性は低下する．

本稿では，オブジェクト指向プログラミング言語である Java をターゲットとし，初学者がコーディングを行う際に犯しがちな問題の中から，ソースコードの保守の妨げとなるもの，初学者の意図とは異なる動作をするもの，バグの原因となるものを整理し，初学者向けのソースコード品質評価基準として利用する方法を提案する．また，この方法に基づいて開発したツールを用いて初学者のソースコードを評価し，評価ツールの有効性を議論する．

2. ソースコードの品質評価

2.1 品質評価基準の定義

初学者が提出するソースコードに含まれる問題は，(1)コンパイル不可なもの，(2)コンパイル可能だが意図した動作を行わないもの，(3)意図した動作を行うが改善の余地があるものに分けられる．このうち(1)については Jackson ら[1]の調査により明らかにされている．本稿で扱う問題は(2)，(3)である．一般的な Java コーディング規約[2]やコードスタイル矯正の観点から，早期に修正することでソースコードの保守性を向上させるものや，バグの原因を作り難くなるものを規約として整理し，表 1 にまとめた．

これらの規約に定める制約は，その制約を必ず満たす必要がある強制的な制約と，制約を満たす必要は必ずしもではないが可能な限り満たす必要がある非強制な制約に分けられる．強制的な制約は，特に，初学者が意図していない動作を引き起こしやすいものとして，オブジェクト同士の比較を定義し，フィールドに対して副作用を及ぼす箇所が明示的でなくなる要因となりやすい public なフィールド，メソッド引数への代入を定義した．制約で閾値が必要な場合は，いくつかのコーディング規約もしくは実際に初学者が作成したソースコードから出現頻度を調査し，全体の 95%程度を基準として定義した．

† 芝浦工業大学大学院工学研究科 Graduate School of Engineering, Shibaura Institute of Technology
‡ 芝浦工業大学大学院理工学研究科 Graduate School of Science and Engineering, Shibaura Institute of Technology

表 1 ソースコード品質評価で用いる規約

規約名	動機	制約
複雑なメソッド	制御文の多いメソッドは可読性を下げる	メソッドの複雑度は8未満
長い行	長すぎる行は可読性を下げる	1行は最大80桁まで
publicなフィールド	publicなフィールドは副作用を与える箇所がわかりづらくなる	フィールドの可視性は極力privateにする
staticなフィールド・メソッド	staticなフィールド・メソッドは副作用が分かり難い	staticなフィールド・メソッドは極力使用しない
ローカル変数の再利用	複数の役割を持つ変数は可読性を下げる	同じ変数を違う目的で使いまわさない
条件文中の代入	代入の混ざった条件文は条件判定の順序がわかりづらい	条件文中での代入は極力使用しない
大小比較演算子	大小比較の順序が統一されていない条件文は可読性を下げる	大小比較では"<"や"<="を使用する
オブジェクト同士の比較	オブジェクト同士は比較演算子で比較するべきではない	オブジェクト同士の比較はequalsメソッドを使用する
メソッド引数への代入	引数に対して副作用を及ぼすべきではない	メソッド引数への代入は行わない
長い条件文	長い条件文は条件分岐の可読性を下げる	1つの条件文で用いる比較演算子の数は5以内にする
thisオブジェクト呼び出しの連鎖	A().B().C().D()のような呼び出しの連鎖は可読性を下げる	呼び出しの連鎖は3以下にする
数値の混ざった名前	数値を混ぜた変数名は可読性を下げる	数値を混ぜた命名は可能な限りしない

表 2 制約の種類

制約の種類	規約名
非強制	複雑なメソッド，長い行，static なフィールド・メソッド，ローカル変数の再利用，条件文中の代入，大小比較演算子，長い条件文，this オブジェクト呼び出しの連鎖，数値の混ざった名前
強制	public なフィールド，オブジェクト同士の比較，メソッド引数への代入

2.2 フィードバック方法

2.2.1 Web 教科書の利用

初学者を対象とした Web 教科書システム[3]に提出されたソースコードに対して，表 1 の規約に当てはまるかを検査し，違反している箇所について，その箇所をハイライトすると同時に，メッセージを添えて出力する．

2.2.2 非強制的な制約のフィードバック

初学者に対する提案という形でメッセージを出力する．例えば，長い条件文が存在した場合は「長い条件文があります．条件文を精査し，簡潔にまとめられないか検討してください」というメッセージを出力する．また，ド・モルガンの法則等により機械的にまとめられる場合は，その手順を説明する．

2.2.3 強制的な制約のフィードバック

public なフィールドとオブジェクト同士の比較は，強制的に置換し，その旨をメッセージとして出力する．メソッド引数への代入は，代入が行われていることを警告するとともに，本当に代入が必要なのか確認するようメッセージを出力する．

3. 実装

3.1 ツールの概要

本ツールでは Java ソースコードの解析に ASTParser を使用する．Java ソースコードの抽象構文木を生成し，制約を満たすかを判断する visitor クラス群に渡すことで Java ソースコードを分析する．分析結果は Web 教科書で使用する形式と tsv 形式のレポートとして出力する．

3.2 分析方法

まず，対象の Java ソースコードに含まれる，クラス，メソッド，フィールド，ローカル変数名を取得する visitor クラスを使い，クラスの情報を保存する．次に規約毎に作成した visitor クラスを用いて分析を行う．例えば長い行の分析では，抽象構文木の ExpressionStatement ノードもしくは FieldDeclarationStatement ノードの長さを取得し，これが 80 文字以下であるかどうかを調べる．80 より大きいノードを発見した場合は，そのノードを含むクラス名，開始位置と終了位置を記録し，初学者に対するフィードバック用メッセージを生成する．

4. 評価ツールによるソースコード検証

ツールを用いて初学者のソースコードの検証を行った．表 3 は public なフィールドと static なフィールド・メソッドについて，初学者に出題した課題別に，全てのフィールド・メソッドに対する比率を計算したものである．

表 3　全フィールド・メソッドに対する出現率

課題	Non Private Field (%)	Static Field (%)	Static Method (%)
A	100	10	2
B	99	17	2
C	98	35	4
D	98	17	3
E	78	22	4
F	45	27	3
G	44	4	4
H	34	6	4

課題 A から F は初めて Java プログラミングを行う学生に出題した課題であり，A から F の順序で出題している．課題 G と H は前期に Java プログラミング演習を受講した学生に出題した課題である．表 2 から，非 private なフィールドの出現率は講義が進んでいく毎に減少していることが分かる．また，課題 F ではいくつかのフィールドのアクセス指定子を private にするように指定した課題であったため，一つ前の課題より非 private なフィールドが大きく減少していることが分かる．一方で static なフィールドやメソッドの出現率は課題毎にばらつきがあったが，同じ学生が static を多用しているケースがみられた．これらは main メソッドからそのメソッドを直接呼び出すためだけに static として定義されており，クラスメソッドとして意味のある定義ではなかった．またテストを行うメソッド，テスト対象であるメソッドに付けられているケースが見られた．

図 1 のソースコードは LineSegment クラスの add メソッドと delete メソッドをテストすることを意図したメソッドであるが，オブジェクト同士の比較を“==”演算子を用いて行っているため，学生が意図した動作と異なった動作を行う可能性がある．また，引数で得た文字列から 2 つの異なる処理を行うような分岐制御が行われているため，メソッドの複雑度が 18 と高い値を示しており，条件分岐の再検討やメソッドの抽出を促す必要があると考えられる．さらに，メソッド名の先頭が大文字であるためクラス名と混同しやすいという問題や，ループ用の変数が for 文のスコープ外で定義されているため，変数の扱い次第でバグの原因となりうる可能性がある．

図 1　複数の問題を含むメソッドの例

5. まとめ

初学者のソースコードに適した品質評価基準とフィードバック方法を定義し，品質評価基準に基づいてソースコードの評価を行うツールを作成した．さらに，作成したツールを用いて初学者のソースコードの検証を行った．品質評価基準を課題毎に適用することで，ソースコード品質の変遷を客観的に見ることができ，初学者の傾向分析や課題作成の支援が容易になることが期待される．

今後の課題として，Web 教科書で学習する初学者に向けた問題点の視覚化の方法を検討し実装する．また，図 1 のソースコードにあるループ変数の問題のような，変数のスコープに関する問題点や，変数の命名規則に関する問題点を明らかにし品質評価基準として取り込む方法を実装する．

参考文献

[1]James Jackson, Michael Cobb, Curtis Carver, “Identifying Top Java Errors for Novice Programmers”, Frontiers in Education, T4C(2005)
[2]Scott W. Ambler, “Java スタイルブック”, 翔泳社(2009)
[3]文科省 平成 21 年度大学教育・学生支援推進事業【テーマ A】大学教育推進プログラム「工学系技術者のソフトウェア開発技能育成」
http://www.shibaura-it.ac.jp/about/support_program/program13.html