プログラミング演習におけるプログラム作成時の名前付けに対するシステムによる指導

上村 拓磨*1・北 英彦*1
Email:416n207@m.mie-u.ac.jp

*1：三重大学工学研究科電気電子専攻

◎Key Words プログラミング演習，プログラミング演習システム，名前付け

1. はじめに

現在多くの大学がプログラミングの講義を行っている．しかし，プログラミング初心者に対するプログラミング演習において，プログラミングスタイルについて十分な指導を行う時間がないというのが現状である．著者らは，今までに学習者にプログラミングスタイルの一つである字下げ，空白の入れ方を自己学習させるためのシステムを開発した[1]．結果として，以前と比べ学習者が字下げ，空白の入れ方を適切にできるようになった．

本研究では，前研究では対象外としたプログラミングスタイルの内の一つである適切な変数名を学習者に名付けさせるためのシステムを開発する．学習者がコーディング中のプログラムに対してどのような変数名にすればよかったのか提示することにより適切な変数名を名付けさせるようにする．なお，本機能は著者らの所属する研究室で開発および運用を行っているプログラミング演習システム PROPEL に付け加える．三重大学電気電子工学科でのプログラミング演習は Java を用いたプログラミングの学習を行っているため，変数名の命名規則は Java のものに従うことにする．

2. プログラミングスタイル

プログラミングスタイルとはプログラムを書くときの規則のことであり，特定の規則に沿ってプログラムを書くことで，他の人が読んでも理解しやすいプログラムを作成することが可能である．カーニハン[2]は「プログラムを書くということは，正しい構文を使いバクを直し，それなりに動くようにすれば済むわけではない」と言っている．一般に，ソフトウェア開発はひとりで行うのではなく複数人でプロジェクトを組んで行うことが多い．そのため，プログラムを書く上で他の人にも読まれるということを意識して書くことが重要である．

プログラミングスタイルとしては，字下げの仕方，空白の入れ方，コメントの付け方，変数名の付け方，などがある．

これらを用いることの利点を以下に述べる．

- 字下げ：波括弧{}で囲まれた範囲を 1 段字下げすることで，ブロックの範囲を視覚的に分かりやすくする．
- 空白：演算子の前後に空白を入れることにより演算子を視覚的に分かりやすくする．
- コメント：プログラムの動作には関係ないが，他のプログラマに式の条件式などの意味を伝えるために使用する．
- 変数名：適切な変数名をつけることで変数の役割が分かりやすくなる．Java では Camel 形式，Pascal 形式，Snake 形式を使い分けている．表 1 に例を示す．

表 1 Java で用いている名前の書式

パスカル形式	Lecture SquareOfDistance	クラス名
キャメル形式	lecture squareOfDistance	変数名 メソッド名
スネーク形式	LECTURE SQUARE_DISTANCE	定数

Java の特徴として C と比べてプログラミングスタイルの流派が少ない点がある．理由として Java を最初に開発した Sun Microsystems[3]がコーディング規約を提示したことがある．企業ではそれをもとに自社でのコーディング規約を作成し用いることが多い．

プログラムのスタイルが不適切な例を図 1 に，適切な例を図 2 に示す．

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int a=-1;
        int b=-1;
        int c=-1;
      while(a<0||b<0||c<0 ||a>100||b>100||c>100){
      System.out.println("input range 0~100");
         System.out.print("Math point:");
         a=input.nextInt();
         System.out.print("Japanese point:");
          b=input.nextInt();
           System.out.print("English point:");
          c= input.nextInt();
      }
      int d;
          d=a+b+c;
      double e;
      e=d/3;
      System.out.printf("Total:%d Average:%.1f",d,e);

    }
}
```

(*a) 変数の役割が分からない

(*b) 字下げが不統一

図 1 スタイルが不適切なプログラム

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int math = -1;
 *a     int japanese = -1;
        int english = -1;
        while (math < 0 || japanese < 0 || english < 0 ||
                math > 100 || japanese > 100 || english > 100) {
            System.out.println("input range 0~100");
            System.out.print("Math point:");
 *b         math = input.nextInt();
            System.out.print("Japanese point:");
            japanese = input.nextInt();
            System.out.print("English point:");
            english = input.nextInt();
        }
        int sum;
        sum = math + japanese + english;
        double average;
        average = sum / 3;
        System.out.printf("Total:%d Average:%.1f", sum, average);
    }
}
```

(*a) 役割の分かる名前

(*b) 統一された字下げ

図 2　スタイルが適切なプログラム

3.　演習時の名前付けの現状

2016 年度のプログラミング演習で学習者が作成したプログラムを調べ，変数名が命名規則に則した形で名付けられているかどうかを確認した．結果を表 2 に記す．学習者の数は 78 人である．

結果として，約半数が不適切な変数名を付けていることが分かった．講師が事前に変数名の名付け方について説明を行っているにも関わらず，このような結果になったということは名前の重要性が理解できていないか，または，理解したつもりでも実践できていないことを示している．

表 2　変数名の付け方の現状

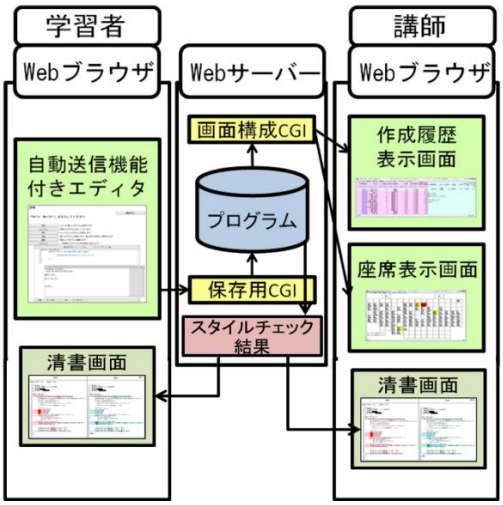
内容	個数	例
変数名の合計	1118 個	
不適切な変数名	558 個	
Camel 形式，Pascal 形式でない	474 個	AVERAGE englishpoint
誤字	43 個	avarage
意味が不適切	52 個	分散を dispersion
無意味な名前	26 個	a など一文字

※　Camel 形式でなく，かつ，誤字のものがある

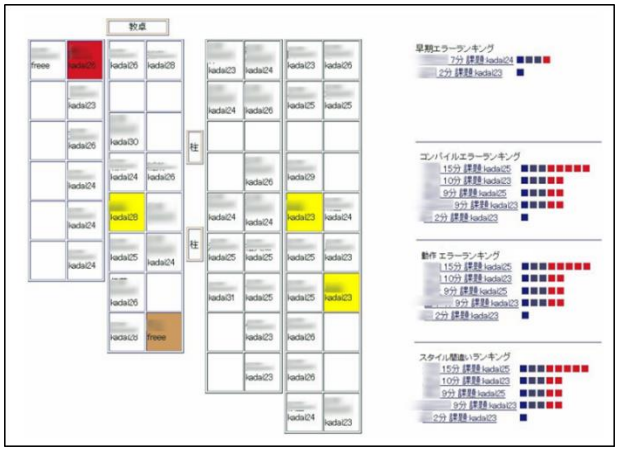
4.　プログラミング演習支援システム

著者らの所属する研究室では，学習者のプログラミング状況の把握および学習者への迅速な対処を目的として，プログラミング演習システム PROPEL の開発および運用を行っている．PROPEL のポリシーは，誤字・脱字などの軽微な間違いはできるだけシステムで検出し，その場で学習者に通知し，学習者に考えさせて修正させることである．これにより，講師は動作間違いなどより本質的な間違いの指導に専念することができるようになる．

図 3 (a)にシステム構成図を示す．PROPEL では，演習に必要なプログラムの記述，コンパイル，実行，提出，提出したものをシステムが清書したものの閲覧をウェブブラウザ上で行うことができる．エディタには，プログラムをサーバに自動的に送信する機能を埋め込んである．サーバでは，収集した学習者のプログラムを整理し，図 3 (b)に示すような講師用画面を生成する．この画面では，学習者の座席，取り組んでいる課題などを座席表の形で，コンパイルエラーなどの間違いを修正できずにいる時間をランキング形式で閲覧できる．また，学習者を選択すると，その学習者が作成しているプログラムを学習者の座席まで行くことなくその場で閲覧することができる．講師やティーチングアシスタントは机間巡回中に iPad のようなタブレットを用いて学習者のプログラミング状況を把握し必要があれば指導に行く．



(a) PROPEL のシステム構成図



(b) 講師用画面

図 3　プログラミング演習システム PROPEL

5.　システムによる名前付けの指導

表 2 からわかる通り，Java のコーディング規約に従わず，Camel 形式，Pascal 形式を用いていない名前付けが多数されている．これらは書式の確認をするだけなのでシステムで比較的簡単に検出することができる．その他のものは英単語の意味に関係するので単純な方法では検出できないが，本研究ではこれらについても取り組むことにする．

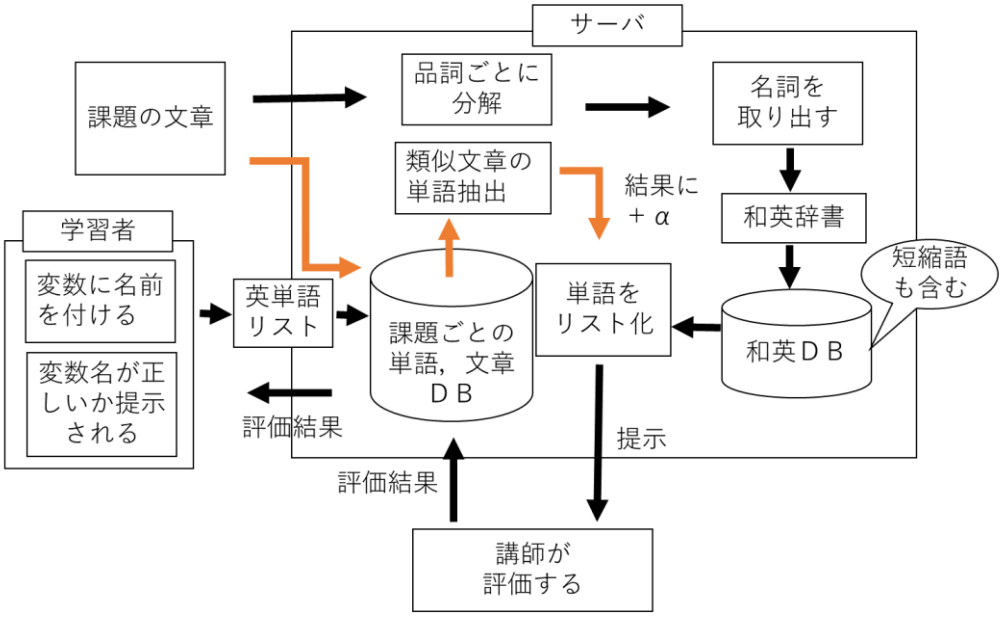


図 4　変数名指導モジュールのシステム構成図

5.1　名前の形式違反の指導

表 1 を見てわかる通り Camel 形式と Pascal 形式との違いは先頭のアルファベットが小文字か大文字かである. 学習者に多い間違いとして, 変数名を Pascal 形式で付けていることがある．これについてはシステムで以下の指導を行う.

- 表 1 の内容の説明を載せた Web ページを用意し名前の付け方の規則について学習者がいつでも確認できるようにする.
- 学習者の作成中のプログラムをシステムが定期的に, 具体的には 30 秒毎に監視し, プログラムのコード中に変数宣言またはメソッドの定義の開始行があれば，変数名・メソッド名が Camel 形式で書かれているかをチェックする. すなわち, 最初の英単語の語頭が小文字で, 2 番目以降の英単語の語頭が大文字であるかをチェックする．また，大文字で分かち書きされた英単語が英単語リストの中にあるかをチェックする. 名前が Camel 形式で書かれていない場合には，名前の付け方の形式に違反していると学習者に通知し，その場で修正させる.

5.2　誤字脱字の指導

学習者の英語の語彙力があまり高くないこともあって，綴り間違いをしているものがある．表 2 のように 1 割近くがこの間違いである. 例えば「average」を「avarage」,「height」を「haight」と間違っているものがある．これについてはシステムで以下の指導を行う.

名前の形式違反のチェックと，同時に，使われている文字列が英語単語リストにあるかどうかをチェックする．リストにない場合には，綴り間違いの可能性を調べるために，英単語リスト中の個々の単語との間で文字列の類似度を表す Levenshtein 距離[4]を計算し，距離が 1 ならば綴り間違いと判断する. Levenshtein 距離が 1 というのは文字列中で 1 文字が違っている，文字列中から 1 文字抜けている，などが相当する.

学習者への通知は，綴り間違いと判断した場合には，学習者の入力した文字列と綴り間違いだと判断した英単語リストの中の単語を併記して提示する. 英単語リストの中に類似した単語がない場合には，辞書中にそのような単語がないことを注意する.

5.3　意味が適切でない名前の指導

インターネットで公開されている翻訳ツールや辞書を使って，名前付けをする学習者が多い．検索の一番初めにでてきた単語をよく調べずに使用するのでこのような間違いが起こると考えられる．例えば「分散」と訳す英単語には「variance」「dispersion」などがあるが課題の内容によって使い分けなくてはならない.

これに関しては，課題の文章から形態素解析エンジン MeCab[5]を用いて名詞を抽出し，さらに，課題の文章の表現の特徴から変数名になりそうなものを絞り込む。その後，和英辞書を引いて，変数名になりそうな名詞，および，その英語表現（「分散」のように複数ある場合にはそれらのリスト）を講師に提示し，妥当な英語表現を講師に選択してもらい，それを用いて学習者の付けた名前をチェックする。

5.4　無意味な名前の指導

課題の内容と関係なく無意味な名前を付ける学習者がいる。特に，それらは一文字で付けていることが多い。

これに関しては，一文字の変数名でもよい課題のときは講師にそれを指示してもらい，それ以外の課題のときは不適切な名前としてチェックする。

6. 実装

上記で説明した指導のうち，名前の形式違反の指導に関しては実装を完了し，運用を始めた。

誤字脱字，意味が適切でない名前，無意味な名前への指導に関しては図 4 のシステムを実装することで対策する．学習者への提示はコーディングするエディタの上部に名前付けに関する注意を示す枠を作成し，提示する．その図を図 5 に示す．図 6 は拡大図である．その変数名がどのような理由で指導対象になっているかを明確に示すことにより，学習者自身に訂正させる．

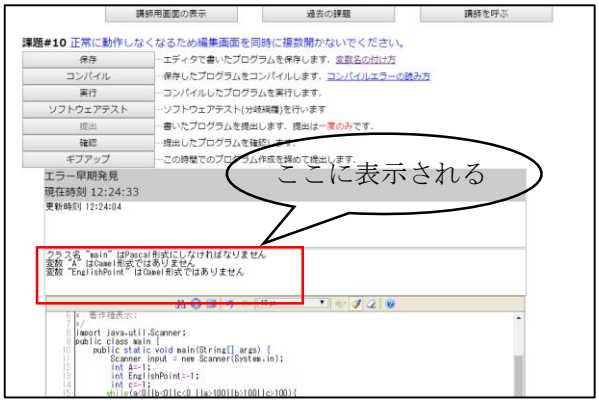


図 5　変数名の指摘（学習者側）

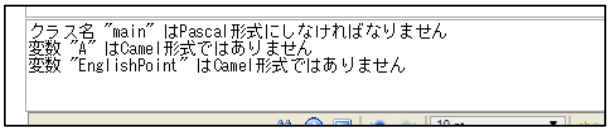


図 6　変数名の指摘（拡大）

7. 指導の効果

名前の形式違反の指導に関しては，実装が終わり運用を行っているので，その結果を表 3 に示す．表 2 で分かるように従来は学習者が付けた名前のうち約 42%が名前の形式違反であったが，それを約 6%に減らすことができた．それによって，従来は変数名のうち半数近くが不適切な名前であったものを，1 割以下に減らすことができた．

表 3　名前の形式違反の指導

内容		個数
変数名の合計		395 個
不適切な変数名		32 個
	Camel 形式，Pascal 形式でない	25 個
	誤字	4 個
	意味が不適切	0 個
	無意味な名前	7 個

8. まとめ

プログラミング演習において，従来は名前付けに関して十分な指導を行うことができず，学習者が書くプログラム中の変数名の約半数が命名規則に則したものではないという問題があった．本研究では，講師の負担を増やすことがなく，システムによって名前付けの指導を行うことを目的としている．実装が終わり運用を行っている名前の形式違反の指導によって，不適切な名前付けを 1 割以下に減らすことができた．他の指導を実装し運用することで現時点では指導できていないものについても不適切な名前が減ることを期待している．

参考文献

[1] 上村拓磨，北英彦：プログラミングスタイル習得のための自己学習環境，コンピュータ利用教育議会，2016PC カンファレンス（2016）

[2] Brain W.Kernighan，Rob Pike，訳：福崎俊博，プログラミング作法，17-51，株式会社アスキー，2004

[3] Sun Microsystems: https://www.oracle.com/sun/ （2017 年 6 月）

[4] Damerau-Levenshtein Distance in Python: https://www.guyrutenberg.com/2008/12/15/damerau-levenshtein-distance-in-python （2017 年 6 月）

[5] MeCab Yet Another Part-of-Speech and Morphological Analyzer: http;//taku910.github.io/mecab/ （2017 年 6 月）

[6] 伊冨昌幸，北英彦，高瀬治彦，林照峯：コーディング状況に応じたアドバイスを可能にするプログラミング演習システムに関する研究，コンピュータ利用教育議会，2010PC カンファレンス（2010）

[7] 戸上稔崇，北英彦：プログラミング演習システム PROPEL の Java 対応とエラーメッセージ改善，コンピュータ利用教育協議会，2015PC カンファレンス（2015）

[8] 北英彦：プログラミングスタイルの収得のための自己学習機能，コンピュータ利用教育議会，2014PC カンファレンス（2014）