情報処理学会論文誌　Vol. 52　No. 11　3065–3078 (Nov. 2011)

# プログラミング入門教育における インデントと実装能力の関係

高野辰之[†1]　宮川　治[†2]　小濱隆司[†2]

プログラミングにおけるインデントとプログラミング能力の関係が従来から研究されてきた．従来研究では，インデントの効果がプログラミング能力の読解において確認されている．また，インデントとプログラミング能力である実装能力の関係について議論されている．そこで，本研究ではプログラミング入門教育科目の定期試験の解答からインデントと実装能力の関係を分析した．その結果，いくつかの項目で低い相関がみられたことから，インデントを正しく行える能力とプログラムを正しく書ける能力には何らかの関係があるのではないか，と考えられる．

# The Relation between the Indentation and the Ability to Implement in Introductory Programming Courses

Tatsuyuki Takano,[†1] Osamu Miyakawa[†2] and Takashi Kohama[†2]

The relation between the indentation and the programming skill is researched. The effect is confirmed with the reading skill using the indentation. In our approach, the aspect is put on the description of the program. In this paper, the examination of the programming subjects for the beginner was analyzed. As a result, it may be related to the indentation and ability to implement programs.

†1 東京電機大学大学院先端科学技術研究科
Graduate School of Advanced Science and Technology, Tokyo Denki University
†2 東京電機大学情報環境学部
School of Information Environment, Tokyo Denki University

## 1. はじめに

プログラミングにおいて従来からインデントはプログラムの構造を明確にする手法として用いられてきた．

そして，ソフトウェア産業では各種プログラミング言語に対してコーディング規約という形でプログラムを作成する際のプログラムの整形から命名法まで定めたものを活用しており，そのなかにインデントに関する項目も含まれている．これらコーディング規約の普及により，プログラムに対するインデントは一般に認知されている．

インデントはプログラムの構造を明確にすることから，プログラミングの入門教育において教材のプログラムに活用されている．インデントは，その深さにより論理構造や変数のスコープを理解する助けになっている．

そして，従来からインデントとプログラミング能力との関係が研究されている．ここでのプログラミング能力とはプログラムの読解力と実装能力に分けられる．

まず，プログラムの読解力を主体としたインデントとの関係の研究について述べる．

Shneiderman[1] はインデントをしたプログラムとしていないプログラムについて比較する実験を行った．実験内容は，2 つのプログラムについて，それぞれインデントしたもの・していないものによって，あらかじめ挿入した 1 つのバグを見つけさせることである．その結果，バグを見つけることにおいての差はみられなかったと報告している．

その後，Miara ら[2] は Shneiderman とともにインデントされたプログラムを学生に読ませ，インデントされた場合は，そうでない場合と比較して，プログラムに対する読解力に有意な差があることを示している．

そして，Oman ら[3] は if・else 節を含むプログラムに対して，構造を反映したインデントされたものとそうでないものを読解させた場合では，読解力に有意な差があることを確認している．

したがって，Miara らと Oman らは構造化プログラミングにおいて，インデントがプログラムの読解力の向上につながる要素であることを示している．

次に，プログラムの実装能力を主体としたインデントとの関係の研究について述べる．

Sheil[4] はインデントと実装能力に関する研究として Weissman[5] と Love[6] による結果を報告している．Weissman は ALGOL-W と PL/I によるプログラムでの編集における評価を行っている．また，Love は FORTRAN によるプログラムの再構築作業について評価を行っている．

**表 1**　インデントとの関係
Table 1　The relationship of the indentation.

| 発表年 | 著者 | 読解力 | 実装能力 | 備考 |
|---|---|---|---|---|
| 1974 | Weissman | | × | 編集 |
| 1977 | Love | | × | 再構築 |
| 1980 | Shneiderman | × | | 読解（バグ発見） |
| 1981 | Sheil | ×（※） | ×（※） | サーベイ論文 |
| 1983 | Miara | ○ | | 読解 |
| 1988 | Oman | ○ | | 読解 |

○：有意な差を確認，×：有意な差が確認されず，空欄：判別なし
（※）：他論文の結果により判断

これらの結果から，インデントがプログラムの読解力と実装能力に対して効果がみられなかったとしている．そして，Sheil は Shneiderman の報告[1) ]もふまえ，プログラムにおけるインデントは読解力，実装能力ともその効果はみられなかったと考察している．

**表 1** は先行研究を一覧表にしたものである．これらの結論の違いは，対象プログラムや評価方法の差異によるものと考えられる．

また，これらの研究は，あらかじめ入力されたプログラムに対するインデントとの関係である．そこで，本研究では白紙からプログラムを構築する実装能力とインデントの関係を調べる．

本稿では 2 章で本研究での仮説の設定，3 章で仮説の検証するための分析方法，4 章で実際の分析，5 章で分析で得られた結果，6 章で結果に対する考察，7 章で本論でのまとめを述べる．

## 2. 仮　　説

インデントと実装能力の関係を検証するため，2 つの仮説を立てる．1 つ目は「インデントを適切に使用してコーディングできることと実装能力との関係はない」という帰無仮説である．さらにプログラミング入門教育では，学習者における事前のプログラミング経験の有無は実装能力に対して大きな影響を与えると推測される．そして，インデントと実装能力の関係を補足するため，2 つ目は「入学前におけるプログラミング経験は実装能力との関係はない」という帰無仮説である．

## 3. 分 析 方 法

プログラミング入門教育におけるインデントと実装能力の関係を分析する方法を述べる．

まず，学習者にプログラムをコーディングさせる．そして，学生が提出したプログラムのインデントの検査とユニットテストを行い，これらの結果を分析する．ユニットテストとは，メソッド単位で期待した動作を行うか判断するものである．プログラミング経験の有無の情報は受講前にアンケートを実施することによって収集する．

### 3.1 対 象 科 目

今回は，プログラミング入門教育の科目を対象とする．本学部における対象科目は「コンピュータプログラミング A」（以下 CA と略す）と「コンピュータプログラミング B」（以下 CB と略す）である．CA は第 2 セメスタ，CB は第 3 セメスタに配当された選択科目である．これらの科目は，情報技術を学ぶための基礎科目に該当し，ほとんどの学生が履修する．プログラミング言語は Java を使用し，それぞれの科目の教育内容は，CA では構造化プログラミングを，CB ではオブジェクト指向プログラミングを扱っている．

また，これらの科目は，パソコンを使用してプログラムをコーディングさせる演習が含まれる．履修者が多いため，クラス分割をしている．クラス分割の方法は学籍番号を基準としたものであり，学生の習熟度などは関係していない．

#### 3.1.1 講義・演習形式

講義では Web 上に用意したテキストを使用する．テキストの基本的な構成は本の見開きをイメージしており，左側にプログラムや仕様について，右側にプログラムの解説や演習問題を掲載している．仕様については UML（Unified Modeling Language）のクラス図を基本とした図とプログラムの API（Application Programming Interface）仕様について記述した表を用いている．仕様として掲載しているクラス図の例を**図 1** に，API 仕様を**図 2** に示す．

このテキストを大型スクリーンに映し出し解説をする．そして，教員はノートパソコンを用いて，プログラムのコーディング・実行過程をスクリーンに映し，学生に見せる．また，テキストで用いるプログラムや教員がコーディングするプログラムのコーディングスタイルは統一している．

学生はノートパソコンを持参し，Web 上のテキストを参照できる環境で受講する．そして，教員の指示に従って演習を進める．演習では，スクラッチプログラミングを指導している．スクラッチプログラミングとは，新規にファイルを作成し，プログラムをコーディングすることである．

学生がプログラムを作成する手順は次のとおりである．

(1)　クラス図を読む

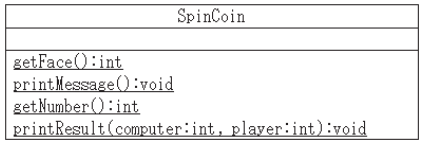


**図 1**　クラス図の例
Fig. 1　An example of class diagram.

| API 仕様 | |
|---|---|
| getFace | 計算機がコインの表裏を乱数で決め，<br>その数を返却します．<br>表は 1，裏は 0 とします． |
| printMessage | ”表(1)ですか，裏(0)ですか？”を表示します．<br>改行はしません．<br><br>（実行例）<br>表(1)ですか，裏(0)ですか？ |
| getNumber | キーボードから表裏を数字で入力し，<br>その数を返却します． |
| printResult | 計算機が決めた数と，<br>人間が入力した数を引数で受け取ります．<br>一致している場合は「当たり」を，<br>一致していない場合は「外れ」を表示します． |

**図 2**　API 仕様の例
Fig. 2　An example of API specification.

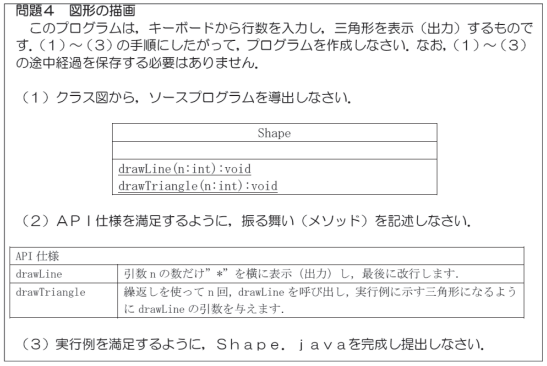


**図 3**　問題例
Fig. 3　An example of examination paper.

(2)　機械的導出
(3)　API 仕様を読む
(4)　実装

機械的導出とは，クラス図に示されている情報からプログラムの骨組み（コンパイルが可能な最低限のプログラム）をコーディングすることである．教員は，これらの段階を確認しながら授業を進行し，指導を行っている．また，機械的導出や実装の際にはインデントを意識するように指導している．

演習の際には SA（Student Assistant）や TA（Teaching Assistant）が在室して，巡回しながら学生の理解の補助をする．

### 3.2　試験形式

定期試験は，仕様に従ってプログラムを作成させ，そのプログラムを提出させる形式である．試験問題は紙によって出題される．参考書の参照はできるが，ネットワークの使用は禁止している．

試験問題の例を**図 3** に示す．各問題は基本的にプログラムの作成手順を示した問題形式である．学生は試験時間内に仕様を満たすプログラムを持参したノートパソコンでコーディングする．また，問題には動作を確認するためのプログラムと実行例を掲載しているため，学生は実際の動作を確認することができる．そして，各学生に USB メモリを配布し，コーディングされたプログラムを回収する．

### 3.3　試験の評価方法

試験の評価によって得られたデータを分析するにあたり，プログラムを評価する判定基準を明確に設定する．プログラムの評価は，提出されたプログラムをコンパイルして，ユニットテストによってプログラムの実行結果を判定する．また，インデントは各プログラムのファイルごとに判定をする．

プログラミングの初学者が白紙の状態からプログラムを作成するため，スペルミスに代表されるさまざまなエラーが発生する．次項からその判定基準の詳細とエラーへの対処について述べる．

#### 3.3.1　インデント

インデントは，プログラムのブロック構造を明確にし，それにともない変数などのスコープの範囲を明瞭にする．また，インデントはスペースやタブによってプログラムの入れ子になったブロック構造を示し，その字下げ幅によって入れ子の深さを示す．今回は，プログラムの構造を同じ字下げ幅によってコーディングしているものを正しいインデントとして評価する．

インデントの幅を半角スペース 8 個分にしてコーディングした例を**図 4** に示す．図 4 の class A は適切なインデントとし，class B と class C はインデントが誤っていると評価する．この例では 1 つのタブによってインデントをしているが，複数のタブによってインデントをしていても同じ字下げ幅であれば，適切にコーディングされていると評価をする．また，半角スペースによるインデントでも同様である．

```
public class A{
        public void print(){
                System.out.print("a");
        }
}

public class B{
        public void print(){
                        System.out.print("b");
        }
}

public class C{
        public void print(){
        System.out.print("c");
        }
}
```

**図 4**　インデントのコーディング例
Fig. 4　Examples of indentation.

提出されるプログラムには，タブと半角スペースを混在してインデントされたものがある．その扱いについては 3.3.3.4 で説明する．

### 3.3.2　ユニットテスト

ユニットテストは，プログラムのメソッドが正しく動作するかを判定するため，実際にプログラムを実行して評価をすることである．提出されたプログラムは，正しく実装されたプログラムの動作と比較して評価される．

その評価方法は「返却値」と「コンソールへの出力」,「値の受け取り」に分けられる．以下では，それぞれの場合についての評価方法を説明する．また，キーボードの入力や乱数，Java の標準 API を利用する場合はプログラムの機械的な書き換えを行う．その書き換え方法についても述べる．

#### 3.3.2.1　返　却　値

返却値が基本型や文字列の場合は期待される値と比較することにより正誤を判断する．引数が必要な場合は出題した仕様に従って，適切な値を渡す．また，文字列以外のオブジェクトが返却値となる場合は，そのオブジェクトの各インスタンス変数の値をそれぞれ比較し評価する．また，返却する値がコンストラクタによって渡された値に関係する場合はコンストラクタでの値の受け渡し後，正しい値を返却するかを評価する．

#### 3.3.2.2　コンソールへの出力

返却値がない場合は主にコンソールへ出力するメソッド，または引数として渡された値を保持するメソッドとなる．引数が必要な場合は出題した仕様に従って，適切な値を渡す．ここではコンソールへの出力についての評価方法を述べる．

```
：。、　？＊（）０１２３４５６７８９＜＞／
:.,　?*　()０１２３４５６７８９＜＞/
```

**図 5**　全角文字と半角文字の対応例
Fig. 5　An example of correspondance between a full-width and a half-width character.

コンソールへ出力するメソッドは，紙によって出力例を示すことから，文字列の並びに対してある程度読み取りに個人差が生じる．**図 5** に全角文字と半角文字の対応例を示す．この図が示すように印刷されたある文字に関して，全角文字と半角文字かを判別することは容易ではない．したがって，問題文の文字の読み取りに対する誤差は，実装能力と関係がないものとし，ユニットテストの評価は以下の手順で行う．

- 表示された文字列に対して同様の文字を意味する全角文字を半角文字に統一する．
- スペースと改行を取り除く．
- 出力される文字列と解答の類似度を比較して一定基準以上かを判断する．

なお，文字列間の類似度については後述する．

#### 3.3.2.3　値の受け取り

引数として渡された値を保持する場合の評価方法について説明する．値を渡すメソッドの評価は，プログラムが正しく値を受け取れているかを判定する．まず，保持された値を取得できるメソッドがある場合は，そのメソッドを使用して正しい値が返却されるかを判定する．つまり，間接的に値が取得可能な場合は返却値で評価する．

間接的に値を取得できない場合や Java の標準 API などを使用して値を保持している場合は，プログラムの書き換えをした後，評価する．

#### 3.3.2.4　プログラムの書き換え

キーボード入力や乱数，Java の標準 API を利用するプログラムのユニットテストでは，学生のプログラムを機械的に書き換えることにより評価する．

本学では，キーボード入力や乱数の API を教員が用意している．このような教員が用意した API の中身を書き換えることにより，ユニットテストに必要な値を返却できるようにしている．

Collection などの標準 API を用いたプログラムの場合は import 文の書き換えを行い，評価用に用意したクラスを利用するプログラムに変更する．

たとえば，標準 API の java.util.ArrayList を評価用のパッケージである test.ArrayList に書き換える．この書き換えにより，ユニットテストで使用する値を評価用クラスにあらかじめ初期値として保持したり，返却したりする．また，ユニットテストで求められる値が正しく渡されているか判定可能となる．

#### 3.3.3　不適応プログラムへの対処

試験で提出されたプログラムは，スクラッチプログラミングすることから，プログラムの各所でスペルミスなどの誤りが含まれる．そのため，これまでの評価方法では適応できないさまざまなプログラムが存在する．ここでは，それらの対処方法を説明する．

##### 3.3.3.1　スペルミス箇所の特定

コーディングによるスペルミスの主な箇所としては，ファイル名，クラス名，メソッド名などがある．これらの箇所にスペルミスがある場合，正しくプログラムを評価できない．したがって，対象箇所を特定する必要がある．そこで，判定手順を以下のように定めた．

- 名前のリストを作成する．
- 名前が完全に一致している場合はリストから取り除く．
- リストに残った名前に対して類似度を測定する．
- 類似度が基準以上であれば類似したものに対する解答として扱う．

メソッド名でのスペルミス判定手順を**図 6** に示す．

##### 3.3.3.2　文字列の類似度

文字列の類似度を測定するためにレーベンシュタイン距離を使用した．レーベンシュタイン距離とは文字列間の編集距離を測定するもので，ある文字列から目的の文字列まで何文字編集することで到達できるかを算出するアルゴリズムである．たとえば，「student」という文字列を「Student」にするためには「s」を「S」に置換する必要がある．この編集における挿入，削除，置換の手順数を合計し，距離とするのがレーベンシュタイン距離である．距離における編集の手順の重み付けは目的や実装によって異なる．今回の類似度は挿入，削除，置換の重み付けをすべて 1 とするアルゴリズムを使用した．また，距離の合計を 2 つの文字列の長い方の字数で割り，その値を 1 から引いた値を類似度としている．つまり，類似度は正規化され，範囲は 0.0 から 1.0 の値をとる．例として示した「student」と「Student」の類似度は 0.857 の値をとる．そして，プログラムの評価には文字列が類似していると判断する基準を 0.8 以上と設定している．

##### 3.3.3.3　ユニットテストのスペルミス処理

ユニットテストにおいてクラス名やメソッド名にスペルミスが存在するとプログラムの評価ができない．そのため，コンパイルの評価でプログラムがコンパイル可能だったものを抽出し，クラス名やメソッド名が類似する場合についてスペルミス処理をする．スペルミスを許容してのユニットテストの例を**図 7** に示す．

図 7 (a) は CA における基本的なユニットテストの例であり，「クラス」と「メソッド」と表記されている部分がスペルミス判定によって変化を許容する部分である．同様に，図 7 (b) では CB におけるユニットテストの例であり，「コンストラクタ」と「メソッド」の部分がスペルミス判定によって文字列の変化を許容する部分である．

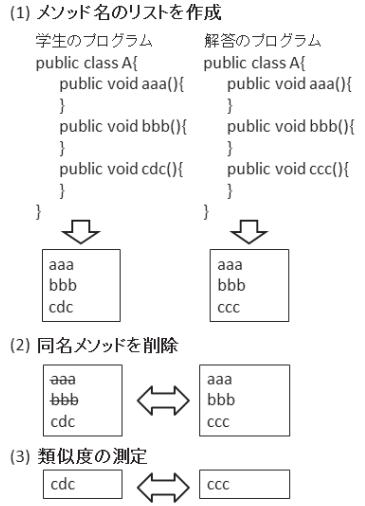


**図 6**　メソッドでのスペルミス判定手順
Fig. 6　Spelling error judging procedure in method.

```
Object result =「クラス」.「メソッド」(「引数」);
```

(a) CA での例

```
「クラス」 obj = new「コンストラクタ」(「引数1」);
Object result = obj.「メソッド」(「引数2」);
```

(b) CB での例

**図 7**　ユニットテスト例
Fig. 7　Example of unit test.

```
public␣class␣D{
␣␣␣␣␣␣␣␣public␣void␣print(){
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣System.out.print("d");
␣␣␣␣␣␣␣␣}
}
```

(a) タブ幅が 8 スペースの例

```
public␣class␣D{
␣␣␣␣public␣void␣print(){
␣␣␣␣␣␣␣␣␣␣␣␣System.out.print("d");
␣␣␣␣}
}
```

(b) タブ幅が 4 スペースの例

**図 8**　タブと半角スペースの混在例
Fig. 8　Example of mixture of tab and space.

#### 3.3.3.4　タブと半角スペースの混在

インデントは，タブと半角スペースの混在を許容して評価をする．混在を許容する理由は本研究の分析として学生が正しくインデントを認識してプログラムを作成しているかを問うことに目的があるからである．

本来はタブか半角スペースに統一するべきだが，学生のエディタ上で見た目は正しくても混在している場合がある．混在している例を**図 8** に示す．図 8 (a) はタブが半角スペース 8 個分の場合であり，このときエディタの見た目では正しくインデントされていると判断できる．しかし，タブ幅はエディタでの設定によって見た目が変化するため，たとえば別のエディタでタブが半角スペース 4 個分の場合は図 8 (b) のようにインデントがずれる．

そのため，学生からは適切なインデントによるコーディングと見えても，教員のエディタでは正しいインデントに見えない場合がある．今回はインデントについて，学生が理解していることが重要であるため，タブと半角スペースの混在があった場合でも学生はインデントを正しく理解していると判断する．

そして，タブの範囲を半角スペースが 1 個から 10 個までの場合でインデントを検査し，正しくインデントされているかを評価する．

#### 3.3.3.5　コンパイル不可の場合のインデント評価

提出されたプログラムには意味エラーのためにコンパイルエラーとなるものがある．インデント評価では意味エラーの有無によらず評価を行う．意味エラーとは，プログラムの構文解析は可能だがコンパイルできないものを指す．**図 9** の (a) に意味エラーとなるプログラムを，(b) にコンパイルエラーのメッセージを示す．図 9 では，Java の文法上において構文解析が可能な記述である．しかし，F クラスが存在しないため，コンパイルエラーとなっていることが示されている．

```
public class E{
  public void print(){
    F.print();
  }
}
```

(a) プログラム

```
E.java:3: シンボルを見つけられません。
シンボル: 変数 F
場所    : Eの クラス
    F.print();
    ^
エラー 1 個
```

(b) エラーメッセージ

**図 9**　意味エラーの例
Fig. 9　An example of semantic error.

このように，存在しないクラス名やメソッド名，または変数名などのコーディングによって，コンパイルエラーが引き起こされる場合がある．

しかし，プログラムの構造はインデントによって明確にコーディングされていることから，このようなプログラムを提出した学生はプログラムの構造を理解し，適切にインデントをコーディングしていると評価する．

## 4. 分　　析

本研究では 2009 年度 CA と 2010 年度 CB の両科目を受講した再履修を含まない学生 215 名を対象として，それぞれの定期試験の結果を分析した．分析は各問題として出題されたプログラムごとにインデントとユニットテストによるクロス集計表を作成する．例を**表 2** に示す．そして，各クロス集計表に対してフィッシャーの正確確率検定を行い，p 値（表では p として示す）を求める．表 2 における p 値は式 (1) により計算する．

$$\mathrm{p} = \frac{(a+b)!(c+d)!(a+c)!(b+d)!}{a!b!c!d!(a+b+c+d)!} \tag{1}$$

また，ファイ係数（$2 \times 2$ のクロス集計表におけるピアソンの積率相関係数，表では $\phi$ として示す）は式 (2) により求める．

$$\phi = \frac{ad - bc}{\sqrt{(a+b)(c+d)(a+c)(b+d)}} \tag{2}$$

集計におけるインデントは，すべてのプログラムのインデントを適切にコーディングできた学生の集団と，それ以外の学生の集団とを分ける．ただし，対象となるプログラムが提出さ

**表 2**　クロス集計表例
Table 2　Example of cross table.

| | | 要素 A | | 計 |
|---|---|---|---|---|
| | | True | False | |
| 要素 B | True | a | b | a + b |
| | False | c | d | c + d |
| 計 | | a + c | b + d | a+b+c+d |

$\phi$, p

れていない場合やコンパイルの評価ができない場合はそのプログラムにおけるインデントは評価はできない．このため，分析する際の対象から除外している．つまり，すべての問題に対してインデントが正しくできている学生を「インデントを適切にコーディングできる学生」とした．そして，インデントを適切にコーディングできる集団とそうでない集団に分類した．その結果，CA では 160 名，CB では 186 名を分析の対象とした．

クロス集計表はインデントとユニットテストの組合せを CA と CB において分析をした．なお，ユニットテストを行わないメソッドやインタフェースなどは除外する．

また，CA のインデントと CB のユニットテストにおけるクロス集計表も作成し，両科目を通した分析を行う．さらに，CA の受講前に学生へアンケートを実施し，入学前のプログラミングの経験の有無についてを調査する．プログラミングの経験がある場合を True，ない場合を False として，プログラミングの経験とユニットテストのクロス集計表も作成し分析する．なお，これらのクロス集計表はアンケートに回答した学生（202 名）を分析対象とする．

CA と CB で実施した試験問題における各メソッドの複雑度を示したものをそれぞれ**表 3** と**表 4** に示す．

表における NCSS（Non Commenting Source Statements）はコメントを除いたコードの行数であり，CCN（Cyclomatic Complexity Number）は McCabe[7] による繰返しや条件文などに基づいた複雑度の指数である．また，この CCN は 1 を最小とし，数値が大きいほど複雑なメソッドであることを示す．

表 3 と表 4 における NCSS の値は，現実に作成されるプログラムに比べて非常に小さい値となっている．これはオブジェクト指向におけるモジュール性[8] を意識した内容で教育を行っているため，小さい機能のメソッドでプログラムを構成している結果である．

CA では，このことを意識した構造化プログラミングの設計を扱い，CB への基礎としている．そして，CB ではオブジェクト指向におけるオブジェクトの状態の理解が求められる内容となっており，状態の取得や変更に関するメソッドを扱うため CA より NCSS の値が小さいものが多く存在する．

**表 3**　振る舞いの複雑度（CA）
Table 3　Complexity of method (CA).

| NCSS | CCN | 問題（クラス名，メソッド名）<br>メソッドの概要 |
|---|---|---|
| 4 | 1 | Cube main<br>3 乗を求める |
| 2 | 1 | Score main<br>getSum の返却値を表示 |
| 6 | 2 | Score getSum<br>配列の値の総和を計算 |
| 3 | 1 | SpinCoin getFace<br>乱数を使用して 0 か 1 を返却 |
| 2 | 1 | SpinCoin printMessage<br>キーボード入力を促す表示 |
| 3 | 1 | SpinCoin getNumber<br>キーボード入力による値の返却 |
| 5 | 2 | SpinCoin printResult<br>2 つの引数の値が同値かを判定 |
| 4 | 2 | Shape drawLine<br>繰り返しを用いた文字の表示 |
| 3 | 2 | Shape drawTriangle<br>drawLine を使用した表示 |

**表 4**　振る舞いの複雑度（CB）
Table 4　Complexity of method (CB).

| NCSS | CCN | 問題（クラス名，メソッド名）<br>メソッドの概要 |
|---|---|---|
| 2 | 1 | LooseLeaf getItem<br>オブジェクトの状態を返却 |
| 2 | 1 | LooseLeaf getColor<br>オブジェクトの状態を返却 |
| 2 | 1 | LooseLeaf getNumber<br>オブジェクトの状態を返却 |
| 5 | 2 | FlatFile bind<br>オブジェクトをリストに追加 |
| 2 | 1 | FlatFile remove<br>オブジェクトをリストから削除 |
| 2 | 1 | CdPocket getItem<br>オブジェクトの状態を返却 |
| 2 | 1 | CdPocket getColor<br>オブジェクトの状態を返却 |
| 2 | 1 | CdPocket getNumber<br>オブジェクトの状態を返却 |
| 7 | 2 | FlatFile printList<br>リストにあるすべての<br>オブジェクトを表示 |

## 5. 結　　果

分析した結果を表 5 から表 11 に示す．**表 5** は CA のインデントと CA のユニットテストによるクロス集計表の結果，**表 6** は CB のインデントと CB のユニットテストによるクロス集計表の結果である．**表 7** は入学前のプログラミング経験の有無と CA のユニットテストによるクロス集計表の結果，**表 8** は入学前のプログラミング経験の有無と CB のユニットテストによるクロス集計表の結果である．**表 9** は CA のインデントと CB のユニットテストによるクロス集計表の結果である．**表 10** は CA のインデントと入学前のプログラミング経験の有無によるクロス集計表，**表 11** は CB のインデントと入学前のプログラミング経験の有無によるクロス集計表である．

表 5 から表 9 は，ユニットテストの単位であるメソッドごとにクロス集計表を作成し，分

**表 5**　CA インデントと CA ユニットテストの関係
Table 5　Relation between indentation (CA) and unit test (CA).

| | $\phi$ | 正答率 | メソッド |
|---|---|---|---|
| | 0.0201 | 0.694 | Cube main |
| * | 0.164 | 0.831 | Score main |
| * | 0.177 | 0.788 | Score getSum |
| | 0.129 | 0.744 | SpinCoin getFace |
| | 0.153 | 0.875 | SpinCoin printMessage |
| | 0.0778 | 0.750 | SpinCoin getNumber |
| ** | 0.260 | 0.731 | SpinCoin printResult |
| ** | 0.218 | 0.481 | Shape drawLine |
| ** | 0.217 | 0.419 | Shape drawTriangle |

* $p < 0.05$, ** $p < 0.01$, n = 160

**表 6**　CB インデントと CB ユニットの関係
Table 6　Relation between indentation (CB) and unit test (CB).

| | $\phi$ | 正答率 | メソッド |
|---|---|---|---|
| | 0.154 | 0.963 | LooseLeaf getItem |
| | 0.0234 | 0.973 | LooseLeaf getColor |
| | 0.102 | 0.989 | LooseLeaf getNumber |
| | 0.0750 | 0.257 | FlatFile bind |
| | −0.0466 | 0.588 | FlatFile remove |
| | 0.110 | 0.845 | CdPocket getItem |
| | 0.0794 | 0.882 | CdPocket getColor |
| | 0.0913 | 0.861 | CdPocket getNumber |
| * | 0.175 | 0.529 | FlatFile printList |

* $p < 0.05$, ** $p < 0.01$, n = 187

**表 7**　プログラミング経験と CA ユニットの関係
Table 7　Relation between programming experience and unit test (CA).

| | $\phi$ | 正答率 | メソッド |
|---|---|---|---|
| | 0.0936 | 0.663 | Cube main |
| | 0.126 | 0.718 | Score main |
| * | 0.148 | 0.663 | Score getSum |
| | 0.0712 | 0.658 | SpinCoin getFace |
| | 0.0184 | 0.767 | SpinCoin printMessage |
| | 0.0346 | 0.668 | SpinCoin getNumber |
| | 0.134 | 0.649 | SpinCoin printResult |
| * | 0.166 | 0.396 | Shape drawLine |
| ** | 0.270 | 0.327 | Shape drawTriangle |

* $p < 0.05$, ** $p < 0.01$, n = 202

**表 8**　プログラミング経験と CB ユニットの関係
Table 8　Relation between programming experience and unit test (CB).

| | $\phi$ | 正答率 | メソッド |
|---|---|---|---|
| | 0.0962 | 0.960 | LooseLeaf getItem |
| | 0.0897 | 0.965 | LooseLeaf getColor |
| | 0.0581 | 0.985 | LooseLeaf getNumber |
| ** | 0.277 | 0.218 | FlatFile bind |
| | 0.108 | 0.535 | FlatFile remove |
| | −0.109 | 0.772 | CdPocket getItem |
| | −0.0405 | 0.817 | CdPocket getColor |
| | 0.0162 | 0.797 | CdPocket getNumber |
| | 0.139 | 0.475 | FlatFile printList |

* $p < 0.05$, ** $p < 0.01$, n = 202

析した結果を集計したものである．表の各行は定期試験における問題の各ユニットテストに対してインデントの正否とのクロス集計表を作成し，検定を行った結果を示したものである．また，クロス集計表における人数を n で示す．有意水準 5%（$p < 0.05$）で有意な差が確認されたものは，アスタリスク（*）で示す．また，有意水準 1%（$p < 0.01$）で有意な差が確認されたものは，アスタリスク 2 個（**）で示す．$\phi$ はクロス集計表におけるファイ係数である．ファイ係数は −1 から 1 までの範囲をとる．したがって，有意な差がある場合ファイ係数は一般的な相関係数の解釈が適用できる．正の場合を例にとると，0 から 0.2 は「ほとんど相関がない」，0.2 から 0.4 は「低い正の相関がある」，0.4 から 0.7 は「かなり正の相関がある」，0.7 以上は「高い正の相関がある」と解釈される．そして，今回はファイ係数が 0.2 以上の場合には，下線を引いて示す．

**表 9**　CA インデントと CB ユニットの関係
Table 9　Relation between indentation (CA) and unit test (CB).

| | $\phi$ | 正答率 | メソッド |
|---|---|---|---|
| | 0.0928 | 0.950 | LooseLeaf getItem |
| | 0.151 | 0.963 | LooseLeaf getColor |
| | 0.106 | 0.981 | LooseLeaf getNumber |
| * | 0.196 | 0.269 | FlatFile bind |
| | 0.101 | 0.594 | FlatFile remove |
| * | 0.194 | 0.813 | CdPocket getItem |
| ** | 0.245 | 0.850 | CdPocket getColor |
| ** | 0.245 | 0.850 | CdPocket getNumber |
| | 0.150 | 0.531 | FlatFile printList |

* $p < 0.05$, ** $p < 0.01$, n = 160

**表 10**　CA インデントとプログラミング経験のクロス集計表
Table 10　Relation between indentation (CA) and programming experience.

| | | CA インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| プログラミング経験 | True | 23 | 8 | 31 |
| | False | 84 | 39 | 123 |
| 計 | | 107 | 47 | 154 |

$\phi = 0.0514$, $p = 0.664$

**表 11**　CB インデントとプログラミング経験のクロス集計表
Table 11　Relation between indentation (CB) and programming experience.

| | | CB インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| プログラミング経験 | True | 28 | 4 | 32 |
| | False | 122 | 24 | 146 |
| 計 | | 150 | 28 | 178 |

$\phi = 0.0415$, $p = 0.789$

正答率は各ユニットテストの正解数を学生数で割ったものであり，0.0 から 1.0 に値を正規化している．問題は対象のクラス名とメソッド名を示している．各行の順番は試験における出題順となっている．基本的に問題はプログラムのクラス単位で出題しているため，クラスに含まれるメソッドは連続して出題される．しかし，表 6，表 8，表 9 にある FlatFile の printList メソッドは他のクラスのプログラムをすべて作成した後に，FlatFile クラスに printList メソッドを追加するという問題を出題しているため，各表の最後に記されている．

これらの表から，各表のすべての項目において次の結果が得られた．

- 有意水準 5%で有意な差が確認された項目は，ファイ係数によると「ほとんど相関がない」であった．
- 有意水準 1%で有意な差が確認された項目は，ファイ係数によると「低い正の相関」であった．

そこで，特に有意水準 1%で有意な差が確認された場合を「低い相関を示す有意な差」と記す．そして，個々の表における結果を次に示す．

- 表 5 より CA のインデントの正否とユニットテストの正否に対し 9 項目中 3 項目において低い相関を示す有意な差が確認された．
- 表 6 より CB のインデントの正否とユニットテストの正否に対し 9 項目中 1 項目において有意水準 5%で有意な差が確認された．しかし，ファイ係数によると「ほとんど相関のない」であった．
- 表 7 よりプログラミングの経験とユニットテストにおける分析結果から，9 項目中 1 項目に低い相関を示す有意な差が確認された．
- 表 8 よりプログラミングの経験とユニットテストにおける分析結果から，9 項目中 1 項目に低い相関を示す有意な差が確認された．
- 表 9 よりプログラミング入門科目である CA でのインデントの正否が CB のユニットテストにおける分析結果から，9 項目中 2 項目に低い相関を示す有意な差が確認された．
- 表 10 より CA のインデントの正否はプログラミングの入学前の経験とは有意な差はみられなかった．
- 表 11 より CB のインデントの正否はプログラミングの入学前の経験とは有意な差はみられなかった．

以上のことから本研究における仮説を検証する．

CA においてインデントの正否とユニットテストの正否によって分析した表 5 より，低い相関を示す有意な差のある項目が確認された．このことから，仮説「インデントを適切に使用してコーディングできることと実装能力との関係はない」を棄却する．

また，プログラミング経験と CA・CB のユニットテストの正否を分析した表 7・表 8 より，低い相関を示す有意な差のある項目が確認された．このことから，仮説「入学前におけるプログラミング経験は実装能力との関係はない」を棄却する．

## 6. 考　　察

Miara ら[2] と Oman ら[3] の文献ではインデントはプログラムの読解力に対して有意な差があることが示されている．また，Weissman[5] と Love[6] ではプログラムの編集や再構築において，インデントとプログラムの実装能力について有意な差は確認されなかったと報告している．そして，本研究では白紙からプログラムを構築する実装能力とインデントの関係を調査した．各分析の結果は有意な差が確認された項目において，ファイ係数により，低い相関である項目が存在する．したがって，インデントを適切にコーディングできると実装能力が高いという関係，および受講前にプログラミング経験があると実装能力が高いという関係が成り立つ．

次に，インデントの習得時期に着目し，各表の結果の比較を行う．また，受講前のプログラミング経験の関係についても考察する．

表 5 は CA での定期試験において，インデントとユニットテストの関係を分析した結果である．また，同様に表 6 は CB での定期試験においてインデントとユニットテストの関係を分析した結果である．2 つの表を比較すると，CA の結果では低い相関を示す有意な差のある項目が確認される．一方，CB では有意な差はあるがほとんど相関がない項目しかみられない．このことから，CB より CA の方がインデントによる影響が大きいと読み取れる．

これは，CA が CB の前に受講する基礎科目であり，インデントを適切にコーディングできることがプログラムの構造の理解と関係しているため，CA が CB よりユニットテストに対してインデントが大きな影響を与えていると考えられる．

さらに，CB が CA に比べ大きな影響がみられないのは，CB ではオブジェクト指向によるモジュール性を意識した設計であるためと考えられる．これは，構造化プログラミングで学習した内容に加え，オブジェクト指向における状態の概念やクラス構造の概念が加わるため，相対的にインデントの役割が小さくなるためと推察される．

また，インデントの役割が減少する要因として，オブジェクト指向の設計により各メソッドの NCSS の値は短くなり，インデントによる入れ子の構造も浅いものとなることが考えられる．

CA・CB でのインデントの正否と CB のユニットテストとの関係を分析した結果が表 9・表 6 である．これらを比較すると表 9 のみに，低い相関を示す有意な差のある項目が確認される．

これらの表における分析の違いは，CB でのユニットテストに対するインデントの正否による時期である．表 9 は CA でのインデントの正否で集団を分割したものであり，表 6 は CB でのインデントの正否で分割したものである．

このことから，CB のユニットテストに対して CB より CA でインデントを習得することが大きく影響するといえる．つまり，プログラミング入門教育の初期である CA でのインデントは CA のユニットテストだけでなく，CB のユニットテストにも関係があるといえる．

この結果は初期の段階（CA）でインデントが適切にコーディングできた学生は，変数のスコープなどによるプログラムの構造に基づいた概念を理解している可能性が高く，その理解の差が CB で反映された結果であると考えられる．

受講前のプログラミング経験の有無に対して CA と CB におけるユニットテストの分析

した結果が表 7 と表 8 である．まず，これらの表から低い相関を示す有意な差のある項目が確認されたことにより，受講前のプログラミング経験の有無は実装能力に関係があるといえる．そして，低い相関を示す有意な差のある項目の数が 1 個ずつであることから，プログラミング経験の有無による実装能力への影響は限定的であると考えられる．

CA・CB のユニットテストにおいて，プログラミング経験で低い相関を示す有意な差が確認された項目はそれぞれ 1 個（表 7）と 1 個（表 8）であるのに対し，CA のインデントの正否では 3 個（表 5）と 2 個（表 9）である．このことから，入門教育の初期におけるインデントの習得は，受講前のプログラミングの経験に比べ，実装能力に影響があると推察する．

さらに CB では低い相関を示す有意な差がプログラミングの経験での 1 個（表 8）に比べ，CA のインデントの正否では 3 個（表 5）と項目が多い．このことから，CA のインデントの正否の方が後続の科目である CB に対しても影響が継続すると考える．

CA・CB におけるインデントの正否と受講前のプログラミング経験の有無についての関係を考察する．これらを分析したクロス集計表（表 10・表 11）では，有意水準 5%で有意な差が確認されなかった．このことからインデントの習得にはプログラミングの経験の影響を受けることなく習得可能であることが示唆される．

## 7. ま と め

本稿ではプログラミング入門教育におけるインデントと実装能力の関係および受講前のプログラミング経験と実装能力の関係に対して，2 つの帰無仮説を設定し，定期試験における結果を用いて，仮説の検証を行った．実装能力は試験におけるユニットテストの結果を使用して評価をした．

検証の結果，「インデントを適切に使用してコーディングできることと実装能力との関係はない」および「入学前におけるプログラミング経験は実装能力との関係はない」のそれぞれに対する，低い相関を示す有意な差のあることが確認され，2 つの帰無仮説は棄却された．

インデントと実装能力の関係を分析した結果，いくつかの項目で低い相関がみられたことから，インデントを正しく行える能力とプログラムを正しく書ける能力には何らかの関係があるのではないか，と考えられる．

また，このことから，インデントはプログラミング入門教育において十分ではないが必要な教育項目であると思われる．

そして，インデントを適切にコーディングできる能力を習得することは，受講前のプログラミング経験との関係がみられなかった．このことから，プログラミング経験の有無がインデントの習得を妨げるものではないことが示唆された．

今後は，プログラミング入門教育において，プログラムの実装能力に対して影響のある要素の研究を進める．

**謝辞**　この研究を進めるにあたり，ご指導をいただいた東京電機大学情報環境学部田窪昭夫教授に深く感謝いたします．

## 参 考 文 献

1) Shneiderman, B.: *Software psychology: Human factors in computer and information systems* (*Winthrop computer systems series*), Winthrop Publishers (1980).
2) Miara, R.J., Musselman, J.A., Navarro, J.A. and Shneiderman, B.: Program indentation and comprehensibility, *Comm. ACM*, Vol.26, No.11, pp.861–867 (online), DOI:http://doi.acm.org/10.1145/182.358437 (1983).
3) Oman, P.W. and Cook, C.R.: A paradigm for programming style research, *SIGPLAN Not.*, Vol.23, No.12, pp.69–78 (online), DOI:http://doi.acm.org/10.1145/57669.57675 (1988).
4) Sheil, B.A.: The Psychological Study of Programming, *ACM Comput. Surv.*, Vol.13, pp.101–120 (online), DOI:http://doi.acm.org/10.1145/356835.356840 (1981).
5) Weissman, L.M.: A methodology for studying the psychological complexity of computer programs., Ph.D. Thesis (1974).
6) Love, L.T.: Relating individual differences in computer programming performance to human information processing abilities, Ph.D. Thesis (1977).
7) McCabe, T.: A Complexity Measure, *IEEE Trans. Softw. Eng.*, Vol.SE-2, No.4, pp.308–320 (online), DOI:10.1109/TSE.1976.233837 (1976).
8) Meyer, B.: *Object-oriented software construction, 2nd ed.*, Prentice-Hall, Inc. (1997).

## 付　　録

表 5 から表 9 で使用したクロス集計表を**表 12**，**表 13**，**表 14**，**表 15**，**表 16** に示す．各表にはファイ係数と p 値を記す．また，分析に用いたプログラムの解答例を**図 10**，**図 11**，**図 12**，**図 13**，**図 14**，**図 15**，**図 16**，**図 17** に示す．

**表 12**　CA インデントと CA ユニットテストの関係
Table 12　Relation between indentation (CA) and unit test (CA).

(a) Cube main

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 77 | 34 | 111 |
| | False | 33 | 16 | 49 |
| 計 | | 110 | 50 | 160 |

$\phi = 0.0201$, p = 0.854

(b) Score main

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 96 | 37 | 133 |
| | False | 14 | 13 | 27 |
| 計 | | 110 | 50 | 160 |

$\phi = 0.164$, p = 0.0434

(c) Score getSum

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 92 | 34 | 126 |
| | False | 18 | 16 | 34 |
| 計 | | 110 | 50 | 160 |

$\phi = 0.177$, p = 0.0361

(d) SpinCoin getFace

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 86 | 33 | 119 |
| | False | 24 | 17 | 41 |
| 計 | | 110 | 50 | 160 |

$\phi = 0.129$, p = 0.120

(e) SpinCoin printMessage

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 100 | 40 | 140 |
| | False | 10 | 10 | 20 |
| 計 | | 110 | 50 | 160 |

$\phi = 0.153$, p = 0.0705

(f) SpinCoin getNumber

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 85 | 35 | 120 |
| | False | 25 | 15 | 40 |
| 計 | | 110 | 50 | 160 |

$\phi = 0.0778$, p = 0.332

(g) SpinCoin printResult

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 89 | 28 | 117 |
| | False | 21 | 22 | 43 |
| 計 | | 110 | 50 | 160 |

$\phi = 0.260$, p = 0.00183

(h) Shape drawLine

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 61 | 16 | 77 |
| | False | 49 | 34 | 83 |
| 計 | | 110 | 50 | 160 |

$\phi = 0.218$, p = 0.00658

(i) Shape drawTriangle

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 54 | 13 | 67 |
| | False | 56 | 37 | 93 |
| 計 | | 110 | 50 | 160 |

$\phi = 0.217$, p = 0.00913

**表 13**　CB インデントと CB ユニットの関係
Table 13　Relation between indentation (CB) and unit test (CB).

(a) LooseLeaf getItem

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 155 | 25 | 180 |
| | False | 4 | 3 | 7 |
| 計 | | 159 | 28 | 187 |

$\phi = 0.154$, p = 0.0697

(b) LooseLeaf getColor

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 155 | 27 | 182 |
| | False | 4 | 1 | 5 |
| 計 | | 159 | 28 | 187 |

$\phi = 0.0234$, p = 0.560

(c) LooseLeaf getNumber

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 158 | 27 | 185 |
| | False | 1 | 1 | 2 |
| 計 | | 159 | 28 | 187 |

$\phi = 0.102$, p = 0.278

(d) FlatFile bind

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 43 | 5 | 48 |
| | False | 116 | 23 | 139 |
| 計 | | 159 | 28 | 187 |

$\phi = 0.0750$, p = 0.357

(e) FlatFile remove

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 92 | 18 | 110 |
| | False | 67 | 10 | 77 |
| 計 | | 159 | 28 | 187 |

$\phi = -0.0466$, p = 0.678

(f) CdPocket getItem

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 137 | 21 | 158 |
| | False | 22 | 7 | 29 |
| 計 | | 159 | 28 | 187 |

$\phi = 0.110$, p = 0.156

(g) CdPocket getColor

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 142 | 23 | 165 |
| | False | 17 | 5 | 22 |
| 計 | | 159 | 28 | 187 |

$\phi = 0.0794$, p = 0.336

(h) CdPocket getNumber

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 139 | 22 | 161 |
| | False | 20 | 6 | 26 |
| 計 | | 159 | 28 | 187 |

$\phi = 0.0913$, p = 0.236

(i) FlatFile printList

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 90 | 9 | 99 |
| | False | 69 | 19 | 88 |
| 計 | | 159 | 28 | 187 |

$\phi = 0.175$, p = 0.0231

**表 14**　プログラミング経験と CA ユニットの関係
Table 14　Relation between programming experience and unit test (CA).

(a) Cube main

| | | プログラミング経験 | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 28 | 106 | 134 |
| | False | 9 | 59 | 68 |
| 計 | | 37 | 165 | 202 |

$\phi = 0.0936$, p = 0.248

(b) Score main

| | | プログラミング経験 | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 31 | 114 | 145 |
| | False | 6 | 51 | 57 |
| 計 | | 37 | 165 | 202 |

$\phi = 0.126$, p = 0.105

(c) Score getSum

| | | プログラミング経験 | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 30 | 104 | 134 |
| | False | 7 | 61 | 68 |
| 計 | | 37 | 165 | 202 |

$\phi = 0.148$, p = 0.0363

(d) SpinCoin getFace

| | | プログラミング経験 | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 27 | 106 | 133 |
| | False | 10 | 59 | 69 |
| 計 | | 37 | 165 | 202 |

$\phi = 0.0712$, p = 0.344

(e) SpinCoin printMessage

| | | プログラミング経験 | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 29 | 126 | 155 |
| | False | 8 | 39 | 47 |
| 計 | | 37 | 165 | 202 |

$\phi = 0.0184$, p = 1.00

(f) SpinCoin getNumber

| | | プログラミング経験 | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 26 | 109 | 135 |
| | False | 11 | 56 | 67 |
| 計 | | 37 | 165 | 202 |

$\phi = 0.0346$, p = 0.702

(g) SpinCoin printResult

| | | プログラミング経験 | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 29 | 102 | 131 |
| | False | 8 | 63 | 71 |
| 計 | | 37 | 165 | 202 |

$\phi = 0.134$, p = 0.0597

(h) Shape drawLine

| | | プログラミング経験 | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 21 | 59 | 80 |
| | False | 16 | 106 | 122 |
| 計 | | 37 | 165 | 202 |

$\phi = 0.166$, p = 0.0250

(i) Shape drawTriangle

| | | プログラミング経験 | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 22 | 44 | 66 |
| | False | 15 | 121 | 136 |
| 計 | | 37 | 165 | 202 |

$\phi = 0.270$, p = 0.000348

**表 15**　プログラミング経験と CB ユニットの関係
Table 15　Relation between programming experience and unit test (CB).

(a) LooseLeaf getItem

| | | プログラミング経験 | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 37 | 157 | 194 |
| | False | 0 | 8 | 8 |
| 計 | | 37 | 165 | 202 |

$\phi = 0.0962$, p = 0.355

(b) LooseLeaf getColor

| | | プログラミング経験 | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 37 | 158 | 195 |
| | False | 0 | 7 | 7 |
| 計 | | 37 | 165 | 202 |

$\phi = 0.0897$, p = 0.354

(c) LooseLeaf getNumber

| | | プログラミング経験 | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 37 | 162 | 199 |
| | False | 0 | 3 | 3 |
| 計 | | 37 | 165 | 202 |

$\phi = 0.0581$, p = 1.00

(d) FlatFile bind

| | | プログラミング経験 | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 17 | 27 | 44 |
| | False | 20 | 138 | 158 |
| 計 | | 37 | 165 | 202 |

$\phi = 0.277$, p = 0.000261

(e) FlatFile remove

| | | プログラミング経験 | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 24 | 84 | 108 |
| | False | 13 | 81 | 94 |
| 計 | | 37 | 165 | 202 |

$\phi = 0.108$, p = 0.146

(f) CdPocket getItem

| | | プログラミング経験 | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 25 | 131 | 156 |
| | False | 12 | 34 | 46 |
| 計 | | 37 | 165 | 202 |

$\phi = -0.109$, p = 0.132

(g) CdPocket getColor

| | | プログラミング経験 | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 29 | 136 | 165 |
| | False | 8 | 29 | 37 |
| 計 | | 37 | 165 | 202 |

$\phi = -0.0405$, p = 0.638

(h) CdPocket getNumber

| | | プログラミング経験 | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 30 | 131 | 161 |
| | False | 7 | 34 | 41 |
| 計 | | 37 | 165 | 202 |

$\phi = 0.0162$, p = 1.00

(i) FlatFile printList

| | | プログラミング経験 | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 23 | 73 | 96 |
| | False | 14 | 92 | 106 |
| 計 | | 37 | 165 | 202 |

$\phi = 0.139$, p = 0.0677

**表 16**　CA インデントと CB ユニットの関係
Table 16　Relation between indentation (CA) and unit test (CB).

(a) LooseLeaf getItem

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 106 | 46 | 152 |
| | False | 4 | 4 | 8 |
| 計 | | 110 | 50 | 160 |

$\phi = 0.0928$, p = 0.258

(b) LooseLeaf getColor

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 108 | 46 | 154 |
| | False | 2 | 4 | 6 |
| 計 | | 110 | 50 | 160 |

$\phi = 0.151$, p = 0.0769

(c) LooseLeaf getNumber

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 109 | 48 | 157 |
| | False | 1 | 2 | 3 |
| 計 | | 110 | 50 | 160 |

$\phi = 0.106$, p = 0.230

(d) FlatFile bind

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 36 | 7 | 43 |
| | False | 74 | 43 | 117 |
| 計 | | 110 | 50 | 160 |

$\phi = 0.196$, p = 0.0131

(e) FlatFile remove

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 69 | 26 | 95 |
| | False | 41 | 24 | 65 |
| 計 | | 110 | 50 | 160 |

$\phi = 0.101$, p = 0.226

(f) CdPocket getItem

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 95 | 35 | 130 |
| | False | 15 | 15 | 30 |
| 計 | | 110 | 50 | 160 |

$\phi = 0.194$, p = 0.0173

(g) CdPocket getColor

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 100 | 36 | 136 |
| | False | 10 | 14 | 24 |
| 計 | | 110 | 50 | 160 |

$\phi = 0.245$, p = 0.00349

(h) CdPocket getNumber

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 100 | 36 | 136 |
| | False | 10 | 14 | 24 |
| 計 | | 110 | 50 | 160 |

$\phi = 0.245$, p = 0.00349

(i) FlatFile printList

| | | インデント | | 計 |
|---|---|---|---|---|
| | | True | False | |
| ユニットテスト | True | 64 | 21 | 85 |
| | False | 46 | 29 | 75 |
| 計 | | 110 | 50 | 160 |

$\phi = 0.150$, p = 0.0625

```
import jp.ac.dendai.sie.usd.util.KeyboardReader;

public class Cube{
    public static void main(String[] args){
        System.out.print("整数を入力してください：");
        int n = KeyboardReader.readInt();
        System.out.println("三乗は" + (n * n * n) + "です。");
    }
}
```

**図 10**　Cube.java（CA）
Fig. 10　Cube.java (CA).

```
public class Score{
    public static void main(String[] args){
        System.out.print("合計は " + Score.getSum() + " 点です。");
    }
    public static int getSum(){
        int[] scores = {60, 50, 30, 80, 100};
        int sum = 0;
        for(int i = 0; i < scores.length; i++){
            sum = sum + scores[i];
        }
        return sum;
    }
}
```

**図 11**　Score.java（CA）
Fig. 11　Score.java (CA).

```
import jp.ac.dendai.sie.usd.util.RandomNumber;
import jp.ac.dendai.sie.usd.util.KeyboardReader;

public class SpinCoin{
    public static int getFace(){
        int face = RandomNumber.nextInt(2);
        return face;
    }
    public static void printMessage(){
        System.out.print("表 (1)ですか，裏(0)ですか？");
    }
    public static int getNumber(){
        int player = KeyboardReader.readInt();
        return player;
    }
    public static void printResult(int computer, int player){
        if(computer == player){
            System.out.print("当たり");
        } else {
            System.out.print("外れ");
        }
    }
}
```

**図 12**　SpinCoin.java（CA）
Fig. 12　SpinCoin.java (CA).

```
public class Shape{
    public static void drawLine(int n){
        for(int i = 0; i < n; i++){
            System.out.print("*");
        }
        System.out.println();
    }
    public static void drawTriangle(int n){
        for(int i = 0; i < n; i++){
            Shape.drawLine(i + 1);
        }
    }
}
```

**図 13**　Shape.java（CA）
Fig. 13　Shape.java (CA).

```
public interface Refill{
    public String getItem();
    public String getColor();
    public int getNumber();
}
```

**図 14**　Rifill.java（CB）
Fig. 14　Rifill.java (CB).

```
public class LooseLeaf implements Refill{
    private String item = "LooseLeaf";
    private String color;
    private int number = 35;
    public LooseLeaf(String color){
        this.color = color;
    }
    public String getItem(){
        return this.item;
    }
    public String getColor(){
        return this.color;
    }
    public int getNumber(){
        return this.number;
    }
}
```

**図 15**　LooseLeaf.java（CB）
Fig. 15　LooseLeaf.java (CB).

```
import java.util.ArrayList;

public class FlatFile{
    private int maxPage = 3;
    private ArrayList<Refill> refills = new ArrayList<Refill>();
    public FlatFile(){
    }
    public void bind(Refill refill){
        if (this.refills.size() < this.maxPage){
            this.refills.add(refill);
        } else {
            System.out.println("Page Full");
        }
    }
    public Refill remove(int number){
        return this.refills.remove(number);
    }
    public void printList(){
        for(int i = 0; i < this.refills.size(); i++){
            Refill refill = this.refills.get(i);
            System.out.println(refill.getItem());
            System.out.println(refill.getColor());
            System.out.println(refill.getNumber());
            System.out.println();
        }
    }
}
```

**図 16**　FlatFile.java（CB）
Fig. 16　FlatFile.java (CB).

```
public class CdPocket implements Refill{
    private String item = "CD Pocket";
    private String color = "Clear";
    private int number;
    public CdPocket(int number){
        this.number = number;
    }
    public String getItem(){
        return this.item;
    }
    public String getColor(){
        return this.color;
    }
    public int getNumber(){
        return this.number;
    }
}
```

**図 17**　CdPocket.java（CB）
Fig. 17　CdPocket.java (CB).

(平成 22 年 12 月 29 日受付)
(平成 23 年 9 月 12 日採録)

**高野　辰之**（学生会員）

1983 年生．2007 年東京電機大学情報環境学部情報環境デザイン学科卒業．2009 年東京電機大学大学院情報環境学研究科情報環境デザイン学専攻修了．現在，東京電機大学大学院先端科学技術研究科情報通信メディア工学専攻博士課程（後期）．ソフトウェアの設計・開発，教育工学に興味を持つ．電子情報通信学会，日本教育工学会各会員．

**宮川　　治**（正会員）

1964 年生．2001 年東京電機大学大学院工学研究科情報通信工学専攻博士後期課程修了．博士（工学）．現在，東京電機大学情報環境学部情報環境学科准教授．ソフトウェアの設計・信頼性，教育支援システムに興味を持つ．電子情報通信学会，日本教育心理学会各会員．

**小濱　隆司**（正会員）

1966 年生．1997 年東京電機大学大学院工学研究科電子工学専攻博士後期課程修了．博士（工学）．現在，東京電機大学情報環境学部情報環境学科准教授．ディジタル信号処理，教育支援システムに興味を持つ．電子情報通信学会，日本教育工学会各会員．