Vol.2018-CE-146 No.3
2018/10/20

クラスとオブジェクトの扱いの意識付けに着目した Java 学習支援ツールの提案

木間塚達[†1]　野田夏子[†1]

概要：初心者がプログラミング言語を学ぶことは難しい．さまざまなアプローチで学習支援を行う研究がなされているが，オブジェクト指向プログラミングの概念を学ぶ支援をする研究は少ない．本研究では，学習者がプログラミング中にクラス構造の把握とオブジェクトの生成及びオブジェクトに対してのメソッド呼び出しに意識を向けるように誘導する Java 学習支援ツールの提案をする．

Proposal of Java learning support tool focusing learner's attention on defining class and using object

ITARU KIMATSUKA[†1]　NATSUKO NODA[†1]

1. はじめに

従来からさまざまなアプローチでプログラミング教育を支援するツールの提案や研究が行われている．Scratch[1]や Alice[2]といったブロックベースのビジュアルプログラミング言語を利用したツールもその一つである．これらのツールを利用して，プログラミング初心者にビジュアルプログラミングで視覚的な支援を行うことは効果的であると考えられている．

オブジェクト指向プログラミング教育の支援を行う研究も複数存在する．それらの多くが，プログラミング初心者が起こしやすい構文エラーに対する支援である．

筆者らは，ティーチングアシスタント（TA），講義担当者の立場で，初めてオブジェクト指向を学ぶ学生に対して Java の教育を行っている．この経験から，オブジェクト指向プログラミングを学ぶ学生の多くが躓く問題は構文ではなく，オブジェクト指向プログラミング特有の概念であるクラスやオブジェクトの概念に関する内容だと考える．

そのため，Alice などのオブジェクト指向プログラミング言語をベースとしたビジュアルプログラミング言語での支援は，オブジェクト指向の概念の理解促進に対するサポートが少ないため，初心者には効果が発揮されない場合があると考えられる．

C 言語等の手続き型の言語でプログラミングの基礎的な学習を終えた学生がオブジェクト指向プログラミングの学習で躓く理由として，1.）クラスに対応するものが手続き型の言語には存在しない， 2.）クラスを作成し，オブジェクトを生成するというイメージができていない， 3.）メソッドと手続き型の関数を同じものだと認識している，などが考えられる．

そこで本研究では，オブジェクト指向プログラミング言語の Java を対象とし，手続き型の言語と異なる部分のコーディングを集中的に支援する Java 学習支援ツールを提案する．

2 章では，プログラミング教育支援ツールや支援ツールの適用実験等の先行研究について述べる．3 章では，筆者らの講義と TA の経験から得られたオブジェクト指向プログラミング言語の初心者が Java を学習する際の問題点について述べる．4 章では，提案する Java 学習支援ツールの内容について述べる．5 章で本稿を締め括る．

2. 関連研究

本章では，プログラミング教育支援ツールとそれらのツールに関わる研究について述べる．

Scratch や Alice はブロックを用いてプログラミングを行うため，利用者は構文を知らなくてもツールのガイドの下で，構文エラーを起こすことなくアルゴリズムの実装を行える．そのためプログラミング初心者の学習に利用されることが多い．

Greenfoot[3]はオブジェクト指向教育用に作成されたプログラミング開発環境である．Scratch や Alice とは異なり，基本的にはテキストベースでプログラミングする必要があるが，クラス階層の可視化等によって概念の理解をサポートする．そのため Utting らはオブジェクト指向プログラミング言語を学ぶには Scratch→Greenfoot, Alice→Greenfoot または Scratch→Alice→Greenfoot の順番で学習を進めること

†1 芝浦工業大学
Shibaura Institute of Technology

が望ましいと述べている[4].

Bau らはビジュアルプログラミング言語を用いた教育方法に関する研究を調査した結果とその考察を[5]で述べている．それによると，ブロックベースのビジュアルプログラミングのツールの目的は，プログラミングを簡単に学習させることであると述べている．また，ビジュアルプログラミングの利点と欠点について述べている．

利点として，次の３つが挙げられている．

(1) ブロックを選んでプログラムを作成する方が，思い出しながらコーディングするより楽である
(2) 文字列を経験豊富なプログラマーが認識するように表示してくれる
(3) ブロックなどで支援しているため，構文エラーなどの間違いが起きにくい

また欠点としては以下のような点が挙げられている．

(1) シンプルな変更に時間がかかる
(2) 検索に時間がかかる

また，Java や C#を学ぶ学生に対しての調査では，Scratch や Alice を先に学ぶ学生の方がはじめからテキスト言語を学習する学生よりも速く学習しているという研究結果[6][7]などがあることも紹介している．しかし，ブロックベースの学習からテキストベースの学習への移行の際に潜在的な問題がある可能性が存在するという研究結果[8][9]があることも紹介している．

これらのことから Bau らは，ビジュアルプログラミング言語は初心者の学習には効果的であると述べている．しかし，[8][9]の結果からテキストベースの学習への移行の際に問題があることや，プロフェッショナルなプログラマーにとってはかえってプログラミングの効率が悪くなることから，学習者のレベルが向上した場合は使いにくい部分もあると述べている．

3. 初学者が学習する際の問題点

筆者らは TA，講義担当者の立場で，オブジェクト指向プログラミングの初学者に対して Java の教育を行っている．本章では，対象となる講義の概要とその講義の経験からわかった初学者の問題点を述べる．

3.1 対象講義の概要

対象の講義は，芝浦工業大学デザイン工学部の「オブジェクト指向プログラミング」並びに「(同) 演習」である．受講者は，他の授業ですでに C 言語を学んだ学生である．それぞれ 1 コマずつで，講義と演習の各回を同じテーマで行い，履修者は講義で学んだ内容を演習で実践する．

次の表 1 は全 14 回の講義のテーマをまとめたものである．

演習では特定の開発環境を用いることはせず，通常のテキストエディタを用いてプログラムを作成し，コマンドラインインタフェースによりコンパイルと実行を行う．

表 1　対象講義の各回のテーマ

1	演習の説明と演習の準備
2	プログラミングの基本の復習（３基本構造）
3	クラスとインスタンス
4	フィールド・メソッド・コンストラクタ
5	スーパークラスとサブクラス
6	インターフェースと実装
7	パッケージ
8	可視性の修飾子
9	例外
10	例外の設計
11	ファイル操作の基礎・ファイルの入出力
12	ファイルの読み書き
13	スレッド
14	最終問題

3.2 講義での経験からわかった問題点

本節では，TA として演習中に学生を指導し，観察した経験からの知見を述べる．

学生の演習での解答例や質問から以下のようなことがわかった．

(1) Java のクラスの構造が理解できていない学生が多い
(2) オブジェクトやクラスといった概念を座学で学んでも理解できていない学生が多い
(3) 作成したオブジェクトに対してメソッド呼び出しを行うという意識が薄い学生が多い

(1)の問題は，プログラミングを初めて学ぶ学生が起こしやすい構文エラーと同様の問題である．そのためこの問題については，2 章で述べたようにプログラミングを視覚的に支援することが有効であると考えられる．

(2)について，クラスとオブジェクトが理解できていないため質問をする学生が全 14 回の講義の中盤以降もいた．特にオブジェクトの生成が必要な処理においても全くオブジェクトの生成をしていない学生が多かった．オブジェクト指向プログラムは一般にオブジェクトがなければ動かない，オブジェクトはプログラム中で生成しなければならない，といったオブジェクトの必要性に対する意識を持っていないことがこの問題の原因の一つであると考える．この問題に対しては，オブジェクトの生成をするという意識付けを行うことが有効であると考える．

(3)について，オブジェクトのメソッドを呼び出す記述が必要なところで，オブジェクトを指定せず，単にメソッド（メソッド名とパラメータ）だけを書いて，コンパイルが

成功しないと質問をする学生が多くいた．このことから，作成したオブジェクトに対してメソッド呼び出しを行うという意識が薄いことが問題であると考える．そのため，この問題については，学習者に意識付けを行うことのできる支援が有効であると考える．

また上記の３つの問題に総じて言えるのは，C 言語から Java への移行に伴う構文の違いと新たな概念を同時に学習することは難しいということである．

これらのことから，どの部分が手続き型の言語と同じでどの部分が異なるのかを明確にすることにより，上記で述べた問題点を解決できるのではないかと考える．

4. Java 学習支援ツールの提案

3 章で述べた Java 初心者が学習する際の問題点から，オブジェクト指向に関して，新たに学習する部分と今まで学習してきた部分との区別がつかないというのが，オブジェクト指向プログラミングの初心者が Java を学習する際の問題の原因の一つであると考える．つまり，パラダイムの違いが学習者の混乱を引き起こしている．

Java の初心者に対しても 2 章で述べた先行研究を踏まえると，ビジュアルプログラミングは一定の効果があると考える．しかし，既存のツールの多くは Scratch や Alice などのブロックベースの形式をとるものか，Greenfoot などのテキストベースで，クラス階層の可視化によってサポートをするものであり，パラダイムの違いによる混乱をサポートするツールはないと思われる．

これを踏まえて，本研究ではオブジェクト指向の概念を学ぶ際に利用する Java 学習支援ツールを提案する．

本章では，提案ツールの目的，対象，概要，支援項目について述べる．

4.1 目的

本研究の提案ツールの目的は，オブジェクト指向プログラミングを未学習の学生に対して，正しいクラスの構造の把握とオブジェクトの生成及びオブジェクトに対してのメソッド呼び出しについての Java コードが記述できるようにすることである．

3 章で述べたように，オブジェクト指向に関して，新たに学習する部分と今まで学習してきた部分との区別がつかないというのが，オブジェクト指向プログラミング初心者が Java を学習する際の問題の原因の一つであると考える．

この問題を解決するために本研究では，学習者がプログラミング中にクラス構造の把握とオブジェクトの生成及びオブジェクトに対してのメソッド呼び出しに意識を向けるように誘導する Java 学習支援ツールを提案する．

4.2 対象

本研究で提案するツールの利用対象者は，C 言語などの手続き型のプログラミングを学習済み，かつオブジェクト指向プログラミングを未学習の学生を対象とする．

また提案ツールは，自習時及び講義での利用を想定している．

4.3 提案ツールの概要

本研究では，Java 学習支援にブロックを利用する．本研究では，Java における一つのクラス，メソッド，コンストラクタをそれぞれ個別にまとめたものをブロックとする．またクラス内のフィールドをまとめたものとクラス内で生成された一つ一つのオブジェクトもブロックとして扱う．

提案ツールで支援する正しいクラスの構造の把握とオブジェクトの生成及びオブジェクトに対してのメソッド呼び出しについては，ブロックをドラッグ＆ドロップし，ポップアップで表示されたメソッドなどを選択することで，コードと同等のものを作成できる．そして，基本三構造などの手続き型の言語と共通する部分は通常のテキストベースでのコーディングを行うことで Java プログラムが作成できるようになっている．

提案ツールの画面構成は，図 1 のようになる．

学習者がプログラムを作成する際に操作する部分は，図 1 の中央の緑のブロックである．基本的に学習者は画面左側を操作する．右側のコード部分は，作成したプログラムのコードがテキストで表示されるようになっている．

画面左上にある「型の箱」には変数宣言の際に利用する型が表示される．基本データ型と参照型の違いがわかるように基本データ型は通常の黒文字で表示され，参照型は緑色のブロックで表示される．学習者が作成したクラスは，保存すると自動でコンパイルされる．もしコンパイルエラーならばエラーメッセージを表示する．コンパイルが正常に終了すれば，型の箱に作成したクラスのブロックが追加される．

また画面左下の「オブジェクトの箱」には作成中のクラス内で生成されたオブジェクトがブロックで表示されるようになっている．

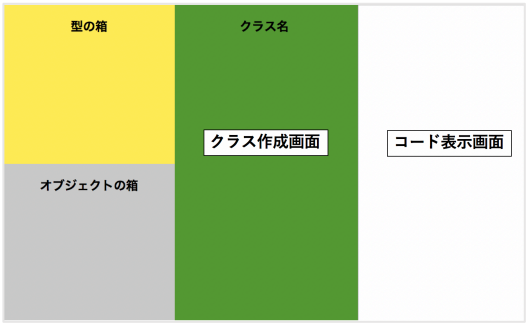

図 1　提案ツールの画面

4.4 支援内容

本提案ツールが支援するのは，正しいクラスの構造の把握とオブジェクトの生成，オブジェクトに対してのメソッド呼び出しの３つである．

4.4.1 正しいクラスの構造の把握

初心者の多くがクラスの構造を把握出来ていないため，このツールでは，クラスの内部にはフィールド，メソッド，コンストラクタがあるということを学習者が視覚的に把握できるように，また覚えられるようにする．

学習者がクラスの構造を把握しやすいようにフィールド，メソッド，コンストラクタを色分けしたブロックで区切ることにより，学習者がクラスの構造を視覚的に把握しやすいようにする．図 2 はクラスのブロックの構成である．

図 2　クラスのブロックの構成

- クラスのブロック

学習者がクラス作成画面で右クリックし，作成するクラスの名前を決めるとクラスのブロックができる．クラスの名前を決める際にメインクラスかそれ以外のクラスかを選択でき，メインクラスを選択するとメインメソッドのブロックが埋め込まれる．

- フィールドのブロック

図 3 はフィールドのブロックの例である．

図 3　フィールドのブロック

フィールドのブロックでは，変数宣言と変数への代入が行える．フィールドのブロックをクラスのブロック内部に埋め込むには，クラスのブロックを右クリックし，フィールドの作成を選択することで埋め込むことができる．

- メソッドのブロック

図 4 はメソッドのブロックの例である．

メソッドをクラスのブロック内部に埋め込むには，クラスのブロックを右クリックし，メソッドの作成を選択すると戻り値の型とメソッド名，引数の型と引数名の記述を促すメッセージが表示され，メッセージに従って入力をする必要がある．

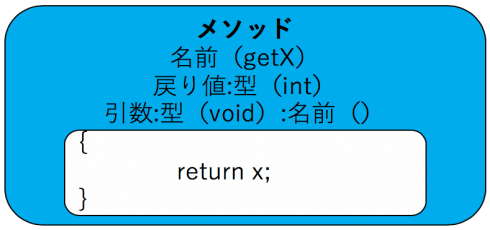

図 4　メソッドのブロックの例

メソッドの処理部分はメソッドのブロック内部にある白抜きの部分にコードで記述する．

- コンストラクタのブロック

図 5 はコンストラクタのブロックの例である．

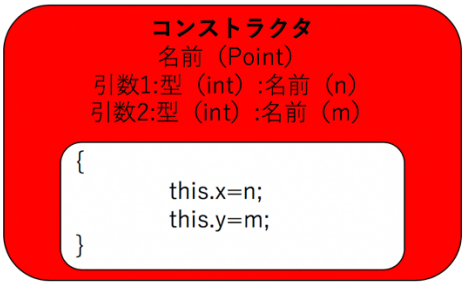

図 5　コンストラクタのブロックの例

コンストラクタをクラスのブロック内部に埋め込むには，クラスのブロックを右クリックし，コンストラクタの作成を選択する．そして，引数の数と型と名前を記述し，その引数と対応させるフィールドを選択することでクラスのブロックに埋め込むことができる．

4.4.2 オブジェクトの生成

Java では，メソッド本体やフィールド宣言において，オブジェクトの生成を行う．しかし，オブジェクト指向初心者は，オブジェクトを生成するという意識が低い．そこで提案ツールでは，フィールドのブロック内とメソッド(メインメソッドを含む)のブロック内部で，オブジェクトの生成を視覚的に支援する．

提案ツールではオブジェクトを生成するかしないかの返答をすることでオブジェクトの作成ができるようにする．

型の箱から宣言したい参照型をフィールドのブロックやメソッド（メインメソッドを含む）の処理部分を記述するブロック内部にドラッグ＆ドロップすることで変数宣言ができる．この方法で宣言した場合，変数名を入力できるよう空白のブロックが表示される．図 6 は参照型の変数宣言をした後に変数名を入力した際の画面である．矢印はドラッグ＆ドロップを表している．

そして図 7 のように変数名の右上に「ここでオブジェクトを生成しますか？」という質問が表示される．

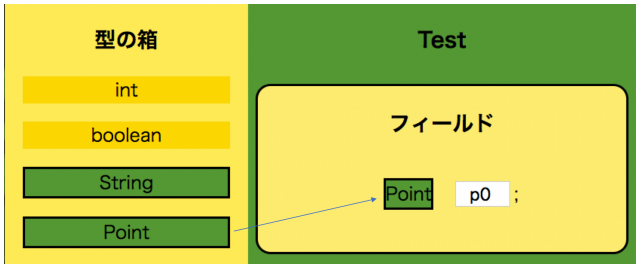

図 6　変数宣言をした後に変数名を入力した際の画面

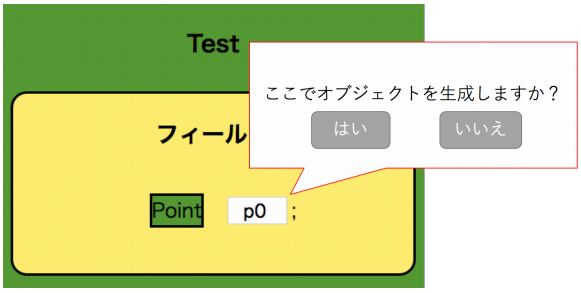

図 7　オブジェクトを生成する際の画面

学習者が「はい」を選択すると変数名のあとに自動で = と new が記述され，コンストラクタを選択するポップアップが表示される．そして学習者は利用したいコンストラクタを選択し，場合によっては必要な引数を入力することでオブジェクトが生成できる．また学習者が「ここでオブジェクトを生成しますか？」という質問に「いいえ」と答えると変数宣言のみが行えるようになっている．

オブジェクトの生成を完了すると画面左下にあるオブジェクトの箱に生成されたオブジェクトがブロックとして表示される．また作成中のクラスのフィールド内で宣言した参照型の変数もオブジェクトの箱として表示される．

図 8 はドラッグ＆ドロップを用いてオブジェクトを生成した後の画面である．

4.4.3 オブジェクトに対するメソッド呼び出し

手続き型の言語を学んだオブジェクト指向初学者は，オブジェクトに対してメソッド呼び出しを行うという意識が薄く，オブジェクトの指定がないメソッド呼び出しを記述するミスが目立つ．そのため提案ツールでは，どのオブジェクトに対してメソッド呼び出しを行うのかを直感的に理解できるように，学習者がメソッド呼び出しを行いたいオブジェクトをオブジェクトの箱からドラッグ＆ドロップすることでメソッド呼び出しを行えるようにする．

オブジェクトに対するメソッド呼び出しをする方法は，学習者がメソッド呼び出しを行いたい場所に，オブジェクトの箱からメソッド呼び出しを行いたいオブジェクトをドラッグ＆ドロップする．その後オブジェクトに対して呼び出し可能なメソッドを選択することで行える．

図 9 はオブジェクトに対してメソッド呼び出しを行う際のツールの操作画面である．

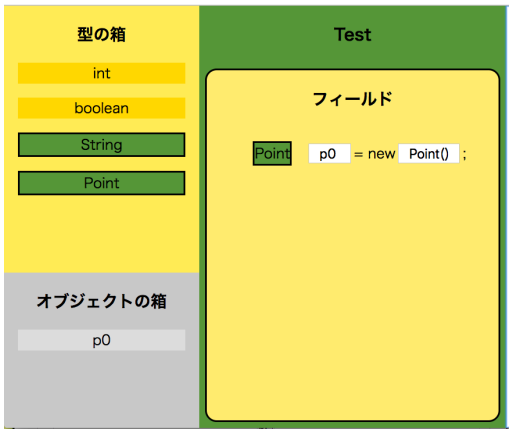

図 8　ドラッグ＆ドロップを用いてオブジェクトを生成した後の画面

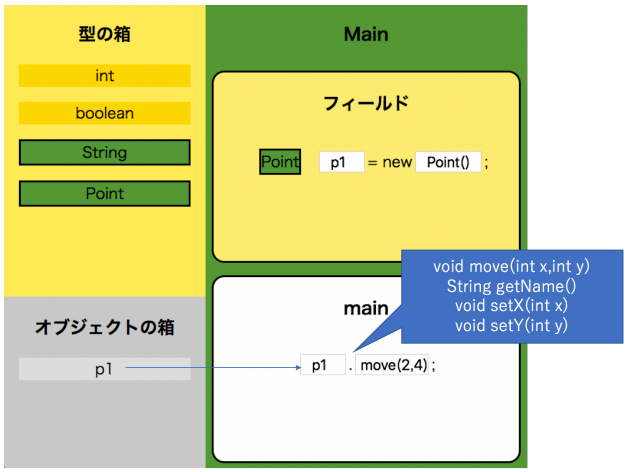

図 9　オブジェクトに対してメソッド呼び出しを行う際の操作画面

5. おわりに

本稿では，オブジェクト指向プログラミング初心者が利用する Java 学習支援ツールについて検討し，その仕様について述べた．

プログラミング教育に関する支援方法とその関連調査と TA として演習中に学生を指導し観察した経験から，オブジェクト指向プログラミング初心者が Java を学習する際の問題として，パラダイムの違いが学習者の混乱を引き起こしていると考えた．そこでオブジェクト指向に関して，新たに学習する部分にのみ意識を向けさせることが効果的であると考え，クラス構造の把握とオブジェクトの生成及びオブジェクトに対してのメソッド呼び出しを支援する Java 学習ツールを提案した．

今後はこのツールを完成させ，評価を行う．評価方法としては，基本的なクラスの作成とオブジェクトの生成，オブジェクトに対するメソッド呼び出しの範囲に関する Java 演習問題を提案ツールを利用して被験者に解いてもらう．その解答の正答率とツールを利用しなかった場合の解答の正答率との比較を予定している．この評価により，提案ツールの有効性を明らかにしたいと考えている．

参考文献

[1] “Scratch 教育者のためのスクラッチ”. https://scratch.mit.edu/educators/, (参照 2018-09-21).
[2] “About Alice”. https://www.alice.org/about/,(参照 2018-09-21).
[3] “Greenfoot Teach &Learn Java Programming”. https://www.greenfoot.org/door,(参照 2018-09-21).
[4] Ian Utting, Stephen Cooper, Michael Klling, John Maloney, Mitchel Resnick. Alice, Greenfoot, and Scratch -- a discussion. ACM Transactions on Computing Education. 2010, vol.10, no.4.
[5] David Bau, Jeff Gray, Caitlin Kelleher, Josh Sheldon, Franklyn Turbak. Learnable programming: blocks and beyond. Communications of the ACM. 2017, vol.60, no.6, p.72-80.
[6] Michal Armoni, Orni Meerbaum-Salant, Mordechai Ben-Ari. From scratch to “real” programming. ACM Transactions on Computing Education. 2015, vol.14, no.4.
[7] Barbara Moskal, Deborah Lurie, Stephen Cooper. Evaluating the effectiveness of a new instructional approach. ACM SIGCSE Bulletin. 2004, vol.36, no.1, p. 75-79.
[8] Kris Powers, Stacey Ecott, Leanne Hirshfield. Through the Looking Glass: Teaching CS0 with Alice. ACM SIGCSE Bulletin. 2007, vol.39, no.1, p.213-217.
[9] Ryan Garlick, Ebru Celikel Cankaya. Using Alice in CS1 – A Quantitative Experiment. Proceedings of the fifteenth annual conference on Innovation and technology in computer science education. 2010. p.165-168.