BlockEditor Hinoki：オブジェクト指向に対応した ビジュアル-Java 相互変換技術の開発

BlockEditor Hinoki：Development of Visual to Java through Mutual Language Transrlation Technology That Supports Object-Oriented

大畑　貴史[*1]，　酒井　三四郎[*2]，　松澤　芳昭[*3]
Takashi OHATA [*1]，　Sanshiro SAKAI[*1]，Yoshiaki MATSUZAWA[*1]
*1*2*3 静岡大学情報学部
*1*2*3Faculty of Informatics,Shizuoka University
Email:ohata@sakailab.info

あらまし：Visual Programming Language(VPL)とテキスト記述型言語の相互変換によるプログラミング入門教育支援システムが提案されている(松澤ら，2014)．初学者が構文エラーに悩まされることなく，かつテキスト記述型言語にシームレスに移行可能である．本研究では，当システムのオブジェクト指向構文への拡張を行った．BlockEditor Hinoki の特徴は，1)カプセル化，継承，ポリモーフィズムの表現能力を持った VPL の設計，2)学習者が設計したクラスオブジェクトを VPL で利用可能，3)より多くの文や式の Java と VPL の相互変換に対応，である．1 つのクラスを表現したキャンバスに，プログラム構成要素を表したブロックを 2 次元空間に設置してクラス設計をすることもできる．
キーワード：プログラミング教育，オブジェクト指向，ビジュアルプログラミング，Java

1.　はじめに

Visual Programming Language(VPL)とテキスト記述型言語の相互変換によるプログラミング入門教育支援システム(1)が松澤らにより提案されている．初学者が構文エラーに悩まされることなく，かつテキスト記述型言語にシームレスに移行可能である．当システムを用いたプログラミング入門教育は有用であることが支持されているが，オブジェクト指向プログラミング（OOP）初学において有用であるということは支持されていない．

本研究では，OOP の学習においても当システムが有効であるという仮説を立て，当システムのオブジェクト指向構文への対応を行い，これを検証する．

2.　先行研究

近年，Scratch や Alice などの VPL が開発されている．これらは文法エラーや構文エラーを回避でき，アルゴリズムの構築に注力した学習が可能である．しかしこれらのシステムは言語変換機能を持たないため，テキスト記述型言語の学習が不可能である．

Google Blocky は言語変換能力を持つ VPL であるが，VPL からテキスト記述型言語への 1 方向のみの変換である．更にこれらの言語は，Java のようなクラスベース方式のオブジェクト指向プログラミングの学習が不可能である．

3.　アーキテクチャ

BlockEditor Hinoki（Hinoki）の基盤となるシステムとして OpenBlocks(2)を利用している．このシステムでは，命令となる「ブロック」を組み合わせることでプログラムの実装が可能である．

Hinoki の外観を図 1 に，アーキテクチャを図 2 に示す．「ファクトリ」からブロックを取り出し，「キャンバス」に設置することでプログラムの実装が可能である．Java とブロックの相互変換も可能である．

図 1：BlockEditor Hinoki の外観

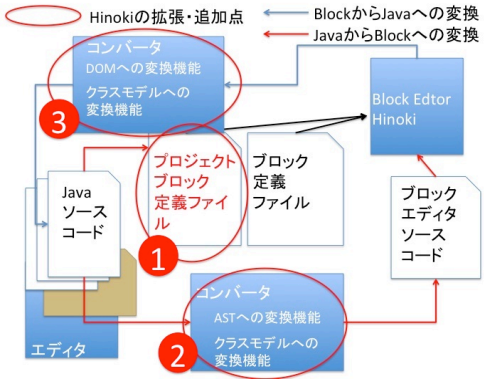

図 2：BlockEditor Hinoki のアーキテクチャ図

アーキテクチャの変更点は，1)．プロジェクトを動的に解析し，「プロジェクトブロック定義ファイル」を作成する，2)．式や文の抽象構文木(AST)への変換

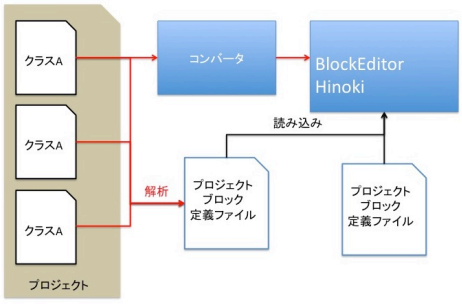

図 3：動的なブロック作成のアーキテクチャ図

の拡張，3)ドキュメントオブジェクトモデル(DOM)への変換のオブジェクト指向構文への拡張，である．

3.1　動的なブロックモデル作成

図 3 に動的にブロックモデルを作成するアーキテクチャの詳細を示す．基本的なブロックモデル（変数や，分岐ブロックなど）は，「ブロック定義ファイル」に記述されている．Hinok では同一プロジェクト内の Java ファイルに記述されたクラスの継承関係や，メソッドを動的に解析し，学習者の作成したクラスのブロックモデルを定義した「プロジェクトブロック定義ファイル」を作成している．

この 2 つのファイルを読み込むことで，基本的なブロックと学習者自身が設計したクラスオブジェクトのブロックを利用可能にしている．

3.2　オブジェクト指向構文への相互変換の対応

オブジェクト指向構文の Block から Java へ変換する際の AST への変換，Block から Java へ変換する際の DOM への変換に対応した．これにより，VPL で表現できるプログラム構成要素が増え，OOP に対応したプログラムの表現が可能である．

4.　特徴

4.1　キャンバス

キャンバス（図 1 の 1）は，ひとつのクラスに対応している．学習者はキャンバス上にブロックを設置することでクラスの設計が可能である．

キャンバスに設置されたブロック位置情報をソースコードに保持し再起動時には位置情報を読み取り同じ位置に復元されるようにもしている．

4.2　インスタンス変数

他のオブジェクトのインスタンス変数に対し直接操作するということはオブジェクト指向の考え方として好ましくない．

Hinoki では private 変数のみを扱い，そのアクセスには getter メソッドと setter メソッドによるメッセージングで行うよう設計にした．実際のインスタンス変数とその変換例が図 4 に示す．

4.3　継承メソッド呼び出し

Hinoki では図 5 のように，ポップアップメニューでメソッドをクリックすることで継承メソッドブロックの作成が可能である．図 5 では ClassA を継承した ClassB の利用可能なメソッドを表示している．

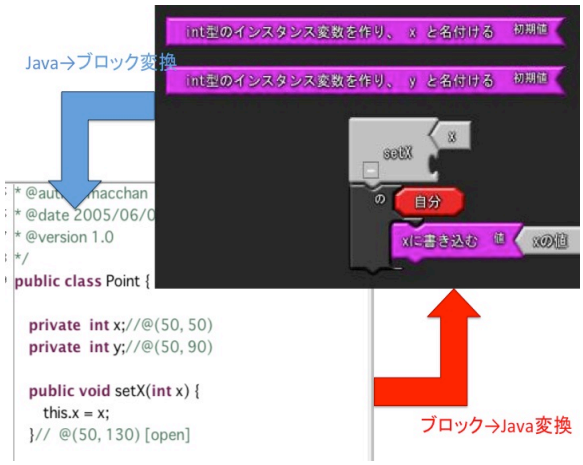

図 4：インスタンス変数と setter の変換例

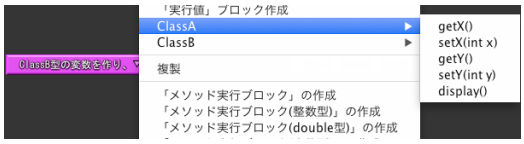

図 5 継承メソッドの参照例

4.4　this・super キーワード

this，super キーワードのブロックを追加した．図 4 の「自分」というブロックは this のことを指す．super は「親クラス」というブロックに変換される．

ドット演算子は「が」や「の」というラベルの貼られた黒いブロックに変換され，「自分の x に書き込む」という日本語で表現している．メソッドのドット演算子も同様に「が」や「の」という黒いブロックに変換される．

4.5　配列・リスト

配列やリストのブロックを新たに追加した．リストは右クリックにより利用可能なメソッドの作成ができるようになっている．配列は右クリックにより要素の代入を行うブロックと，要素の参照を行うブロックの作成が可能になっている．

5.　評価・考察

Hinoki は OOP 初学講義で利用されている 48 個の構文要素のうち 45 個に対応している．対応していない構文要素は他の要素で代用可能であることから，OOP 初学講義で利用可能な相互変換能力があると評価できる．

将来的に講義で利用する．Hinoki を用いた学習者は，カプセル化したクラス設計，アルゴリズム構築に集中した学習が可能になることが考えられる．

参考文献

(1) 松澤芳昭，保井元，杉浦学，酒井三四郎：“ビジュアル-Java 相互変換によるシームレスな言語移行を指向したプログラミング学習環境の提案と評価 “，情報処理学会論文誌, No.55, No.1, pp.57-71, 2014.

(2) Ricarose Vallarta Roque. Openblocks: An extendable framework for graphical blockprogramming systems. Master thesis at MIT, 2007.