N-009

オブジェクト指向プログラミング技法のイメージ理解を目指した学習ソフトウェアの開発

Development of Educational Software for Visual Understanding on Object-Oriented Programming Techniques

稲垣　宏†　　岩橋和哉†
Hiroshi Inagaki　　Kazuya Iwahashi

1．はじめに

高専の情報工学科では，中学卒業後の学生を受け入れ，5 年間に亘る実践的なプログラミング教育を実施している．本校においても，社会的要求に適合するコンピュータエンジニアの育成を目指し，長年に亘って，効果的なプログラミング教育のカリキュラムを検討してきた．

現在のカリキュラムでは，プログラミング初学者を対象に，C 言語などの手続き型言語を利用したプログラミング基礎教育を行なった後，「アルゴリズムとデータ構造」に関する様々な知識や技法を習得している．そして，それらの知識が一通り身についたことを確認した後，Java 言語を利用したオブジェクト指向プログラミングのスキルの習得を始めている．

このときには，すでに基本的なプログラミングスキルを有しているので，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「インスタンス」の説明から入ることになる．しかし，現状では，この移行がスムーズにいかない．C 言語を使えば作成できる課題であっても，Java 言語では作成できないという学生が多い．「例題を真似してプログラムを記述することはできるが，深く理解しているわけではない」とか「オブジェクト指向型言語のどこが優れていて，どんなメリットがあるのかピンとこない」といった意見がよく聞かれる．

そこで，どこに問題があるのかを探るため，プログラミング演習や補講などを通して，多くのヒアリング調査を行なってきた．そこでわかってきたことは，「オブジェクト指向プログラミング」を，手続き型言語の延長線上にあるものとして捉えようとするために混乱している学生が多いことである．

オブジェクト指向プログラミングで登場する新しい概念を，手続き型言語の枠の中で捉えようとするために，明確なイメージがわかないのである．したがって，オブジェクト指向プログラミングへの移行をスムーズに行なうためには，個々の文法事項の説明よりも，新しく取り入れられる概念を明確にイメージできることが大切である．そして，オブジェクト指向プログラミングは，人間のモノづくりの感覚に近く，生産性を高めるしくみが備わっているということを実感してもらう必要がある．

そこで本研究では，オブジェクト指向プログラミングの概念を直感的にイメージで理解してもらうことを目指した学習ソフトウェアの開発を行なっている．このようにオブジェクト指向プログラミングを直観的なイメージで理解させる試みは文献[1]の書籍内にも見られる．

† 豊田工業高等専門学校, Toyota College of Technology

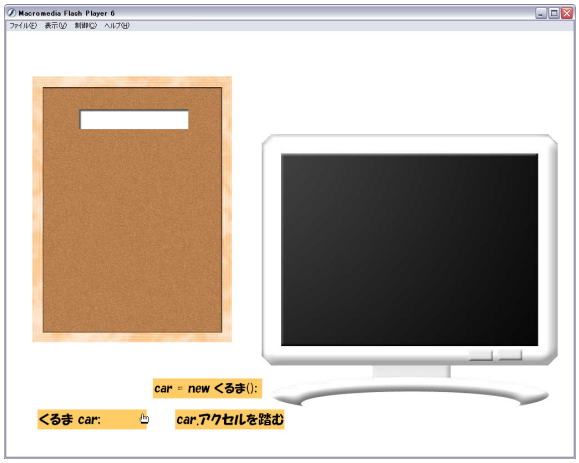

図１．画面レイアウト

2．開発した学習ソフトウェアの主な仕様

2.1　画面レイアウト

開発した学習ソフトウェアの画面レイアウトを図１に示す．左側にあるコルクボードの画像で表されている領域が「プログラム組み立てエリア」であり，右側にあるディスプレイの画像で表されている領域が「イメージ提示エリア」である．これら二つの領域の下側に，「コード断片エリア」がある．この領域には，ソースコードの断片（プログラムの１ステップに相当）が散らばっている．

2.2　基本的な機能

本システムの基本的な操作は次のとおりである．学習者は，「コード断片エリア」に散らばっているコード断片の中から一つを選び，それをドラッグ＆ドロップ操作によって「プログラム組み立てエリア」に順にはめ込んでいく．そして，この操作を繰り返すことで，正しいソースコードを完成させていくのである．

ここで特徴的なことは，一つのコード断片がはめ込まれると同時に，そのコードによってなされるプログラムの動作イメージが「イメージ提示エリア」に現れることである（図２）．この機能によって，オブジェクト指向プログラミングにおけるソースコードとプログラムの動作イメージが直感的に結び付けられ，基本概念のイメージ理解に大いに役立つ．

さらに，バラバラにしてあるソースコードの断片を正しい順序に組み上げていくためには，それぞれのコード断片の内容をきちんと理解していることが要求されるため，理

解度を確認する方法として効果的である（文献[1]の練習問題にも採用されている）．

なお，使用しているソースコード及び画像ファイルの情報は，XML 形式の設定ファイルにより別に管理している．そのため，データの変更を簡単に行なうことができ，汎用性や拡張性に優れている．使用するソースコードと画像ファイルを変更した例を図３に示す．

3. 実行例

3.1 オブジェクト参照変数

「オブジェクト参照変数の宣言」に該当するコード

```
くるま car;
```

に対応して，「イメージ提示エリア」内に，リモコン付きの箱のイメージが現れる．これは，オブジェクト参照変数によって，それが参照しているインスタンスの操作が行なわれることを，直感的に把握してもらうための工夫である（文献[1]でも同様のイメージが使われている）．

3.2 インスタンス化

インスタンスの生成に該当するコード

```
car = new くるま();
```

に対応して，設計図から自動車が生成されるアニメーションが実行される（図４）．これは，「クラスは設計図であり，実体ではないこと」，「インスタンス化によって初めて実体が現れること」をイメージにより直感的に理解してもらうための工夫である．

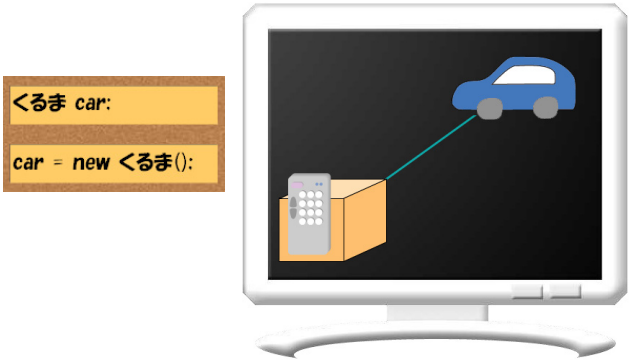

図２．コードに対応するイメージの出現

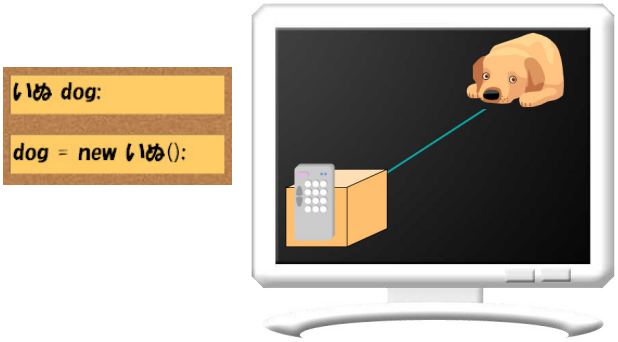

図３．ソースコードデータとイメージデータの変更

図４．インスタンスの生成

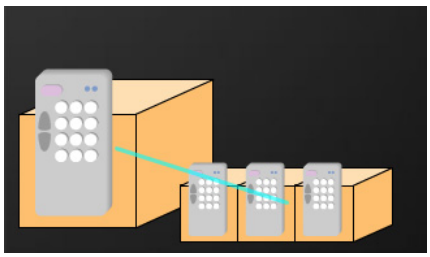

図５．オブジェクト配列の生成

3.3 オブジェクト配列

オブジェクト指向プログラミングにおいては，オブジェクトとして配列を扱うため，C 言語における配列と同じイメージをもっていると混乱する．特に，オブジェクト配列を理解するには，オブジェクト指向プログラミングにおける配列の正確なイメージをもっていることが不可欠である．オブジェクト配列の生成に該当するコード

```
くるま[]　car;
car = new くるま[3];
```

と対応して，図５に示すようなオブジェクト配列のイメージが現れる．これを見ると，この時点ではまだ実体としての自動車が生成されていないことが直観的にわかる．

4. おわりに

本稿では，オブジェクト指向プログラミングの概念を直感的に理解するための学習ソフトウェアについて，現在までの取り組みを紹介した．そこでは，「クラス」，「インスタンス」，「配列」といったオブジェクト指向プログラミングにおける最も基本的な概念を，ソースコードと対応して現れるイメージによって，効果的に学習することができる．

また，使用するソースコードや画像ファイルの情報を外部の設定ファイルにより管理することで，汎用性や拡張性を高めている．

今後は，オブジェクト指向プログラミングにおけるその他の重要な概念である「カプセル化」，「継承」，「ポリモーフィズム」に関しても，イメージで直感的に捉えることができるしかけを考えていきたい．

参考文献

[1] Kathy Sierra, Bert Bates: Head First Java, O'Reilly, 2003.（キャシー シエラ，バート ベイツ（夏目 大訳）：Head First Java ― 頭とからだで覚える Java の基本，オライリージャパン，2004）