# オブジェクト指向言語「ドリトル」を利用した情報教育について*

兼宗 進*, 中谷 多哉子†, 御手洗 理英‡, 福井 眞吾*, 久野 靖*
筑波大学大学院 経営・政策科学研究科* (有)エス・ラグーン† (株)アーマット‡
〒112-0012 文京区大塚3-29-1
{kanemune,fukui,kuno}@gssm.otsuka.tsukuba.ac.jp*
tina@slagoon.to† rie@armat.com‡

## 概要

プログラミング言語「ドリトル」を利用した情報教育について提案する。ドリトルはオブジェクト指向言語であり、日本語を用いた簡単な記述により、小さい労力で現代的なソフトウェアを体験できる。我々はプログラミングを通じて計算機に能動的に働きかけることにより、学校教育の中で生徒が計算機やネットワークの動作原理を体感できることを目指している。本発表ではドリトルの現状を報告するとともに、オブジェクトを教室内で共有する形の新しいプログラミング環境について提案する。

## 1 はじめに

新教育課程において、初中等教育における情報教育の準備が進められている。また、「情報社会」の時代を迎え、計算機の原理を理解し、その特徴や限界を知ることがますます重要になっている。

計算機の動作原理を学ぶ題材として、たとえば論理回路、マイクロチップの動作、基本的なアルゴリズムなどが考えられる。しかし、これらと日常使われる各種の応用ソフトウェアとの間には大きな隔たりがあり、これらを学んだからといって、計算機の仕組みを想像できるものではない。

本稿では前記の目標のためのよりよい学習手段として、オブジェクト部品を活用したプログラムを行うことで手軽に応用ソフトウェアの構成を体験し、プログラミングを通じて計算機そのものの動作原理にも触れられる学習環境を提案する [6] [7]。

*K12 Education using "Dolittle" Object-oriented Programming Language, Susumu Kanemune*, Takako Nakatani†, Rie Mitarai‡, Shingo Fukui*, Yasushi Kuno* (Tsukuba University*, S.Lagoon Co.,Ltd.†, Armat Corporation‡)

## 2 教育におけるプログラミング言語

初中等教育で利用することを想定したときに、プログラミング言語に望まれる性質を挙げる。

(1) わかりやすく、習得が容易であること。限られた時間の中でプログラミングを体験できる必要がある。よって、言語自体の習得に訓練が必要なCやJavaといったシステム開発用の言語は適していない。
(2) 楽しんで使えること。プログラムが完成する過程で小さな成功体験を積み重ねていける環境が望ましい。
(3) 計算機の動作原理を体感できること。プログラミングを体験することにより、日常接するアプリケーションプログラムの動作について、その動作を類推できることが望ましい。
(4) 現代的なプログラミングの概念を学べること。既成のオブジェクト部品を活用することで、短時間に高度な機能を持つソフトウェアを作ることができる。従来、入門用として利用されてきたBASICやLOGOは、設計が古

いこともあり、このような概念を学ぶことには適していない。

筆者らは、既存の言語でこれらの性質をすべて満たすものはないと考え、新たに教育用のプログラミング言語「ドリトル」を開発した。次節でその設計について述べる。

## 3 ドリトルの設計

### 3.1 設計方針

我々が採用した設計方針を以下に示す。

1. オブジェクト指向言語であること [3]
   オブジェクト部品を活用することで、普段接しているソフトウェアと同等の原理で動作するプログラムを短時間で作成できる。
2. プロトタイプ方式の採用 [1] [5]
   クラス方式におけるクラス、インスタンス、継承などの概念を扱わないことで、オブジェクト指向の理解を容易にする。
3. テキスト表現に基づくソースコード
   テキストで表現されたソースコードを打ち込んで動かすことで「計算機の動作原理を体感する」ことを可能にする。
4. 簡潔であること
   プログラムを1行実行するごとに意味のある動作を観察することで、試行錯誤しながらプログラミングを学べる言語にする。
5. 日本語との対応性
   英語を習っていない生徒を考慮し、職別子、記号、語順などを日本語に近づけた言語を検討する。また、予約語の概念をなくすことで、理解を容易にする。ただし、曖昧さのない人工言語であることを学ぶために、特に自然言語に近づけることはしない。

従来の言語の例として、図1にJavaのプログラムを示す。これは最も簡単なサンプルであり、画面に「こんにちは」というメッセージを表示するだけであるが、初中等教育の入門用言語として使うには、次のような問題がある。

- 記号が多い。「( )」「{ }」「;」「" "」「[ ]」などを正確に入力する必要がある。1文字でも入力ミスがあると構文エラーになり動かない。
- 意味不明の予約語が多い。"public", "class", "static", "void", "System.out" の意味をすべて理解できるのはしばらく学習を続けた後であり、それまでは「おまじない」として意味のわからない文字列を記述し続ける必要がある。
- 階層構造がある。クラスの中に関数があり、その中に命令がある多重の入れ子構造になっている。このような構造の理解は易しくない。

```
public class HelloWorld {
   public static void main(String args[ ]) {
      System.out.println(" こんにちは");
   }
}
```

図 1: Javaのプログラム例

```
窓＝フィールド！作る。
窓！『こんにちは』書く。
```

図 2: ドリトルのプログラム例

図2に同様のプログラムをドリトルで書いた例を示す。意味不明の予約語はなく、「テキストフィールドを作り窓という名前にする」「窓に"こんにちは" と書く」のように初心者でも素直に読み下すことができる。しかも、このプログラムでは新しいフィールドができてそこにテキストが表示されるため、普段使っているGUIプログラムと同様のものが作成できたという達成感が得られる。

### 3.2 ドリトルの主要な機能

図3にドリトルのプログラム例、図4にその実行画面を示す。以下で、この例に出てくるコードを主に用いて、言語の主要な機能を解説する。

#### オブジェクトの生成

```
カメ太 ＝ タートル！作る。
```

あらかじめ用意された "タートル" オブジェクトから新しいオブジェクトを生成し、"カメ太" という広域変数に格納する。変数の宣言は不要であ

```
カメ太＝タートル！作る。
カメ太！100 歩く 120 左回り 100 歩く 120 左回り 100 歩く 90 左回り。
三角形＝カメ太！図形にする。
三角形！(青) 塗る。
時計＝タイマー！作る。
時計！1秒 間隔 10秒 時間。
時計！「三角形！36度 右回り」実行。
```

図 3: プログラム例 (図形の回転)

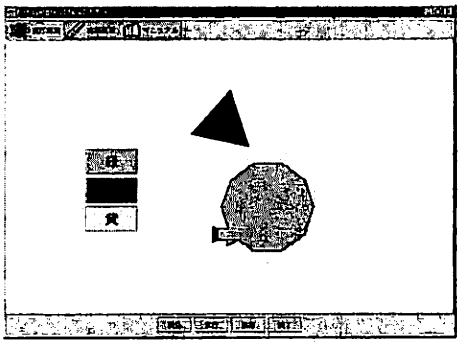


図 4: ドリトルの画面例

る。“作る” は自分の複製を作るメソッド (オブジェクトが持つ命令) である。新しいオブジェクトは元のオブジェクトをプロトタイプとして参照し、必要に応じて属性とメソッドを引き継いでくる。

### メッセージ送信

```
カメ太！100 歩く 120 左回り 100 歩く
120 左回り 100 歩く 120 左回り。
```

オブジェクトには複数のメッセージを送ることができる。この例はレシーバ “カメ太” に “100 歩く” というメッセージを送り、続けて “120 左回り” を送っている。

ドリトルのメソッドはオブジェクトを値として返す。“歩く” はタートルを移動させた後、値として元のオブジェクト (カメ太) をそのまま返す。返されたオブジェクトをレシーバとして次のメッセージが送られる。

### 図形の生成

```
三角形 ＝ カメ太！図形にする。
三角形！(青) 塗る。
```

タートルグラフィックスを拡張し、メソッド “図形にする” により、そこまでに描いた軌跡を図形オブジェクトに変換できるようにした。

ドリトルでは色もオブジェクトであり、“赤”、“青” など代表的な色があらかじめ定義されている。この例では “塗る” などのメソッド名と区別するために “( )” で囲んでいる。

### タイマーによる繰り返し

```
時計 ＝ タイマー！作る。
時計！1秒 間隔 10秒 時間。
時計！「三角形！36度 右回り」実行。
```

タイマーは一定間隔でブロックを実行するオブジェクトであり、これを用いて容易にアニメーションを実現できる。タイマーによる動作は非同期に実行されるため、複数のアニメーションを並行して実行することが可能である。

### 属性の定義

先の例題には現れていないが、オブジェクトの属性やメソッドを定義するコード例を示す。

```
カメ太：歩幅 ＝ 3。
カメ太：走る ＝ 「|n|！(n * 歩幅) 歩く」。
カメ太！10 走る。
```

オブジェクトに “歩幅” と “走る” という属性を定義している。属性に “「...」” のようなブロックを代入することでメソッドを定義できる。この例では “n” という引数を受け取るメソッドを定義している。レシーバが省略された場合は、そのメソッドを実行するオブジェクトがレシーバになる。

### 3.3 標準オブジェクト

表1に現在提供しているオブジェクトを示す。

表 1: 基本オブジェクトの一覧

| オブジェクト | 説明 |
|---|---|
| 文字列 | 文字列の表現と操作 |
| 数値 | 数値の表現と操作 |
| 論理値 | 論理値の表現と操作 |
| ブロック | メソッド定義、繰り返し、条件分岐など |
| 配列 | 集合データ表現 |
| タートル | タートルグラフィックスを描く |
| 図形 (パス) | 点や線の集合 |
| 色 | 3原色からの生成、混ぜ合わせなど |
| 画像 | 拡大、回転演算など |
| GUI 部品 | ボタン、テキストボックスなど |

## 4 動作環境

ドリトルの処理系は Java 2 [4] で記述されており、Java 2 が動くさまざまな環境で動作する。処理系は評価版として配布を行っており、インターネット経由でダウンロードして授業等で使用することが可能である [2]。本稿執筆時点での最新版のファイルサイズは約 300KB であった[1]。

## 5 情報教育における効果

本節では、ドリトルが想定しているユーザーである生徒と教員を対象に試験評価を行った結果について述べる。

### 5.1 高校での実験授業

筑波大学附属高校の1年生有志3人を対象に、3時間の授業を行った。1人は Visual Basic の使用経験があり、2人はプログラミングの経験がなかった。全員が基本的なキーボードやマウス操作を行えた。実施したカリキュラムを表2に示す。授業は次のように進めた。

1. テキストを配る。
2. 講師がプロジェクタで動かしながら説明する。
3. 生徒が手元のノート PC で例題を実行する。
4. 例題を修正する。
5. 課題のプログラムに挑戦する。

表 2: 実験授業のカリキュラム

| 時間 | 内容 |
|---|---|
| 1 | オブジェクトに慣れる<br>・オブジェクトの生成<br>・タートルグラフィックス<br>・図形オブジェクトの生成 |
| 2 | オブジェクトを意識したプログラム<br>・複数のオブジェクト<br>・タイマーによるアニメーション |
| 3 | メソッドの定義<br>・属性にメソッドを定義<br>・作品作り |

評価は授業中の生徒の観察とアンケート、作品プログラムから行った。

(1) 計算機の動作原理

生徒の以下のような感想から、プログラミングが計算機の動作原理を理解するための有効な手段であることを確認した。

- 「プログラムは上から順に実行される」(順序実行の概念)
- 「同じ動作を2回記述しないで済ませるのは便利」(繰り返しや手続きの必要性)
- 「画面でマウスカーソルが動くのは計算機の中でプログラムが動いているから」(OS の概念)
- 「画面は点からできているのではないか」(画素の存在)

(2) プログラミングの楽しさ

同様に、プログラミングという知的行為の楽しさを体験できたことが確認された。

- 「自分で考えたことが、うまく画面に現れるととても嬉しかった」
- 「どんなことをしようか決めてからプログラムを組むのはパズルのようで面白かった」

(3) 言語の習得

3時間という限られた授業の中で、オブジェクトの生成、メッセージ送信 (命令の実行)、メソッド定義 (手続きの概念) というプログラミングの基本概念が、実際に作品を作るレベルまで理解できたことを確認した。

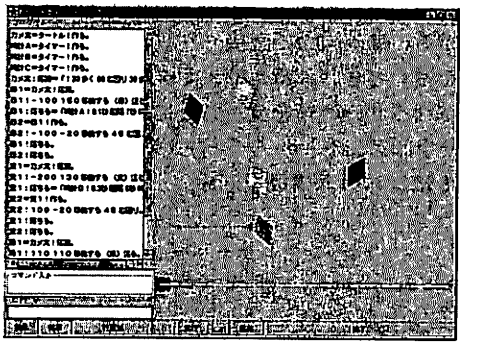
(4) 作品

[1] このほかに、Java 2 の実行環境 (JRE) が必要である。JRE は SUN のサイトから無償でダウンロードできる。

表 3: 教員の内訳

| 学校 | 人数 (うち未経験者) |
|---|---|
| 小学校 | 6 (3) |
| 中学校 | 7 (2) |
| 高校 | 2 (0) |
| 大学 | 2 (0) |
| その他 | 3 (0) |

タートルグラフィックを利用して、複数個のオブジェクトを同時に動かすアニメーションプログラム作品を全員が作ることができた。図 5 に作品例を示す。図形を落ち葉に見立てたアニメーションである。3 色の葉に落ち方がメソッドとして埋め込まれており、同じ色の葉は、1 枚目のメソッドを継承して動作する。オブジェクト指向の継承にあたる概念が無理なく使われている。



```
カメ太＝タートル！作る。
時計Ａ＝タイマー！作る。
時計Ｂ＝タイマー！作る。
時計Ｃ＝タイマー！作る。
カメ太：落葉＝「！30 歩く 60 右回り 30 歩く 120 右回り
    30 歩く 閉じる 図形にする」。
赤１＝カメ太！落葉。
赤１！-100 150 移動する （赤）塗る。
赤１：落ちる＝「「 ！18 右回り 5 -5 移動する」実行」。
赤２＝赤１！作る。
赤２！-100 -20 移動する 45 右回り。
時計Ａ！0.1秒 間隔 7秒 時間。
赤１！落ちる。
赤２！落ちる。
黄１＝カメ太！落葉。
黄１！-200 130 移動する （黄）塗る。
黄１：落ちる＝「「 ！30 右回り 10 -13 移動する」実行」。
黄２＝黄１！作る。
黄２！100 -20 移動する 45 右回り。
時計Ｂ！0.3秒 間隔 8秒 時間。
黄１！落ちる。
黄２！落ちる。
緑１＝カメ太！落葉。
緑１！110 110 移動する （緑）塗る。
緑１：落ちる＝「「 ！40 右回り -10 -10 移動する」実行」。
時計Ｃ！0.2秒 間隔 6秒 時間。
緑１！落ちる。
カメ太！ペンなし 90 左回り 200 歩く 90 右回り -250 歩く。
カメ太！ペンあり 500 歩く 図形にする （黄）塗る。
```

図 5: 生徒の作品例

### 5.2 教員による評価

小学校から大学までの教員に使ってもらい、感想を聞いた。表 3 に参加者 20 名の内訳を示す。15 名は LOGO、BASIC、Pascal 等でプログラミングの経験があった。

テキストは高校の授業で使った 3 時間分の資料を配布した。30 分で文法とオブジェクト指向の考え方を説明し、60 分間の実習を行った。

実験の結果、初心者を含めて全員がドリトルを使えるようになったことを確認した。

終了後の感想やアンケートでは、初心者の教員から「自分がプログラムを作れるとは考えてもみなかった。とても楽しかった」という意見が聞かれた。

## 6 オブジェクトを共有する教育環境の提案

学校における計算機を使った学習は、集合教育でありながら、生徒ごとに端末に向き合う形の個人的な世界になりがちである。生徒は、それぞれの端末の前でプログラミングに取り組み、与えられた課題を達成する。しかし我々は、プログラミ

ングとは、自分で使うだけでなく、他人の役に立ち、評価されるべきものであると考える。

そこで筆者らは、教室内でオブジェクトを共有することで、プログラミングの学習に共同学習の要素を取り入れることを提案する。

本稿では、教室や学校内などのLAN環境を想定する。端末同士で直接やり取りをすることは混乱を招くと思われるため、オブジェクトバンクと呼ばれるサーバーを置くことにより、端末との間でクライアント・サーバー方式の通信を実現する。図6にシステム構成を示す。

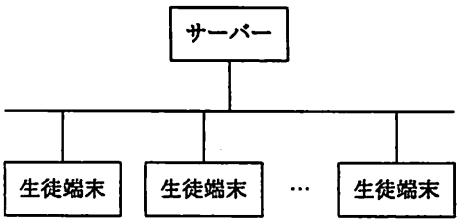



図 6: システム構成

## 6.1 オブジェクト共有の設計

以下に、オブジェクトを共有するときに用いるいくつかの方式と記述例を示す。

### 複製による交換

オブジェクトをサーバーと端末の間で複製することにより受け渡す方式である (図 7)。

**バンク!『カメ１号』 (カメ太) 書き込む。**

オブジェクト “カメ太” の複製を “カメ１号” という名前でサーバーに登録する。

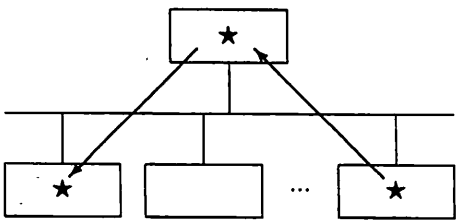



図 7: オブジェクトの複製

**カメ吉=バンク!『カメ１号』 読み出す。**

サーバー上のオブジェクト “カメ１号” をローカルに複製する。これにより、他の生徒が作ったプログラムを取り寄せて試してみることができる。

### オブジェクトの共有

サーバー上のオブジェクトを複数の端末で共有する方式である (図 8)。

**カメ太=バンク!『カメ２号』 共有。**

サーバー上のオブジェクト “カメ２号” をローカルで共有する。オブジェクトの実体はサーバー上にあるが、端末ではローカルにあるように操作することができる。これにより、ネットワークを経由した遠隔資源の透過的な利用や、情報の共有などを体験することができる。

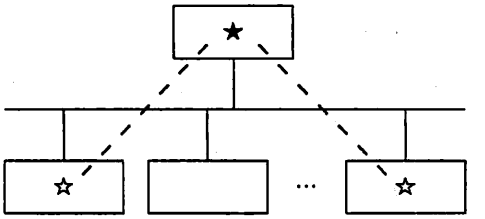



図 8: オブジェクトの共有

### 移動による交換

オブジェクトをサーバーと端末の間で移動することにより受け渡す方式である (図 9)。

**バンク!『カメ３号』 (カメ太) 入れる。**

オブジェクトを “カメ３号” という名前でサーバーに登録する。ローカルのオブジェクトは存在しなくなる。

**カメ吉=バンク!『カメ３号』 出す。**

サーバー上のオブジェクト “カメ３号” をローカルに移動する。サーバーのオブジェクトは存在しなくなる。これにより、分散環境においてもトークンのようなものを使うことでデータを矛盾なく保持できることが体験できる。

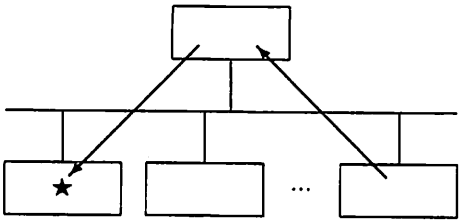


図 9: オブジェクトの移動

### オブジェクトの送信

オブジェクトを特定の端末に送る方式である(図 10)。

パンク！『端末１』(カメ太) 送る。

オブジェクトを“端末１”に送る。ローカルのオブジェクトは存在しなくなる。これにより、分散システムにおける非同期なイベントなどの概念を体験できる。

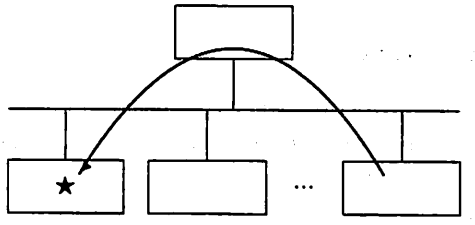


図 10: オブジェクトの送信

## 6.2 オブジェクト共有の効果

オブジェクト共有により可能になると思われることがらと、学習効果を列挙する。

### オブジェクトの交換

自作のオブジェクトをサーバーに登録し、他の端末から複製して使うことにより、オブジェクトの情報の交換、感想のフィードバックなど、生徒同士の交流が生まれてくる。

自作のオブジェクトが同級生の役に立つことで、「プログラミングは知的な生産行為であり、それによって社会に貢献できる」という感覚を養うことができる。これは社会における職業全般に通じる感覚である。

自作のオブジェクトを使ってもらうためには、オブジェクト自身に個性や特徴が必要であり、さらに同級生が必要としている特性を取り込んでおく必要がある。これは商品の企画やマーケティングの感覚を体験することにつながる。

また、ひとつのオブジェクトが複数の同級生に使われることで、「複製のコストが低い」というソフトウェアの特性を学ぶことができる。

### プログラム部品の活用

オブジェクトをサーバー上で永続的に保存することにより、ソフトウェア部品の蓄積につながる。使い捨てではなく、保存しておき再利用するプログラミングスタイルの意識が芽生える。別のプログラムで再利用することを意識したり、汎用的に作ることの意味を体験する。プログラミングを行う場合も、すべてを自作するのではなく、既存のプログラム部品を探してきて活用するスタイルになる。既存のオブジェクトを拡張して使うことはオブジェクト指向の利点を存分に活かすことに相当する。

### 共同作品作り

複数の生徒が共同で作品を作ることもできる。はじめは個人で作ったオブジェクトを持ち寄って組み合わせることから始まるが、そのうちに分担して作業するようになる。あらかじめ分担を決めることで、「設計」という概念が生まれる。また、お互いの分担を言葉で表現する必要があることから、仕様を客観的に記述する体験になる。

### オブジェクトを集めて実行

教師の端末などに生徒のオブジェクトを集めて実行することが考えられる。

アルゴリズムを工夫することにより、オブジェクトごとの強さを競うじゃんけんゲームなどが考えられる。

タイマーに合わせてオブジェクトを同時に実行することにより、紙芝居のように複数の登場人物が動作するアニメーションが可能になる。総合的な学習として、国語など他の教科での活用も可能

である。

教師の端末に特定の世界を表す環境を設定することにより、各種のシミュレーションが考えられる。生物ではライフゲームや遺伝のシミュレーションが、物理では物体の運動や惑星の運動など。少しずつ特徴を変えた生物的な性質や物理的な性質を持つオブジェクトを持ち寄り結果を観察することは、他の教科の学習に応用できるものと言える。

#### オブジェクトの共有

オブジェクトを複数の端末から共有して互いに操作することにより、ネットワークを介したオブジェクトの同期など、分散プログラミングの概念を体験できる。

ひとつのタートルを共有して図形を描くことや、テキストフィールドを共有して電子掲示板のように使うことなどが考えられる。

#### オブジェクトの移動

オブジェクトをサーバーから取り出し、操作して戻すことが基本になる。同時にひとりだけが操作できることから、排他の概念を体験できる。

#### オブジェクトの送信

相手の端末を指定してオブジェクトを送りつける。特定の相手とだけオブジェクトを共有したり、オブジェクトを順にまわすことによる回覧などに利用できる。

### 6.3 オブジェクト共有の実装

オブジェクトの共有を実現するために、オブジェクトをクライアントとサーバー間で受け渡す必要がある。今回の実装では、Javaのネットワーク機能である直列化 (Serialization) とリモート呼び出し (RMI) の機能を利用している。

ドリトルのオブジェクトは内部で相互に結び付いた形で存在する (図 11)。あるオブジェクトを渡すときに、そのオブジェクトに関連したどの範囲のオブジェクトを渡すかなどを、今後実験を通して検証する予定である。

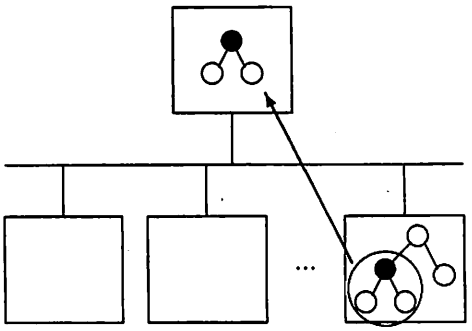


図 11: 共有/複製の範囲

## 7 おわりに

教育用に適したオブジェクト指向言語を考案し、実装と評価を行った。今後はオブジェクト共有の機能を充実させていく。初心者でも手軽に利用できることから、教育現場での利用を期待したい。

## 参考文献


[1] ECMAScript Language Specification. http://www.ecma.ch/ecma1/stand/ecma-262.htm.

[2] プログラミング言語「ドリトル」. http://www.logob.com/dolittle/.

[3] Adele Goldberg and David Robson. *Smalltalk-80: The Language and Its Implementation*. Addison-Wesley, 1983.

[4] James Gosling, Bill Joy, and Guy Steele. *The Java Language Specification*. Addison-Wesley, 1996.

[5] David Ungar and Randall B. Smith. Self: The power of simplicity. In *OOPSLA'87*, pp. 227–242, 1987.

[6] 兼宗進, 久野靖. 学校教育用オブジェクト指向言語/環境の構想について. 情報教育シンポジウム (SSS2000), pp. 79–82, 2000.

[7] 兼宗進, 久野靖. 学校教育用オブジェクト指向言語 "Dolittle" の提案. 第 42 回プログラミングシンポジウム, pp. 11–20, 2001.