

Java プログラミング教育統合環境「Java Editor」

高岡詠子、米田毅浩、澤田英敏、山本啓介
千歳科学技術大学

千歳科学技術大学では、来年度、Java プログラミング言語入門を目的とした授業を単位認定型完全 e-learning 形式で行うことが決まった。完全 e-learning 授業においてもプログラミング能力をつけさせるため、Java のインストールのように学習者が手間とを感じる箇所の簡略化を行い、学習者がコーディングを行いながらプログラムを作成する手順、コンパイル時に発生するエラー文の読み方と取り除き方を学ぶことができる Java プログラミング教育統合環境「Java Editor」の紹介を行う。

Java Editor : An Integrated Environment for Java Novices

E. Takaoka, T. Yoneta, H. Sawada and K. Yamamoto
Chitose Institute of Science and Technology

Abstract

In this paper, we introduce Java Editor, an integrated environment of Java programming for novices. In the next year, Java programming course shifts face-to-face class to school credit certification by completely e-learning class. Java Editor provides them some features so that students should achieve programming skills even if completely e-learning style. We will apply our system to department junior year on the next spring semester.

1. はじめに

千歳科学技術大学では、平成 16 年度に採択された現代的教育ニーズ取組支援プログラム、テーマ 6 : IT を活用した実践的遠隔教育 (e-learning) において全学的に e-learning プロジェクトが進められている [1]。このプロジェクトにおいて、情報講義系科目の一つであり学部 3 年生を対象とした「ソフトウェアデザイン」では、Java プログラミング言語入門の授業を平成 18 年度に単位認定型完全 e-learning で行う。この科目の担当は筆者であり、他の科目はブレンド型 e-learning をめざす中、今回のプロジェクトにおいては唯一完全 e-learning 化を行う科目である。コンテンツ作成は筆

者の研究室でおこなっている [4]。来年度の授業形態を視野に入れて、17 年度春学期、試験的に e-learning システムを導入してブレンド型で授業を行った [2]。アンケートの結果からは、本大学での e-learning の利用価値は学生には十分認められており、学生に対する利用促進によって、自主学習での有効利用性も高まるであろうことが予測されている [3]。

来年度本格運用に向けて浮上した問題は、完全 e-learning 授業においてプログラミング能力をつけさせるためのなんらかの仕組みが必要であるということであった。実際、プログラミング言語のような分野を学習する際には、実際に講師からコーディング方

法についてのメッセージやヒントなどの指導をしてもらうことが重要である。ソフトウェアデザインを完全 e-learning に移行すれば、スクーリングやメンタによる指導は受けられるとはいえ、学習者は基本的に個人で学習を受けることとなる。以下のような問題点が生じてくるであろう。

- プログラミングの概念の理解ができない
- Java 等言語自体のインストールができない
- プログラミングの技術が身につかない
- コンパイルエラーが取れない

一方、本研究室では昨年「コーディングしながら学ぶ」ことを実現するために PC マエストロ HTML バージョンの構築を行った[5]。この PC マエストロ HTML バージョンには、エディタ機能とその他の教材が実装されており、実際に学習者にアンケートを取った結果、「コーディングしながら学ぶ」ことの有効性が確認されている。

このような背景をもとに、完全 e-learning 型授業の支援システムとして Java プログラミング言語教育統合環境「Java Editor」を開発した。これは「コーディングしながら学ぶ」ことが実現できるよう、Java 言語学習者を対象とした、エディタと教材を含む学習システムであり、Java の実行環境・環境変数の設定などを全てパッケージ化することにより、学習者が Java のインストールに費やす時間・手間がなくなり、抵抗を感じることなくプログラムの勉強段階に入っていくことができるような環境を実現したものである。

2. Java Editor 概要

Java プログラミング教育統合環境「Java Editor」は実際にプログラムを記述する「専用エディタ」部と、プログラムの流れを学習することができる「コーディングフロー学習システム」、文法エラーを解析しプログラミング初心者向けに文法のエラー箇所を

示す機能を持つ「Java コードアナライザ」を備えた自学用教材である。学習者は Java プログラムの大まかな流れをコーディングフロー学習システムで学び、その後実際にエディタを用いてプログラムを記述し、コンパイル・実行するという一連の流れを一貫して本システム上で学ぶことができる。

3. 設計方針

本学におけるプログラミングの授業は、C 言語、Java 言語ともに統合環境は使わず、Linux 上で vi を使ってプログラムを書き、コンパイル・実行に関しては、シェル上でコマンドラインから gcc, javac といったコマンドを打ち込むことで行っている(一部、物質光科学科系では Visual Basic 等を使っている授業もある)。ソフトウェアデザインの授業は、学習者が Java 言語を使ってプログラムを書き、コンパイルして実行できるようになることを目的としている。アルゴリズムの習得に関しては秋学期の「アルゴリズム論」という別の授業で重点的に行うことになっている。

学内のコンピュータ環境は Windows XP と LinuxOS のデュアル OS であり、学生のアクセス認証およびファイルシステムは統一管理されており、どちらの OS からでも学生個人のファイルシステムにアクセスできるようになっている。2つの実習室は合わせて 160 台、その他図書室には 40 台のコンピュータが設置されており、学生は実習室での授業時間を除けば、時間内(平日 9 時~19 時)であればいつでも自習することが可能である。

ソフトウェアデザインの授業が完全 e-learning 化されると、学習者は、プログラミング言語で扱われている語句の意味、もしくは概念などといった知識は e-learning コンテンツ内の教科書やドリルから学ぶことができるが、実際にプログラムを書く機会を自分で作らなければならない。本年度のアンケートによれば、授業時間外に e-learning を主にどこで使用したか

の質問には大学が 50%、家が 38%、両方が 9%であることから、学生は大学でも家でも気軽に e-learning にアクセスできる環境にあることがわかっている[3]。学内で自習する分には Java 言語をインストールする必要はないが、自宅で Windows を使って自習する学生が多いことを考えると、まずは Java の環境を整えるところから始めることになるだろう。しかし、jdk のパッケージをダウンロードするということにたどり着くことができない。たとえたどり着いたとしても英語のページであったり、ダウンロードできて解凍できたとしても、パスの設定に手間取り、環境を整える頃には疲れきってしまい、プログラミングの学習を始めてもいないのにプログラミングが嫌いになってしまうという弊害が起こる。これを回避するため、Java Editor は Windows 上で動く必要があり、かつ Java の実行環境を内包する必然性に迫られた。

一方、プログラミングの学習では、開発言語の基本的な特徴・構文の書き方を学習し、実際にプログラミングを行うという手順を踏む。ここでいう「プログラミング」とは、ある目的を達成するためにプログラムを実際にコーディングし、プログラムを完成させることである。「問題解決を行うために、あるアルゴリズムに従ってプログラムを書く」プログラミングの主な作業はこのアルゴリズムを実装することである。しかし、多くの場合、アルゴリズムのエラーの前に構文エラーがなくなると、アルゴリズムが正しいのかさえ確認することができない。

今年度のソフトウェアデザインにおいて「アルゴリズムを実装する」以前に学習者がつまづいていた点は大きくわけて、自宅で Java の環境が構築できない コンパイルをしたときに表示されるエラー文を理解できない コンパイルエラーを一人で取れない。の 3 点である。 に関しては、セミコロンの付け忘れや括弧の数が合わない、予約語のスペルミスなどの初歩的なミスが

ほとんどであった。

従来の対面型授業においてはこれら学習者のつまづきのフォローは実習室において教師・TA が担っていた重要な役割であった。授業が完全 e-learning 形式になった場合、このフォローに対応するものが必須となる。したがって学習者のつまづきフォロー機能として以下のような環境を構築した。

- Java 環境の内包
専用エディタの中に Java 環境を整えることで、学習者が自宅等自分のコンピュータ環境で Java 言語の学習を始める際に抵抗を感じる部分である、Java 言語のインストールを行う必要がないようにすることにより、学習者がプログラムの学習を始める前に諦めてしまうことを避けることができる。
- コンパイルエラーのフィードバック
Java コードアナライザを専用エディタに組み込み、初心者にとって難解な表現をするコンパイルエラーのメッセージを初心者向けにカスタマイズして出力することで、コーディングのサポートを行う。
- コメントアウト・予約語の色付け
コメントアウト部や予約語に色をつけ、間違いがすぐにわかる。
- 行検索・行強調機能
コンパイルエラーが出た行にすぐに行けるような行検索の機能と、検索された行をハイライトする機能である。
- エディタ機能（ファイル操作、印刷等）
- コンパイル前のチェック機能
スペルミスやセミコロンの付け忘れ、括弧の数のチェックなどの初歩的な文法的誤りを発見し、ポップアップとして画面内に表示することができる機能

- Shell 機能

コンパイル・実行作業に関してはコマンドプロンプトの扱い方をプログラムと同時に学ぶようにするため、自動コンパイル機能は実装せず、手動でコンパイル、実行を行うようになっている。

4. Java コードアナライザ

Java コードアナライザは、専用エディタに組み込むことにより、エディタで編集したプログラムをコンパイルする前に、初歩的な文法ミスを指摘したり、コンパイラの出す難解なエラーメッセージを初心者向けにカスタマイズして出力することでコーディングをサポートすることができる。本システムは単体で使うこともできる。以下に詳細を述べる。

4.1. Java コードアナライザの機能

4.1.1. エラーメッセージのカスタマイズ

既存の javac コマンドが出力するエラーメッセージを解析し、既存のエラーに対応する初心者にわかりやすいオリジナルメッセージを出力する。付録 1 にオリジナルメッセージの一部を記す。

4.1.2. エラーが発生する原因の予測

1つの原因から複数のエラーが発生したり、エラーメッセージの内容とは全く別の原因によってエラーになることがある。これらのエラーメッセージコードの組み合わせを記録した辞書を参照して、エラー原因を予測し表示する（付録 2 参照）。

4.1.3. 発生したエラー情報の蓄積

Java コードアナライザを用いた場合に発生したエラーは、専用に構築したサーバに転送されデータベース化される。サーバでは発生エラーをカウントし、発生しやすいエラーを検出することができる。これをエラーメッセージのカスタマイズに反映させることが可能である。

4.1.4. 設定 XML の更新機能

オリジナルエラーメッセージは XML フ

ァイルに記述されているが、このファイルが更新された場合、ネットワークを通じて最も新しいファイルをダウンロードし更新する機能を持っている。これによりエラーメッセージファイルを更新しても、学習者は最新のオリジナルメッセージを利用可能である。

4.2. エラーメッセージの例

図 1 の Java のプログラムをコンパイルした場合、this は予約語として定義されているためコンパイルエラーとなる。しかし既存のコンパイラのエラーは、「8 行目が文ではない」とか、セミコロンがあるにもかかわらず「8 行目にセミコロンがない」というものである。初心者は一生懸命 8 行目を見るだろうが、文になっているし、セミコロンがあるのでさっぱりわからないという状態になってしまう(図 2 参照)。

```
1: import java.io.*;
2:
3: public class Sample {
4:     public static void main(String[] argc){
5:         int i;
6:         String string = "";
7:         string = "hello";
8:         int this = 1;
9:         System.out.println( string );
10:    }
11: }
```

図 1 プログラム

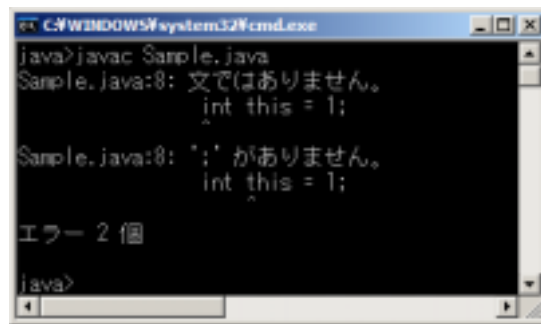
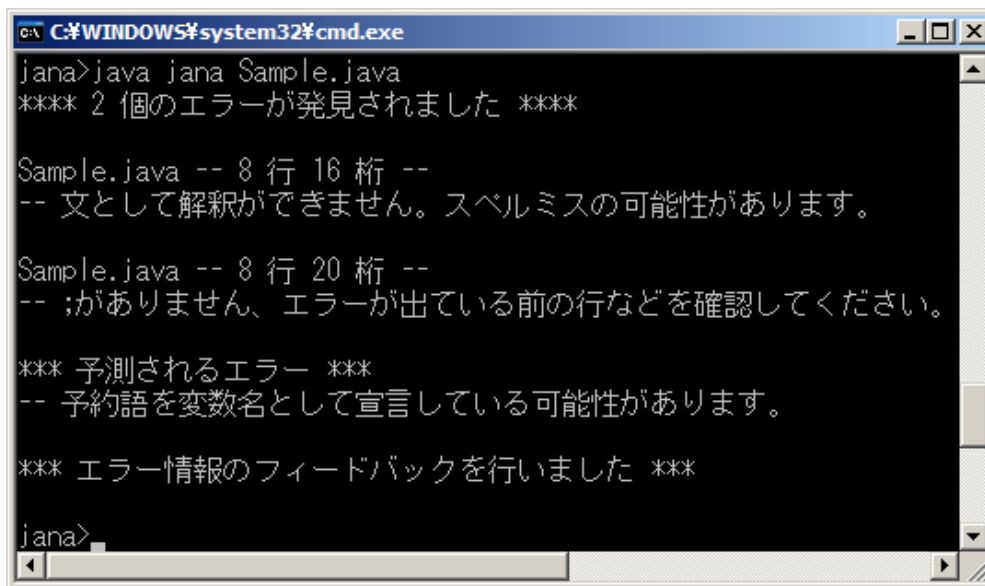


図 2 既存のコンパイルエラー

これを、付録 2 の 2 行目にあるように「予約語を変数名として宣言している可能性があります。」という出力にすることでわかりやすい表記にする(図 3 参照)。



```
C:\WINDOWS\system32\cmd.exe
jana>java jana Sample.java
**** 2 個のエラーが発見されました ****

Sample.java -- 8 行 16 桁 --
-- 文として解釈ができません。スペルミスの可能性があります。

Sample.java -- 8 行 20 桁 --
-- ;がありません、エラーが出ている前の行などを確認してください。

*** 予測されるエラー ***
-- 予約語を変数名として宣言している可能性があります。

*** エラー情報のフィードバックを行いました ***

jana>
```

図 3 オリジナルエラー

5. Java Editor による学習の流れ

5.1. インストール

本研究で開発した「Java Editor」はアプリケーションソフトであるため、学習者は使用時にシステムのインストールを行う必要がある。はじめに「JavaEditor.exe」というファイルをダウンロードし任意の場所に置く。その後、インストール手順に従うことで、Java 言語をインストールする手間やパスの設定等をする必要が無く、初心者でも簡単に学習者の PC に Java 学習を行う環境を整えることができる。標準でスタートメニュー登録、デスクトップへのショートカット作成を行う。

5.2. 学習の流れ

「Java Editor」にはコーディングフロー学習システムと専用エディタの 2 つのコンテンツが含まれており、学習者はどちらかのコンテンツを選択し学習する。Java コードアナライザは専用エディタに組み込まれている。コーディングフロー学習システムは、サンプルソースを表示し、そのソースを確認しながら学習を進めていく教材である。専用エディタは、実際にプログラムを記述することで学習を進めていく。プログラムを記述し、チェックボタンを押すこと

で画面に表示されるエラー画面を見てスペルミスや括弧の数の間違いがないことを確認した後、コマンドプロンプトを起動し、実際にコマンドを打ち込むことでコンパイル・実行をして、出力を確認する。それを繰り返し行うことでコーディングの能力を身に付けていくことができる。各画面、ボタンの機能について以下に詳細を示す。

5.3. 機能詳細

5.3.1. エディタ

エディタ画面を図 4 に示す。ここに学習者はプログラムを書き込んでいく。このエディタはリアルタイムでコメントアウトを判断し色を付けることができる視覚的なサポート機能も実装している。画面下にあるクラスボタンを押すことで複数のプログラムを同時に記述することも可能となっている。また、「DOUBLE」ボタンを押すことで、画面を二分割することも可能である。これらの機能を用いて、以前に書いたプログラムや講師などから配布されたサンプルを表示し、見比べながらプログラムを記述することもできるようになっている。

5.3.2. シェル機能

「コマンドプロンプト」を起動する。Java 実行環境などをあらかじめ構築してあるた

め Java 環境を整えていない PC でもコンパイル、実行が可能である。

5.3.3. 編集機能

「オープンダイアログボックス」を表示して、任意のファイルを開き編集する、「セーブダイアログボックス」を表示して編集したファイルを保存する、編集しているプログラムを消去する、編集したプログラムを接続されているプリンタへ出力し、テキスト形式で印刷する等、エディタとしての標準的な機能はひとつおりに備えている。

5.3.4. エディタ分割

図 4 右上に示す DOUBLE ボタンを押すことでエディタ画面を分割、統合する。プログラムを見比べながら、コーディングを行うことができる。起動時には「SINGLE」画面である。

5.3.5. ヘルプ

「？」ボタンにマウスをあわせると各ボタンの使い方などの説明が書かれた「ヘルプ」画面が表示される。

5.3.6. 行検索・強調

検索したい行をテキストボックスに入力しボタンを押すと、検索した行が強調される。スペースで区切ることで複数行検索もできるようになっている。

5.3.7. 文法チェック機能

文法解析を実行し、その結果をポップアップとして画面(図 5)に表示する。同時に予約語の走査を行い色を変化させる。図のプログラムでは、4 行目にセミコロンがない。標準で出されるエラーは「5 行目に ';' がありません。」である。初心者は 5 行目にセミコロンがないかどうか一生懸命見るだろうが原因は 1 行前である。本システムでは図に示すように、前の行などを確認するような指示をエラー表示として出力する。

5.3.8. クラスボタン

図 4 の左下にあるクラスボタンによってファイルを複数編集することも可能である。

6. 結論と展望

本稿では Java プログラミング教育統

合環境 Java Editor について述べた。「Java Editor」は来年度春学期の単位認定型 e-learning 形式で行われるソフトウェアデザインの授業で本運用されることが決まっている。運用後の教員・学生の評価により、本システムの運用方法、Java プログラミングの完全 e-learning 化の限度・可能性などが明らかになるだろう。

今後の方針としては、ケアレスミスや記述ミスなど多種のエラーの統計データと詳細、コンパイル回数、作成時間などの推移など、本システムを利用した学習者の学習履歴を取得するシステムを構築する予定である。これにより、講師はテストなどを行う際の難易度の設定や多数の学生が苦手だと感じたところへ集中してサポートを行うことができ、学習者は自分の苦手な箇所やケアレスミスが多い箇所のチェックがしやすくなる。運用・評価・改善を繰り返していくことでプログラミング教育統合環境としてさらなる発展をめざす。

References:

- [1] 高岡詠子, "大学におけるデジタルコンテンツ開発の変遷および e-learning プロジェクト", 情報教育シンポジウム SSS2005 プレカンファレンス論文集, pp. 46—52(2005).
- [2] 高岡詠子, 石井和佳奈, "Java プログラミング入門単位認定型完全 e-learning へ向けての試み～コンテンツ構築および実践バージョン～", 情報処理学会研究報告, CE-81, pp.73—80(2005).
- [3] 高岡詠子, 石井和佳奈, "Java プログラミング入門単位認定型完全 e-learning へ向けての試み～評価バージョン～", 情報処理学会研究報告, CE-82, pp. 53—60(2005).
- [4] 高岡詠子, 石井和佳奈, "Java プログラミング入門 e-learning コンテンツの紹介", 第 4 7 回プログラミングシンポジウム, 情報処理学会, pp. 167—172(2006).
- [5] 高岡詠子, 佐藤 威, "専用ブラウザを使ってコーディングしながら学ぶ「PC マエストロ HTML バージョン」", 情報処理学会研究報告, CE-79, pp. 25—32(2005).



図 4 エディタ画面

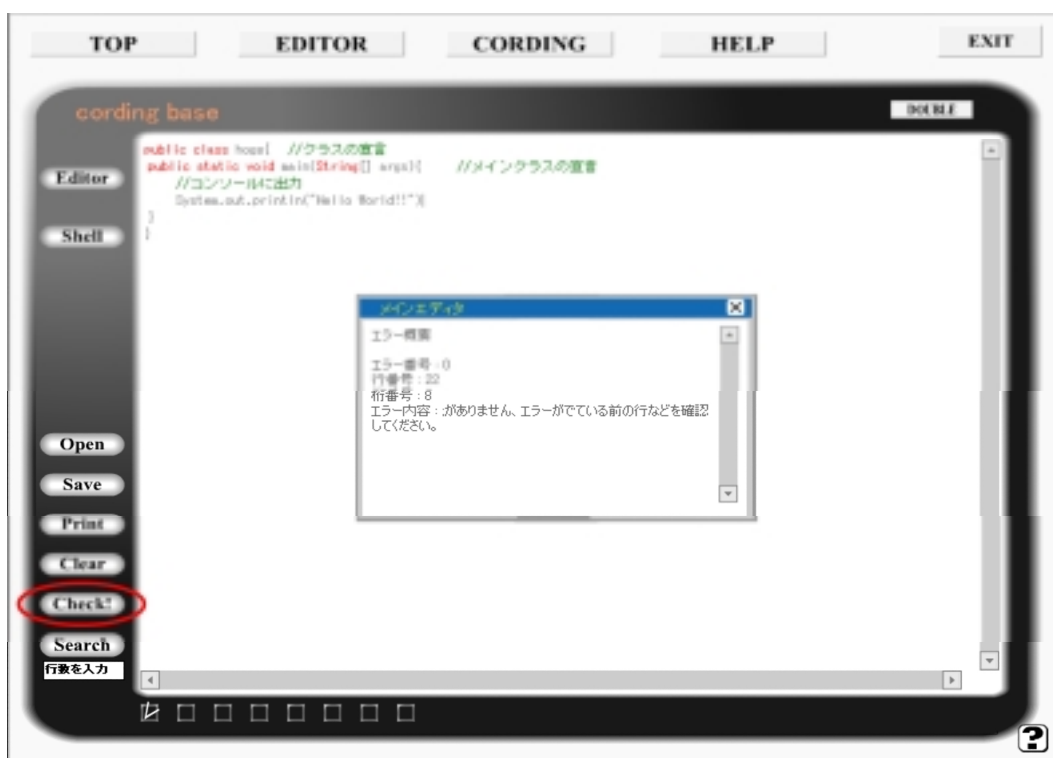


図 5 文法チェック画面

付録 1 : Java コードアナライザのオリジナルメッセージ抜粋

既存の Java コンパイラのエラー	Java コードアナライザのオリジナルエラー(エラーコード)
;がありません	";"がありません、エラーが出ている前の行などを確認してください。(100)
キーワード "class" または "interface" がありません。	"class"または"interface"が記述されていません。"class"や"interface"の綴りミスが無いか確認してください。(105)
)'がありません。	")"がありません。開始と終了の括弧の数が一致していません。(106)
識別子がありません。	メソッドやクラスの宣言に間違いや綴りミスがあります。public class **** や public static void *** などではなりません。また、public などの識別子は全て小文字で記述します。(107)
文ではありません。	文として解釈ができません。スペルミスの可能性があります。(600)
式の開始が不正です。	式として認識できません。エラーが発生している行に適した記述を行っていません。(例：メソッドの中でメソッドを定義している)(601)
型の開始が不正です。	型として認識できません。型が記述されるところに型の記述が行われていません。(補足: void は関数引数の型として宣言することはできません)(602)
¥12288 は不正な文字です。	全角スペースが入っています

付録 2 : エラーメッセージの組み合わせ

(上の表のエラーコードの組み合わせを左に、右側にアナライザのオリジナルエラーを示す)

エラーコード	Java コードアナライザのオリジナルエラー
100	オブジェクトとメソッドの間に"."が無い可能性があります。
100,600	予約語を変数名として宣言している可能性があります。
100,600,601	for 文の制御式に";"が 2 つ以上記述されている可能性があります
600,601	"."と";"を書き違えている可能性があります。
100,105,106,107,602	配列の初期化付き宣言がある場合、括弧などのスペルミスがある可能性があります。