DEIM Forum 2010 F8-3

教育指向 Java コンパイラの試作

服部　峻†　山崎　正貴†　亀田　弘之†

† 東京工科大学 コンピュータサイエンス学部
〒 192–0982 東京都八王子市片倉町 1404–1
E-mail: †{hattori,kameda}@cs.teu.ac.jp, ††c0106400c8@css.teu.ac.jp

あらまし　今日，ソフトウェアは重要な社会基盤の一つであり，極めて高いレベルでソフトウェアを開発することが出来る人材が求められており，大学をはじめ多くの教育機関でソフトウェア（プログラミング）教育が盛んに行われている．しかしながら，コンピュータサイエンス（情報技術）系の学生にも関わらず，プログラミング能力が十分に身に付いていないだけでなく，プログラミング嫌いである者が増加して来ている．主な原因として，プログラミングを初めて学ぶ際，初学者にとっては難解で取っ付き難いコンパイルエラーに直面することが挙げられる．また，プログラミング演習時，多種多様なレベルの学習者からの質問に対して，教員や TA によるアドバイスが必ずしも適時，適切ではないことも一因であると考える．そこで我々は，従来のコンパイラのようにソースコードだけに依存して静的なコンパイルエラーを出すだけでなく，ソースコードの先のユーザや状況も考慮することで初学者にも易しいアドバイスを適時（動的）に呈示する教育指向 Java コンパイラを試作する．

キーワード　コンパイラ，コンパイルエラー，教育指向，Java，プログラミング教育, タンジブル・ソフトウェア教育

A Prototype of Education-oriented Java Compiler

Shun HATTORI†, Masaki YAMAZAKI†, and Hiroyuki KAMEDA†

† School of Computer Science, Tokyo University of Technology
1404–1 Katakura, Hachioji, Tokyo 192–0982, Japan
E-mail: †{hattori,kameda}@cs.teu.ac.jp, ††c0106400c8@css.teu.ac.jp

Abstract　Software is an essential infrastructure today. Our society requires capable human resources who can develop softwares at the highest level possible, and various educational institutions such as universities provide software engineering (programming) education aggressively. However, those who do not acquire enough programming ability and also hesitate in programming have been increasing in recent years, even though they are Computer Science (CS) or Information Technology (IT) students/graduates. A main reason behind this is that they could not avoid facing very difficult compile errors when they were new learners of programming. Another reason is that teachers and TAs (Teaching Assistants) cannot always give timely and appropriate advices to varying levels of learners. In this paper, we develop a prototype of education-oriented Java compiler to present not only static compile errors as-is, but also easy-to-understand advices appropriately and on a timely basis (dynamically).

Key words　Compiler, Compile Error, Education-oriented, Java, Programming Education, Tangible Software Engineering (SE) Education

1. はじめに

今日，ソフトウェアは重要な社会基盤の一つである．例えば，事務職員は Microsoft 社の Word や Excel などのオフィス（文書作成）ソフトが無いともはや仕事にならないであろう．また，研究者にとっても LaTeX や Adobe 社の Acrobat などの PDF 文書作成ソフト，Microsoft 社の PowerPoint などのプレゼンソフトは必須に近い．他にも，OS や Web ブラウザなど広く一般ユーザに必要なソフトウェアから，個々人の要求を満たしてくれるものまで，ソフトウェアへの依存度は大きくなる一方である．ソフトウェア産業で現在開発されているソフトウェアは 1000 万行を超えるものも珍しくないが，その品質はソフトウェアが非常に重要な社会基盤となっているために極めて高いレベルが要求される．このような社会的需要に応えられるソフトウェア開発の人材を育成するため，大学をはじめ多くの教育機関でソフトウェア（プログラミング）教育が熱心に行われて

いる．しかしながら，コンピュータサイエンス（情報技術）系の学生にも関わらず，プログラミング能力が十分に身に付いていない者だけでなく，プログラミングすること自体が嫌いである者まで増加して来ている．この状況を打破するべく，我々はタンジブル・ソフトウェア教育の研究プロジェクトを立ち上げ，時代に則したソフトウェア教育に関する研究・開発を行っている．本稿ではそのうち，プログラミングの初学者を念頭に置き，従来の一般的なコンパイラのようにソースコードだけに依存して不親切で固定的なエラーメッセージを出すだけでなく，ソースコードの先のユーザや状況も考慮することで初学者にも易しいアドバイスを適時に呈示する教育指向 Java コンパイラのプロトタイプシステムを試作したので報告する．

理工系の学生でありながらプログラミングが嫌い，あるいは，不得意な学生が少なからずいる原因として，我が国における理科離れや工学離れの影響もあるが，プログラミングの初学時に英単語や専門用語が混じった難解で予想外に大量なコンパイラのエラーメッセージに直面し諦めてしまうことが挙げられる．従来のコンパイラはプログラミング教育ではなくソフトウェア開発にその主眼があり，基本的には中級者以上のユーザを想定し，エラーメッセージもプログラミング言語を既習であることが前提となっている．しかしながら，初学者にとっては，プログラミング言語はもとより，プログラミング自体に関する知識も不十分な場合があり得るため，従来のコンパイラの不親切なエラーメッセージだけでは，自分が書いたプログラムの何が問題でエラーが出たのか，どのように修正すればエラーを取り除けるのかを自力で学習して行くことは容易ではない．従来の一般的なコンパイラが開発者を想定して呈示する静的なエラーメッセージはソースコードの内容に依存するだけである．ソースコードの先で実際にエラーメッセージを読むユーザがどのようなレベルなのか，どのような知識を持っているのか，どのような状況にいるのか（例えば隣に TA がいたり教科書片手だったり），などを考慮してプログラミング初学者にも理解し易い動的なアドバイスを提供してはくれるような知的なコンパイラは見当たらない．ソースコード中のエラー数に比べて非常に多過ぎるエラーメッセージが返される場合も少なくなく，中・上級者にとってでさえ参考にならないエラーメッセージも含まれているのが現状である．

我々の大学では，コンピュータサイエンス学部の 1 年生を対象に，Java 言語によるプログラミング入門を座学と演習をセットにして開講している．演習ではいくつかの課題が与えられ，Emacs などのエディタでソースコードを書き，CUI（コマンドライン）で javac によってコンパイルを行い，エラーがあれば修正して再コンパイルといったプロセスを繰り返すことで，Java 言語によるプログラミングを習得して行くことになる．しかしながら実際には，コンパイルエラーで立ち往生してしまう初学者が多々見られる．最も一般的な Java コンパイラである javac が呈示するエラーメッセージの不親切さ，難解さが，少なくとも我々の大学でプログラミング嫌いを生んでいる原因の一つになっていると考えられる．この障壁を適切に下げることでドロップアウトする者を減らすため，1 クラス約 60 名の学生に対して教員 1 名および院生 TA（Teaching Assistant）4 名で，コンパイラのエラーメッセージを学生の代わりに読み解いてやり，学生のソースコードの何が問題でエラーが出たのか，どのように修正すればエラーを取り除けるのかといったアドバイスを適宜与えるなどのサポートを行っている．このような教員や TA など中級者以上によるサポートによってプログラミング嫌いの増加を抑制する効果は幾分はあると考えられるが，次のような限界も存在する．

- 学生数に対して十分な TA を配備できていないと，学生からの質問がバーストしてしまった場合に対応し切れず，適時にアドバイスを与えることができない
- 学生へのアドバイス方法などについて教員から TA への指導が不十分であると学生からの質問に対して必ずしも適切で一様な回答ができない
- コンパイルエラーで立ち往生しているにも関わらず教員や TA に質問できない（人見知りな）学生もいる

以上のような知見から，ソフトウェア開発ではなくソフトウェア教育を重視し，従来のコンパイラのように初学者には難解で固定的なエラーメッセージを出力するだけではなく，教員や TA など中級者以上が常にマンツーマンで傍に付いているかのように，（エラーの解決策をそのまま教えるなど）過剰に手助けし過ぎるでもなく，（エラーの解決策に出来る限り学習者自身で辿り着けるように）適切に適時にアドバイスを呈示してくれる，より知的なコンパイラが必要であると考える．

もちろん，我々の大学のように Emacs で編集し，javac でコンパイルするといった CUI な演習環境ではなく，入力補完機能や高度なデバグ機能を備えた Eclipse [1] などの GUI な統合開発環境を利用することで，タイプミスに起因するコンパイルエラーを防ぐ効果などを享受することができる．また，初学者向けのプログラミング言語 [2], [3] や初学者向けのプログラミング開発環境 [4] などの研究も行われている．

しかし，我々の大学での初学者向けプログラミング演習では，Java 言語でプログラムを単に書けるようになることだけが教育の目的ではなく，タイピングの練習や CUI への慣れ，エラーメッセージからソースコードを学習者自身が考えて修正する論理的思考なども目的であり，Emacs エディタでソースコードを作成し，コマンドラインで Java コンパイラの javac を用いてソースコードをコンパイルするというプロセスがコンピュータサイエンス学部のカリキュラムとして強く想定されており，このプロセス自体を教員個々の判断で変えることは難しい．そこで本稿で我々は，Java 言語の開発元であるサン・マイクロシステムズ社が提供している Java コンパイラの代表格である javac のエラーメッセージに対して，演習時に教員や TA が行っているようなアドバイスを学習者に呈示することが可能な教育指向 Java コンパイラを目指し，プロトタイプシステムを試作する．

本論文の以下の構成を示す．2 章では，従来のコンパイラおよび演習形態の問題点について述べる．3 章では，教育指向コンパイラの設計として，その要件や基本構成について述べる．4 章では，試作した教育指向 Java コンパイラの動作例を示す．最後に，5 章で本稿をまとめ，今後の課題についても述べる．

2. ソフトウェア教育における従来のコンパイラおよびプログラミング演習形態の問題点

本章では，従来の Java コンパイラである javac のソフトウェア教育（プログラミング学習）における問題点，及び，我々の大学でのプログラミング演習の授業形態に関する問題点を考察し，教育指向 Java コンパイラの試作に向けて要件を抽出する．

2.1 Java コンパイラ javac の問題点

以下のソースコード HJW.java は典型的な Java 入門プログラムである．我々の大学のコンピュータサイエンス学部の新入生は，Emacs などのエディタでそのまま打ち，javac でコンパイルしてみるようにまず指示されることになる．

HJW.java（模範解答）

```
class HJW {
  public static void main(String[] args) {
    System.out.println("Hello Java World");
  }
}
```

しかし，何も考えずに，そのまま打てば良いだけのはずの課題でもミスは起きる．例えば，ある初学者は以下のように予約語「class」を「clas」とタイプミスしてしまうかもしれない．実際，英語（英単語）が不得意な学生も多く，英単語から成る予約語のタイプミスは非常に多い．

HJW.java（タイプミス）

```
clas HJW {
  public static void main(String[] args) {
    System.out.println("Hello Java World");
  }
}
```

学習者自身はそのまま打ったつもりであり，コンパイルの成功を疑わないであろう．しかし，もちろん，このタイプミスを含んだ Java プログラムソースを（JDK1.6 の）javac でコンパイルすると，以下の 3 個のエラーが出てしまうため，予想外の出来事に驚き，戸惑うことになるかもしれない．

javac HJW.java（タイプミス）

```
HJW.java:1: class、interface、または enum がありません。
clas HJW {
^
HJW.java:2: class、interface、または enum がありません。
  public static void main(String[] args) {
                  ^
HJW.java:4: class、interface、または enum がありません。
  }
  ^
エラー 3 個
```

これらのエラーが初学者にとっても理解し易く，エラーを修正する方法を独力で見付けることが可能であれば問題無いが，立ち往生してしまう初学者は実際に存在する．以上の例の場合，従来の一般的な Java コンパイラである javac には次のような問題があると我々は考える．HJW.java は最も初歩的な入門プログラムであるが，些細な（但し学習者自身では気付き難いかもしれない）タイプミスによって，1 番目のエラーメッセージ中，やや高度な予約語「interface」や「enum」に直面してしまう．初学者にとっては，これらの高度な予約語は未だ習っておらず，未知の難しそうな概念に直面すると，初学者には大きなストレス（障壁）となり得る．また，エラーが複数個あり，どれから対応し，どのように修正すれば良いか，具体的には示してくれていない．1 番目のエラーメッセージだけであれば参考になるかもしれないが，2 番目と 3 番目のエラーメッセージはエラー原因であるタイプミスの位置（行）とは無関係であり役に立たず，むしろ有害である．中級者以上にとっては十分参考になるエラーであるかもしれないが，初めてデバグをする初学者にとっては全く不親切である．さらに，元々のエラー原因であるタイプミスは 1 箇所に過ぎないにも関わらず，3 個のエラーメッセージが出ている．従来の一般的な Java コンパイラである javac は予想外に多数のエラーメッセージを返してしまう傾向があり，どのエラーメッセージから手を付けて良いか分からず，立ち往生してしまう学生が多い．CUI のターミナル（コマンドプロンプトや Cygwin など）が小さいとエラーメッセージで溢れてしまい，一番最後のエラーから対応しようとしてしまうかもしれない．デバグに慣れた者からすると，一番上のエラーメッセージから取り組むというのが一つの決まり事のようになっていると思われるが．

このタイプミスのエラー例の場合，TA は例えば，1 行目の予約語「class」を「clas」とタイプミスしていることがエラー原因であることを把握した上で，最初はそのまま解決策を教えることはぜずに「まずは 1 番目のエラーメッセージに取り組みなさい」などとアドバイスするかもしれない．続いて「どこかにタイプミスがある」「1 行目にタイプミスがある」「1 行目の clas を何かに置き換える」，それでも解決できなければ最終的には解決策そのものである「1 行目の clas を class に置き換える」などとアドバイスすることになるであろう．

また，別のミスの例として，ある初学者は以下のように「}」で閉じ忘れるかもしれない．初学者に限らず,「{」「}」の対応ミスも非常に多く，TA でも修正方法を見付け難いエラーの一つである．加えて，インデント（字下げ）をしない初学者も多い．

HJW.java（「}」の閉じ忘れ）

```
class HJW {
public static void main(String[] args) {
System.out.println("Hello Java World");
}

_
```

このソースコードを javac でコンパイルすると，

javac HJW.java（「}」の閉じ忘れ）

```
HJW.java:4: 構文解析中にファイルの終わりに移りました
}
 ^
エラー 1 個
```

という初学者には意味不明であろうエラーメッセージが出る．TA は例えば,「5 行目に}が抜けている」と直接的にアドバイスする前に,「インデントしなさい」と言うべきかもしれない．

2.2 プログラミング演習の授業形態の問題点

1 学年約 500 名という非常に多種多様なレベルの学生が入学して来る我々の大学のコンピュータサイエンス学部では，初年次プログラミング教育として，Java 言語によるプログラミング入門を座学と演習をセットにして開講している．スライドや教科書による座学は 1 教室 120 名の学生を 1 教員で行っているが，演習時には図 1 のように 1 教室 60 名の学生を 1 教員および 4 名の院生 TA によってサポートしている．教員はいくつかの課題を与え，学生は自身のノート PC で Emacs などのエディタを用いてソースコードを書き，それを javac でコンパイルし，携帯端末（psp）を持った TA によってチェックを受ける．

現状のプログラミング演習の授業形態の問題点としては，学生数に対して十分な数の TA を配備できていないため，学生からの質問がバーストしてしまった場合に対応し切れず，適時にアドバイスを与えることができなかったり，学生へのアドバイス方法などについて教員から TA への指導が不十分であると，TA のレベルも様々であるため，学生からの質問に対して必ずしも適切で一様な回答ができなかったり（間違った回答をすることさえあったり），コンパイルエラーで立ち往生しているにも関わらず，教員や TA に質問できず分からないままにする人見知りな学生もいたりすることなどが挙げられる．

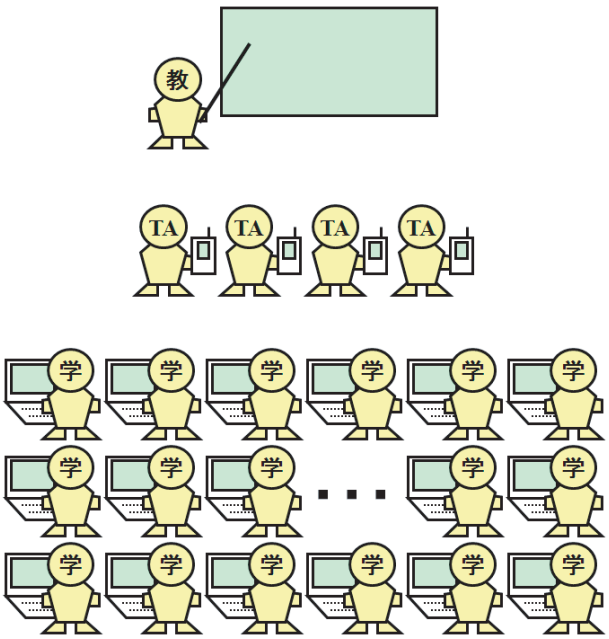

図 1　プログラミング演習の授業形態（現状）

現状の問題点を解決するアプローチの一つとして，図 2 のように TA の増員が考えられる．学生数まで TA を増員して，TA が学生をいつでもマンツーマンにサポートできる授業形態にするのが一つの理想であるようにも思われる．しかしながら，多くの TA を雇うことで人員コストが増大するというだけでなく，教員が指導する必要がある TA の数が増えるため，教員の教育方針を TA に徹底させることがより困難になる．また，人見知りな学生が存在する問題の解決策にはならないかもしれない．

以上のように TA を増員しても解決困難な問題点に対して，我々が提案する教育指向 Java コンパイラを用いれば，図 3 のように TA を増員することなく，人間の TA の代わりにコンピュータプログラムの e-TA（electronic Teaching Agent）が学生を常にサポートする授業形態を実現できる．教育指向 Java コンパイラの e-TA は，学生の質問に対するアドバイス方法を教員が人間 TA に指導するように，その内容をコンピュータプログラムが理解可能な形式で記述しておくという初期コストは小さくはないが，予め正しく記述しておきさえすれば，全てのコンピュータプログラムの e-TA は教員の教育方針に従って，学生に対して一様なサポートを提供することができる．また，人間の TA だけでなく，コンピュータの e-TA も用いることで，人見知りな学生にも対応できると考える．さらには，人間 TA を e-TA で完全に置き換えることも可能かもしれない．

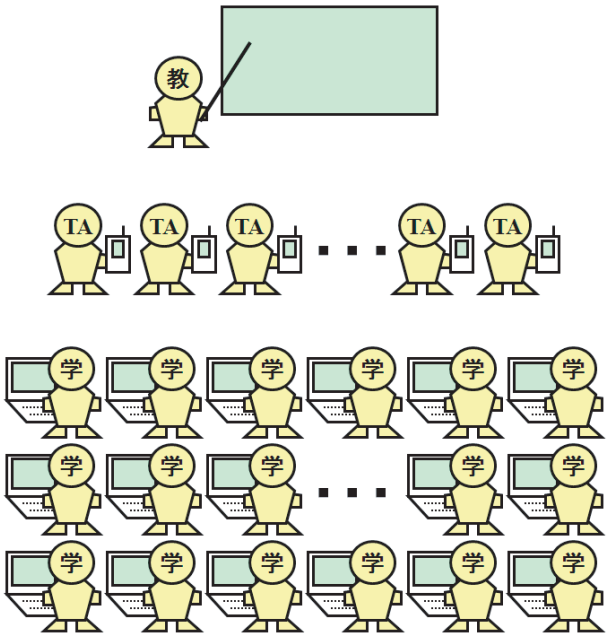

図 2　プログラミング演習の授業形態（TA の増員）

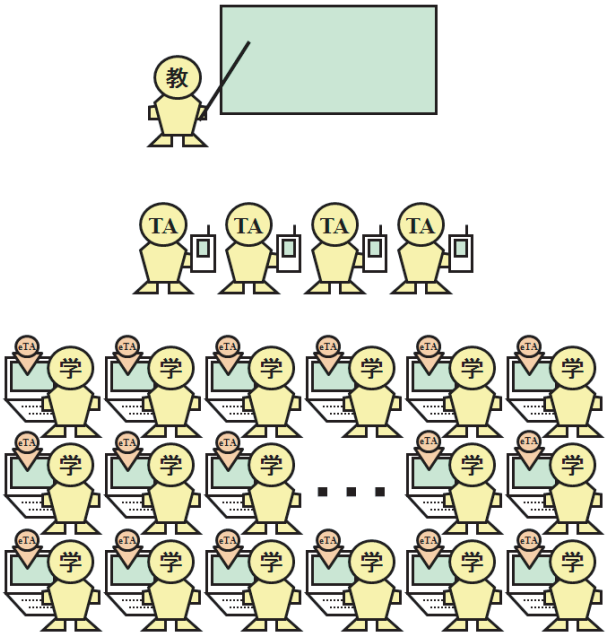

図 3　プログラミング演習の授業形態（教育指向 Java コンパイラ）

3. 教育指向コンパイラの設計

現在，我々の大学でのプログラミング演習において使用することを目標にして教育指向 Java コンパイラを試作しているが，本章では，Java 言語や，我々の大学でのプログラミング演習でのローカルな要件などに限定せず，できる限り一般論としての教育指向コンパイラの設計方針について述べる．

3.1 目　　的

従来のコンパイラの多くは，中・上級者であるソフトウェア開発者が利用することを想定し，初学者には非常に難解なエラーメッセージになりがちである．一方，教育指向コンパイラでは，これまで捨て置かれて来たプログラミング学習者，特に初学者を主な対象とする．今後は英語のように，誰もがプログラミング言語を学び，個々の要求タスクをこなすプログラムを自作するようになるような社会が訪れることを否定できないため，誰もが理解し易いエラーメッセージを返すユニバーサルなコンパイラは非常に重要であると考える．

プログラミングの初学時にストレスを掛け過ぎないように，従来のコンパイラのエラーメッセージを学生の代わりに読み解いてやるといった演習時の教員や TA と同等のサポートを人間を介さずにコンピュータプログラムが提供し，学習者にとって分かり易くコンパイルエラーを解説し，プログラミングに必要な知識やスキルを習得するための助言も合わせて行う．オブジェクトコードの生成は主な目的ではないため，従来のコンパイラに任せる．最終的な目標は，従来のコンパイラのエラーメッセージだけを参考にして，学習者が自力でソースコードを修正できるようになることであり，コンパイルエラーを取り除くことが可能な尤もらしい解（修正方法）を常にそのまま直接的にアドバイスするのが必ずしも教育的に良いわけではないため，過剰に手助けし過ぎるでもなく，適切に適時にアドバイスを呈示する必要がある．

3.2 要　　件

プログラミング演習時にコンパイルエラーに対して教員や TA が学習者にアドバイスするプロセスを考察して，教育指向コンパイラが備えるべき要件を抽出すると，次の 2 点が特に重要であると考える．

- 学習者依存：演習時に教員や TA は模範解答も参照しつつ，学習者の行動履歴（ソースコードの変遷など）やエラー傾向などを観測したり，適宜質問したりすることによって，学習者の脳内の知識形成状況を推定しながら，学習者のレベルや理解度の側面から学習者モデルを構築し，それに基づいて適切なアドバイスを生成している．従って，学習者に対しては，学習者のレベルや理解度，学習した知識（予約語など）に応じた，教員や TA と同等に気の利いたアドバイスを人間を介さずにコンピュータから常に受けることを可能にする必要がある．
- 教育者依存：演習前に教員は，学習者にどのようにアドバイスを与えるべきか，TA に指導を行う場合がある．例えば，演習時間が 90 分なので残り 30 分になったら，あるいは，15 分以上悩んで立ち往生していそうであれば積極的にアドバイスを与えるといったタイミングに関する方針や，エラーを取り除くことが可能な尤もらしい解（修正方法）をそのまま直接的にアドバイスするのか，それとも，学習者に自力で考えさせる余地をある程度残すのかといったアドバイス内容の直接性・具体性に関する方針などが考えられる．

  従って，教育者に対しては，学習者からの質問に対するアドバイスの方針を柔軟に指定可能にする必要がある．

3.3 基 本 構 成

教育指向コンパイラは，学習者からソースコードの入力を受け取ると，エラーメッセージを解説して欲しい従来のコンパイラ（javac など）でソースコードをコンパイルする．もしソースコードにコンパイルエラーが無ければ，従来のコンパイラでコンパイルしたオブジェクトコードをそのまま出力する．もしソースコードにコンパイルエラーがあれば，演習時の教員や TA によるサポートを代替する e-TA（e-Teaching Agent）にアドバイス生成を依頼する．コンピュータプログラムの e-TA は，人間の TA が個々の学生に応じて教員の教育方針に従ったアドバイスを行うように，学習者モデルと教育者モデルを有し，入力されたソースコードと従来のコンパイラのエラーメッセージを参考に，学習者依存かつ教育者依存なアドバイスを生成する．

3.4 e-TA におけるアドバイス生成

e-TA は，まず，従来の一般的なコンパイラ（javac など）のエラーメッセージ群を参考にして，コンパイルエラーを取り除くことが可能な尤もらしい解（修正方法）を求める．挿入・削除・置換・移動などのオペレーション，修正位置・範囲などのロケーション，修正対象のトークンのペアから成るモデルに基づいて修正候補を作り，尤もらしさ（修正量や修正後再コンパイルした場合のエラー数など）を評価してランキングすることで，最も尤もらしい解決策を求める．また，より知的なコンパイラ（jikes や ecj など）によるエラーメッセージや，課題毎に教員が予め指定した模範解答も参考にする．

次に，元々のエラーメッセージと最も尤もらしい修正方法との間を補間し，学習者のレベルや理解度に合った修正候補を選択して，教育者モデルに合ったタイミング・表現でアドバイスを生成する．例えば，最初のコンパイル時には元々のエラーメッセージをそのまま出力し，学習者が自力で修正することができずに再びコンパイルしてくると，最も尤もらしい具体的な解へと少しずつ近付けて行く

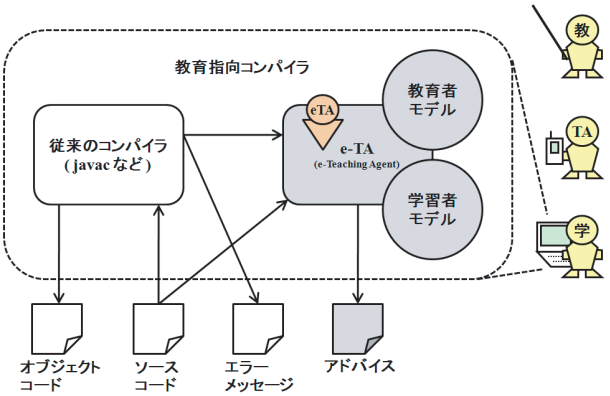

図 4　教育指向コンパイラの概要

4. 教育指向 Java コンパイラの試作

本章では，我々の大学での初年次プログラミング演習で使用するために試作中の教育指向 Java コンパイラのプロトタイプシステムである edujavac について述べる．

4.1 概　　要

我々の大学での初学者向けプログラミング演習では，Java 言語でプログラムを書けるようになることだけが教育の目的ではなく，タイピングの練習や CUI への慣れ，エラーメッセージからソースコードを修正する論理的思考なども目的であり，Emacs エディタでソースコードを作成し，コマンドラインで Java コンパイラの javac を用いてソースコードをコンパイルするというプロセスを想定している．従って，試作中の教育指向 Java コンパイラもコマンドラインで使用し，図 4 のように基本構成中の従来のコンパイラとしては javac を用いる．

一般に，ソースコードには複数のエラーが含まれ，修正箇所も複数あり得るため，複数個所を修正しないとエラーを取り除くことはできない．本稿の試作システムでは，複数のエラーが含まれていても，最も効果的な修正候補一つに絞り，各個撃破で修正して行くようなアドバイスを生成する．また，学習者のレベルや理解度の推定は行わないが，知識フォルダに入れられた電子文書からキーワードを抽出し，その出現頻度が多いキーワードは既に習得したものと見なす．一方，教育者モデルとしては，最も尤もらしい修正方法へと内容を具体化して行くタイミング設定だけを実装する．

4.2 アドバイス例

2.1 節で用いた HJW.java（タイプミス）を本稿で試作した教育指向 Java コンパイラである edujavac でコンパイルすると，図 5 のように最初は直ちに，従来の一般的な Java コンパイラである javac のエラーがそのまま出力される．

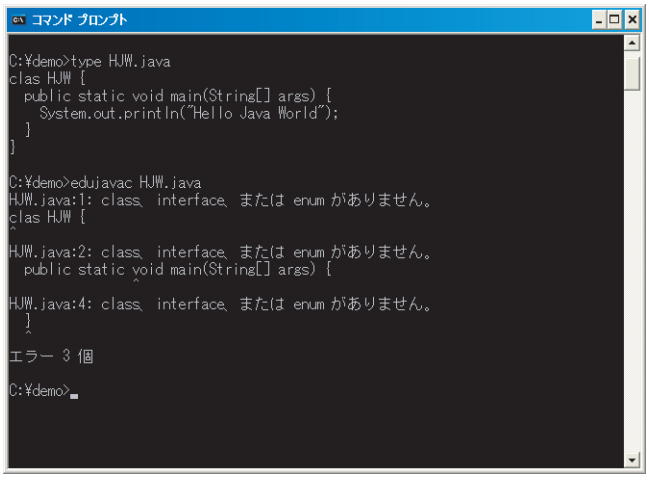

図 5　1 回目の edujavac HJW.java（タイプミス）

学習者モデルの知識フォルダに予約語「`class`」に関する電子文書しか入っておらず，教育者モデルにおいてアドバイス間隔の指定や出力時刻の絶対指定が無い場合，図 5 に続いて図 6 のように最も直接的な修正方法が出力されるまで，edujavac する度に新たなアドバイスが直ちに出力されて行く．

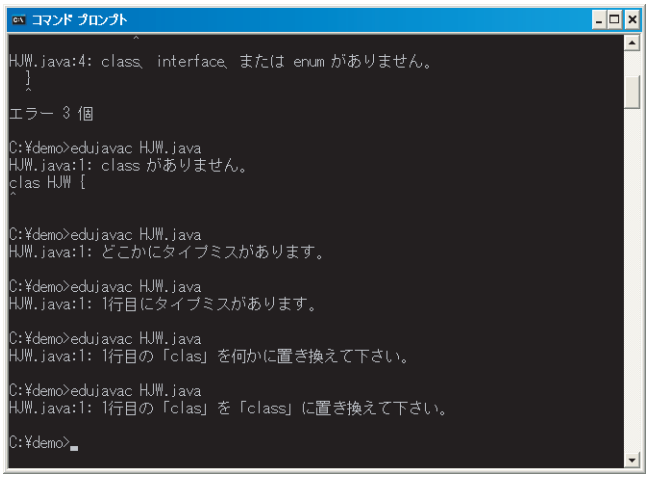

図 6　2〜6 回目の edujavac HJW.java（タイプミス）
（知識フォルダに「class」文書のみ，タイミング指定無し）

2 回目の edujavac では，javac の 1 番目のエラーメッセージが取り組むべき優先順位が最も高いと選択した上で，予約語「`interface`」「`enum`」に関する電子文書が知識フォルダに無いために未習得であると判断し，これらをマスクすることによって，難解な予約語から初学者を守っている．

実際には，edujavac する度に新たなアドバイスを呈示してしまうと学習にならないため，教育者モデルにおいてアドバイス間隔の指定やアドバイス毎に出力時刻の絶対指定が教員によって記述される．タイミング条件が指定されている場合には，これらの条件を満たすまでは edujavac しても新たなアドバイスは呈示されることはない．アドバイス後にソースコードがどのくらい修正されたかを追尾することにより，学習者の努力の度合いを評価し，それでもエラーが解決できないのであれば次のアドバイスを呈示するといったタイミングの取り方も考えられる．

知識フォルダに予約語「`class`」だけでなく「`interface`」に関する電子文書も入っていると，図 7 のように 2 回目のアドバイスにおいて予約語のマスクのされ方が変わる．さらに，予約語「`enum`」に関する電子文書も入っているか，知識推定がオフであると，2 回目のアドバイスとして javac の 1 番目のエラーメッセージがそのまま呈示される．

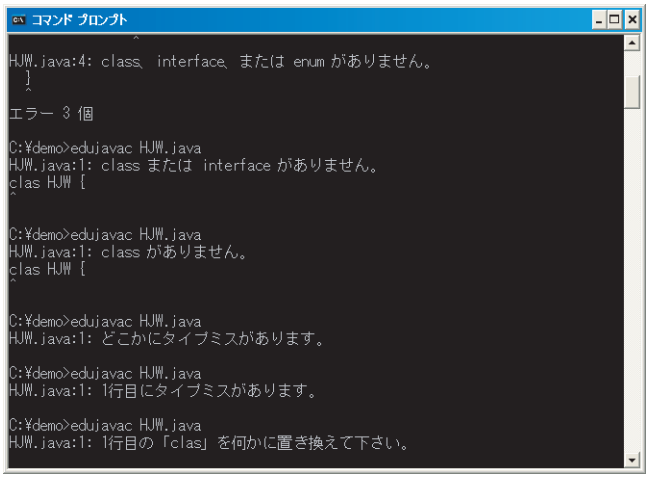

図 7　2〜6 回目の edujavac HJW.java（タイプミス）
（知識フォルダに「class」「interface」文書，タイミング無し）

同様に，2.1 節で用いた HJW.java（「}」の閉じ忘れ）を本稿で試作した教育指向 Java コンパイラである edujavac でコンパイルすると，図 8 のように最初は直ちに，従来の一般的な Java コンパイラである javac のエラーがそのまま出力される．以下の図 8 および図 9 では，教員によってタイミング条件が指定されていない場合のアドバイス系列であり，基本的には edujavac でコンパイルされる度に新たなアドバイスが呈示される．「`class`」「`interface`」「`enum`」などの予約語がエラーメッセージ中に現れていないため，知識フォルダにどんな電子文書が入っているかには依存せしない．もちろん，知識フォルダ中の電子文書に依存して，専門用語「インデント」「字下げ」「クラス定義部」などを言い換える必要はあるであろう．

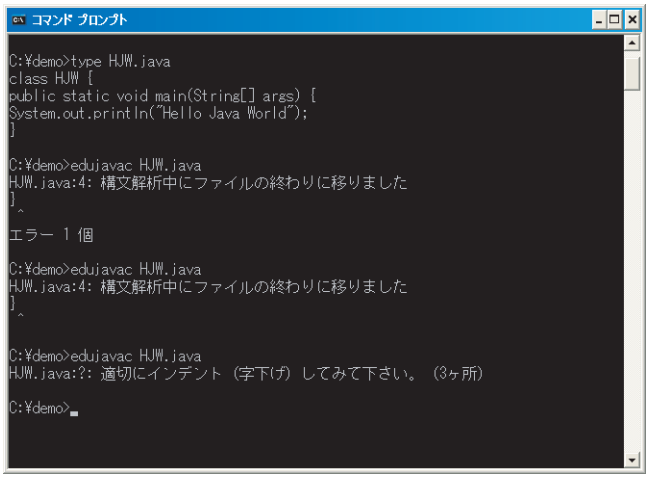

図 8　1〜3 回目の edujavac HJW.java（「}」の閉じ忘れ）
（知識フォルダの中身には依存せず，タイミング指定無し）

しかし，3 回目の edujavac で返されたアドバイスの指示に従わず，適切にインデント（字下げ）しないまま再びコンパイルした 4 回目の edujavac に対しては，図 9 のように 3 回目の edujavac で呈示されたアドバイスと全く同じものが返される．そこで，HJW.java をエディタで編集し，タブによって適切にインデント（字下げ）した後，それでも「}」の閉じ忘れミスに気付かないまま再びコンパイルすると新たなアドバイスが呈示される．最終的には「}」を挿入するように指示される．

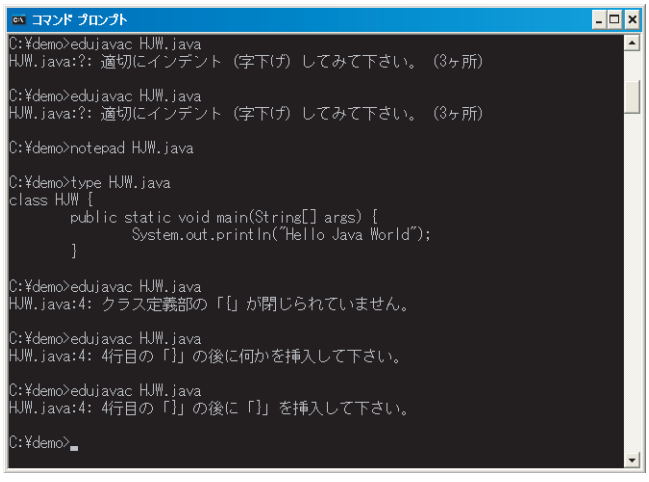

図 9　3〜7 回目の edujavac HJW.java（「}」の閉じ忘れ）
（知識フォルダの中身には依存せず，タイミング指定無し）

5.　まとめと今後の課題

本稿で我々は，ソフトウェア教育においてプログラミング嫌いを減らすため，従来のコンパイラのエラーメッセージをより分かり易く解説し，プログラミングに必要な知識やスキルを習得するための助言も合わせて行う，初学者を念頭に置いた教育指向コンパイラを提案し，その基本的な考え方，要件および基本構成について述べた．また，我々の大学でのプログラミング演習で活用するために現在試作中の教育指向 Java コンパイラの動作についても述べた．従来の一般的なコンパイラである javac の静的なエラーメッセージに対して，学習者モデルや教育者モデルに基づいて動的なアドバイスを呈示するコンピュータプログラムの e-TA によって javac をラッピングすることで実現している．教育指向 Java コンパイラの e-TA によって，人間の TA は不要になるかもしれない．また，人間の TA が e-TA のアドバイスを参考にして，学生へのアドバイス方法を研修することにも使える可能性がある．

今後は，アドバイスを呈示できるエラーメッセージの対応数を増やした上で，実際のプログラミング演習で初学者に試用してもらうなどのユーザ評価を行う予定である．また，人間の TA が行っている他の行動も e-TA が行えるように拡張する．例えば人間の TA は，教材を参照してアドバイスしたり，Web 検索して教材の範疇外のアドバイスを行ったり，対応し切れずに他の TA や教員（スーパー e-TA）にヘルプを求めたりもする．エラーメッセージ中に未習得の予約語や専門用語が現れた場合，マスクせずに「`enum` とは、･･･」「インデントとは、･･･」などと補足説明を添えたり，教材や Web ページを参照したりして，新しい概念の学習を促す方法も考えられる．教員や TA が学習者と対話することで学習者のレベルや理解度を確認するプロセスを e-TA も行えるように対話インタフェースを追加したり，学習者からのコマンド実行に対して受動的にアドバイスするだけでなく，e-TA が立ち往生している学習者に対して能動的にアドバイスするような機能も検討して行く．

謝　　辞

本研究はタンジブル・ソフトウェア教育の研究プロジェクト(注1)（研究代表者：中村太一教授，東京工科大学）の助成を受けたものである．ここに記して謝意を表す．

文　　献

[1] Eclipse, http://www.eclipse.org/.
[2] 長慎也, 甲斐宗徳, 川合晶, 日野孝昭, 前島真一, 筧捷彦: “Nigari ― Java 言語へも移行しやすい初学者向けプログラミング言語,” 情報処理学会 コンピュータと教育研究会報告, 2003-CE-71(3), pp.13–20 (2003).
[3] 岡田健, 杉浦学, 松澤芳昭, 大岩元: “教育用プログラミング言語としての「言霊」と「ことだま on Squeak」の試み,” cybermedia forum, No.7, pp.17–22 (2006).
[4] 西田知博, 原田章, 中村亮太, 宮本友介, 松浦敏雄：“初学者用プログラミング学習環境 PEN の実装と評価,” 情報処理学会論文誌, Vol.48, No.8, pp.2736–2747 (2007).

(注1)：http://www.teu.ac.jp/tangible/