プログラミング初学者のための教育指向コンパイラの提案

A Proposal of Education-oriented Compiler for Programming Novices

服部 峻♥　亀田 弘之♦

Shun HATTORI　Hiroyuki KAMEDA

今日，ソフトウェアは重要な社会基盤の一つである．極めて高いレベルでソフトウェア開発できる人材が現代社会では求められており，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の既習の知識やスキル，学習の状況も考慮することでプログラミング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コンパイラを提案する．

Software is an essential infrastructure today. So, our society requires capable human resources who can develop software 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 critical problem is that they could not avoid facing very difficult compile error messages when they were novices of programming. Another reason is that teachers and TAs (Teaching Assistants) cannot always give timely and appropriate advices to varying levels of new learners. In this paper, we propose an education-oriented compiler to present not only static error messages as-is but also easy-to-understand advices dynamically (appropriately and on a timely basis) for programming novices, and show a prototype of education-oriented Java compiler with e-TA to deal with several kinds of error messages.

♥ 正会員　東京工科大学コンピュータサイエンス学部
hattori@cs.teu.ac.jp
♦ 非会員　東京工科大学コンピュータサイエンス学部
kameda@cs.teu.ac.jp

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．このような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[*1]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edujavac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，或いは，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）等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くれる知的なコンパイラは見当たらない．ソースコード中の実際のエラー数に比べて非常に多過ぎるコンパイル時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．

[*1] http://www.teu.ac.jp/tangible/

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するといった過程を繰り返すことで，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学生を減らすため，1クラス約60名の学生に対して教員1名および院生TA 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過程が強く想定されている．そこで本稿で我々は，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コンパイラを提案し，プロトタイプシステムedujavacを試作する．

2. プログラミング教育における既存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 のプログラミング教育における問題点，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 edujavac の試作に向けて要件を抽出する．

2.1 Java コンパイラである 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，習うのは随分と後になるであり，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．

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出来なかったり，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），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．

現状の問題点を解決するアプローチの一つとして，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大幅に増大するというだけでなく，各TAのJavaプログラミングに関する知識やスキルのレベルもより多様になってしまう．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，相変わらず，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2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．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e-TAが理解可能な形式で予め正しく記述しておきさえすれば，e-TAは教員の教育方針に常に従って，学生に対して一様なアドバイスを提供できる．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．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．

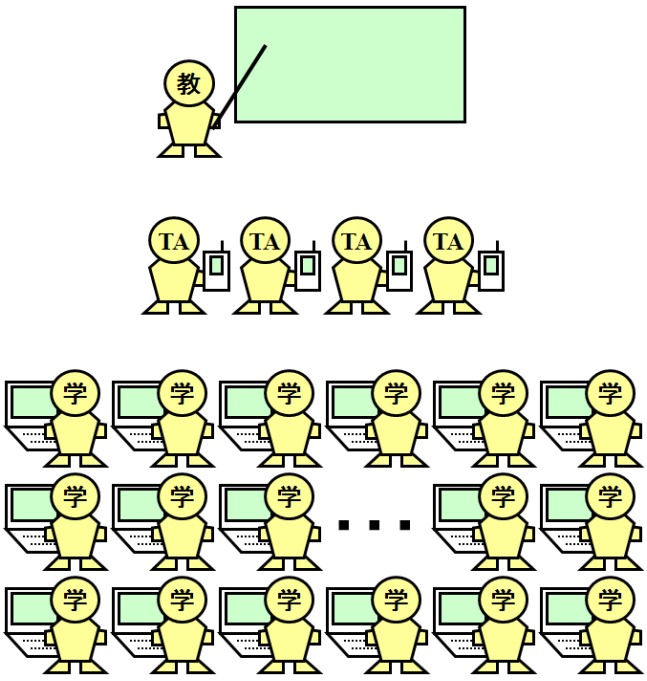

図１　プログラミング演習の授業形態（現状）

Fig.1　Programming Exercise Form (Current)

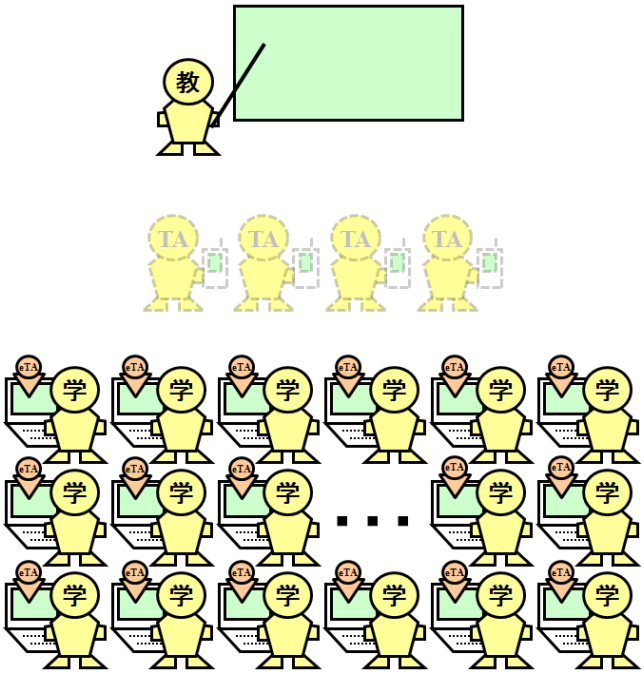

図２　プログラミング演習の授業形態（e-TA）

Fig.2　Programming Exercise Form (e-TA)

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，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（暗に）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，個々人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が否定できないため，誰もが理解し易いエラーメッセージ，及び，アドバイス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過程を考察して，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 3 点が特に重要である．前者二つはユーザ依存性にまとめられる．

1. 学習者依存（適応）性：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の生成に努めている．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）に応じて，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られるようにする必要がある．
2. 教育者依存（適応）性：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，或いは，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て欲しい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，補足情報として教科書や Web ページのどこを参照させて欲しいかなどが考えられる．従って，教育者に対しては，学習者からの質問に対する TA によるアドバイス生成の方針を柔軟に指定可能にする必要がある．
3. 能動（自律）性：教員や TA はプログラミング演習時，学習者の方から質問などの明示的なアクションが無かったとしても，学習者をよく観察することによって，能動的にアドバイスを行う場合もある．既存コンパイラの多くはユーザ（学習者）側が実行しない限り何も動作しないが，教育指向コンパイラは，ユーザが明示的にコンパイル実行しなくても，能動的に適宜，適切なアドバイスできると良い．

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（electronic-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 アドバイス生成

既存のコンパイラ（javac など）のエラーメッセージ群を参考にして，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）を評価して最も尤もらしい解決策を求める．また，他のコンパイラ（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，最も尤もらしい直接的・具体的な解へと少しずつ近付けて行く．

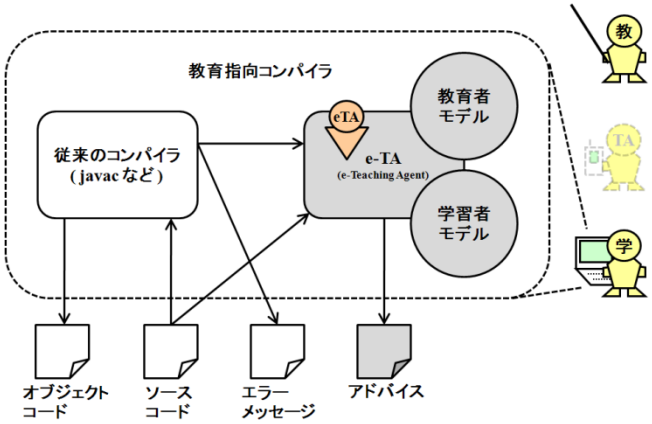

図 3　教育指向コンパイラの基本設計

Fig.3　Design of Education-oriented Compiler

4. 教育指向 Java コンパイラ edujavac 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

我々の大学での学部 1 年生向けプログラミング演習では，単に 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 edujavac もコマンドラインで使用し，図 3 のように基本構成における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，学習者モデルとして学習者毎の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を既に習得したものと見なしている．一方，教育者モデルとしては，未習得キーワードをマスクするか補足するか，補足する場合には教科書や Web ページ等の参照情報，及び，最も尤もらしい修正方法へとアドバイスをより直接的にして行くタイミング設定を実装している．

4.2 アドバイス系列の生成方法

2.1 節のエラー3 個の 1 番目からアドバイス系列を生成する場合，パターン「expected」からオペレーションを定め，キーワードやロケーションも抜き出すことで以下の 6 種類の修正案が収集できる．各々の修正案を適用後に再コンパイルした場合のエラー数に基づいて，修正案を一つ選択する．

(insert, 1, 1, “class”)	3→1 error
(insert, 1, 1, “interface”)	3→1 error
(insert, 1, 1, “enum”)	3→2 errors
(replace, 1, 1, “class”)	3→0 error
(replace, 1, 1, “interface”)	3→2 errors
(replace, 1, 1, “enum”)	3→4 errors

次に，javac エラーと修正案との間をルールベースで補間する．修正案という形式表現から自然言語表現へ変換し，置換前「clas」と置換後「class」とのエディット距離が小さいことからエラー原因が「タイプミス」と判定される．

4.3 動作例

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．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未習得なキーワードに対してマスク処理が選択され，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，最も直接的な修正方法が出力されるまで，図 5 のように edujavac する度に新たなアドバイスが直ちに出力されて行く．

2 回目の edujavac では，javac のそのままのエラー3 個に対して，予約語「interface」「enum」に関する電子文書が学習者の知識フォルダに無いために未習得であると判断し，これらのキーワードをマスク処理（隠蔽）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．3 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として最も高いことを学習者に教えている．その後，4 回目，5 回目，6 回目の edujavac では，「置換」といった直接的な修正方法ではなく，コンパイルエラーが出てしまっている原因が「タイプミス」であることだけを学習者に伝え，学習者自身で修正方法を考えさせようと促している．

実際には，edujavac する度に直ち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．

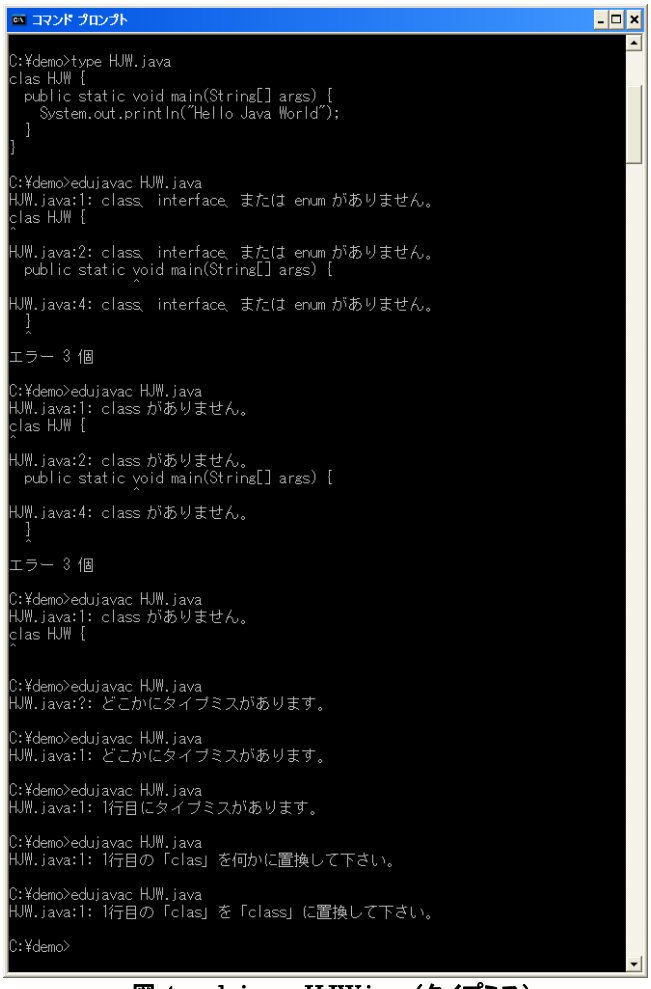

図 4　edujavac HJW.java（タイプミス）
（マスク，know={“class”}，タイミング指定無し）

Fig.4　edujavac HJW.java (with a typo)
(mask, know={“class”}, timing-opt: null)

知識フォルダに予約語「class」「interface」に関する電子文書が入っており，教育者が補足処理を選択している場合，図 5 のように 2 回目，3 回目の edujavac において予約語「enum」がマスクされるのではなく，補足が追加される．

また，図 6 は能動的な edujavac の例で，プロンプト「you>>」に一定時間入力が無いと e-TA が自律的にアドバイスを行う．

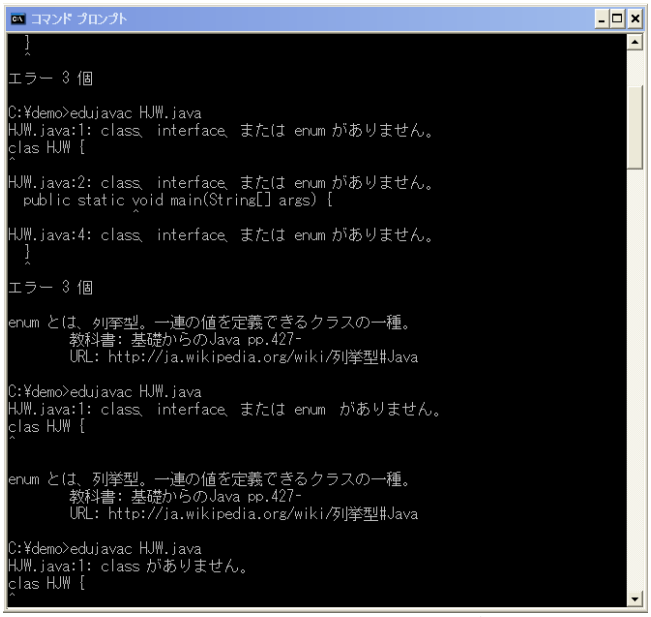

図 5　edujavac HJW.java（タイプミス）

（補足，know={“class”, “interface”}，タイミング指定無し）

Fig.5　edujavac HJW.java (with a typo)

(supplement, know={“class”, “interface”}, timing: null)

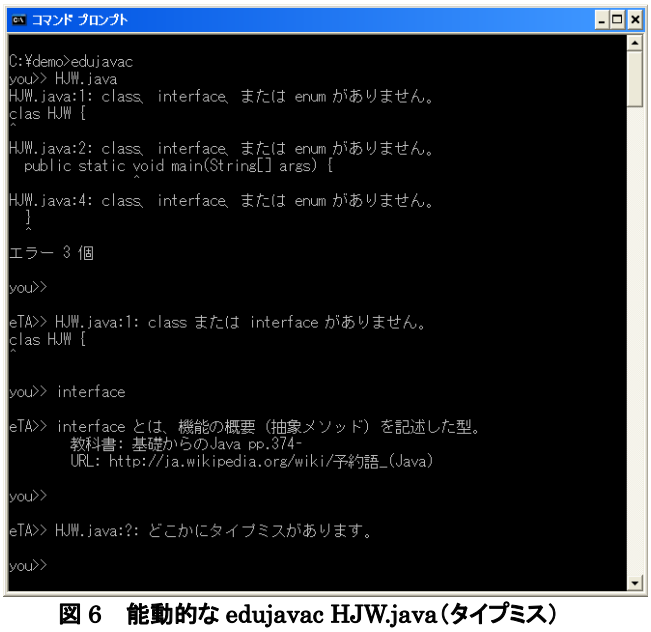

図 6　能動的な edujavac HJW.java（タイプミス）

（mask，know={“class”, “interface”}, proactive）

Fig.6　edujavac HJW.java (with a typo)

(mask, know={“class”, “interface”}, proactive)

5. まとめと今後の課題

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直接的に説明し，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，我々の大学での学部 1 年生向け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 edujavac の動作例についても述べた．従来の一般的な Java 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．人間の TA は不要になるかもしれないし，或いは，人間の TA が e-TA のアドバイスを参考に学生へのアドバイスの仕方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．

[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（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/ (2010).

[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).

服部 峻　Shun HATTORI

東京工科大学コンピュータサイエンス学部助手．2007 日本学術振興会特別研究員 DC2．2009 京都大学大学院情報学研究科社会情報学専攻博士後期課程修了．2009 埼玉大学地圏科学研究センター非常勤研究員．京都大学博士（情報学）．ユビキタス社会基盤としての情報アクセス技術研究に従事．情報処理学会，電子情報通信学会，日本データベース学会，画像情報学フォーラム各会員．

亀田 弘之　Hiroyuki KAMEDA

東京工科大学コンピュータサイエンス学部教授．1987 東京大学大学院博士課程単位取得満期退学．1987 東京工科大学工学部専任講師．1991 東京工科大工学部情報工学科助教授．工学博士（東京大学）．自然言語処理，知的情報検索など，主に思考と言語に関する研究に従事．タンジブル・ソフトウェア教育の研究，認知リハビリテーションの研究にも従事．電子情報通信学会，ACM，人工知能学会，日本認知科学会，ゲーム学会，実践的ソフトウェア教育コンソーシアム等会員．