# Javaプログラミングの課題のオンライン自動採点システム

北谷 宏紀† 井上 潮‡

†‡東京電機大学大学院 工学研究科 〒120-8551 東京都足立区千住旭町５番

E-mail: †13kmc09@ms.dendai.ac.jp, ‡inoue@c.dendai.ac.jp

**あらまし** 大学でのプログラミング授業において、演習課題を採点する作業は人手で行っているのが現状である。多くのプログラムを手作業で採点するのは採点者にとって非常に負担となり、採点ミスも生じる。全ての採点が終わるまで学生には結果が分からず、内容を忘れてしまう学生もいるだろう。その為、本研究では課題提出と採点をウェブ上で行い、学生に即座にフィードバックを返すシステムを構築する。予め課題の模範解答を作っておき、サーブレットと JSP を用いて作成したウェブサイト上で課題を提出させる。その後コンパイルチェックおよびプログラムの実行が行い、実行結果を模範解答のものと比較することにより採点する。本システムにより、非常に高い精度での自動採点が可能になり、かつ採点時間は手動と比べ 1/7 程に短縮出来た。

**キーワード** プログラム自動評価, 教育学習支援システム

## 1. はじめに

近年、多くの大学でプログラミング入門授業が行われている。そのようなプログラミング入門授業において、学生に学習内容を復習させるために演習課題が出されることが多い。しかし多くの学生が演習課題で作成したプログラムを提出する一方、それらを採点し、フィードバックを返却する作業は人手で行うのが一般的である。一つひとつのプログラムを手作業で採点するのは採点する者にとって非常に負担となり、採点ミスも生じうる。また、全ての採点が完了するまで学生には結果が分からないため、学生が課題の内容を忘れてしまう場合もある。

そこで本研究では課題の提出をウェブ上で行い、リアルタイムで自動採点し学生にフィードバックを返すシステムを提案する。本研究における課題としては、①ウェブ上でどうやって採点業務を行うのか、②フィードバック返却を自動でどのように行うのか、という点である。

課題①に対する解決策として本研究では、サーブレットと JSP を用いる。それらを用いて作成したウェブサイト上で課題を提出させ、コンパイルチェックおよびプログラムの実行・テストを行い、実行結果を予め用意した模範解答のものと比較することにより採点する。サーブレットを用いることによりサーバ上で Java プログラムを実行でき、この中で自動採点が行えるようになる。また、自動化することで出題者側が採点を行う負担をなくすことができ、採点時間を約 1/7 に削減出来るという結果を得られた。

次に課題②に対する解決策として、コンパイルチェックが失敗した場合には､そのエラーを学生に提示し、テストや実行結果の比較を行う部分での誤りについてはどの結果が間違えたかをメッセージで表示し、どこで間違えたのかを学生に分かりやすくする。これらのフィードバックを即座に返すことで学生のプログラミングに対する理解を深めることが出来る。

本論文では初めに関連研究について紹介し、次に対象となる授業課題についての説明をする。その後、本システムに用いるツールについて説明したあと、採点をどのように行っていくかのアプローチを解説する。更にシステム全体についての構成及び採点部分の実装方法について、実際のウェブページの図を交えながら説明し、システムを評価した結果を示す。最後に、本システムについてのまとめと今後の課題を述べる。

## 2. 関連研究

関連研究として、オブジェクト指向プログラミング教育における採点支援システムの開発[1]がある。この研究では期末試験において学生にクラス図と API 仕様を参照させながらスクラッチからプログラムを作成させ、完成したプログラムを一度に採点するシステムである。そのシステムを実際に試験の採点で導入し、結果として 1400 個のプログラムファイルを 2 分以内に処理することができ、採点業務にかかる時間を約 3 分の 1 に短縮できたとある。

また、和田らのプログラミング演習課題の評価支援システム[2]では、コンパイルチェック・実行結果の正規表現による採点・ソースコードに指定した単語を含むかどうかを判断し、採点を行う。盗用検出も同時に行い、その実行結果は 100 ファイルに対して 176 秒で行え、大幅に採点時間を削減できたとある。

また、Python での課題に対する自動フィードバック生成[3]では課題のリファレンス実装と学生の出すエ

ラーに対する訂正で構成されるエラーモデルを用い、誤解答であった学生のプログラムの 64%に正しくフィードバックを返せたとある。

他にも、実行テストと e-learning を組み合わせたものや、プログラムの意味的等価性により評価を行う研究、アセンブラの自動評価など、プログラミング採点の自動化に関して様々な研究がある[4][5][6][7]。

本研究では隔週で出される課題に対する採点であり学生にフィードバックを返す目的もあるため、即時性を重視してウェブ上で個別に採点を行うようにする。また、入門授業の初期段階における簡単なプログラムを採点する必要もあるため、実行画面の比較をするという採点手法も取り入れ、学生に分かりやすくなるよう工夫する。また、全ての誤解答に対しフィードバックを返すことが出来るようにする。

## 3. 前提条件

### 3.1. 本研究で対象とする授業

本研究では、東京電機大学の工学部情報通信工学科 2 年次で行われている Java 言語入門の授業内で出題される課題を採点対象とする。

課題の内容は詳しくは後述するが、ユーザからの入力に対して処理をして結果を返すものから、指定された仕様のクラスを作成するものまでの、基礎レベルの課題である。

### 3.2. 本システムの要求条件

システムの要求条件として、授業の初期に出題されるクラス 1 つ・メインメソッド 1 つといった一般のテストケースによるテストが行えないような極めて単純な構成のプログラムを採点できる必要がある。もちろん、授業の後半ではメソッドやクラスを作成させる課題があるため、それらをチェックする必要もある。また、コンソールからの入力が必要な課題に対しても、標準入力をどのようにウェブ上で実現するかを考える必要がある。

## 4. 採点システムの検討

### 4.1. 採点方法

本研究では前述の要求条件を満たすため、採点方法として、コンパイルチェック及び実行結果の比較・リフレクションを用いた JUnit テストケースによるテストを行い、3 つの観点からの採点をする。そしてそれらをウェブ上で行うため、同じ Java 言語を使うサーブレットを用いた授業用のサイトを作り、サーブレット内（サーバ上）で採点プログラムを実行させる。

課題の採点を行う上で、コンパイルのチェックは必須であるため、これはどのような課題に対しても行う必要がある。実行結果のチェックは学生のプログラムを実際に実行させた結果を模範解答の実行結果と比較し、採点をする。この手法により、メソッドやクラスをテスト出来ないような課題に対しても採点を行うことが出来るようになる。この二つの採点手法に加え、授業の課題がメソッドやクラスなどを作成するような課題の場合、JUnit テストケースを用いたテストを行い、メソッド・クラスの振る舞いをチェックする。但し、この手法を用いるのはあくまでメソッドやクラスを作らせるような課題の場合のみで、入門授業の始めの課題に多い、1 つのメインメソッドのみで完結するような課題の場合には用いることは出来ない。

また、実行時にユーザからの入力が必要なプログラムに関しては、入力情報を格納したファイルをリダイレクトして読み込ませ実行させることで解決することが出来る。

### 4.2. 採点の流れ

学生がウェブ上に課題ファイルを提出した後、「採点をする」という旨のボタンを押すと、採点が行われる。その流れを図 1 に示す。

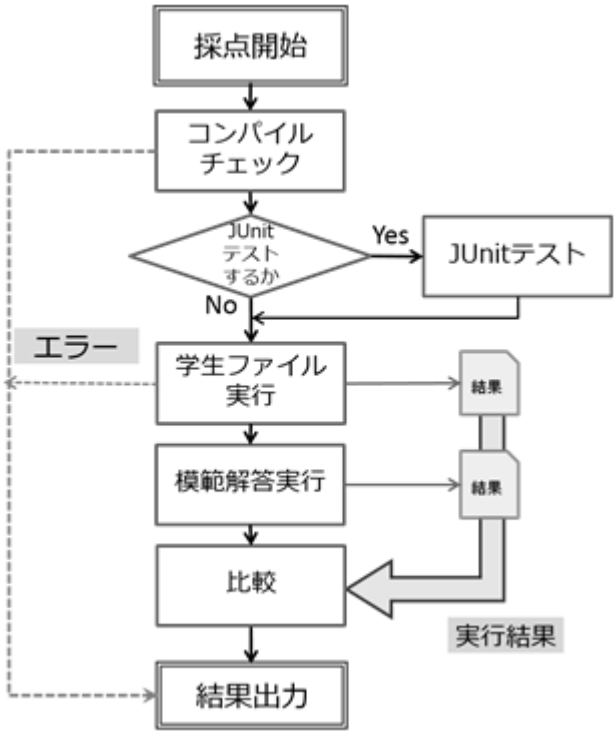


図 1. 採点の流れ

図 1 のように、まずコンパイルを行い、そこでコンパイルエラーが出なければテストケースによるテストを行うかどうか判断する。JUnit はメソッドやクラスの振る舞いをテストするツールであるため、Java 入門授業でも初期レベルである、複数のメソッド・クラスを使わないような課題の場合はこの工程を飛ばして次のファイル実行へと移る。JUnit を用いたテストを行う場

合は予め課題ごとに作成しておいたテストを行い、その結果を保存した後にファイルの実行へと移る。ファイル実行ではまず学生プログラムを実行し、結果をテキストファイルで保存する。そこでエラーが出なければ模範解答のファイルを実行し、学生の実行結果と比較することで採点を行うこととした。比較方法に関しては **5.4.3** で詳しく述べる。

コンパイルエラーや学生のファイルの実行時にエラーが出た場合はその時点で結果の出力を行う画面へ遷移する。

## 5. システム概要

### 5.1. システムの実装手段

本研究では学生にすぐにフィードバックを返却するため、ウェブ上での採点という手法を採った。この採点サイトは、Apache Tomcat 上で Java Servlet と JSP を用いて作成した。Apache Tomcat は、Apache Software Foundation で開発されているサーブレットコンテナの一つであり、サーブレットコンテナはウェブサーバ上で Java プログラム(Java Servlet)を動かすためのサーバである。サーブレットや JSP と同じように、サーバサイドでプログラムを実行する CGI があるが、これはアクセスがあるたびにサーバが新しいプロセスを起動し、パフォーマンスが低下する原因となる。一方サーブレットや JSP を用いることで一度プログラムが起動すれば一つの共通インスタンスが生成され、当該のプログラムに対する HTTP リクエストはそのインスタンスのスレッドとしてリダイレクトされる。そのため、プロセス起動や終了処理を減らすことができ、リソース削減にもつながり複数の学生がアクセスしたときのパフォーマンスを保つことが出来る。

また、採点結果や学生の情報などを管理するために MySQL データベースを使用する。

### 5.2. システム構成

システムの概略を図 2 に表す。

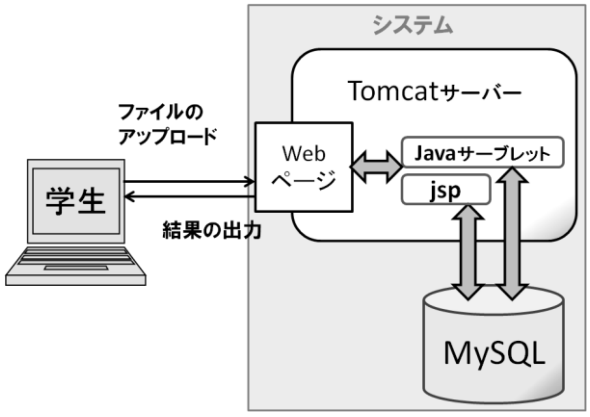


図 2. システム概略

ウェブページの生成において処理が簡単なところは JSP を使い、コンパイルチェックを行ったり、JUnit によるテストを行ったり、プログラムを実行したりなどの処理を行う部分では Java Servlet を用いる。

システムの内容としては、教員が指示した内容の課題を学生が Java ファイルで提出し、サーバ上にファイルをアップロードする。アップロード後に自分の提出したファイルの内容が表示されるので、それを確認後、採点する旨のボタンを押す事で採点が開始され(**4.2** 参照)、結果が MySQL に保存される。採点の為のプログラムは、現段階では教員が正規表現やリフレクションなどを使って自ら作る必要がある。

採点以外に出来ることとしては自分が提出した課題の採点結果の確認、提出したファイルの削除(今後も残すかは未検討)のほか、学生が結果を確認した後再提出も出来るようになっている。

### 5.3. ファイルのアップロード

課題ファイルのアップロード画面を図 3 に示す。


図 3. 課題アップロード画面

Java Servlet で用意されているマルチバイトデータを扱うことの出来るアノテーションを用いることで、課題ファイルのアップロードをすることが出来るようになる。Java Servlet 2.5 より取り入れられたこの機能により、リクエストから送信されたファイルを特別なライブラリなどを使わずに扱うことが出来る。

課題の一覧が上図のように表示され、リストの中からそれぞれの課題に対する操作を行えるようになっている。学生が一覧の中から提出する課題を選択し、アップロードのボタン( )を押すことでアップロードを行う。また、アップロード済みのものに関しては、ファイル名の後に提出した時間が表示される。

アップロードボタンの隣はそれぞれ採点結果の確認ボタン、提出ファイルの削除ボタンである。本研究では、学生のプログラムに対する理解を深めるため、正しい課題を提出できるまでは何度も修正・提出・テストを繰り返させるようなスタイルを想定している。

そのため、提出済みのものに関してもアップロードボタンから再度ファイルを上げ、再提出を行うことが出来る仕様となっている。

次に、アップロード後の画面を図 4 に示す。図 4 では Exercise1 のファイルを提出した場合の画面である。


図 4. 提出画面

図 4 のように、提出すると学生が課題で作成した java ファイルの内容を確認出来るようになっている。確認後、コードの右上に表示されている「→採点する」というボタンを押すことにより Servlet へリクエストが送られ、採点を開始する。

## 5.4. 課題の採点

課題ファイルの採点では図 1 の流れに則って行う。採点を行う上で必要なコンパイル・クラスのテスト・ファイル実行を行うためのディレクトリは以下、図 5 のような階層とした。

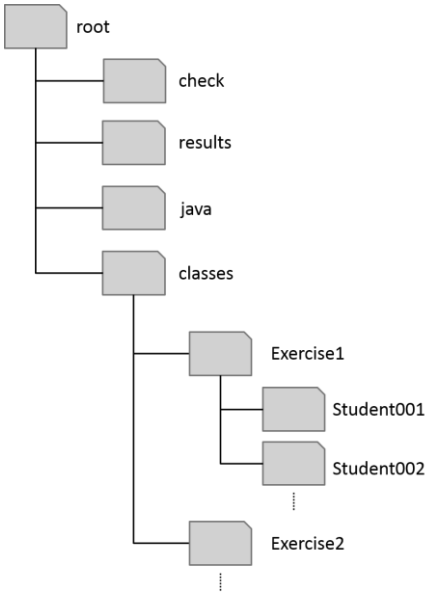


図 5. 採点作業のディレクトリ階層

まず、作業ディレクトリの root 内には 4 つのディレクトリが存在している。図中には記述していないが java, classes, results の 3 つのディレクトリ下は全て同じ構造(classes 以下と同じ)になっている。Java ソースファイル、コンパイル後の class ファイル、実行結果を保存したテキストファイルを学生ごとにフォルダ分けし、互いに干渉することがないようにしている。check フォルダ内にはコンパイルやテストを行う再に必要なファイルが含まれている。

### 5.4.1. コンパイルチェック

初めに、学生がアップロードしたファイルは java フォルダ内に格納される。次に、コンパイルが行われるとそのクラスファイルを classes フォルダに保存する。課題が、学生が提出したファイルに含まれるクラス以外のクラスを参照する場合、必要なクラスファイルを予め check¥ExerciseXX¥lib というフォルダ内に入れておき、クラスパスとして含めておく。コンパイルには javax.tools.JavaCompiler クラスを使用する。Java Compiler API は動的にコンパイルを行えるもので、プログラムの実行中に別の javaファイルをコンパイルするために有効である。また、コンパイル時の様々なオプションも細かく指定することが出来、クラスパスの指定などもこのツールにより行う。

コンパイルチェックの時点でエラーが起きるとそれ以降の処理を中止し、results フォルダにエラーの情報を保存し結果表示へと移る。

### 5.4.2. テストケースによるテスト

コンパイルチェック後、テストケースによるテストを行う課題に関してはテストを行い、結果を保存する。ここでのテストでは一般的な JUnit によるテストを更に発展させ、java.lang.reflect を用いてリフレクションが使えるようにした。リフレクションとはクラスを動的に扱うことが出来るもので、クラスやメソッドのアクセス修飾子の判断を行うことができ、private なクラス・メソッド・フィールドも扱うことが出来る。クラスの構成要素を動的に扱うことが出来るため、課題で指示されたメソッドやフィールドの存在の有無を判断したり、仕様通りに作られているかといったチェックを private・public 関係なく出来るため、幅広いテストが行えるようになる。これにより、授業後半で出題されるような課題に対して採点精度を高めることが出来る。

また、この時点でエラーが出たとしても、次の実行結果の比較による採点も行う。

### 5.4.3. 実行結果の比較

最後に、コンパイルしたクラスを実行させ、その結果を results フォルダ内にテキストで保存する。プログラム実行の際にはサブプロセスを作り、コマンドプロンプトを呼び出すことでファイルの実行を行う。

コマンドプロンプトを用いることで、**4.1** で述べたようにコンソールからの入力が必要な課題においてコマンドプロンプトのリダイレクトの機能を使い、テキストファイルに格納された値を渡すことが可能となる。

ファイル実行時にエラーが起きなければ同じようにして模範解答を実行し、テキストで保存し、保存した実行結果同士を比較することにより採点を行う。コンソールから入力した値に対する計算結果が模範解答と等しければ正解というように採点を行うが、特定の入力値の場合のみ特定の数値を表示するだけというプログラムを作成する学生もいるかもしれない。そういったプログラムへの対策のため、実行結果の比較は複数の入力値を用いて検証を行う(学生には入力する値のうち一部しか見えないようにする)。ただし、乱数を用いて何かを表示させる課題の場合、結果の比較は行わずに結果の値を読み取り、直接判定することもある。しかし、模範解答との比較を行わないだけで、基本的な動作は変わらない。

### 5.5. 採点結果の表示

これらの手順でテストを行った結果､以下の図 6 のような画面が表示される。この例ではメインメソッド一つで完結するような課題のため、テストケースによるテストは行っていない。そのため、コンパイルチェック及び実行結果の表示のみとなっている。

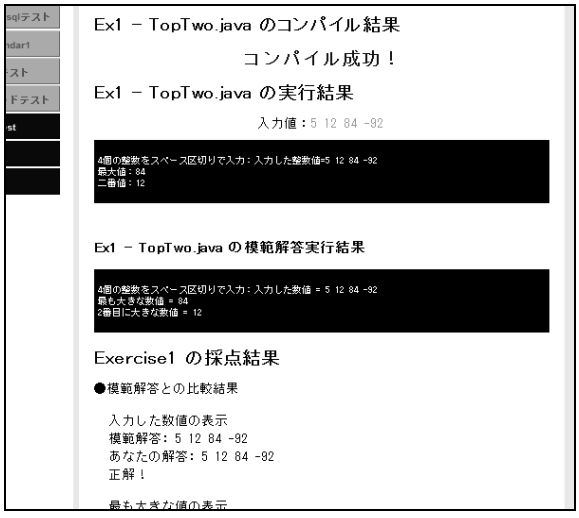


図 6. 採点結果画面

図 6 のように､初めにコンパイルチェックの結果を表示する。もしここでエラーがあれば「実行結果」以下の部分はなくなり、コンパイルエラーの詳細が出力される。コンパイルが通ればその下に実行結果の一部を表示し､模範解答の実行結果も表示する｡これにより、学生が画面を確認した際にどこで間違っているか一目で分かるようにすることが出来る。その後、採点結果についての詳細を表示している。

## 6. システム評価

本手法を検証するために 2010 年度に実際に提出された課題を用いてテストを行った。その後、本システムの採点部分のみを応用して本年度の授業課題に対して資料を集めつつ、採点システムを適用してシステムの改善を行った。

### 6.1. コンパイルチェック

Java Compiler API によりコンパイル及びそのエラー表示はコマンドプロンプトで行う時と同じ品質で行えた。ただし、文字コードを Shift-JIS にて採点していたため、UTF-8 で書かれたプログラムなどをコンパイルした時に警告が出された。採点するときには文字コードを選択できるようにするなどの改善点が必要となった。また、Java 初心者にとってはコンパイルエラーの意味が分からない場合も多いため、その点に関しては今後日本語で分かりやすくエラーを説明するなどの工夫が必要である。

### 6.2. 採点精度

#### 6.2.1. 過去問題の採点

まず、プロトタイプとして 3 種類の課題に対して採点を行うシステムを **5.**で述べたように製作した。採点を行うためのプログラムは採点者が自ら作成する必要があるため、まずは自分で予め起こりうるケースを予想し採点用プログラムを作成した。採点に使用した課題の内容は､課題 1:入力した 4 つの値(int 型)から最大･二番値を出力するもの、課題 2:与えられた仕様に則ってビンゴカードを表示するもの、課題 3:三角形を表す Triangle クラスを作り、toString メソッド等の必要なメソッドを作る、というものである。

これらの課題を過去に実際に提出されたものを使って採点し､手動で採点した時と比べ[正しく採点出来た/採点ミスがあった]の二種類に評価した。その結果を表 1 に示す。

表 1. 採点精度評価(値：件数)

| | 課題 1 | 課題 2 | 課題 3 |
|---|---|---|---|
| 正しく採点 | 101 | 119 | 88 |
| 採点ミス | 18 | 0 | 4 |
| 合計 | 119 | 119 | 92 |
| 採点精度 | 85% | 100% | 96% |

表 1 のように、課題 2, 3 は高い精度で採点を行えたが、課題 1 での採点はあまり精度が高くなかった。この原因は課題内容として、出力部分を自分で作成するものだったため正規表現での抽出が上手くいかなかったためである。また、結果を一行ずつ比較し採点したため、模範解答では二行であるものが一行にまとめ

られていると比較する時点で失敗してしまった事なども挙げられる。その他、それぞれの課題における採点ミスの原因、およびその件数を表 2 に示す。

表 2. 採点ミスの原因(値：件数)

| 原因 | 行 | 誤字 | 表記 | 数値 | コード |
|---|---|---|---|---|---|
| 課題 1 | 8 | 2 | 7 | 1 | 1 |
| 課題 2 | 0 | 0 | 0 | 0 | 0 |
| 課題 3 | 0 | 0 | 0 | 4 | 0 |
| 合計 | 8 | 2 | 7 | 5 | 1 |

(表の原因の説明　行:行数の変化による抽出ミス、誤字:学生の誤字による抽出ミス、表記:学生が出力を英語にするなどの原因での抽出ミス､数値:int 型を double にしたり端数を丸めたりしたなどの比較ミス、コード:文字エンコードの違いによる抽出ミス)

表 2 のように、採点方法として実行結果の比較による部分は改善すべき点が残った。しかし、正規表現の改善、出力値を文字列で比較していた部分を数値に直してから比較するように修正、文字コードが違った場合は他の文字コードによってコンパイルを複数回行うなどして採点プログラムを修正すれば、これらの採点ミスはなくなった。つまり、実際には正規表現による結果の比較プログラムさえ上手くいけばほぼ全てのプログラムを正しく採点できることが分かった。

### 6.2.2. 本年度授業における課題採点

また、プロトタイプとして作成した採点システムの採点部分を応用し、学生が提出したファイルを一度に全て採点するシステム(ローカルで採点を行うシステム)を作り、2013 年度の授業における課題採点に用いた。もちろん採点システムとして未完成であるため、まずは手動で採点を行った後に採点用プログラムを作成し、採点を行った。ここでもやはり実行結果の比較による採点で採点ミスが生じたが、採点プログラムを修正することによりそれをなくすことが出来た。但し修正するという手間を省くのであれば、出題する課題の仕様を厳密にするなど、課題に対する多少の制限が必要になると思われる。

また､2013 年度でのメソッドやクラスを作成する課題ではテストケースによるテストにより、コンストラクタにアクセス修飾子が付いていない・指定したアクセス修飾子が付けられていないといった、実行結果の比較だけでは分からない誤りを検出することが出来た。そのため、リフレクションとテストケースを組み合わせたテストにも有用性があると思われる。

## 6.3. 採点時間

本システムの採点時間はプロトタイプとして作成したウェブサイト上で採点するとき、サーバ側でリクエストを受け取る doPost メソッドが呼ばれてから終了するまでの時間とした。ここで Tomcat の性質として、初めてアクセスされたサーブレットにはその初回アクセス時にインスタンスが一つ生成されるという処理がある。そのため、初回アクセス時にはやや時間がかかるという問題がある。そのため、プロトタイプにおいてそれぞれ(3 種類)の課題に対し、初回 Tomcat 起動時及び 2 回目以降のリクエスト送信を 10 回ずつ行い､一つの課題に対して計 20 回、すべて合わせて 60 回の計測を行った。その平均値をとった結果が以下の表 3 である。

表 3. 採点時間(単位：秒)

| | 課題 1 | 課題 2 | 課題 3 |
|---|---|---|---|
| 初回アクセス | 1.00 | 0.95 | 1.03 |
| 2 回以降 | 0.60 | 0.57 | 0.63 |

実験に用いたマシンの仕様：
OS: Windows7 Home Premium SP1 32bit
CPU: Intel Core i5 CPU M460 (2.53GHz)
RAM: 4.0GB

表 3 より、どの課題でも初回アクセス時ですら 1 秒程度、2 回目のアクセス以降は 0.6 秒前後と体感的には十分速いと言える結果だった。そのため、学生に即時フィードバックを返すという点においては十分効力を発揮するものと思われる。

次に、本システムで採点した場合と手動で採点した場合の全体の課題に対する採点時間の比較を行うため、コンパイル・テストケースによるテスト・実行結果の比較をして点数を決定するまでの採点業務をそれぞれ計測した。この計測に関しては本年度の授業課題も用いて行った。その結果、手動による採点で 30 分～40 分程度かかっていた業務が､3 分～5 分で終わるという結果となった。そのため、学生が課題を提出してからしか出来ない手動採点と比べ、採点時間を大幅に減らすことが出来た。しかし、採点プログラムを作成するのに､テストケースを行わない課題でも 30 分以上､テストケースを行う課題に対しては 1 時間以上かかったこともあったため、採点プログラムを作成するのが非常に負担となるということが新たな問題として浮上した。ただし、採点プログラムを予め用意しておくことが出来るため、学生の課題提出を待たずとも作業を行えるという点に関しては融通が利くシステムだと思われる。

## 7. まとめと今後の課題

プログラミング課題をウェブ上で提出･採点できるシステムを提案し、構築した。ウェブ上でプログラムの採点や実行を行うため､サーバ上で Java を動かすた

めの Java Servlet を主に用いてシステムを実装した。採点手法としてコンパイルチェック・テストケースによるテスト・実行結果の比較を用いた。テストケースによるテストではリフレクションを用いることで private なクラス・メソッドも採点することが可能となった。また、実行結果の比較による採点をすることでメソッドやクラスをテスト出来ないようなプログラムにも対処することが出来た。採点時間に関しては、学生が一つの課題を提出し、採点するリクエストを送り、結果が表示されるまでは 1 秒以下であり、即時フィードバックを返すことが出来た。プログラムが学生の課題全部を採点する実行時間も手作業と比べれば非常に短い時間に収まり、採点者の負担を軽減できた。また、実行結果の比較部分では模範解答の結果と自分の結果を目視で比較できるため、学生が理解しやすくなっている。

なお、実行結果の比較による採点を用いると採点精度が悪くなってしまうこともあるため、今後の改善点である。しかし、その精度も課題の内容にもよるところが大きいため、単純に精度を上げるなら課題の内容を厳密にするなどの対策を採れば解決出来る問題だと思われる。また、採点者の負担を更に減らすためには、今後、自動採点を行うための採点プログラムを容易に作成できるようにする必要がある。

そのため、今後は、ウェブ上でフォームを用い、簡単にテストを作成できるようなシステムなどを作成することを考えている。また、コンパイルエラーなどもプログラミング初心者に分かりやすい日本語に置き換えて表示するなどの改善を行い、更に分かりやすく表示することで学生の理解度を深めていくことを目指す。

その他、学生のプログラムによるバグに対して強いシステムの検討などをしていき、実用段階まで完成させることが目標である。

## 参 考 文 献


[1] 高野辰之，宮川治，小濱隆司，“オブジェクト指向プログラミング教育における採点支援システムの開発とその評価”，社団法人 情報処理学会研究報告， no.2008-CE-96(7)， pp.41-45,Oct.2008

[2] 和田修平，井上潮，“盗用発見と自動採点によるプログラミング演習課題の評価支援システム”，DEIM Forum 2011，E4-2

[3] Rishabh Singh， Sumit Gulwani， Armando Solar-Lezama，” Automated Feedback Generation for Introductory Programming Assignments”, PLDI’13 ，ACM 978-1-4503-2014-6/13/06，June 16-19，2013

[4] Jose Luis， Fernandez Aleman， ”Automated Assessment in a Programming Tools Course”，Digital Object Identifier 10.1109/TE.2010.2098442

[5] Jinrong Li，Wei Pan，Ren Zhang，Feiquan Chen，Shenglong Nie，and Xiaoming He，” Design and Implementation of Semantic Matching Based Automatic Scoring System for C Programming Language”，Spring-Verlag Berlin Heidelberg 2010，LNCS 6249，pp.247-257

[6] 武井 惠雄，渡辺 博芳，荒井 正之，”事例に基づく初等アセンブラプログラミング評価支援システム”, 情報処理学会論文誌, Jan 2001, Vol.42 No.1

[7] G. García-Mateos， J. L. Fernández-Alemán，“A course on algorithms and data structures using on-line judging”， Proc. 14th Innov. Technol. Comput. Sci. Educ., 2009, pp. 45–49.