JARファイルを対象とする自動採点機能を有した Javaプログラミング実習用システムの構築と運用

毛利　元昭（愛知大学経営学部）

要旨

プログラミングの実習系授業の実施において，担当教員側の負担軽減や学習者側でのテストの重要性から，自動採点機能を有した実習用システムやウェブサービスを利用するケースは多い。しかし，既存のそれらはソースコードを提出させる形にしている。そのような形式を単純に用いた場合，学習効果の低下やシステム負荷の増大を招く恐れがある。それらを抑制すべく，筆者が本学経営学部で担当しているプログラミング系授業では，JARファイルを対象とするJavaプログラミング実習用システムを独自に構築・運用している。本システムは実行可能JARファイルを受講者のローカルな開発環境で作成させることを前提とするが，本稿では，なぜ本システムによって問題を抑制できるのか，どのように設計・実装・実現しているのかについて説明するとともに，他の様々な機能について説明する。また，受講者の反応を中心に本システムの運用の状況について述べる。

キーワード：自動採点，プログラミング学習，Java，GUI，PHP，シェルスクリプト

1. はじめに

プログラミングの実習系授業の実施において，担当教員側の負担は大きい。資料・教材作成と講義だけでなく，課題の細かな出題，受講者の課題進捗状況の把握，採点とフィードバック，質問やトラブルへの対応などが求められる。しかも，その多くが個別対応と即時性を求められ，授業中の担当教員とアシスタントは，講義を中断して対応に追われ続けることになる。情報工学や情報科学を専攻していない初心者50～60人規模の複数クラスを単独で担当するような場合，授業を正常に実施するには何らかの効率化・自動化が必要と言える。

また，プログラミングの学習において，作成したプログラムが想定通りの動作を行うものであるかを確認すること，いわゆるテストは必須と言える。テストデータと期待される結果を担当教員側で用意できる場合は多いが，簡単なコーディングでも苦戦する初心者にとって，テストを正しく実施することは容易でない。開発環境に付属するテスト用のツールなどを利用してもらうおうとしても，

さらなる学習コストを必要とするため，やはり初心者には難しい。

こういった背景から，プログラミングの実習系授業の実施において，自動採点機能を有した実習用システムを授業担当者が構築・運用するケースがある。例えば，名古屋工業大学のCAPES[1)]や早稲田大学のWaseda Online Judge[2)]がある。同様に，近年には様々なプログラミング学習用ウェブサービスが開始され，そのほとんどは課題提示と自動採点（オンラインジャッジ）機能を有し，自身の課題進捗状況を保存・把握できるものもある。簡易的なプログラム開発環境を有しているものもある。日本人の利用者が多いものとしては，前述のAizu Online Judge[3)]やProgate[4)]，paiza[5)]などがあり，世界的にはAtCoder[6)]が有名である。しかし，授業で用いられる実習用システムは授業に閉じたものであるため，学外者には利用できない。また，プログラミング学習用ウェブサービスは内容のアレンジができない。そのため，筆者も本学経営学部で担当しているプログラミング系授業の実習用システムを独自で構築・運用することとした。

ここで，既存の多くの実習用システムやプログラミング学習用ウェブサービスは，ソースコードを提出させる形にしている。具体的には，ウェブページ上に穴埋め式のソースコードと条件や期待される働きが提示され，解答欄に解答を記入あるいは作成したソースコードのファイルを投稿し，文字列やプログラムの実行結果を正答例のものと比較することで，点数化や合否判定を行う。しかし，このような方法を単純に用いた場合には下記のような問題が生じる。

1. 受講者がプログラムの全体像を把握・コントロールする機会の減少
2. 受講者のコーディングに関する自由度の減少
3. コンパイルや実行の機会の減少
4. コンパイル不可能な状態での提出の増加
5. システム負荷の増加

これらの問題を抑制するために，筆者はローカルの開発環境にて作成された実行可能JARファイルを提出させることを前提とした，自動採点機能を有するプログラミング実習用システム[注1)]を考案・構築した。本稿では，主にそのシステム設計・構成や運用の状況を説明する。

2. 受講者に実行可能JARファイルを提出させる意義

JARとはJava Archiveの略であり，Javaのクラスファイルやデータファイ

注1) ソースコードや課題バンク等は下記のURLにて公開している。

https://halo.aichi-u.ac.jp/~mouri/works/programming_training_system_java.zip

ルなどをZIP圧縮でひとまとめにした形式である。実行可能JARファイルとは，実行用のマニフェストファイルを内に含んだJAR形式のファイルである。Javaがインストールされた環境であれば，「java -jar」コマンドとともにファイルを指定することでプログラムを実行できる。

本システムにおいて特徴的な点は，受講者のローカル環境にて実行可能JARファイルを作成させてから，そのファイルを提出させることである。このような形式をとる自動採点システムは，筆者が調べた限りでは見当たらなかった。既存のプログラミング実習用のシステムでよくみられるのは，ブラウザ上にソースコードを入力させる形である。しかしこの形は，第1章で挙げたような問題を抱えている。ここでは，実行可能JARファイルを提出させることで，なぜそれらの問題が解決されるのかを述べる。

自動採点がプログラムの実行を伴わない形であれば，「正解となる文字列を入力する」ことと等価である。ソースコードに対してリッチな自動採点を行う場合，字句解析・構文解析とパターンマッチングなどを利用することとなる。このような形は，計算機リソースと時間が必要になることに加え，自動採点スクリプト等の作成に高い技術力と多大な労力が必要になる。筆者を含め，言語処理などの分野を専門としない者には荷が勝ちすぎる状況と言える。他方で，簡易的な自動採点を行う場合，空白や改行を除いて比較差分を評価することとなる。このような形は，受講者のコーディングに関する自由度がほぼ無い。答案が記号・単語あるいは短文になるような工夫がなされていることが前提であり，「デバッグせよ」「○○ができるように改良せよ」のような，プログラムの全体像を把握・コントロールさせるような大まかな指示による課題の答案を，実質的に採点できない。つまり，応用力を養うことが目的の課題の出題が困難である。

自動採点がプログラムの実行を伴う形であれば，入出力の関係は「仕様」つまり厳密にしつつ，コーディングに関する自由度は高くできる。よって，応用力を養うことが目的の課題でも自動採点可能なものが作りやすい。しかし，ソースコードを提出させる形であれば，サーバ側でコンパイルが必要となる。コンパイルには計算機リソースと時間が必要であり，システム負荷を増大させるため，多数の答案提出が重なった場合などにリクエストのタイムアウトが生じやすくなる。タイムアウトした場合には採点結果を正しく示せないため，答案の再提出が必要となり，その結果，さらなるシステム負荷が生じることとなる。それ以上に問題なのは，実行を受講者の主体で行う必要がないため，受講者からコンパイルや実行の機会を奪うことにつながる点である。

プログラミングにおいて自らの手によるテストは重要であり，学習という観点からも試行錯誤は重要である。そもそもコンパイル不可能なソースコードが提出される可能性も大きく，デバッグしにくい状況は避けるべきである。なお，近年にはブラウザ上で利用できるクラウド開発環境も現れてきたが，導入が困難な上に様々な制約やリソースの問題を抱えており，授業での利用は現時点では考えていない。

一方で，JARファイルを提出させることを前提とした場合，話は変わってくる。筆者の授業では，開発環境のEclipse[注2)]を受講者に利用させている。このEclipseは，実行可能JARファイルの作成（エクスポート）に際し暗に実行を必要とする。エクスポートには起動構成が必要なのだが，ソースコードから実行させることで，起動構成が自動生成されつつコンパイルが行われるのである。よって，受講者は実行とそれに付随するコンパイルを，提出のたびに自然に繰り返す[注3)]ことになる。つまり，データ入力を含め実行の様子を主体的に確認することにつながる。また，コンパイル時にエラーが生じた場合はエクスポートされないため，コンパイル不可能な状態での提出が防止できる。もちろん，Eclipseが有するハイライト機能や入力補完機能など，リッチな開発環境としての機能をローカルのリソースで利用するため，コードの誤りを避けつつデバッグがしやすい上に，システム負荷は総合的に小さい。

これらのメリットを成立させる要因は，JARファイルがOSに依存せず実行できる点である。Javaは仮想マシン（VM）上で中間言語ファイルを実行する仕組みをとっており，受講者のローカル環境（Window，Mac，Linux問わず）で作成された実行可能JARファイルであっても，多くの場合サーバ側（本システムの場合はLinux）でそのまま実行できる。なお，JavaFX[注4)]など外部ライブラリが必要な場合やVMのバージョン違いによっては若干の工夫が必要であるが，現行の演習課題およびシステムでは解決済みである。具体的には，受講者にはEclipse上でJava SE-1.8の形式でプロジェクトを作成・実行・エクスポートしてもらい，サーバ側では実行時にオプ

注2) Javaの主要な統合開発環境。日本語化ツールPleiadesとともに無償利用できる。

注3) 実際には，起動構成が一旦生成されたらその後のエクスポートの際に実行は不要であるが，受講者にそれを知らせないことで実行が習慣化される。

注4) Java VM上で動作するGUIライブラリ。現在はJavaに同梱されなくなったため，OpenJFXを導入・利用する形となり，初心者にとっては利用・保守し難くなっている。

ションを付け加えている。

実行可能JARファイルによるメリットは他にもある。「javap[注5)]」コマンドを用いてJARファイルを解析することで，プログラムの構造を確認できる点である。コンパイル後，つまり字句解析・構文解析とそれらの整理が行われた後であり，ソースコードと比べて表現の差異がある程度吸収されるため，構造が同じであれば正答例と比較しやすい。すなわち，入出力の関係だけでなく，プログラムの構造も採点の対象にできる。もちろん，採点者には「javap」コマンドに関する知識や解析結果の読解・利用技術が必要となるが，複雑な状況を求めなければ対応できるケースも多い。筆者の場合，標準的には，クラスやメソッドの情報などを抽出し必要に応じて正答例のものと比較する形で自動採点の際に利用している。本システムでの具体的な利用方法については第5章で説明する。

3. システム設計

3.1 システム要求

JARファイルを対象とする自動採点機能を有したJavaプログラミング実習用実習用システムを構成するにあたり，当初から考慮したシステム要求は以下の通りである。

1. 履修者がウェブブラウザからアクセスでき，課題の表示や提出，評価の確認もウェブブラウザ上で可能
2. 利用にはログインを必要とし，テスト用ユーザを除いて学内の認証サービスにて認証
3. 記述式の課題も表示や提出ができ，正答例を別途用意することで自動採点にも対応
4. JARファイルが提出された場合，ファイルの解析および実行を自動で行い，その結果をファイルとして保持
5. 自動採点の場合，提出から結果の表示までを20秒以内とし，その間に60人程度がアクセスしてもシステムダウンが発生しない（タイムアウトは許容）
6. 管理者の画面から，提出されたファイルや答案の確認，採点の修正や手動採点が可能
7. 管理者の画面から，クラスごと・授業の回ごとに課題の公開・提出期間等の設定が可能

前述の要求が満たせるように，ハードウェアの選定と環境構築およびシステム設計を行っている。現行機のハードウェアと環境は表1の通りである。なお，当初のマシンはスペックやバージョンが現行

[注5)] JavaのSDKに付属する，バイトコードで書かれたクラスファイルを解析・逆アセンブルして人間が理解できる文字列に変換するツール。

表1　システムのハードウェアと環境

本体	PC/AT互換機
CPU	Core i5 760 2.80GHz
RAM	4 GB
ストレージ	HDD 320GB
ネットワーク	1000Base-TX
OS	Lubuntu 20.04 64bit
ウェブサーバ	Apache 2
サーバ側処理	PHP 7, Bash 5
ブラウザ側処理	jQuery 3
データベース	Maria DB 10
Java環境	OpenJDK 16, OpenJFX
画面キャプチャ	Xvfb, ImageMagick

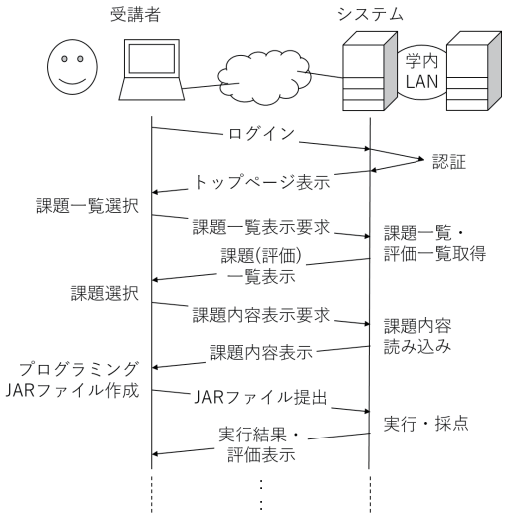

図1　受講者によるシステム利用の流れ

より大きく劣るが，HDDの障害を除き，運用中の過負荷によるシステムダウン等は発生しなかった。

受講者によるシステムの利用の流れは図1のようなものを想定した。まずログイン認証を行い，受講者として登録されていればトップページが表示される。トップページには各機能や資料へのリンクが示されており，受講者が課題一覧を選択すると，その時点に出題されている課題の一覧が表示される。このとき，進捗状況を把握できるよう，各課題の評価を記号化したものが各課題の内容へのリンクとして表示される。受講者が課題を選択すると，それに対応した課題内容が表示される。受講者は表示された条件やサンプルプログラムをもとにプログラミングを行い，ローカル環境にて実行可能JARファイルを作成する。受講者が課題内容の画面からJARファイルを提出すると，サーバ側で実行や採点が行われ，実行結果や評価が表示される。その後，受講者はページを戻ったり課題一覧を表示させたりして，課題への再挑戦や次の課題への挑戦を行っていく。

前述の要求に基づくシステムの具体的な要件および追加機能は詳細多岐にわたるため，必要に応じて後述する。

3.2　フォルダ構成

本システムはウェブサービスとして運用するため，ウェブ上から直接アクセスできる表層部と，そうでない内部に分かれたフォルダ構成とした。概要は図2の通りである。

表層部に関しては，PC内の著者用の

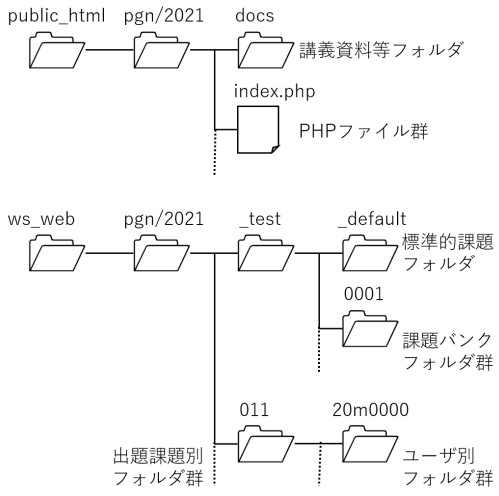

図2　システムのフォルダ構成

ホームフォルダにあるフォルダ「public_html」内に「授業略称・年度」のフォルダが用意してあり，その下にPHPファイル群を置いてある。「index.php」がトップページとなっており，そこから各ページへのリンクをたどる形となる。また，簡便のため，講義資料等は「docs」というフォルダに別途置き，トップページにもリンクを載せている。表層部とそれに付随する各機能は主にPHPの形でモジュール化して用意するが，自動採点そのものはPHPでは行わない。内部による処理結果を受けて，結果を表示するにとどめる。なお，PHPファイル群については第章で詳細を述べる。

内部に関しては，これもホームフォルダ下のフォルダ「ws_web」内に「授業略称･年度」のフォルダが用意してあり，その下に出題課題別のフォルダ群と，各課題内容と採点用スクリプトがフォルダ群（課題バンク）として収められた「_test」がある。

出題課題別のフォルダ群は，出題する課題を設定する際に，その課題の番号で自動作成される。例えば，第02回の授業の3つ目の課題であれば「023」となる。この出題課題別フォルダの下にさらに受講者ごとのフォルダを自動作成し，提出されたJARファイルやその実行結果，自動採点のログ，その他のファイルなどを，それぞれ指定の名前で保存する形である。

課題バンクにおいては1つの課題につき1つのフォルダとしており，課題の識別子をフォルダ名としている。それぞれのフォルダに，データベースへの登録用設定ファイル，課題の説明，ソースコード，正答例，自動採点スクリプト，その他のファイルなどを指定の名前で収めてある。また，課題バンクとは別に「_default」フォルダを用意してあり，ここには標準的な課題の説明と自動採点スクリプトをタイプ別で収めてある。課題バンクのフォルダ内に説明やスクリプトのファイルが存在しなければ，これらが利用される。なお，自動採点スクリプトについては第5章で詳細を述べる。

課題バンクフォルダ内には，主に下記のファイルが収められている。

● setting.txt

データベースへの登録用設定ファイル

- direction.txt
  課題の説明
- figure.png
  説明の補足用の画像
- code.txt
  ソースコード（提示用）
- answer.txt
  正答例
- answer.png
  画像を提出させる課題用の正答例
- test_cmd.sh
  自動採点スクリプト
- input*.dat（*は任意の数値）
  自動実行時の入力
- output*.dat（*はinputに対応する数値）
  自動実行時の想定出力（比較用）
- output_jp*.txt（*は数値）
  javapによる解析結果の抜粋（比較用）
- diff_res*.log（*は数値）
  実行結果に対する想定差分
- その他，配布ファイルなど

3.3　データベース構成

データベースは授業略称・年度で作成し，その中に下記の名称・項目を持つテーブル群を用意する。

- users　　ユーザ情報テーブル
  - u_id　　学内アカウント名
  - u_name　氏名
  - cls_id　クラス識別子
  - grd_id　権限
- p_bank　　課題バンクテーブル
  - p_id　　課題識別子
  - type　　課題タイプ
  - title　　課題タイトル
  - dist　　配布ファイル名（任意）
- p_link　　各授業の設定テーブル
  - lnk_id　授業の識別子（cls_id+l_id）
  - cls_id　クラス識別子
  - l_id　　授業の回
  - t_open　授業開始日時
  - t_view　課題公開日時
  - t_late　課題締切日時
  - p_id1～p_id8
    各出題課題の課題バンク内識別子
- evals　　評価テーブル
  - u_id　　学内アカウント名
  - s010～s159
    各出題課題の評価点数

各テーブルおよび各項目の意味の詳細は，以降の項でそれぞれ述べる。

3.3.1　users

ユーザ情報を管理するテーブルである。主キーとなるユーザ識別子を学内アカウント名と同一とするのは，本システムが学内のシステムであり，学内の別サーバで認証を行うためである。また，テスト用の特別なユーザを除き，パスワードやそのハッシュ値は記録しない。

クラスごとに時限が異なるためクラス識別子を，受講者と担当教員では利用できる機能を分けるため権限をそれぞれ項目として持たせてある。担当教員が進捗

などを把握しやすくするため，受講者には本システムを通して氏名を登録させる。

3.3.2 p_bank

課題バンクを管理するテーブルであるが，実質的にインデックスである。主キーとなる識別子は前述のフォルダ名に対応しており，他の項目はフォルダ内の設定ファイルに記述された情報が登録されている。メンテナンス性とデータベースの負荷の観点から，各課題の詳細はここに登録していない。

課題タイプは下記の通りである。

1. JARファイルを受け取るのみ
2. JARファイルを受け取って自動採点（テキストベースで自動実行）
3. JARファイルを受け取って自動採点（起動時にスクリーンショット）
4. テキストデータを受け取るのみ
5. テキストデータを行ごとに自動採点
6. PNGファイルを受け取るのみ
7. ドキュメントを受け取るのみ
8. ZIPファイルを受け取るのみ

課題を出題する際に内容を把握しやすくするため，タイトルを項目として持たせてある。また，課題によってはファイルを配布するため，そのファイル名を登録できるようにしてある。ただし，そのファイルの実体は課題バンクフォルダに収めてあることとする。

3.3.3 p_link

各授業の設定を管理し，課題バンクを紐づけするテーブルである。主キーとなる識別子はクラスと授業回（2桁）から構成されているため冗長であるが，検索の観点からクラス識別子と授業回も項目として持たせてある。

授業の開始日時は，実質的に，その回の課題の提出が可能となる日時である。それとは別に，課題の公開日時を設けたのは，予習を促すためである。また，学期の終盤に用意した総合的な課題などは，早めに公開することでモチベーションにつながることが期待され，それを可能とするためである。課題の締切日時は，受付を停止するというよりは，以降を遅延提出として扱う日時である。

1つの授業に最大8つの課題を設定できるよう，課題バンク内の識別子を登録するための項目を8個設けてある。上限を8にした理由は，難易度にもよるが，課題数が多すぎると受講生が対応しきれずモチベーションを下げる恐れがあるためと，現在までの授業実施において不足していないためである。

3.3.4 evals

各出題課題の評価点数を管理するテーブルである。主キーとなるユーザ識別子を除き，項目名は先頭の"s"に続いて授業回（2桁）とその中での課題の順番からなる。現在の実装では，クラスを分け

ずひとまとめで扱っている。また，項目は後から追加していく形をとっている。

1つの授業で設定できる課題は8つまでとしたが，それとは別に，授業ごとに出欠を0番，感想を9番とし，その状況も登録対象としている。現状では，出欠に関しては「無断欠席」「欠席」「遅刻」「出席」を区別し，感想に関しては「短文」「長文」を区別している程度である。

4. PHPファイル群

PHPファイル群は，大きく分けて以下の4つに分けられる。

1. ユーザ情報および認証関連
2. 課題および評価の提示関連
3. 課題の提出および採点関連
4. 管理関連

それぞれについて，以降の節で述べる。

4.1 ユーザ情報および認証関連

該当するファイルは以下の通りである。

- index.php
  トップページ
- config.php
  サイト内の共通設定，セッションの維持，アクセス制限など（サイト内のほぼ全てのPHPファイルから冒頭で呼び出し）
- add_u.php
  受講者の初回登録ページ
- edit_u.php
  ユーザ情報の確認・更新ページ
- update_u.php
  ユーザ情報の更新
- login.php
  ログイン認証，データベースからのユーザ情報の読み込み
- logout.php
  ログアウト

これらのファイルが担当する機能は論旨に大きく関わらないため，説明は割愛する。

4.2 課題および評価の提示関連

該当するファイルは以下の通りである。

- show_eu.php
  課題（評価）一覧ページ
- show_p.php
  課題内容ページ
- show_p9.php
  授業への意見・感想を入力するページ
- disp_png.php
  内部に保存された画像の表示
- dl_file.php
  内部に保存された提出ファイルのダウンロード
- dl_file_d.php
  配布ファイルのダウンロード

この中のshow_ls.phpとshow_p.phpについて，以降の項で説明する。

4.2.1 show_eu.php

現在出題されている全ての課題内容へのリンク一覧ページであるが，同時に

図3　課題（評価）一覧ページの例

ユーザの評価一覧としての機能を持つ(図3)。受講者が自身の進捗を把握しやすいよう，授業回ごとの行に締切日時と出題課題の評価の列を持たせ，表にしてある。各評価は記号で表し，

- ×　　未提出
- ▲　　不合格
- 確認中 手動での採点が未完了
- ●　　合格（遅刻提出）
- ○　　合格
- ◎　　合格（加点対象）

としてある。クリックするとそれぞれの課題内容ページへ遷移する。なお，各授業回の最後の課題は授業の感想を任意で書かせる形である。

評価の達成度は概算評点という形で数値的にも示してある。これは，このページ内での各評価記号の数を単純に重みづけ和した値であり，実際の評点とは異なるが，目安となる。

4.2.2　show_p.php

課題の詳細な内容が表示されるとともに，答案の提出を受け付けるページである。その一例を図4に示す。課題番号はクラスと授業回（2桁）と課題の順番からなるが，EclipseでJavaプロジェクトを作成する際のプロジェクト名も兼ねている。そのため，課題の説明文中のプロジェクト名が課題番号になるよう，表示時に置換している。タイトルに続き，説明，コード，図，配布物，その他，提出，判定，コメント，正答例が並ぶが，コード以下はそれぞれ不要な場合には表示されない。

「説明」には前述のdirection.txtの中身を修飾済みhtmlとして表示し，「コード」にはcode.txtの中身を標準テキストとして読み込んだ後にhighlight.js[注6)]を用いて装飾表示する。そのため，「説明」にはhtmlの機能を利用でき，「コード」にはソースコードファイルをほぼそのまま流用できる。なお，「コピー」のボタンを押すとソースコードがクリップボードにコピーされる。「図」にはdisp_png.phpを用いてfigure.pngを表示し，「配

注6)　Webページ上に表示したソースコードに色を付けるJavaScriptライブラリ。
https://highlightjs.org

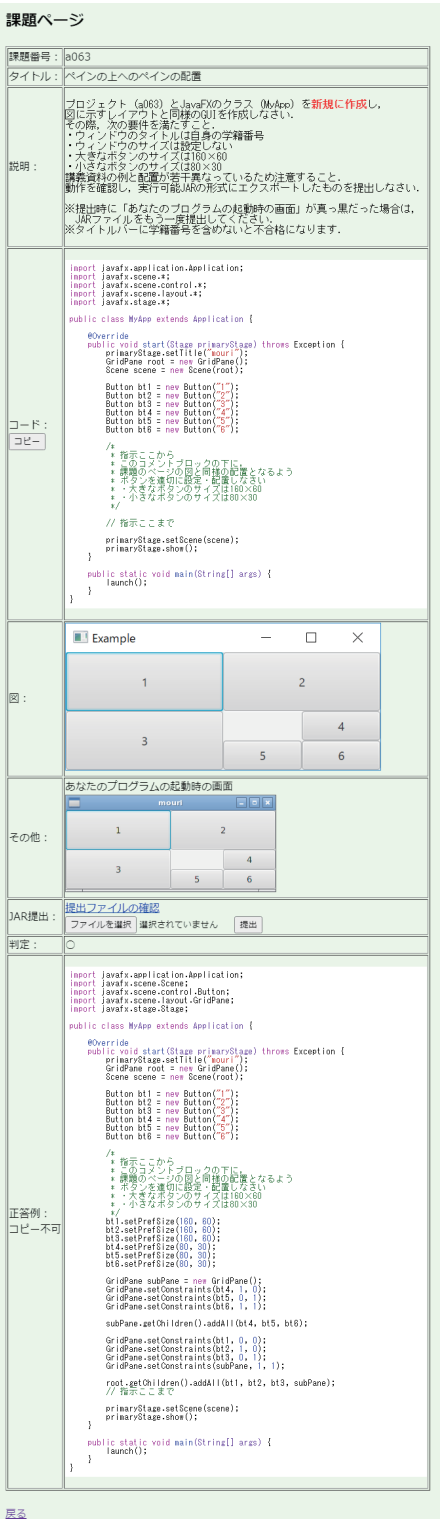

図4　課題ページの例

布物」にはdistに記載されたファイルをダウンロードできるリンクを表示する。「その他」には，現在はJavaFXを用いる課題について，提出されたJARファイルをサーバで自動実行した際のスクリーンショットを表示する。「図」の近くに配置することで受講者自身が比較しやすいようにしてある。

「提出」は課題タイプによって表示が変わる。タイプ1〜3ではJARファイルの提出用inputと提出ファイルのダウンロードリンクを，タイプ4〜5では解答欄のテキストエリアを，タイプ6ではPNGファイルの提出用inputと提出済みの画像を，タイプ7〜8ではそれぞれドキュメントファイルやZIPファイルの提出用inputと提出済みファイルのダウンロードリンクを表示する。

「判定」には答案の評価結果を表示する。なお，合格（加点対象）の際はページ背景全体に花火が上がり続けるアニメーションが表示されるようになっている。「コメント」には答案に対する教員からのコメント（出題課題ごとの受講者のフォルダにcomment.txtとして保存）を表示する。

「正答例」は課題バンクフォルダ内に該当ファイルが存在する場合にそれを表示する。answer.txtに関しては，その中身を標準テキストとして読み込んだ後にhighlight.jsを用いて装飾表示する。ただし，タイプ5ではanswer.txtの各行の

第1タブ前まで（または改行前まで）を標準テキストとして表示する。これは，タイプ5が別解をサポートしているためである。answer.pngに関しては，disp_png.phpを用いて表示する。なお，「正答例」は課題の提出期限後にのみ表示し，その部分のテキストはコピーできないようにしてある。期限後にも提出を受け付けることと併せて，受講者が復習する機会を設けるためである。

4.3　課題の提出および採点関連

該当するファイルは以下の通りである。

- ul_jar.php
  JARファイルのアップロード，自動採点スクリプトの呼び出し，結果表示
- ul_txt.php
  テキストエリアのデータの投稿，自動採点スクリプトの呼び出し，結果表示
- ul_file.php
  その他の指定ファイルのアップロード
- submit_p9.php
  授業への感想・意見を提出

この中のul_jar.phpとul_txt.phpについて，以降の項で説明する。

4.3.1　ul_jar.php

最初に，POSTされたJARファイルを既定の名前「a.jar」にリネームし，出題課題のユーザ別フォルダに保存する。続いて，実行フォルダを同フォルダに移し，「timeout[注7)]15」コマンドとともに自動採点スクリプトを実行する。これにより，無限ループなど実行時のトラブルが発生してもシステムが機能不全に陥らない。なお，JavaFXの課題の場合はGUIが含まれるため，「xvfb-run[注8)] -a」コマンドを「timeout」の前に付け加えて仮想ディスプレイ上で自動採点スクリプトを実行する。

自動採点スクリプトは，リダイレクトを用いて途中の標準出力や標準エラー出力を全てそれぞれ指定のファイルに吐き出し，最終的な評価値を表す1文字（不合格："F"，確認中："K"，合格："G"，加点対象の合格："S"）のみを標準出力する形をとる。この評価値をPHP側で受け取り，それに基づいてデータベースへの評価の登録と，ユーザへ示す評価結果ページを作成・表示する。

評価結果ページには，正解プログラムの実行結果と提出されたプログラムの実行結果を併記し，diff_match_patch.js[注9)]を

注7) 引数で指定した時間で強制的に停止する形で実行するためのコマンド。

注8) 仮想のディスプレイフレームバッファを生成して実行するためのコマンド。

注9) 2つのテキストを比較し，異なる部分を色付け表示するJavaScriptライブラリ。
https://github.com/google/diff-match-patch

用いて両者の差異を色付けする。出題課題のユーザ別フォルダ内にa.pngが存在していれば，それを「起動時の画面」として表示する。これらの他，同フォルダ内に存在する「*.log」ファイルの中身を列記するが，これは出題者側のデバッグ兼，評価結果に対する受講者への説明用である。なお，合格（加点対象）の際は，課題内容ページと同様に花火のアニメーションが表示されるようになっている。

4.3.2　ul_txt.php

ul_jar.phpと大部分が共通しているため，異なる部分についてのみ説明する。最初に，POSTされたテキストエリアのデータを既定の名前「a.txt」にリネームし，出題課題のユーザ別フォルダに保存する。評価結果ページには，テキストデータを受け取るのみの課題については，提出された解答を表示する。行ごとに採点する課題については，提出された解答とともに，誤りが存在する箇所を行ごとに表示する。

4.4　管理関連

該当するファイルは以下の通りである。

- admin.php
  管理用の機能へのアクセスページ
- show_ls.php
  授業情報の一覧ページ
- edit_l.php
  授業ごとの設定ページ
- update_l.php
  授業情報の更新
- edit_el.php
  授業ごとの評価の管理ページ
- show_ap.php
  課題ごとの答案の一覧・編集ページ
- read_p9.php
  授業ごとの感想・意見の確認ページ
- update_es.php
  評価の更新
- update_ps.php
  課題バンク情報をデータベースに登録
- show_es.php
  数値化した評価の一覧ページ
- edit_us.php
  ユーザ情報の管理ページ
- update_us.php
  ユーザ情報の更新

この中のedit_el.phpとedit_ap.phpについて，以降の項で説明する。

4.4.1　edit_el.php

同一授業における同一クラスの全受講者の全評価を表示することに加え，各評価の編集も可能なページである。その一例を図5に示す。出題した課題について，どれだけの受講者が答案を提出したか，合格したかが分かるため，授業中の進度調整に利用できる。評価の編集については，誤りを避けるため，編集対象の受講者を選択してから課題ごとのドロップダウンリストで評価を編集する形にした。

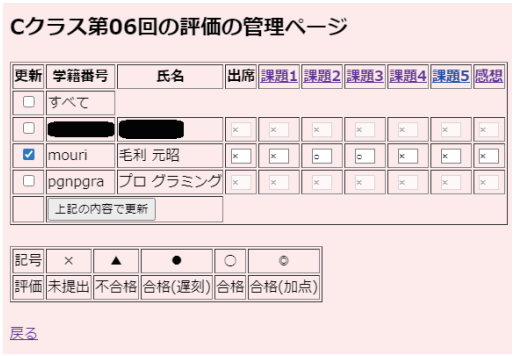

図5　edit_el.php

図6　show_ap.php

ただ，ほとんどの場合，評価の編集の際は答案を確認する必要がある。項目のリンクから後述の「課題ごとの答案の一覧・編集ページ」に遷移できるため，筆者はそこで評価を編集することが多い。

4.4.2　show_ap.php

同一課題における同一クラスの全受講者の答案・評価を表示することに加え，各評価の編集も可能なページである。その一例を図6に示す。提出されたファイルへのリンクやテキストは原則全員分を，ユーザ別フォルダ内の画像ファイル「a.png」に関しては「確認中」のものを表示する。これにより，手動採点が必要な対象を優先的に確認できる。オプションを指定することで，全員分の画像や，ユーザ別フォルダの任意のテキストファイルの中身を表示できる。コメント欄には，受講者個人へのコメントを書きこむことができる。筆者は主に，手動採点で評価を「不合格」にした場合に，その理由を伝えるために利用している。特にオンライン学習の際に本機能が活躍した。評価およびコメントの編集については，誤りを避けるため，対象の受講者を選択してから課題ごとのドロップダウンリストで評価を編集する形にした。

5. 自動採点スクリプト

5.1　要求

自動採点を実装するにあたって要求としたことは次の通りである。

1. Linuxで利用できる無償のソフトウェア群・コマンド群で構成・構築
2. PHPから呼び出し可能かつ採点結果をPHPから参照可能
3. JARファイルを解析し採点に利用可能
4. JARファイルの実行中に標準入力からデータ入力する形に対応し，標準出力は指定のファイルに保存
5. JavaFXの課題の場合は起動時のスクリーンショットを撮影可能

6．テキストの自動採点は別解に対応

それぞれの理由を説明する。まず1については，筆者がシステム用の予算を有していないためである。また，サーバとなるマシンのOSがLinuxであることは前提のため，その上で利用可能な形で全てを構成・構築する必要がある。2については，表層部をPHPで構成することが前提のためである。PHPからの呼び出し方法や参照方法の自由度は高いため，容易な要求と言える。3については，第2章で述べたように，プログラムの構造も採点対象に含めるためである。

4については，入出力に着目した採点のためである。ソースコードから実行する形であればコンパイル前にソースコードの一部を置換することでデータを差し替えられるが，本システムの場合はそれができない。実行時引数かファイル，標準入力で与える必要がある。ここで，実行時引数やファイルを用いる場合は，初心者にとってハードルが高い。例えばEclipseの場合，起動構成の適切な設定やファイルの適切な配置・編集などが必要となり，受講者が対応しきれずトラブルが多発する恐れがある。一方，標準入力で与える場合，その部分を提示するソースコードにあらかじめ含めることができる。また，実行時の標準出力を指定の名前のファイルへ一旦保存することで，実行と採点を切り離して考えることができる。

5については，GUIを表示するプログラムの採点を高効率化するためである。本来，GUIを表示するプログラムを採点するには，提出されたプログラムを採点者の手元で実行する必要がある。その負担は大きい。しかし例えば，GUIのレイアウトのみを採点対象とする場合，起動時のスクリーンショットが表示されていれば，それで採点できる。また，提出先の誤りなどを受講者側でも採点者側でも採点前に気付きやすくなる。

6については，答案に自由度を持たせておきたいためである。例えば数式や論理式を記述させる課題には，同じ結果を生むが表現の異なる答案を想定できる。現実的には全てを網羅できない場合も多いが，別解はできるだけ正解としたい。

5.2　テキストベースでの自動実行・採点

タイプ2の課題，つまり，実行可能JARファイルをテキストベースで自動実行および採点する場合の標準のスクリプトを図7に示す。1～4行目はフォルダの把握と移動，6行目は以前のファイルの削除である。7～12行目では，JARファイル内に含まれるクラスファイルをそれぞれ「javap」コマンドで解析し，その結果を一旦output_jp.txtに保存している。環境によっては元のファイル名のエンコードがShift-JISであるため，それに対応してある。

14～22行目では，自動実行および実

行結果と正答例による出力結果との比較を行っている。まず，課題バンクフォルダ内にファイル「input*.dat」（*はワイルドカード）が存在する個数分，それぞれを標準入力に流し込みながら実行し，それぞれの標準出力を対応するファイ

```
1  d_sh=$(pwd)
2  p_id=$(basename ${d_sh})
3  flag_k=0
4  cd ../../${1}/${2}
5
6  rm -f diff* output* error* 2>> error.log
7  zipinfo -1 a.jar *.class | sed 's/.class$//g' ¥
8   | xargs javap -classpath a.jar -verbose 1> output_jp.txt 2>> error.log
9  if [ ! -s output_jp.txt ]; then
10     zipinfo -1 -O sjis a.jar *.class | sed 's/.class$//g' ¥
11  | xargs javap -classpath a.jar -verbose 1> output_jp.txt 2>> error.log
12 fi
13
14 for f_i in ${d_sh}/input*.dat
15 do
16     f_o=$(basename ${f_i/input/output})
17     cat ${f_i} | java -jar a.jar | sed -r 's/¥x1b¥[[0-9;]*m//g' > ${f_o}
18     if [ -s ${d_sh}/${f_o} ]; then
19        flag_k=1
20        diff -dBw --strip-trailing-cr ${f_o} ${d_sh}/${f_o} >> diff_res.log
21     fi
22 done
23
24 pjnm=`sed -n 1p output_jp.txt | sed -e 's/.*¥!¥///' -e 's/¥/.*//'`
25 cat output_jp.txt | grep "Classfile" | sed -e 's/.*¥///g' | sort > output_jp2.txt
26 cat output_jp.txt | grep "static .*);$" | sed -e 's/^.*static/static/g' -e "s/${pjnm}//g" ¥
27  | sort >> output_jp2.txt
28 cat output_jp.txt | grep "extends" | sed -e "s/${pjnm}//g" | sort >> output_jp2.txt
29 if [ -s ${d_sh}/output_jp2.txt ]; then
30     flag_k=1
31     diff -di --strip-trailing-cr output_jp2.txt ${d_sh}/output_jp2.txt >> diff_jp2.log
32 fi
33
34 if [ ${flag_k} -eq 1 ];then
35     if [ -s diff_jp2.log ]; then
36        echo -n "F"
37     else
38        if [ -s diff_res.log ]; then
39           echo -n "F"
40        else
41           echo -n "G"
42        fi
43     fi
44 else
45     echo -n "K"
46 fi
```

フォルダの把握と移動

javap による解析

自動実行および
実行結果の比較

※1

※2

※1　javap の結果から情報を抽出

※2　正答例のものと比較

評価値を決定
比較無し：　"K"（確認中）
差分がある：　"F"（不合格）
差分がない：　"G"（合格）

図7　JARファイルをテキストベースで自動実行・採点するスクリプト
（タイプ2標準）

ル「output*.dat」に装飾用の制御コードを省いて保存する。それと同時に，「diff[注10)]」コマンドで正答例による出力結果と比較し，差分をdiff_res.logに追記していく。こうすることで，入出力の関係を採点の対象とできる。

24～32行目では，まずoutput_jp.txtからクラスファイル名，クラスメソッド名，子クラス名をそれぞれ抽出しoutput_jp2.txtに追記保存している。この際，パッケージ名(プロジェクト名)を消去している。課題バンクフォルダ内にoutput_jp2.txtが存在する場合は「diff」コマンドで比較し，その結果をdiff_jp2.txtに保存する。こうすることで，プログラムの構造の大枠部分を確認し，採点の対象とできる。

34～46行目では，これまでに得られた結果から評価値を決定している。何らかの比較が行われた場合，それらの差分が存在すれば（差分ファイルが空でないならば）不合格（F），存在しなければ合格（G）とし，比較が行われていない場合は確認中（K）としている。

この標準の自動採点スクリプトの利点は，課題バンクフォルダ内に比較用のファイルが存在するかどうかによって，自動的に適切な挙動をとることにある。そのため，入出力の関係やプログラムの構造の大枠のみ確認したい課題のほぼ全てが，比較用のファイルを用意するだけで自動採点できる。

5.3　JavaFXの課題の自動起動・採点

タイプ3の課題，つまりJavaFXを利用しており，自動起動時のスクリーンショット撮影を行う場合の自動採点スクリプトを抜粋し図8に示す。なお，割愛された部分はタイプ3の課題のものとほぼ同一である。

```
11 timeout 14 openbox 2>> error.log 1>> error1.log &
12 timeout 12 java --module-path /usr/share/openjfx/lib ¥
13  --add-modules javafx.controls,javafx.fxml,javafx.media,javafx.swing ¥
14  -jar a.jar 2>> error.log 1>> error1.log & sleep 5 2>> error.log 1>> error1.log
15 w_id=$(xdotool search --onlyvisible --name ${2}) 2>> error.log 1>> error1.log
16 xdotool windowactivate ${w_id} 2>> error.log 1>> error1.log ;¥
17  import -window ${w_id} -frame a.png 2>> error.log 1>> error1.log
27 if [ -s diff_jp2.log ]; then
28     echo -n "F"
29 else
30     echo -n "K"
31 fi
```

※1　仮想ディスプレイ上で起動
※2　スクリーンショットを撮影

評価値を決定
差分あり：　"F"（不合格）
差分なし：　"K"（確認中）

図8　JavaFXの課題を自動起動・採点するスクリプト（タイプ3標準）の抜粋

注10) Linuxに標準で搭載されている，2つのテキストファイルを比較し，差分を出力するコマンド。空白や改行を無視した比較も可能。

11～16行目は，タイプ2の場合の14～22行目を置き換える部分である。ここでは，まずウィンドウマネージャである「openbox[注11)]」を時限的に立ち上げた後に，JARファイルを実行している。その後，5秒待ってから「xdotool[注12)]」コマンドを用いてウィンドウを検索，最前面にし，「import[注13)]」コマンドでウィンドウ指定して撮影し「a.png」として保存している。ここで，検索に用いるのは学内アカウント名であり，受講者に提示するソースコード内に事前に含まれるよう工夫してある。こうすることで，採点のページにて起動時のスクリーンショットを表示できる。なお，第4.2節で述べたように，スクリプトの実行前に仮想ディスプレイを生成しておく必要がある。

27～31行目は，タイプ2の標準の場合の34～46行目を置き換える部分である。タイプ3の標準では，javapでの解析結果との比較のみで評価値を決定し，不備がなければ「確認中」としている。

5.4　テキストデータの行ごとの自動採点

タイプ5の課題，つまりテキストデータを行ごとに自動採点する場合は，例えば図9に示すような説明文と正答例を用意しておく。行ごとに採点するため，小設問の番号と回答の行・正答例の行とを対応させておく必要がある。正答例については，タブ文字で区切りながら別解を併記することができる。タブ文字はブラウザ上のテキストエリアへ入力することが困難であるため，区切り文字として適している。

direction.txt （課題の説明）より 回答させる部分を抜粋	1	変数 a の値が 56 である
	2	変数 a の値が 78 でない
	3	変数 a の値が正である
	4	変数 a の値が 2 以下である
	5	変数 a の値が 9 の倍数である
	6	変数 a の値が奇数である
	7	変数 a の値を 4 で割った余りが 3 である
answer.txt （正答例） タブ文字を網掛けで表現	1	a == 56　56 == a
	2	a != 78　78 != a　!(a == 78)　!(78 == a)
	3	a > 0　0 < a
	4	a <= 2　2 >= a
	5	(a % 9) == 0　0 == (a % 9)　a % 9 == 0　0 == a % 9
	6	(a % 2) == 1　1 == (a % 2)　a % 2 == 1　1 == a % 2
	7	(a % 4) == 3　3 == (a % 4)　a % 4 == 3　3 == a % 4

図9　タイプ5の課題の例（日本語の条件式をJava風に表記）

注11) Linux用のウィンドウマネージャ。シンプルかつ軽量で立ち上がりが早い。

注12) コマンドラインからウィンドウやマウス・キーボードを操作するツール。

注13) ImageMagickに付属する，スクリーンショットを撮影するコマンド。

タイプ5の課題の自動採点スクリプトを図10に示す。1～2行目はフォルダの把握と移動，4行目は以前のファイルの削除である。

6～21行目では，正答例との比較を行っている。answer.txtから1行ずつ読み取り，さらにタブ文字で分割した文字列と，答案であるa.txtの同じ行とを比較していく。もし文字列が一致した場合は，その行を正解したと見做してcount.txtに記録し，次の行に移る。タブ文字で分割された文字列群に一致するものが存在しなかった場合はそのまま次の行に移る。全ての比較を終えたら不要なファイルを削除する。この形であれば，タブ文字で区切ることで行ごとの別解を複数用

```
 1  d_sh=$(pwd)
 2  cd ../../${1}/${2}
 3
 4  rm -f diff* output* res.log count.txt
 5
 6  count=0
 7  cat ${d_sh}/answer.txt | while read line1
 8  do
 9      count=$((count+1))
10      echo ${count} >> count.txt
11      echo "${line1}" | tr '¥t' '¥n' | while read line2
12      do
13          diff -dBw --strip-trailing-cr <(sed -n ${count}p a.txt) <(echo "${line2}") > diff_res.log
14          if [ ! -s diff_res.log ]; then
15              echo ${count} >> res.log
16              break
17          fi
18      done
19  done
20
21  rm -f diff_res.log
22
23  count=1
24  cat count.txt | while read i
25  do
26      j=`sed -n ${count}p res.log`
27      if [ ${i} -eq ${j} ]; then
28          echo "" >> output.dat
29          count=$((count+1))
30      else
31          echo ">>" >> output.dat
32          echo ${i} >> diff_res.log
33      fi
34  done
35
36  if [ -s diff_res.log ]; then
37      echo -n "F"
38  else
39      echo -n "G"
40  fi
```

正答例から1行ずつ読み取り
タブ文字で区切る（=line2）

line2と回答の差分がない物があった場合
現在の行番号をres.logに記録して次の行へ

全行番号とres.logに記録した行番号を比較し
res.logに含まれる場合は空行
含まれない場合は誤り箇所を指摘する">>"を出力
（併せて，差分ファイルに記録）

評価値を決定
差分なし： "F"（不合格）
差分あり： "G"（合格）

図10　テキストデータを行ごとに自動採点するスクリプト（タイプ5標準）

意することができる。

23～34行目では，誤りのあった行を受講者に指摘するためのテキストファイル「output.dat」をcount.txtから作成している。それと同時に，誤りのあった行の番号をdiff_res.logに追記保存している。

36～40行目はdiff_res.logをもとに評価を行っている。ファイルが空でなければ不合格（F）とし，そうでなければ合格（G）としている。

5.5　特殊なケースの自動採点

ここまで，タイプ別の標準的な自動採点スクリプトについて説明してきたが，課題バンクフォルダ内にtest_cmd.shを用意することで，特殊なケースの自動採点も可能である。ただし，特殊なケースの自動採点を行うには，課題作成者による課題への深い理解が必要になるとともに，現状では「javap」コマンドの結果に対する読解力やLinuxシェルスクリプトでの記述力が必要になる。筆者自身，幾度となく試行錯誤を繰り返して課題を作成しており，かかった労力は大きい。しかし，学習効果を高めるような工夫のある課題には，このような機能が必要であると考える。

5.5.1　含まれる条件式の個数で評価値を変更する場合

例えば，条件式の個数で評価値を変更したい課題を作成した際は，図7の34行目以降を図11のように置き換えたものを用意した。図の34行目では「javap」コマンドによる解析結果output_jp.txtから条件式の個数を数え上げ，output_jp3.txtに出力している。40行以降にて，条件式の数が6以下であれば加点対象の合格（S）とし，10以下であったら合格（G），いずれでもなければ不合格（F）としている。

```
34  cat output_jp.txt | grep -c "if_" > output_jp3.txt
35
36  if [ -s diff_res.log ]
37  then
38      echo -n "F"
39  else
40      if [ `cat output_jp3.txt` -le 6 ]; then
41          echo -n "S"
42      elif [ `cat output_jp3.txt` -le 10 ]; then
43          echo -n "G"
44      else
45          echo -n "F"
46      fi
47  fi
```

→ javapの結果から条件式の個数（"if_"を含む行数）を数え上げてoutput_jp3.txtに記録

評価値を決定
実行結果に差分あり：　"F"（不合格）
条件式の個数が6以下：　"S"（加点対象）
条件式の個数が10以下：　"G"（合格）
それ以外：　"F"（不合格）

図11　特殊な自動採点スクリプトの例
（含まれる条件式の個数で評価値を変更する場合の抜粋）

5.5.2　大量のデータ入力を必要とする場合

例えば，大量のデータ入力を必要とするアプリケーションを想定した場合，ファイルからデータを入力する方が効率的であるその実装方法としては，プログラム内で特定のファイルを参照する固定的な形，実行時の引数でファイル名を指定する形，実行中に標準入力でファイル名を指定する形が挙げられる。しかし，これらのような課題は標準的な自動採点スクリプトでは対応できない。

ファイルでデータを指定する場合の自動採点スクリプトの例をまとめて図12

特殊な自動採点スクリプトの例（ファイルでデータを指定する場合の抜粋）図12

（a）プログラム内で特定のファイル（「data.txt」）を参照する固定的な形

```
14 for f_i in ${d_sh}/input*.dat
15 do
16         cp ${f_i} data.txt
17         f_o=$(basename ${f_i/input/output})
18         java -jar a.jar | sed -r 's/¥x1b¥[[0-9;]*m//g' > ${f_o}
19         if [ -s ${d_sh}/${f_o} ]; then
20                 flag_k=1
21                 diff -dBw --strip-trailing-cr ${f_o} ${d_sh}/${f_o} >> diff_res.log
22         fi
23 done
```

（b）実行時の引数でファイル名（「input*.dat」内にそれぞれ記載）を指定する形

```
14 for f_i in ${d_sh}/input*.dat
15 do
16         f_ii=`cat ${f_i}`
17         cp ${d_sh}/${f_ii} ${f_ii}
18         f_o=$(basename ${f_i/input/output})
19         java -jar a.jar ${f_ii} | sed -r 's/¥x1b¥[[0-9;]*m//g' > ${f_o}
20         if [ -s ${d_sh}/${f_o} ]; then
21                 flag_k=1
22                 diff -dBw --strip-trailing-cr ${f_o} ${d_sh}/${f_o} >> diff_res.log
23         fi
24 done
```

（c）実行中に標準入力でファイル名（「input*.dat」内にそれぞれ記載）を指定する形

```
14 for f_i in ${d_sh}/input*.dat
15 do
16         f_ii=`cat ${f_i}`
17         cp ${d_sh}/${f_ii} ${f_ii}
18         f_o=$(basename ${f_i/input/output})
19         cat ${f_i} | java -jar a.jar | sed -r 's/¥x1b¥[[0-9;]*m//g' > ${f_o}
20         if [ -s ${d_sh}/${f_o} ]; then
21                 flag_k=1
22                 diff -dBw --strip-trailing-cr ${f_o} ${d_sh}/${f_o} >> diff_res.log
23         fi
24 done
```

図12　特殊な自動採点スクリプトの例（ファイルでデータを指定する場合の抜粋）

に示す。それぞれ，(a) プログラム内で特定のファイルを参照する固定的な形，(b) 実行時の引数でファイル名を指定する形，(c) 実行中に標準入力でファイル名を指定する形であり，図7に示したスクリプトの14〜22行目を置き換えるものである。

(a) では，タイプ2標準でも用いる入力用ファイル「input*.txt」を，JARの実行前にプログラム内で固定的に参照するファイル名（図の例では「data.txt」）に書き換えつつ実行フォルダにコピーしている。その代わり，実行時の標準入力にファイルの中身を流し込まないようにしている。

(b) では，「input*.txt」内に書かれたファイル名をあらかじめ取り出しておき，それをもとに課題バンクフォルダ内のファイルをJARの実行前に実行フォルダにコピーしている。その代わり，実行時の標準入力にファイルの中身を流し込まないようにするとともに，実行時引数にファイル名を指定している。

(c) では，「input*.txt」内に書かれたファイル名をあらかじめ取り出しておき，それをもとに課題バンクフォルダ内のファイルをJARの実行前に実行フォルダにコピーしている。(b) (c) と異なり，実行時の標準入力にファイルの中身を流し込んでいる。

注意点としては，セキュリティ上の制約から，実行フォルダの下にファイルを配置する必要があり，相対パスでのみファイル名を指定する必要がある。また，ファイルを正しく受け取ることができるよう，課題のプログラムを作成する必要がある。

6. システムの運用状況と受講者の反応

本システムの運用開始が2017年であり，以降，受講者やアシスタントへのインタビューの他，毎回の授業における任意課題の一環として授業への意見・感想を募ってきた。寄せられたものの大半は授業内容や進度，課題の難易度に関するものであったが，システムへの意見や要望もいくつか見られた。統計処理をはじめ分析を行っていなかったため，数値的に示すことができないが，受講者の反応を中心に本システムの運用の状況を述べる。

まず，実行可能JARファイルの作成については，授業回を重ねるごとに，筆者やアシスタントへ質問することが減っていった。時折見られたのが，「ローカルでは正しく実行できるのに自動採点で不合格になる」というもので，大半が提出ファイルの間違い（エクスポート時の起動構成の選択ミスを含む）であったが，特殊な課題などプログラム構造も採点対象としていることが原因の物もあった。その旨と課題の説明文を改めて伝えると，大半は納得し，後に合格を得ていた。いずれにしても，受講者によるロー

カルでの実行が習慣づいており，本システムの目的が達せていることが示唆される。また，受講者がプログラム構造について考えるきっかけにもなっていることが分かる。

受講者が考えるきっかけ，あるいはモチベーション維持のきっかけとしては，加点対象の合格の存在も大きい。加点対象の合格のある課題を出題した際は，「自力で加点がもらえて嬉しかった」「加点がもらえなくて悔しい」「どうすれば加点になるのかヒントが欲しい」という意見が，特に成績上位の受講者から多く寄せられた。また，第4.2節で述べた，加点対象の合格時の花火の演出を喜ぶ受講者も多かった。成績上位者は通常の課題の合格が早く，そのままでは面白みを感じないことが多い。かといって，課題数を単純に増やしたり難易度を高めたりでは，対応しきれない通常レベルの受講者のモチベーションの低下が懸念される。加点対象の合格がそれらを緩和できていると言える。

受講者の理解につながる機能としては，課題の提出締切後に正答例を表示する機能とコメント機能が挙げられる。「締切に間に合いそうにないため，正答例を見て学ぶつもり」という感想が時折見られ，遅刻提出の合格も度々見られた。また，コメント機能はコロナ禍の影響で授業がオンデマンド型学習になった際，双方向に情報量の大きな対面で対応できなくなることから導入した。これらの機能なしでは受講者・課題ごとの細かな対応が難しかったはずである。いずれも，理解しようとする受講者の一助になっていたと言える。

運用中のシステム面の改善としては，課題内容ページにて表示されるソースコードの装飾とクリップボードへのコピー機能の導入も挙げられる。当初は等幅フォントで表示するのみであったが，「Eclipseやインターネット上にある解説記事のように色付けしてほしい」という要望があり，また課題の際にコピー＆ペーストの範囲を間違える受講者が多かったためである。機能の導入により，ソースコードを把握・利用しやすくなったためか，同様のトラブルは大幅に減少した。

評価結果ページに表示する実行結果について，正答例のものと異なる部分を装飾表示する機能も途中から導入した。「実行結果の違いが分かりにくい」という意見に加え，採点対象が入出力のみでないという本システムの特殊性から，実行結果の違いを明確に伝える必要が生じたためである。違いがなければプログラム構造など別の要因を検討しやすくなるのである。

廃止した機能には，出欠の受講者への表示が挙げられる。筆者はグループワークを除いて出席を重視していないが，連絡のあった欠席者のフォローをしつつ，

無断欠席者のフォローを後回しにしたいため，出欠を記録していた。ところが，授業回が進むと出欠に関する成績の問い合わせのような意見が増えてしまった。出欠の受講者への表示はプレッシャーを与え，その解消のための説明機会を増やす要因となるとともに，有用な意見・感想を得る機会を減らす要因にもなる。よって，出欠は授業担当者のみが把握できる形とした。

これまでに述べた様々な状況から，本システム自体は目的を十分に達成していると言える。あとは課題・資料の作成と授業の実施といった，授業担当者のマターであるが，もう少しの機能拡張を考えると，課題の作成を補助する機能が挙げられる。現行では，メンテナンスの容易さから，サーバのフォルダ・ファイルを直接編集あるいはそれに準じる形をとっている。ブラウザ上から安全に課題を作成・編集する機能など，検討してみたい。ただし，近年はJavaやJavaFXの導入やメンテナンスに関して難が生じていることから，来年度は使用プログラミング言語を変更する予定である。まずは同様の目的を果たせるよう，新たなシステムを構築していきたい。

7. まとめ

本稿では，ローカルの開発環境にて作成された実行可能JARファイルを提出させることを前提とした，自動採点機能を有するプログラミング実習用システムの設計・構成や運用の状況を説明した。実行可能JARファイルを提出させる意義に始まり，外部･内部のシステム設計，自動採点スクリプトについて，抜粋しながら詳細に示した。また，受講者からの反応を中心に，システムの運用状況についても示し，本システム自体は目的を十分に達成していることも示した。

今後の課題として，来年度は使用プログラミング言語を変更する予定であるため，まずは同様の目的を果たせるような新たなシステムを構築していきたい。また，受講者の学習効率を向上させるようなさらなる工夫を検討していきたい。

参考文献

1）中島秀樹，宮地恵佑，髙橋直久：プログラミング演習支援システムCAPESのための答案評価機構の実現，情報処理学会研究報告，2006（16），pp.127-134，2006。

2）中村慎司，筧捷彦：プログラミング授業支援システムWOJの開発，情報処理学会第77回全国大会講演論文集2015（1），pp.939-940，2015。

3）Aizu Online Judge, https://judge.u-aizu.ac.jp

4）Progate, https://prog-8.com

5）paiza, https://paiza.jp

6）AtCoder, https://atcoder.jp

（URL閲覧日は全て2021年11月23日）