

初学者向けデバッガDENOの利用実態の分析

袴田 大貴^{1,a)} 松澤 芳昭^{1,b)} 太田 剛^{1,c)}

概要：本研究の仮説は、初学者がプログラムの実行動作を理解するのにデバッガが有用だということである。既存のデバッガはプロフェッショナルのために開発されたものであるため、初学者にとって利用法習得における負荷が大きいという問題がある。本研究では、プログラムの実行動作理解を支援する初学者向けデバッガ「DENO」を開発した。DENOの特徴は（１）省略可能なブレークポイント、（２）ワンボタンステップ実行である。これを文科系大学生向けプログラミング入門授業で学生 100 名に対して導入した。利用は強制せず、DENO を解説や指導に利用するに留めることで、学生が自発的に DENO を利用するようになることが期待された。授業中の様子を観察するとともに、学習者のエディタ操作記録を用いて利用状況の分析を行った。結果、回を追う毎に利用者は増加し、最終的に受講者の約 7 割が 1 度以上 DENO を利用した。デバッグのためだけでなく、プログラムの動作を理解するため、他の学習者に説明するためにも利用されていることが確認できた。

1. はじめに

プログラミング教育において重要なことの 1 つはプログラムの実行動作を理解することである。プログラミング初学者はプログラムの実行動作理解が不十分であるため、アルゴリズムが理解できているのにも関わらず、それを実現するコードを実装できないという事態が発生する。今泉らは、初学者は制御構造、関数を含んだプログラムの適切な実行動作イメージが作成できないと述べている [1]。

初学者がプログラムの実行動作を理解するためには、デバッガが有用であるというのが本研究の仮説である。山本らはプログラムの実行動作の理解が難しい原因をプログラムの動きが目に見えないことにありと述べている [2]。西田らはプログラムの実行状況を観察できるようにすることでプログラムの動作を理解しやすくなると述べている [3]。

ところが既存のデバッガには、初学者にとって利用法習得における負荷が大きいという問題がある。なぜならば既存のデバッガはプロフェッショナルのために開発されているからである。例として、デバッガの機能の 1 つである「ブレークポイント」は、初学者が適切に設定するのは難しいと言われている [4]。DENO はこの問題を解決するために開発した。

本稿では我々が行なっている文科系大学生向けプログラ

ミング入門授業で DENO を使わせてみたときの利用実態を報告し、得られた知見について述べる。

2. DENO

2.1 DENO とは

まず「DENO[5]」の概要を述べる。対象言語は Java である。DENO は JDK に同梱されている JDI(Java Debug Interface) のデモである簡易 GUI デバッガ「javadt」を基盤として開発し、我々が Java 教育に用いている学習環境「RonproEditor」に組み込んだ。

DENO は初学者にとって使いやすいインタフェースを目標としている。多様な機能を持った複雑なインタフェースは、初学者にとって利用法習得の負荷が大きく、好ましいとは言えないからである。

DENO の動作画面を図 1 に示す。DENO は 4 つの部分から構成されている。上段右の 1 は「操作パネル」である。ここからステップ実行などの操作を行う。中段左の 2 は「ソースコードビュー」である。対象プログラムのソースコードが表示され、現在の停止位置に線が引かれる。中段右の 3 は「変数ビュー」である。その時点で生存している全ての変数が表示される。実行中のスコープ内の変数はハイライト表示される。下段は「コンソールビュー」である。プログラムの標準出力はこのフィールド（４）に出力される。プログラムへの標準入力はこのフィールド下部にある入力欄（５）から行う。

¹ 静岡大学情報学研究科
Graduate School of Informatics, Shizuoka University

a) gs12029@s.inf.shizuoka.ac.jp

b) matsuzawa@inf.shizuoka.ac.jp

c) ohta@inf.shizuoka.ac.jp

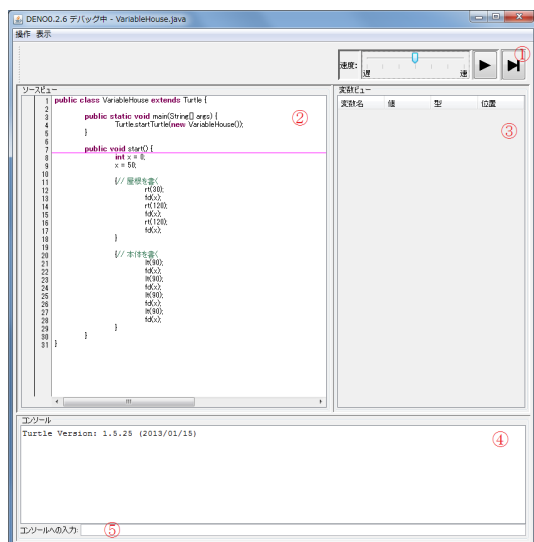


図 1 DENO のインタフェース

2.2 DENO の特徴

2.2.1 省略可能なブレークポイント

DENO はブレークポイントを使用せずともデバッグが可能のように設計した。吉村らの研究 [4] によると、ブレークポイントはプログラムの実行動作を理解していなければ適切に設定することができず、初学者にとっては困難である。

そこでブレークポイントを設定する方式ではなく、デバッグ起動時にプログラム先頭で停止し、以降はステップ実行のみでプログラム終了まで動作する方式にした。これによって、利用者はブレークポイントの設定位置に悩むことなく、デバッグを行うことができる。

スライダーで指定した速度で、自動ステップ実行する機能を実装した。これはプログラムの動きを観察しやすくすることを意図している。

従来のブレークポイントを設定する機能を実装はしてあるが、学習者に対して使用を許可していない。

2.2.2 ワンボタンステップ実行

DENO はステップ実行の使い分けを意識せずともデバッグが可能のように設計した。一般的なデバッガにはステップイン、ステップオーバー、ステップアウトという 3 種のステップ実行機能がある。これを状況に応じて使い分ける必要があり、初学者にとっては難しい。

DENO ではステップ実行 3 種を統合し自動で使い分けられるようにした。これによって、利用者が使い分けを意識する必要はなくなった。

自動使い分けの基準は、対象のメソッドがユーザの自作メソッドであるか否かである。これによってユーザ自身が作成したコードにのみ集中することができる。

ステップアウトは不要だと考えた。学生が作成する課題では高々 5 ステップ程度のメソッドしか実装しないためである。

表 1 実験の実施環境

対象学部	情報学部情報社会学科 (文科系)
対象授業	プログラミング (第 3-11 回)
学年と受講者数	1 年生 : 100 名
授業時間	90 分 × 2 コマ
指導体制	教員 2 名, アシスタント 8 名

3. 実験

3.1 目的と仮説

実験目的は、提案システムについて、強制しなくとも学習者が利用するようになるかどうか確認すること、利用実態を分析することである。

我々が立てた仮説を以下に示す。

仮説 1 適切な機能・インタフェースであれば、強制せずとも、学習者は自然に利用するようになる

仮説 2 初学者はデバッグのためだけでなく、プログラムの動作を観察するためにも DENO を利用する

仮説 3 プログラムの実行動作が分かり辛い課題では、DENO 利用が増える

3.2 実施環境

著者所属学部で 2013 年度後期に行われている、文科系の学生を対象にした授業「プログラミング」の第 3-11 回において実験を行った。実験環境の詳細を表 1 に示す。この授業は 100 名の学生が一教室で同時に受講しており、教員 2 名とティーチングアシスタント 8 名が配置されている。

授業開始から 1 時間程は教員による講義が行われる。講義終了後は課題に取り組む時間である。課題の進度次第では教員から補足説明が入ることもある。

課題の実装を終えた学生から、ティーチングアシスタントを呼んでチェックをしてもらう。授業時間内に終わらなかった課題は宿題として各自実装する必要がある。課題の提出期限は原則翌週の授業前日 0 時までである。

授業内容を表 2 に示す。第 1-4 回の授業では、タートルグラフィックスを利用した図形描画、第 5-6 回の授業でアニメーション作品の制作、第 7-8 回ではコンソールアプリケーションについて学ぶ。後半はメソッド、集合データ構造、アルゴリズムについて学習する。中間課題は第 6 回にて告知された。提出期限は第 7 回と同じである。

プログラミング言語は Java を用いている。ビジュアル言語とテキスト型言語 (Java) の相互変換ができる「BlockEditor[6]」も併用できる環境が与えられている。

3.3 DENO の導入

本実験では、強制せずとも学習者が自然に DENO を利用することが期待されている。従って、DENO の利用を強制する課題は実施していない。

表 2 授業内容

授業回	対象	内容
第 1 回		環境設定
第 2 回		タートルグラフィックスの基本的な命令
第 3 回	○	変数, 条件分岐
第 4 回	○	繰り返し, ブロックの入れ子
第 5 回	○	アニメーション作品の制作
第 6 回	○	アニメーション作品の制作
第 7 回	○	コンソールアプリケーション
中間課題		自由作品
第 8 回	○	コンソールアプリケーション
第 9 回	○	メソッド (引数なし, 引数あり)
第 10 回	○	メソッド (戻り値あり)
第 11 回	○	再帰メソッド
第 12 回		集合データ構造
第 13 回		並び替えアルゴリズム
第 14 回		辞書アルゴリズム
第 15 回		最終作品の発表会

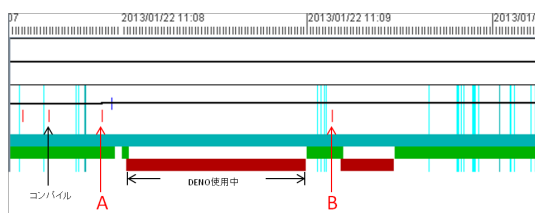


図 2 操作ログを視覚化したタイムラインの例

学習者に対する DENO の存在の周知および利用方法の説明は、講義にとりあげた例題プログラムの動きを教員が解説するときや、ティーチングアシスタントによる課題プログラムの実装実演に用いることで行った。授業時間内に学習者から課題に関する質問を受けた際、ティーチングアシスタントが DENO を使うのに適したケースと判断した場合は、質問への解答とともに DENO を使うことでそれが容易に実現できることを示唆するようにした。

導入はトレースの必要性が生じる「変数・条件分岐」を扱う第 3 回から行った。

3.4 データの採取

開発環境付属の操作記録保存機能を用いてプログラミングの過程の記録を行った。加えて、筆頭筆者自身がティーチングアシスタントとして授業中に指導を行いながら、学習者の様子を観察した。

操作記録に基づき、筆頭筆者が Programming Process Visualizer[7] を用いて受講者の作業プロセスを分析した。Programming Process Visualizer によって操作ログを視覚化したタイムラインの例を図 2 に示す。中央下部の焦げ茶色の長方形が DENO を利用している時間を表している。矢印で示した赤線はコンパイルしたことを表している。水色の線はソースコードを編集したことを表している。この例では、コンパイル (A) して DENO を利用後、ソースコー

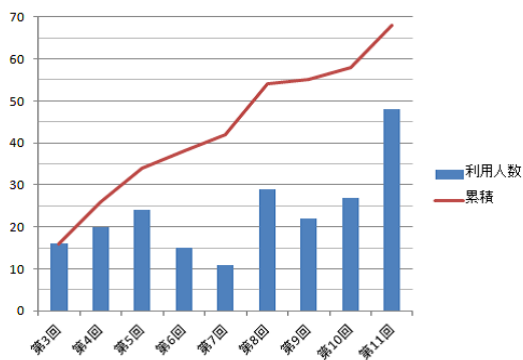


図 3 DENO 利用人数の推移

ドを編集し、再びコンパイル (B) して DENO を利用していることが読み取れる。

4. 結果

4.1 利用人数の推移

各回毎の利用人数の推移を図 3 に示す。減少する回もあるが、回を追うごとに DENO 利用者が増加していくことが分かる。

第 6-7 回に利用人数が減少した理由として、中間課題の存在が挙げられる。該当回では中間課題へ注力させるために課題の数を減らして授業時間内に中間課題のための作業時間を用意している。課題の内容自体も簡単なものになっている。課題の詳細は 4.4 節で述べる。

第 8 回における利用人数の急激な増加の理由は課題がコンソールアプリケーションになったことが挙げられる。タートルグラフィックスでは、亀の動きを見ることで DENO を用いずともプログラムの簡易なトレースを行うことができる。対して、コンソールアプリケーションでは DENO を用いたりデバッグプリントを行わなければプログラムのトレースをするのは難しい。この違いが利用者急増の原因だと考える。実際にタートルグラフィックスを扱う課題に戻った第 9 回では利用者が減少したのに対して、コンソールアプリケーションを扱う課題となった第 10 回では再び利用者が増加している。

最終的に、第 3-11 回の授業において 67 人が 1 度以上 DENO を利用した。これは受講者の約 7 割であり、評価すべき点だと考えている。

4.2 著者主観による DENO 利用者に見られた特徴

筆頭筆者を含めたティーチングアシスタントが学習者を観察した結果から DENO 利用者に見られた特徴を分析した。

DENO 利用者が別の学習者に対して、自身のプログラムを DENO を使って解説したり、DENO を使うことを促したりといった行動が観察された。学習者が別の学習者に対してプログラムを解説するために DENO を使うというの

は、プログラムの実行動作理解を支援を目的とした DENO の使い方としては望ましい使い方である。この行動は回を追うごとに DENO 利用者が増えていった要因の一つでもある。

4.3 作業プロセスに基づく DENO の使い方

DENO 利用者の DENO の使い方を調べた結果、以下の 2 つの傾向が見られた。

- (1) サンプルプログラムや実装を終えたプログラムに対して DENO を利用する
- (2) 実装中のプログラムに対して利用し、その後ソースコードの編集を行う

(1) の使い方は、プログラムの動作に対する理解を深めるためのものと考えられる。これはサンプルプログラムや実装を終えたプログラムといったデバッグを行う必要のないプログラムに対して DENO を利用していることから分かる。授業を観察した結果、サンプルプログラムやティーチングアシスタントがチェックを終えたプログラムの動きを、DENO の自動ステップ実行を活用して観察している学生が複数確認された。他の使い方として、前節で述べた他の学習者への解説に利用していることも観察された。

(2) の使い方は、デバッグとしての使い方である。ただし、課題が難航している場合は教員による解説が行われること、ティーチングアシスタントや他の学習者に質問できることが影響したためか、この使い方をしたケースは (1) に比べて少ない。

4.4 各回毎の利用実態

4.4.1 第 3 回:変数, 条件分岐

第 3 回授業における、各課題の DENO 利用人数を図 4 に示す。

第 3 回はタートルグラフィックスを用いた課題である。課題は以下の 5 つである。

Pentagon2 ユーザの入力を受け取り、その長さを一辺の長さとして正五角形を描く

Star2 長さに乱数を使い、実行するたびに大きさが変わる星を描く

FigureShop 前回作ったプログラムを改造して、ユーザの入力に応じて、2 種類以上の絵を描き分ける

Omikuji 乱数と条件分岐を使って、「大吉」「中吉」「小吉」「凶」を表示する。大吉のときだけ、絵を描く

Grade ユーザが評点を入力すると、「秀」「優」「良」「可」「不可」を表示する

導入されたばかりの DENO 利用者が受講者の 16 % 存在したことは、評価すべき点だと考えている。学習者は未だプログラミングに不慣れで、開発環境である RonproEditor

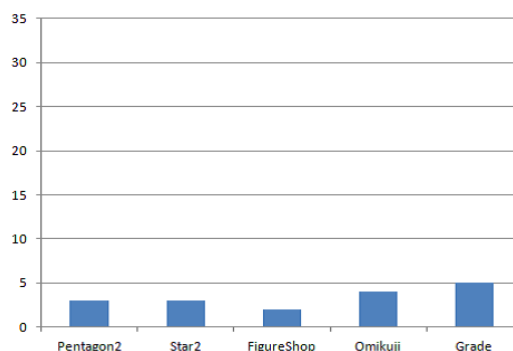


図 4 第 3 回の各課題毎の DENO 利用人数

も使い始めたばかりという学習における負荷が大きい状態で利用しているからである。導入は、変数に値が格納される様子や条件分岐によって実行される行が変化する様子を DENO を用いて解説するとともに、そうした機能によってプログラムのバグを見つけて修正する様子を実演することで行った。

条件分岐が複雑化する Omikuji, Grade で利用人数が増えている。これは、条件分岐の複雑化によってプログラムの実行動作がイメージしづらくなったことが影響していると考えられる。

DENO の特徴的な使い方として、段階的に DENO を用いて動作を確認している学習者が存在した。Omikuji において各運勢の条件分岐だけを実装した直後に DENO を用いて動作を観察する。その後、大吉の際の絵を描く処理を追加し、再び DENO を用いて動作を確認しているのである。プログラミングにおいて問題を小問題に分割することは重要なことである。DENO を用いて段階的に動作を確認するというのは、好ましい使い方である。

4.4.2 第 4 回:繰り返し, ブロックの入れ子

第 4 回授業における、各課題の DENO 利用人数を図 5 に示す。

第 4 回はタートルグラフィックスを用いた課題である。課題は以下の 7 つである。

Circle サンプルプログラムを、円を一周分描いたら終了するように改良する

Box 繰り返しを使って、四角形を描く

Star3 繰り返しを使って、星を描く

Asterisk 繰り返しを使って、アスタリスクを描く

TenBoxes 四角形を 10 個描く

HundredBoxes 四角形を 100 個描く

HundredBoxes2 四角形と三角形を交互に 100 個描く

1 重ループの課題では 2 人であったのに対して、2 重ループの TenBoxes では 2 倍の 4 人、3 重ループの HundredBoxes では 5.5 倍の 11 人となっている。これはループの入れ子が深くなることで、プログラムの実行動作理解が難しくなっているためである。プログラムの実行動作が分か

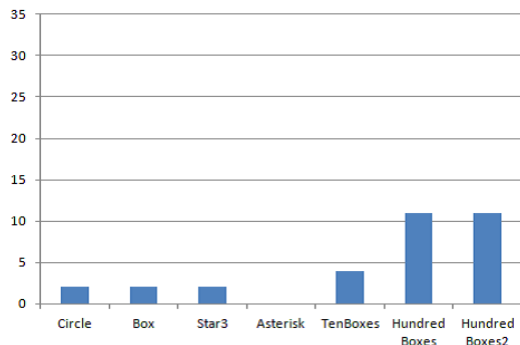


図 5 第 4 回の各課題毎の DENO 利用人数

らないときは DENO を使う，というのは望ましい傾向である。

4.4.3 第 5-6 回:アニメーション作品の制作

第 5-6 回授業における，各課題の DENO 利用人数を図 6，図 7 に示す。

第 5 回はタートルグラフィックスを用いた課題である。課題は以下の 5 つである。

Swastika 4 匹のタートルを同時に動かして，卍を描く

TurtleRace 3 匹のタートルがレースをする．順位は実行毎に変化すること

TurtleRelay 4 匹のタートルでリレーをする

Rotation これまでの課題で作った図形を回転させるアニメーションを作る

ShootingStar 星を描くプログラムを使って，流れ星を描く

ShootingStar で利用人数が増加した原因はこの課題の難易度が高いことが挙げられる。これまでの変数・条件分岐・繰り返し・アニメーションの全てが複合しているのである。この課題では，Star クラスのインスタンスを複数生成した上で，それぞれを回転させながら画面上を移動させる必要がある。加えて，画面端まで移動した際は反対端に戻す必要もある。

この回から，ある時点のソースコードをコピーして，コピー先で実装および DENO を利用した後にコピー元に反映させるといった行動が見られるようになった。これは正常に動作する状態を保存した上で，試行錯誤を行おうとした行動と推察できる。どの状態までが正常なのか判断する，試行錯誤を行って理解を深める，というのは好ましい事例と言える。

第 6 回はタートルグラフィックスを用いた課題である。課題は以下の 2 つである。

Zero655 「たなくじ」または「おれねこ」アニメーションを作る

KeyHandler プレイヤーの画像を用意し，矢印キーを使って，プレイヤーを上下左右に操作する

Zero655 の利用者が多い理由は，今までの課題と異なり

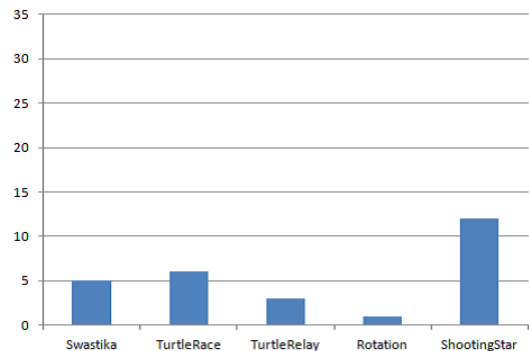


図 6 第 5 回の各課題毎の DENO 利用人数

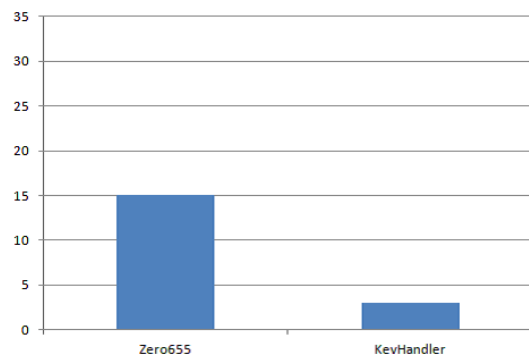


図 7 第 6 回の各課題毎の DENO 利用人数

アニメーションが題材となったこと，画像や音楽といった新たな要素を扱う必要があることが考えられる。アニメーションはパラパラマンガの要領で実装するが，この「1 コマ分の処理を繰り返す」ことに苦労している様子や，表示する画像を切り替えるのに苦労している様子が見受けられた。

KeyHandler では，上と左の操作ができるサンプルプログラムがあるため，利用者が激減している。

4.4.4 第 7-8 回:コンソールアプリケーション

第 7-8 回授業における，各課題の DENO 利用人数を図 8，図 9 に示す。

第 7 回はコンソールアプリケーション課題である。課題は以下の 3 つである。

DoubleChecker1 0.1 を 10 回加算し，その結果を出力する

DoubleChecker2 0.125 を 8 回加算し，その結果を出力する

BillSplit 合計金額を人数で割った数を表示する「割り勘計算機」を作る

DoubleChecker1 の利用人数が 10 人と多い原因は，この課題では計算の結果に誤差が生じることが挙げられる。問題文では誤差が発生することは述べていないため，学習者からは「答えが 1.0 にならなくて悩んだ」といった声が多数得られた。誤差が発生しない DoubleChecker2 は利用人数が激減していることも，この考えを補強している。

DoubleChecker1 における DENO の使い方を操作ログを基に調査したが、誤差が生じることによる変化は見られなかった。0.1 を 10 回加算する際には、計算過程において 3 回目、8 回目、9 回目、10 回目で誤差が生じる。DENO によってプログラムの動作を観察した学習者は、誤差発生時にステップボタンを押す手が止まる、DENO の利用を中断する、といった行動を行うことが予想された。しかしながら、ステップボタンを押す間隔に大きな乱れはなく、プログラムを最後まで実行していた。つまり、10 回加算が行われることと計算結果の確認はしているが、計算過程に誤差が生じていることには気づいていない、もしくは注意を払っていないのである。

第 8 回はコンソールアプリケーション課題である。課題は以下の 6 つである。

RandomGenerator 乱数を使って 10 から 20 の整数をランダムに出力する

Round 小数点第二位を四捨五入する

BMI 身長、体重を入力し、BMI 指数とコメントを出力する

LoveChecker 男女の名前を入力し、恋愛成就率とコメントを出力する

Binary 数当てゲームを作る。誤答の場合、大きい小さいかを表示する

FizzBuzz ある芸人のマネをする。任意の自然数を受け取り、その数だけ数を数える

単純な計算を行うだけの第 7 回から、より複雑な処理を行うようになったため、第 8 回での利用人数は 29 人 (受講者の 3 割) とこれまでの最高値となった。

課題の中では四捨五入を扱う Round、BMI で利用人数が多い。桁移動、加算による繰り上げ、型変換による切り捨てを組み合わせたアルゴリズムが難しいことが要因である。授業中に指導を行ったティーチングアシスタントから「具体的な数を挙げて、計算手順を示しても理解に苦しんでいる学生が多かった」という報告が多数上がっている。四捨五入は 2012 年度以前の授業においても、「学生が非常に苦労していた」と記録が残っている。

Binary の利用人数が多い要因としては、DENO を使って答えが予め分かることでテストが容易になることが挙げられる。

4.4.5 第 9 回:メソッド (引数なし、引数あり)

第 9 回授業における、各課題の DENO 利用人数を図 10 に示す。

第 9 回はタートルグラフィックスを用いた課題である。課題は以下の 4 つである。

Circle2 円を描く

TwoHouses 大きさを指定して家を描く

Flower 花を描く (STEP1:円弧を描く、STEP2:レモンを描く)

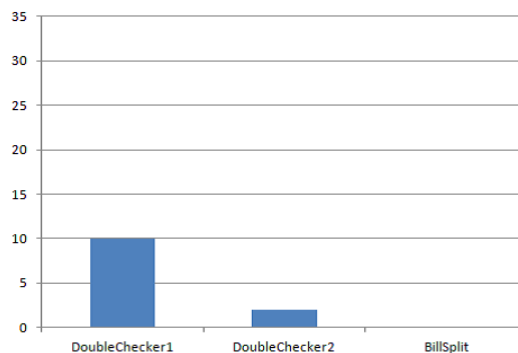


図 8 第 7 回の各課題毎の DENO 利用人数

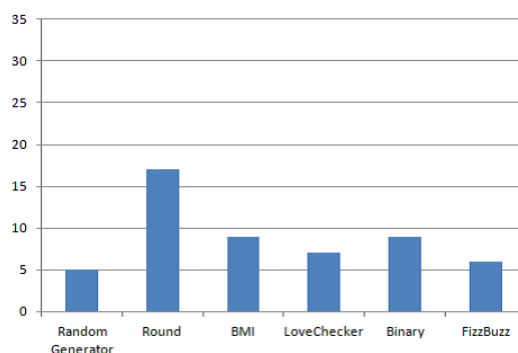


図 9 第 8 回の各課題毎の DENO 利用人数

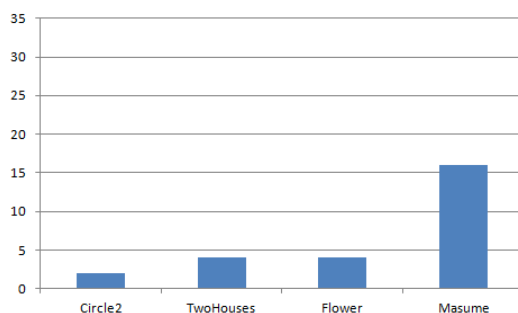


図 10 第 9 回の各課題毎の DENO 利用人数

Masume x 行 y 列のマスを描く

タートルグラフィックスを用いた課題に戻ったため、第 9 回の利用人数は第 8 回に比べて減少した。

4 つの課題の中で特に Masume における利用人数が多い。この課題は第 4 回の HundredBoxes の発展課題であり、学習者が多重ループを苦手としていることが伺える。

4.4.6 第 10 回:メソッド (戻り値あり)

第 10 回授業における、各課題の DENO 利用人数を図 11 に示す。

第 10 回はコンソールアプリケーション課題である。課題は以下の 6 つである。

HowOld 生まれ年を入力して、今年の年齢を出力する

ConsumptionTax 税抜き価格を入力して、税込み価格を出力する

WinningRate 勝数・敗数を入力して、勝率を出力する

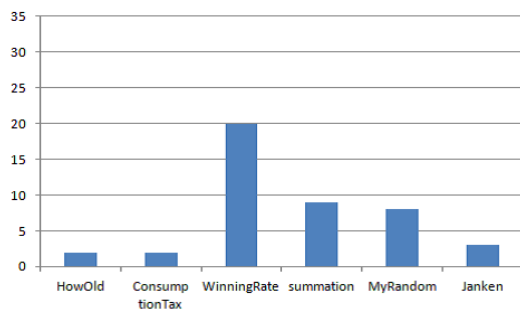


図 11 第 10 回の各課題毎の DENO 利用人数

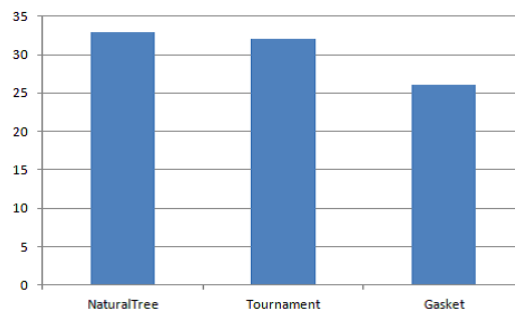


図 12 第 11 回の各課題毎の DENO 利用人数

Summation 正整数 n を入力して、1 から n までの総和を出力する

MyRandom 最小値と最大値を指定して乱数を生成するメソッドを作る

Janken じゃんけんの判定をする

再びコンソールアプリケーション課題となり、第 9 回に比べて利用人数は増加した。

6 つの課題の中では WinningRate での利用人数が特に多い。これは、整数同士の除算では型変換を行わなければ小数点以下の値が切り捨てられてしまうことが要因だと考えられる。これを裏付けるものとして「式は正しいはずなのに、答えが合わない」といった声が多数確認された。

4.4.7 第 11 回:再帰メソッド

第 11 回授業における、各課題の DENO 利用人数を図 12 に示す。

第 11 回はタートルグラフィックスを用いた課題である。課題は以下の 3 つである。

NaturalTree 木を描くサンプルプログラムを改造して、自然な木を描く

Tournament トーナメント表を描く

Gasket ギャスケットを描く

難しいと言われることの多い「再帰」を扱った回であり、第 11 回の利用人数は受講者の約 5 割 (48 人) となった。いずれの課題でも受講者の約 3 割が DENO を利用している。初学者にとって再帰の理解が困難であることが伺える。

Gasket の利用人数が少ない要因としては、再帰に対する「慣れ」の他に、授業時間以降に他者に教えてもらいながら実装を行ったことが考えられる。授業時間内に Gasket の実装を終えた学生は受講者の 1 割に満たない。

この回ではサンプルプログラムに対して DENO を利用して動作を確認する行動が見られた。サンプルプログラムを実行してみるだけでなく、DENO を用いていることから、DENO が理解の助けになると学習者からも認識されていることが伺える。

4.5 課題の進行度に見られた学習者の特徴

各回の課題を授業中に終えた者の上位 10 人に DENO 利

表 3 課題早期回答者に含まれる DENO 利用者数の割合

第 3 回	30 %	第 7 回	70 %
第 4 回	40 %	第 8 回	71 %
第 5 回	60 %	第 9 回	70 %
第 6 回	40 %	第 10 回	60 %

用者が何人含まれていたかを表 3 に示す。第 8 回、第 10 回は授業中に課題を終えた者が 10 人に満たず、それぞれ 7 人、5 人に対する割合である。DENO 利用者とは、該当回までに DENO を 1 度以上利用した者を指す。

表 3 から、課題を早期に終える者には DENO 利用者が多いことが分かる。導入初期である 3-4 回、中間課題のためにアニメーションという特殊な回となった 6 回を除いて、いずれの回でも DENO 利用者が 50 %を超えている。これは DENO によってプログラムの実行動作に関する理解が深まった結果と考えられる。

5. 考察

5.1 仮説 1:適切な機能・インタフェースであれば、強制せずとも、学習者は自然に利用するようになる

DENO の利用を強制する課題は無かったにも関わらず、第 3 回の 16 人に対して第 11 回では 48 人と利用人数は約 4 倍に増加した。これは仮説 1 を支持する結果である。

第 3-11 回において受講者の約 7 割である 67 人が 1 度以上 DENO を利用した。これは DENO の省略可能なブレークポイント、ワンボタンステップ実行といった初学者に配慮した機能が効果的に働いた結果と言える。

第 6 回、第 7 回、第 9 回ではそれぞれ 1 つ前の回に比べて利用人数が減少している。しかしながら、第 6-7 回は中間課題のために課題数が少なく難易度も容易に設定された回であること、第 9 回はプログラムの動きが分かり辛いコンソールアプリケーション課題からプログラムの動きが分かり易いタートルグラフィックスを用いた課題に戻ったために DENO を使う必要性が薄れたこと、といった DENO によるものではなくカリキュラムの影響による減少であるため問題は無いと考えている。

DENO 利用者でない者に見られた好ましくない傾向として、テストをしないということが観察された。ティーチ

ングアシスタントにチェックをしてもらう段階で初めてプログラムを実行するのである。プログラムの出力を見てもそれが課題の解答として適切なものか判別がつかない学習者の存在も確認された。これらは DENO を利用してプログラムの動きを観察する、デバッグする以前の問題である。テストの重要性を周知徹底することが必要だと考える。DoubleChecker1 のように学習者からは出力の予想が難しい課題では、出力例の提示が必要だろう。

5.2 仮説2：初学者はデバッグのためだけでなく、プログラムの動作を観察するためにも DENO を利用する

DENO はデバッグのためよりも、プログラムの動作を観察するためにより多く使われたという結果が得られた。これは仮説2を支持する結果である。

この結果は本研究の仮説「初学者がプログラムの実行動作を理解するのにデバッグが有用である」を支持する結果と言える。後半の授業において、DENO を用いて動きを解説し「ここからうまく動かなくなるが、どうすればよいか」といった内容の質問が見られるようになったこと、DENO 利用者が他の学習者に DENO を用いて説明を行う様子が観察されたことも、これを支持する結果である。

5.3 仮説3：プログラムの実行動作が分かり辛い課題では、DENO 利用が増える

以下に示す要素が含まれた課題では他の課題に比べて DENO の利用人数が多かった。これらの課題は、デバッグやデバッグプリントを行わなければ値の確認ができなかったり、構造が複雑なためトレースが難しい課題である。この結果は、仮説3を支持する結果と言える。

- 多重ループ
- 制御構文の複合
- 四捨五入
- 型変換の必要な演算
- 再帰

6. 先行研究

一般的なデバッグの例として Eclipse JDT が挙げられる。Java のデファクトスタンダード IDE「Eclipse」に搭載されたデバッグであり、ブレークポイントを基にステップ実行や変数値参照などの機能が GUI で提供されている。しかし、1章で述べたように初学者が利用するには負担が大きい。例として、Debug ビュー・BreakPoint ビュー・Outline ビューなど情報量過多、ブレークポイントの適切な設定、ステップ実行の適切な使い分けが挙げられる。

初学者用プログラミング学習環境として西田らの PEN[3]がある。大学入試センター入試科目で用いられている手続記述言語 DNCL を拡張した xDNCL を用いた学習環境である。入力支援によって文法的なエラーが発生しないとい

う特徴を持つ。プログラムの実行状態表示機能を持つが、当該機能単体の評価がなされていないことが問題点である。

デバッグ支援システムとして江木らの DESUS[8]がある。システムによってデバッグプリントによるトレース指導が行われるものである。しかし、デバッグを用いることでデバッグプリント無しでトレースを行うことができる。

7. まとめ

プログラムの実行動作理解支援を目的として初学者向けデバッグ DENO を開発し、実際の演習における利用状況を分析した。得られた知見は以下の3つである。

- (1) 利用を強制せずとも利用者は増加し、少なくとも1度以上利用した受講者は約7割を数えた。これは初学者にとって DENO が十分利用可能なものだったからである。
- (2) 学習者が他の学習者に解説するために DENO が利用されたのは、DENO がプログラムの実行動作理解に寄与した結果である。
- (3) 初学者はプログラムの実行動作理解を苦手としており、DENO を使うことでそれを補うことができる。

DENO が初学者にとって十分利用可能なものであり、プログラムの実行動作理解に寄与したという知見は、本研究の仮説を支持する結果である。

参考文献

- [1] 今泉俊幸, 橋浦弘明, 松浦佐江子, 古宮誠一: ブロック構造の可視化によるプログラミング学習支援環境 azur 〜関数の動作の可視化〜, 電子情報通信学会技術研究報告. ET, 教育学, Vol. 109, No. 193, pp. 45-50 (2009).
- [2] 山本義一, 辻野嘉宏, 都倉信樹: プログラミング教育を目的としたチャート型言語システム, 電子情報通信学会技術研究報告. ET, 教育学, Vol. 94, No. 425, pp. 49-56 (1994).
- [3] 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. 44, No. 8, pp. 2736-2747 (2007).
- [4] 吉村巧朗, 亀井靖高, 上野秀剛, 門田暁人, 松本健一: ブレークポイント使用履歴に基づくデバッグ行動の分析, 電子情報通信学会技術研究報告: 信学技報, Vol. 109, No. 307, pp. 85-90 (2009).
- [5] 袴田大貴, 松澤芳昭, 太田 剛: 初学者向けデバッグ DENO の設計とアルゴリズム構築能力育成授業への適用効果, 情報教育シンポジウム (SSS2013), pp. 197-202 (2013).
- [6] 松澤芳昭, 酒井三四郎: ビジュアル型言語とテキスト記述型言語の併用によるプログラミング入門教育の試みと成果, 情報処理学会研究報告. コンピュータと教育 (CE), Vol. 119, No. 2, pp. 1-11 (2013).
- [7] 松澤芳昭, 岡田 健, 酒井三四郎: Programming Process Visualizer: プログラミングプロセス学習を可能にするプロセス観察ツールの提案, 情報教育シンポジウム (SSS2012), Vol. 2012, No. 4, pp. 257-264 (2012).
- [8] 江木鶴子, 竹内 章: プログラミング初心者にはトレースを指導するデバッグ支援システムの開発と評価, 日本教育工学会論文誌, Vol. 32, No. 4, pp. 369-381 (2009).