

仕様をもとに入力と期待される結果を整理する機能をもった 初学者向けプログラミング学習支援*

片桐 健伍[†]

静岡大学大学院 総合科学技術研究科

酒井 三四郎[‡]

静岡大学 情報学部

1 はじめに

初学者がプログラミングを学習するにあたっていくつかの課題がある。例えば、プログラミング初学者においては、関数(メソッド)の引数に対して期待する結果を十分に予測できないという課題が考えられる。この予測を十分に行うことなくメソッドの実装を行い、結果的に予期せぬ出力をするメソッドを実装してしまう。これについては、入力値と期待される結果を仕様などの情報から整理し、それを基にして開発を行うことで回避できる問題ではないかと考えた。しかし、初学者が入力値と期待する結果を整理し、コードと一致していることを自力で確認することは難しいと考え、学習者自身が仕様を基にして整理した入力値と期待する結果がコードと一致しているかを容易に確認できる初学者向けのコーディング支援ツール *neko* を開発する。

2 先行研究

本研究に類似している手法として、テスト駆動開発[1]が挙げられる。この手法は、まず単体テストを作成してプログラミングを行うことで効率よく開発を行うというものである。本研究においてもまず仕様を整理して成功条件を作成することで、この手法に類似した開発を行うことを期待している。

本研究の先行例として、入力値と期待される結果の組み合わせからプログラミング学習を行うという点で CodingBat[2]が挙げられる。CodingBatでは、入力値と期待される結果の組み合わせがいくつか設定され、それをすべて満たすようなコードを実装することでコーディングを学習する。本研究ではこの手法によってドリルのようにコーディングを学習する方法に加えて、使用者が自ら入力値と期待される結果の組み合わせを考慮することができる。これを使用者自身が考えることで仕様の整理や入力値と期待される出力値の整理を促し、正確なコーディング学習を支援できると考えている。

3 提案ツール

本研究では、1節で挙げた問題を解決し、初学者のプログラミング学習を支援するためのツール *neko* を

*Learning support for programming beginner with functions to organize inputs and expected results based on specifications

[†]Kengo KATAGIRI, Graduate School of Integrated Science and Technology, Shizuoka University

[‡]Sanshiro SAKAI, Faculty of Informatics, Shizuoka University

開発した。今回開発した *neko* には、次のような機能がある。

- 提示された仕様を学習者自身が整理する機能
- 整理した仕様や入力値、期待される出力値と自身のコードが一致しているかを確認する機能

実装したコードに対して成功条件を判定する機能の実現にあたっては、JUnit を使用している。具体的には、成功条件に基づいて単体テストコードを自動で生成し、それを実行する。正常に実行が完了した場合、その出力を整形して使用者に表示する。成功条件は保存して読み込むことができ、これによって課題等をドリルのように解答することができる。これによって、ツールに対する理解や、条件に沿ってプログラミングを行う基礎を学習することを期待している。また、この成功条件は自分で作成することもでき、その後コードを実装すると成功条件の判定も行われる。これらの機能によって、成功条件が実行時に判定される。これによって学習者自身が整理した仕様や入力値、期待される出力値と自身のコードが一致しているか簡単に把握でき、仕様に沿ったコーディングができるようになると期待している。

図1で示した画面では、自身で整理した仕様の概略や、それをもとに設定したクラス名やメソッド名、成功条件を設定するように求められる。設定した各項目は、左側半分に表示され、これを参考にして右側半分にコードを実装する。実装したコードをコンパイル・実行すると成功条件の判定が行われ、その結果が成功条件と一緒に表示される。この他に、整理した仕様のメモを記述する画面や成功条件を設定する画面が用意されている。



図1: *neko* の実行例

4 評価実験

本実験では, neko を使用したほうが使用しないよりも正確なコーディングができるという仮説の検証を行う。

4.1 実験方法

被験者は, 学部1年生4人で, 全員がプログラミングの講義を受講している。本実験では, まず neko を使用することなくコーディング課題を3問行う。その後, 同難易度の別のコーディング課題を neko を使用して3問行う。なお, neko で解答する課題3問のうち, はじめの2問については課題に対する仕様の整理や成功条件の設定がすでに行われおり, それに沿ってコーディングをするように求めた。また, 残りの1問については被験者自身が仕様の整理や成功条件の設定を行うようにした。また, 実験の前後でプログラミングに対する自身の評価等をアンケートによって確認する。実験結果やアンケートの結果から仮説の検証を行い, neko の有効性を評価する。

4.2 実験結果と考察

各課題ごとの neko 未使用・使用における被験者1,2,3,4のコードの正確さを数値化した得点を表1, 表2に示す。なお, 各問題ごとのカッコ内の値はその問題における満点を示している。

結果として, すべての被験者において neko 使用時のほうが点数が高く, 課題で示された仕様に対して正確なコーディングがされていた。一部の被験者は neko 未使用時において, 仕様に対して自身のコードが適切であるかの確認をほとんど行っておらず, コンパイルが成功し出力がなされることだけを確認して次の課題に進んでいた。これによって, neko 未使用時のコーディングの正確性が低下したのではないかと考える。一方で, neko 使用時には成功条件の判定を neko が行っており, それによって仕様から逸脱していることにコーディング中に気づいていた。これによって, コーディングの正確性が保たれていたのではないかと考えた。このことから, 予期せぬ出力を行うコードの実装を抑制する効果はあるのではないかと考えた。

また, 被験者自らが neko を用いて仕様の整理や成功条件の設定を行う課題においては, 被験者全員が1つ以上の成功条件を設定し, それを参考にしてコーディングを行っていた。仕様を読んでそれを整理する過程では, どのような項目に着目するとよいか等のサポー

トを neko に表示しており, それに沿った整理がされていた。neko におけるこの機能を使用する課題は1問のみであったが, 被験者は実験前の5分程度の簡単な説明で機能を適切に使用することができていた。このことから, neko における仕様の整理や成功条件の設定機能は, 簡単に使用でき一定の効果があるのではないかと考えた。

アンケートでは被験者のうち3人が neko は使いやすかったと回答した。その理由として, 「テストプログラムをコード内に書き込まなくていいので便利」「自分が間違っているところが一目瞭然でわかるところが良かった」「どこまでできているか色で判別できたのでわかりやすかった」といった意見があった。一方で, 「成功条件が既定してあるのでやりやすかったがもしかすると既定の成功条件にしか対応しないものができるかもしれない」といった改善案もあった。このことから, 被験者にとって即時にコードの適格性をフィードバックできることやテストコードを実装する作業による負担の軽減やコードの可読性の向上など, プログラミング学習に一定の効果が期待できると考えた。一方で, 改善案の指摘のように, 成功条件設定の拡充など初学者がより学習効果を得られるような改修も必要であると感じた。

5 おわりに

本研究では, プログラミング学習支援として仕様から整理された入力値と期待される結果がコードと一致しているかを容易に確認できる初学者向けのコーディング支援ツール neko を開発し, その効果を検証するための実験を行った。実験の結果, neko において学習者へのフィードバックの即時性やテストコード実装による学習の負担軽減, 学習者自身による仕様や成功条件の整理によるコード正確性の向上など, 一定の効果があることが分かった。今後は, neko の機能や評価実験内容の向上などを通して, さらに有効性の高いツールへと発展させていくことを検討している。

参考文献

- [1] Kent Beck: *Test-driven development: by example*, Addison-Wesley Professional (2003).
- [2] Nick Parlante: *CodingBat*, <https://codingbat.com/java> (2019.01.07).

表 1: neko 未使用時の各課題の得点

ID	1	2	3	4
第1問 (2)	1	0	1	2
第2問 (3)	2	0	3	3
第3問 (1)	0	0	1	1

表 2: neko 使用時の各課題の得点

ID	1	2	3	4
第4問 (2)	2	2	2	2
第5問 (3)	3	3	3	3
第6問 (1)	1	1	1	1