

# Java プログラム理解のための習熟度評価フレームワーク javacefr と その適用実験

奈良井 洸希

岡山大学大学院環境生命自然科学研究科  
parn6hdc@s.okayama-u.ac.jp

稲吉 弘樹

岡山大学学術研究院環境生命自然科学学域  
inayoshi@okayama-u.ac.jp

門田 暁人

岡山大学学術研究院環境生命自然科学学域  
mondenn@okayama-u.ac.jp

## 要旨

本稿では、Java ソースコードを解析することでその理解に必要な習熟度を測定する枠組みである *javacefr* を提案する。習熟度の区分は、自然言語の習熟レベルの評価基準のひとつであるヨーロッパ言語共通レファレンス枠組 *CEFR* に倣って 6 段階とし、プログラムに含まれる各構文要素に対し、その理解に必要な習熟度を割り当てている。また、与えられたプログラムに対して、プログラム中で用いられている構文要素とそれらの理解に必要な *javacefr* の習熟レベルを出力するシステムを開発した。本システムでは、Java プログラムの解析ライブラリである *JavaParser* を用いて抽象構文木を導出し、各構文要素の出現頻度を計測できる。本システムを用いることで、与えられたプログラムについて、習熟が必要な構文とその習熟度が明らかとなることから、プログラム学習に役立つと期待される。ケーススタディとして、*ChatGPT* に指示や問題を与えて生成させたプログラムを解析した。

## 1. はじめに

Java は多くのプラットフォームにおいて利用可能であり、Web サービスから大規模場業務システム、組み込みシステム、スマホアプリに至るまで広く利用されていることから、その効果的な習熟が求められている。一方、近年では *ChatGPT* などの生成 AI が普及しつつあり、

初心者であっても Java プログラムを生成させることが可能である。ただし、生成されたプログラムを利用して開発を進めていくためには、バグへの対応や仕様に沿っている物かどうかの確認の面などからそのプログラムを理解できる必要がある。しかし、AI を利用するプログラミング初心者にとっては、生成されたプログラムがどの程度の難易度でどのような要素で構成されているのかを把握することは容易でない。そこで、本稿では、AI から生成されたプログラムに対し、その理解に必要な習熟度を定量化することを目的とする。従来、Python プログラムを対象とした習熟度の測定フレームワーク *pycefr* [1] があり、本稿ではこれを参考に Java の構文要素に注目した *javacefr* を提案する。

## 2. 関連研究

Robles らは、与えられた Python プログラムの必要習熟度 (competency level) を定量化する枠組みとして、*pycefr* を提案している [1]。 *pycefr* では、解析対象の python プログラムから構文要素を抽出し、各構文要素を理解して使用するために必要な習熟度を算出する。この習熟度は、欧州言語共通参照フレームワーク (CEFR) の設計に従って、A1, A2, B1, B2, C1, C2 の 6 段階としている。CEFR はヨーロッパを中心に使用されている自然言語能力の評価基準で、ヨーロッパ評議会によって 1990 年代に作成された。主にヨーロッパ全ての言語に適用できるような学習状況の評価および指導の方法の提供を目

的としている。自然言語におけるフレームワークである CEFR の利用は「プログラミング言語の学習は、自然言語の学習プロセスのようにいくつかの要素と簡単な構造から学習を始め、経験と学習によって拡張し、より複雑で高度な要素を獲得する」という仮定に基づいている。一つのプログラム中に複数のレベルの構文要素が出現する場合、もっとも高いレベルを、そのプログラムの理解に必要な習熟レベルとみなすことになる。

Robles らはまず、6冊の Python に関する本を調査し、そこで説明されている Python の要素が出現する順番を記録、コードの理解に必要な習熟度の各レベルに割り当てることで pycefr を作成した。次に、著者のうち 2 人が大学で教えているコンピュータネットワーク科目でテストを行った。この科目における学生 33 名にはプログラミングの予備知識があるものの Python の知識はないため、Python の 8 時間短期集中コースが提供される。著者らは pycefr における B1 レベルまでを教えることとし、コース中で学生に 5 つの小さな課題を課した。結果として、学生が使用する Python 要素の大半は A レベルにあり、B1 および B2 の要素は 14 個のみ、C レベルはほとんど見つからなかった。B2 のうちの「if \_\_name\_\_ == '\_\_main\_\_」は、短期集中コースでは紹介しなかったものの、後のレッスンで導入があったため A2 か B1 などに属するはずだった、つまり割り当てが間違っていたことによるものであった。他の B および C レベルの要素については、インターネット検索で見つけた Stack Overflow からのコピーであることが分かった。著者らはこの研究から、pycefr に従って短期集中コースに対応した課題を設計できることを確認している。

### 3. javacefr の提案

javacefr では pycefr と同様に、対象となるプログラムを解析して「そのプログラムを理解するのにどの程度の習熟度が必要か」を測定することを目的としており、Java のそれぞれの構文要素についてあらかじめ各レベル (A1 ~ C2) に割り当てておく必要がある。Java ではその版權を持つ Oracle による certified programmer の認定制度があり、Bronze, Silver, Gold の 3 種類の認定レベルが設定されている。各認定レベルについて認定資格教科書 [2, 3, 4] が出版されており、各レベルの認定試験を受けるのに必要な Java の構文やその使用方法が解説されている。そこで本稿では表 1 の通り、認定資格教科書に

表 1. プログラム理解に必要な習熟度の区分と分類方法

習熟度	分類方法
A1	Bronze にのみ記載
A2	Bronze と Silver(クラス登場前) に記載
B1	Bronze と Silver(クラス登場後) に記載
B2	Silver にのみ記載
C1	Silver と Gold に記載
C2	Gold にのみ記載

基づいて習熟度への割り当てを決定することにした。

また表 1 の「分類方法」に示したように、2 つの異なる認定レベルの教科書にまたがって現れる構文要素が存在する。これは、「Bronze」では基礎的な内容の説明にとどまり、「Silver」ではそれをより詳しく補足している場合などであった。さらに「Silver」では、Java がオブジェクト指向としての特性を発揮する「クラス」が登場する章の以降において、より発展的な内容が説明されている場合があった。そこで、「Bronze」と「Silver」の両方に出現する構文要素については、「Bronze」のみに記載されている要素を A1、「Bronze」と「Silver」の前半(クラス登場前)に記載されている要素を A2、「Bronze」と「Silver」の後半(クラスの登場後)に記載されている要素を B1 としている。

javacefr の策定のためには、Java にはどのような構文要素があり、それぞれ構文木からどのように検出できるかを検討するとともに、それらと理解に必要な習熟度とを対応付ける必要があった。本稿では、構文解析ツール JavaParser[7] を用いて得られる構文要素と理解に必要な習熟度を紐づけた。構文要素は、JavaParser による AST ノードのクラスの名前となる。結果として、A1 に分類される要素は 5 個、A2 は 29 個、B1 は 16 個、B2 は 4 個、C1 は 6 個、C2 は 13 個となった。詳細を表 2 に示す。

## 4. 実験およびその結果

### 4.1 概要

本稿における実験では、生成されたプログラムの理解に必要な習熟度を測定するために、次の手法を用いて解析を行った。

表 2: Java の構文要素と理解に必要な習熟度への割り当て

割り当て先	構文要素	詳細
A1	ArrayType	配列の型
	BinaryExpr	二項演算子による式
	EnclosedExpr	丸括弧で囲まれた式
	LineComment	//形式のコメント
	MethodReferenceExpr	メソッド参照
A2	ArrayAccessExpr	配列の要素へのアクセス
	ArrayCreationExpr	配列の作成
	ArrayInitializerExpr	配列の初期化
	AssignExpr	値の代入
	BlockStmt	ブロック
	BooleanLiteralExpr	真偽値を表すリテラル
	BreakStmt	break 文
	CharLiteralExpr	文字を表すリテラル
	ClassOrInterfaceDeclaration	クラスまたはインタフェースの宣言
	ContinueStmt	continue 文
	DoStmt	do-while 文
	DoubleLiteralExpr	小数点を含む数値を表すリテラル
	ExpressionStmt	式の文としての使用
	ForEachStmt	拡張 for 文
	ForStmt	for 文
	IfStmt	if 文
	IntegerLiteralExpr	整数値を表すリテラル
	LabeledStmt	ラベル付きのステートメント
	LongLiteralExpr	long 型の数値を表すリテラル
	NameExpr	変数やクラスの名前を表す
	NullLiteralExpr	null を表すリテラル
	PrimitiveType	プリミティブ型
	StringLiteralExpr	文字列を表すリテラル
	SwitchEntry	switch 文の各 case や default
	SwitchStmt	switch 文
	UnaryExpr	単項演算子
	VariableDeclarationExpr	変数の宣言と初期化
	VariableDeclarator	個々の変数宣言
WhileStmt	while 文	
B1	CastExpr	キャスト
	ClassOrInterfaceDeclaration	クラスまたはインタフェースの宣言
	ConstructorDeclaration	コンストラクタの宣言
	ExplicitConstructorInvocationStmt	コンストラクタの中で別のコンストラクタを呼び出す
	FieldAccessExpr	変数へのアクセス

	FieldDeclaration	クラス内でのフィールドの宣言
	ImportDeclaration	import 文
	MethodCallExpr	メソッド呼び出しの式
	MethodDeclaration	メソッドの宣言
	ObjectCreationExpr	オブジェクトの作成
	PackageDeclaration	パッケージの宣言
	Parameter	引数
	ReturnStmt	return 文
	SuperExpr	super キーワードを表す式
	ThisExpr	this キーワードを表す式
	VoidType	void 型
B2	ArrayCreationLevel	配列の各次元を表す
	BlockComment	複数行コメント
	InitializerDeclaration	イニシャライザ
	JavadocComment	/** ... */形式のコメント
C1	CatchClause	try-catch 文における例外のキャッチ
	LambdaExpr	ラムダ式
	MarkerAnnotationExpr	値を持たないアノテーション
	ThrowStmt	throw 文
	TryStmt	try-catch 文
	UnionType	複数の型を1つの文脈で扱う型
C2	AnnotationDeclaration	アノテーション宣言
	AnnotationMemberDeclaration	アノテーションのメンバー宣言
	AssertStmt	アサーション
	EnumConstantDeclaration	列挙型の定数の宣言
	EnumDeclaration	列挙型の宣言
	InstanceOfExpr	instanceof キーワードを使用した式
	LocalClassDeclarationStmt	メソッド内で定義されたローカルクラス
	MemberValuePair	アノテーションの属性とその値のペアを表す
	NormalAnnotationExpr	属性を持つアノテーションを表す
	SingleMemberAnnotationExpr	単一の属性値を持つアノテーション
	SynchronizedStmt	synchronized 文
	TypeParameter	型パラメータ
	WildcardType	ワイルドカード型

まず、OpenAI の言語モデルである ChatGPT 4o に対して具体的な課題を与え、プログラムを生成させる。この生成されたプログラムで使用されている構文要素を解析するために、Java プログラムの解析ライブラリである JavaParser を用いて構文解析を行った。近年では生成 AI を利用したコード自動生成が注目されている。その生成結果を分析することで AI がどの程度の「理解に必要な習熟度」を要するコードを生成するかを明らかにし、学習者やエンジニアにとって適切なサポートツールを設計することが可能になる。JavaParser による構文解析の過程では、プログラムから抽象構文木を取得した。AST はプログラムの構造を表現するデータ構造であり、各ノードが特定の構文要素を表す。この AST をもとに、3 章で述べた javacefr に従って各構文要素の頻度をカウントする。

次に、集計結果に基づき、構文要素が 1 つ以上出現したカテゴリの中で最も高い習熟度を識別する。この最も高い習熟度が、そのプログラムを理解するために必要な習熟度となる。具体的には、習熟度が「A1」「B1」「B2」に該当する構文要素が含まれているプログラムの場合、そのプログラムを理解するには「B2」の習熟度が求められると判断するということである [5]。

次節以降では様々なプロンプトを ChatGPT に与えてプログラムを生成させ、生成させたプログラムを解析した結果についてまとめる。

## 4.2. 実験 1

1 つ目の例として、「名簿管理プログラム」を ChatGPT に生成させる。各段階ごとのプロンプトを以下に示す。original code 部分は ChatGPT が直前に生成したコードである。

・ 1 回目  
[指示]  
あなたは Java に精通したプロの Web エンジニアです。下記の [仕様] と [条件] を満たす、名簿を管理するプログラムを作ってください。  
[仕様]  
1. プログラムを実行すると入力を受け付け、名簿データ、またはコマンドを受け付ける。  
2. プログラム実行中は、入力された名簿データを保持し続ける。  
3. 名簿データはコンマ区切りの 5 項目で、ID、氏名 (最大 50 文字)、誕生日 (ハイフン区切りの 3 つの整数)、住所 (最大 50 文字)、備考 (任意の長さ) とする。  
4. 名簿データの誕生日部分に、13 月や 32 日などのあり得ない日付が含まれている場合は、例外であるとしてその名簿デー

タを保持しないようにする。  
5. コマンドは、%Q と入力するとプログラムが終了する。  
[名簿データのサンプル]  
9154,Kaho,2002-04-10,Tokyo,Student  
6487,Takumi.2014-09-14,Aomori,Student and eldest son  
7256,Toru,1982-12-12,Tokyo,Doctor  
0231,Naomi,1994-02-01,Chiba,Actor  
1228,Hanayo,2006-08-05,Tokyo,Student and only child  
[条件]  
1. 各ブロックや行で何が行われているかが理解しやすいよう、具体的で丁寧なコメントを日本語でつけるようにしてください。  
2. Java のコードはファイルに分けず、単一のファイルに含めて貼り付けてください。

・ 2 回目  
[コードの修正]  
コマンドを追加します。%C と入力すると、保持しているデータの件数を出力するようにしてください。  
-----original code-----

・ 3 回目  
[コードの修正]  
コマンドを追加します。%P n と入力すると、保持しているデータを先頭から n 件だけ出力するようにしてください。  
[仕様]  
n は 1 以上の整数で、n が保持しているデータの件数を超える場合は保持しているデータ全てを出力するようにしてください。  
-----original code-----

・ 4 回目  
[コードの修正]  
コマンドを追加します。%R filename と入力すると、filename からデータを読み込むようにしてください。  
-----original code-----

・ 5 回目  
[コードの修正]  
コマンドを追加します。%W filename と入力すると、filename にデータを書き出すようにしてください。  
-----original code-----

このプロンプトにより生成されたプログラムを解析した結果は、表 3 の通りであった。特筆すべき点は、理解に必要な習熟度として「C1」レベルとして「CatchClause」「ThrowStmt」「TryStmt」がみられたことがある。これらが含まれる理由について、名簿管理という機能を実現するうえで予期せぬエラーや入力データの問題に対処し、プログラムの安全性と信頼性を確保する必要があったためであると考えられる。

まず、ファイル操作やユーザー入力にはエラーが発生する可能性が常に存在するため、プログラム全体がクラッシュせずに適切なメッセージをユーザーに提供しない

表 3. 「名簿管理プログラム」の解析結果

習熟度	見つかった構文要素 (個数)
A1	ArrayType(4), BinaryExpr(36), EnclosedExpr(4), LineComment(25)
A2	ArrayAccessExpr(10), ArrayInitializerExpr(1), AssignExpr(3), BlockStmt(1), BooleanLiteralExpr(1), BreakStmt(1), ContinueStmt(5), ExpressionStmt(51), ForEachStmt(2), ForStmt(1), IfStmt(14), IntegerLiteralExpr(43), NameExpr(110), NullLiteralExpr(1), PrimitiveType(13), StringLiteralExpr(33), UnaryExpr(5), VariableDeclarationExpr(25), VariableDeclarator(26), WhileStmt(2)
B1	ClassOrInterfaceDeclaration(1), FieldAccessExpr(19), FieldDeclaration(1), ImportDeclaration(3), MethodCallExpr(72), MethodDeclaration(10), ObjectCreationExpr(14), Parameter(17), ReturnStmt(2), VoidType(8)
B2	
C1	CatchClause(5), ThrowStmt(7), TryStmt(7)
C2	

がら継続的に動作する仕組みが求められる。このような要件を満たすために、例外処理の構文要素が用いられている。また、ファイル読み書きや入力データ処理の実装では例外処理が標準的に含まれるため、自然にこれらの構文要素が採用されたと考えられる。この結果、エラー発生時にも具体的なフィードバックを提供しながら名簿管理プログラムとしての実用性を確保するコードが生成された。

#### 4.3. 実験 2

2つ目の例として、「シューティングゲーム」の Java プログラムを生成させる。シューティングゲームでは、床面を左右に移動可能なオブジェクトがあり、敵オブジェクトが天井のランダムな位置から真っ直ぐに下降してくるものを撃ち落とす。撃ち落とす数をカウントで

きるような状態を最終的な完成とした。ここにおいても同様に [指示] や [仕様] を以下のように段階的に明記して ChatGPT に与えた。

・ 1

[指示]

あなたは Java に精通したプロの Web エンジニアです。下記の [仕様] と [条件] を満たすシューティングゲームを実装してください。

[仕様]

1. 画面の一番下には床があり、画面の左右には壁があります。
2. マウスを左クリックすると床に 1 つあるオブジェクト (味方オブジェクト) から球が生成され、一定の速度で天井まで向かいます。
3. 生成された球は天井に達すると、消滅します。
4. 床にあるオブジェクトは、A キーで左に、D キーで右に移動します。移動可能範囲は壁に挟まれている部分のみです。

[条件]

1. 各ブロックや行で何が行われているかが理解しやすいよう、具体的で丁寧なコメントを日本語でつけるようにしてください。
2. Java のコードはファイルに分けず、単一のファイルに含めて貼り付けてください。

・ 2

[コードの修正]

下記のコードにおいて、[仕様] をみたとすように敵オブジェクトを実装してください。

[仕様]

1. 0.5~2 秒ごとに敵オブジェクトが天井に接するランダムな x 座標に生成されるようにしてください。
2. 敵オブジェクトは天井側から床側に向かってゆっくりと直進するようにしてください。
3. 敵オブジェクトは、床に触れる、または味方オブジェクトが生成した球にあたると消滅するようにしてください。

-----original code-----

・ 3

[コードの修正]

下記のコードにおいて、敵オブジェクトが生成される際に「1.png」というファイルの画像を丸いオブジェクトに貼り付けたい場合はどうすれば良いか教えてください。

「1.png」の画像のサイズは 215px \* 215px です。

-----original code-----

・ 4

[コードの修正]

下記のコードにおいて、敵オブジェクトに貼り付けられるテクスチャ画像を「1.png」から「8.png」の 8 種類に増やしてください。

張り付けられる画像の種類はランダムとします。

-----original code-----

・ 5

[コードの修正]

下記のコードにおいて、味方オブジェクトが生成した球にあたって消滅した敵オブジェクトの数をカウントしてください。

カウントは、天井の右端に半透明で表示してください。

-----original code-----

```

・ 6
[コードの修正]
下記のコードにおいて、[仕様]に従ってカウントに関するコードの修正を行ってください。
[仕様]
味方から生成された球が「1.png」の敵オブジェクトに当たった場合、カウントを1進める。さらに、「2.png」ではカウントを2進め、「3.png」ではカウントを3進める。
4～8でもこのようにカウント数を変更してください。
-----original code-----

```

表 4. 「シューティングゲーム」の解析結果

習熟度	見つかった構文要素 (個数)
A1	ArrayType(1), BinaryExpr(23), EnclosedExpr(), LineComment(22)
A2	AssignExpr(17), BlockStmt(27), BooleanLiteralExpr(3), BreakStmt(1), ContinueStmt(1), ExpressionStmt(62), ForEachStmt(2), ForStmt(3), IfStmt(5), IntegerLiteralExpr(37), NameExpr(116), NullLiteralExpr(2), PrimitiveType(27), StringLiteralExpr(4), UnaryExpr(6), VariableDeclarationExpr(13), VariableDeclarator(35)
B1	ClassOrInterfaceDeclaration(4), ConstructorDeclaration(4), FieldAccessExpr(32), FieldDeclaration(20), ImportDeclaration(5), MethodCallExpr(48), MethodDeclaration(13), ObjectCreationExpr(15), Parameter(17), SuperExpr(1), ThisExpr(9), VoidType(13),
B2	
C1	LambdaExpr(1), MarkerAnnotationExpr(10)
C2	

このプロンプトにより生成されたプログラムを解析した結果は、表 4 の通りであった。特筆すべき点は、理解に必要な習熟度として「C1」レベルとして「LambdaExpr」と「MarkerAnnotationExpr」が見られたことに

ある。「LambdaExpr」は以下のように使用されていた。

```

Timer enemyTimer
= new Timer(random.nextInt(1500)
+ 500, e -> createEnemy());

```

これは敵生成のタイマー処理で、短く直感的にイベントリスナーを記述することが可能となっている。また、アノテーションは @Override が各所で使用されていた。

#### 4.4. 実験 3

3つ目の例として、Codeforces から取り出してきた問題 30 問に対する回答プログラムを生成させる。Codeforces はロシアの競技プログラミングのサイトで、問題にはそれぞれ 800～3500 の範囲で難易度のスコアが割り振られている。これを、800～1249, 1250～1699, 1700～2149, 2150～2599, 2600～3049, 3050～3500 の範囲で 6 分割し、それぞれの範囲から 5 つずつランダムに選出した問題を ChatGPT に与えた。

表 5 は、Codeforces の問題への回答プログラムの解析結果である。プログラム 1 から 5 は Codeforces で難易度 800～1249 の問題への回答プログラム、プログラム 6 から 10 は難易度 1250～1699 の問題への回答プログラム、プログラム 11 から 15 は難易度 1700～2149 の問題への回答プログラム、プログラム 16 から 20 は難易度 2150～2599 の問題への回答プログラム、プログラム 21 から 25 は難易度 2600～3049 の問題への回答プログラム、プログラム 26 から 30 は難易度 3050～3500 の問題への回答プログラムである。問題が異なるため出現する構文要素の違いはあったが、どの難易度においても難しい構文要素とした「C」レベルのものが見られなかった。

表 5. Codeforces の問題への回答プログラムの解析結果

プログラム	A1	A2	B1	B2	C1	C2
プログラム 1	29	112	23	1	0	0
プログラム 2	20	143	16	1	0	0
プログラム 3	14	80	16	0	0	0
プログラム 4	12	73	18	1	0	0
プログラム 5	27	226	24	1	0	0
プログラム 6	43	282	39	6	0	0
プログラム 7	20	121	26	0	0	0
プログラム 8	51	340	82	6	0	0
プログラム 9	11	90	16	1	0	0
プログラム 10	12	74	17	0	0	0
プログラム 11	20	162	27	2	0	0
プログラム 12	29	226	33	4	0	0
プログラム 13	22	151	40	0	0	0
プログラム 14	21	198	49	0	0	0
プログラム 15	10	118	22	0	0	0
プログラム 16	32	238	50	2	0	0
プログラム 17	31	145	31	0	0	0
プログラム 18	11	67	20	0	0	0
プログラム 19	28	213	37	2	0	0
プログラム 20	16	82	16	2	0	0
プログラム 21	15	67	20	0	0	0
プログラム 22	26	193	30	4	0	0
プログラム 23	31	102	28	0	0	0
プログラム 24	42	190	42	1	0	0
プログラム 25	32	224	33	4	0	0
プログラム 26	23	126	18	3	0	0
プログラム 27	49	304	46	3	0	0
プログラム 28	18	138	38	1	0	0
プログラム 29	22	157	23	1	0	0
プログラム 30	31	267	33	6	0	0

このことについて、競技プログラミングの性質と ChatGPT がコードを生成する際の方針が関係していると考えられる。

まず Codeforces をはじめとする競技プログラミングでは、問題の回答において最も重要視されるのはコードの実行効率や正確性であり、時間内に回答を完成させるためにコードのシンプルさが強く求められる。そのた

め、アルゴリズムとデータ構造の実装が主な焦点となり、高度な Java の構文要素を使用する必要がほとんどない。例えば、問題の前提として入力形式やデータ範囲が厳密に保証されているため、Try-Catch 文や Throw 文によるエラー処理や Assert 文での入力検証は不要である。また、アノテーションやラムダ式、列挙型などの高度な構文要素は設計の拡張性を求められる場面で活用されるものであり、競技プログラミングのシンプルかつ短時間での解決を求める性質にはそぐわないものである。

さらに、ChatGPT はインターネット上で公開されている大規模なデータセットをもとに学習されており、このデータセットには競技プログラミング関連の情報 (Codeforces, AtCoder などのコードや解説) が含まれている可能性がある。競技プログラミングの解答コードは一般的に「シンプルかつ効率的である」「冗長な記述を避け、短時間で書ける」といった特徴をもつため、生成されたコードにもその傾向が反映されたと考えられる。

## 5. おわりに

本研究では、Java ソースコードを解析することでその理解に必要な習熟度を測定する枠組みである javacefr を提案した。javacefr は、プログラムの習熟レベルを、自然言語の習熟レベルの評価基準のひとつである CEFR に倣って 6 段階とし、プログラムに含まれる構文要素によってその理解に必要な習熟度を判定できる。また、与えられたプログラムに対して、プログラム中で用いられている構文要素とそれらの理解に必要な javacefr の習熟レベルを出力するシステムを開発した。本システムにより、特に、生成 AI によって生成されたプログラムについて、習熟が必要な構文とその習熟度が明らかとなることから、プログラム学習に役立つと期待される。

ケーススタディとして、ChatGPT に指示や問題を与えて生成させたプログラムを解析したところ、実用的なプログラムでは難しい (高い習熟度を要求する) 構文要素が検出されたのに対し、競技プログラミング用の問題を与えて生成させたプログラムでは難しい構文要素は含まれなかった。このことから、プログラミング学習においては、競技プログラミングの問題を解くのみならず、多様な構文要素が必要となるプログラミング課題に取り組むことが重要であるといえる。また、javacefr は実驗によって生成 AI から出力されるプログラムの難易度が推定できることが確かめられたため、プログラミング初

心者は生成 AI の出力が自分の理解能力に沿ったものかどうかを判断できる。

javacefr は、与えられたプログラムに対して必要な習熟度を判定できるが、プログラマ自身の習熟度を判定することも可能である。任意のプログラマが開発したプログラム群に対して javacefr を適用することで、そのプログラマがどの程度の難しさの構文要素を使用しているのか、また、未使用の構文要素にはどういったものがあり、それらはどの程度の習熟度が要求されるのか、といったことを明らかにでき、今後のプログラミング学習に役立つと期待される。このようなプログラマの習熟度の測定は、今後の重要な課題である。

## 6. 謝辞

本研究は JSPS 科研費 JP20H05706 の助成を受けた。

## 参考文献

- [1] Robles, G., Kula, R. G., Ragkhitwetsagul, C., Sakulniwat, T., Matsumoto, K., Gonzalez-Barahona, J. M.: pycefr: Python competency level through code analysis, *Proc. Int'l Conf. Program Comprehension (ICPC'22)*, pp. 173-177, 2022.
- [2] 山本道子, オラクル認定資格教科書 Java プログラマ Bronze, 翔泳社, 2020.
- [3] 山本道子, オラクル認定資格教科書 Java プログラマ Silver SE 11, 翔泳社, 2019.
- [4] 山本道子, オラクル認定資格教科書 Java プログラマ Gold SE 11, 翔泳社, 2021.
- [5] Brian North, The CEFR illustrative descriptor scales. *The Modern Language Journal* 91, 4, 656-659, 2007.
- [6] R. Minelli, A. Mocci, and M. Lanza, I know what you did last summer: an investigation of how developers spend their time, in *Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension, ICPC 2015, May 16-24, 2015*. IEEE Computer Society, pp. 25-35. [Online]. Available: <https://doi.org/10.1109/ICPC.2015.12>, 2015.
- [7] N. Smith, D. Van Bruggen, and F. Tomassetti, *Javaparser: visited*, Leanpub, oct. de, 2017.